UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

PROYECTO FIN DE CARRERA

Implementación de un sistema de comunicaciones basado en Software Radio

Ingeniería de Telecomunicación

Juan Pablo Montero Hidalgo

Enero 2014

Implementación de un sistema de comunicaciones basado en Software Radio

AUTOR: Juan Pablo Montero Hidalgo
TUTOR: Jorge A. Ruiz Cruz

Grupo de Radiofrecuencia: Circuitos, Antenas y Sistemas (RFCAS)

Dpto. de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Enero 2014

Resumen

En este trabajo, se propone la implementación de un sistema de comunicaciones basado en la emergente tecnología conocida como Software Radio haciendo uso de los periféricos disponibles en el laboratorio.

En primer lugar, se realiza un estudio de la tecnología para conocer cuál es el fundamento y sus posibles alcances en el ámbito de las telecomunicaciones. Conocer los fundamentos de la tecnología reside en indagar los principios de ésta, así como los componentes físicos que la conforman.

Una vez comprendido el objetivo de la tecnología e identificados los componentes físicos que la ponen en práctica, se realizará un estudio con la intención de mostrar las características que estos presentan.

Posteriormente, se realiza un estudio acerca de la herramienta software a utilizar; la cuál se utilizará a la hora de diseñar el sistema de comunicaciones.

Finalmente, con los conocimientos adquiridos en los pasos previos se dispondrá a realizar un sistema de comunicaciones basado en esta tecnología.

Palabras Clave

- Software Radio
- Software Defined Radio
- Universal Radio Peripheral (USRP)
- Placa principal (madre)
- Placa secundaria (hija)
- USRP N210
- RFX2400
- XCVR2450
- GNU Radio
- GNU Radio companion

Abstract

The proposal of this project is the implementation of a communication system based on the Software Radio technology using the available peripherals in the laboratory.

First of all, a research of the technology is carried out in order to provide an overview of its fundamentals and its scope relating the communications environment. The research of technology fundamentals includes not only exploring its principles, but also the components used.

Once the aim of the technology and components has been understood, these will be deeply studied in order to show their characteristics.

Afterwards, a research of the software tool is carried out. This tool will be used in order to design a complete communication system.

Finally, with all the knowledge acquired, a full communication system based on Software Defined Radio will be deployed.

Keywords

- Software Radio
- Software Defined Radio
- Universal Radio Peripheral (USRP)
- Motherboard
- Daughterboard
- USRP N210
- RFX2400
- XCVR2450
- GNU Radio
- GNU Radio companion

Agradecimientos

En primer lugar, quiero agradecer a mi tutor Jorge A. Ruiz por brindarme la posibilidad de realizar este proyecto. Gracias por aportarme ideas y facilitarme los recursos necesarios para la realización del proyecto.

También quería agradecer al resto de profesores del grupo RFCAS Bazil Taha, Jose Luis Masa y Juan Córcoles por la ayuda prestada, así como al resto de compañeros del grupo. Gracias por todo y ha sido un verdadero placer conoceros.

Quiero dar las gracias a mis padres y mis hermanos por todo el apoyo y toda la confianza que han depositado en mí, animándome tanto en los buenos momentos como en los más duros.

A lo largo de esta etapa, he hecho buenos amigos en la universidad con los que he tenido la suerte de conocerles y compartir esta travesía con ellos. Gracias Xarlos, Cecilia, Dieguete, Guille y Gonzalo por ser como sois.

Por último y no menos importante, también quería agradecer a todos mis amigos, los cuales hemos vivido muchas cosas juntos entre otras cosas largos días de estudio en la biblioteca.

Gracias a todos
Enero 2014

Índice general

Índice de figuras

Índice de ecuaciones

Índice de tablas

1 Introducción

1.1 Introducción general

La tecnología conocida como Software Radio (SR) consiste en acercar el procesamiento de señal via software lo más proximo posible a la antena, convirtiendo los problemas que hasta ahora habían recaido en hardware (moduladores, amplificadores, filtros…) en problemas tipo software. El foro del Software Defined Radio (recientemente renombrada a Wireless Innovation Forum), define intuitivamente dicha tecnología como: *"Radio in which some or all of the physical layer functions are software defined"* [1]. Una descripción mas detallada es dada por Joseph Mitola quien acuñó el término software radio como: *"A software radio is a radio whose channel modulation waveforms are defined in software. That is, waveforms are generated as sampled digital signals, converted from digital to analog via a wideband digitalto-analog converter (DAC) and then possibly upconverted from intermediate frequency (IF) to RF. The receiver, similarly, employs a wideband ADC that captures all of the channels of the software radio node. The receiver then extracts, downconverts and demodulates the channel waveform using software on a general purpose processor. Software radios employ a combination of techniques that include multi-band antennas and RF conversion; wideband ADC and DAC; and the implementation of IF, baseband and bitstream processing functions in general purpose programmable processors. The resulting software defined radio (or " software radio") in part extends the evolution of programmable hardware, increasing flexibility via increased programmability"* [2].

Esta evolución ha sido posible gracias a los grandes avances alcanzados con los *Digital Signal Processors* (DSP) y las *Field Programmable Gate Arrays* (FPGA).
Un sistema basado en SDR está formado por un ordenador (PC), o bien un sistema embebido, conversores analógicos digitales y módulos de radio frecuencia. La tecnología SDR se puede entender como la base de la Radio Cognitiva.

1.2 Motivación

La motivación de este proyecto final de carrera (PFC) se debe al gran auge de las nuevas tecnologías *Wireless* y a la gran evolución de los ordenadores de propósito general. La proliferación de estándares radio requiere terminales e infraestructuras diferentes, se propone la tecnología conocida como Software Radio como posible solución a estos problemas. La característica principal que presenta esta tecnología, es la reducción de la electrónica asociada a los trasmisores y receptores. Trasladando ciertas funcionalidades que antes recaían en el hardware a funciones software. Introducir más funcionalidades software permite añadir más flexibilidad al sistema, facilitando además la capacidad de una rápida reconfiguración del enlace y de sus parámetros tales como frecuencia, ancho de banda, modulaciones y potencia.

Maximizar la flexibilidad de un sistema de comunicaciones, conlleva numerosas ventajas tanto para el proveedor de servicios como para el usuario final, entre las que se podrían destacar [3]:

- El software es reutilizable, por lo que el coste de desarrollo se ve reducido drásticamente.
- Actualización de las infraestructuras existente minimizando los costes de despliegue.
- Unificar diferentes tecnologías en una misma plataforma radio multi-estándar reduce enormemente el coste logístico.
- Dispositivos reutilizables, al hacer uso de una misma plataforma radio, un mismo dispositivo puede ser utilizado en diferentes contextos.

La principal ventaja de una reconfiguración rápida es:

- Reconfigurar “over the air” el enlace, permite solucionar problemas surgidos en el sistema sin llegar a detener el servicio. Esto reduciría significativamente el tiempo y coste del mantenimiento.

1.3 Objetivos

Este proyecto tiene por finalidad dar una primera aproximación a la tecnología SR. Para ello, en primer lugar se llevará a cabo un estudio en profundidad del estado actual de dicha tecnología, seguido de un análisis detallado de los componentes (tanto software como hardware) típicamente usados en esta.

Como objetivo principal del PFC, se propone desarrollar un sistema de comunicaciones basado en SDR[1]. Para llevar a cabo la implementación de un sistema de comunicación basado en esta tecnología, será necesario un hardware específico y una plataforma software que permita la configuración tanto del hardware como del sistema. En el laboratorio de RFCAs[2] se disponía previamente de dos transceptores (RFX2400 y

[1] SDR: Software Defined Radio.
[2] RFCAs: Grupo Radiofrecuencia: Circuitos, Antenas y Sistemas de la Universidad Autónoma de Madrid (UAM)

XCVR2450[3]). Se propone como frecuencia de operación del sistema las bandas 2.3- 2.9 GHz y/o los 4.9-6 GHz.
Por último, con la intención de acercar la tecnología al usuario, se ha decidido desarrollar un script de tal forma, que el uso de la plataforma software quede en segundo plano y se permita realizar determinadas aplicaciones de manera intuitiva e interactiva con el usuario final. Dicho script, permitirá operar tanto en modo half-duplex[4] como full-duplex[5] según indique el usuario, además constará de las siguientes aplicaciones:

- Transmisor:
    - Dos tonos: El usuario podrá emitir dos tonos donde podrá manipular ciertos parámetros de forma gráfica.
    - Barrido de frecuencia: El usuario podrá realizar un barrido en frecuencia con un tono, para ello tendrá la posibilidad de modificar ciertos parámetros de forma gráfica.
- Receptor:
    - Analizador de espectro: El usuario podrá implementar un analizador de espectros donde podrá manipular de manera gráfica ciertos parámetros.

1.4 Posibles aplicaciones

En este apartado se mencionará algunos proyectos que se están desarrollando actualmente con tecnología SDR así como algunos proyectos ya realizados:

- OpenBTS:El proyecto OpenBTS es *open-source* y consiste en una aplicación Linux que utiliza SDR para combinar la interfaz GSM[6] del usuario (La interfaz entre la BTS[7] y el usuario móvil, Um[8]) con tecnologías VoIP[9] generando así una nueva red compatible con los usuarios tradicionales GSM.
- GNSS-SDR: El proyecto GNSS-SDR tiene por objetivo desarrollar en open-source un receptor GNSS[10] basado SDR.
- GPS-SDR: El proyecto GPS-SDR tiene por objetivo desarrollar un receptor GPS[11] basado en SDR
- Radar: Se puede utilizar la tecnología SDR para la implementación de un sistema radar.
- ATSC[12]: Se puede implementar un receptor de televisión digital mediante la tecnología SR.

[3] RFX2400 y XCVR2450: placas transceptoras del fabricante ETTUS LLC.
[4] Half-duplex: Modo de operación que permite la transmisión y recepción no simultánea.
[5] Full-duplex: Modo de operación que permite la transmisión y recepción simultánea.
[6] GSM: Global System for Mobile communications.
[7] BTS: Base Transceiver Station.
[8] Um: Interfaz GSM entre el usuario final (móvil) y la estación base.
[9] VoIP: Voice over IP.
[10] GNSS: Global Navigation Satellite Systems.
[11] GPS: Global Position System.
[12] ATSC: Estándar de televisión digital adoptado en países como Estados Unidos.

- Gr-ieee802-15-4: Está basado en el proyecto UCLA Zigbee e implementa un transceptor para IEE802.15.4[13] mediante SR.
- IEE802.11b: Se puede implementar un receptor para IEE802.11b[14] mediante SR.

1.5 Organización de la memoria

El PFC está estructurado en 8 capítulos y 6 apéndices

- Capítulo 1: Introducción, motivación, objetivo y posibles aplicaciones.
- Capítulo 2: Estado del arte: orígenes del SR, arquitectura en transmisores y receptores de la tecnología SDR, Cognitive Radio y modulaciones.
- Capítulo 3: Componentes hardware: Ordenador de propósito general y USRP
- Capítulo 4: Componentes software: UHD y GNU Radio.
- Capítulo 5: Pruebas iniciales: Transmisión y recepción.
- Capítulo 6: Experimentos y resultados en entorno de simulación:
- Capítulo 7: Experimentos y resultados con USRP:
- Capítulo 8: Conclusiones y trabajo futuro.
- Anexo A: Manual de instalación
- Anexo B: GNU Radio companion-Python-C++
- Anexo C: How to add a block
- Anexo D: Dial tone
- Anexo E: Preguntas frecuentes
- Anexo F: Código para la estimación de la BER

[13] IEE 802.15.4: Estándar para transmisión a baja tasa en las redes inalámbricas de área personal (LR-WPAN).
[14] IEE 802.11 b: Extensión del estándar para las redes inalámbricas de área local (WLAN).

2
Estado del arte

2.1 Introducción

Este capítulo, tiene por objetivo dar a conocer los orígenes de Software Radio, la arquitectura típica utilizada en los receptores y transmisores utilizados en esta tecnología, la evolución de Software Radio (Cognitive Radio) y por último, una breve descripción de las modulaciones fundamentales, más en concreto de las modulaciones digitales.

2.2 Orígenes del Software Radio.

El origen de la tecnología Software Radio está ligado al ámbito militar, en concreto al Departamento de Defensa de los Estados Unidos.
A finales de los años 70´s, la Fuerza Aérea Norteamericana trabajó en un sistema conocido como *Integrated Communications Navigation, Identification and Avionics system* (ICNIA). Éste sistema utilizaba un DSP en el que se realizaban las funciones de programación y control para obtener una plataforma integrada para comunicaciones aéreas. A este proyecto finalmente se le llamó SPEAKeasy [4].

En los primeros años de la década de los 90, surge el proyecto SPEAKeasy organizado por el Departamento de Defensa. Éste tenía como finalidad iniciar el desarrollo de un software programable para sistemas de radio que operara de los 2MHz a los 2GHz (HF[15], VHF[16], UHF[17],), pudiéndose considerar éste como la base de la tecnología SDR.

A mediados de la década de los 90, el departamento de defensa estadounidense crea el programa *Joint Tactical Radio Systems* [5]. El objetivo de éste, era desarrollar sistemas de radiocomunicaciones reconfigurables vía software para el ejército americano, reemplazando los diferentes sistemas radio que disponían y terminar así con el problema de la interoperabilidad entre los diferentes dispositivos.

[15] HF:La banda High Frecuency comprende el espectro de 3-30MHz.
[16] VHF:La banda Very High Frecuency comprende el espectro de 30-300MHz
[17] UHF:La banda Ultra High Frecuency comprende el espectro de 300-3000MHz

En 1996 Joseph Mitola funda el SDR *Forum* dando una descripción detallada de la tecnología. En la actualidad este foro se conoce como *Wireless Innovation Forum*, considerándose este como uno de los centros neurálgicos que acerca la tecnología Software Defined Radio y Cognitive Radio al mercado. En él, los miembros colaboran para acercar la experiencia obtenida. Este foro cuenta con miembros pertenecientes a diferentes tipos de organizaciones, ya sean organizaciones comerciales, de defensa o gubernamentales.
La siguiente imagen muestra organizaciones comerciales que emplean la tecnología SDR, alguna de ellas miembros de *Wireless Innovation Forum*:

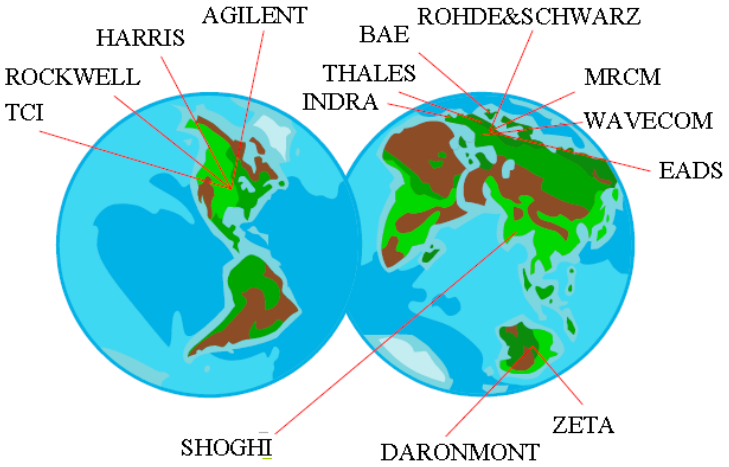


Fig 2.0.1 Organizaciones que utilizan SDR ([18])

2.3 Arquitectura de transmisores y receptores basados en Software Radio.

Este apartado, tiene por objetivo enfocar el concepto ya introducido de la tecnología SR para implementar transmisores y receptores, explicando que es lo que se desearía tener y por limitaciones existentes, lo que se dispone.

Como ya se había mencionado en el capítulo anterior, la idea de SR, proviene de trasladar funciones que anteriormente habían recaído en el hardware para poder realizarlas mediante software, acercando así el procesado software a la antena.

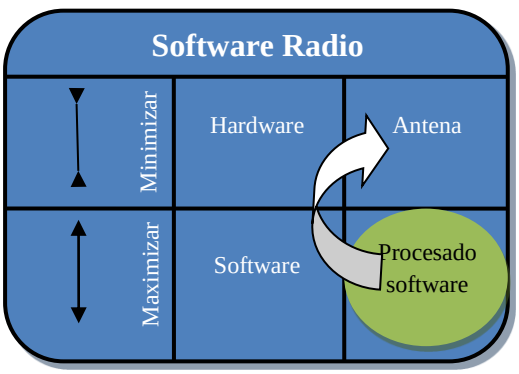


Fig 2.0.2: Principio de la tecnología SR

[18] **Fuente:** **ftp://gic.dsc.ulpgc.es/ETSIT/TS_Com_ETSIT/doc_docente/formato_simple/por_temas/sdrhf_t04.pdf**

El caso ideal, sería que toda la cadena tanto para el transmisor como para el receptor, fuese definida y configurada mediante software, dejando únicamente como componentes hardware del sistema a los conversores y las antenas tal y como muestra la siguiente imagen:

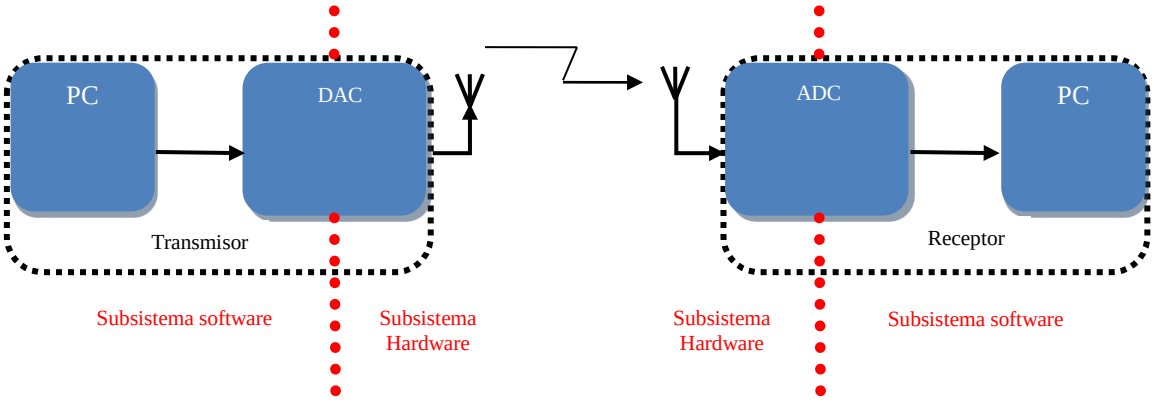


Fig 2.0.3: Tx y Rx SR ideales

Por lo tanto, según este esquema, se sustituiría todas las funciones que anteriormente recaían en la electrónica analógica (salvo los conversores y las antenas), realizándose ahora vía software. De esta forma, se solucionan problemas que surgen debido a comportamientos no ideales en los mezcladores, filtros y amplificadores. Minimizando así la distorsión introducida en la señal transmitida y recibida.

Sin embargo, analizando esta propuesta con cierto detalle, se puede llegar a demostrar que al menos por ahora no es realizable, además presenta algunas limitaciones:

- Conversores: Según la teoria de Nyquist, la frecuencia de muestreo ha de ser al menos dos veces la frecuencia máxima de la señal a muestrear para evitar el indeseado efecto del solapamiento espectral o aliasing, a pesar de que los conversores actuales soportan tasas de muestreo lo suficientemente altas para proporcionar ancho de banda para la mayoria de aplicaciones, el mandar una señal de RF directamente al conversor es inviable, debido a que la frecuencia de las portadoras típicamente es de GHz, lo que supondría disponer de un conversor que soportase anchos de banda de varios GHz (una señal trasmitida en la banda de 2 GHz necesitaria un DAC cuyo ancho de banda mínimo fuese de 4GHz), mientras que los conversores actuales soportan cientos de MHz.
  Otra limitación a tener en cuenta, viene dado en el lado del receptor. Las señales recibidas se caracterizan por sufrir una alta atenuación, esto supone un riesgo de recuperar mal la señal si esta no se cuantifica adecuadamente, este proceso como ya se ha mencionado dependerá del rango dinámico que presente el ADC, en general si los niveles de potencia son muy bajos en comparación con el rango dinámico diferentes muestras que correspondian a diferentes niveles inicialmente podrían ser cuantificados en uno mismo. También se ha de mencionar que un efecto similar puede ocurrir (perder la señal), si se reciben señales interferentes cuyos niveles de potencia sean significativamente superiores a los de la señal a recibir, provocando una mala cuantificación de la señal de interes. Por lo tanto, se necesitarán amplificadores y filtros en el lado del receptor para mitigar los efectos descritos.

- Tasas binarias: Se acaba de mencionar que existe cierta limitación en referencia a los anchos de banda que pueden llegar a soportar los conversores, sin embargo, también habrá que tener en cuenta la limitación que viene dada por la velocidad de transmisión de los buses de comunicación que suele ser típicamente de varios cientos de Mbps hasta algún Gbps, teniendo en cuenta que no todos los bits enviados a través del bus son de datos, esto afectará directamente a la tasa binaria máxima.

- Procesamiento: Otra limitación potencial reside en la dificultad que presenta el procesamiento en tiempo real de las aplicaciones en los PCs convencionales, tanto para la precisión temporal, como para el numero de operaciones soportadas por la computadora.

Las limitaciones presentadas, mas en concreto las dos primeras, hacen inviable la arquitectura presentada en la figura 2.2, haciendose indispensable el introducir una parte hardware (a la cuál se le llama RF-front end) que realice parte de las funciones detalladas anteriormente (filtrado en rf, amplificación, desplazos en frecuencia), a la tecnología SR implementable con la tecnología disponible se la denomina Software Defined Radio (SDR), aunque en la actualidad esta distiction ya no se emplea utilizando el termino SR o SDR indistintamente. Por lo tanto, el transmisor y receptor basado en Software Radio se distanciará del ideal, tal y como muestra la siguiente figura:

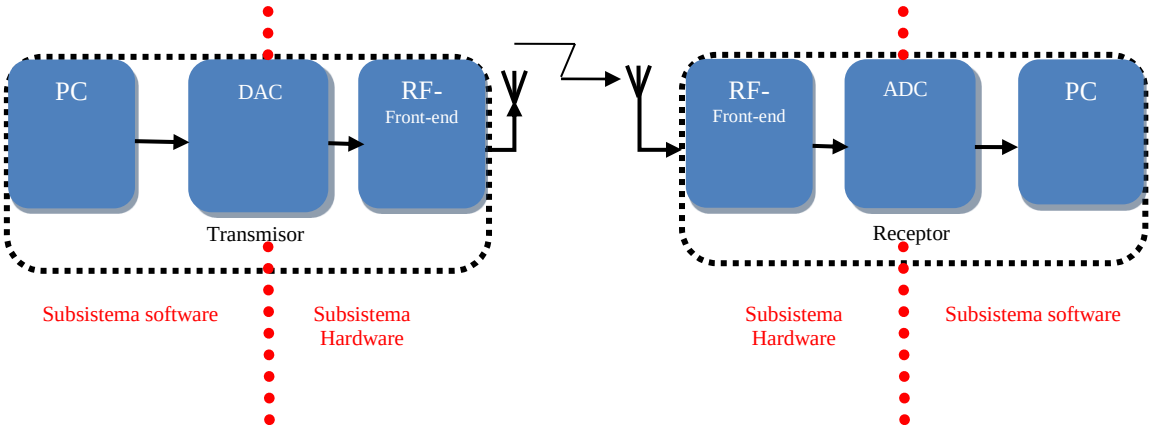


Fig 2.0.4: Tx y Rx SDR con módulos de RF

Al introducir componentes hardware en los transmisores y receptores (además de los conversores y las antenas) basados en la tecnología SDR, se buscan componentes que presenten un buen comportamiento en el mayor posible ancho de banda, evitando en la medida de lo posible comprometer el resto de parámetros que caracterizan a los filtros, los amplificadores y los mezcladores. De este modo, se busca mantener la flexibilidad de la frecuencia de operación que brinda la tecnología SDR.

Para minimizar el problema potencial que conlleva las operaciones que tiene que soportar la unidad de procesamiento, se propone introducir un elemento más, este puede tratarse de un DSP, una FPGA o un ASIC. Las operaciones de procesado que conlleven un alto coste computacional serán relalizadas por este componente, liberando así de carga computacional al ordenador.

Incluyendo este elemento en el transmisor y el receptor SDR, el esquema quedaría:

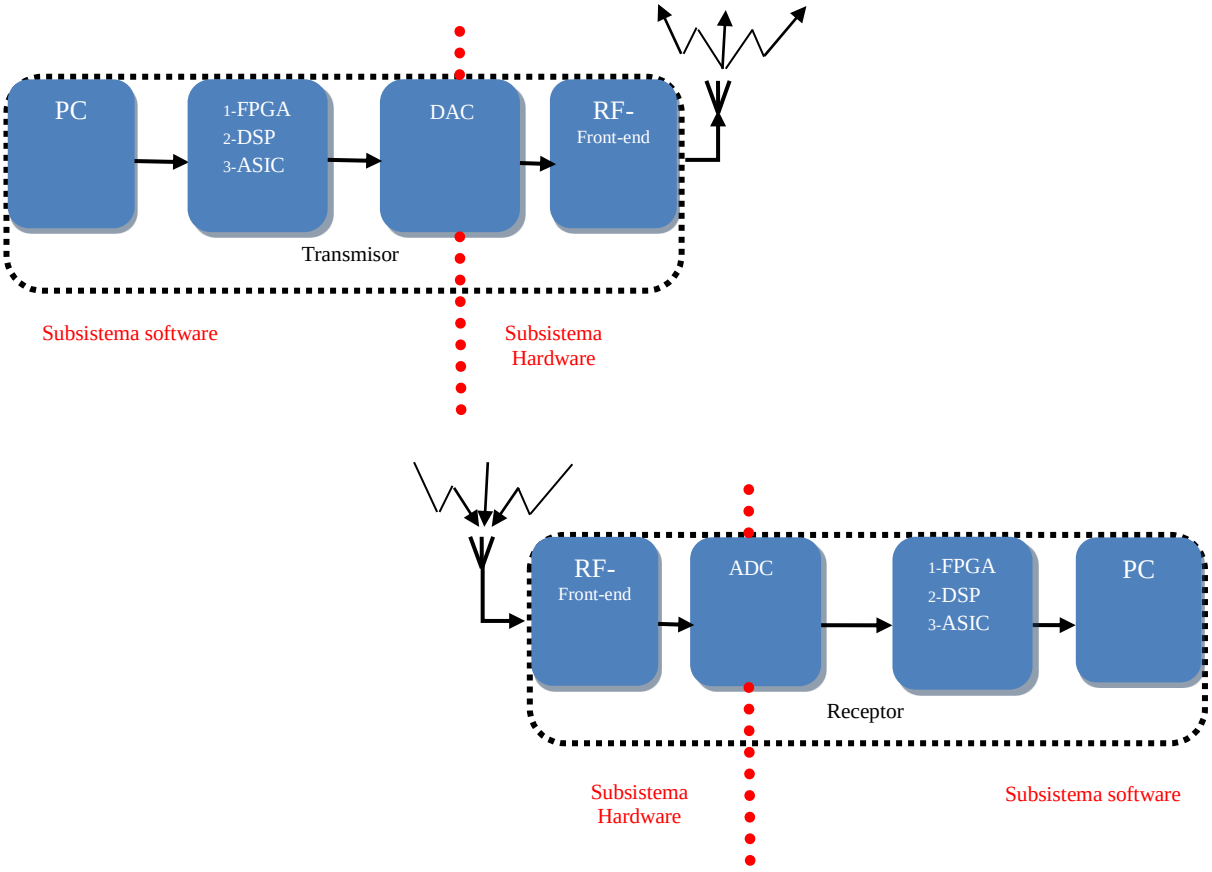


Fig 2.0.5: Tx y Rx SDR con FPGA

Cabe mencionar, que este último elemento no es estrictamente necesario aunque si muy recomendable para poder implementar sistemas que requieran alto coste computacional o minimizar la latencia en el sistema (muy importante para implementar enlaces en tiempo real).
La elección sobre este componente es un tema de discusión, sin embargo en lineas generales hay predilección por la FPGA debido a su alto grado de flexibilidad pese a que no sea la solución que presente el mayor *throughput*.

2.4 Cognitive Radio

La tecnología conocida como *Cognitive Radio* (CR) pretende ser un punto de ruptura en el manejo y compartición del espectro (brindando así una solución a la congestión de este) [6],[7]. Es la evolución natural propiciada por la tecnología SDR y propuesta por Joseph Mitola III:
"Cognitive Radio is a radio that employs model based reasoning to achieve a specified level of competence in radio-related domains" [8]. La Comisión Federal de Comunicaciones (FCC), anteriormente conocida como *National Telecommunications and Information Administration*, define la tecnología CR como: *"Cognitive radio: A radio or system that senses its operational electromagnetic environment and can dynamically and autonomously adjust its radio operating parameters to modify system operation, such as maximize throughput, mitigate interference, facilitate interoperability, access secondary markets."* [6],[9].

Se entiende entonces como una personalización del enlace radio, un sistema con un núcleo basado en SDR, que debe monitorear su entorno para que mediante su criterio y el conocimiento adquirido de parámetros tales como posición geográfica, estado de la red y espectro utilizado, sea capaz de indicar al transmisor y/o receptor basado en SDR que modifique la configuración del sistema (frecuencia de portadora, modulación utilizada, velocidad de transmisión y potencia) y de operar en distintas redes de comunicación.
La siguiente imagen muestra la arquitectura simplificada [10] que se tendría en esta tecnología:

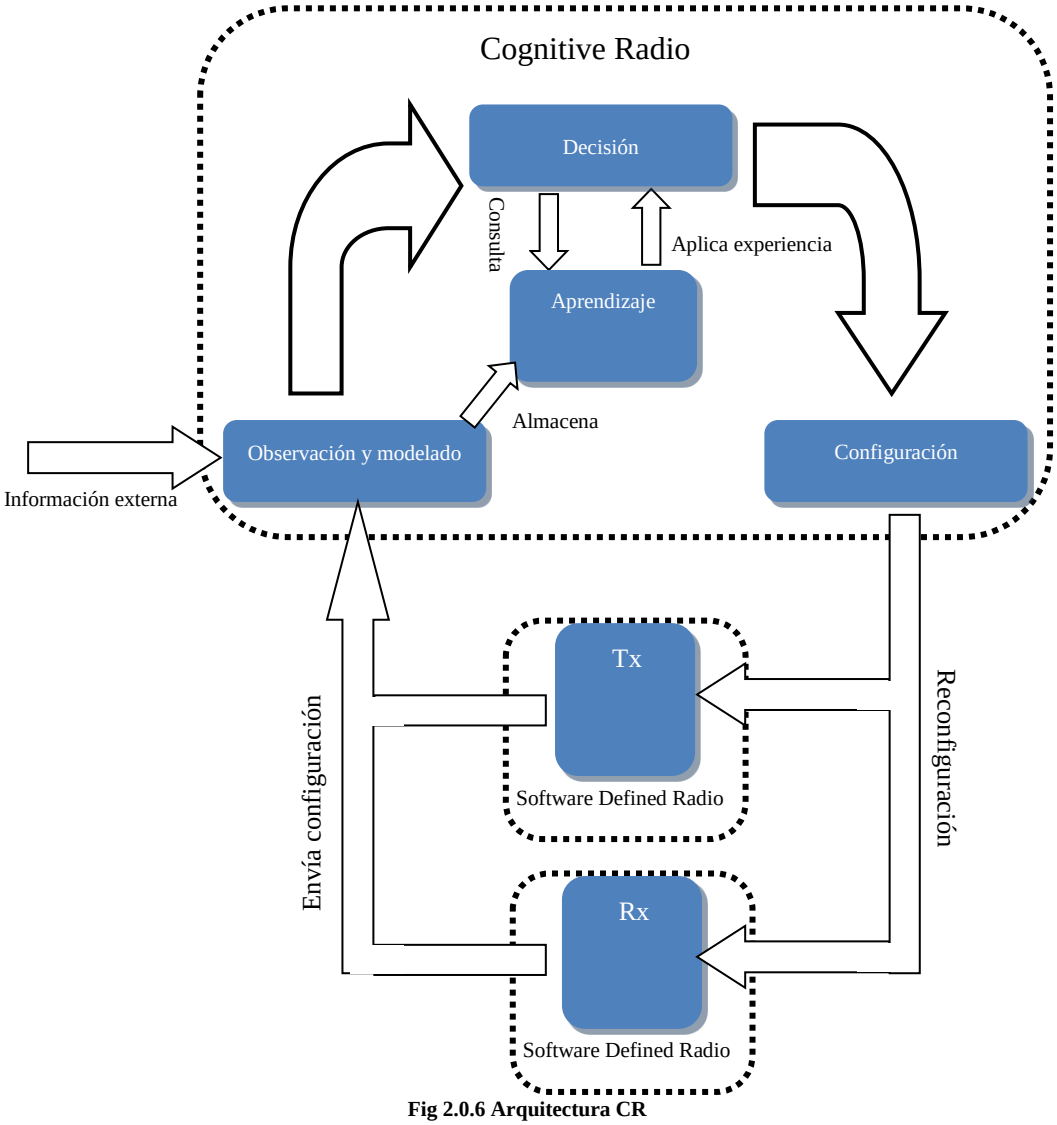


Fig 2.0.6 Arquitectura CR

2.5 Modulaciones

La modulación de una señal consiste en modificar algún parámetro de la señal portadora (amplitud, fase o frecuencia), de acuerdo a las características de la señal de información o moduladora, obteniendo así una señal resultante conocida como señal modulada [11], [12], [13]. Típicamente, la señal portadora es un seno o un coseno de frecuencia muy superior con respecto a la señal de información.

Dependiendo de la naturaleza de la señal de información y de la portadora, se pueden distinguir varios tipos de modulaciones como muestra la siguiente tabla:

	Señal analógica	Señal digital
Portadora analógica	Modulación analógica	Modulación digital
Portadora digital	Modulación por pulsos	Códigos de línea

Tabla 2.0.1: Tipos de modulaciones monoportadoras

Este capítulo, se centrará en el estudio de las modulaciones digitales debido a que serán las que se utilizarán para el sistema de comunicación que se desea implementar, mostrando el resto con una breve introducción.

2.5.1 Modulaciones analógicas

La principal característica de este tipo de modulación reside en que tanto la moduladora como la portadora son analógicas.

Sea x(t) la señal a transmitir (señal de potencia) en banda base y de ancho de banda B, donde X(f) = 0 |f| > B (asumiendo que B es el espectro positivo de la señal) y c(t) la portadora caracterizada como $c(t)= A_c \cdot \cos(2 \cdot \pi \cdot f_c \cdot t + \varphi_c)$, donde A_c, f_c y φ_c son la amplitud, frecuencia y la desviación de fase de la portadora respectivamente. Esta notación será la utilizada en los apartados posteriores, llamando y(t) a la señal resultante (señal modulada).

Dependiendo de cuál sea el parámetro de la señal portadora que se module por la señal de información, se pueden distinguir distintos tipos de modulaciones analógicas:

2.5.1.1 Modulación de amplitud

La modulación de amplitud es un tipo de modulación lineal cuyo principio de funcionamiento es hacer que la amplitud de la señal portadora dependa de la señal moduladora, a este tipo de modulaciones se las conoce como modulaciones de envolvente variable. Típicamente la expresión de la señal modulada será:

$$y(t) = A_c x(t) cos(2\pi f_c t + \varphi_c)$$

Ec 2.0.1: Expresión señal modulada en amplitud

Aunque esta expresión variará dependiendo de los diferentes tipos de modulaciones. En la literatura se habla de las siguientes modulaciones analógicas de amplitud:

- **Doble banda lateral (DBL):** La modulación en doble banda lateral se obtiene al multiplicar la señal portadora y la señal moduladora, la expresión de la señal modulada es:

$$y(t) = A_c x(t) cos(2\pi f_c t + \varphi_c)$$
Ec 2.0.2: Expresión señal DBL

- **Amplitud Modulada (AM):** La modulación AM consiste en modular la señal del mismo modo que se realizaba en DBL, pero añadiendo a la señal resultante la portadora para simplificar el esquema del receptor , la expresión de la señal modulada es:

$$y(t) = A_c(1 + a x_N(t)) cos(2\pi f_c t + \varphi_c), \quad donde\ x_N = \frac{x(t)}{x_{max}}$$
Ec 2.0.3: Expresión señal AM

Donde a es el parámetro conocido como índice de modulación.

- **Banda Lateral Única (BLU):** La modulación BLU surge de la idea que al trasladar en frecuencia la señal de banda base, la información está duplicada. Por lo tanto se podrá realizar un filtrado paso banda que elimine una de las bandas, o bien la superior (BLS) o la inferior (BLI), por lo tanto la expresión de la señal modulada será la señal DBL filtrada.

- **Banda Lateral Vestigial (BLV):** Es una modificación de la modulación BLU que relaja las condiciones del filtro de banda base (no tan abrupto) para que sea realizable, dejando pasar un vestigio de la señal de la banda eliminada, por lo tanto la expresión de la señal modulada será la señal DBL filtrada usando un filtro menos abrupto que el utilizado en BLU.

2.5.1.2 Modulación angular

La modulación angular es un tipo de modulación no lineal cuya idea principal es que la señal de información se encuentre en la fase de la portadora, por lo tanto la amplitud de la señal modulada permanece constante, a este tipo de modulaciones se las conoce como modulaciones de envolvente constante. Típicamente la expresión de la señal modulada será:

$$y(t) = A_c cos(\omega_c t + \varphi(t)) = A_c cos(\theta(t))$$
Ec 2.0.4: Expresión señal modulación angular

Dentro de las modulaciones angulares, se puede distinguir:

- **Modulación de fase (PM):** La modulación de fase queda caracterizada por el parámetro β (ó K_p), el cual representa la máxima desviación en fase. Típicamente la expresión de la señal modulada será:

$$y(t) = A_c cos(\omega_c t + \varphi(t)) = A_c cos(\omega_c t + \beta x_N(t)), \qquad donde\ \beta = K_p x_{max}$$

Ec 2.0.5: Expresión señal PM

  Donde $x_n(t)$ es el valor normalizado de la señal.

- **Modulación de frecuencia (FM):** La modulación de fase queda caracterizada por el parámetro f_D (ó K_F), el cual representa la máxima desviación en frecuencia. Típicamente la expresión de la señal modulada será:

$$y(t) = A_c cos(\omega_c t + \varphi(t)) = A_c cos\left(\omega_c t + 2\pi f_D \int_{-\infty}^{t} x_N(\tau)\, d\tau\right), \qquad donde\ f_D = K_F x_{max}$$

Ec 2.0.6: Expresión señal FM

  Donde $x_n(t)$ es el valor normalizado de la señal.

2.5.2 Modulaciones digitales

La principal característica de este tipo de modulación reside en que a diferencia de las modulaciones conocidas como analógicas, la fuente de información ahora es digital.
La señal de información será por lo tanto un flujo de bits. Dependiendo del número de bits que se agrupen se formará una modulación de M niveles, donde $M=2^k$ y k los bits agrupados. Por lo tanto, las señales moduladas, estarán contenidas en un conjunto finito conocido como el alfabeto de señales o símbolos.

Un concepto que cobra gran importancia en este tipo de modulaciones es el concepto de constelación, la constelación es la representación del alfabeto de señales en un sistema de L dimensiones. Esto se debe a que el conjunto de señales forman un espacio vectorial, por ende se podrá buscar una base ortonormal que represente dicho espacio[19]

En este tipo de modulaciones se habla de modulaciones en banda base y modulaciones paso banda.

2.5.2.1 Modulaciones Banda Base

La modulación digital en banda base se conoce como *Pulse Amplitude Modulation* (PAM), esta modulación consiste en asignar a cada símbolo un pulso arbitrario de período el del símbolo y con amplitudes diferentes (M-PAM).

[19] Véase tema IV.3 Análisis en el espacio de señales: TCO 2007-08

La siguiente imagen muestra una modulación 4-PAM:

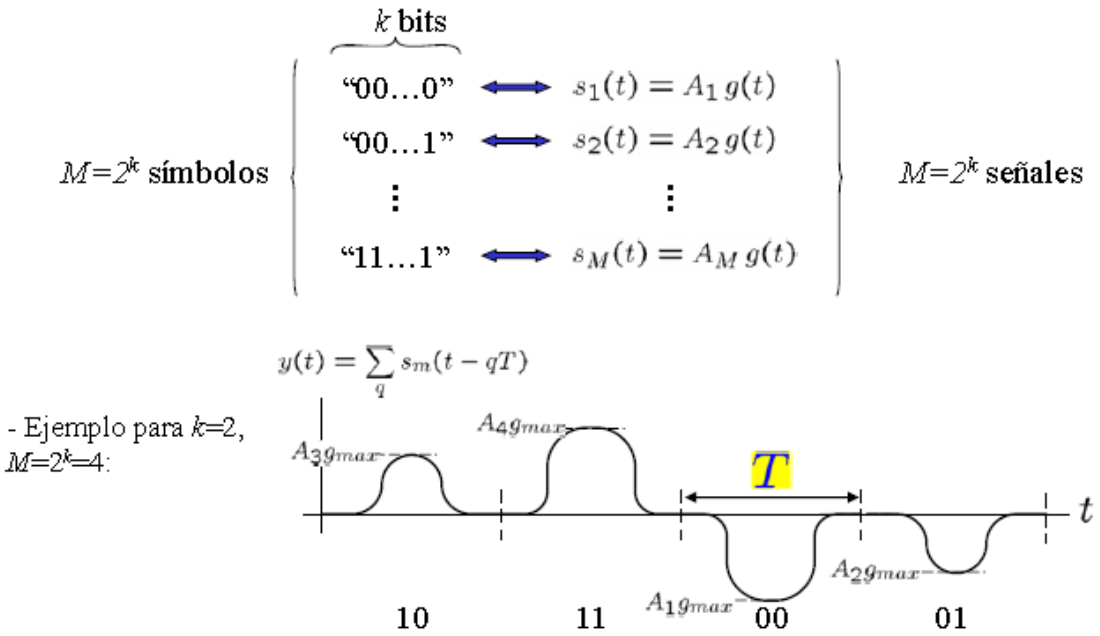


Fig 2.0.7 Modulación 4-PAM ([20])

Por lo tanto una señal M-PAM se podrá expresar como:

$$s_m(t) = A_m g(t), \qquad \text{m} = 1,2,\ldots \text{M}, \qquad 0 \leq \text{t} \leq \text{T}$$

Ec 2.0.7: Expresión señal M-PAM

Donde g(t) es un pulso de periodo el del símbolo y de forma arbitraria, esta expresión puede escribirse con respecto a la base $\overline{\psi}$(t):

$$s_m(t) = s_m \bar{\psi}(t), \qquad \Rightarrow \overline{s_m} \equiv (s_m)$$

Ec 2.0.8: Expresión señal M-PAM con respecto a la base ψ

Definiendo la base $\overline{\psi}$(t) como:

$$\bar{\psi}(t) = \frac{1}{\sqrt{\varepsilon_g}} g(t)$$

Ec 2.0.9: Expresión señal de la base ψ

Donde ε_g es la energía del pulso g(t) y se calcula como:

$$\varepsilon_g(t) = \int_{-\infty}^{\infty} |g(t)|^2 \, dt$$

Ec 2.0.10: Energía del pulso g(t)

El coeficiente s_m representa la coordenada de la señal con respecto a la base, este coeficiente se expresa como:

$$s_m = \sqrt{\varepsilon_g} A_m$$

Ec 2.0.11: Coordenada de la señal con respecto a la base

[20] Fuente: TCO IV.2.5 Sistemas PAM (2007-08)

Se ha de mencionar que sólo hará falta un vector para describir la base del subespacio, por lo tanto este es unidimensional. La representación geométrica de las señales (constelación).será a lo largo de un eje, el que forma la propia base. Dependiendo de si las amplitudes son simétricas o no se obtiene:

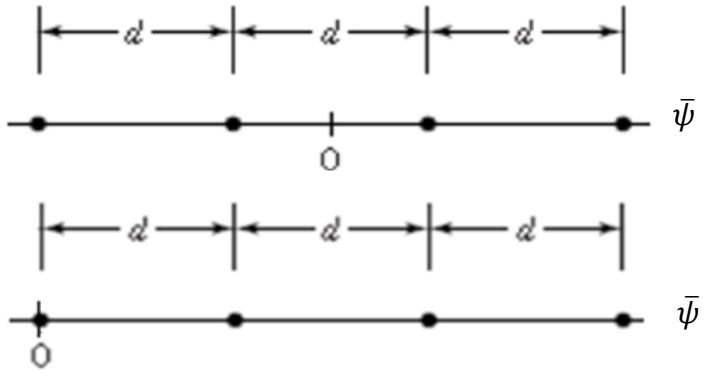


Fig 2.0.8: Constelación 4-PAM simétrica (superior) y asimétrica (inferior)

2.5.2.2 Modulaciones paso banda

Con respecto a las modulaciones en paso banda, ocurre como en las modulaciones analógicas, también se encuentran modulaciones de amplitud, fase y frecuencia. Se obtiene así una clasificación típica de las diferentes modulaciones digitales:

2.5.2.2.1 Modulación de amplitud por desplazamiento (ASK)

La modulación ASK se obtiene al desplazar en frecuencia una modulación PAM, se puede entender por lo tanto como una modulación DBL sobre la modulación PAM. La señal modulada se obtendrá al multiplicar la señal modulada PAM con una portadora.
Por lo tanto una señal modulada mediante ASK podrá expresarse como:

$$s_m(t) = A_m g(t)\cos(w_c t), \qquad \text{m} = 1,2,\ldots \text{M}, \qquad 0 \le \text{t} \le \text{T}$$

Ec 2.0.12: Expresión señal ASK

Donde g(t) es un pulso de periodo el del símbolo y de forma arbitraria (su frecuencia muy inferior al de la portadora), la señal modulada puede escribirse como:

$$s_m(t) = s_m\bar{\psi}(t) \quad \Rightarrow \overline{s_m} \equiv (s_m)$$

Ec 2.0.13: Expresión señal ASK con respecto a la base ψ

Definiendo la base ψ(t) como (la dimensión del subespacio vectorial es 1):

$$\bar{\psi}(t) = \sqrt{\frac{2}{\varepsilon_g}}\, g(t)\cos(w_c t)$$

Ec 2.0.14: Base del subespacio vectorial ASK

El coeficiente s_m representa la coordenada de la señal con respecto a la base, este coeficiente se expresa como:

$$s_m = \sqrt{\frac{\varepsilon_g}{2}} A_m$$

Ec 2.0.15: Coordenada con respecto a la base ASK

Las amplitudes (A_m) de una modulación ASK se suelen tomar como en una PAM asimétrica, es decir, empezando por el 0 y separadas uniformemente.

La representación geométrica de las señales (constelación) será a lo largo de un eje (como ocurría en la modulación PAM), el que forma la base, obteniéndose así la siguiente constelación:

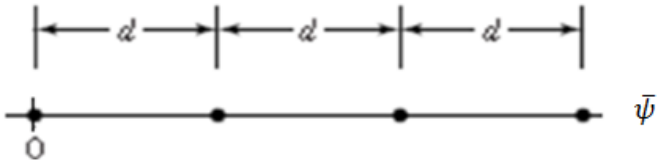


Fig 2.0.9: Constelación 4ASK

Una modulación ASK binaria (BASK) con un valor de amplitud 0, también recibe el nombre de modulación On-Off Keying (OOK).

2.5.2.2.2 Modulación de fase por desplazamiento (PSK)

Este tipo de modulación es la versión digital de la modulación PM, se puede entender por lo tanto como una modulación PM sobre la modulación M-PAM.
Por lo tanto una señal modulada mediante PSK podrá expresarse como:

$$s_m(t) = Ag(t)\cos(w_c t + \theta_m),\ \ \theta_m = \theta_0 + \frac{2\pi}{M}(m-1), \qquad 0 \le t \le T \ \ m = 1,2,\dots M$$

Ec 2.0.16: Expresión señal PSK

Donde Θ_o es la fase inicial arbitraria y g(t) es un pulso de periodo el del símbolo y de forma arbitraria (de frecuencia muy inferior al de la portadora).
Desarrollando la expresión anterior, se puede reescribir como:

$$s_m(t) = Acos(\theta_m)g(t)\cos(w_c t) - Asen(\theta_m)g(t)sen(w_c t)$$[21]

Ec 2.0.17: Expresión señal PSK en componentes IQ

Donde las componentes $A_{Im} = Acos(\theta_m)$ y $A_{Qm} = Asen(\theta_m)$ son conocidas como las componentes en fase y cuadratura respectivamente.

En este caso, el subespacio tiene dos dimensiones. Se necesitará dos vectores ortonormales para describirla, pudiéndose expresar la señal en función de éstas:

$$s_m(t) = s_{m_I}\overline{\psi_I}(t) + s_{m_Q}\overline{\psi_Q}(t) \Rightarrow \overline{s_m} \equiv (s_{m_I}, s_{m_Q})$$

Ec 2.0.18: Expresión señal PSK con respecto a la base ψ_I, ψ_Q

Definiendo las bases $\psi_I(t)$ y $\psi_Q(t)$:

$$\overline{\psi_I}(t) = \sqrt{\frac{2}{\varepsilon_g}} g(t)\cos(w_c t)$$

Ec 2.0.19: Componente ψ_I de la base del subespacio vectorial PSK

[21] cos (a + b) = cos(a)cos(b) – sen(a)sen(b).

$$\overline{\psi_Q}(t) = -\sqrt{\frac{2}{\varepsilon_g}}g(t)\text{sen}(w_c t)$$

Ec 2.0.20: Componente ψ_Q de la base del subespacio vectorial PSK

Los coeficientes s_{mI} y s_{mQ} representan la coordenada de la señal con respecto a la base, estos coeficientes se expresan como:

$$s_{m_I} = \sqrt{\frac{\varepsilon_g}{2}}A\cos(\theta_m)$$

Ec 2.0.21: Coeficiente s_{mI} con respecto a la base PSK

$$s_{m_Q} = \sqrt{\frac{\varepsilon_g}{2}}A\,\text{sen}(\theta_m)$$

Ec 2.0.22: Coeficiente s_{mQ} con respecto a la base PSK

La representación geométrica de las señales (constelación) será a lo largo de una circunferencia de radio la raíz de la energía de la señal, obteniéndose así la siguiente constelación:

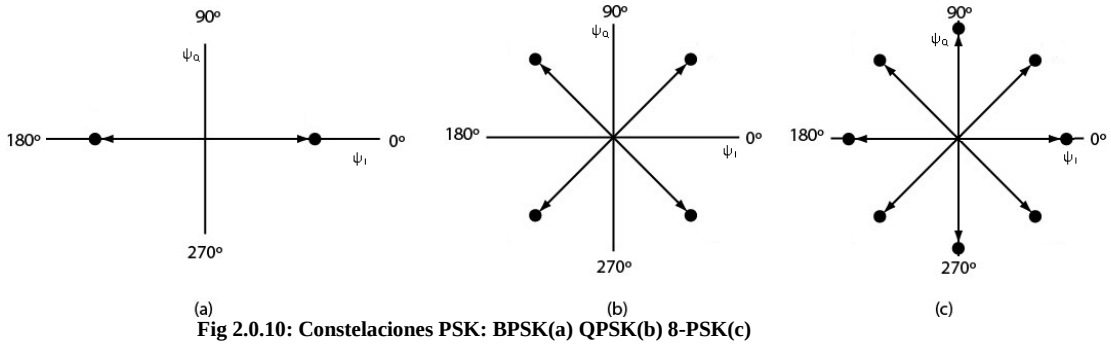


Fig 2.0.10: Constelaciones PSK: BPSK(a) QPSK(b) 8-PSK(c)

2.5.2.2.3 Modulación de frecuencia por desplazamiento (FSK)

Este tipo de modulación es la versión digital de la modulación FM, se puede entender como una modulación FM sobre la modulación M-PAM.
Por lo tanto una señal modulada mediante FSK se podrá expresar como:

$$s_m(t) = Ag(t)\cos(2\pi f_m t),\ \ \text{f}_\text{m} = \text{f}_\text{c} + \frac{1}{2}(2\text{m} - \text{M} - 1)\Delta_\text{f},\ \ \text{m} = 1,2,\ldots\text{M}$$

Ec 2.0.23: Expresión señal FSK

Donde f_c es la frecuencia de la portadora, Δ_f es la separación de frecuencias entre símbolos consecutivos (esta separación es tomada para que las señales sean ortogonales) y g(t) es un pulso de periodo el del símbolo y de forma arbitraria.

Al elegir el parámetro Δ_f para que las señales sean ortogonales entre sí, se necesitarán tantos vectores (que definan la base del subespacio) como señales tenga el alfabeto, es decir el número de dimensiones del espacio vectorial (L) coincidirá con el número de posibles símbolos que se tenga (M).

Como ocurría en FM, todas las señales generadas tendrán la misma energía (envolvente constante), expresando la señal modulada la señal en función de la base se obtiene:

$$s_m(t) = s_{mm}\overline{\psi_m}(t) \Rightarrow \overline{s_m} \equiv (0,\ldots,0,\overbrace{\widetilde{s_{mm}}}^{pos\ m},0,\ldots,\overbrace{\widetilde{0}}^{pos\ L=M})$$

Ec 2.0.24: Coordenada s_m con respecto a la baseFSK

Definiendo las componentes de la base ψ(t) como:

$$\overline{\psi_i}(t) = \sqrt{\frac{2}{\varepsilon_g}} g(t)\cos(w_i t), i = 1, \dots, L(= M)$$

Ec 2.0.25: Componente ψi de la base del subespacio vectorial FSK

Los coeficientes s_{mi} representan la coordenada de la señal con respecto a la base, estos coeficientes se expresan como:

$$s_{mm} = \sqrt{\frac{A^2 T}{2}})$$

Ec 2.0.26: Coeficiente smm con respecto a la base FSK

$$s_{mi} = 0, m \neq i, \qquad i = 1, \dots m, \dots L$$

Ec 0.27: Coeficiente smi con respecto a la base FSK

La representación geométrica de las señales (constelación) será por lo tanto para el caso L=2 (BFSK):

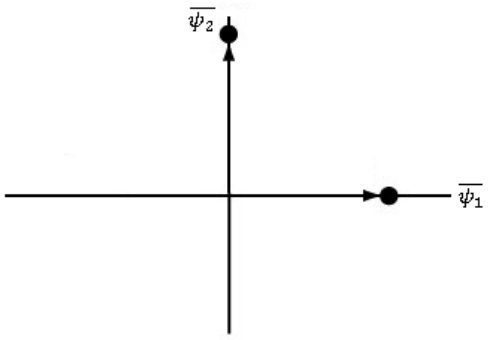


Fig 2.0.11: Diagrama de constelación BFSK

Cuando se toma como parámetro $\Delta_f = \frac{1}{4T}$, esta modulación se conoce como MSK[22]

2.5.2.2.4 Modulación de amplitud en cuadratura (QAM)

Este tipo de modulación es una modulación digital híbrida en el que el mensaje está contenido tanto en la amplitud como en la fase de la señal transmitida, se puede entender como dos señales N-PAM (simétricas) que modulan a una portadora en fase y a otra en cuadratura, obteniéndose así una modulación M-QAM (el número de niveles ha de ser potencia de 2). Por lo tanto una señal modulada QAM se podrá expresar como:

$$s_m(t) = A_{m_I} g(t)\cos(w_c t) - A_{m_Q} g(t) sen(w_c t), \qquad \text{m} = 1,2, \dots \text{M},$$
$$0 \leq \text{t} \leq \text{T}, \; A_{m_I}, A_{m_Q} \in \{A(2i - 1 - N), i = 1, \dots, N, N = \sqrt{M}\}$$

Ec 2.0.28: Expresión señal QAM

El espacio vectorial tiene dos dimensiones en este caso, por lo que hará falta dos vectores ortonormales que formen la base, pudiéndose expresar la señal en función de estas:

[22] MSK: Minimum Shift Keying.

$$s_m(t) = s_{m_I}\overline{\psi_I}(t) + s_{m_Q}\overline{\psi_Q}(t) \Rightarrow \overline{s_m} \equiv (s_{m_I}, s_{m_Q})$$

Ec 2.0.29: Expresión señal QAM en componentes IQ

Definiendo las bases $\psi_I(t)$ y $\psi_Q(t)$:

$$\overline{\psi_I}(t) = \sqrt{\frac{2}{\varepsilon_g}}g(t)\cos(w_c t)$$

Ec 2.0.30: Componente ψI de la base del subespacio vectorial QAM

$$\overline{\psi_Q}(t) = -\sqrt{\frac{2}{\varepsilon_g}}g(t)\text{sen}(w_c t)$$

Ec 2.0.31: Componente ψQ de la base del subespacio vectorial QAM

Los coeficientes s_{mI} y s_{mQ} representan la coordenada de la señal con respecto a la base, estos coeficientes se expresan como:

$$s_{m_I} = \sqrt{\frac{\varepsilon_g}{2}}A_{I_m}$$

Ec 2.0.32: Coeficiente sm_I con respecto a la base QAM

$$s_{m_Q} = \sqrt{\frac{\varepsilon_g}{2}}A_{Q_m}$$

Ec 2.0.33: Coeficiente sm_Q con respecto a la base QAM

La representación geométrica de las señales (constelación) será:

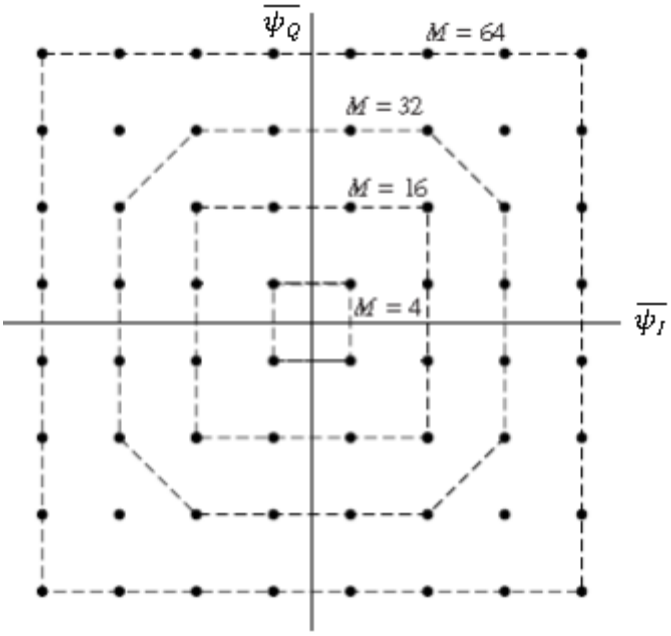


Fig 2.0.12: Constelación M-QAM ([23])

[23] Fuente: Communication Systems Engineering (2nd Edition) [John G. Proakis, Masoud Salehi]. Section 7.3, Pag 359.

2.5.3 Modulaciones por pulsos

La principal característica de este tipo de modulación reside en que la señal de información es analógica mientras que la portadora es digital del tipo tren de pulsos.
En este tipo de modulaciones se pueden distinguir típicamente los siguientes tipos:

- **Modulación por amplitud de pulsos (PAM):** Este tipo de modulación es la consecuencia inmediata del muestreo de una señal analógica. Si una señal analógica se muestrea a intervalos regulares, en lugar de tener señal continua, se tendrá una señal discreta en tiempo.

- **Modulación por duración de pulsos (PDM):** Este tipo e modulaciones también se conoce como PWM[24], en este caso se empleará la muestra de la señal para designar la longitud o duración del pulso, en este tipo de modulaciones la amplitud de los pulsos permanece constante.

- **Modulación por posición de pulsos (PPM):** En este tipo de modulaciones, tanto la amplitud como la duración de los pulsos permanece constante, la muestra de la señal determina el desplazamiento que se aplica al pulso (se varía el instante en el que se transmiten).

- **Modulación por frecuencia de pulsos (PFM):** En este tipo de modulaciones, tanto la amplitud de los pulsos como la duración de ellos permanece constante siendo el parámetro a variar la frecuencia de ellos.

- **Modulación por codificación de pulsos (PCM):** Este tipo de modulación es la consecuencia inmediata de la digitalizar la señal analógica. En primer lugar se muestrea la señal consiguiendo una señal PAM, discreta en el tiempo pero infinitos valores de amplitud, para posteriormente cuantificar los niveles de amplitud asignándoles a cada uno un código binario.

[24] PWM: Pulse Width Modulation

La siguiente imagen muestra de manera gráfica los procesos llevados a cabos en las modulaciones anteriormente descritas:

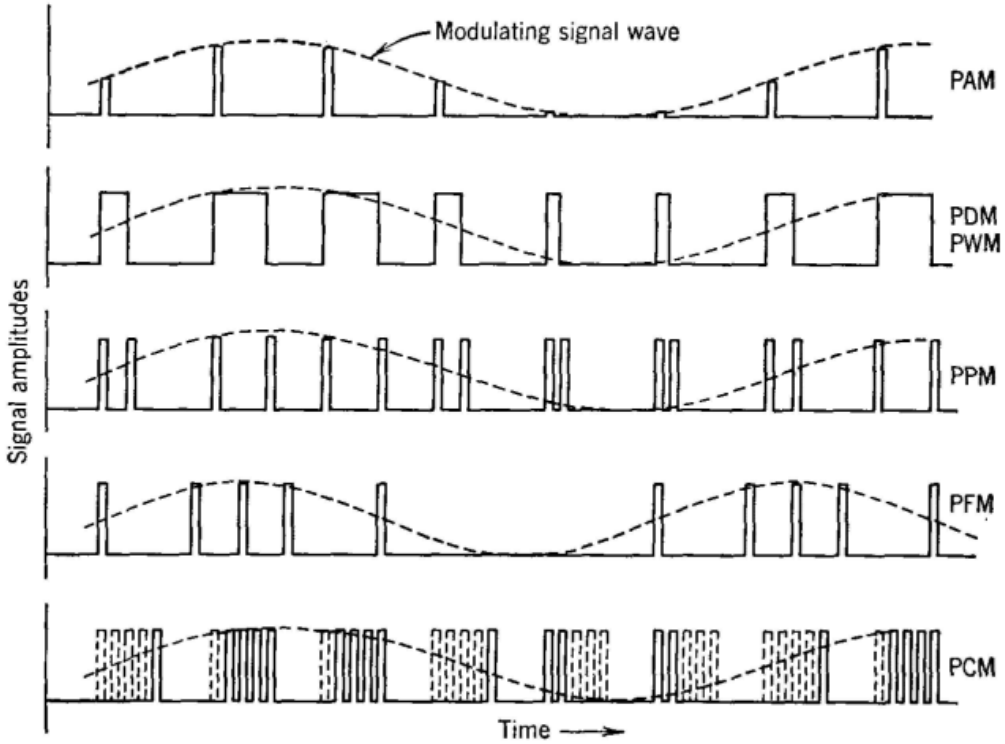


Fig 2.0.13: Modulaciones por pulsos ([25])

2.5.4 Códigos de línea

La principal característica de este tipo de modulación reside en que tanto la señal de información como la portadora son digitales. En este tipo de modulaciones de banda base se pueden distinguir típicamente los siguientes tipos:

- **Non Returning Zero (NRZ):** La amplitud de la señal permanece constante todo el periodo de bit, al "1" lógico se le asigna una amplitud +V y al "0" una amplitud –V.

- **Returning Zero (RZ):** La amplitud de la señal no permanece siempre constante, si no que vuelve a "0" tras indicar el nivel de bit.

- **Alternate Mark Inversion (AMI):** La amplitud de la señal permanece constante todo el periodo de bit, los "1" lógicos producen valores alternos en la amplitud de la señal.

- **Manchester:** En este código siempre hay una transición en la mitad del intervalo de duración de los bits. Dependiendo de si se transmite un "1" o un "0" dicha transición se llevará cabo en la primera mitad del periodo de bit o en la

[25] Fuente: eBook "Electrical Communication" is based on the printed copy of the book "Electrical Communication" by A.L. Albert (http://www.vias.org/albert_ecomm/)

segunda, el instante de la transición puede variar según sea el tipo de modulación Manchester utilizada.

En estos tipos de modulaciones existen las versiones diferenciales (NRZ-diferencial, RZ-diferencial, Manchester-diferencial), en donde un "1" lógico produce un cambio de estado, así como versiones unipolares y bipolares.
La siguiente imagen muestra de manera gráfica los procesos llevados a cabos en las modulaciones anteriormente descritas:

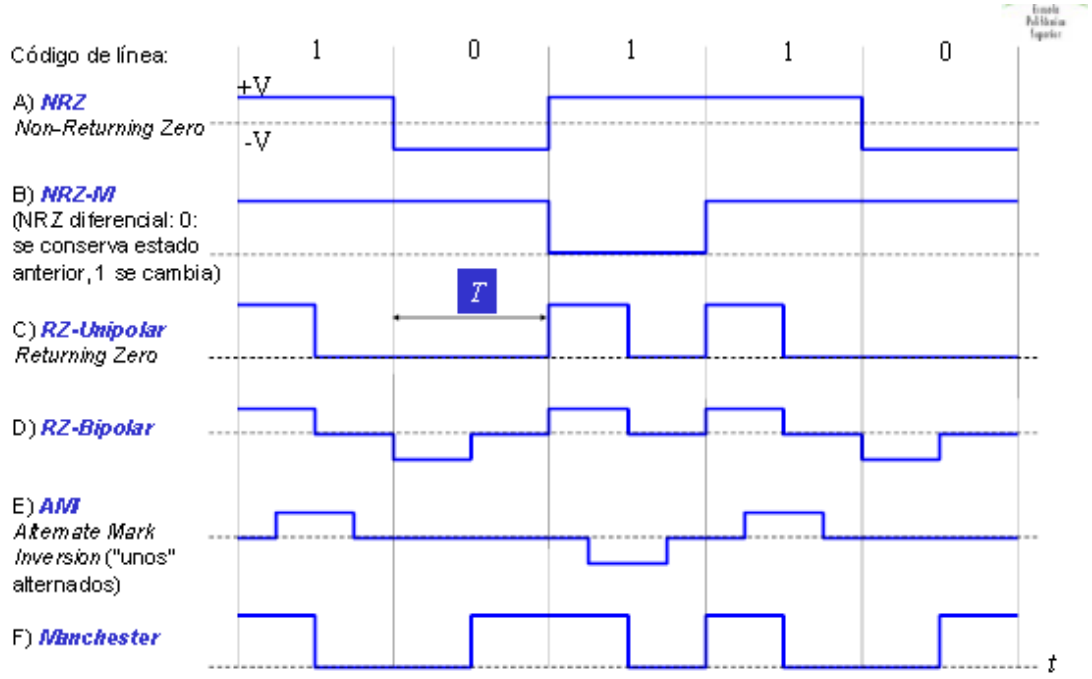


Fig 2.0.14: Moduladores digitales binarios ([26])

Se ha presentado diferentes tipos de modulaciones monoportadoras, otra clasificación posible hubiese sido presentarlas como modulaciones con memoria y sin memoria. Se dice que una modulación tiene memoria cuando la salida actual depende de un valor anterior de la señal, un ejemplo de este tipo serían las modulaciones diferenciales.

2.5.5 Modulaciones en SDR

Conocer las expresiones matemáticas que describen las señales de información moduladas y su representación (constelación) ayuda a la hora de comprender como es llevada la mayoría de las modulaciones digitales en la tecnología SDR. Típicamente los bits generados son agrupados (en función del logaritmo en base dos del índice de la modulación) y mapeados según los puntos de la constelación de dicha modulación. Posteriormente estos símbolos pasarán por un filtro conformador de pulsos (generalmente filtros gaussianos o filtros de coseno alzado) con el fin de reducir así al mínimo la interferencia entre símbolos (ISI). Este proceso será explicado con más detalle en el capítulo 6, en el cuál, se diseñará un sistema de comunicaciones.

[26] Fuente: TCO IV1.3. Concepto de modulación en sistemas digitales. (2007-08)

3
Componentes Hardware

Como previamente se había introducido, la tecnología SDR se caracteriza por utilizar los siguientes componentes hardware:

- PC
- Periférico Software Radio

También se puede encontrar sistemas embebidos, pero el caso típico es un ordenador interactuando con un periférico de software radio.
La siguiente imagen muestra de manera esquemática el diagrama de bloques de estos componentes:

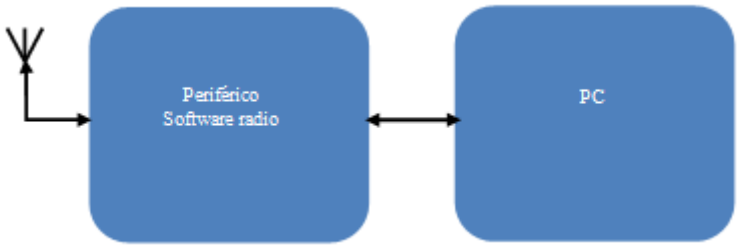


Fig 3.0.1: Esquema componentes hardware

En la actualidad, hay varios fabricantes encargados de proveer plataformas hardware para la tecnología SDR, de entre los que se pueden destacar los siguientes:

- Ettus Research: Es el creador del Universal Software Radio Peripheral.
- National Instruments: Es el creador del NI-Universal Software Radio Peripheral
- Pentek: Fabricante que permite crear una plataforma SDR personalizada según las necesidades del desarrollador.
- Datasoft: Creador del Thunder-SDR
- FlexRadio Systems: Es el creador de SDR-1000.
- Realtek: Creador del rtl2832, un demodulador DVB-T[27] COFDM[28] que puede ser utilizado como un receptor SDR.

El grupo de RFCAs disponía de dos periféricos llamados Universal Software Radio Peripheral fabricados por Ettus, por los que serán los utilizados en este PFC.

[27] DVB-T: Digital Video-Broadcasting Terrestrial es un estándar para la transmisión de televisión digital terrestre.
[28] COFDM:Coded Ortogonal Frecuency Division Multiplexing.

3.1 Ordenador de propósito general

El ordenador de propósito general es uno de los dos componentes hardware necesario para desarrollar un sistema de comunicaciones basado en SDR. La función de procesar la señal en banda base recae en el PC, por lo tanto funciones que antes recaían en la electrónica como por ejemplo la codificación, modulación y demodulación, son llevadas a cabo vía software.
La siguiente imagen representa las funciones desempeñadas por el PC:

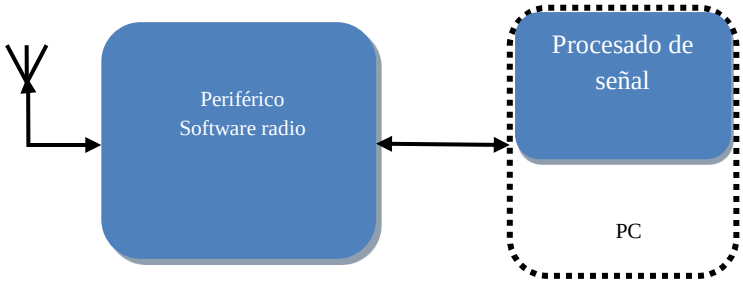
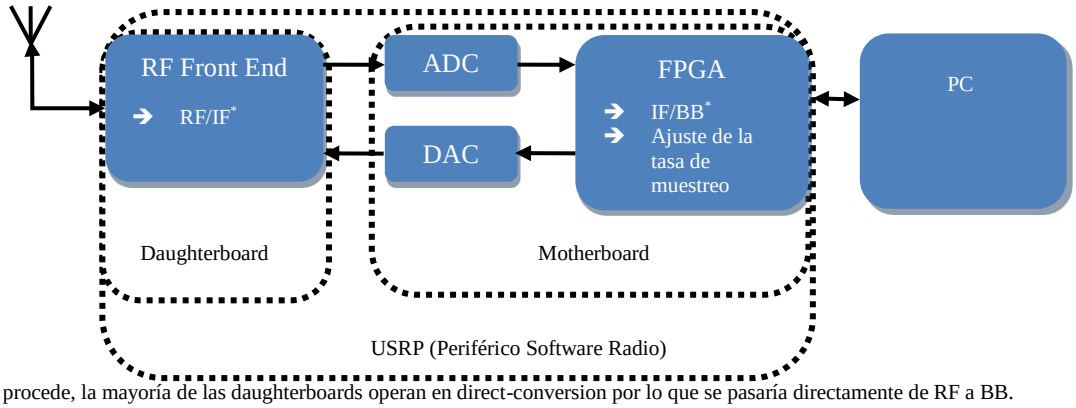


Fig 3.0.2: Función realizada por el PC

3.2 Universal Software Radio Peripheral

El Universal Software Radio Peripheral (USRP) es un periférico que está diseñado específicamente para la implementación de sistemas software radio.
Para poder llevar a cabo el desarrollo de un sistema de comunicaciones basado en plataformas software, el USRP cuenta con una placa madre y una placa hija. Las funciones que realizan estas serán detalladas en los siguientes apartados. Sin embargo, de manera introductoria, se presenta la siguiente figura que representa esquemáticamente las funciones a realizar por el periférico presentado:



*Si procede, la mayoría de las daughterboards operan en direct-conversion por lo que se pasaría directamente de RF a BB.

Fig 3.0.3: Función realizada por el USRP

3.2.1 Familias de USRP

A continuación se presentan los diferentes tipos de USRP [14] actualmente disponibles y sus características más relevantes.

3.2.1.1 Familia Bus Series

La familia de USRP Bus Series se caracteriza porque la conectividad al host se realiza a través de un puerto USB 2.0, permitiendo operar desde DC hasta 6GHz. Para desarrollar aplicaciones usando este hardware, se podrá hacer uso de la herramienta software GNU Radio. Los modelos de USRP de esta familia disponibles son:

3.2.1.1.1 Modelo USRP B100[29]

Fig 3.0.4: USRP B100

El USRP B100 está diseñado para aplicaciones sensibles al coste, cuenta con una Xilinx Spartan 3A 1400 FPGA, con un conversor DAC de doble canal (fase y cuadratura), de 14 bits de resolución y 128 MS/s de tasa de muestreo, con un ADC de doble canal de resolución 12 bits y una tasa de muestreo de 64 MS/s, con DDC[30] y DUC[31] de resolución 15 MHz y conectividad USB 2.0.

3.2.1.1.2 Modelo USRP1

Fig 3.0.5: USRP1

El USRP1 está diseñado para aplicaciones sensibles al coste, cuenta con una Altera Cyclone FPGA. Pensada para proporcionar conectividad a dos *daughterboards* habilitando dos cadenas completas de transmisión/recepción para ello cuenta con dos conversores DAC de doble canal (fase y cuadratura) de 14 bits de resolución y 128 MS/s de tasa de muestreo, con dos ADC de doble canal, de resolución 12 bits y una tasa de muestreo de 64 MS/s, con DDC y DUC de resolución 15 MHz y conectividad con el host mediante USB 2.0.

3.2.1.2 Familia Embedded Series

La familia de USRP Embedded Series se caracteriza por tener dos procesadores uno de propósito general y otro específico. Permitiendo la conexión al host mediante USB 2.0 o bien mediante Ethernet o Fast Ethernet. El periférico puede operar desde DC hasta 6GHz, este hardware está diseñado para desarrollar aplicaciones en modo *standalone*, de tal forma que no se requiere el uso de un PC, los modelos de USRP de esta familia actualmente disponibles son:

[29] Fuente de las imágenes: www.ettus.com
[30] DDC: Digital Down Conversor es un mezclador implementado digitalmente.
[31] DUC: Digital Up Conversor es un mezclador implementado digitalmente.

3.2.1.2.1 Modelo USRP E100

Fig 3.0.6: USRP E100

El USRP E100 cuenta con un procesador OMAP-3 basado en ARM Cortex-A8, con una Xilinx Spartan 3A-DSP 1800 FPGA, con un conversor DAC de doble canal (fase y cuadratura), de 14 bits de resolución y 128 MS/s de tasa de muestreo, con un ADC de doble canal de resolución 12 bits y una tasa de muestreo de 64 MS/s, con DDC y DUC de resolución 15 MHz. La manera de interactuar con el dispositivo puede ser mediante el puerto *console* a través de un puerto serie, mediante el puerto de red y gráficamente mediante una pantalla, teclado y ratón, siendo la primera de ellas el método más cómodo para el usuario, mientras que la interfaz de red es el método más rápido y el requerido para transferencia de archivos desde y para el dispositivo.

3.2.1.2.2 Modelo USRP E110

Fig 3.0.7: USRP E110

El USRP E100 cuenta con un procesador OMAP-3 basado en ARM Cortex-A8, con una Xilinx Spartan 3A-DSP 3400 FPGA, con un conversor DAC de doble canal (fase y cuadratura), de 14 bits de resolución y 128 MS/s de tasa de muestreo, con un ADC de doble canal de resolución 12 bits y una tasa de muestreo de 64 MS/s, con DDC y DUC de resolución 15 MHz. La manera de interactuar con el dispositivo puede ser mediante el puerto *console* a través de un puerto serie, mediante el puerto de red y gráficamente mediante una pantalla, teclado y ratón, siendo la primera de ellas el método más cómodo para el usuario, mientras que la interfaz de red es el método más rápido y el requerido para transferencia de archivos desde y para el dispositivo.

3.2.1.3 Familia Network Series

La familia de USRP Network Series, es la que presenta un mayor rendimiento, soportando la tecnología MIMO. La conexión al host tiene que ser obligatoriamente vía Gigabit Ethernet.
El periférico operara en el rango comprendido entre DC y 6GHz, para poder desarrollar aplicaciones usando este hardware se podrá usar el software GNU Radio, Lab VIEW o Simulink, los modelos actualmente disponibles son:

3.2.1.3.1 Modelo USRP N200

Fig 3.0.8: USRP N200

El USRP N200 está diseñado para aplicaciones que requieran un gran ancho de banda, cuenta con una Xilinx Spartan 3A-DSP 1800 FPGA, con un conversor DAC de doble canal de 16 bits de resolución y 400 MS/s de tasa de muestreo, con un ADC de doble canal de resolución 14bits y una tasa de muestreo de 100 MS/s, con DDC y DUC de resolución 25 MHz, soportando mediante el puerto expansión la sincronización de varios USRP N210 y la configuración 2x2 MIMO[32]. La conectividad entre el dispositivo y el host se lleva mediante Gigabit Ethernet.

[32] MIMO: Multiple inputs multiple outputs

3.2.1.3.2 Modelo USRP N210

Fig 3.0.9: USRP N210

El USRP N210 está diseñado para aplicaciones que requieran un gran ancho de banda, cuenta con una Xilinx Spartan 3A-DSP 3400 FPGA, con un conversor DAC de doble canal de 16 bits de resolución y 400 MS/s de tasa de muestreo, con un ADC de doble canal de resolución 14bits y una tasa de muestreo de 100 MS/s, con DDC y DUC de resolución 25 MHz, soportando mediante el puerto expansión la sincronización de varios USRP N210 y la configuración 2x2 MIMO. La conectividad entre el dispositivo y el host se lleva mediante Gigabit Ethernet.

En el panel frontal del USRP N210 (Fig 3.0.9) aparecen una serie de LEDs los cuáles indican:

- LED A: este led indica que se está transmitiendo.
- LED B: este led se indica si se está utilizando MIMO.
- LED C: este led indica que se está recibiendo.
- LED D: este led indica que el firmware está cargado.
- LED E: este led indica cuando la referencia está en estado locked.
- LED F: este led indica cuando el Complex Programmable Logic Device está cargado.

También es en el panel frontal donde se puede encontrar 4 conectores SMA, dos de ellos (REF1 y REF2), serán los encargados de interconectar la señal de RF con la *daughterboard* para actuar bien como transmisor o bien como receptor. El otro conector SMA es para utilizar una señal de reloj de referencia (REF IN) externa de 10 MHz, mientras que el conector Pulse-Per-Second (PPS IN) se utiliza para mejorar la sincronización de la señal.
Para implementar un sistema de comunicaciones basado en Software Radio será necesario como ya se había mencionado anteriormente el uso de una placa madre y una placa hija, los siguientes apartados concretan las funciones llevadas a cabo por estas placas.

3.2.2 Elección del periférico

La siguiente tabla muestra un resumen de las características más relevantes de los USRP mencionados en el apartado anterior:

Modelo USRP						
	B100	USRP1	E100	E110	N200	N210
Interfaz	USB 2.0	USB 2.0	Embebido	Embebido	Gigabit Eth.	Gigabit Eth.
Ancho de Banda (MSPS 16b/8b)	8/16	8/16	8/16	8/16	50/100	50/100
Nº slots para Daughterboards	1	2	1	1	1	1
Resolución ADC (bits)	12	12	12	12	14	14
Tasa binaria ADC (MSPS)	64	64	64	64	100	100
Resolución DAC (bits)	14	14	14	14	16	16
Tasa binaria DAC (MSPS)	128	128	128	128	400	400
MIMO	No	Sí	No	No	Sí	Sí
Internal GPS	No	No	Sí	Sí	Sí	Sí
1PPS/Ref	Sí	No	Sí	Sí	Sí	Sí

Tabla 3.0.1: Comparativa modelos USRP

En la Escuela Politécnica Superior se disponía de dos periféricos USRP N210, por lo que aprovechando esta situación, será el material empleado para desarrollar el proyecto.

3.2.3 Motherboard

Esta placa es la encargada de comunicar la señal generada vía software desde el host hacia el módulo de RF, el cuál realizará lo necesario a la señal para disponer de esta a la frecuencia requerida para la aplicación a desarrollar. La siguiente figura muestra esquemáticamente las funciones a realizar por la placa madre:

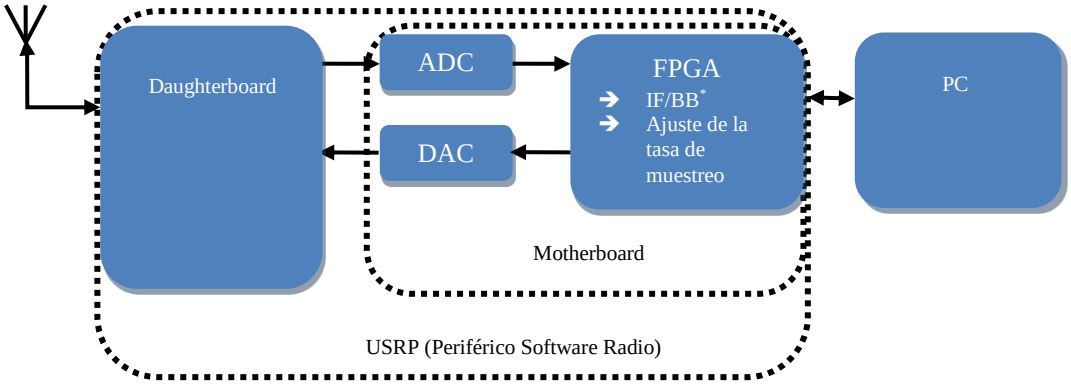


Fig 3.0.10: Función realizada por la Motherboard

Su función comienza a la salida del PC y acaba cuando la señal atraviesa el DAC en el caso del transmisor, mientras que cuando se emplea como receptor su función empieza cuando la señal sale de la *daughterboard* y termina cuando la señal es conducida hacia el PC, encargándose de la conversión de la señal desde el dominio digital al analógico o

del analógico al digital dependiendo de cuál sea el caso y adecuando la tasa de muestreo de la señal para su transmisión.
Con la intención de una mejor comprensión de la *motherboard* del USRP N210 se decide explicar con detalle las funciones de esta placa, la siguiente figura representa las partes fundamentales de ésta:

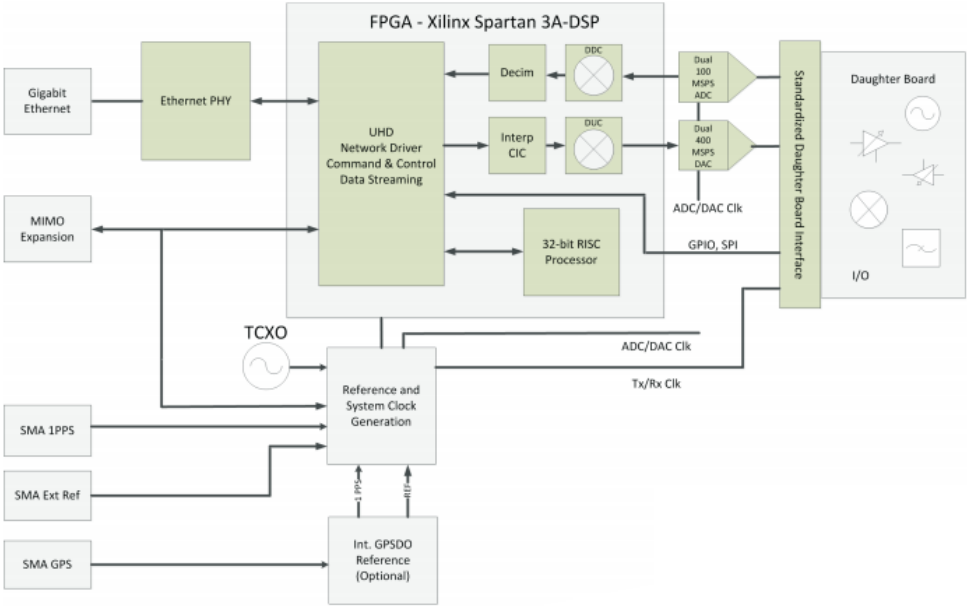


Fig 3.0.11: Placa madre del USRP N210

En el caso de que se desee transmitir una señal, esta será generada vía software y será enviada mediante Gigabit Ethernet hacia el USRP entrelazando las componentes de fase y cuadratura de la señal, una vez la señal llega al puerto Ethernet del dispositivo es conducida hacia la FPGA, esta es la encargada de desentrelazar las componentes de la señal y de realizar tanto operaciones sobre el ancho de banda que se requiera mediante el uso de filtros interpoladores como desplazamientos en frecuencia de la señal haciendo uso de los *Digital Up Converters* (DUC), finalmente la señal a la salida de la FPGA es conducida hacia el DAC de doble canal (uno para cada componente), el cual convertirá las componentes al dominio analógico para su transmisión y las enviará hacia la daughterboard.
Para el caso del receptor, le llega al ADC las componentes de la señal entregada por la placa hija, estas son convertidas al dominio digital y entregadas a la FPGA, quien será la encargada de desplazar estas en frecuencia mediante el *Digital Down Converter* (DDC), de realizar operaciones sobre el ancho de banda la señal mediante filtros diezmadores y de entrelazar las componentes I, Q de la señal para su transmisión a través de Ethernet.

El conversor analógico digital (ads62p44) presenta una ganancia programable 6.5 dB, realizando el ajuste grueso en pasos de 0.5dB (hasta 6 dB) y el ajuste fino en pasos de 0.1 dB (hasta 0.5dB). Mientras que el conversor digital analógico no presenta ganancia configurable.

Como ocurría en los transmisores/receptores de RF tradicionales (superheterodinos) la señal en banda base se trasladaba en frecuencia desde banda base a la frecuencia intermedia y luego posteriormente desde la frecuencia intermedia hasta radio

frecuencia, el USRP N210 (*motherboard+daughterboard*) está diseñado para que el DUC lleve a cabo el traslado de frecuencia desde BB/IF y posteriormente la *daughterboard* llevará a cabo el traslado IF/RF. Sin embargo, muchas *daughterboard* están diseñadas para operar en *direct conversión* eliminando así la necesidad de hacer uso del DUC, por lo que solo se utilizará éste cuando la *daughterboard* no pueda sintetizar la frecuencia requerida por el usuario llevando a cabo un ajuste más fino. En el caso del receptor sucede lo mismo que en transmisor. Tanto el DUC como el DDC implementan el algoritmo CORDIC[33].
Por otra parte los filtros diezmadores o bien interpoladores se utilizan para poner de acuerdo las distintas tasas binarias por una parte que soporta el enlace Gigabit Ethernet y el requerido por la aplicación.

En cuanto a la señal de reloj a utilizar, como se puede apreciar en la figura anterior, esta puede ser bien externa (SMA 1PPS, SMA Ext Ref, SMA GPS) o bien interna. La señal de reloj, por motivos de optimizar la sincronización del dispositivo, será utilizada tanto en la FPGA, en los como conversores ADC, DAC y en la *daughterboard*.

3.2.4 Daughterboard

Estas placas, también conocidas como placas secundarias o placas hija, son las encargadas de cumplir gran parte de las funciones del transmisor y/o del receptor de radiofrecuencia clásico. La siguiente figura muestra esquemáticamente las funciones a realizar por la placa secundaria:

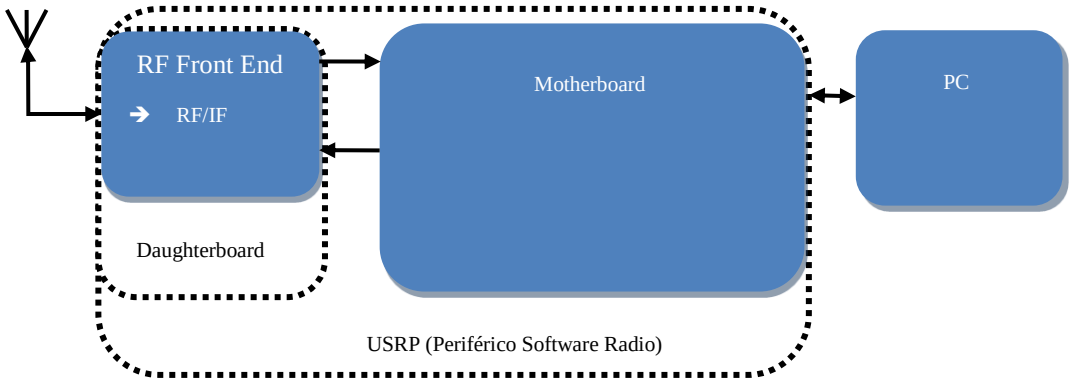


Fig 3.0.12: Función realizada por la Daughterboard

Su función empieza a la salida del DAC y termina cuando es conducida la señal a transmitir hasta el conector SMA en el caso del transmisor, por el contrario en el caso del receptor empieza cuando la señal llega al conector SMA y finaliza cuando ésta es conducida hacia el ADC, por lo tanto este tipo de placas se conecta físicamente a la *motherboard*.
En la placa madre existen dos ranuras etiquetadas como TX y RX, cada ranura tiene acceso a un DAC (para el transmisor) y a un ADC (para el receptor), éstos son de doble canal, permitiendo así la conversión de manera independiente para la componente en fase y en cuadratura de la señal para las placas que lo permitan o para dos canales independientes.

[33] CORDIC: Algoritmo para calcular funciones trigonométricas con bajo coste computacional

Cada placa hija cuenta con una *Electrically Eraseable Programmable Read-Only Memory* (EEPROM), la cual aparte de servir como identificador para el sistema, permite establecer al host la configuración adecuada para su uso correcto (por ejemplo los distintos rangos de ganancia permitidos).

3.2.4.1 Modelos de daughterboards

El siguiente apartado muestra los modelos de *daughterboards* actualmente disponibles y sus características más relevantes:

3.2.4.1.1 Modelo BasicTX[34]

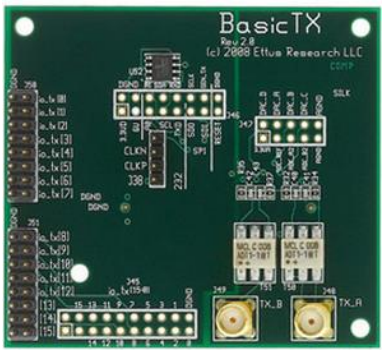


Fig 3.0.13: Daughterboard Basic Tx

Es la placa de transmisor más sencilla, está diseñada para su uso con RF *front-ends* externos, puesto que la salida del conversor DAC está directamente acoplada mediante transformadores de banda ancha a los conectores SMA (TXA y TXB). Esta placa permite operar cada salida de manera independiente (se dispondrían de 2 canales reales), o uno de fase y cuadratura.
Su rango de operación va desde 1 Mhz-250Mhz, su potencia típica de salida es de 1mW y el máximo ancho de banda permitido es 100 MHz, esto ocurre cuando los dos puertos están configurados para fase y cuadratura.

3.2.4.1.2 Modelo BasicRX

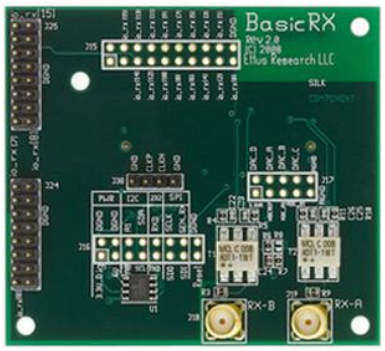


Fig 3.0.14: Daughterboard Basic Rx

Es la paca de receptor más sencilla, está diseñada para su uso con RF *front-ends* externos, puesto que la salida del conversor ADC está directamente acoplada mediante transformadores de banda ancha a los conectores SMA (RXA y RXB). Esta placa permite operar cada salida de manera independiente (se dispondrían de 2 canales reales), o uno de fase y cuadratura. Su rango de operación va desde 1 Mhz-250Mhz, y el máximo ancho de banda permitido es 100 MHz, esto ocurre cuando los dos puertos están configurados para fase y cuadratura.

3.2.4.1.3 Modelo LFTX

Fig 3.0.15: Daughterboard LFTX

En el caso del transmisor LFTX ocurre lo mismo que pasaba con la placa BasicTX, no cuenta con un *front-end*, por lo que está diseñado para aplicaciones que operen en la banda *High Frecuency* (3 MHz-30Mhz), o bien si se requiere otras aplicaciones que operen fuera de este rango, se necesitará un *front-end* externo. La principal diferencia que presenta con

[34] Fuente de las imágenes: www.ettus.com

respecto al transmisor básico es el rango de operación, en este tipo de placas va desde DC hasta 30 MHz esto se debe a que la salida del DAC es conducida al conector SMA (TXA y TXB), a través de un amplificador operacional (uno por cada canal del DAC), su potencia de salida típica también es de 1mW y el máximo ancho de banda soportado es de 60MHz cuando los puertos están configurados para fase y cuadratura.

3.2.4.1.4 Modelo LFRX

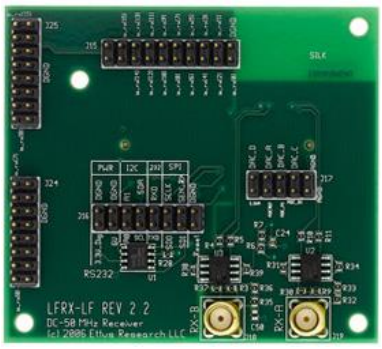


Fig 3.0.16: Daughterboard LFRX

En el caso del transmisor LFRX ocurre lo mismo que pasaba con la placa BasicRX, no cuenta con un *front-end*, por lo que está diseñado para aplicaciones que operen en la banda High Frecuency (3 MHz-30Mhz), o bien si se requiere otras aplicaciones que operen fuera de este rango, se necesitará un *front-end* externo. La principal diferencia que presenta con respecto al transmisor básico es el rango de operación, en este tipo de placas va desde DC hasta 30 MHz esto se debe a que la señal que llega al conector SMA (RXA y RXB), es conducida hacia el ADC a través de un amplificador operacional (uno por cada canal del ADC), el máximo ancho de banda es de 60MHz cuando los puertos están configurados para fase y cuadratura.

3.2.4.1.5 Modelo TVRX2

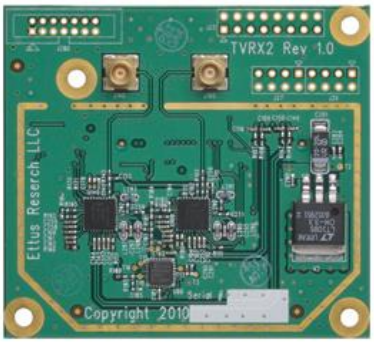


Fig 3.0.17: Daughterboard TVRX2

La placa TVRX2 sirve únicamente para recepción presentando un ancho de banda máximo de 10 MHz (1.7MHz, 6MHz, 7MHz, 8MHz, 10 MHz). Esta placa permite la recepción simultánea en dos bandas diferentes, mediante dos cadenas de receptores independientes (RX1 y RX2). Está diseñada para recepción de las bandas *Very High Frecuency* (VHF) y *Ultra High Frecuency* (UHF) de señales reales operándolas a una frecuencia IF baja. También permite controlar la ganancia vía software, el rango variable de ganancia va de 0dB a 30 dB, presentando una figura de ruido que varía entre 4 y 10 dB. Su rango de operación va desde los 50 MHz hasta los 860 MHz, no soporta la tecnología *Multiple Input Multiple Output* (MIMO).

3.2.4.1.6 Modelo DBSRX2

Fig 3.0.18: Daughterboard DBSRX2

La placa DBSRX2 es un receptor configurable y su frecuencia de operación se mueve desde 0.8GHz hasta los 2.3GHz, soportando la operación con señales complejas. Por defecto esta placa opera convirtiendo la señal de RF directamente a BB y permite tener control sobre la ganancia vía software, este control se realiza en dos etapas, la primera de ellas presenta una ganancia variable que va de 0 dB a 73dB y la segunda de ellas de 0dB a 15dB. Esta placa posee un filtro de canal que permite seleccionar canales desde 1MHz de ancho hasta 60MHz. Presenta una figura de ruido que va desde los 4dB hasta los 8dB y soporta MIMO.

3.2.4.1.7 Modelo WBX RX/TX

Fig 3.0.19: Daughterboard WBX RX/TX

La placa WBX es un transceptor de banda ancha con osciladores independientes para el transmisor y el receptor permitiendo así operar en modo *full-duplex* con señales complejas, por defecto esta placa opera convirtiendo la señal de RF directamente a BB.
El transmisor presenta una potencia típica de salida de 100 mW y un ancho de banda de 40MHz tanto para el transmisor como para el receptor, dicha placa está diseñada también para soportar MIMO.
El rango de operación de la placa es de 50MHz-2200MHz presentando ganancias variables tanto en la cadena del transmisor como en la del receptor, en el caso del transmisor la ganancia variable va desde 0dB hasta los 25dB mientras que en el receptor va desde 0 dB hasta los 31.5 dB, su figura de ruido va desde 5dB hasta 10dB.
La placa WBX tiene dos conectores SMA, uno de ellos se puede configurar bien como transmisor o bien como receptor (TX/RX), mientras que el otro puerto solo puede actuar como receptor (RX2).

3.2.4.1.8 Modelo SBX RX/TX

Fig 3.0.20: Daughterboard SBX RX/TX

La placa SBX al igual que la anterior, se trata de un transceptor de banda ancha con osciladores independientes para el transmisor y el receptor permitiendo así operar en modo *full-duplex* con señales complejas. Por defecto esta placa opera convirtiendo la señal de RF directamente a BB. El transmisor presenta una potencia típica de salida de 100 mW y un ancho de banda de 40MHz tanto para el transmisor como para el receptor, dicha placa está diseñada también para soportar MIMO.
El rango de operación de la placa es de 400MHz-4400MHz presentando ganancias variables tanto en la cadena del transmisor como en la del receptor. Tanto como para el transmisor como para el receptor, la ganancia variable que presenta va desde 0dB hasta los 31.5dB. Su figura de ruido va desde 5dB hasta 10dB.
La placa WBX tiene dos conectores SMA, uno de ellos se puede configurar bien como transmisor o bien como receptor (TX/RX), mientras que el otro puerto solo puede actuar como receptor (RX2).

3.2.4.1.9 Modelo RFX

La familia de placas RFX es un sistema transceptor completo, todas ellas poseen osciladores independientes para la cadena del transmisor y el receptor permitiendo de esta manera la transmisión en modo *full-duplex*. La mayoría de estas placas presentan un indicador de intensidad de señal (*Receive Signal Strength Indication* o RSSI, el cual no está aún implementado) y todas ellas soportan la tecnología MIMO.

Las placas RFX tienen dos conectores SMA, uno de ellos se puede configurar bien como transmisor, o bien como receptor (TX/RX), mientras que el otro puerto solo puede actuar como receptor (RX2), y permiten la operación con señales complejas, las diferentes placas de esta familia que actualmente están disponibles son las siguientes:

- **RFX900:** La placa RFX900 es un transceptor que puede operar en la banda de 750MHz a 1050MHz, está diseñada para operar en la banda de los 900MHz, para poder hacer uso de esta placa en toda la banda, hay que quitar el filtro SAW[35] que limita la banda de operación. Presenta una potencia típica de salida de 200mW.

Fig 3.0.21: Daughterboard RFX900

  El ancho de banda disponible tanto para el transmisor como para el receptor, es de 40MHz. En esta familia de transceptores sólo se puede configurar la ganancia variable en la cadena del receptor presentando así un rango de 70dB.
  Con respecto a la figura de ruido que puede presentar, esta varía entre 5 y 10 dB, siendo 8dB la figura de ruido típica.

- **RFX1800:** La placa RFX1800 es un transceptor que puede operar en la banda de 1500MHz a 2100MHz y presenta una potencia típica de salida de 100mW.
  El ancho de banda disponible tanto para el transmisor como para el receptor, es de 40MHz.En esta familia de transceptores sólo se puede configurar la ganancia variable en la cadena del receptor presentando así un rango de 70dB.

Fig 3.0.22: Daughterboard RFX 1800

  Con respecto a la figura de ruido que puede presentar, esta varía entre 5 y 10 dB, siendo 8dB la figura de ruido típica.

- **RFX2400:** La placa RFX2400 es un transceptor que puede operar en la banda de 2.3GHz a 2.9GHz, está diseñada para operar en la banda ISM. Para poder hacer uso de esta placa en toda la banda, hay que quitar el filtro SAW que limita la banda de operación (en transmisión).

Fig 3.0.23: Daughterboard RFX 2400

  El ancho de banda disponible tanto para el transmisor como para el receptor, es de 40MHz. En esta familia de transceptores solo se puede configurar la ganancia variable en la cadena del receptor presentando así un rango de 70dB.

[35] SAW: Surface Acustic Wave.

Con respecto a la figura de ruido que puede presentar, esta varía entre 5 y 10 dB, siendo 8dB la figura de ruido típica.

3.2.4.1.10 Modelo XCVR2450

Fig 3.0.24: Daughterboard XCVR2450

Es un transceptor diseñado para operar en las bandas de 2.4GHz-2.5GHz y 4.9GHz-5.9GHz, la potencia típica de salida es de 100 mW. La cadena del transmisor y del receptor comparten el mismo oscilador por lo tanto solo puede operar en *half-duplex* con señales complejas, por defecto esta placa opera convirtiendo la señal de RF directamente a BB.

El ancho de banda máximo a utilizar cuando se caracteriza como transmisor es de 48 MHz, pudiendo tomar los valores de 24MHz, 36 MHz y 48MHz, mientras que cuando está configurado como receptor los anchos de banda disponibles son: 15MHz, 19MHz, 28MHz y 36MHz. En cuanto a la ganancia variable que presenta el transmisor, esta se divide en dos etapas, el rango de la amplificación en la primera de ellas es de 5 dB mientras que en la segunda es de 30 dB. Con respecto a la ganancia variable disponible en la cadena del receptor también se diferencian dos etapas siendo el rango disponible en la primera de ellas de 30.5dB, mientras que en la última etapa es de 62 dB.

En cuanto a la figura de ruido que presenta la placa xcvr2450 cuando está caracterizada como receptor puede variar entre los 5 y los 10 dB.

La placa XCVR2450 tiene dos conectores SMA, pudiéndose configurar cada uno de ellos bien como transmisor o bien como receptor (J1 o J2), pero no pudiéndose configurar a la vez (puesto que es *half-duplex*).

Nota: Esta placa en la actualidad no es compatible con el USRP1, si fuese necesario, el fabricante ha creado una placa xcvr2450-u1 exclusivamente para utilizarla con el ya mencionado USRP1.

3.2.4.2 Elección de la placa secundaria

Para poder elegir cuál es la placa que mejor se ajusta a la necesidad del usuario se muestra las siguientes gráficas, las cuales muestran tanto los rangos de operación de las placas como sus posibles aplicaciones:

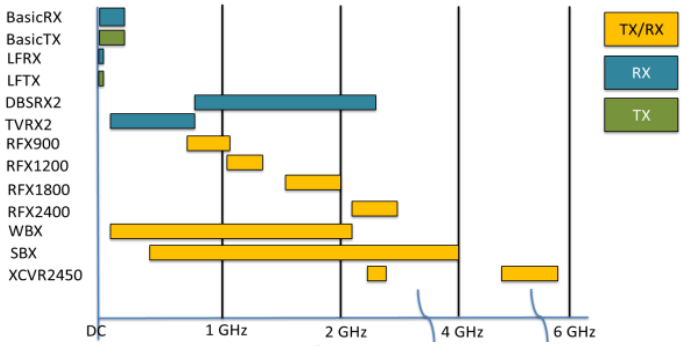


Fig 3.0.25: Elección de la daughterboard

La elección de la placa secundaria puede depender de qué aplicación se quiera desarrollar, la siguiente tabla tiene por objetivo mostrar las *daughterboard* apropiadas para determinadas aplicaciones:

Application Area	Frequency Range(s)	Transmit/Receive	Recommended Daughterboard
TV Broadcast Reception	54-806 MHz	Rx Only	TVRX2, WBX
GPS Reception	L1 – 1575.42 MHz L2 – 1227.60 MHz And others	Rx Only	DBSRX2, WBX, SBX
GPS Record and Playback	L1 – 1575.42 MHz L2 – 1227.60 MHz And others	Rx/Tx	WBX
OpenBTS GSM Basestation	GSM900 – 890-960MHz GSM1800 – 1850-1989 MHz	Tx and Rx	WBX, SBX
WiMAX	2.5 GHz	Rx Only	SBX
Broadcast FM Reception	88-108 MHz	Rx Only	TVRX2, WBX
802.11B/G/N Development	2.4 GHz, 5 GHz	Tx and Rx	XCVR2450
HF Communications	3-30 MHz	Tx and Rx	LFRX + LFTX
Amateur Radio, 2M, 70 cm, 33 cm, 23 cm	~144 MHz, ~430 Mhz	Tx and Rx	WBX
Public Safety/P25 VHF Transceiver	~136-174 MHz	Tx and Rx	WBX
Radar Research	2-4 GHz	Tx and Rx	SBX

Fig 3.0.26: Comparativa entre daughterboards según aplicaciones

En la Escuela Politécnica Superior ya se disponía de dos placas RFX2400 y de dos placas transceptoras xcvr2450 y serán éstas las que se utilizarán para desarrollar el proyecto final de carrera. A continuación se explicará con algo más de detalle el funcionamiento de estas placas.

3.2.4.3 Daughterboard RFX2400

Como ya se había mencionado antes, la placa RFX2400 pertenece a la familia RFX, las cuales son placas transceptoras, es decir, actúan como transmisor y receptor. Además permiten operar en modo *full-duplex*, por lo que se podrá transmitir y recibir al mismo tiempo.

Con la intención de no tomar esta placa como una mera caja negra, se analizará a grandes rasgos su funcionamiento para comprender mejor cuál es su función.
La placa RFX2400 es la encargada de la cadena RF, realizando los traslados en frecuencia necesarios, puesto que su frecuencia de funcionamiento es de 2.3-2.9GHz y están a la salida/entrada del DAC/ADC que operan en banda base. Esta placa se caracteriza por tener una potencia de salida típica de 50mW (17dBm) cuando trabaja como transmisor, además posee un filtro en el *path* del transmisor actuando sobre la banda de ISM[36] (2400-2483 MHz).
A continuación se muestra un diagrama de bloques del transmisor realizado a partir de los esquemáticos [15] y los diagramas observados en la wiki de la página de GNU Radio [16].

3.2.4.3.1 Diagrama del transmisor

A continuación se explica la circuitería empleada para caracterizar la placa RFX2400 como transmisor, para ello se presenta a continuación el siguiente diagrama de bloques:

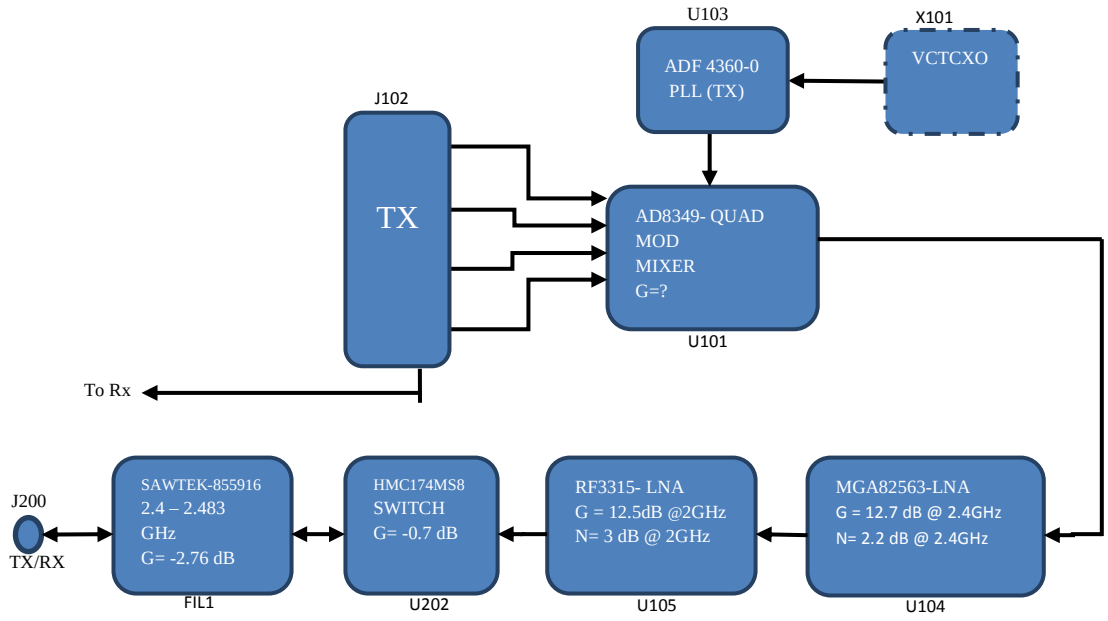


Fig 3.0.27: Diagrama del transmisor (RFX2400)

El diagrama de bloques del transmisor refleja el camino que ha de recorrer la señal, una vez haya salido del conversor Digital-Analógico (DAC, situado en la placa madre) se conduce hasta el módulo de RF, en este caso, la placa RFX2400 a través del conector

[36] ISM: Banda de frecuencia reservada para su uso en entornos industriales, científicos y médicos.

J401 o J102 (dependiendo de la placa en la que se mire). Es hasta este momento cuando se opera con los voltajes de la señal, es por ello por lo que vemos cuatro conexiones al siguiente bloque, dos para representar la señal en fase a transmitir (positivo y negativo) y las otras dos de la señal en cuadratura a transmitir (positivo y negativo).
El bloque principal del transmisor tiene que ver con el mezclador, puesto que es el encargado de trasladar en frecuencia la señal a transmitir, para ello se requiere obligatoriamente:

- Señal de información: Señal que se pretende radiar.
- Oscilador local: Señal de la que nos valemos para realizar el salto en frecuencia.

Si bien ya se ha explicado la primera, es hora de hablar del oscilador local en la placa RFX2400, en el diagrama aparece este bloque en líneas discontinuo, esto se ha hecho así para recalcar que aunque aparezca en los esquemáticos del fabricante, este chip no aparece en la placa, el motivo de esto es que las primeras placas estaban configuradas para que utilizasen osciladores propios. Sin embargo, desde 2006 utilizan el oscilador de la placa madre para conseguir una mejor sincronización entre la placa hija y la madre. Por lo tanto, será el oscilador de la placa madre quien actúe como referencia en el PLL.

El chip que actúa como PLL es en verdad un chip que integra un sintetizador N-entero y un oscilador controlado por tensión, según las especificaciones que da el fabricante del chip, la frecuencia queda descrita por la siguiente ecuación:

$$f_{VCO} = [(P \times B) + A] \times f_{REFIN} / R$$

Ec 3.0.1: Frecuencias generadas en transmisión RFX2400

Dónde:

- P: es el preset del dual-modulus prescaler (P/P+1), pudiendo tomar los valores 8/9, 16/17, 32/33, es el encargado de dividir la frecuencia del reloj del VCO a niveles manejables para los contadores A y B (contadores CMOS).
- FVCO: Es la frecuencia de salida del VCO.
- B: es un contador de 13 bits, toma valores entre 3 a 8191.
- A: es un contador de 5 bits que puede tomar valores entre 0 y 31.
- FREFIN: es la frecuencia de referencia del oscilador de la placa madre.
- R: contador de 14 bits que actúa como divisor, puede tomar valores entre 1 y 16383.

La salida que se obtiene a la salida es diferencial y esta será la entrada del oscilador local al *mixer*.

Una vez se tienen las entradas del mezclador, es hora de analizar el funcionamiento del *mixer*, para ello se muestra la siguiente imagen que representa el diagrama de bloques del chip:

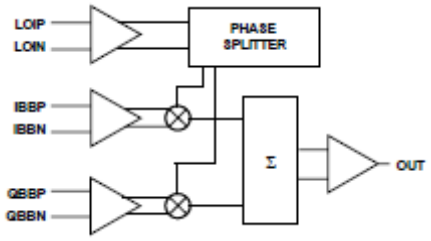


Fig 3.0.28: Mezclador utilizado en la transmisión (RFX2400)[37]

Se trata de un mezclador con un rango de funcionamiento de 700MHz-2700MHz, y el ancho de banda de la señal de entrada es de dC-160MHz.
Como se puede observar, la entrada diferencial del oscilador local es conducida a un amplificador buffer que aumenta la amplitud de la señal de entrada y provee una impedancia de salida de 50Ω, a la salida del buffer se lleva la señal a un *phase splitter*, encargado de obtener la señal del oscilador para el canal fase y otra para el canal de cuadratura (desfase de 90º). Por otra parte, las componentes de la señal a transmitir son conducidas a un amplificador operacional que convierte la señal de tensión diferencial de entrada a una de corriente diferencial, para posteriormente conducirlas hasta el mezclador.
Los mezcladores utilizados en el chip, uno para el canal fase (I) y el otro para cuadratura (Q), están basados en la célula de Gilbert[38], del cual nos servimos para obtener una salida "limpia", cancelando la mayoría de los productos de intermodulación producidos en la multiplicación. Las señales de corriente obtenidas en este proceso son sumadas y conducidas hasta el amplificador de salida.
Una vez se dispone ya de la señal a transmitir en la frecuencia adecuada se hace pasar por dos amplificadores, los cuales quedan caracterizados por presentar buena linealidad, al ser amplificadores de potencia de buena linealidad se sacrifica el no tener mucha ganancia y es por ese mismo motivo por el cual se concatenan los dos amplificadores, hay que mencionar que ambos amplificadores operan en toda la banda.
El siguiente elemento que nos encontramos es un conmutador, el cuál dependiendo de la configuración usada (transmisor o receptor), realizará la conexión física con el circuito necesario, en nuestro caso el puerto TX/RX deberá estar configurado como transmisor y por lo tanto el conmutador dejará pasar la señal hasta el filtro. El filtro opera en la banda de frecuencia de 2.4-2.483GHz (la banda de ISM), la presencia de este se debe entre otros a motivos legales.

Al fijarnos en los esquemáticos que da el fabricante, aparece un condensador (C204) en paralelo al filtro que no está físicamente en la placa, esto es debido a que el fabricante da la posibilidad de quitar el filtro para operar en toda la banda de operación de la placa, para ello habría que quitar el filtro y añadiendo un condensador de capacidad 20-220pF. Otra aplicación que se le da a este condensador es para convertir la placa RFX2400 a una RFX1200. Para ello, es necesario quitar el filtro, soldar un condensador en paralelo

[37] Datasheet AD8349: http://www.analog.com/static/imported-files/data_sheets/AD8349.pdf
[38] Gilbert cell mixer: http://www.radio-electronics.com/info/rf-technology-design/mixers/gilbert-cell-mixer-multiplier.php

al filtro (C204), este condensador ha de tener una carga entre 50-1000pF y por último conectar la placa hija a la madre y grabar de nuevo la
EEPROM usando el comando:

```
$ usrp/host/apps/burn-db-eeprom –f –t rfx1200_mimo_b
```

Fig 3.0.29: Comando para configurar EEPROM

3.2.4.3.2 Diagrama del receptor

A continuación se explica la circuitería empleada para caracterizar la placa RFX2400 como receptor, para ello se presenta a continuación el siguiente diagrama de bloques:

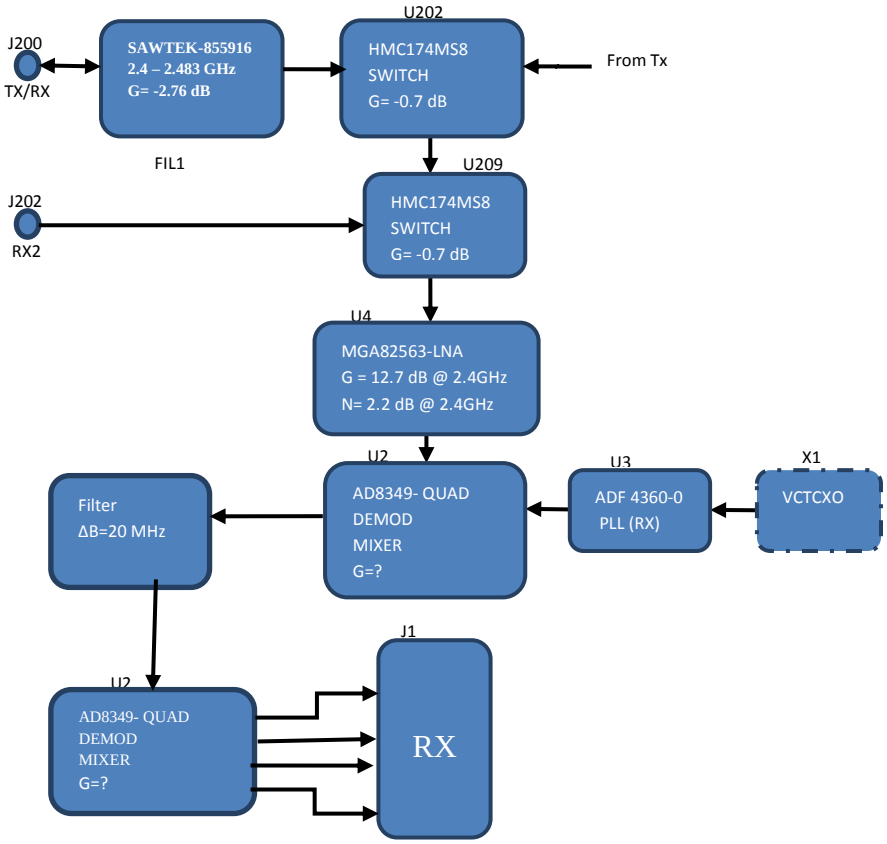


Fig 3.0.30: Diagrama del receptor (RFX2400)

Para una mejor comprensión se decide explicar el funcionamiento de la placa RFX2400 como receptor según el orden de los bloques que la señal atravesará. Lo primero que se puede apreciar, es que en esta placa, la señal se puede recibir por dos puertos distintos (TX/RX y RX2), en este caso se analizará el camino más largo, es decir cuando la señal proviene del puerto TX/RX.
Lo primero que se encuentra la señal cuando llega al puerto TX/RX (J200) es el filtro que como previamente se ha comentado deja pasar la banda ISM (2.4-2.483 GHz), una vez sale del filtro, la señal se encuentra con un conmutador, este era el encargado de elegir la configuración del puerto (bien como receptor o bien como transmisor), la parte de la cadena hasta ahora descrita era compartida con la cadena del transmisor. Una vez

sale del primer conmutador va a parar a un segundo conmutador, este es el encargado de elegir que puerto se ha configurado como receptor, bien TX/RX o bien RX2 (puerto J202). Una vez atraviesa este segundo conmutador, la señal recibida se caracteriza por su bajo nivel de potencia, por lo que hace falta una etapa de amplificación. Para esta etapa, el fabricante utiliza un tipo de amplificador ya usado en la cadena del transmisor, uno que presenta buena linealidad pese a no presentar una alta ganancia, la salida del amplificador se conecta a la entrada del mezclador, que será el encargado de bajar la señal de frecuencia, un dato muy importante que ayudará a entender mejor el funcionamiento global del receptor, es que no existe una etapa de frecuencia intermedia como en los receptores heterodinos (previo paso al traslado de la señal de RF a BB), sino que se realiza en la medida de lo posible una conversión directa, es decir, trasladar la señal RF directamente a BB (de no poder sintonizar la frecuencia en cuestión se emplea una *low-IF* encargándose posteriormente el DUC de trasladarlo definitivamente a BB). Con este proceso se evitan que surjan problemas como es la banda imagen y se elimina también el hardware necesario (filtros) para eliminar esta banda indeseada.
La otra parte que entra en el *mixer* nuevamente es el oscilador local, este aunque es un chip diferente al utilizado en el transmisor, el funcionamiento es exactamente igual. El porqué de usar dos osciladores distintos reside en que la RFX2400 puede operar en *full-duplex*, por lo que hace se hace necesario usar distintas frecuencias para el transmisor y el receptor.
Para entender un poco mejor el funcionamiento de este chip se muestra la siguiente imagen:

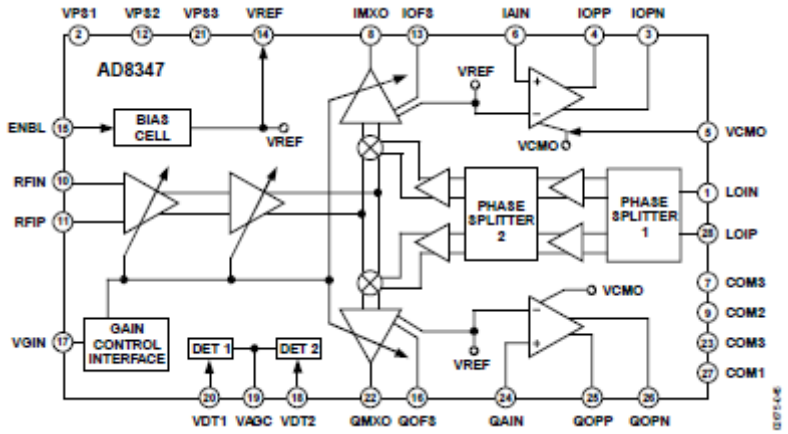


Fig 3.0.31: Mezclador utilizado en la recepción (RFX2400)[39]

Lo primero que hay que destacar de la imagen, es la existencia de dos etapas de amplificación variable, una en RF y una en BB, el rango de ganancia es de 69.5 dB, siendo -30 dB la ganancia mínima (atenuador) y 39.5dB la ganancia máxima, este dato no coincide con el que da Ettus, el cual afirma que el rango total es de 70 dB. El rango en el que el chip opera como conversor directo es de 800MHz a 2.7GHz.
Como se puede apreciar, la señal de RF pasa por dos etapas de amplificación variable antes de entrar a las células de Gillbert.

En cuanto al oscilador, atraviesa un divisor de fase, el encargado de generar la señal del oscilador para el canal de fase y para el de cuadratura.

[39] Datasheet AD8347: http://www.analog.com/static/imported-files/data_sheets/AD8347.pdf

Una vez se ha conseguido trasladar la señal a banda base, atraviesa el amplificador variable de BB y a la salida de éste, la señal es conducida hacia fuera para que realice un filtrado externo. Esto es lo que se puede apreciar en el diagrama de bloques, que la como la señal en banda base atraviesa un filtro de frecuencia de corte 20MHz y de nuevo es conducida hacia el chip para que atraviese una última etapa de amplificación que palie las perdidas introducidas por el filtro y que haga compatible los niveles con los que opera el conversor AD.
Una vez se tiene la señal descompuesta en fase y cuadratura en banda base, es conducida hacia el conector J1 que será el encargado de interconectar esta con el ADC.

3.2.4.4 Daughterboard XCVR2450

Al igual que ocurría con la placa RFX2400, la placa XCVR2450 es un transceptor. Sin embargo, una de las diferencias que presenta con respecto a la RFX2400 es que solo puede operar en modo *half-duplex*. La potencia máxima de salida dada por el fabricante es de 100 mW (20dBm).
Como ya se había hecho con la placa RFX2400, se analizará a grandes rasgos su funcionamiento para comprender mejor cuál es su función.
La placa XCVR2450 es la encargada de la cadena RF, realizando los traslados en frecuencia necesarios puesto que su frecuencia de funcionamiento es de 2.4-2.5 GHz y 4.9-5.9 GHz y se encuentra a la salida/entrada del DAC/ADC. A continuación se muestra un diagrama de bloques realizado a partir de los esquemáticos [15].

3.2.4.4.1 Diagrama del transmisor

A continuación se explica la circuitería empleada para caracterizar la placa xcvr2450 como transmisor, para ello se presenta a continuación el siguiente diagrama de bloques:

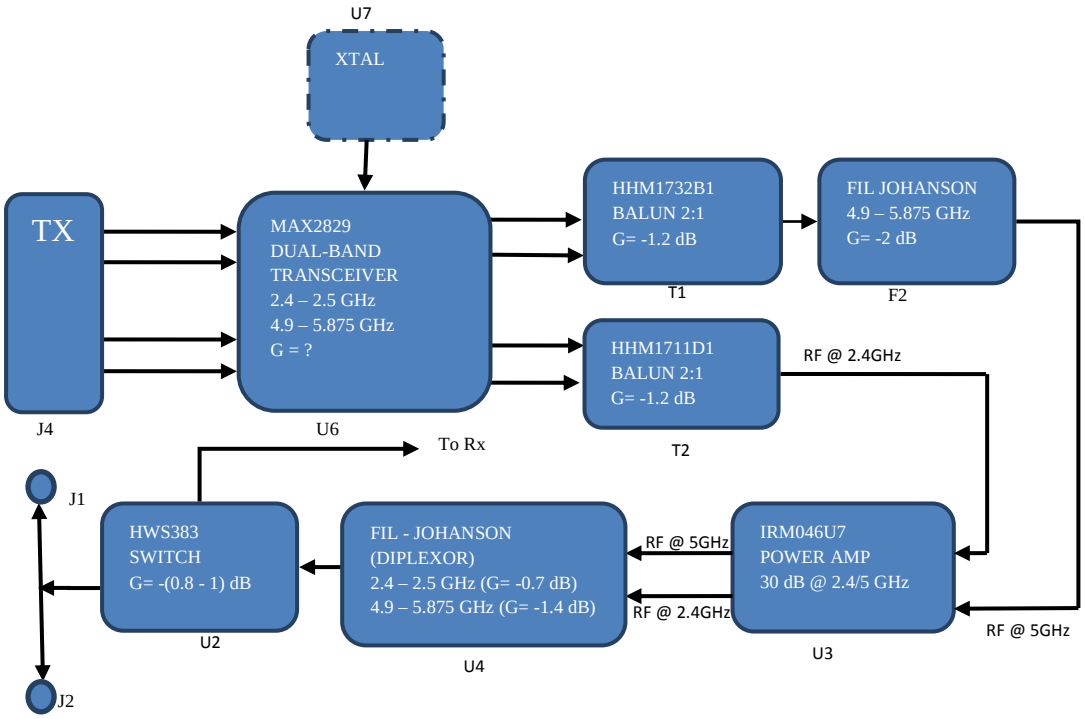


Fig 3.0.32: Diagrama del transmisor (XCVR2450)

El diagrama de bloques del transmisor refleja el camino que ha de recorrer la señal. Una vez haya salido del conversor Digital-Analógico (DAC, situado en la placa madre) se conduce hasta el módulo de RF, en este caso la placa xcvr2450 mediante el conector J401 o J4 (dependiendo de la placa en la que se mire). Es hasta este momento cuando se opera con los voltajes de la señal, es por ello por lo que aparecen cuatro conexiones al siguiente bloque, dos para representar la componente en fase a transmitir (positivo y negativo) y las otras dos de la señal en cuadratura a transmitir (positivo y negativo).
El bloque principal del transmisor tiene que ver con el bloque *dual-band transceiver*, puesto que entre otras funciones es el encargado de trasladar en frecuencia la señal a transmitir, para ello se requiere obligatoriamente:

- Señal de información: Señal que se pretende radiar.
- Oscilador local: Señal de la que nos valemos para realizar el salto en frecuencia.

Si bien ya se ha explicado el camino que ha seguido la señal de información a través del hardware, es hora de explicar las funciones de este segundo chip (*duald-band transceiver*), como ya ocurría anteriormente en la placa RFX2400, en los esquemáticos de la placa que se está analizando aparece un chip (U7) que realiza la función de oscilador local y nuevamente, este no está físicamente. El motivo es el mismo que el del caso anterior, a partir de 2006 las placas de RF utilizan el oscilador local situado en la placa madre para conseguir una mejor sincronización entre ambas placas, en este caso, el chip situado en la posición U6 integra un PLL encargado de obtener una señal de amplitud constante a la frecuencia de referencia.

Para tener una idea de que funciones desempeña el chip que se está tratando de explicar, se muestra a continuación la siguiente imagen:

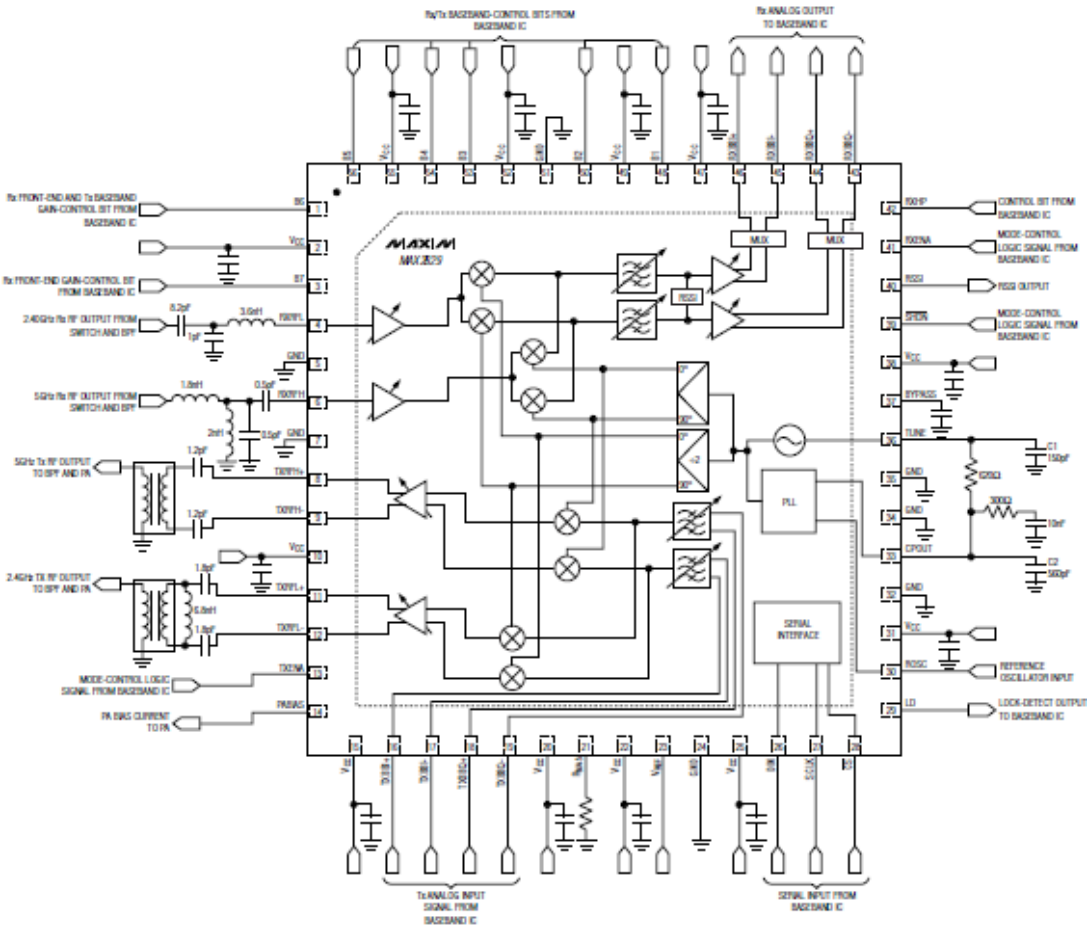


Fig 3.0.33: Mezclador utilizado en transmisión y recepción (XCVR2450)[40]

Cuando el chip es configurado para actuar como transmisor, se distingue dos etapas de ganancia variable, la primera de ellas se trata de la ganancia variable de la señal en banda base y la segunda se trata de la variación de la ganancia de la señal a la salida de los mixers.
El fabricante del chip no da datos acerca de cuáles son los valores extremos, únicamente proporciona el rango de la amplificación, siendo este:

- BB = 5 dB.
- VGA = 30 dB.

Estos datos coinciden con los proporcionados por Ettus.
También se da la posibilidad de elegir la linealidad que se desea que presenten ciertos dispositivos como en el mezclador y el amplificador variable.
Como se puede apreciar en la imagen anterior, la señal de referencia del oscilador local (generada en la placa madre), es conducida al PLL que se encuentra en el interior del chip,una vez se dispone de la portadora a utilizar, ésta va a parar a un *phase splitter*, encargado de obtener la señal del oscilador para el canal fase y otra para el canal de cuadratura (desfase de 90º). Por otro lado, la señal de información en banda base

[40] Datasheet MAX2829: http://datasheets.maximintegrated.com/en/ds/MAX2828-MAX2829.pdf

atraviesa unos filtros programables, los cuales están optimizados para el cumplimiento de los estándares IEEE 802.11 a/g y satisfacer de esta manera el rango de tasas binarias requeridas, para ello los anchos de banda de los que se dispone son: 12 MHz, 18 MHz y 24 MHz. A la salida de estos, el siguiente elemento de la cadena son los *mixers*, en este caso coexisten cuatro multiplicadores para el transmisor, un par de ellos para cada banda (2.4-2.5 GHz y 4.9-5.9 GHz) y para cada componente de la señal (fase y cuadratura). Al igual que ocurría con la placa anteriormente analizada, se realiza en la medida de lo posible una conversión directa.
Antes de que la señal salga del chip (U6), atraviesa un amplificador de ganancia variable (VGA).

Una vez sale la señal de RF en forma diferencial del amplificador programable, esta es conducida a un *balance to unbalance transformer* (balun), que es el encargado de transformar la impedancia de salida de los puertos balanceados (100Ω) a una impedancia de entrada de 50Ω del puerto no balanceado, dependiendo de la frecuencia de la señal RF se utilizará un balun (T1) u otro (T2).
A la salida del balun, la señal será conducida hacia un amplificador de potencia de doble banda (2.4/5GHz), este queda caracterizado por su alta ganancia (30 dB para 2.4GHz y 5GHz), si la señal tiene una frecuencia de 5GHz, antes de ser conducida al amplificador, pasa previamente por un filtro, el cual permite pasar el rango de frecuencias 4900GHz – 5.875GHz.
Una vez se atraviesa el amplificador, la señal va a parar a un diplexor, el cual se encarga de conducir la señal que le entra por alguno de sus puertos (2.4 GHz o 5GHz) hacia su salida, para evitar problemas de acoplamiento de la señal a la otra línea se realiza un filtrado en este chip de las bandas de interés. Finalmente la señal es conducida hasta el puerto que esté configurado como salida, puede ser J1 o J2. Para ello, el conmutador situado justo antes del puerto ha de estar configurado como transmisor.

3.2.4.4.2 Diagrama del receptor

A continuación se explica la circuitería empleada para caracterizar la placa xcvr2450 como receptor, para ello se presenta a continuación el siguiente diagrama de bloques:

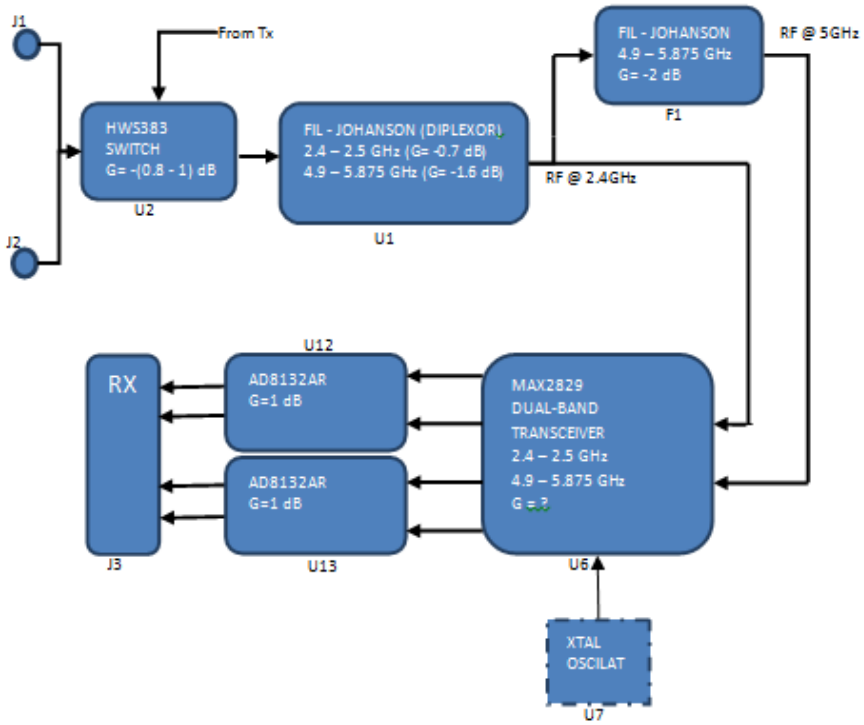


Fig 3.0.34: Diagrama del receptor (XCVR2450)

Para una mejor comprensión se decide explicar el funcionamiento de la placa xcvr2450 como receptor según el orden de los bloques que la señal atravesará. Lo primero que se puede apreciar es que en esta placa la señal se puede recibir por dos puertos distintos (J1 o J2), dependiendo de la frecuencia de la señal recibida en el puerto recorrerá caminos distintos. Lo primero que se encuentra la señal cuando llega al puerto (J1 o J2) es el conmutador, este es el encargado de elegir la configuración del puerto (bien como receptor o bien como transmisor), una vez sale del conmutador va a parar a un diplexor, siendo el encargado de guiar a la señal recibida por un camino u otro dependiendo de su frecuencia. A la salida del filtro situado en el diplexor la señal es conducida hacia el dual-band transceiver, pasando previamente por un filtro dependiendo de la banda de frecuencia de la señal.

Como se puede apreciar, para la cadena del receptor se emplea el mismo oscilador, por lo que esta placa no permite operar en modo *full-duplex* como ocurría con la placa RFX2400.

Cuando el dual-band transceiver está configurado para actuar como receptor, se distingue dos etapas de ganancia variable, la primera de ellas se trata de la ganancia variable de la señal en de RF y la segunda de la señal en banda base.

El fabricante del chip no da datos acerca de cuáles son los valores extremos, únicamente proporciona el rango de la amplificación, siendo este:

- RF = 30.5 dB.
- BB = 62 dB.

Estos datos coinciden con los proporcionados por Ettus.
Una vez la señal sale de los amplificadores variables de RF, la señal se traslada en frecuencia a banda base mediante mixers empleando el método de conversión directa, una vez se dispone de la señal en banda base y como ocurría en el transmisor, la señal es filtrada, para ello se dispone de un filtro con los siguientes anchos de banda: 7.5 MHz, 9.5MHz, 14 MHz y 18 MHz, si comparamos estos anchos de bandas con los que daba el fabricante de la placa hija, se aprecia que hay un factor de x2 de diferencia, esto se debe a la forma de especificar el ancho de bando de una señal en BB, en este caso solo se toma la parte positiva, mientras que Ettus toma tanto la parte positiva como la negativa.
La señal filtrada es conducida hacia el amplificador de banda base, el cual presenta un rango de amplificación de 62dB, A la salida del amplificador ser dispondrá de las componentes I y Q de la señal en modo diferencial, siendo estas llevadas a la salida del chip MAX2829 y conducidas hacia el amplificador diferencial que finalmente lleva la señal recibida hacia el puerto de recepción (J3), el cuál es el encargado de comunicarse con el ADC.

4
Componentes Software

La tecnología SDR se caracteriza fundamentalmente por su componente Software. Dentro de esta, se puede diferenciar dos partes relevantes:

- Software de control.
- Software de procesado de señal.

En general, el Software a utilizar dependerá en gran medida del fabricante de la plataforma hardware (software de control). Recomendando o incluso proveyendo las herramientas software necesarias para el procesado de señal.

4.1 USRP Hardware Driver

El Driver necesario para trabajar con el USRP es el USRP Hardware Driver (UHD), se trata de una librería escrita en C++. Este driver está pensado para trabajar en las plataformas Linux,Windows y Mac.
El objetivo principal del driver es proveer control sobre los productos de Ettus, el uso de este software puede ser utilizado de manera *stand-alone* o recurriendo a otras aplicaciones como son:

- **GNU Radio:** Herramienta software, libre y de código abierto de desarrollo que provee la posibilidad de implementar sistemas basados en software radio mediante el uso de bloques de procesado de señal. Es una herramienta de simulación que puede ser utilizado junto a hardware RF (USRP) para implementar físicamente sistemas de software radio.
- **LabVIEW:** Es una plataforma de desarrollo para diseñar sistemas hardware y software, haciendo uso de un lenguaje de programación gráfico (lenguaje G). Esta plataforma fue creada por National Instruments.
- **Simulink:** Entorno de programación visual de diagramas de bloques integrado en Matlab. Utilizado para la simulación de sistemas, puede ser utilizado junto a hardware USRP para implementar físicamente sistemas de software radio.
- **OpenBTS:** Aplicación de Unix que trata de presentar la interfaz GSM mediante el uso de software radio.

En esta tesis se utilizará el la herramienta GNU Radio como software de procesado de señal.

4.2 GNU Radio

El proyecto GNU radio [17] se inició en 2001 y fue fundado por Eric Blossom con el objetivo de desarrollar un marco de trabajo para software radio. Se trata de una herramienta software libre y de código abierto, constituida por un conjunto de archivos y librerías que proporcionan bloques de procesado de señales, permitiendo así el diseño y la simulación de sistemas basados en software radio.
Esta herramienta software puede ser utilizada con hardware externo adicional (como puede ser el USRP, RTL2832, OsmosSDR…) brindando la posibilidad de implementar físicamente un sistema basado en software radio, o bien, puede ser utilizada en un entorno de simulación.
El funcionamiento de GNU radio se puede concebir como un grafo, donde los nodos simbolizan los bloques de procesado de señal, y la interconexión entre ellos determinará el camino que seguirá la señal comenzando en una fuente y terminando en un sumidero. La siguiente imagen 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:

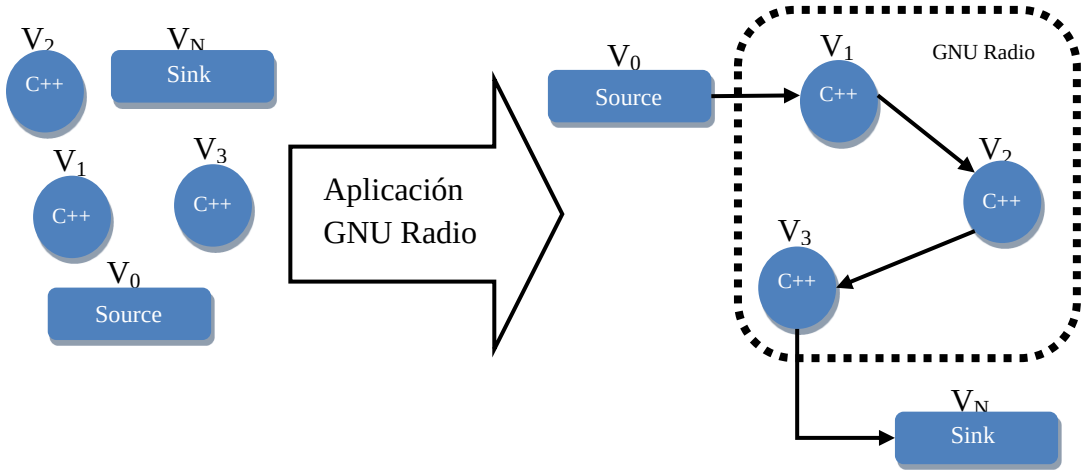


Fig 4.0.1: Interconexión en aplicación GNU Radio

Por lo tanto se puede realizar una primera clasificación en los tipos de bloques usados en GNU radio:

- **Fuentes:** Tales como ficheros, otros programas, hardware radio, micrófono.
- **Sumideros:** Tales como ficheros, otros programas, hardware radio, altavoces, visualizadores gráficos para poder ver la forma de onda de la señal, la FFT, etc.
- **Bloques de procesado de señal:** Tales como filtros, amplificadores, operadores lógicos, operadores matemáticos, moduladores, demoduladores…

Los bloques de GNU Radio se caracterizan por procesar los datos de manera continua desde su entrada hacia su salida, idealmente los bloques desempeñan únicamente una función para así hacer GNU Radio más flexible. Estos, son caracterizados por su número de puertos de entrada y de salida así como del tipo de dato que manejan.
Las fuentes están caracterizadas por tener solo puerto de salida, mientras que los sumideros tienen solo puertos de entrada.

Los tipos de datos que se manejan son byte (1 byte de datos), short (2 bytes de datos), int (4 bytes de datos), float (4 bytes de datos para números en punto flotante), complex (8 bytes de datos, un float para cada componente).
En cualquier aplicación de GNU Radio siempre tendrá que haber algún tipo de fuente y algún tipo de sumidero.

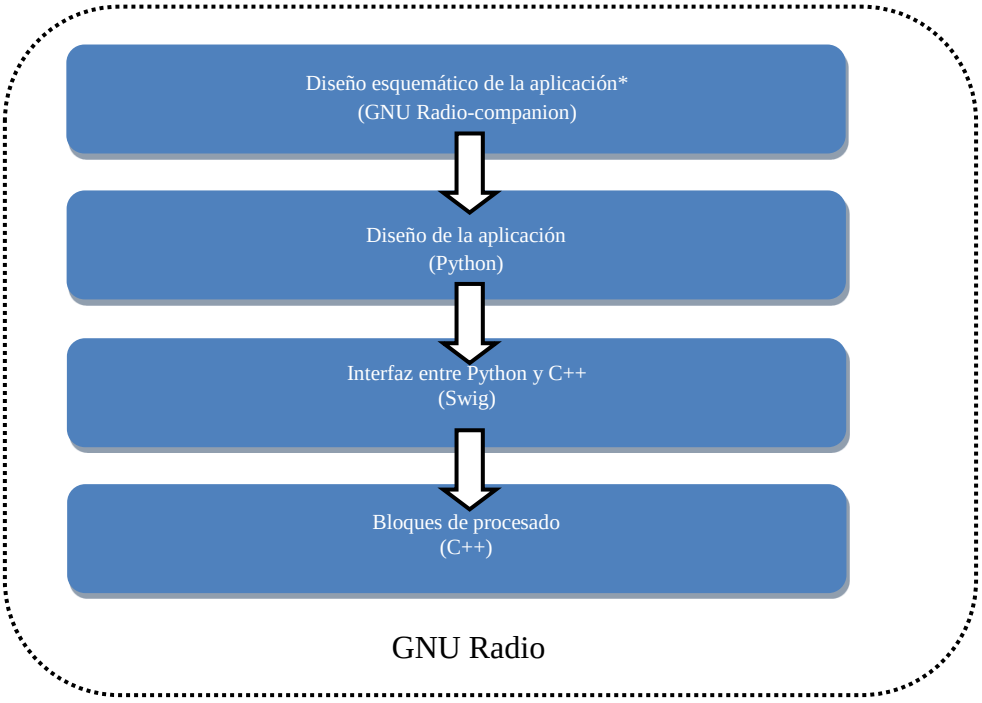
El procesado de señal y en general todo el trabajo a bajo nivel está implementado en C++, mientras que se hace uso de Python para escribir la aplicación, ocupándose de interconectar los bloques a usar. Para que desde un lenguaje de script como es Python se pueda acceder a las funciones implementadas en C++, se utiliza la herramienta software *Simplified Wrapper and Interface Generator* (SWIG).

La creación de la aplicación puede llevarse a cabo típicamente de dos formas:
La primera de ellas es programando directamente la aplicación en Python, mientras que la segunda opción consiste en diseñarla mediante la herramienta gráfica GNU Radio-companion. La segunda opción surge como necesidad de facilitar en la medida de lo posible la tarea al usuario, minimizando así la programación de la aplicación. Por ello, este documento se centrará en el uso de esta herramienta.

Típicamente los bloques de procesado de señal se implementan en cuatro tipos de archivos:

- **Archivos .xml:** En ellos se definen los parámetros del bloque como el tipo de dato a utilizar, el número de puertos de entrada y de salida, las librerías etc. (Requisito necesario para que el bloque esté disponible en GNU Radio-companion).
- **Archivos .h:** Son las bibliotecas de los bloques de procesado de señal
- **Archivos .cc:** Son los archivos que contienen la función que desempeñará el bloque de procesado de señal, se escriben en C++.
- **Archivos .i:** Son los archivos encargados de la comunicación entre los bloques de procesado de señal y la interfaz en Python.

La siguiente imagen muestra la arquitectura software de GNU radio:



*Se puede diseñar una aplicación sin utilizar GNU Radio-companion

Fig 4.0.2: Arquitectura Gnu Radio

GNU Radio funciona sobre plataformas Linux, Mac y Windows, en el apéndice A se indican los pasos a seguir para la correcta instalación de dicha herramienta tanto en Linux como en Windows, así como la correcta configuración de esta para poder utilizar el hardware del que se dispone en este proyecto.

Con motivo de una mejor comprensión en cuanto a los niveles de abstracción utilizados en la arquitectura software utilizada por GNU Radio se presenta el apéndice B, donde se explica el porqué de esta arquitectura.

Una vez se ha presentado el modo en el que funciona la herramienta y su arquitectura software, queda por presentar de qué manera se agrupan las librerías y archivos que conforman GNU Radio. Dichas librerías y archivos son agrupados en módulos dependiendo de la función que desempeñen, los principales módulos que presenta son:

GNU Radio project	
gr	Módulo principal de GNU Radio, esta se necesitará prácticamente en todos los casos puesto que contiene bloques básicos como gran parte de las fuentes, gran parte de los sumideros así como funciones fundamentales como adición, sustracción...
digital	Este módulo contiene las librerías y archivos que se encargan de llevar a cabo las modulaciones y demodulaciones digitales.
audio	Este módulo proporciona control sobre la tarjeta de sonido, permite enviar o recibir señales de audio a través de la tarjeta de sonido.
blocks	Este módulo contiene bloques de procesado comúnmente utilizados en los flow graphs
blks2	Este módulo contiene bloques adicionales escritos en python

trellis	Este módulo provee los archivos necesarios para poder realizar codificaciones convolucionales.
analog	Es en este módulo donde se ubican los archivos relativos a las modulaciones analógicas.
wavelet	Este módulo proporciona bloques de procesado para las trasformadas wavelet.
fft	Este módulo proporciona bloques de procesado para las Fast Fourier Transform (FFT)
window	Este módulo contiene rutinas para el diseño de ventanas.
optfir	Este módulo contiene rutinas para el diseño óptimo de filtros de respuesta al impulso finita (FIR)
filter	Este módulo proporciona bloques de procesado para operaciones de filtrado.
qtgui	Módulo que proporciona sumideros gráficos basados en QT.
wxgui	Módulo que proporciona una GUI basada en Wx.
grc	Módulo necesario para poder utilizar la interfaz gráfica gnuradio-companion.
video_sdl	Este módulo proporciona control para permitir enviar o recibir señales de video.
vocoder	Este módulo incluye varios bloques de procesado los cuales implementan vocoders.
uhd	Este es el módulo que sirve de interfaz a la librería UHD para poder transmitir o recibir datos de los USRP.
How to write a block	Este módulo contiene información para la creación de nuevos módulos e incluirlos al proyecto de gnuradio.
out-of-tree	
osmosdr	Este módulo proporciona el soporte necesario para el uso del hardware osmoSDR.
baz	Este módulo añade nuevas funcionalidades al proyecto GNU Radio, como por ejemplo soporte para el hardware RTL-2832.

Tabla 4.0.1: Módulos GNU Radio

Los módulos de GNU Radio son a su vez estructurados en carpetas, las cuales son las encargadas de agrupar las ya mencionadas librerías y archivos, típicamente un módulo en GNU Radio presenta la siguiente estructura:

	Carpetas	Descripción:
Módulo genérico de GNU Radio	Apps	Esta carpeta contiene ejemplos y aplicaciones de prueba del módulo.
	Cmake	Esta carpeta contiene archivos de configuración necesarios para la correcta instalación del módulo.
	GRC	Esta carpeta contiene los diferentes archivos ".xml" de los bloques para poder usarlos en la aplicación GNU Radio-companion.
	Include	Esta carpeta contiene los archivos fuente de las librerías ".h" de los bloques de procesado.
	Lib	Esta carpeta contiene los archivos fuente ".cc" de los bloques de procesado.
	Python	Esta carpeta contiene los diferentes scripts de Python
	Swig	Contiene los archivos swig ".i" con la configuración del intérprete de C++ y Phyton

Tabla 4.0.2: Estructura de un módulo GNU Radio

Cabe mencionar que cada carpeta (incluida la carpeta raíz) posee un archivo "CMakelist.txt" con la configuración para la compilación de los bloques.

Como previamente se había comentado, el proyecto GNU Radio es de código abierto, por lo que se permite realizar modificaciones en el proyecto, incluso se permite agregar módulos al proyecto existente. Los nuevos módulos que se deseen agregar no forman parte del núcleo de GNU Radio, sino que son instalados fuera del directorio raíz, por este motivo, estos tipos de módulos reciben el nombre de out-of-tree.
Para facilitar esta tarea y dentro del núcleo de GNU Radio, se provee a modo de ejemplo, como se puede realizar este proceso (gr-howto-write-a-block). No obstante, se propone otra forma de realizarlo, haciendo uso de un script llamado gr-modtool.py, el cuál se encargará de generar todo el esqueleto del módulo (tabla 4.0.2), quedando para el desarrollador la tarea de implementar la función a realizar por éste. El apéndice C, muestra con un ejemplo los pasos a seguir para incorporar un nuevo módulo al proyecto GNU Radio.
Otra muestra más de la flexibilidad de la herramienta, es que permite la posibilidad de trabajar junto con otros programas como son octave y/o matlab. Para ello se han desarrollado unos scripts tales como read_tipeofdata_binary.m que permiten leer el contenido de los sumideros de datos, esta es la forma más sencilla de llevar un control del flujo de datos para identificar qué función está desempeñando cada bloque de procesado.

Para interperar con cualquiera de estas herramientas, basta con instalar la nueva herramienta y añadir el directorio donde se encuentran los scripts a su ruta:

```
$ sudo apt-get install octave
$ octave
→ addpath("/home/gnuradio/gnuradio-core/src/utils"))
```

Fig 4.0.3: Comandos Octave

Una vez se ha realizado el proceso anterior bastará con llamar al script correspondiente (dependiendo del tipo de dato que sea el sumidero) y se dispondrá de los datos capturados en modo legible [18].

Para terminar de presentar GNU Radio, faltaría por hablar de que parte del procesado en un sistema tradicional es llevado a cabo en esta herramienta. La siguiente figura muestra las funciones llevadas a cabo por GNU Radio en la cadena de transmisión y recepción.

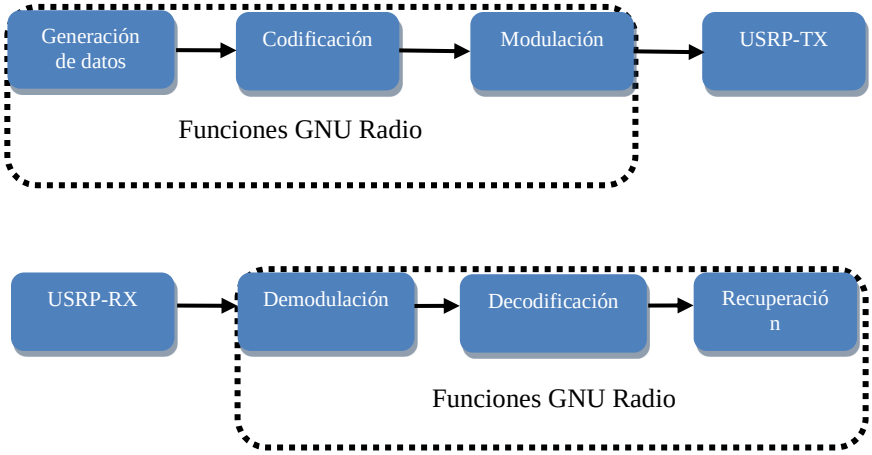


Fig 4.0.4: Funciones desarrolladas en GNU Radio en la cadena del transmisor y receptor

4.2.1 GNU Radio-companion

Como previamente se ha comentado, una aplicación de GNU Radio consiste en la interconexión de bloques (fuente → bloques de procesado → sumidero). GNU Radio-companion surge como alternativa a la programación directa en Python de la aplicación, se trata de una interfaz que permite el diseño de sistemas mediante programación visual. Esta herramienta, muy similar a Simulink de Matlab, genera el código Python de la aplicación de forma automática, permitiendo así, contemplar y modificar directamente el código.
Si se desea modificar la aplicación a partir del código generado, hay que ser consciente que cada vez que se ejecute la aplicación desde la interfaz, ésta sobrescribirá el script generado, por lo que si se requiere modificarlo, se tendrá que renombrarlo evitando así el problema descrito. Cabe mencionar que el código generado generalmente es menos legible que uno que se haya programado a mano.

Para añadir un bloque en el esquemático bastará con dar doble click sobre el que se desea añadir y para interconectar bloques, bastará simplemente con seleccionar los bloques a unir siguiendo el orden en el que vayan.
Con motivo de presentar la interfaz mencionada, se desarrolla un ejemplo en entorno de simulación en el que no entra en juego el hardware descrito en el capítulo anterior. Este ejemplo consiste en dos fuentes (dos cosenos) de frecuencia y amplitud variable, éstas serán sumadas mediante un bloque que lleva a cabo la adición de las señales de entradas y el resultado de esta operación será llevada a la tarjeta de sonido (audio sink) para que pueda ser escuchado. El resultado también será llevado a bloques que ilustrarán la forma de onda de la señal y su espectro.
La siguiente imagen muestra la configuración del sistema detallado anteriormente:

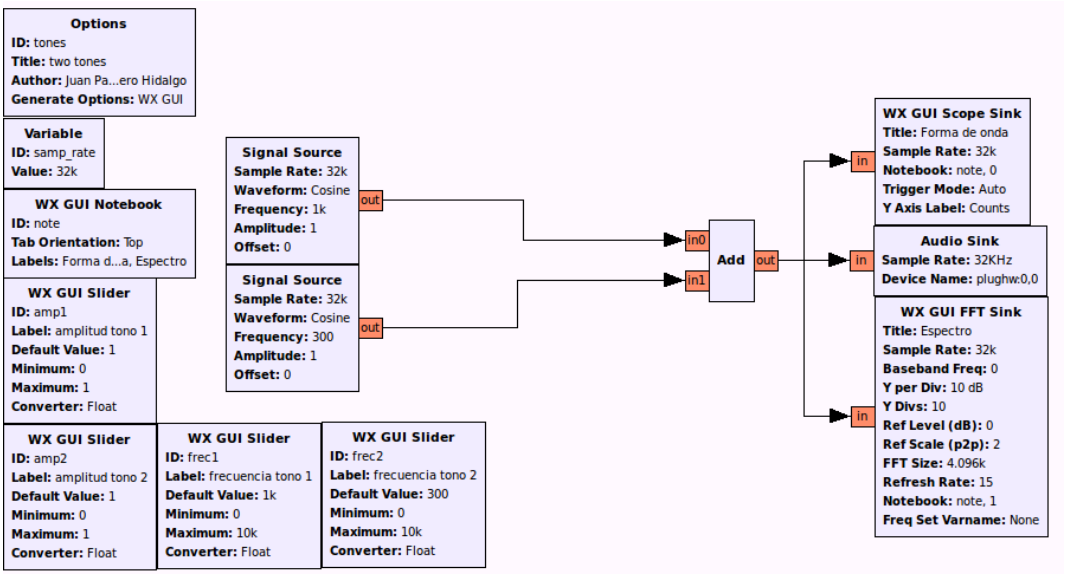


Fig 4.0.5: Diagrama de bloques de la aplicación tones

Como se puede apreciar en la figura anterior, además de los bloques ya descritos, se han introducido ciertos parámetros para poder hacer variables las amplitudes y frecuencias de los tonos. También se ha introducido otro parámetro para poder presentar visualmente la forma de onda de la señal resultante y su espectro en pestañas diferentes.
Un aspecto destacable es que el valor de la amplitud se define en relación al rango que presenta el DAC, posteriormente se hablará de nuevo de este tema.

La siguiente imagen muestra los resultados gráficos obtenidos:

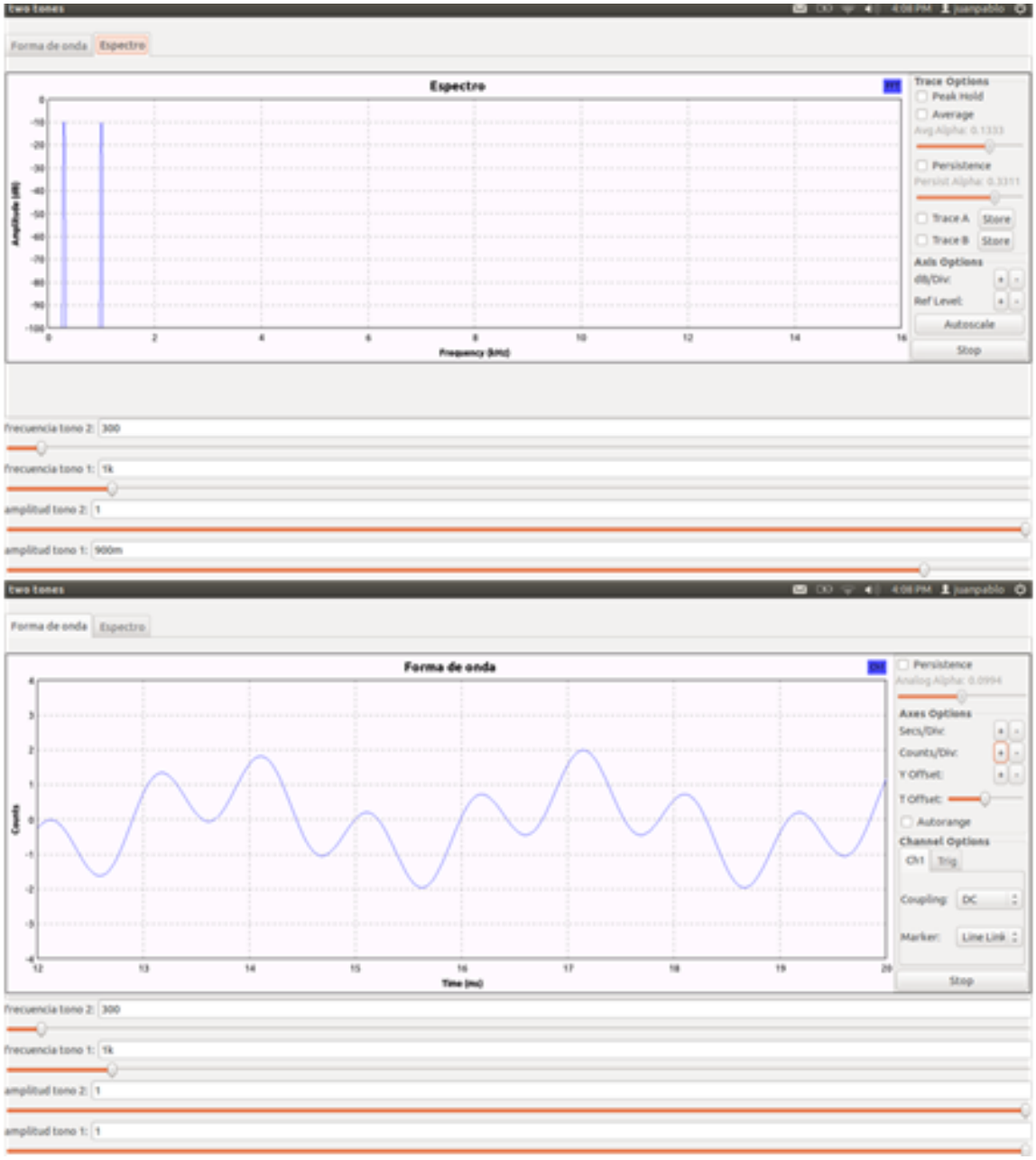


Fig 4.0.6: Espectro y forma de onda de la aplicación tones

Llegado a este punto, se ha presentado la interfaz que se va a utilizar fundamentalmente para el diseño de los sistemas a implementar en este PFC. A modo introductorio al lenguaje Python, se diseña una aplicación muy parecida a la mostrada sin recurrir a esta interfaz, mostrándose el código en el apéndice D.

4.3 Resumen

Este apartado tiene por objetivo ofrecer una perspectiva general esquematizada de las funciones que desempeñan las herramientas software y el hardware presentado en los capítulos previos

Mientras que el USRP es el encargado de realizar todas las operaciones cuando se dispone de la señal en RF o en IF (dependiendo del modelo de *daughterboard* que se utilice), GNU Radio se encarga de realizar las operaciones en banda base. La configuración del dispositivo (USRP) la lleva a cabo GNU Radio, el cual a través de UHD modifica parámetros tales como la elección de la antena (si procede), de la frecuencia de RF deseada, ganancia y tasas de diezmado e interpolación, esta información es conducida hacia la FPGA que es la encargada de realizar el interpolado y diezmado a la señal y configurar los elementos de la cadena de RF tales como mixers, amplificadores variables y filtros.

La siguiente figura muestra a modo resumen como se relacionan los componentes descritos en los capítulos previos:

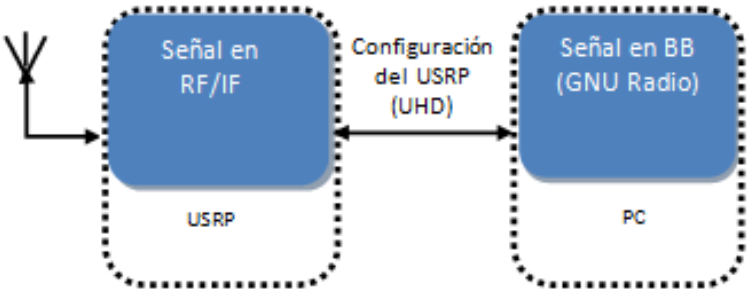


Fig 4.0.7: Componentes Hardware y Software

5
Pruebas iniciales

Las pruebas básicas han sido realizadas tanto para la placa RFX2400 como para la XCVR2450, comprobando así el correcto funcionamiento de estas. Las pruebas llevadas a cabo se detallarán en el siguiente capítulo mostrando los resultados obtenidos.
Hay que recalcar la importancia que juega este capítulo en el proyecto puesto que es en este, donde se verifica mediante elementos externos tales como analizadores de espectro y generadores de señal, que se está transmitiendo una señal RF y no se está en un entorno de simulación. Para ello se realizarán unas pruebas iniciales en las se caracterizarán a los USRP como transmisor y como receptor utilizando los elementos externos ya mencionados. Una vez se haya probado que se está controlando la señal a transmitir y recibir, será el momento en el que se caractericen como transmisor y receptor para la comunicación directa entre ellos.

5.1 Pruebas de transmisión

Para la realización de esta prueba se tendrá que hacer uso del siguiente material:

- USRP N210 (*motherboard* + *daughterboard* (tx))
- Analizador de espectros
- Conectores tipo N macho SMA macho
- Atenuador

Para llevar a cabo las pruebas de transmisión se ha usado la interfaz uhd_siggen_gui a la cual se puede acceder escribiendo en la terminal:

```
$ uhd_siggen_gui
```

Fig 5.0.1: Comando para caracterizar a un transmisor desde la terminal

Esta interfaz contempla los diferentes tipos de *daughterboards* del fabricante ettus y muestra los diferentes rangos de ganancias que estas presentan.
El siguiente esquema representa la conexión física llevada a cabo:

Fig 5.0.2: Conexión física para transmisión

5.1.1 RFX2400

Esta placa tiene un filtro que deja pasar la banda ISM (2.4 – 2.483 GHz), la potencia máxima de salida es de 17 dBm según las especificaciones.
Se ha utilizado un atenuador de 20 dB para proteger el analizador de espectros.

Los resultados obtenidos son los siguientes:

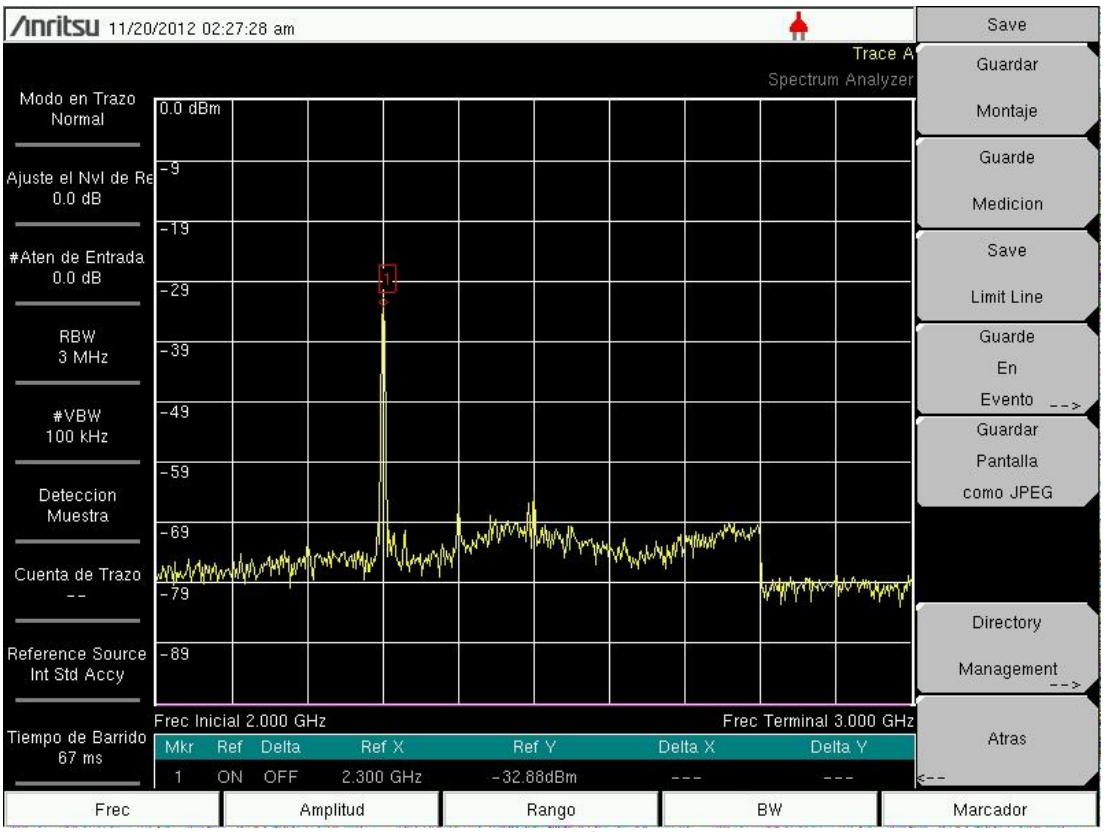


Fig 5.0.3: Prueba transmisión RFX2400 @2.3GHz

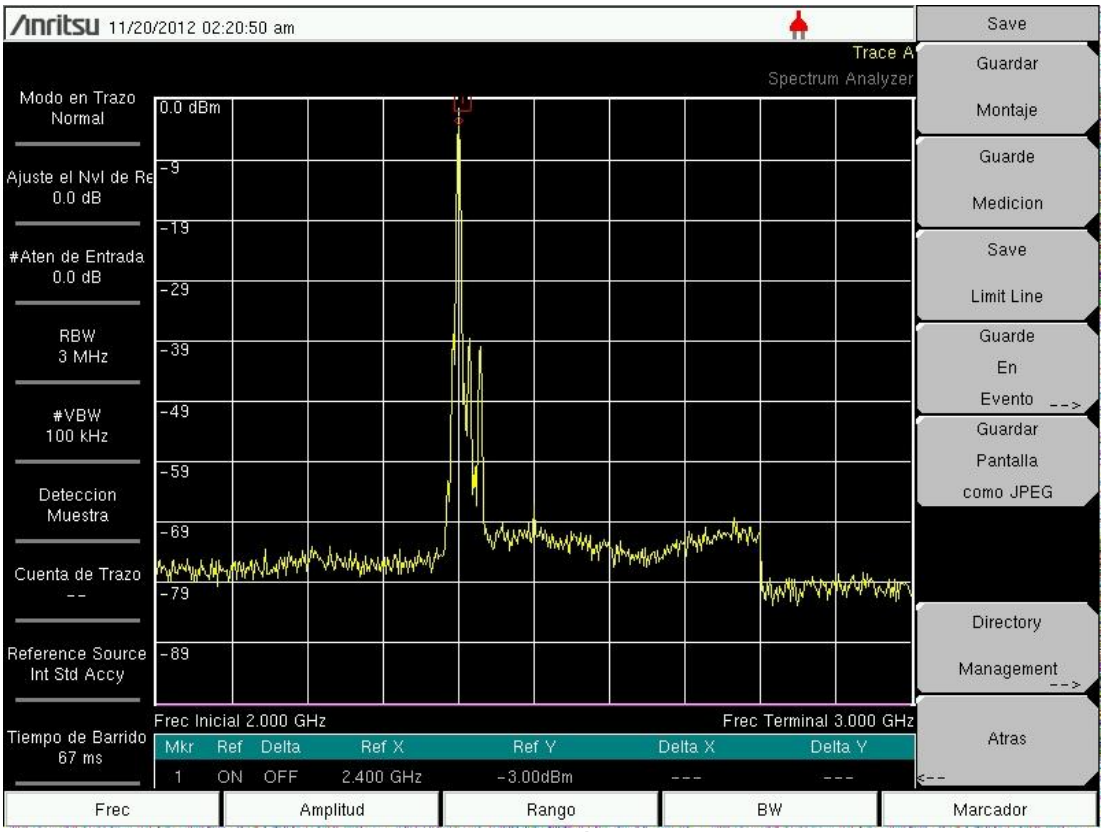


Fig 5.0.4Prueba transmisión RFX2400 @2.4GHz

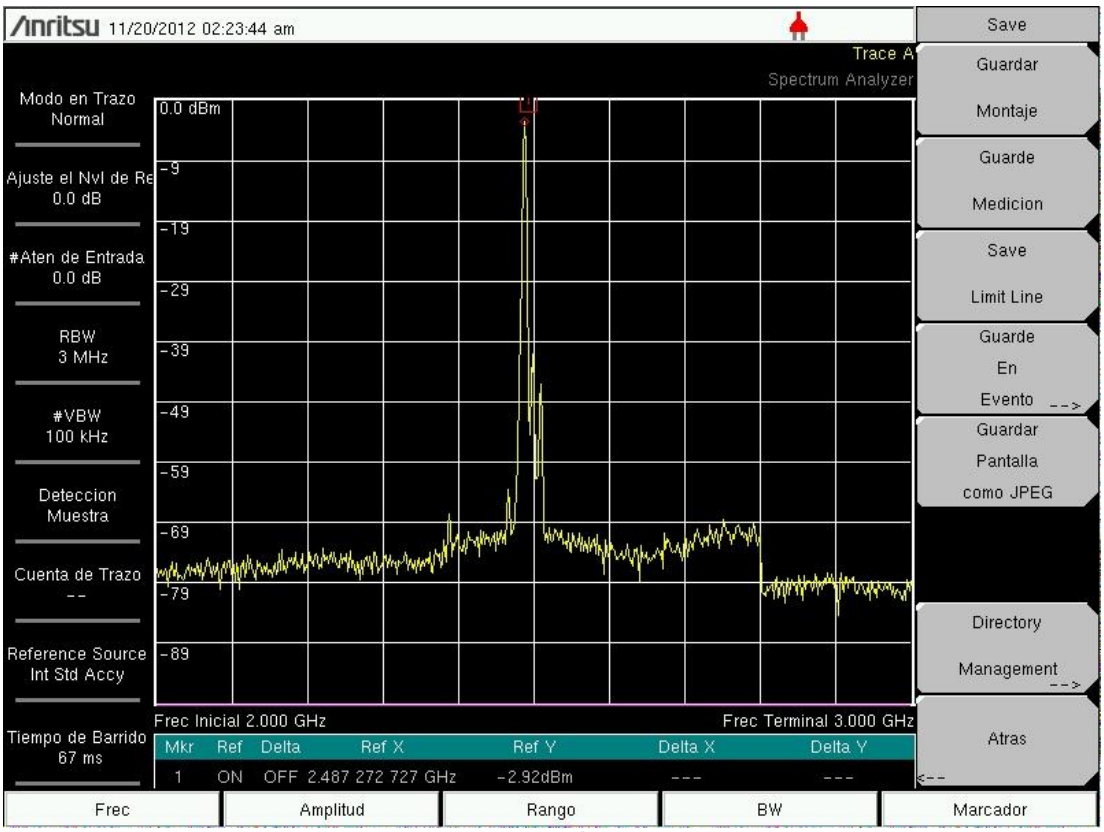


Fig 5.0.5: Prueba transmisión RFX2400 @2.483GHz

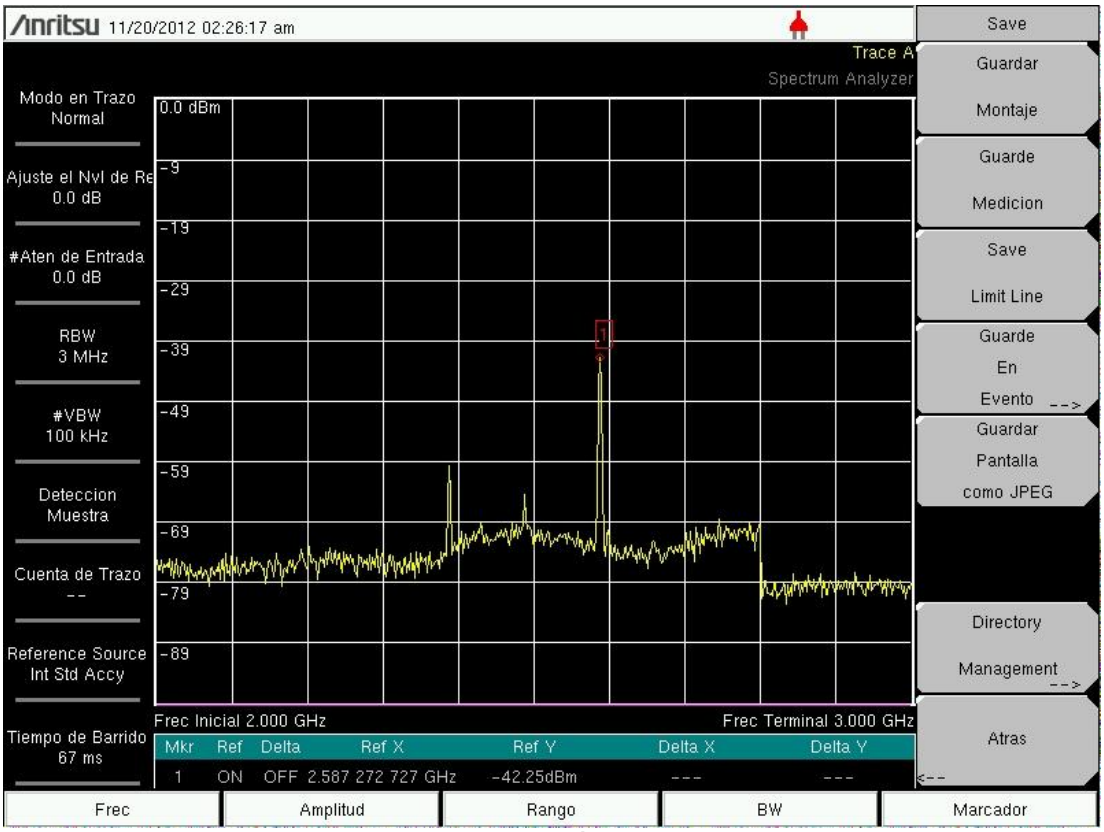


Fig 5.0.6: Prueba transmisión RFX2400 @2.583GHz

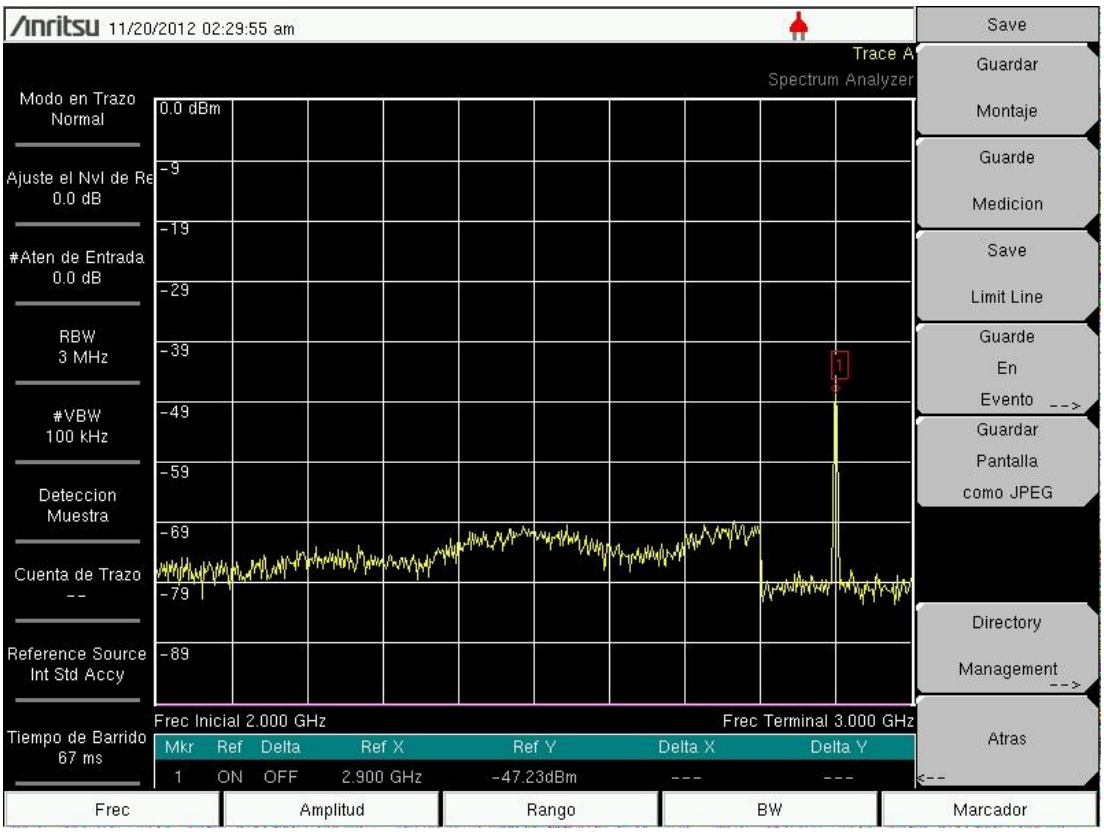


Fig 5.0.7: Prueba transmisión RFX2400 @2.9GHz

Como se puede apreciar, en la banda de transmisión (2.4-2.483 GHz) la potencia de salida se ajusta mucho a la especificada por el fabricante, mientras que al salir de dicha

banda la señal transmitida es atenuada. Otro efecto a destacar según los resultados obtenidos son las emisiones espurias que se producen en el *mixer*, el cuál se encargada de subir en frecuencia la señal a transmitir, el nivel entre las frecuencias espurias y la *carrier* es típicamente de -36dBc, dato que concuerda con el nivel del tercer armónico que presenta el dispositivo AD8349.
En cuanto a la imprecisión que presentan algunas de las frecuencias generadas (2.483GHz) viene dado por dos motivos, el primero de ellos dependerá del *span* tomado en el analizador de espectros (para reducir este efecto se reducirá el *span* utilizado). El segundo motivo es debido a la precisión de los sintetizadores de frecuencia utilizados tanto en la generación de la señal (*daughterboard*) como en el analizador de espectros.

5.1.2 XCVR2450

El rango de funcionamiento de esta placa es de 2.4 -2.5 GHz y 4.9-5.9GHz, la potencia típica de salida de esta es de 20 dBm, al operar en un rango de frecuencia más alto del que trabaja el atenuador utilizado en el caso anterior, se hace uso de un atenuador de 30dB cuyo rango de operación va desde 0 Hz hasta 17GHz

Los resultados obtenidos son los siguientes:

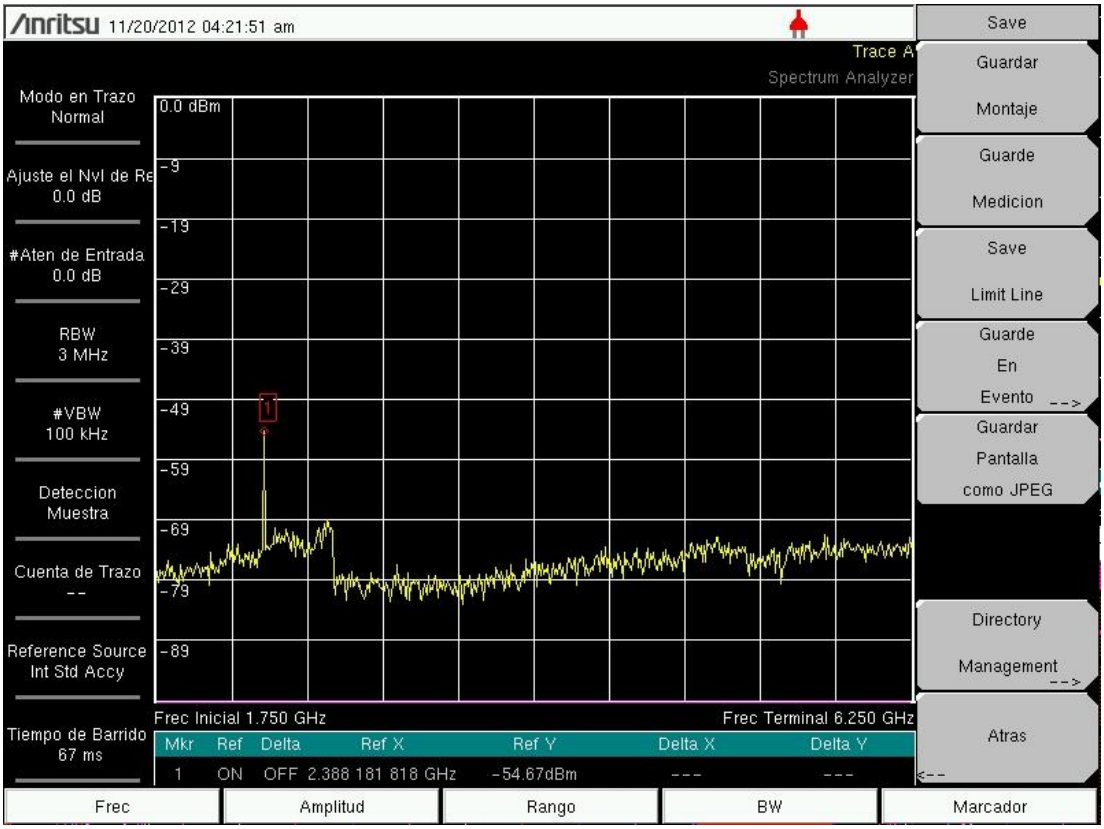


Fig 5.0.8: Prueba transmisión XCVR2450 @2.3GHz

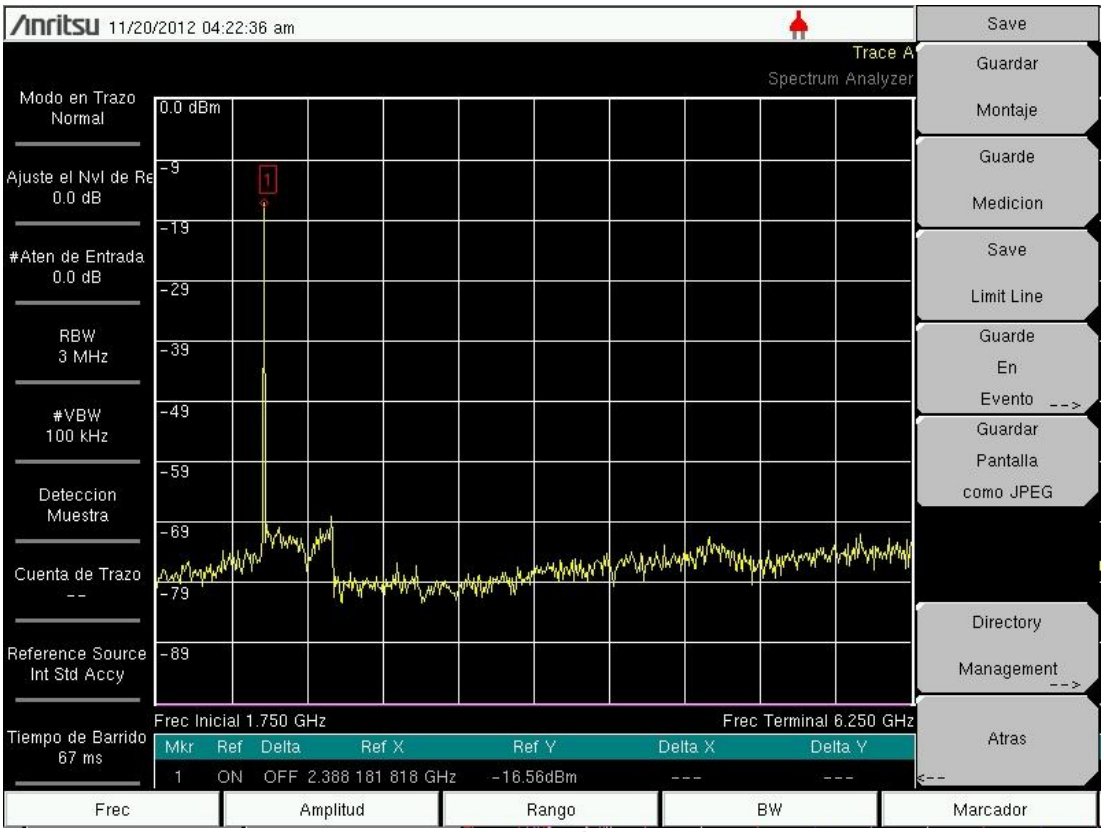


Fig 5.0.9: Prueba transmisión XCVR2450 @2.4GHz

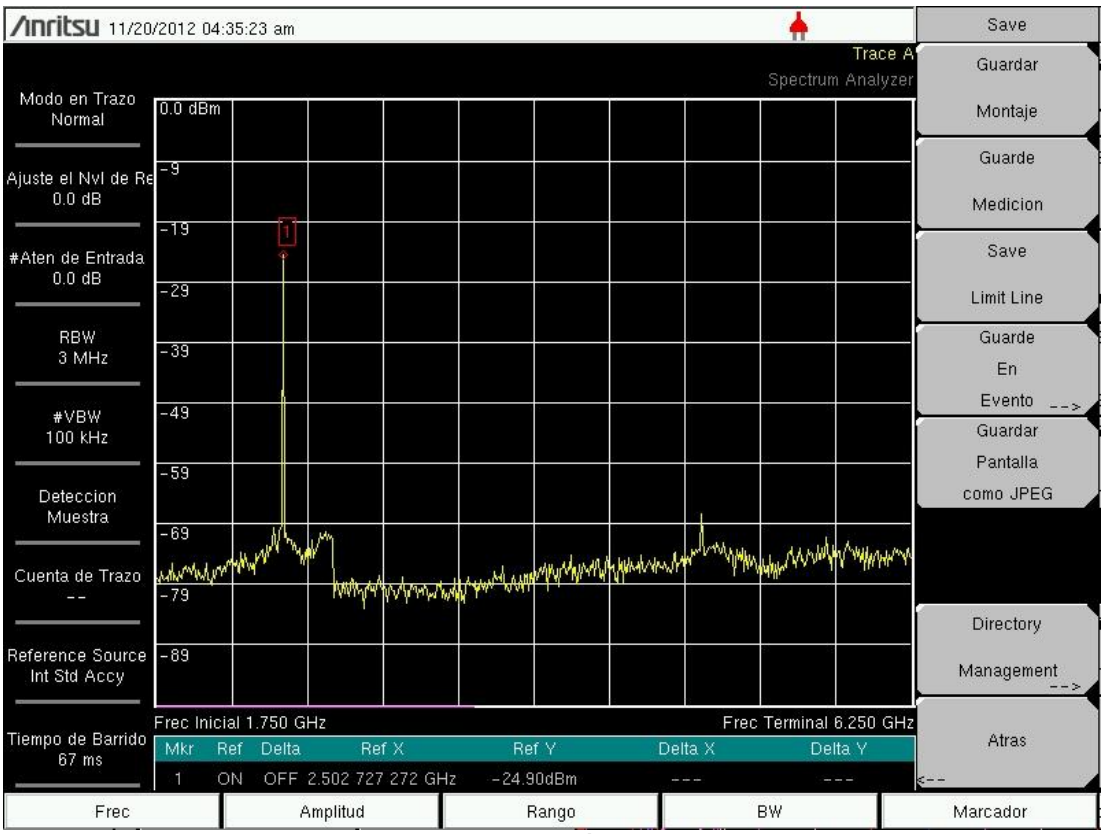


Fig 5.0.10: Prueba transmisión XCVR2450 @2.5GHz

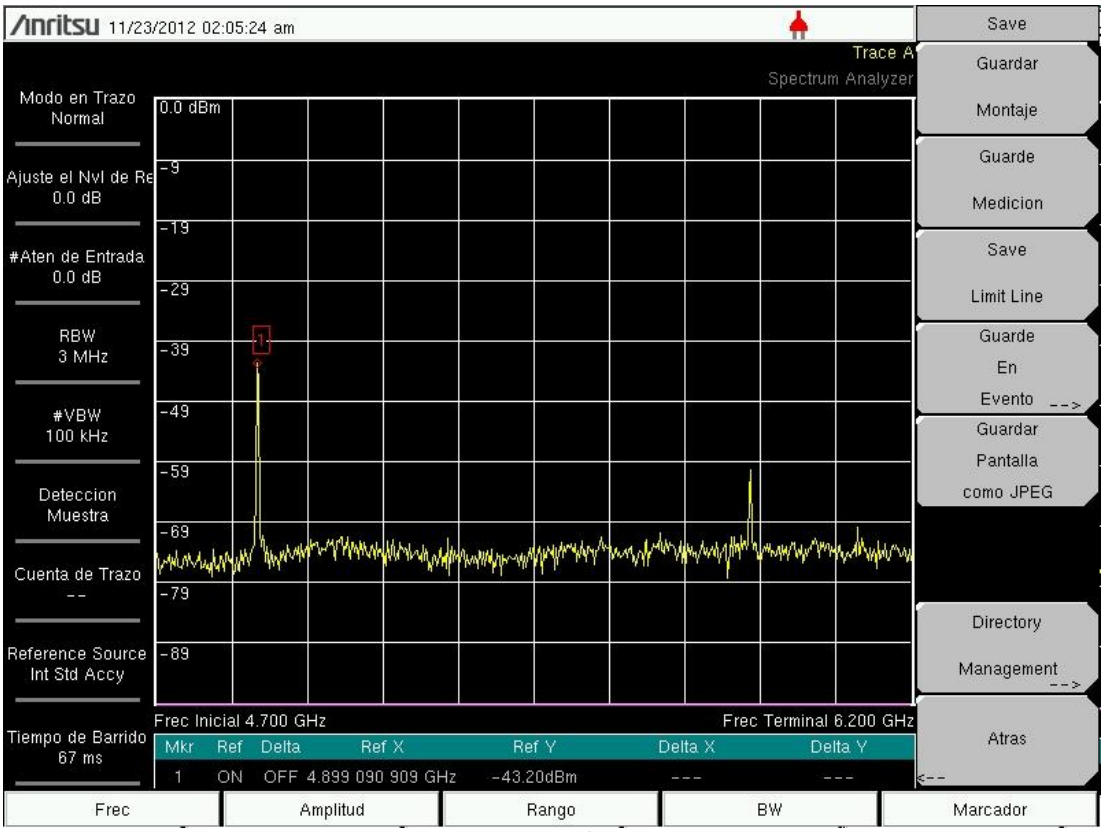


Fig 5.0.11: Prueba transmisión XCVR2450 @4.8GHz

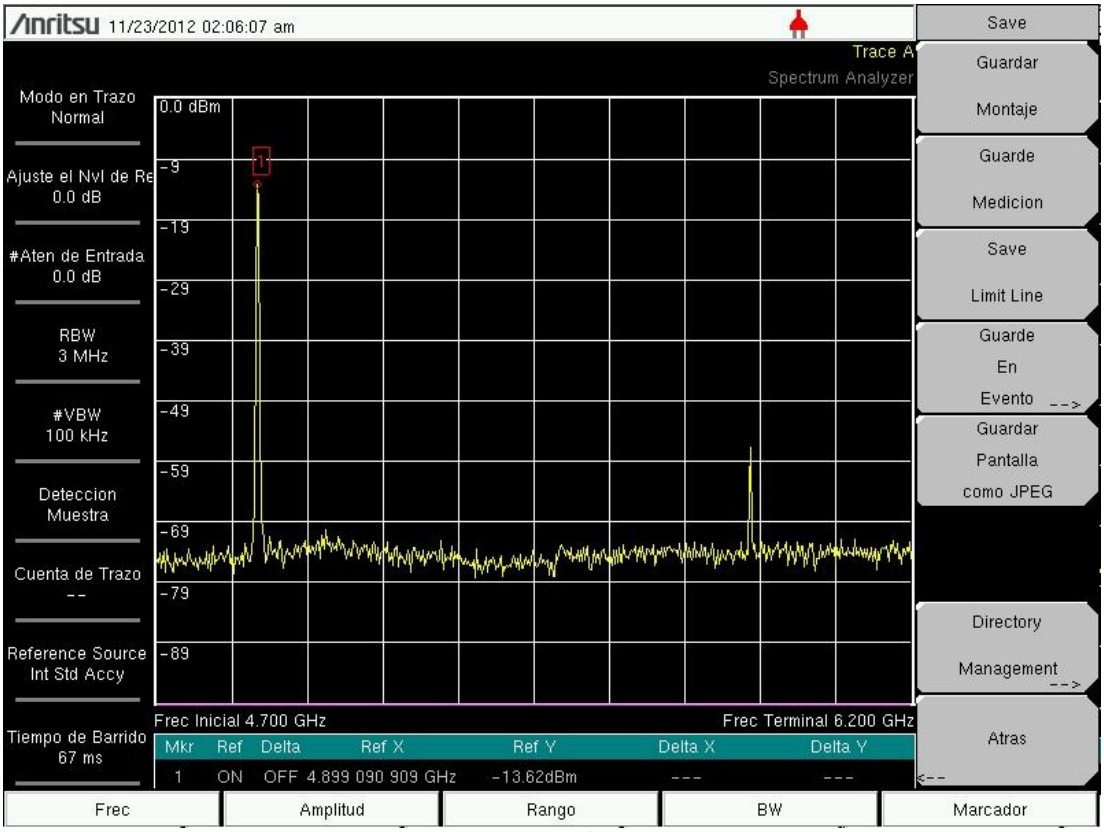


Fig 5.0.12: Prueba transmisión XCVR2450 @4.9GHz

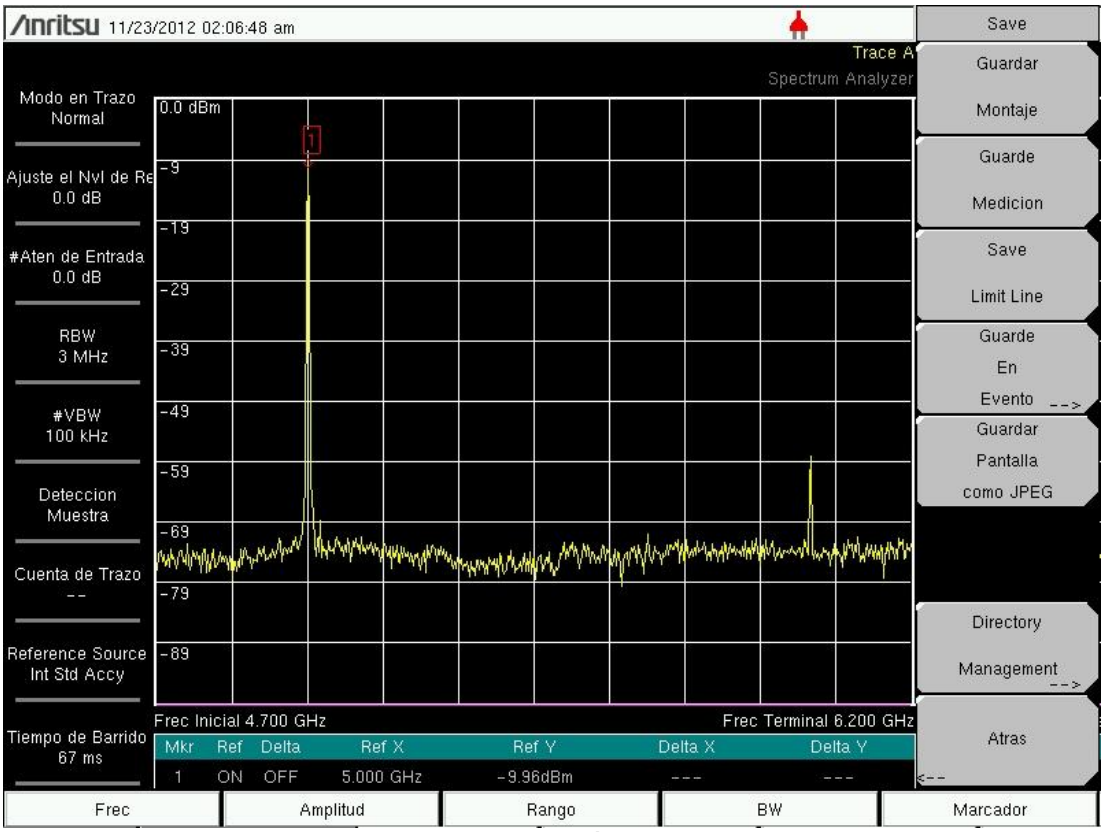


Fig 5.0.13: Prueba transmisión XCVR2450 @5GHz

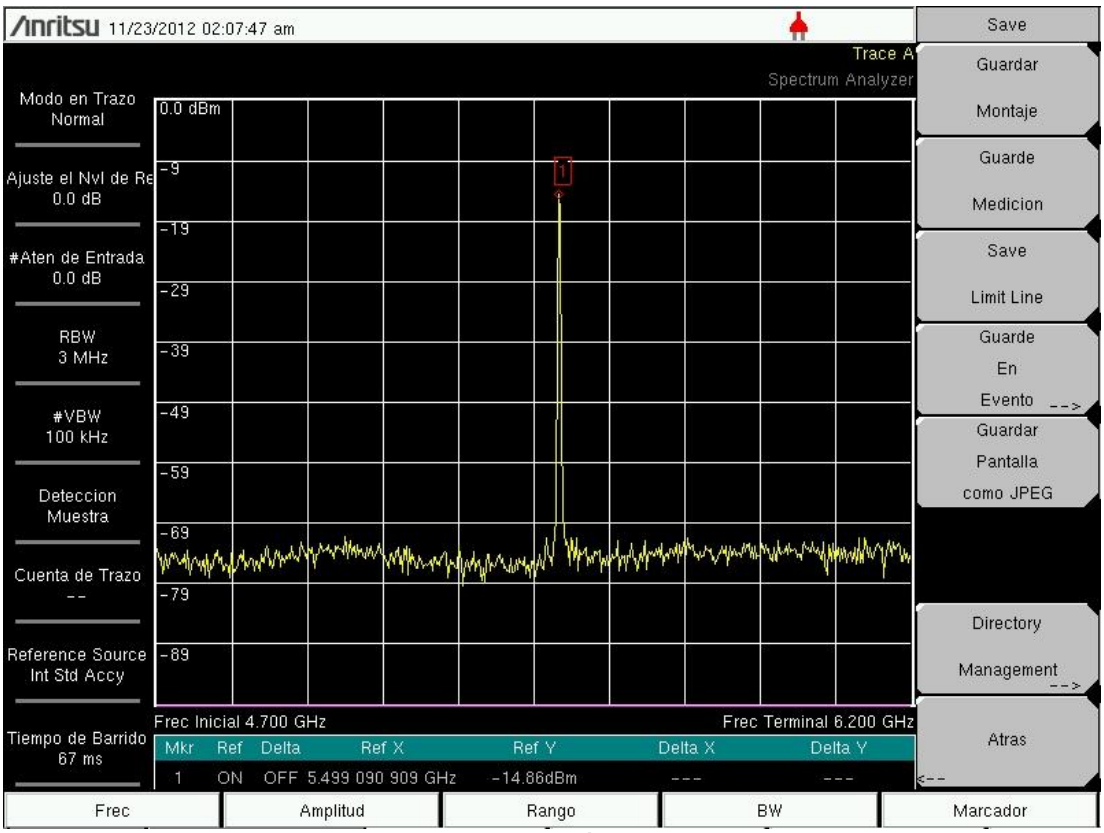


Fig 5.0.14: Prueba transmisión XCVR2450 @5.5GHz

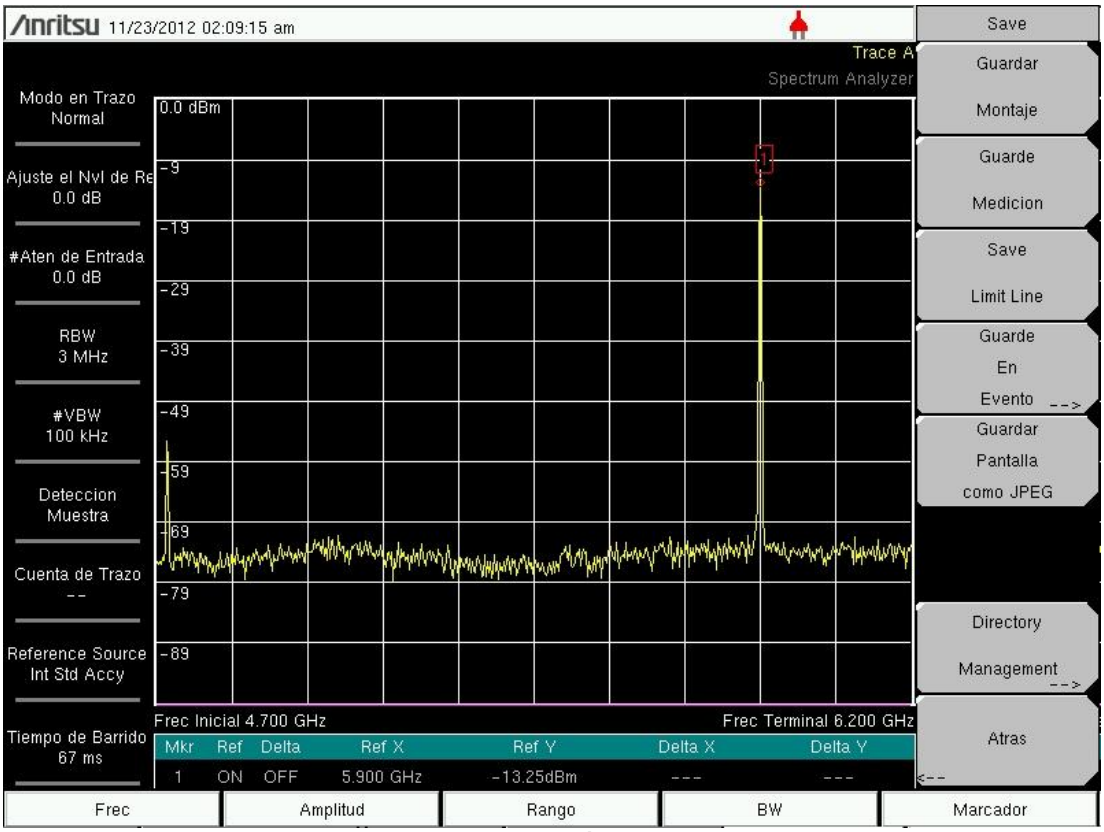


Fig 5.0.15: Prueba transmisión XCVR2450 @5.9GHz

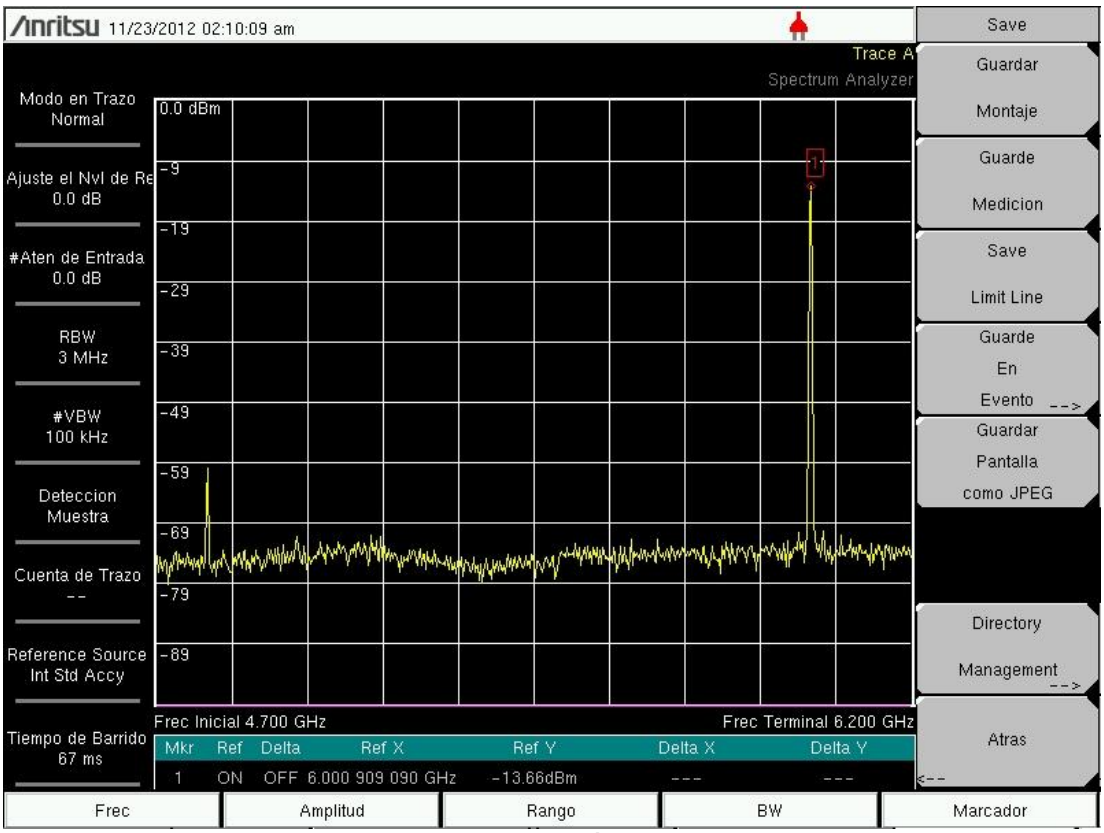


Fig 5.0.16: Prueba transmisión XCVR2450 @6GHz

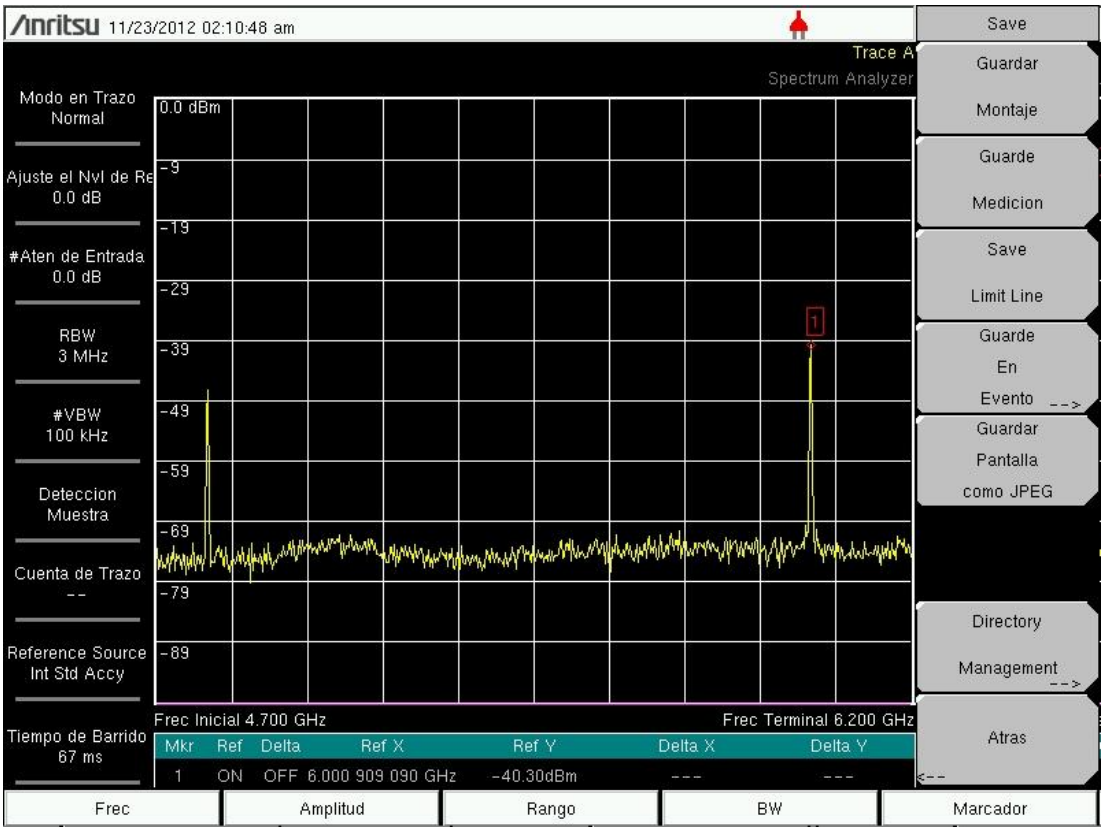


Fig 5.0.17:Prueba transmisión XCVR2450 @6.1GHz

Lo primero a mencionar es que pese a que el fabricante especifique que el rango de operación es de 2.4-2.5 y 4.9-5.9 GHz, vemos como a 6GHz se está transmitiendo con un nivel de potencia de unos 17 dBm.
Para poder obtener la potencia máxima de salida especificada por el fabricante, ha de establecerse como ganancia variable la máxima permitida en esta placa para la transmisión y la máxima amplitud de la señal
En cuanto a la imprecisión que presentan algunas de las frecuencias generadas (2.45GHz) viene dado por dos motivos, el primero de ellos dependerá del *span* tomado en el analizador de espectros (para reducir este efecto se reducirá el *span* utilizado). El segundo motivo es debido a la precisión de los sintetizadores de frecuencia utilizados tanto en la generación de la señal (*daughterboard*) como en el analizador de espectros.

5.2 Pruebas de recepción

Para la realización de esta prueba se hará uso del siguiente material:

- Generador de señal
- USRP N210 (*motherboard* + *daughterboard* (rx))
- Conectores tipo N macho y SMA macho
- Atenuador de 30 dB

Para llevar a cabo las pruebas de recepción se puede hacer uso de la interfaz uhd_fft directamente escribiendo en la terminal:

```
$ uhd_fft
```

Fig 5.0.18: Comando para caracterizar a un receptor como analizador de espectros desde la terminal

O bien se puede ejecutar el archivo uhd_fft.grc desde la interfaz GNU Radio companion (también conocida como grc). En este caso, se ha decidido crear el archivo prueba_rx.grc para realizar una primera toma de contacto con esta interfaz.
Al abrir este archivo nos aparecerá la siguiente pantalla:

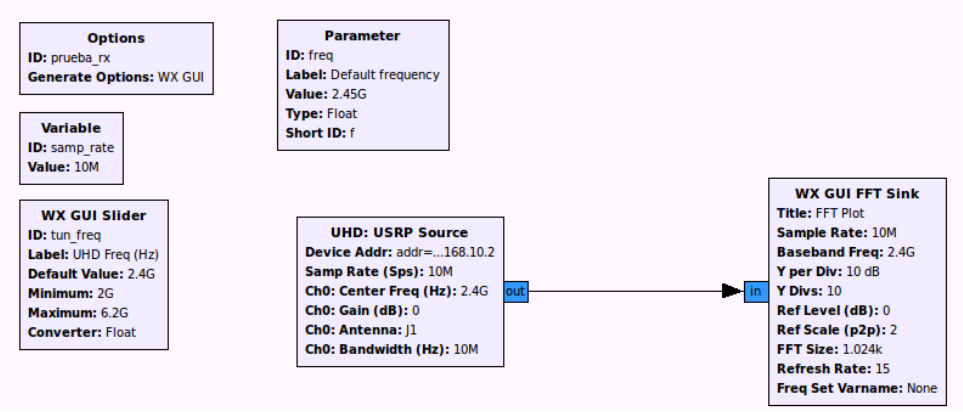


Fig 5.0.19: Diseño prueba de recepción

Al ejecutar este programa se generará un archivo prueba_rx.py que contendrá el código del programa en Python, código que se puede tanto visualizar como modificar.

El siguiente esquema representa la conexión física llevada a cabo:

Fig 5.0.20: Conexión física para recepción

5.2.1 RFX2400

Como potencia de salida del generador se ha tomado 10 dBm para todas las frecuencias, para proteger el dispositivo, cuya potencia máxima de entrada es -10dBm, se ha añadido un atenuador de 30 dB capaz de trabajar en todas las bandas de interés protegiendo así nuestro USRP con una banda de guarda de 10 dB (-20dBm de potencia recibida).
También hay que destacar que la señal generada ha sido conducida mediante un cable con conectores tipo N-sma hasta la entrada RX2 de la placa.

Los resultados obtenidos se muestran a continuación:

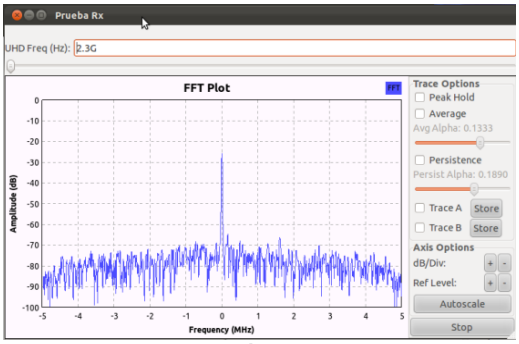


Fig 5.0.21: Prueba recepción RFX2400 @2.3GHz (RX2)

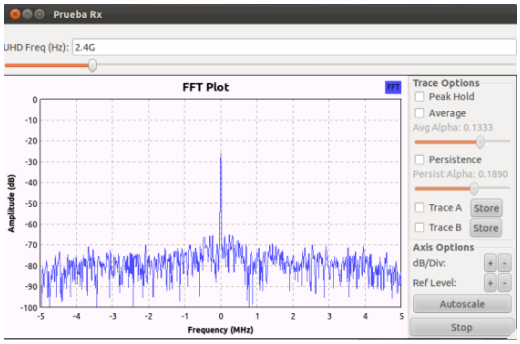


Fig 5.0.22: Prueba recepción RFX2400 @2.4GHz (RX2)

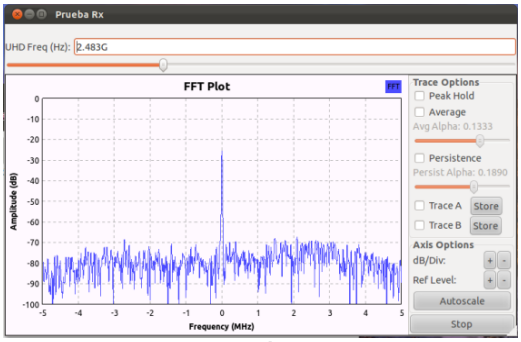


Fig 5.0.23: Prueba recepción RFX2400 @2.483GHz (RX2)

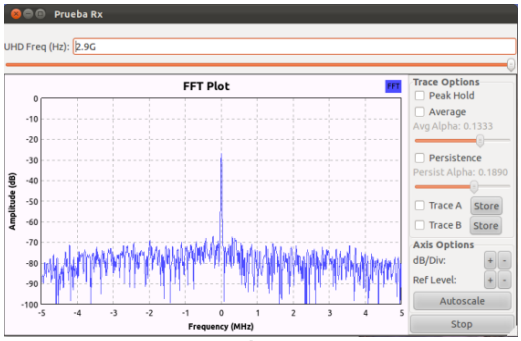


Fig 5.0.24: Prueba recepción RFX2400 @2.9GHz (RX2)

Como se puede apreciar en las imágenes, lo primero que se visualiza es que la magnitud de la señal recibida es adimensional, esto se debe a que los desarrolladores de GNU Radio no dan medidas absolutas puesto que estas varían según cada dispositivo. Por lo tanto al igual que ocurría a la hora de generar una señal (Figura 4.0.5), las medidas de su amplitud será en referencia al rango que presenta el ADC.
Otro aspecto destacable, es que en toda la banda la placa RFX2400 es capaz de actuar como receptor, esto se debe a que la señal ha sido conducida al puerto RX2. Si por el contrario, la señal generada hubiese entrado por el puerto TX/RX, se hubiera encontrado con el filtro, que al tratarse de un dispositivo recíproco, hubiese atenuado la señal tal y como muestran las siguientes imágenes:

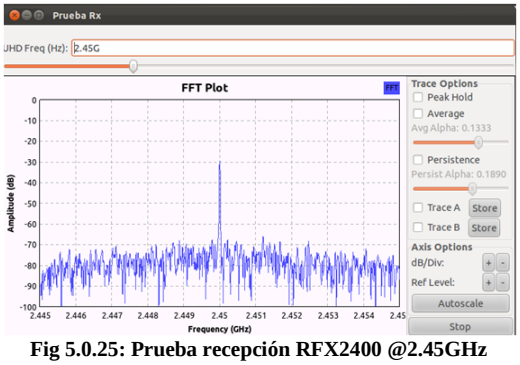


Fig 5.0.25: Prueba recepción RFX2400 @2.45GHz (TX/RX)

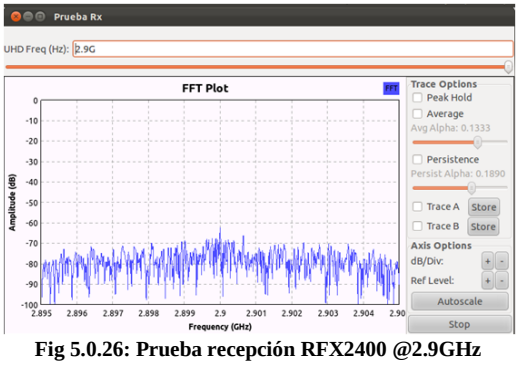


Fig 5.0.26: Prueba recepción RFX2400 @2.9GHz (TX/RX)

Como se puede apreciar en las imágenes mostradas, la misma señal es conducida ahora al puerto TX/RX. Fuera de la banda de operación la señal es filtrada, mientras que en el rango de operación la señal es atenuada por atravesar los elementos pasivos tales como el filtro y el conmutador extra. Nótese que como la señal recibida se puede representar

tanto en banda base como a la frecuencia RF. En realidad la señal representada es la que se está recibiendo en el ordenador, sin embargo, la herramienta representa la señal en las dos bandas.

5.1.2 XCVR2450

Nuevamente como potencia de salida del generador se ha tomado 10 dBm para todas las frecuencias, para proteger el dispositivo, cuya potencia máxima de entrada es de -10dBm, se ha añadido un atenuador de 30 dB capaz de trabajar en todas las bandas de interés protegiendo así el USRP con una banda de guarda de 10 dB.
También hay que destacar que la señal generada ha sido conducida mediante un cable con conectores N-sma hasta la entrada J1 de la placa.
Los resultados obtenidos se muestran a continuación:

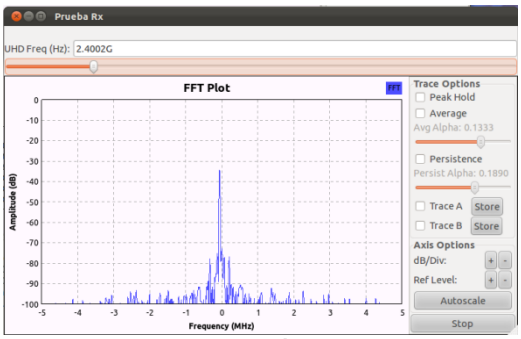


Fig 5.0.27: Prueba recepción XVCR2450 @2.4G

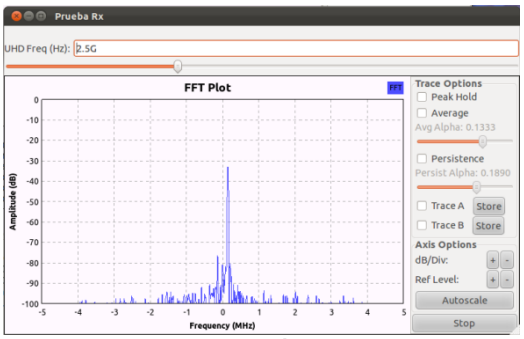


Fig 5.0.28: Prueba recepción XVCR2450 @2.5G

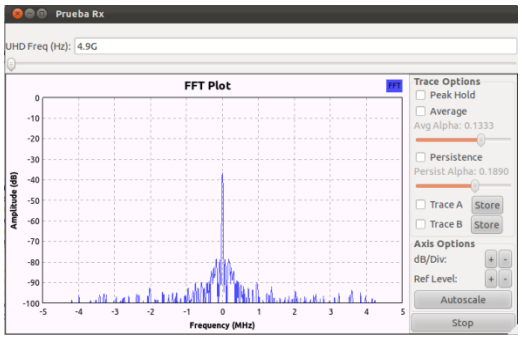


Fig 5.0.29: Prueba recepción XVCR2450 @4.9G

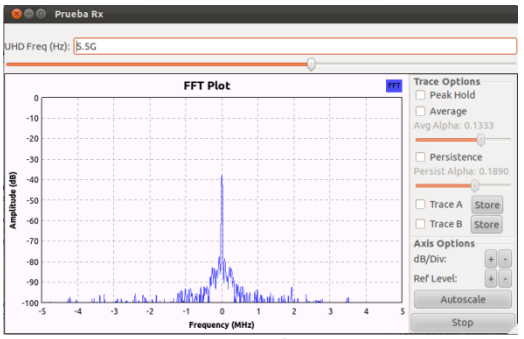


Fig 5.0.30: Prueba recepción XVCR2450 @5.5G

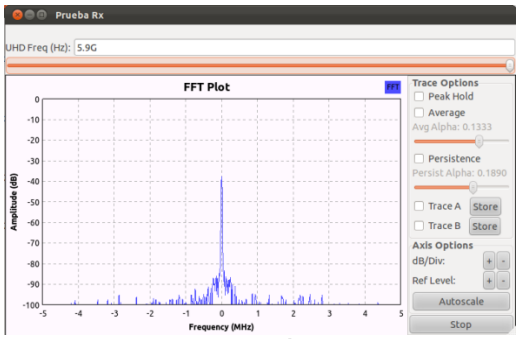


Fig 5.0.31: Prueba recepción XVCR245 @5.9G

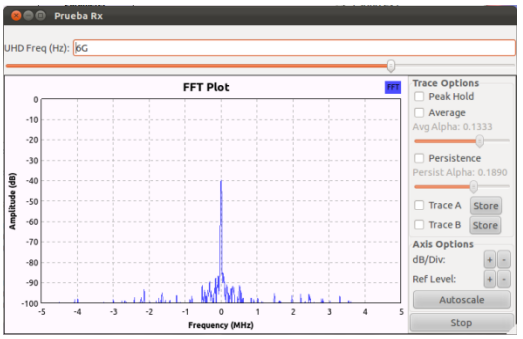


Fig 5.0.32: Prueba recepción XVCR245 @6G

Como se puede apreciar en la imagen, existe una desviación en la recepción de la señal de 2.4GHz debido a la sintetización de las frecuencias en los mezcladores. También se puede apreciar que las señales de la banda superior de frecuencia sufren una atenuación

extra con respecto a la banda inferior, esto se debe principalmente a que las primeras atraviesan un filtro extra el cual atenúa la señal.

5.3 Pruebas de transmisión y recepción

Para la realización de esta prueba se tendrá que hacer uso del siguiente material:

- USRP N210 (*motherboard* + *daughterboard* (tx)).
- USRP N210 (*motherboard* + *daughterboard* (rx)).
- Conectores tipo N macho y SMA macho.
- Atenuador de 30 dB.
- Adaptador N-SMA.

Para llevar a cabo las pruebas de transmisión y recepción se ha usado la interfaz uhd_siggen_gui para caracterizar al transmisor y la interfaz uhd_fft para caracterizar al receptor escribiendo en cada ordenador respectivamente:

El siguiente esquema representa la conexión física llevada a cabo:

Fig 5.0.33: Conexión física para transmisión y recepción

Hay que mencionar tal y como se ha visto que tanto en la generación como en la recepción de señales la herramienta software solo provee medidas relativas al rango de los conversores. Por lo tanto, se ha de tener especial cuidado a la hora de comunicar los dispositivos entre sí para no sobrepasar la potencia máxima de entrada.

5.3.1 RFX2400

Como ya se había mencionado anteriormente, la placa RFX2400 es una placa transceptora. Esto nos ha permitido caracterizar la placa como transmisor y como receptor tal y como se ha mostrado en los apartados anteriores, para la comunicación directa entre USRPs harán falta dos ordenadores que se encargarán de caracterizar a uno de ellos como transmisor y al otro como receptor. La señal se emitirá por el puerto TX/RX del primer USRP y se recibirá por el puerto RX2 del segundo, obteniéndose así los siguientes resultados:

Ordenador 1: Transmisor

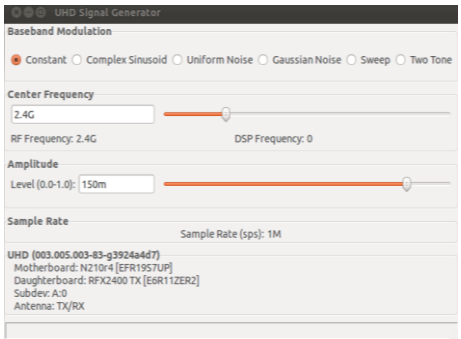


Fig 5.0.34: Tx RFX2400 @2.4GHz

Ordenador 2: Receptor

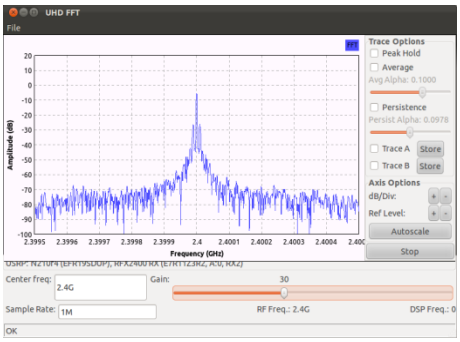


Fig 5.0.35: Rx RFX2400 @2.4GHz

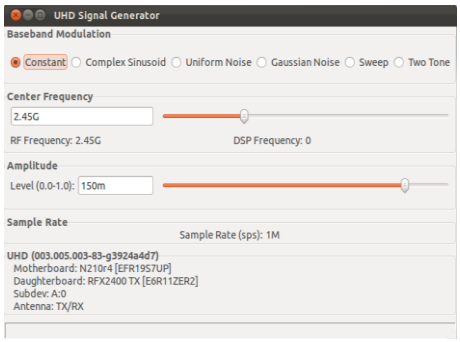


Fig 5.0.36: Tx RFX2400 @2.45GHz

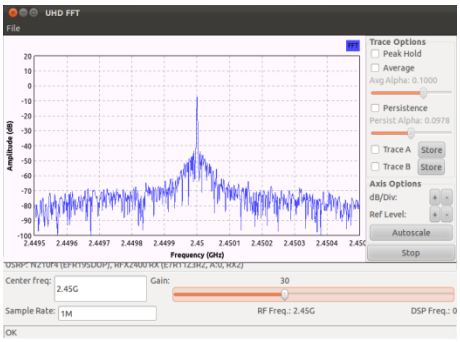


Fig 5.0.37: Rx RFX2400 @2.45GHz

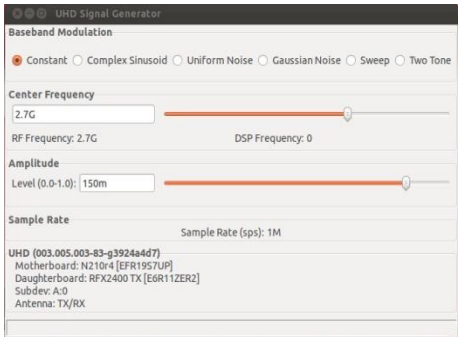


Fig 5.0.38: Tx RFX2400 @2.7GHz

Fig 5.0.39: Rx RFX2400 @2.7GHz

La columna izquierda representa la interfaz uhd_siggen_gui para la placa RFX2400, como se puede apreciar, en ella se puede elegir el tipo de señal que se desea enviar así como su frecuencia y su amplitud. Por otro lado en el ordenador caracterizado como receptor se dispondrá de la interfaz uhd_fft, en la cual, se podrá elegir la frecuencia la ganancia en recepción y el *sample rate*.

Que los niveles de la señal recibida sean entorno a 10dB superiores que en los resultados obtenidos en las pruebas de recepción (Fig: 5.21-5.25) no significa que como potencia de entrada se esté poniendo el nivel de potencia máximo permitido, si no que se están tomando diferentes niveles de ganancia en el sistema.

También se puede apreciar como cuando se transmite la señal a una frecuencia superior a la de corte del filtro situado en el camino del transmisor (2.4-2.483 GHz) la señal se ve filtrada.

Tal y como se contaba en el capítulo 3, la placa transceptora RFX2400 soporta el modo de operación *full-duplex* al disponer de caminos totalmente independientes para el transmisor y el receptor (salvo el puerto TX/RX que comparte parte de la misma cadena). Antes de documentar este modo de operación, se propone el estudio del aislamiento entre el camino del transmisor y del receptor para ello se empleará el archivo generado isolation.grc.

La siguiente imagen muestra el diseño realizado para medir el aislamiento entre el transmisor y el receptor:

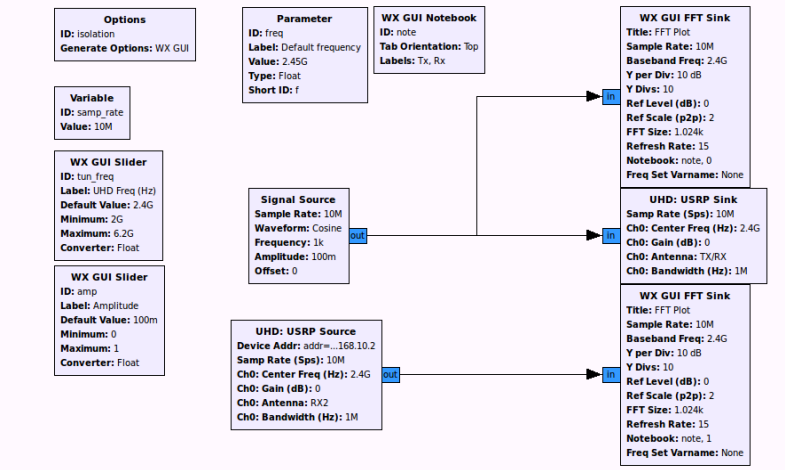


Fig 5.0.40: Diseño para la prueba isolation

Ejecutando este programa se obtienen los siguientes resultados:

Señal generada:

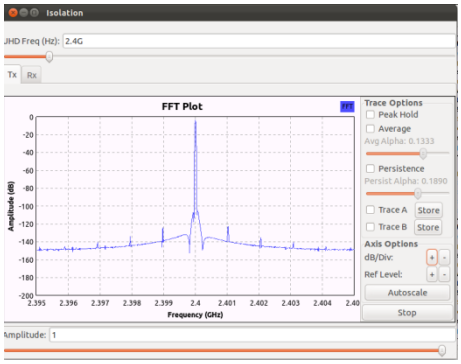


Fig 5.0.41: Tx isolation RFX2400 @2.4GHz

Señal recibida:

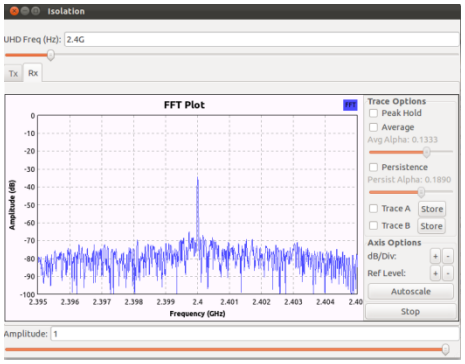


Fig 5.0.42: Rx isolation RFX2400 @2.4GHZ

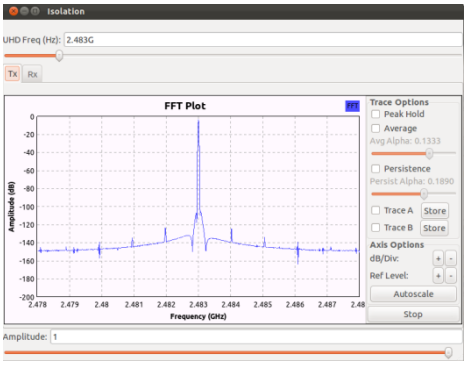


Fig 5.0.43: Tx isolation RFX2400 @2.483GHz

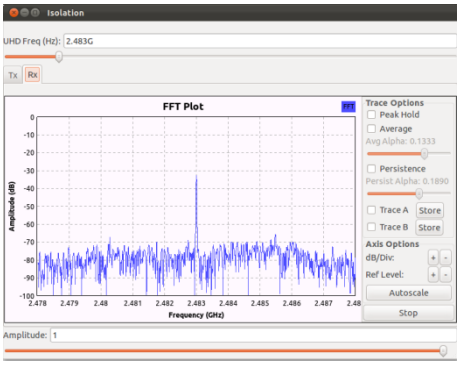


Fig 5.0.44: Rx isolation RFX2400 @2.483GHz

La prueba para medir el aislamiento se basa en la capacidad de la *daughterboard* RFX2400 de operar en modo *full-duplex*. Por lo tanto mientras se transmite a una determinada frecuencia, se configura a la placa también como receptor que opere en la misma banda.
Como se puede comprobar, al radiar una señal, parte de esta se está colando al receptor haciendo obligatorio el uso de distintas frecuencias de transmisión y recepción.

Una vez mostrado el por qué la frecuencia en el enlace ascendente tiene que ser diferente a la utilizada en el enlace descendente, se procede a probar el modo operación *full-duplex*. Para ello se decide crear el fichero prueba_tx_rx.grc el cual permitirá transmitir y recibir al mismo tiempo.

La siguiente imagen muestra el diagrama del enlance:

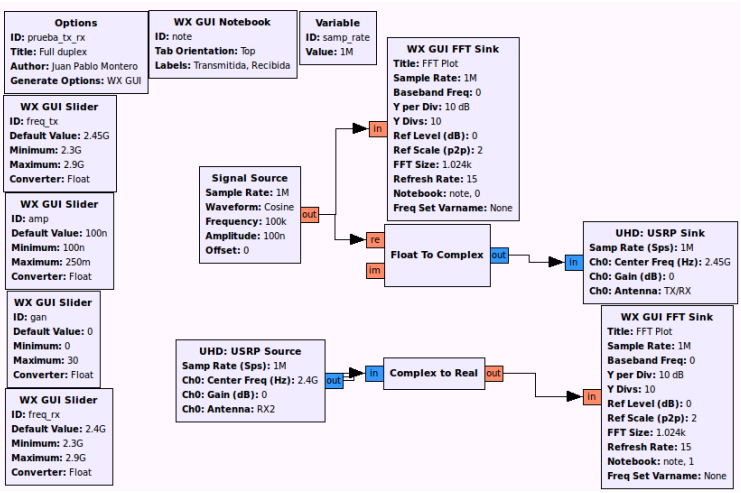


Fig 5.0.45: Diseño para la prueba full-duplex

La imagen anterior muestra el flujo que seguirá la señal hasta llegar desde el ordenador al USRP y viceversa dependiendo de cuál sea el caso. Hay que recalcar que el diagrama variará de un ordenador en cuanto a la frecuencia del coseno (50kHz y 100kHz) y que la frecuencia de transmisión para uno será la de recepción para el otro y viceversa.
Teniendo en cuenta el aspecto mencionado se obtienen los siguientes resultados:

Ordenador 1:

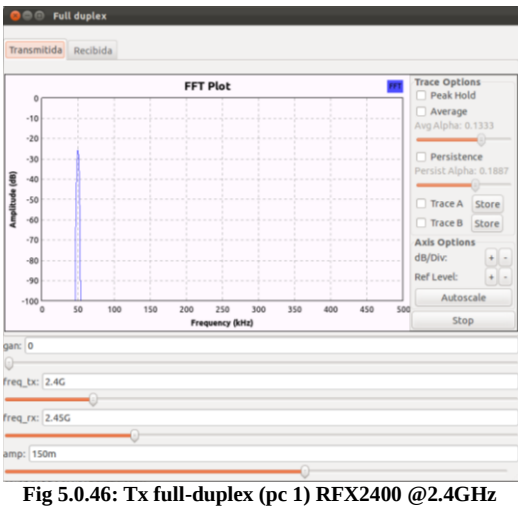


Fig 5.0.46: Tx full-duplex (pc 1) RFX2400 @2.4GHz

Ordenador 2:

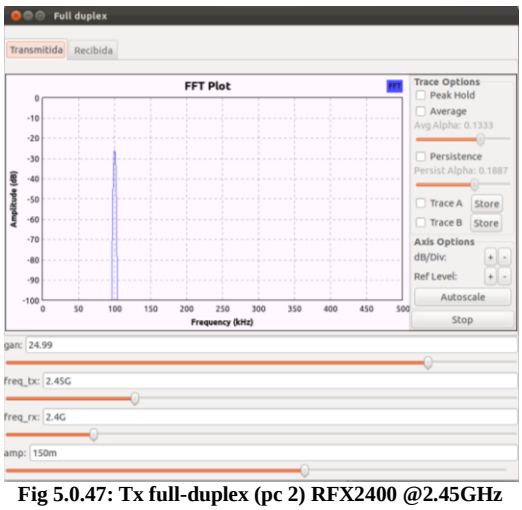


Fig 5.0.47: Tx full-duplex (pc 2) RFX2400 @2.45GHz

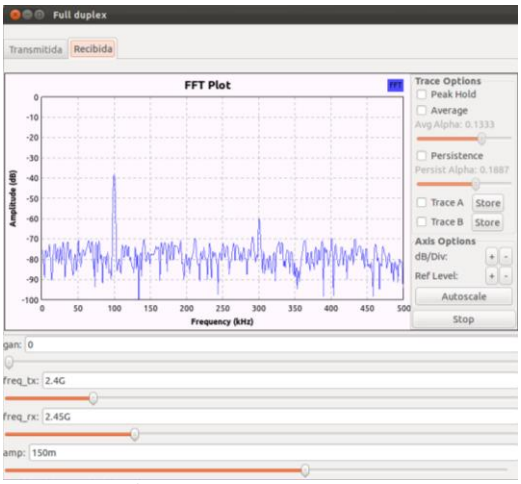


Fig 5.0.48: Rx full-duplex (pc 1) RFX2400 @2.45GHz

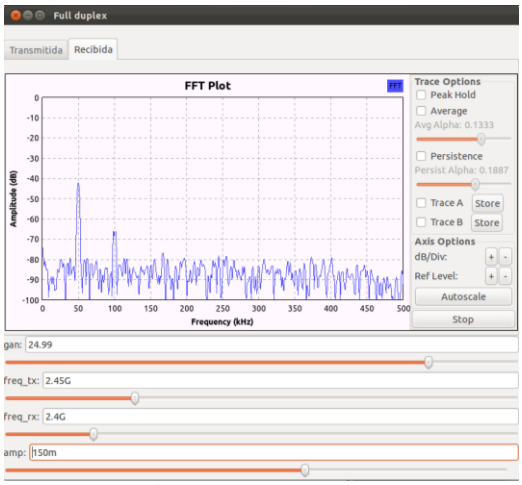


Fig 5.0.49: Rx full-duplex (pc 2) RFX2400 @2.4GHz

En el ordenador 1, se genera un coseno con frecuencia 50kHz y es trasmitido hacia el otro USRP con una frecuencia de 2.4GHz, mientras que el ordenador 2 genera un coseno a una frecuencia de 100 kHz y lo trasmite a 2.45 GHz. Lo más destacable en esta prueba es sin duda, la diferencia en cuanto a niveles de potencia de las señales recibidas. Esto se debe a que algún elemento del dispositivo (*daughterboard*) este dañado puesto que se verificó que no fuese problema del diseño ni de los elementos externos utilizados.
También se puede apreciar la figura 5.0.49 que se está colando parte de la señal generada.

5.3.1 XCVR2450

Como ya se había mencionado anteriormente, la placa XCVR2450 es una placa transceptora. Esto nos ha permitido caracterizar la placa como transmisor y como receptor tal y como se ha mostrado en los apartados anteriores. Para la comunicación directa entre USRP, harán falta dos ordenadores que se encargarán de caracterizar a uno de ellos como transmisor y al otro como receptor. La señal se emitirá por el puerto J1 del primer USRP y se recibirá por el puerto J2 del segundo, aunque cualquier configuración hubiera valido puesto que los puertos J1 y J2 comparten el mismo camino. Obteniéndose así los siguientes resultados:

Ordenador 1 (Transmisor):

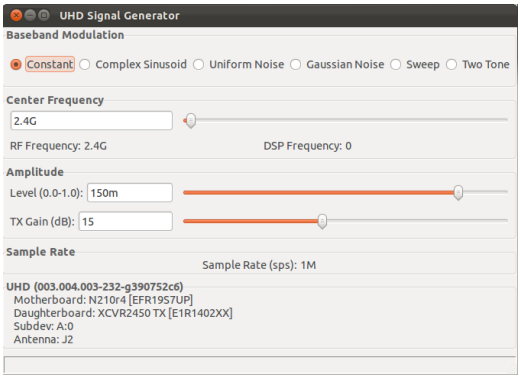


Fig 5.0.50: Tx XCVR2450 @2.4GHz

Ordenador 2 (Receptor):

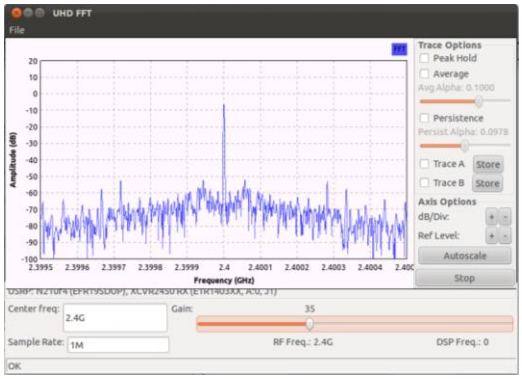


Fig 5.0.51: Rx XCVR2450 @2.4GHz

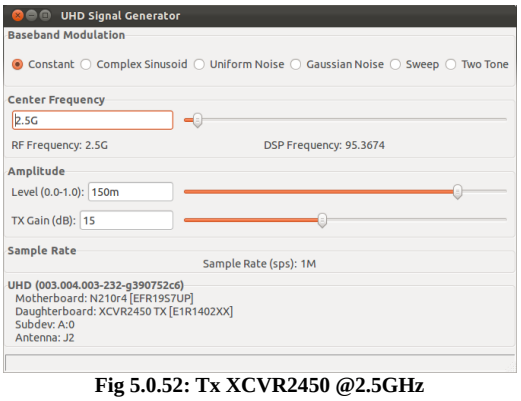


Fig 5.0.52: Tx XCVR2450 @2.5GHz

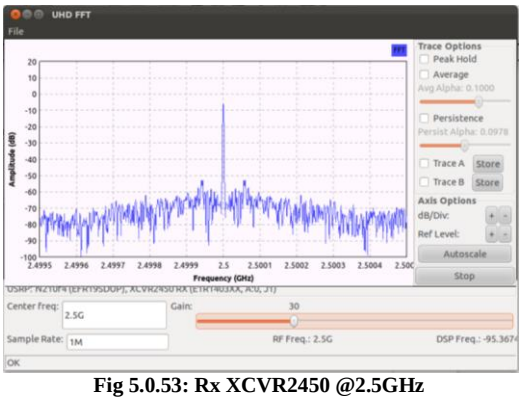


Fig 5.0.53: Rx XCVR2450 @2.5GHz

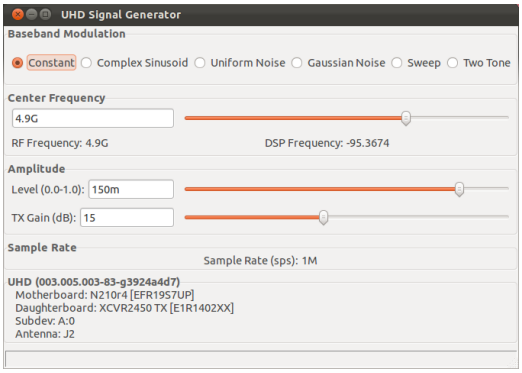


Fig 5.0.54: Tx XCVR2450 @4.9GHz

Fig 5.0.55: Rx XCVR2450 @4.9GHz

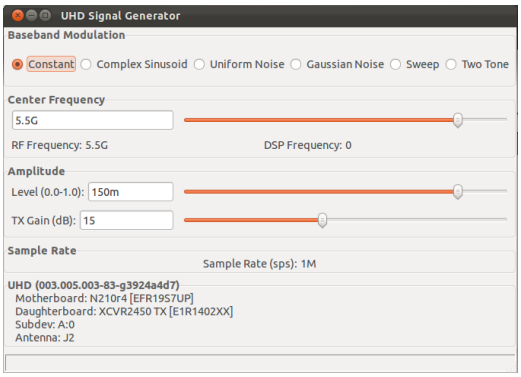


Fig 5.0.56: Tx XCVR2450 @5.5GHz

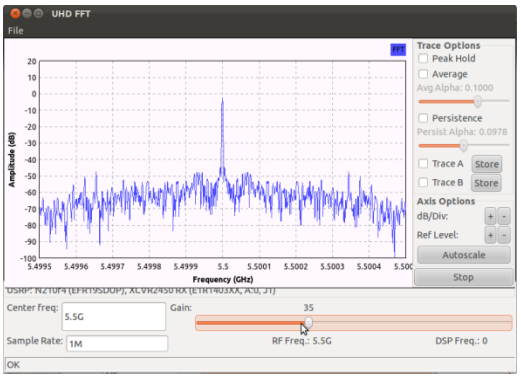


Fig 5.0.57: Rx XCVR2450 @5.5GHz

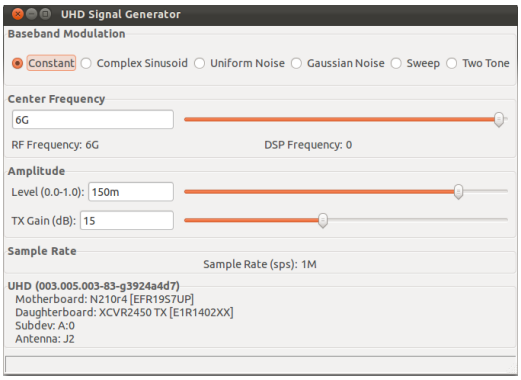


Fig 5.0.58: Tx XCVR2450 @6GHz

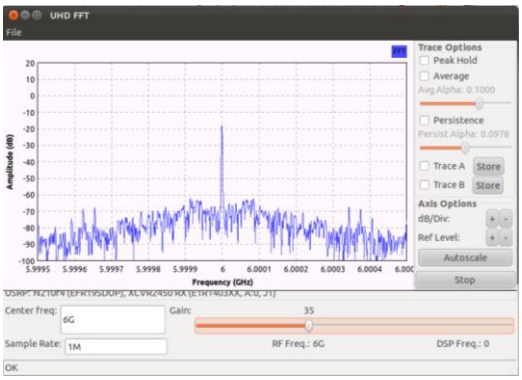


Fig 5.0.59: Rx XCVR2450 @6GHz

La columna izquierda representa la interfaz uhd_siggen_gui para la placa XCVR2450. Como se puede apreciar, en la interfaz se puede elegir el tipo de señal que se desea enviar así como su frecuencia, su amplitud y a diferencia de la placa RFX2400, la

ganancia variable en transmisión. Por otro lado en el ordenador caracterizado como receptor se dispondrá de la interfaz uhd_fft, en la cual, se podrá elegir la frecuencia la ganancia en recepción y el *sample rate*.
Como anteriormente se mencionaba los niveles de señal en la banda superior (4.9-6 GHz) sufren una mayor atenuación por atravesar un filtro el cuál, aunque deje pasar la señal, la atenúa. Sin embargo, se puede apreciar como la señal a 5.5GHz (Fig 5.0.57) llega al receptor con niveles muy similares a los que presentan las señales en la banda de operación inferior, esto se debe a que la respuesta de los elementos tanto activos (amplificadores) como pasivos (filtros, conmutadores) no es tan plana como se desearía y no se comporta de la misma manera en todas las frecuencias.

6

Experimentos y resultados en entorno de simulación

En este capítulo se mostrará un conjunto de diseños generados con la herramienta GNU Radio companion. Estos experimentos se llevarán a cabo en un entorno de simulación, sin hacer uso de las plataformas hardware disponible.
En primer lugar, se realizarán diseños de enlaces que utilicen modulaciones digitales para mostrar como una función que históricamente recaía en la parte hardware, ahora es implementado vía software. Posteriormente se abordará el tema de los sincronismos y finalmente, se mostrará un método para estimar la BER[41]
A la hora de diseñar sistemas, es importante conocer los bloques de procesado disponibles en la herramienta, para ello GNU Radio ha creado una API[19] donde se podrá consultar información acerca de los módulos construidos en C++.

6.1 Diseño de enlaces

Este apartado tiene por objetivo mostrar como son llevadas a cabo las modulaciones vía software. Éstas, se realizarán en banda base y posteriormente (en función del diseño) se trasladan en frecuencia. En este caso, al tratarse de una simulación, este paso será obviado, centrándose únicamente en el proceso de modulación.

Con motivo de facilitar la comprensión de dicho apartado, todos los diseños han sido realizados a partir de la interfaz gráfica disponible (GNU Radio companion), yendo desde lo particular (bloques que conforman el modulador específico) a uno más general, donde se utilizarán bloques que provee la herramienta.

6.1.1 Sistema M-QAM

En este apartado se abordará como se puede diseñar un sistema que utilice una modulación M-QAM.
Para llevar a cabo el diseño, se utilizará como fuente de información una señal de audio cuya fuente es la tarjeta de sonido que envía la información que le llega por el micrófono.
Por último, se utilizará como *sink* un sumidero gráfico que se encargará de representar el espectro de la señal que le llega.

[41] BER: *Bit Error Rate*

6.1.1.1 Transmisor M-QAM

La siguiente figura muestra el diseño realizado para un modulador M-QAM, para ello, se modificará el parámetro bits en función del índice de modulación que se desee transmitir. En concreto, la siguiente figura se corresponde con un modulador 256-QAM:

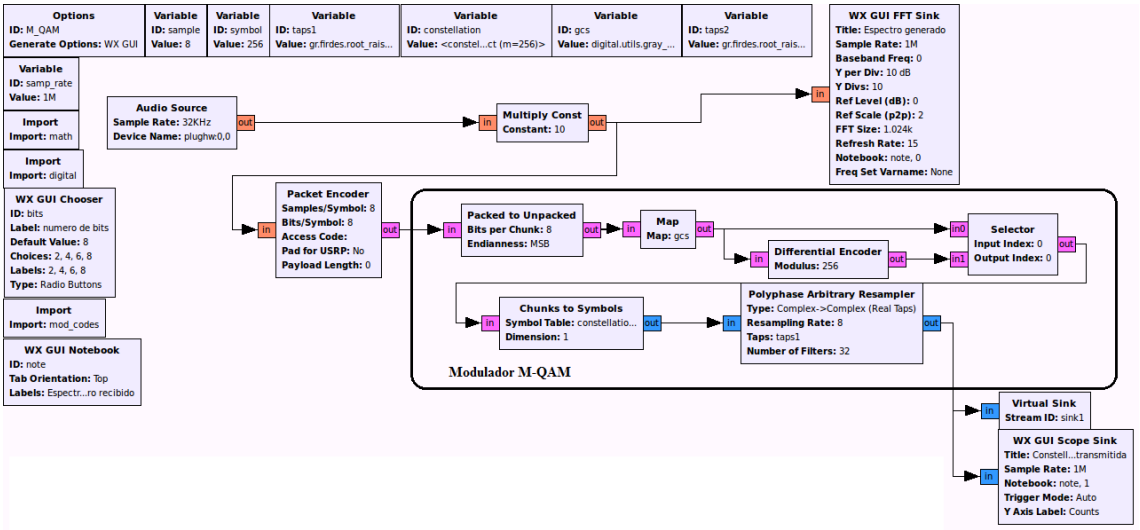


Fig 6.0.1: Diseño modulador M-QAM

En la imagen se pueden apreciar numerosos bloques así como numerosos parámetros de configuración, se entrará en detalle en los bloques que conforman el modulador M-QAM (el diseñado se ha particularizado en un modulador 256-QAM), así como de las variables externas de las que hace uso.

En un principio se explicarán las variables externas al diagrama que aparecen. Éstas están situadas en el margen izquierdo y superior de la figura.
De izquierda a derecha, aparece un bloque en el que se indica el nombre de la aplicación así como si se utilizará módulos gráficos: Wx, Qt o por el contrario ninguno. Además, este bloque permite añadir el nombre del autor del diagrama, nombre que aparecerá en el encabezado del código generado.
El parámetro *sample* del diagrama indicará el número de muestras que se utilizarán por símbolo. *Symbol* servirá para indicar al diseño el número de símbolos que habrá en la modulación, este parámetro se calcula como:

$$Simbolos = 2^{bits}$$

Ec 6.0.1: Definición número de símbolos

El parámetro taps1 se utiliza para describir el filtro conformador de pulso que se va a utilizar, en este caso se trata de un filtro en coseno alzado:

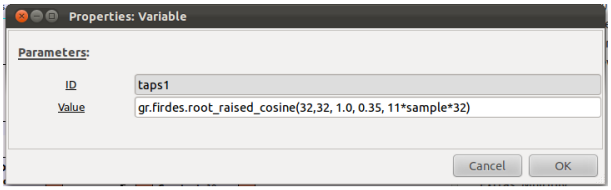


Fig 6.0.2: Parámetro filtro coseno alzado tx

Donde el primer parámetro indica la ganancia, el segundo es la *sample_rate* utilizada por el filtro (posteriormente se hablará de este parámetro), el siguiente es el *symbol rate* y el último el número de coeficientes del filtro.

El siguiente elemento define la constelación que se va a utilizar, en este caso se ha generado partiendo del código existente en el fichero qam.py:

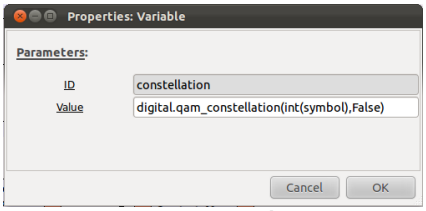


Fig 6.0.3: Constelación QAM

GNU Radio provee ficheros auxiliares de tal forma que facilita la creación de ciertas constelaciones (QAM Y PSK) sin necesitar la invocación directa (internamente el fichero lo hace) de los métodos definidos en el fichero constellation.h.
Los parámetros necesarios para generar una modulación son por una parte los puntos de la constelación y por otra la lista de símbolos a mapear, encargándose este bloque del primero (en verdad genera un objeto constelación donde están definidos una lista de atributos y de funciones, siendo uno de ellos los puntos que definen a esta). De ser necesario, en el proceso de creación de la constelación se puede llevar a cabo la codificación en código Gray y Diferencial, siendo posible también su generación mediante bloques externos.
En general, el objeto constelación puede ser generado a través de distintas clases que derivan de la clase padre o base con la intención de explotar las características geométricas da cada constelación a la hora de llevar a cabo la demodulación.
El método invocado en la figura, invoca a su vez a la clase digital_constellation_rect situada en el archivo digital_constellation.cc encargada de crear el objeto constellation.
Para poder crear la constelación desde el archivo qam.py, se le ha de indicar el índice de modulación y si se trata de una modulación diferencial así como si se desea codificación Gray, aunque por defecto no se realiza.

El siguiente parámetro se utilizará para el cálculo del código gray:

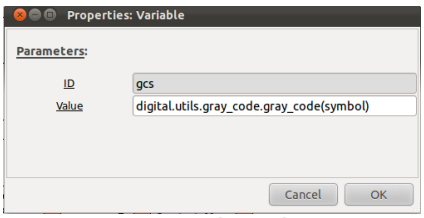


Fig 6.0.4: Generación de código Gray

Para ello habrá que indicar el índice de modulación de la constelación.

Por último, el parámetro taps2 es utilizado para definir el filtro conformador de pulsos que se utilizará en el demodulador. Por ello, se hablará de él posteriormente.

De la parte superior a la inferior, el primer bloque que aparece es el que indica la *sample rate* del sistema.

Las sentencias de *import*, permitirán invocar a métodos situados en otros módulos de GNU Radio o incluso a otras librerías de python.

En cuanto al parámetro bits, será el utilizado para modificar el índice de modulación del sistema. En este caso se permite emplear hasta 8 bits, pero solo aquellos que obtengan un índice de modulación que sea potencia de 4. Como máximo, por limitaciones existentes, solo se puede generar constelaciones hasta 1024-QAM.

El parámetro notebook sirve para poder representar diferentes sumideros gráficos en diferentes pestañas, para ello se le ha de indicar el nombre de las pestañas que se van a incorporar.

Una vez se ha explicado los parámetros o variables que se van a utilizar en el diseño, se procede a explicar el flujo del propio diseño

Inicialmente se encuentra la fuente de audio, encargada de producir muestras a una tasa de 32kHz, cada una de estas muestras requieren 4 bytes para su representación. Las muestras que salen de la fuente de audio son conducidas a un multiplicador el cuál se encarga de aumentar el nivel de la señal producida. Seguidamente se atraviesa el codificador, encargado de transformar las muestras a byte. Estos bytes estarán formados por un símbolo puesto que se necesitan 8 bits para su representación, este proceso variará en función del nivel de modulación requerido empaquetando tantos símbolos como se pueda en cada byte, la siguiente figura muestra las propiedades de este bloque:

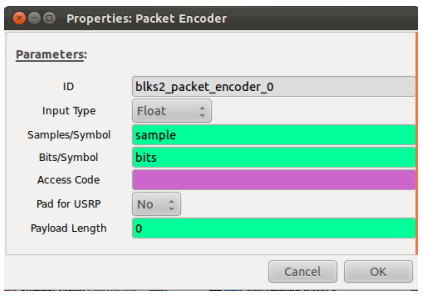


Fig 6.0.5: Packet encoder

El objetivo principal de este bloque es la sincronización de paquete de la cual se hablará en el apartado 6.2.2.1. A este bloque se le ha de indicar el número de muestras a emplear por símbolo, en este caso 8 muestras (*sample*). También se le tendrá que indicar el número de bits que se van a emplear en la modulación.

El primer bloque en la cadena del modulador se encarga de desempaquetar estos bytes, las propiedades de configuración de este bloque se muestran a continuación:

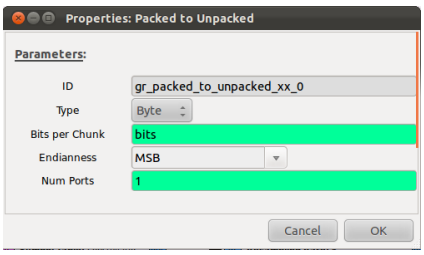


Fig 6.0.6: Packed to Unpacked

Desempaquetar los bytes significa colocar los bits que se han empleado para definir cada símbolo en un nuevo byte en los bits más significativos (MSB[42]). Como en este caso se emplea 8 bits para definir el símbolo, no introducirá ningún cambio. Sin embargo, es un bloque necesario para poder hacer un modulador M-QAM genérico.

El siguiente bloque en la cadena del modulador es el encargado de mapear estos bits en código Gray, la configuración del bloque se muestra a continuación:

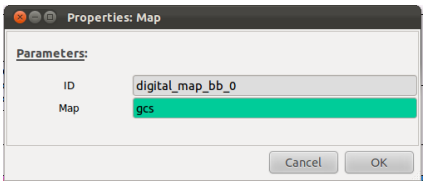


Fig 6.0.7: Gray code

Donde el parámetro que se le indica (almacenado en la variable previamente definida gcs) es el código Gray de tanta longitud como símbolos existan.

Dependiendo de si se elija realizar una codificación diferencial o no, se atravesará el bloque *Differential Encoder* encargado de realizar dicha codificación:

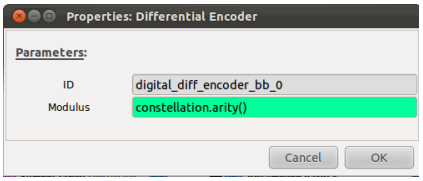


Fig 6.0.8: Differential Encoder

Cuyo único parámetro es la longitud del abecedario de símbolos.

Para determinar si se desea una modulación diferencial o no, se emplea un bloque selector, elemento que sacará por la salida la entrada que se elija.

Llegados a este punto, la información disponible sigue en byte como tipo de dato, el bloque que se encarga de mapear este flujo de bytes en muestras complejas (fase y cuadratura) es el *Chunks To Symbols*:

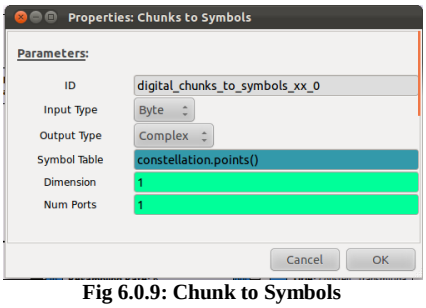


Fig 6.0.9: Chunk to Symbols

[42] MSB: Most Significant Bit

Como previamente se ha introducido, para llevar a cabo una modulación se tenía dos requisitos:

- Los puntos que definen la constelación
- La lista de símbolos

Este bloque, es el que lleva a cabo la generación en sí de la modulación (en general muchas de las modulaciones pueden ser creadas a partir del bloque *packed_to_unpucked* seguidas de *chunk_to_symbols*). Este bloque recibe una cadena de bytes desempaquetados (lista de símbolos) y se encarga de proporcionar una cadena de muestras complejas. Para ello se realiza un mapeo 1 a 1 de los símbolos de entrada a puntos de constelación. En este caso al realizarse el mapeo 1 a 1 de los símbolos el parámetro que define la dimensión de la LUT[43] ha de permanecer fijado a 1. En caso de querer realizar demodulaciones basadas en *soft decisión* este parámetro tendrá que modificarse.

En este momento, ya se dispone de la señal M-QAM en banda base. Por motivos de reducción de la Interferencia entre Símbolos (ISI) se introduce un bloque más, el cual llevara a cabo la tarea de conformar el pulso mediante un filtro en coseno alzado así como de ajustar la tasa de *sample rate* acorde a la que tiene establecida el periférico.

La siguiente figura muestra las propiedades del bloque mencionado:

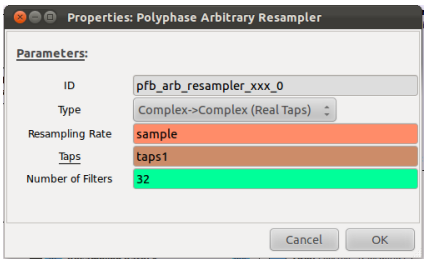


Fig 6.0.10: Polyphase Arbitrary Resampler

Este bloque, además de ser el encargado de conformar el pulso, será el que alinee las las diferentes tasas del sistema en función de las muestras que se han definido por símbolo. Será el encargado por lo tanto de llevar a cabo la interpolación en diseños que operen con plataformas hardware. Anteriormente, en otras versiones de GNU Radio, este bloque no existía y se debía especificar el factor de diezmado y/o interpolación que se requería para alinear las tasas de muestreo.
El número de filtros debe estar alineado con el parámetro *sample rate* impuesto en el dimensionamiento del filtro RRC.

Finalmente se emplaza el sumidero en el diseño, en este caso al estar en un entorno de simulación se coloca un sumidero virtual, cuya función es trasladar todos los datos de entrada a una fuente virtual (la cuál simulará ser la fuente de datos del receptor).

[43] LUT: Look-up table

6.1.1.2 Receptor M-QAM

La siguiente figura muestra el diseño realizado para un demodulador M-QAM. Para ello, se modificará el parámetro bits en función de la modulación transmitida anteriormente. En concreto, la siguiente figura representa a un demodulador 256-QAM:

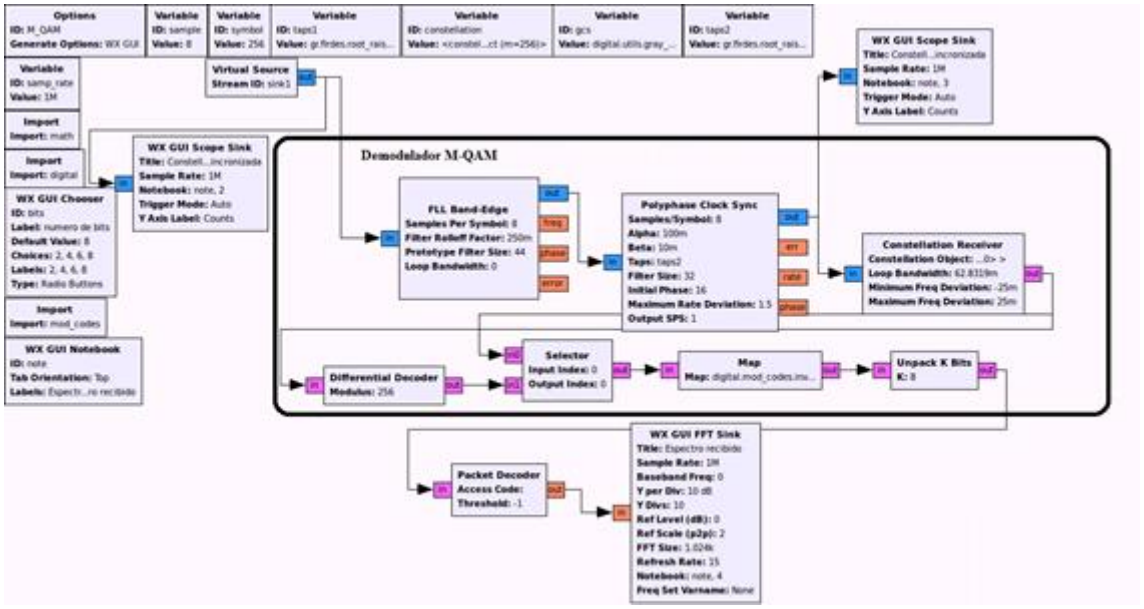


Fig 6.0.11: Diseño demodulador M-QAM

Tal y como muestra la figura, la mayoría de las variables utilizadas en el diseño coinciden con las del transmisor, esto es debido a que en entorno de simulación el transmisor y el receptor se diseñan en un mismo archivo, además de la reutilización de mucha de ellas.
El parámetro taps2 muestra la configuración realizada para la caracterización del filtro adaptado:

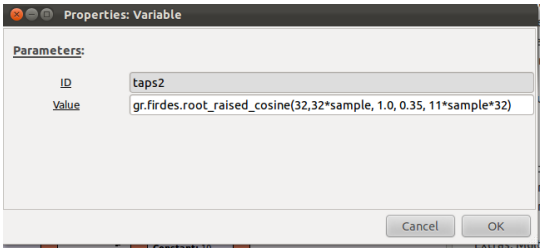


Fig 6.0.12: Parámetro filtro coseno alzado rx

En este caso, el parámetro *sample rate* del filtro difiere del anterior filtro en coseno alzado por el factor número de muestras, el motivo de ello, se explicará más tarde.
Inicialmente, en el flujo del receptor se encuentra una fuente virtual. Ésta, proporcionará como datos aquellos que lleguen al sumidero virtual (con mismo identificador) previamente definido. Es decir, la señal modulada y con pulsos conformados en banda base.

Los datos que provienen de la fuente virtual son conducidos al demodulador, más en concreto a un bloque que se encarga de realizar el compensado de frecuencia:

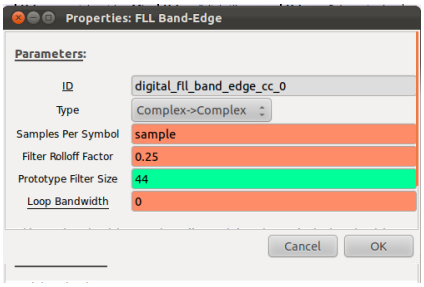


Fig 6.0.13: FLL Band-Edge

Los parámetros que se le indican son el número de muestras por símbolo, la longitud del banco de filtros, el ancho de banda del bucle y el *rolloff* del filtro, los efectos de este bloque se mostrarán en el apartado 6.2.2.2.1.

Una vez se atraviese el bloque encargado del compensado en frecuencia, se encuentra otro bloque que realiza el alineamiento en los tiempos de muestreo (transmisor y receptor). Es en este mismo bloque donde aparece el filtro adaptado (*root raised cosine*) encargado de deshacer los efectos del filtro conformador de pulsos del transmisor:

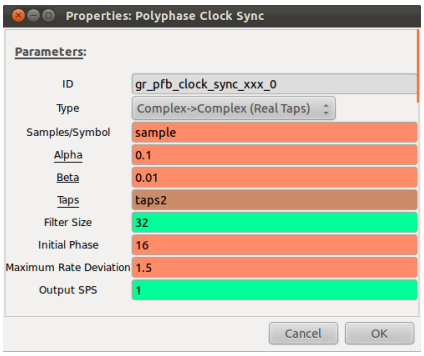


Fig 6.0.14: Polyphase Clock Sync

Además de realizar las tareas ya mencionadas, su salida transforma no en cuanto a tipo (siguen siendo muestras complejas), si no en el número de muestras que definen cada símbolo (*output sample per symbol*). A la entrada de este bloque se emplean el número de muestras definidas por la variable sample (8 en este diseño) y la salida de este, solo se empleará una muestra por símbolo. Es por ello por lo que se había introducido el factor *sample* en el *sample rate* del filtro adaptado.

Llegados a este punto, los siguientes bloque se encargarán de realizar la función inversa a la realizada en el *chunk to symbols*, por consiguiente el siguiente bloque en la cadena es el *costellation receiver,* al que le llega un cadena de muestras complejas y las mapeará (demodulará) ofreciendo a su salida una cadena de bytes:

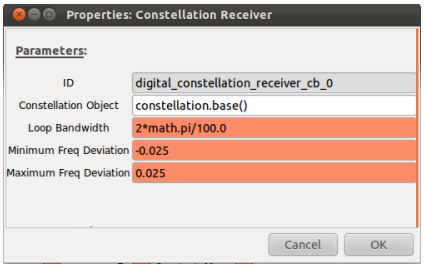


Fig 6.0.15: Constellation Receiver

Este bloque es el encargado de realizar el mapeo de muestras complejas a símbolos, para ello, se emplea el método *decision_maker* definido en el objeto de la constelación. En este caso, el mapeo a realizar será basado en *hard decisión*, permitiéndose también la *soft decisión*. La explicación del porqué de la creación del objeto constelación a través de diferentes clases reside principalmente en este punto, pues con la intención de reducir el coste computacional en el cálculo de las distancias de las muestras a los puntos definidos en la constelación, cada modulación explota las características geométricas de la propia constelación. En particular, en las modulaciones tipo QAM, el cálculo se lleva a cabo por sectores. En el constructor, el espacio que define a la constelación es dividida en sectores cuadrados y cada uno de ellos es asociado a un punto de la constelación.
Para poder utilizar el bloque *constellation_receiver* se le ha de pasar por parámetros la base del objeto (invocando a esta con el método base()). Este bloque también tiene en cuenta las desviaciones que puedan surgir en fase y en frecuencia (aunque esta última fue previamente corregida), la alternativa a este bloque, es el bloque *constellation_decoder* aunque para ello se tendrá que emplazar un bloque que compense la desviación en fase producida (Véase apartado 6.2).

Si se transmitió una modulación con codificación diferencial, el siguiente bloque a atravesar será el decodificador diferencial al que se le indicará el número de símbolos existentes en la modulación empleada:

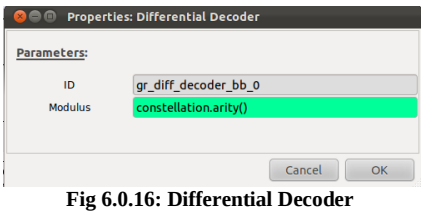


Fig 6.0.16: Differential Decoder

Nuevamente, el bloque selector será el encargado de elegir si se empleó una modulación diferencial o no. Tanto si se utilizó modulación diferencial como si no, el siguiente bloque que sigue al selector es el encargado de desmapear el código Gray.

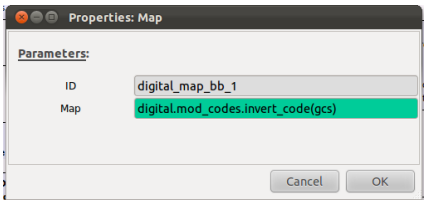


Fig 6.0.17: Gray decode

Para llevar a cabo el desmapeo, se ha de indicar el código inverso de Gray empleado en la codificación. Disponiendo así a la salida del bloque, la cadena de bytes correctamente convertidos a binario.

Por último, el bloque final que conforma el demodulador se llama *unpacked* cuya configuración se muestra a continuación:

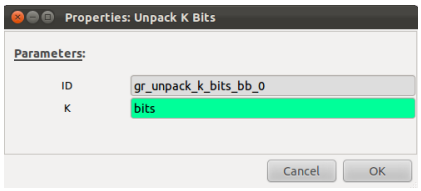


Fig 6.0.18: Unpack K Bits

La función que realiza este bloque no es más que desempaquetizar los bytes. Genera K bytes uno por cada uno de los *chunks* o K bits (los que definen el símbolo) emplazándolos en la posición menos significante (LSB[44]) del byte.

Una vez se disponen de los bytes desempaquetados el siguiente bloque con el que se encuentra el flujo de bits es el *packet decoder*:

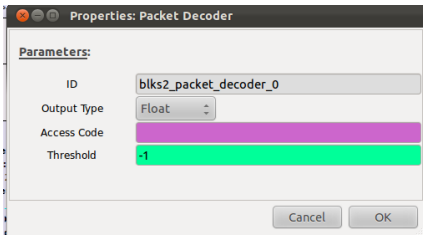


Fig 6.0.19: Packet Decoder

Este bloque, parte de una cadena de bytes desempaquetados y se encarga por una parte de lidiar con el sincronismo a nivel paquete y por otra, transformar el tipo de dato de tipo byte a tipo float para poder finalmente entregar los datos demodulados y decodificados al sumidero pertinente. En este caso, se trata de un sumidero gráfico que representará el espectro de la señal demodulada.

6.1.1.3 Resultados obtenidos

Una vez explicado el flujo del diseño del transmisor y el receptor, así como los bloques de procesado y sus parámetros que conforman tanto el modulador como el demodulador, se pasa a mostrar los resultados obtenidos en la ejecución de este diseño. Para ello, al tratarse de un entorno de simulación, los esquemáticos que representan al transmisor y al receptor han de situarse en un mismo archivo grc[45].

Para las pruebas realizadas, se supondrá una situación ideal en la que no haya presencia de ruido. Para hacer posible la visualización de las figuras que representan la señal transmitida/recibida en distintos puntos de la cadena, se ha recurrido al uso sumideros gráficos a lo largo del flujo, los cuales se encargarán de mostrar el espectro o bien la constelación.

[44] LSB: Less Significant Bit
[45] Grc: es la extensión de los esquemáticos desarrollados en la interfaz gráfica GNU Radio companion.

6.1.1.3.1 Modulación 4-QAM

Inicialmente se muestra el espectro de la señal generada, este espectro procede de visualizar la señal que transmite la tarjeta de sonido. Esta señal ha sido amplificada mediante el uso del bloque multiplicador:

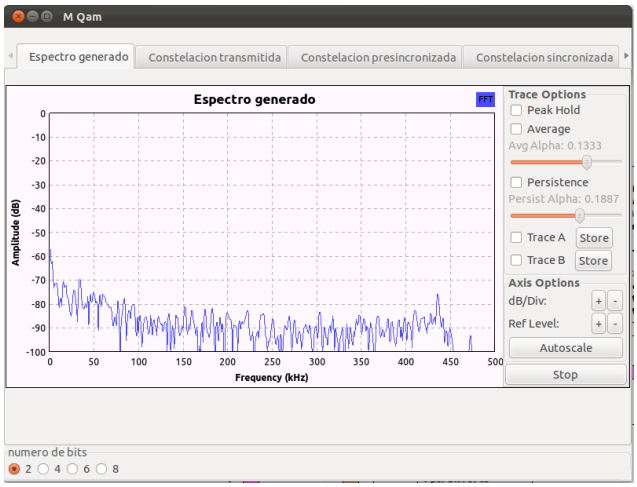


Fig 6.0.20: Espectro de la señal generada para 4-QAM

Esta señal representará el espectro de la señal que reciba por el micrófono.

Según el diagrama de flujo del transmisor, la señal amplificada vía software es conducida al modulador 4-QAM obteniéndose a la salida de dicho bloque la siguiente imagen:

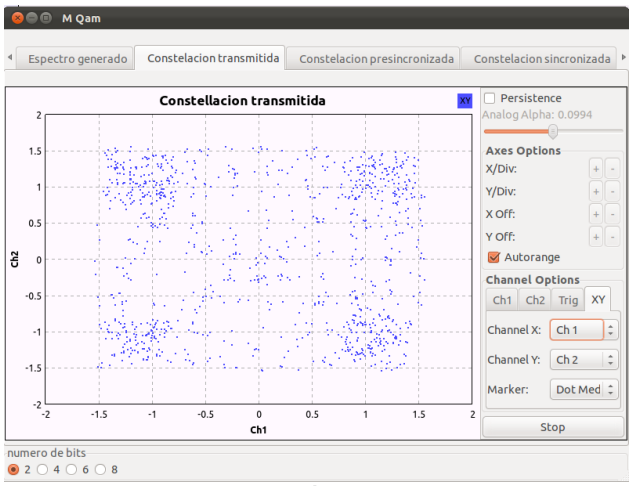


Fig 6.0.21: Constelación 4-QAM transmitida

A medida que se aumente el número de muestras por símbolo se estaría aumentando la resolución del filtro (Véase fig 6.0.11) y se apreciaría con mejor calidad la constelación transmitida.

A continuación se mostrará la constelación de la señal recibida que al no introducir ruido ni ningún efecto que haga necesaria su compensación, coincidirá en gran medida con la constelación transmitida.

Aunque pueda apreciarse alguna diferencia, esto se debe a que la señal generada varía con el tiempo y las capturas de las imágenes se hicieron de manera secuencial:

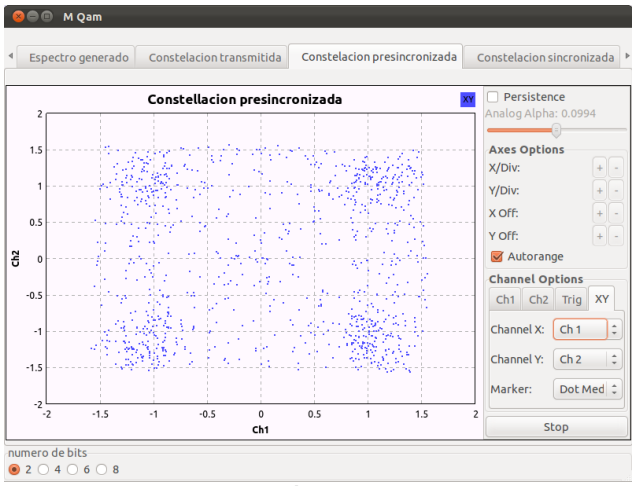


Fig 6.0.22: Constelación 4-QAM presincronizada

Como previamente se había comentado, la constelación recibida antes de la demodulación no varía significativamente con la constelación de la señal transmitida.

Esta señal atravesará de nuevo un filtro en coseno alzado, de esta forma se consigue reducir el fenómeno de la ISI:

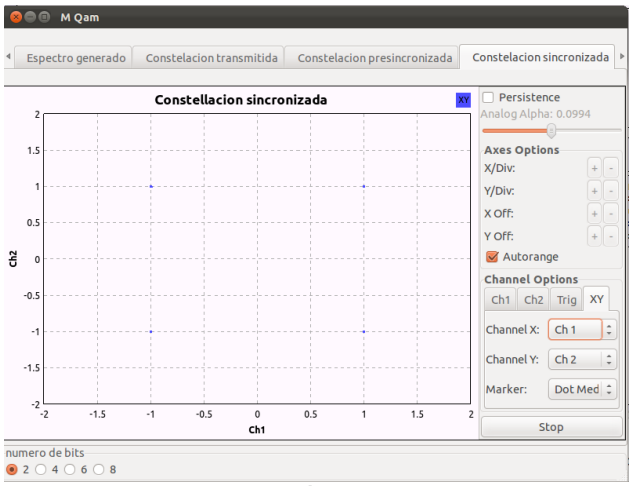


Fig 6.0.23: Constelación 4-QAM sincronizada

Donde ahora ya sí, es posible apreciar la constelación que presenta una señal típica modulada en 4-QAM.

Finalmente, la señal atraviesa la parte restante de la cadena del demodulador y del receptor desembocando en un sumidero gráfico que representará el espectro de la señal recibida:

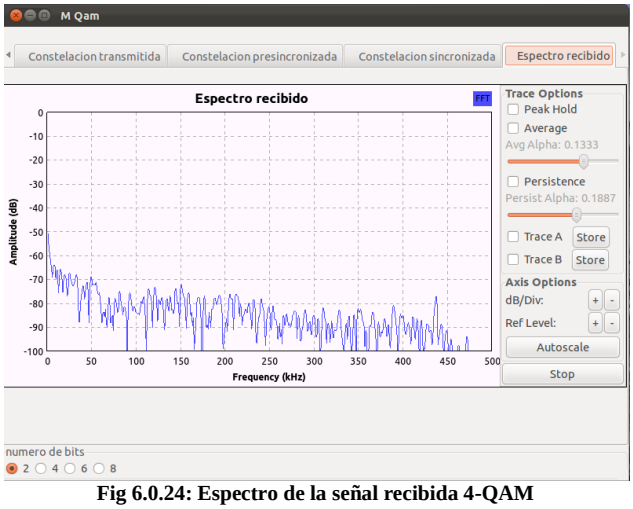


Fig 6.0.24: Espectro de la señal recibida 4-QAM

Esta es la figura que ilustra que el proceso se ha llevado correctamente, de no haber sido así, si se hubiera producido errores el bloque *packet decoder*, no hubiera devuelto datos (Véase el apartado 6.2.2.1) y esta imagen quedaría completamente en blanco. Cabe destacar, que si existe alguna diferencia entre los espectros de la señal generada y recibida se debe a la variabilidad de la señal en el tiempo y a los diferentes momentos de capturas.

6.1.1.3.2 Modulación 16-QAM

Inicialmente se muestra el espectro de la señal generada, al igual que el apartado anterior, la señal generada es la emitida por la tarjeta de sonido amplificada vía software mediante el uso del bloque multiplicador:

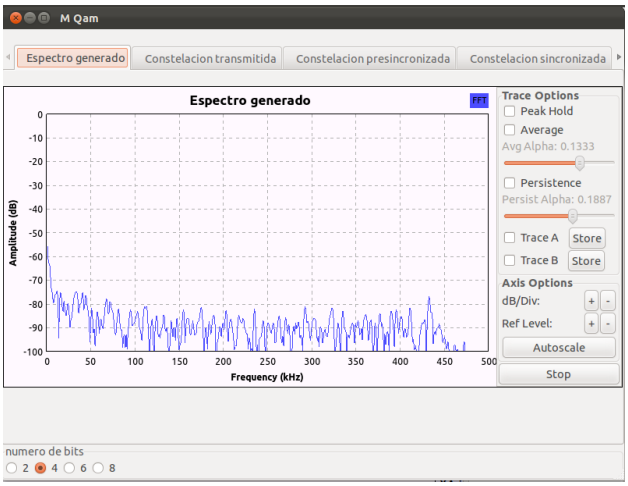


Fig 6.0.25: Espectro de la señal generada para 16-QAM

Según el diagrama de flujo del transmisor, la señal amplificada vía software es conducida al modulador 16-QAM, obteniéndose a la salida de dicho bloque la siguiente imagen:

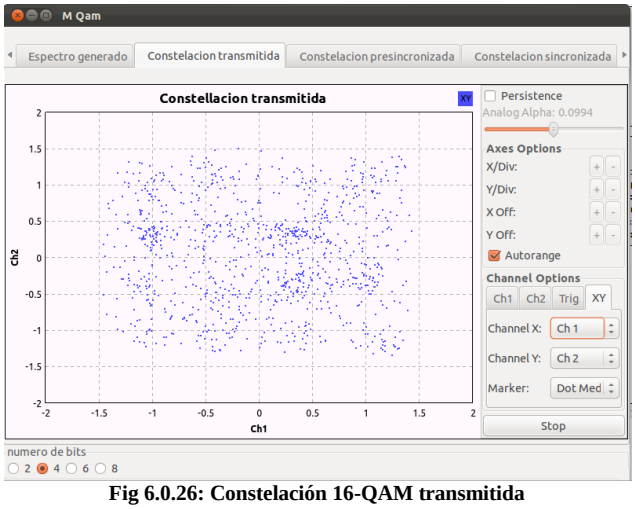


Fig 6.0.26: Constelación 16-QAM transmitida

Como se puede apreciar, al aumentar el índice de la modulación la constelación que en el apartado anterior era visible ahora resulta inapreciable.

A continuación se mostrará la constelación de la señal recibida que al no introducir ruido ni ningún efecto que induzca a la desincronización, coincidirá en gran medida con la constelación transmitida:

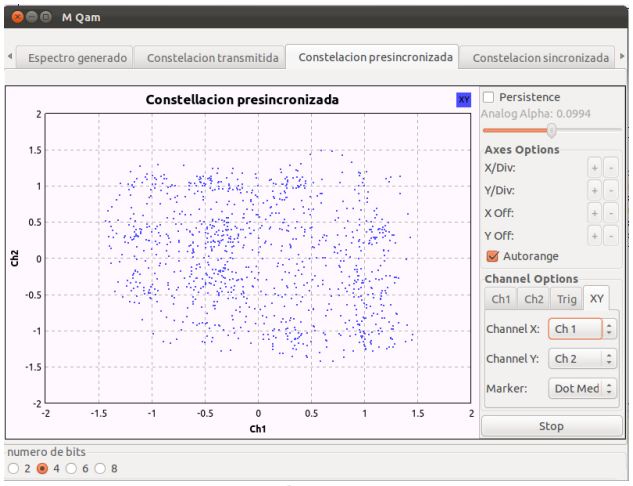


Fig 6.0.27: Constelación 16-QAM presincronizada

Como previamente se había comentado, la constelación de la señal que se recibe antes del demodulador no varía significativamente con la constelación de la señal transmitida.

Esta señal atravesará de nuevo un filtro en coseno alzado solucionando así el problema de la interferencia entre símbolos:

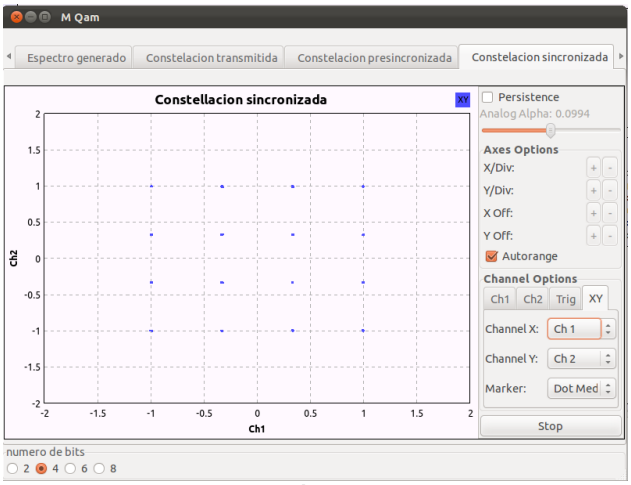


Fig 6.0.28: Constelación 16-QAM sincronizada

Donde ahora ya sí, es posible apreciar la constelación que presenta una señal típica modulada en 16QAM.

Finalmente, la señal atraviesa la parte restante de la cadena del demodulador y del receptor donde es conducida a un sumidero gráfico que representará el espectro de la señal recibida:

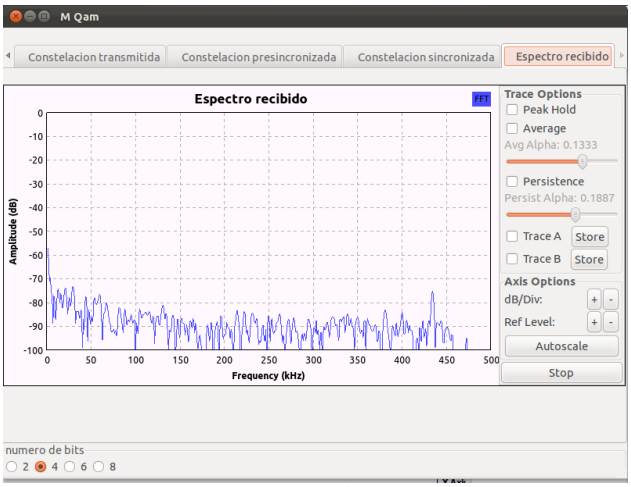


Fig 6.0.29: Espectro de la señal recibida 16-QAM

Nuevamente se recurrirá a la figura anterior para verificar que el proceso de modulación y demodulación se ha llevado a cabo de manera correcta.

6.1.1.3.2 Modulación 64-QAM

Inicialmente se muestra el espectro de la señal generada, al igual que el apartado anterior, la señal generada es la emitida por la tarjeta de sonido amplificada vía software mediante el uso del bloque multiplicador:

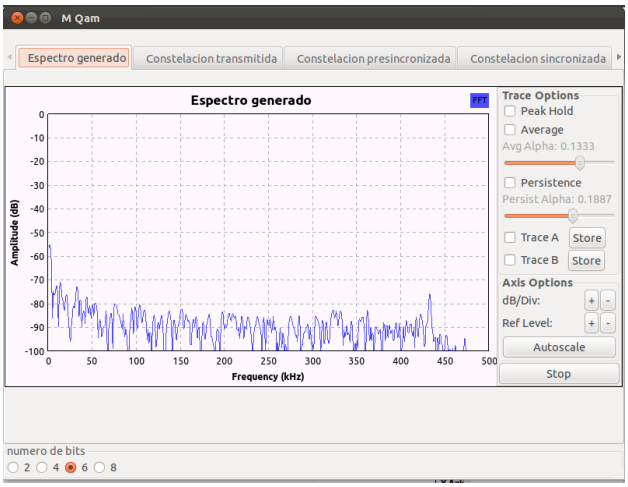


Fig 6.0.30: Espectro de la señal generada para 64-QAM

Según el diagrama de flujo del transmisor, la señal amplificada vía software es conducida al modulador 64-QAM obteniéndose a la salida de dicho bloque la siguiente imagen:

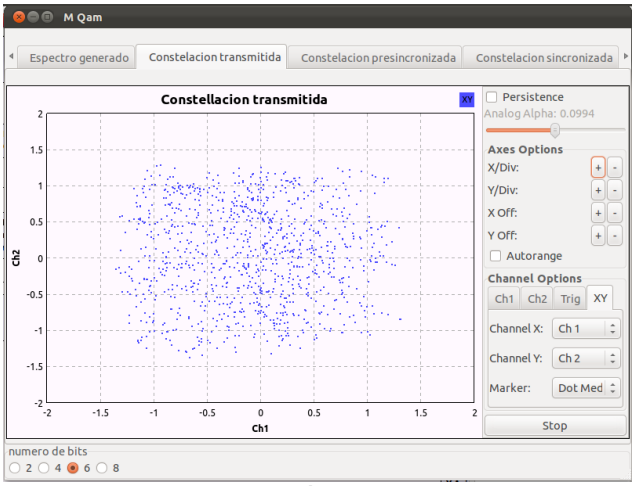


Fig 6.0.31: Constelación 64-QAM transmitida

A continuación se mostrará la constelación de la señal recibida que al no introducir ruido ni ningún efecto que induzca a la desincronización coincidirá en gran medida con la constelación transmitida.

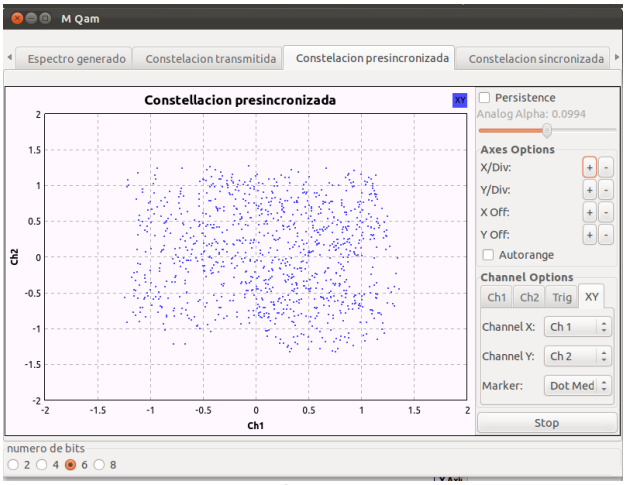


Fig 6.0.32: Constelación 64-QAM presincronizada

En este caso, al igual que ocurría con la constelación transmitida, no se aprecian los símbolos transmitidos, por lo tanto habrá que esperar a mostrar la señal una vez haya atravesado el filtro adaptado.

La siguiente figura muestra la señal una vez atraviesa el filtro adaptado:

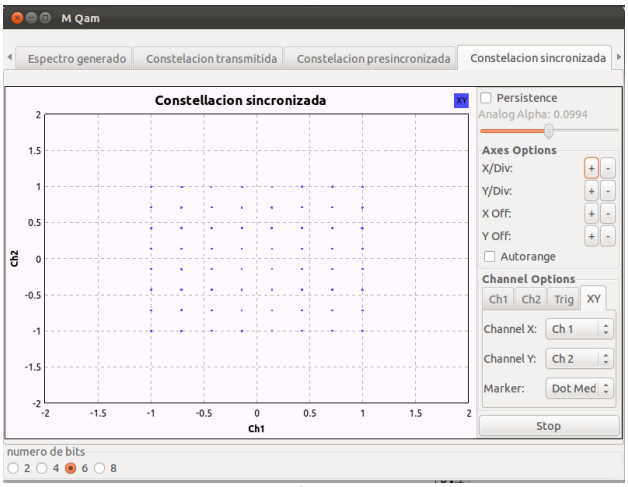


Fig 6.0.33: Constelación 64-QAM sincronizada

Una vez la señal atraviesa el filtro adaptado, se puede apreciar la constelación de una señal 64-QAM.

Finalmente, la señal atraviesa la parte restante de la cadena del demodulador y del receptor donde es conducida a un sumidero gráfico que representará el espectro de la señal recibida:

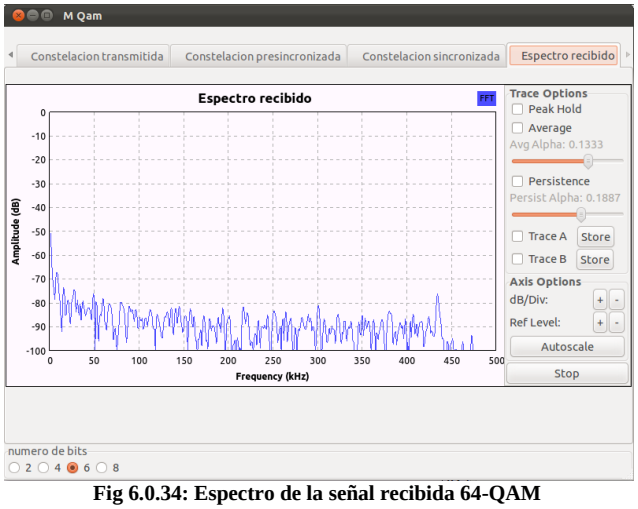


Fig 6.0.34: Espectro de la señal recibida 64-QAM

Nuevamente se recurrirá a la figura anterior para verificar que el proceso de modulación y demodulación se ha llevado a cabo de manera correcta.

6.1.1.3.2 Modulación 256-QAM

Inicialmente se muestra el espectro de la señal generada, al igual que el apartado anterior, la señal generada es la emitida por la tarjeta de sonido amplificada vía software mediante el uso del bloque multiplicador:

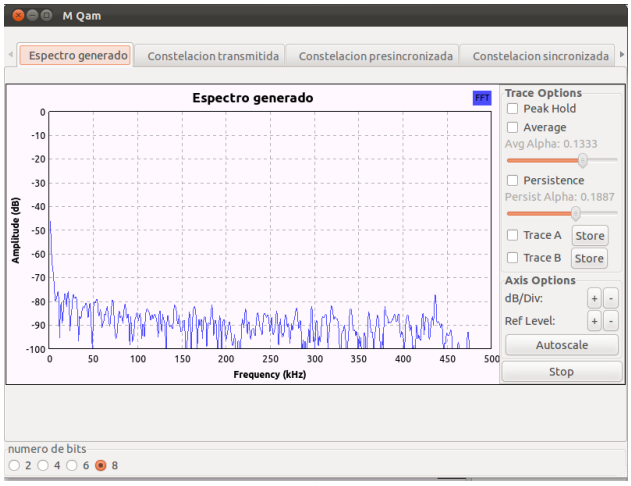


Fig 6.0.35: Espectro de la señal generada para 256-QAM

Según el diagrama de flujo del transmisor, la señal amplificada vía software es conducida al modulador 256-QAM obteniéndose a la salida de dicho bloque la siguiente imagen:

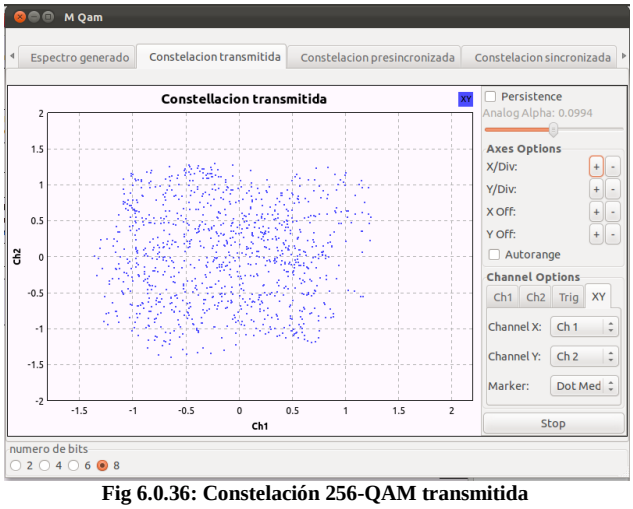


Fig 6.0.36: Constelación 256-QAM transmitida

A continuación se mostrará la constelación de la señal recibida bajo las mismas condiciones que el apartado anterior. Debido a la densidad de la constelación, tal y como ocurría en el apartado anterior, en este punto de la cadena, no se podrá diferenciar ningún símbolo.

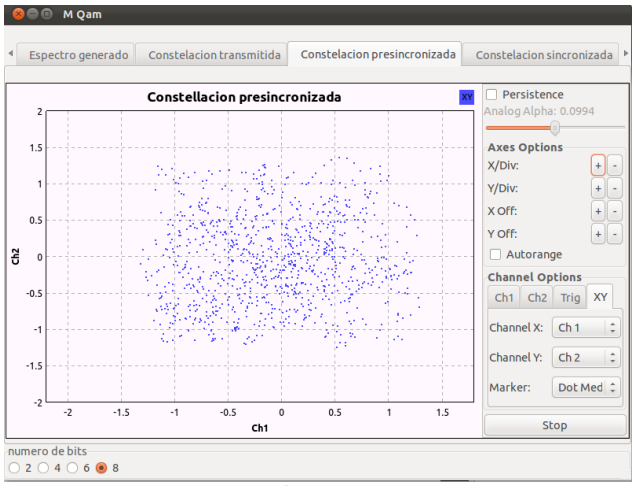


Fig 6.0.37: Constelación 256-QAM presincronizada

Una vez la señal atraviese el filtro adaptado, los símbolos serán diferenciables tal y como muestra la siguiente figura:

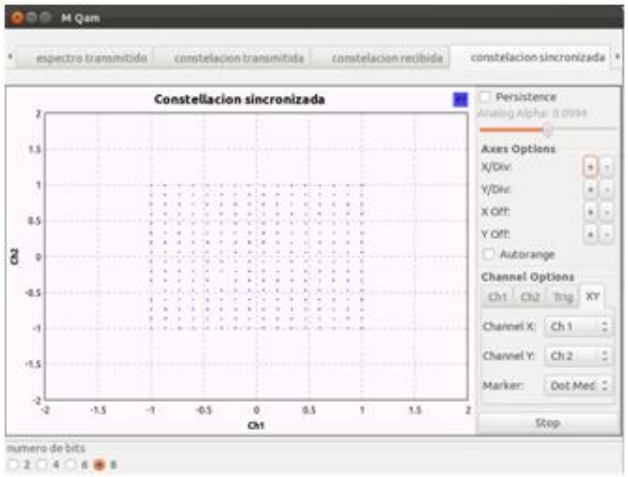


Fig 6.0.38: Constelación 256-QAM sincronizada

La figura anterior muestra la constelación de una señal 256-QAM. Tal y como se puede apreciar, la constelación está distribuida dentro de un cuadrado de dimensiones ±1 ±j. GNU Radio limita a 1024 el índice máximo de la modulación QAM.

Finalmente, la señal atraviesa la parte restante de la cadena del demodulador y del receptor, donde es conducida a un sumidero gráfico que representará el espectro de la señal recibida:

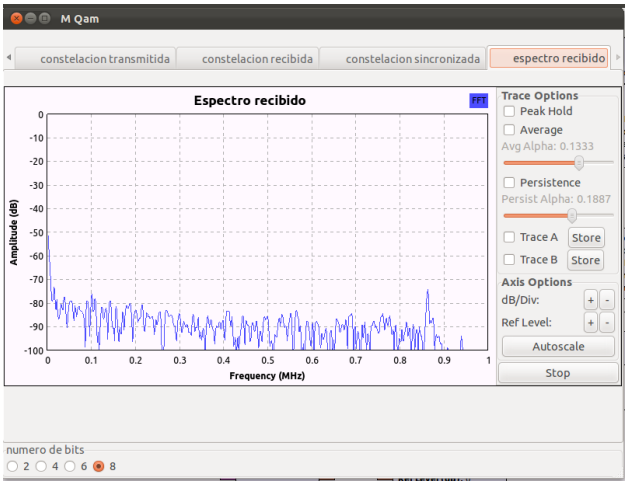


Fig 6.0.39: Espectro de la señal recibida 256-QAM

Tal y como muestra la imagen anterior, la señal recibida guarda un gran parecido con la señal generada. En caso de apreciar alguna diferencia, se debe fundamentalmente a la variabilidad de la señal.

6.1.2 Sistema M-PSK

En este apartado se abordará la manera de diseñar un sistema que utilice una modulación M-PSK
Para llevar a cabo el diseño, se utilizará como fuente de información una señal de audio cuya fuente es la tarjeta de sonido que envía la información que le llega por el micrófono.

Por último, se utilizará como *sink* un sumidero gráfico que se encargará de representar el espectro de la señal que le llega.
La cadena utilizada para diseñar el modulador y el demodulador M-PSK será muy similar a la utilizada en la modulación M-QAM por lo que solo se hablará de los bloques que cambien su configuración.

6.1.2.1 Transmisor M-PSK

La siguiente figura muestra el diseño realizado para un modulador M-PSK, para ello, se modificará el parámetro bits en función de la modulación que se desee transmitir. En concreto, la siguiente figura se corresponde con un modulador 8-PSK:

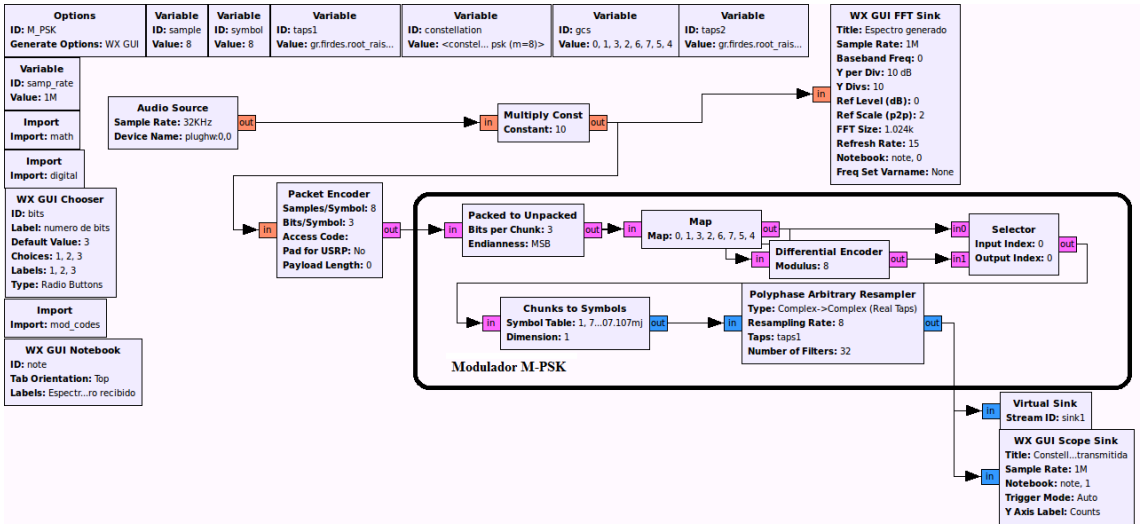


Fig 6.0.40: Diseño modulador M-PSK

Tal y como muestra la figura 6.0.1, donde se diseñaba un modulador M-QAM, el esquema de modulación es muy similar, este hecho induce a pensar que aquellas modulaciones digitales cuya constelación puede ser representada en el plano IQ(también para modulaciones ASK y BPSK) podrán hacer uso de este flujo, modificando los parámetros necesarios, como por ejemplo el número de bits (en el caso de la QAM, el índice de la modulación ha de ser potencia de 4) y el objeto que se encarga de crear la constelación.

En este caso, la modulación es creada invocando al script psk.py tal y como muestra la siguiente imagen:

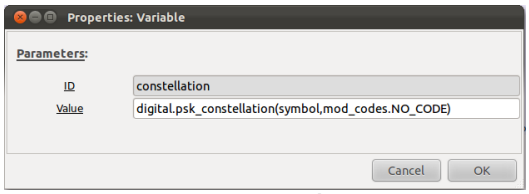


Fig 6.0.41: Constelación M-PSK

En este caso, tal y como ocurría con la modulación QAM, se aprovecha la existencia de un fichero auxiliar encargado de llevar a cabo la creación de la constelación. Este fichero internamente invoca a la clase constellation_psk, siendo ésta la clase genérica que realiza el proceso en las modulaciones tipo PSK. También existen clases que generan un tipo específico de constelación PSK (BPSK, QPSK, 8-PSK).
Para poder crear la constelación desde el archivo psk.py, se le ha de indicar el índice de modulación y si se desea codificación Gray, aunque por defecto no se realiza.

6.1.2.2 Receptor M-PSK

La siguiente figura muestra el diseño realizado para un demodulador M-PSK, para ello, se modificará el parámetro bits en función de la modulación transmitida anteriormente. En concreto, la siguiente figura se corresponde con un demodulador 8-PSK:

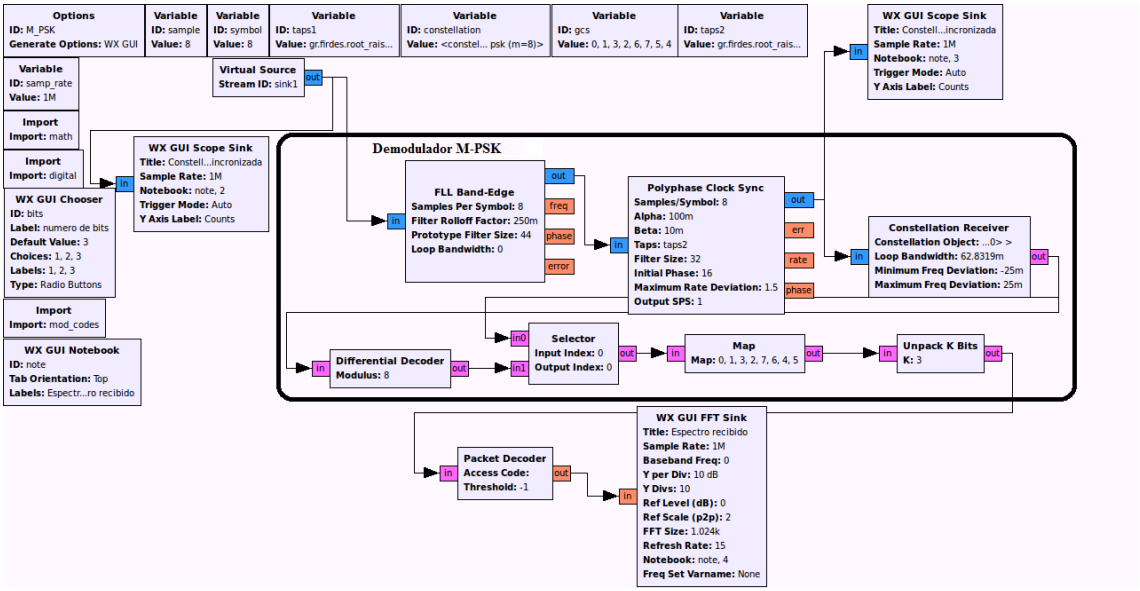


Fig 6.0.42: Diseño demodulador M-PSK

Nuevamente, tal y como se ha mencionado anteriormente, el demodulador M-PSK es muy similar al demodulador M-QAM. Sin embargo, al haberse definido el objeto constelación mediante la clase constellation_psk la manera en que se realizará el mapeo de señales complejas a símbolos se hará de manera diferente. En este caso, en las modulaciones tipo PSK, el cálculo se lleva a cabo también por sectores. En el constructor de la clase, el espacio que define a la constelación es dividida en sectores

(gajos de la circunferencia) y cada uno de ellos es asociado a un punto de la constelación.

6.1.2.3 Resultados obtenidos

Para las pruebas realizadas, se supondrá una situación ideal en la que no haya presencia de ruido. Para visualizar la señal transmitida/recibida en distintos puntos de la cadena tal y como ocurría en el apartado anterior, se ha recurrido al uso sumideros gráficos a lo largo del flujo, los cuales se encargarán de mostrar el espectro o bien la constelación.

6.1.2.3.1 Modulación 2-PSK

Inicialmente se muestra el espectro de la señal generada, este espectro procede de visualizar la señal que transmite la tarjeta de sonido. Esta señal ha sido amplificada mediante el uso del bloque multiplicador:

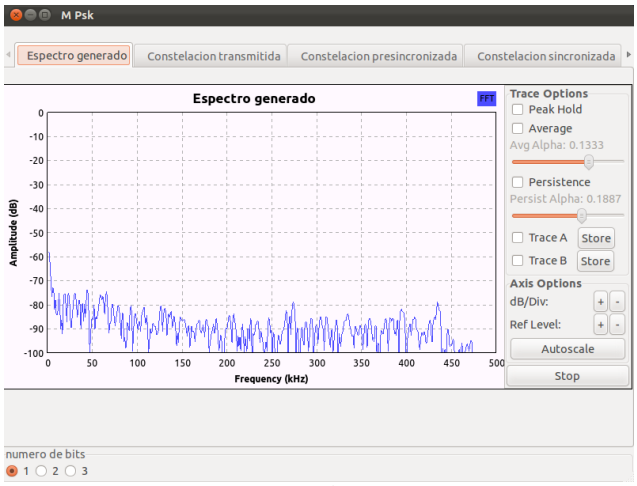


Fig 6.0.43: Espectro de la señal generada para 2-PSK

Según el diagrama de flujo del transmisor, la señal amplificada vía software es conducida al modulador 2-PSK obteniéndose a la salida de dicho bloque la siguiente imagen:

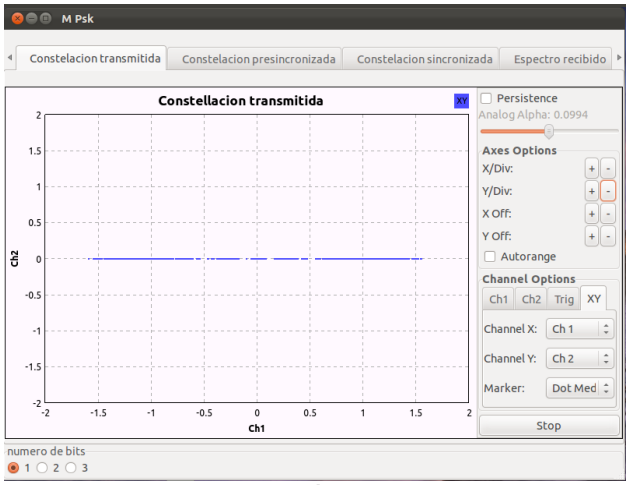


Fig 6.0.44: Constelación 2-PSK transmitida

A continuación se mostrará la constelación de la señal recibida. Al tratarse de una situación ideal, la constelación recibida coincidirá en gran medida con la transmitida.

La siguiente imagen representa a la señal recibida antes de atravesar el filtro conformado:

Fig 6.0.45: Constelación 2-PSK presincronizada

Como previamente se había comentado, la constelación recibida antes de la demodulación no varía significativamente con la constelación de la señal transmitida.

El siguiente paso en la cadena receptora es el filtro adaptado, el porqué del uso de este tipo de filtros es minimizar la ISI:

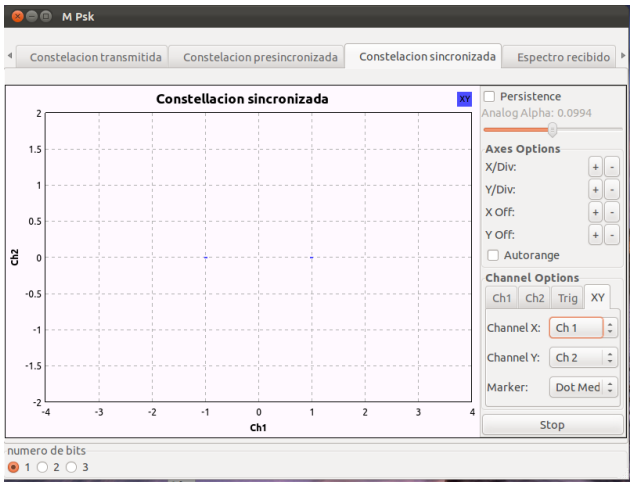


Fig 6.0.46: Constelación 2-PSK sincronizada

Tal y como se puede apreciar, una vez la señal atraviesa el filtro, la constelación BPSK es apreciable. Al tratarse de una situación ideal, dicha constelación se representará únicamente por dos puntos (-1 +1).

Finalmente, la señal atraviesa la parte restante de la cadena del demodulador y del receptor, donde es conducida a un sumidero gráfico que representará el espectro de la señal recibida:

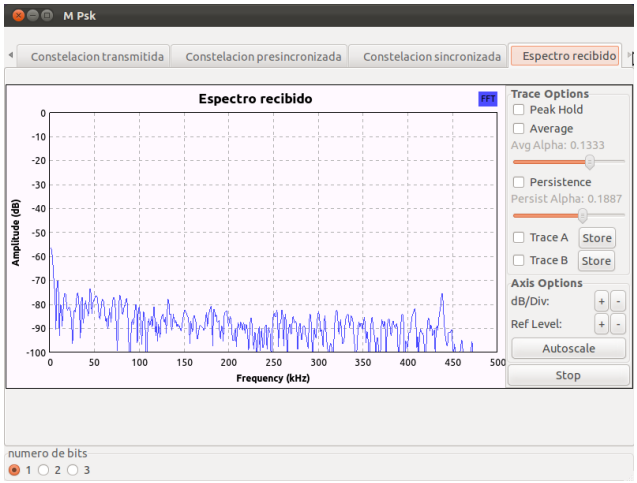


Fig 6.0.47: Espectro de la señal recibida 2-PSK

Tal y como muestra la imagen anterior, la señal ha sido correctamente demodulada. Las diferencias entre la señal generada y recibida se debe a la naturaleza variable de la señal.

6.1.2.3.2 Modulación 4-PSK

Inicialmente se muestra el espectro de la señal generada, este espectro procede de visualizar la señal que transmite la tarjeta de sonido. Esta señal ha sido amplificada mediante el uso del bloque multiplicador:

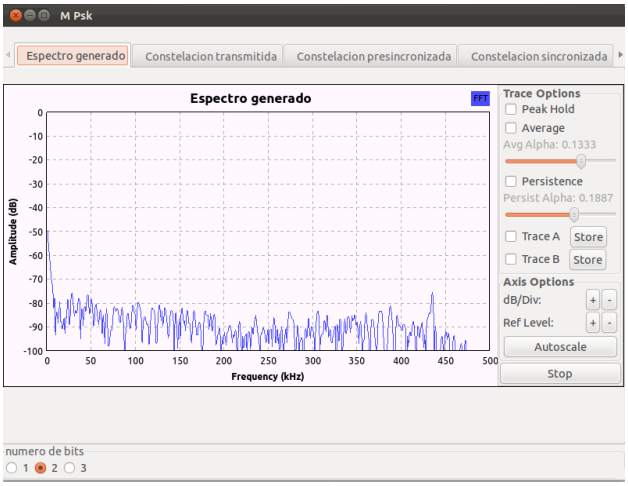


Fig 6.0.48: Espectro de la señal generada para 4-PSK

Según el diagrama de flujo del transmisor, la señal amplificada vía software es conducida al modulador 4-PSK, obteniéndose a la salida de dicho bloque la siguiente imagen:

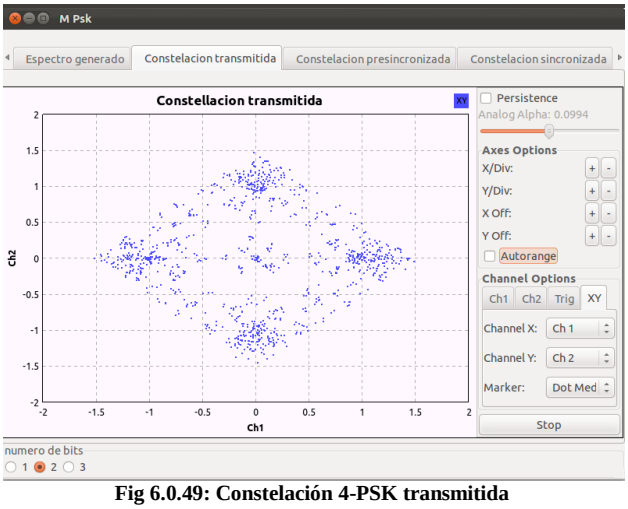


Fig 6.0.49: Constelación 4-PSK transmitida

A continuación, se mostrará la constelación de la señal recibida (antes de la demodulación), al tratarse de una situación ideal donde no hay presencia de ruido ni otros efectos indeseados, ambas imágenes serán muy similares:

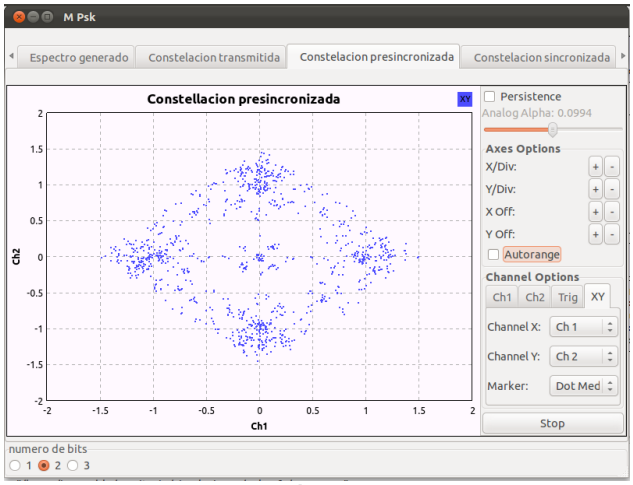


Fig 6.0.50: Constelación 4-PSK presincronizada

Esta señal atravesará un filtro adaptado (filtro en coseno alzado) de tal forma que el efecto de la ISI será minimizado, obteniéndose así la siguiente imagen:

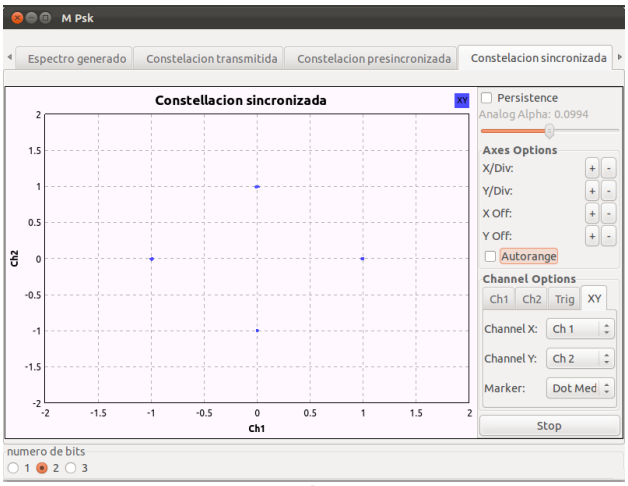


Fig 6.0.51: Constelación 4-PSK sincronizada

Tal y como se puede apreciar, la figura coincide con el diagrama de constelación de una señal 4-PSK.

Finalmente, la señal atraviesa la parte restante de la cadena del demodulador y del receptor, donde es conducida a un sumidero gráfico que representará el espectro de la señal recibida:

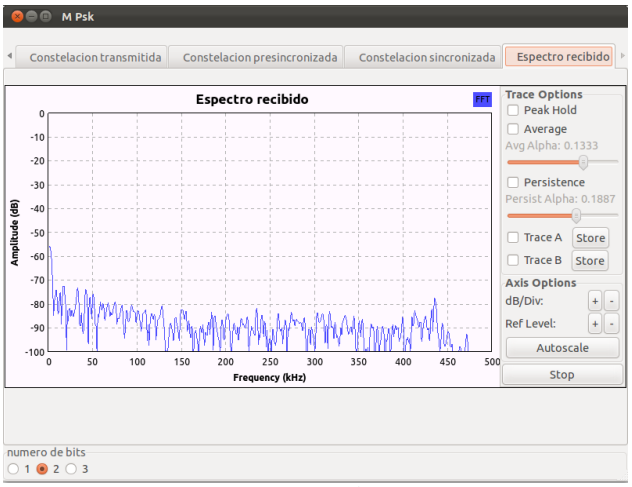


Fig 6.0.52: Espectro de la señal recibida 4-PSK

Tal y como muestra la imagen anterior, la señal ha sido correctamente demodulada. Las diferencias entre la señal generada y recibida se debe a la naturaleza variable de la señal.

6.1.2.3.3 Modulación 8-PSK

Inicialmente se muestra el espectro de la señal generada, este espectro procede de visualizar la señal que transmite la tarjeta de sonido. Esta señal ha sido amplificada mediante el uso del bloque multiplicador:

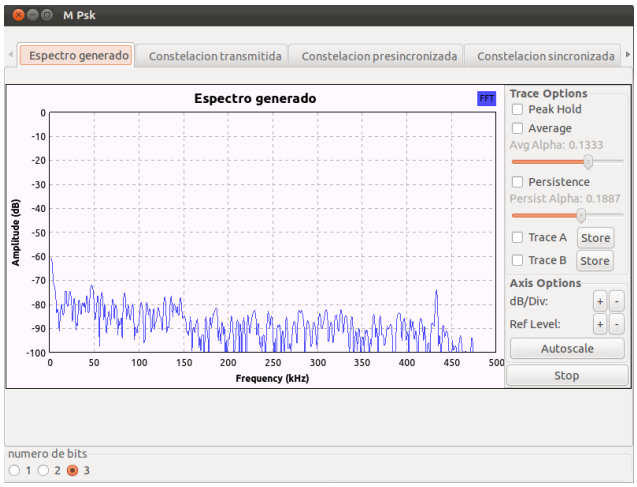


Fig 6.0.53: Espectro de la señal generada para 8-PSK

Según el diagrama de flujo del transmisor, la señal amplificada vía software es conducida al modulador 8-PSK, obteniéndose a la salida de dicho bloque la siguiente imagen:

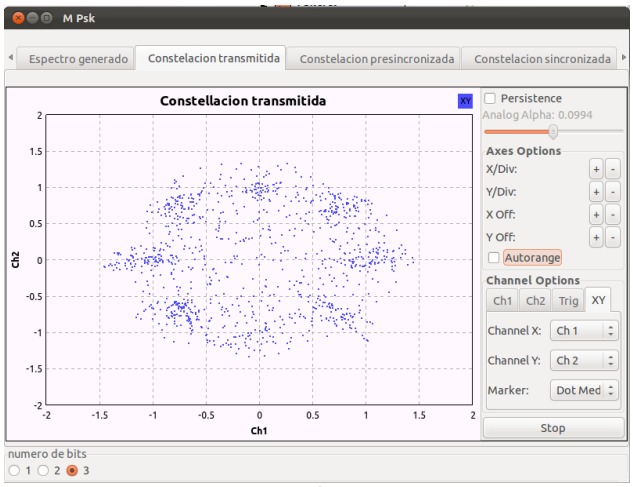


Fig 6.0.54: Constelación 8-PSK transmitida

A continuación se mostrará la constelación de la señal recibida, al tratarse de una situación ideal donde no hay presencia de ruido ni otros efectos indeseados, ambas imágenes serán muy similares:

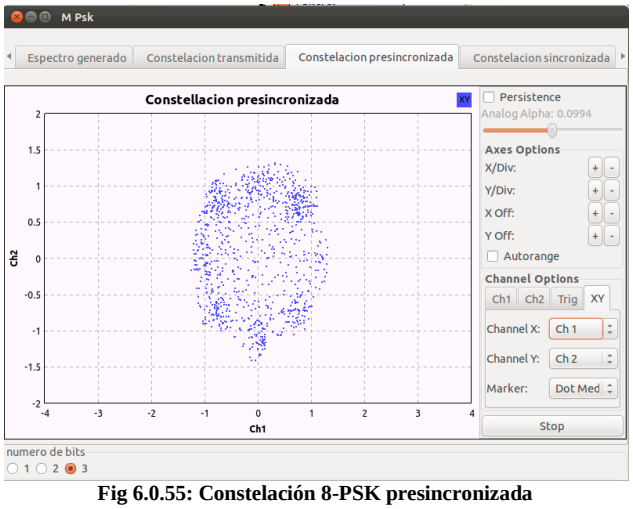


Fig 6.0.55: Constelación 8-PSK presincronizada

Como previamente se había comentado, la constelación recibida antes de la demodulación no varía significativamente con la constelación de la señal transmitida.

Esta señal atravesará de nuevo el filtro adaptado encargado de minimizar la ISI:

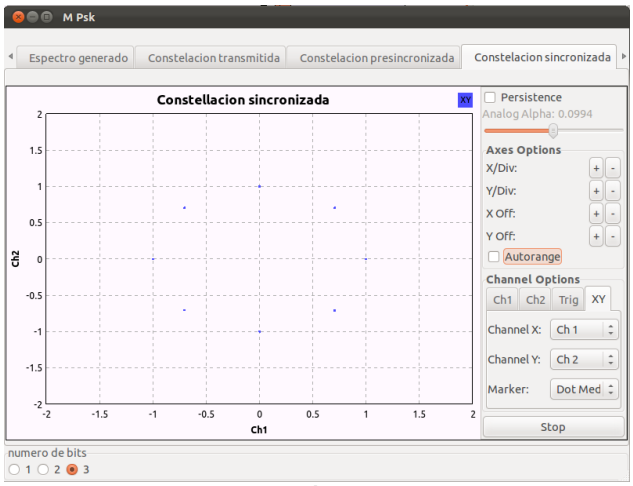


Fig 6.0.56: Constelación 8-PSK sincronizada

Tal y como se puede apreciar, la imagen anterior coincide con la constelación de una señal modulada PSK empleando 3 bits para determinar cada símbolo (modulación 8-PSK).

Finalmente, la señal atraviesa la parte restante de la cadena del demodulador y del receptor, donde es conducida a un sumidero gráfico que representará el espectro de la señal recibida:

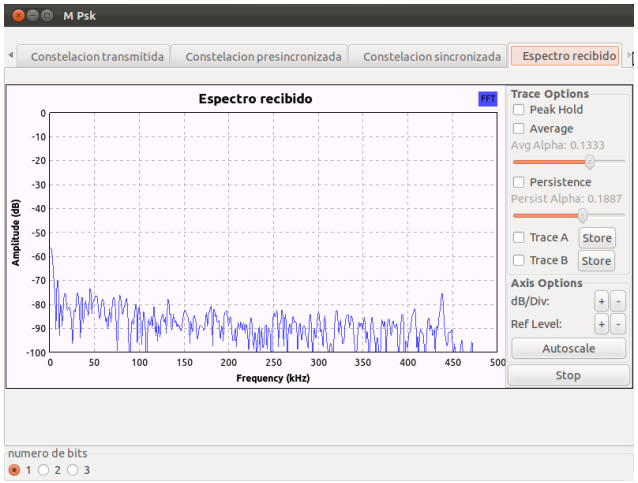


Fig 6.0.57: Espectro de la señal recibida 8-PSK

Tal y como muestra la imagen anterior, la señal ha sido correctamente demodulada. Las diferencias entre la señal generada y recibida se debe a la naturaleza variable de la señal.

6.1.3 Sistema M-ASK

En este apartado se abordará como se puede diseñar un sistema que utilice una modulación M-ASK.
Para llevar a cabo el diseño, se utilizará como fuente de información una señal de audio cuya fuente es la tarjeta de sonido que envía la información que le llega por el micrófono.

Por último, se utilizará como sumidero un sumidero gráfico que se encargará de representar el espectro de la señal que le llega.
La cadena utilizada para diseñar el modulador y el demodulador M-ASK será muy similar a la utilizada en la modulación M-QAM por lo que solo se hablará de los bloques que cambien su configuración.

6.1.3.1 Transmisor M-ASK

La siguiente figura muestra el diseño realizado para un modulador M-ASK, para ello, se modificará el parámetro bits en función de la modulación que se desee transmitir. En concreto, la siguiente figura se corresponde con un modulador 4-ASK:

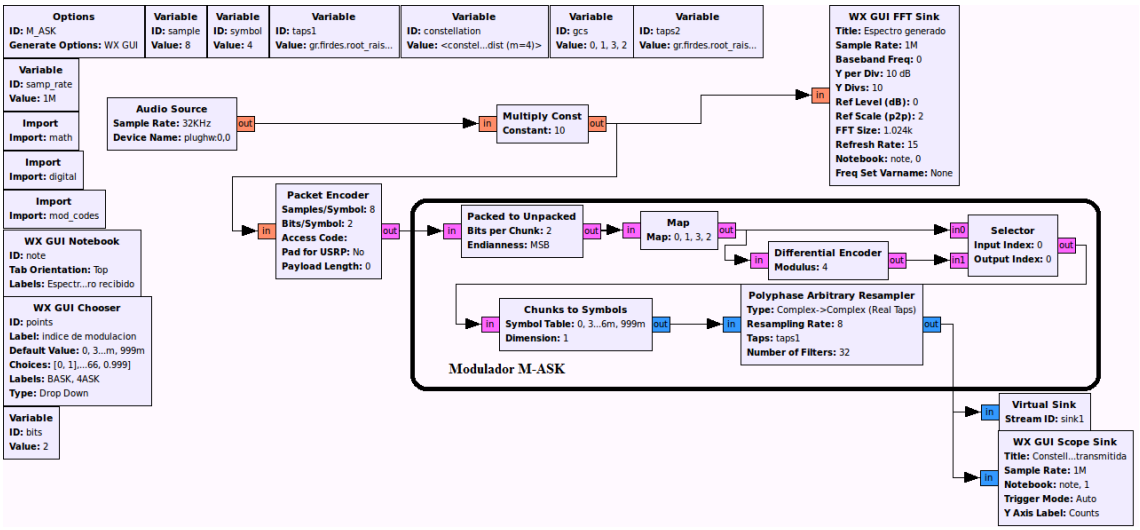


Fig 6.0.58: Diseño modulador M-ASK

La manera en que la constelación es creada es diferente con respecto a los casos anteriores no solo por los puntos que la describen, sino porque a la hora de demodular (asignar las muestras recibidas a puntos definidos en nuestra constelación creada) se emplearán diferentes métodos explotando las características de cada una de las modulaciones. En este caso, no existe ningún script que contemple una modulación ASK. Sin embargo, aprovechando la disposición de métodos que sirven para la creación de constelaciones genéricas, se definirán los puntos de la constelación que se desee y a partir de estos, se creará el objeto que definirá a la constelación invocando al método calcdist:

Fig 6.0.59: Constelación M-ASK

La figura presentada, representa la invocación de la creación de la constelación genérica. En general, cualquier constelación de las anteriormente vistas puede ser generada mediante esta clase, aunque no es recomendada para modulaciones largas. Los parámetros que se le ha de indicar son los puntos que conforman el espacio complejo, si se desea emplear codificación Gray en la generación de la constelación, se le ha de indicar el alfabeto. También se le ha de indicar el índice de rotación de la constelación (número de rotaciones a lo largo de la circunferencia unidad que tienen la misma representación) y el número de dimensiones. Este último parámetro estará fijado a 1 puesto que no coincide con el número de dimensiones de la constelación, sino que representa el número de números complejos mapeados en un símbolo.

Como ye se había introducido anteriormente, la creación de los puntos de la constelación se hace de manera manual:

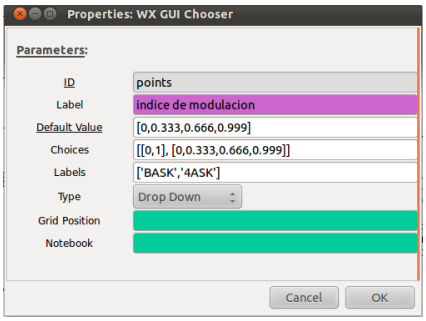


Fig 6.0.60: Constellation points M-ASK

Otro parámetro que se utiliza en el diseño es el número de bits:

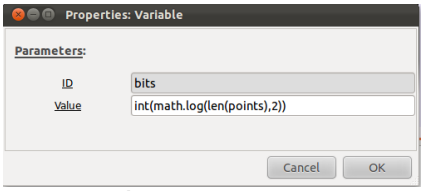


Fig 6.0.61: Número de bits utilizados M-ASK

6.1.3.2 Receptor M-ASK

La siguiente figura muestra el diseño llevado a cabo para un demodulador M-ASK, para ello, se modificará el parámetro bits en función de la modulación transmitida anteriormente. En concreto, la siguiente figura se corresponde con un demodulador 4-ASK:

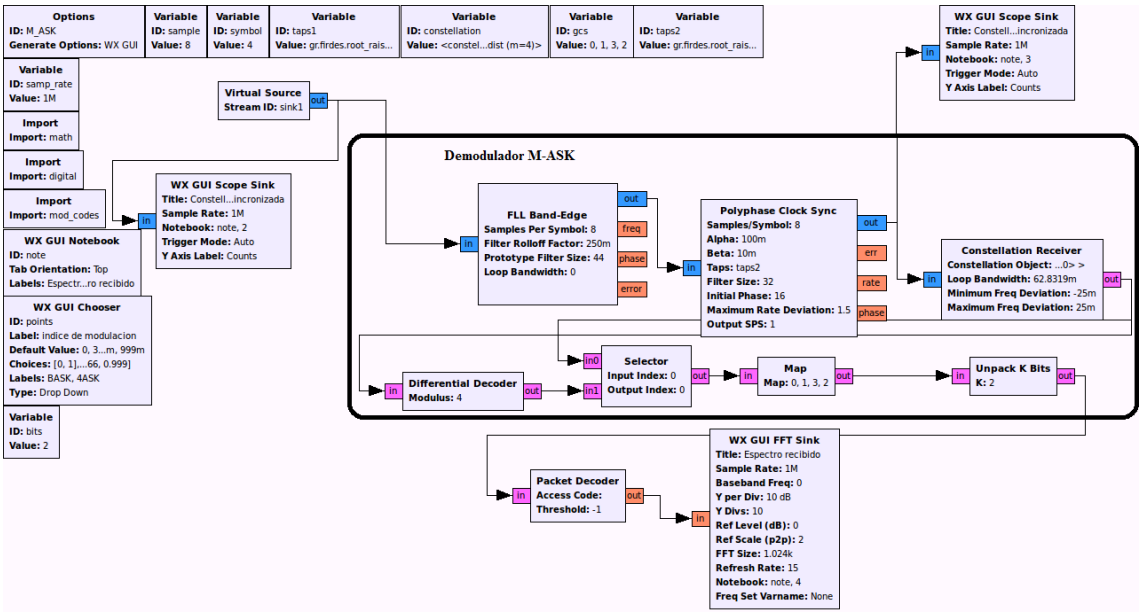


Fig 6.0.62: Diseño modulador M-ASK

Con respecto a la demodulación (mapeo de muestras complejas en símbolos), se invocará de la misma manera que ocurría en los casos QAM y PSK. Siendo

nuevamente, la manera en que se calculan las distancias la diferencia entre las demodulaciones anteriormente vistas. En este caso, la distancia no se calculará en relación a los sectores asociados a puntos de la constelación, sino que para cada una de las muestras recibidas se calculará la distancia euclídea a cada uno de los puntos de la constelación. Se asociará por lo tanto, la muestra al punto de la constelación cuya distancia obtenida sea mínima, mapeando así las muestras complejas en símbolos.

6.1.3.3 Resultados obtenidos

Para las pruebas realizadas, se supondrá una situación ideal en la que no haya presencia de ruido, para hacer posible la visualización de las figuras que representan la señal transmitida/recibida en distintos puntos de la cadena, tal y como ocurría en el apartado anterior, se ha recurrido al uso sumideros gráficos a lo largo del flujo los cuales se encargarán de mostrar el espectro o bien la constelación.

6.1.2.3.1 Modulación 2-ASK

Inicialmente se muestra el espectro de la señal generada, este espectro procede de visualizar la señal que transmite la tarjeta de sonido. Esta señal ha sido amplificada mediante el uso del bloque multiplicador:

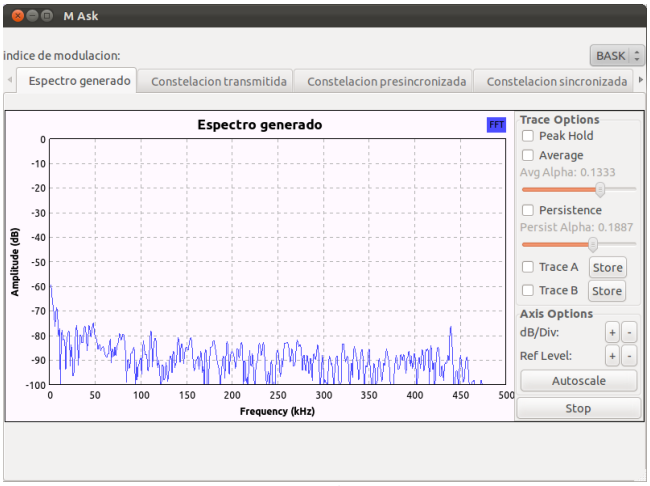


Fig 6.0.63: Espectro de la señal generada para 2-ASK

Según el diagrama de flujo del transmisor, la señal amplificada vía software es conducida al modulador BASK obteniéndose a la salida de dicho bloque la siguiente imagen:

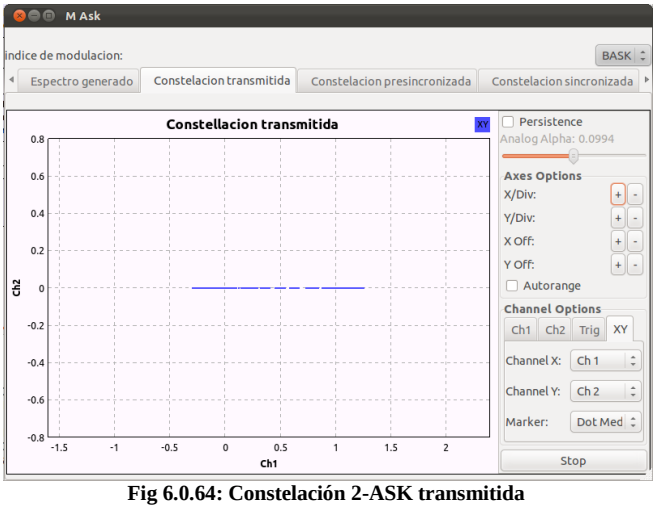


Fig 6.0.64: Constelación 2-ASK transmitida

A continuación se mostrará la constelación de la señal recibida, al tratarse de una situación ideal en donde no hay presencia de ruido ni otros efectos indeseados, ambas imágenes serán muy similares:

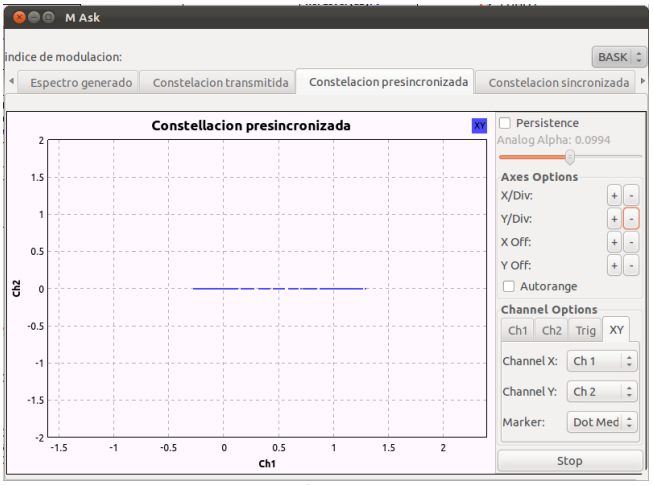


Fig 6.0.65: Constelación 2-ASK presincronizada

Esta señal atravesará un filtro adaptado (filtro en coseno alzado) de tal forma que el efecto de la ISI será minimizado, obteniéndose así la siguiente imagen:

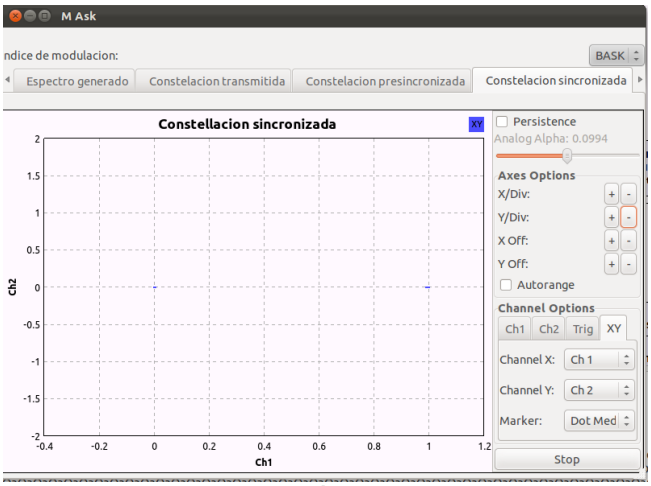


Fig 6.0.66: Constelación 2-ASK sincronizada

Tal y como se puede apreciar, la figura coincide con el diagrama de constelación de una señal BASK.

Finalmente, la señal atraviesa la parte restante de la cadena del demodulador y del receptor, donde es conducida a un sumidero gráfico que representará el espectro de la señal recibida:

Fig 6.0.67: Espectro de la señal recibida 2-ASK

Tal y como muestra la imagen anterior, la señal ha sido correctamente demodulada. Las diferencias entre la señal generada y recibida se debe a la naturaleza variable de la señal.

6.1.2.3.1 Modulación 4-ASK

Inicialmente se muestra el espectro de la señal generada, este espectro procede de visualizar la señal que transmite la tarjeta de sonido. Esta señal ha sido amplificada mediante el uso del bloque multiplicador:

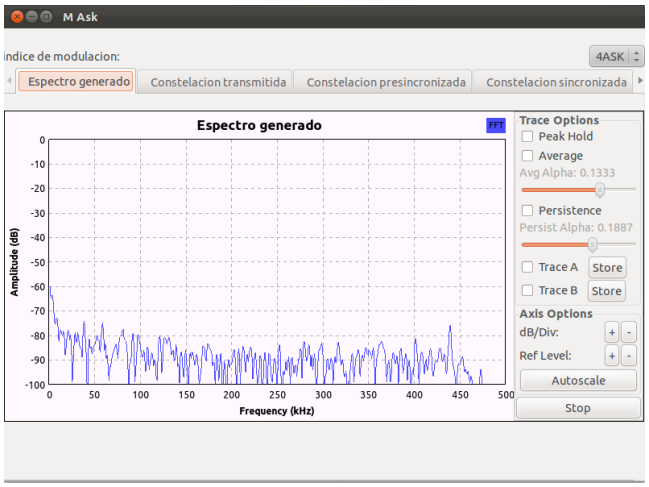


Fig 6.0.68: Espectro de la señal generada para 4-ASK

Según el diagrama de flujo del transmisor, la señal amplificada vía software es conducida al modulador 4-ASK obteniéndose a la salida de dicho bloque la siguiente imagen:

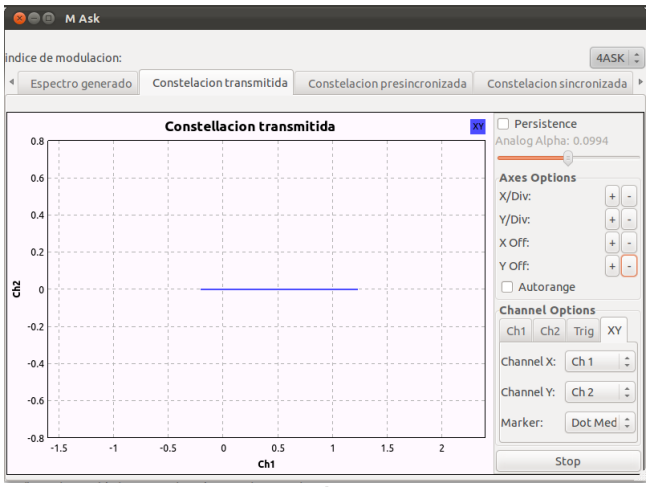


Fig 6.0.69: Constelación 4-ASK transmitida

A continuación se mostrará la constelación de la señal recibida, al tratarse de una situación ideal en donde no hay presencia de ruido ni otros efectos indeseados, ambas imágenes serán muy similares:

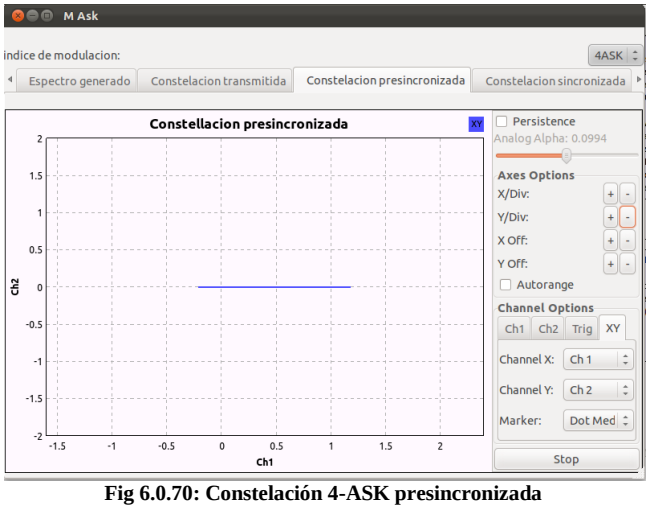


Fig 6.0.70: Constelación 4-ASK presincronizada

Esta señal atravesará un filtro adaptado (filtro en coseno alzado) de tal forma que el efecto de la ISI será minimizado, obteniéndose así la siguiente imagen:

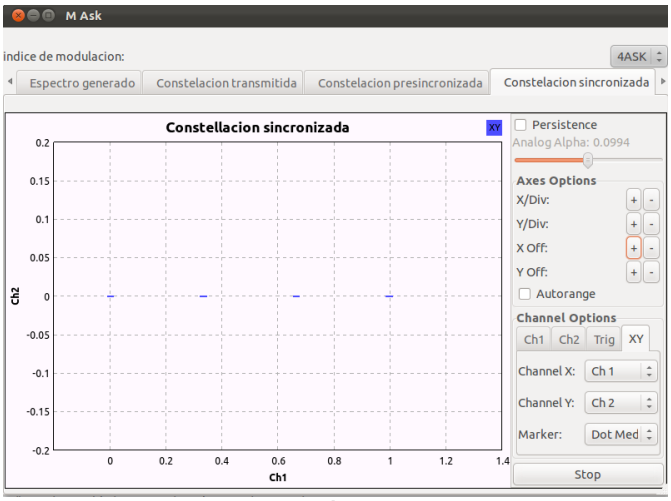


Fig 6.0.71: Constelación 4-ASK sincronizada

Tal y como se puede apreciar, la figura coincide con el diagrama de constelación de una señal 4-ASK.

Finalmente, la señal atraviesa la parte restante de la cadena del demodulador y del receptor, donde es conducida a un sumidero gráfico que representará el espectro de la señal recibida:

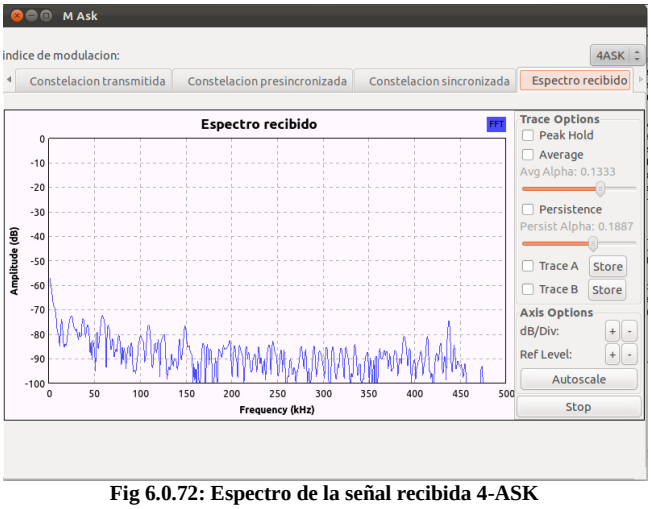


Fig 6.0.72: Espectro de la señal recibida 4-ASK

Tal y como muestra la imagen anterior, la señal ha sido correctamente demodulada. Las diferencias entre la señal generada y recibida se debe a la naturaleza variable de la señal.

6.1.4 Sistema genérico

En este apartado se llevará a cabo el diseño de un sistema digital formado por tres tipos de modulación a elegir: 4-QAM, 4-PSK y GFSK. A diferencia del anterior apartado, con el fin de alcanzar un nivel más de abstracción, se implementarán los moduladores y demoduladores mediante los bloques proporcionados por la herramienta GNU Radio. Básicamente, estos módulos engloban la mayor parte de los bloques que se han visto en el apartado anterior, o en su defecto utilizan bloques muy similares, simplificando así la tarea de diseño.

En cada tipo de modulación se tomarán las medidas oportunas para mostrar resultados intermedios.

Para llevar a cabo el diseño, se utilizará como fuente de información una señal de audio, cuya fuente es la tarjeta de sonido que envía la información que le llega por el micrófono.
Por último, se utilizará como *sink* un sumidero gráfico que se encargará de representar el espectro de la señal que le llega.

6.1.4.1 Transmisor genérico

La siguiente figura muestra el diseño llevado a cabo para un transmisor genérico:

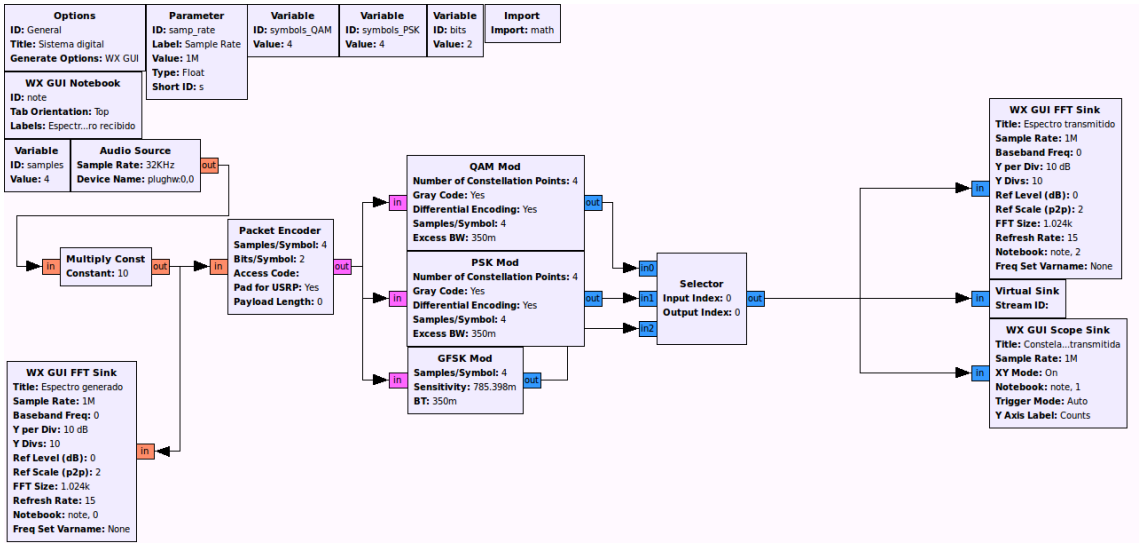


Fig 6.0.73: Diseño transmisor genérico

Estableciendo el parámetro de número de símbolos, se elegirá el índice de modulación deseado para las modulaciones QAM y PSK, mientras que para la modulación de tipo FSK el número de bits estará fijado a 1 (BFSK).

La siguiente figura muestra la configuración realizada para el bloque modulador QAM:

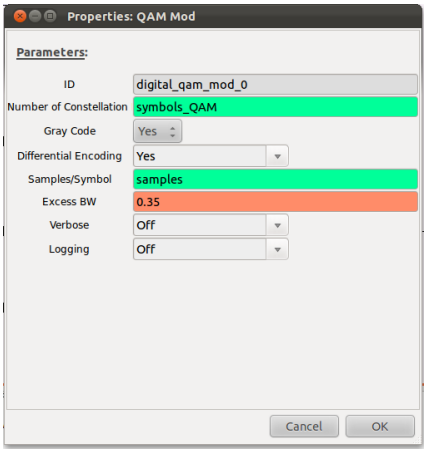


Fig 6.0.74: QAM modulator

A este bloque se le indica mediante parámetros el índice de la modulación a utilizar (número de símbolos), el número de muestras por símbolo, el factor de roll-off de los filtros y la codificación deseada ya sea tipo Gray y/o diferencial. Los parámetros restantes sirven para ofrecer información al diseñador del modulador utilizado. Con respecto al apartado 6.1.1 cabe destacar el grado de simplificación haciendo trasparente al diseñador gran parte de la configuración del modulador así como de los bloques utilizados.

La siguiente figura muestra la configuración realizada para el bloque modulador PSK:

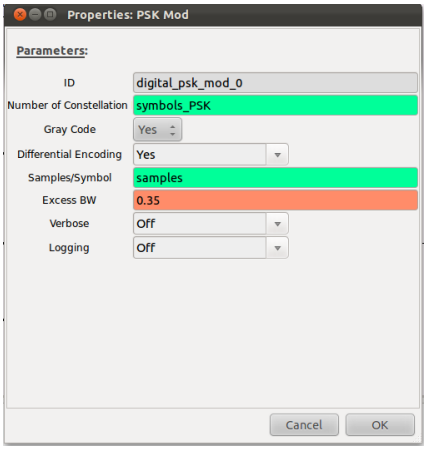


Fig 6.0.75: PSK modulator

A este bloque se le indica mediante parámetros el índice de la modulación a utilizar (número de símbolos), el número de muestras por símbolo, el factor de roll-off de los filtro y la codificación deseada ya sea tipo Gray y/o diferencial. Los parámetros restantes sirven para ofrecer información al diseñador del modulador utilizado. Con respecto al apartado 6.1.2 cabe destacar el grado de simplificación haciendo trasparente al diseñador gran parte de la configuración del modulador así como de los bloques utilizados.

La siguiente figura muestra la configuración realizada para el bloque modulador FSK:

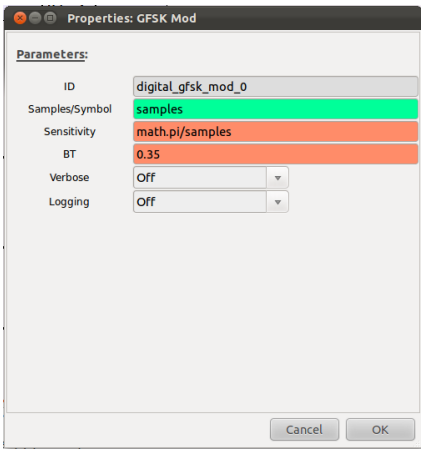


Fig 6.0.76: FSK modulator

La idea fundamental llevada a cabo en este bloque para implementar una modulación G-FSK, reside en convertir el flujo de bytes en símbolos (de valor ±1 en función del valor del byte) para atravesar un modulador en frecuencia, al que se le indica por parámetro la sensibilidad a utilizar. El parámetro de la sensibilidad queda descrito por la siguiente fórmula:

$$sensibilidad = \frac{\pi\, h}{muestras\_por\_símbolo}$$

Ec 6.0.2: Sensibilidad de la modulación FSK

Donde el parámetro h es definido como:

$$h = \frac{\Delta_f}{\frac{R_s}{2}}$$

Ec 6.0.3: Índice de modulación FSK

El parámetro Δ_f representa la desviación en frecuencia y R_s la tasa de símbolo. Tomando como desviación en frecuencia la mitad de la frecuencia de símbolo (1/2T) y al tratarse de una modulación binaria (Rs = Rb), el parámetro h toma el valor 1 y la sensibilidad por lo tanto (π/muestras_por_simbolo).

Con la intención de reducir el efecto de la ISI, se introduce un filtro adaptado que será el encargado de conformar el pulso del símbolo. En este caso, en vez de utilizar un filtro en coseno alzado que conforme el pulso, se introduce un filtro gaussiano

6.1.4.2 Receptor genérico

La siguiente figura muestra el diseño llevado a cabo para un receptor genérico:

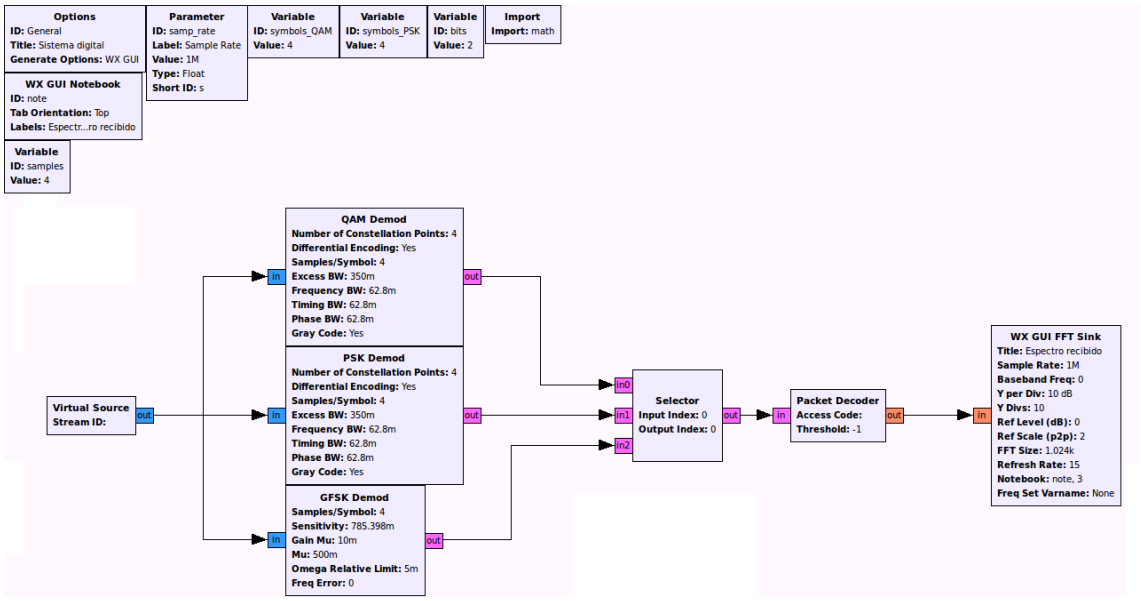


Fig 6.0.77: Diseño receptor genérico

La siguiente figura muestra la configuración realizada para el bloque demodulador QAM:

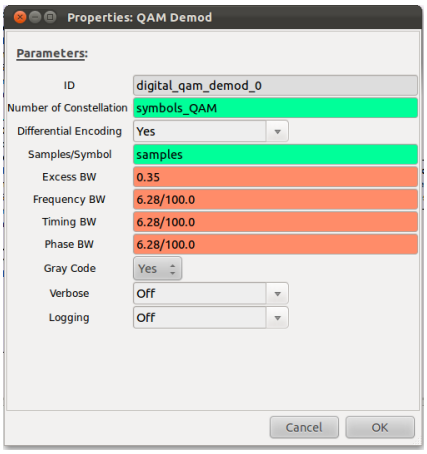


Fig 6.0.78: QAM demodulator

Nuevamente, como ocurría con el modulador QAM mostrado en el apartado anterior, hay un mayor nivel de abstracción en el diseño del bloque de procesado siendo trasparente gran parte de la configuración interna. Como parámetros de configuración, se ha de indicar el número de muestras por símbolo, el factor de roll-off de los filtro los *bandwith* de los filtros que llevarán a cabo el compensado en cuanto a tiempo, fase y frecuencia y la codificación deseada ya sea tipo Gray y/o diferencial. Los parámetros restantes sirven para ofrecer información al diseñador del modulador utilizado.

La siguiente figura muestra la configuración realizada para el bloque demodulador PSK:

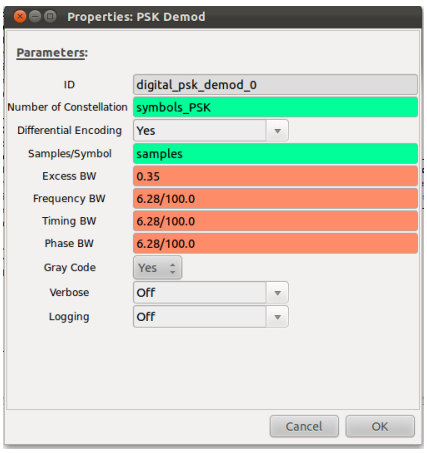


Fig 6.0.79: PSK demodulator

Tal y como se ha mencionado anteriormente, las diferencias del demodulador PSK son mínimas con respecto al demodulador QAM (en los moduladores ocurre lo mismo), las diferencias residen en el índice de modulación permitido y en cuanto a la creación de la constelación (proceso transparente al diseñador utilizando este bloque). Por lo tanto, la configuración será idéntica al caso QAM

La siguiente figura muestra la configuración realizada para el bloque demodulador FSK:

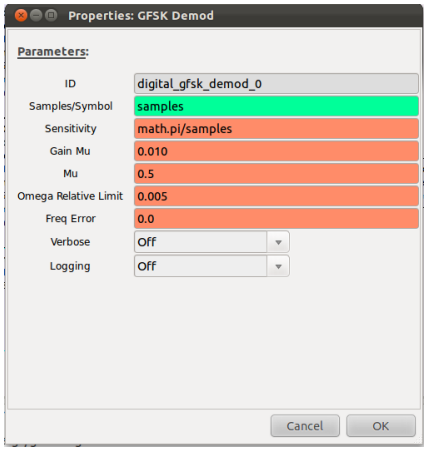


Fig 6.0.80: FSK demodulator

Al bloque demodulador, se le indicará por parámetros el número de muestras por símbolo y la sensibilidad del demodulador, parámetro que ha de coincidir con el utilizado en el modulador.
El resto de parámetros son los utilizados para llevar a cabo compensaciones en cuanto a tiempo, fase y frecuencia.

6.1.4.3 Resultados obtenidos

Para las pruebas realizadas, se supondrá una situación ideal en la que no haya presencia de ruido, para hacer posible la visualización de las figuras que representan la señal transmitida/recibida en distintos puntos de la cadena, tal y como ocurría en el apartado anterior, se ha recurrido al uso sumideros gráficos a lo largo del flujo los cuales se encargarán de mostrar el espectro o bien la constelación.

6.1.4.3.1 Sistema 4-QAM

Inicialmente se muestra el espectro de la señal generada, este espectro procede de visualizar la señal que transmite la tarjeta de sonido. Esta señal ha sido amplificada mediante el uso del bloque multiplicador:

Fig 6.0.81: Espectro de la señal generada para 4-QAM

La siguiente imagen muestra la señal a la salida del modulador:

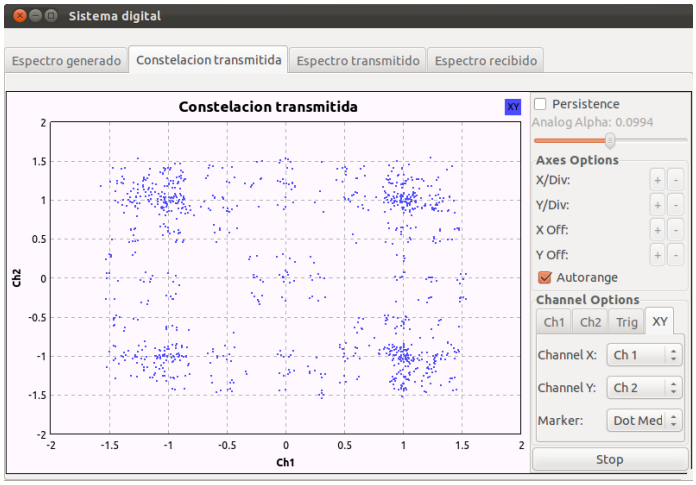


Fig 6.0.82: Constelación 4-QAM transmitida

Tal y como muestra la figura anterior, la constelación transmitida coincide con la figura 6.0.21 que se muestra en el apartado 6.1.1.3.1 tal y como era de esperar

A continuación se muestra el espectro de la señal transmitida (a la salida del modulador):

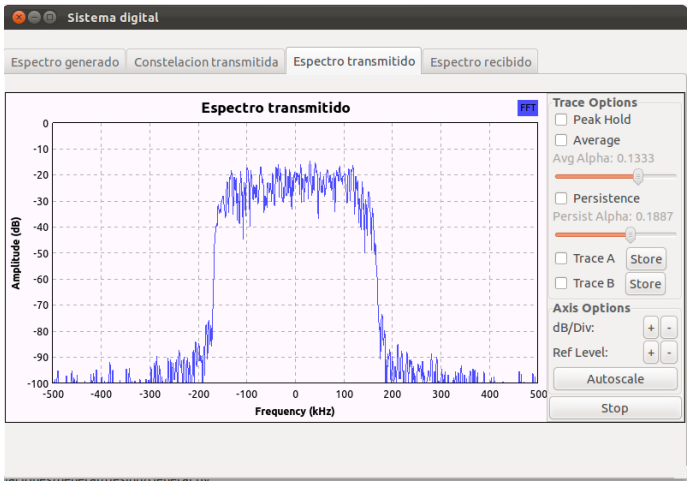


Fig 6.0.83: Espectro transmitido 4-QAM

La imagen anterior muestra el espectro de la señal modulada, tal y como se había mencionado anteriormente la modulación es llevada a cabo en banda base.

Debido a que no se puede visualizar la señal en los pasos intermedios del proceso de demodulación y que al trabajar en entorno de simulación, se ha supuesto una situación ideal en donde no existe la presencia de ruido, se presenta directamente la señal demodulada:

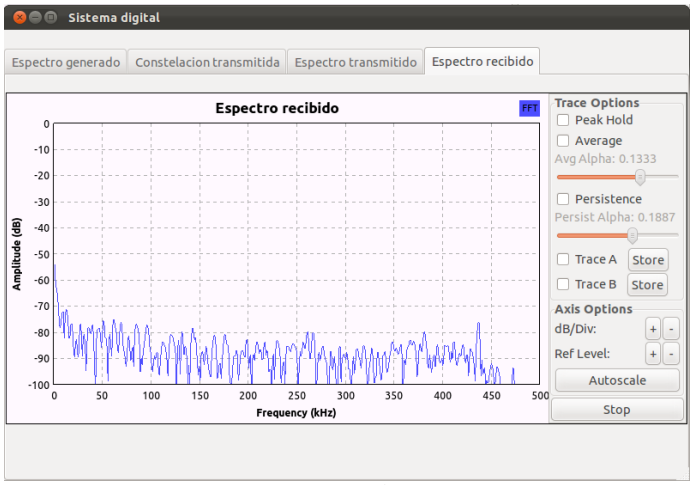


Fig 6.0.84: Espectro de la señal recibida 4-QAM

Esta es la figura que ilustra que el proceso se ha llevado correctamente, de no haber sido así, si se hubiera producido errores, el bloque *packet decoder* no hubiera devuelto datos y esta imagen quedaría completamente en blanco. Cabe destacar, que si existe alguna diferencia entre los espectros de la señal generada y recibida se debe a la variabilidad de la señal y a los diferentes instantes de capturas.

6.1.4.3.2 Sistema 4-PSK

Inicialmente se muestra el espectro de la señal generada, este espectro procede de visualizar la señal que transmite la tarjeta de sonido. Esta señal ha sido amplificada mediante el uso del bloque multiplicador:

Fig 6.0.85: Espectro de la señal generada para 4-PSK

La siguiente imagen muestra la señal a la salida del modulador:

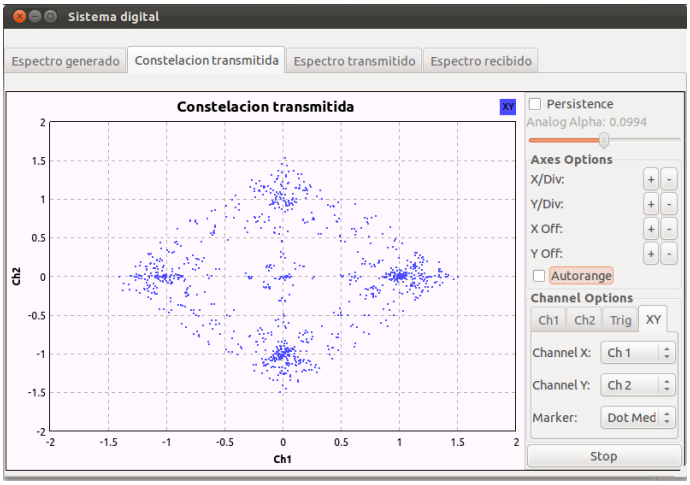


Fig 6.0.86: Constelación 4-PSK transmitida

Tal y como muestra la figura anterior, la constelación transmitida coincide con la figura 6.0.49 que se muestra en el apartado 6.1.2.3.1 tal y como era de esperar.

A continuación se muestra el espectro de la señal transmitida (a la salida del modulador):

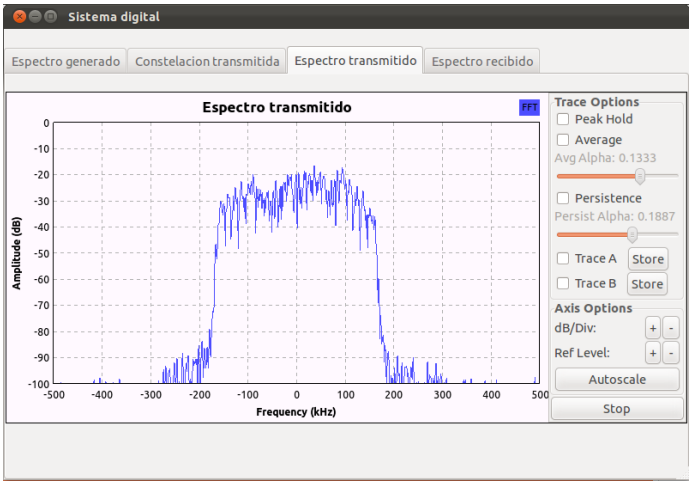


Fig 6.0.87: Espectro transmitido 4-PSK

La imagen anterior muestra el espectro de la señal modulada, tal y como se había mencionado anteriormente la modulación es llevada a cabo en banda base.

Debido a que no se puede visualizar la señal en los pasos intermedios del proceso de demodulación y que al trabajar en entorno de simulación, se ha supuesto una situación ideal en donde no existe la presencia de ruido, se presenta directamente la señal demodulada:

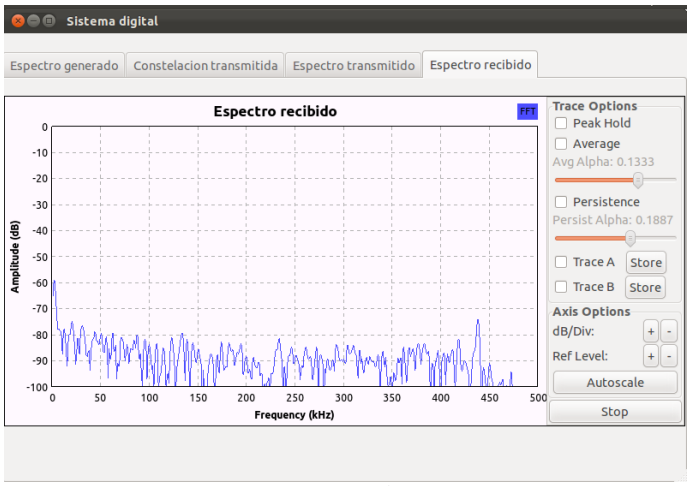


Fig 6.0.88: Espectro de la señal recibida 4-PSK

Tal y como se puede apreciar, las imágenes transmitida y recibida guardan gran similitud concluyendo con que el proceso de modulación y demodulación ha sido llevado correctamente.

6.1.4.3.3 Sistema FSK

Inicialmente se muestra el espectro de la señal generada, este espectro procede de visualizar la señal que transmite la tarjeta de sonido. Esta señal ha sido amplificada mediante el uso del bloque multiplicador:

Fig 6.0.89: Espectro de la señal generada para FSK

La siguiente imagen muestra la señal a la salida del modulador:

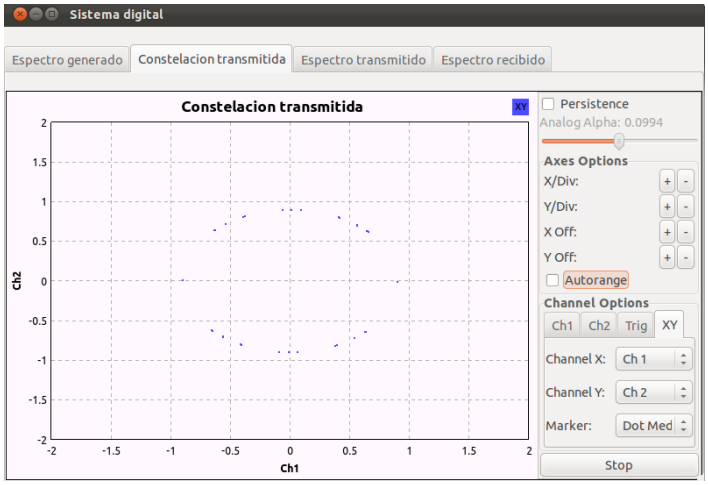


Fig 6.0.90: Constelación FSK transmitida

A continuación se muestra el espectro de la señal transmitida (a la salida del modulador):

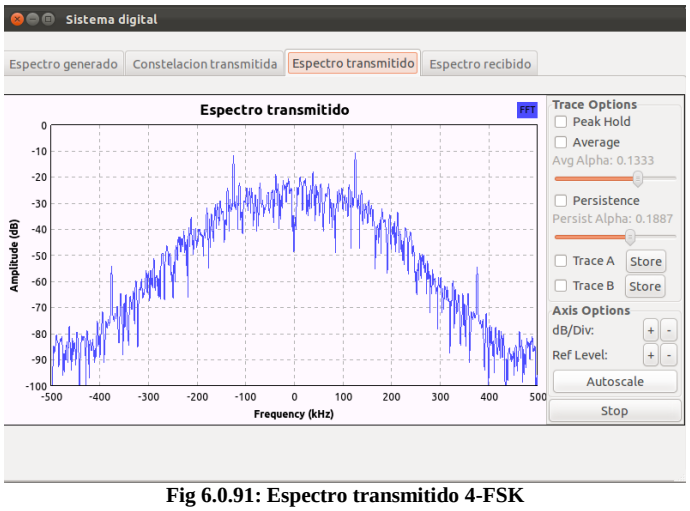


Fig 6.0.91: Espectro transmitido 4-FSK

La imagen anterior muestra el espectro de la señal modulada, tal y como se había mencionado anteriormente la modulación es llevada a cabo en banda base.

Debido a que no se puede visualizar la señal en los pasos intermedios del proceso de demodulación y que al trabajar en entorno de simulación, se ha supuesto una situación ideal en donde no existe la presencia de ruido, se presenta directamente la señal demodulada:

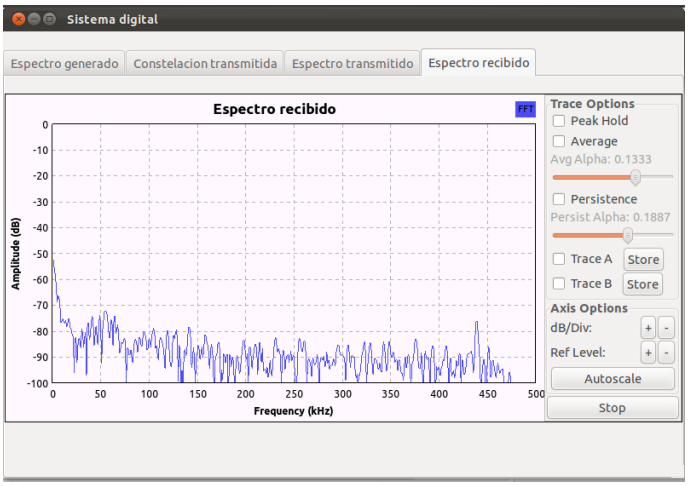


Fig 6.0.92: Espectro de la señal recibida 4-QAM

Puesto que se ha demostrado la fiabilidad de los bloques disponibles en la herramienta, a la hora de implementar moduladores, se ha empleado el modulador G-FSK que provee GNU Radio obteniendo a la salida del sistema (el sumidero gráfico) la señal correctamente demodulada.

6.2 Efectos del canal y sincronismo

Este apartado es un punto clave a la hora de migrar del entorno de simulación al mundo físico (utilizando las plataformas hardware). El porqué de trabajar en un entorno de simulación viene dado por la facilidad que este entorno presenta a la hora de estudiar los diferentes efectos que se explicarán en el apartado de manera aislada, facilitando así la comprensión de estos.

El ruido es el efecto indeseado más típico existente en toda comunicación, en este capítulo se modelará el canal mostrando cómo puede llegar a afectar en la recepción de la señal.

El sincronismo es un problema clásico de las comunicaciones y juega un papel fundamental en los receptores digitales.

A estos efectos se le han de añadir otro problema clásico en comunicaciones inalámbricas: el multitrayecto.

Para poder estudiar todos estos efectos aisladamente, se ha utilizado el siguiente bloque de procesado de señal, el cuál modela al canal:

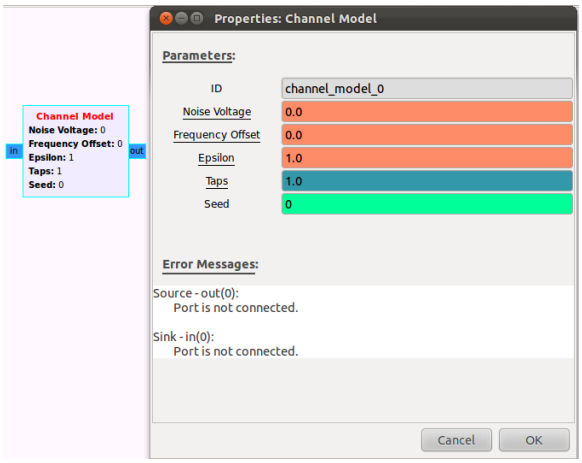


Fig 6.0.93: Propiedades del bloque channel model

Dónde:

- Noise Voltage representa el nivel de ruido.
- Frequency Offset representa el desplazamiento en frecuencia normalizado.
- Epsilon emula la diferencia en los tiempos de muestreos del transmisor y el receptor
- Taps emula los rayos debidos al multitrayecto.

Para llevar a cabo el estudio de dichos fenómenos se ha creado un programa el cual se muestra a continuación:

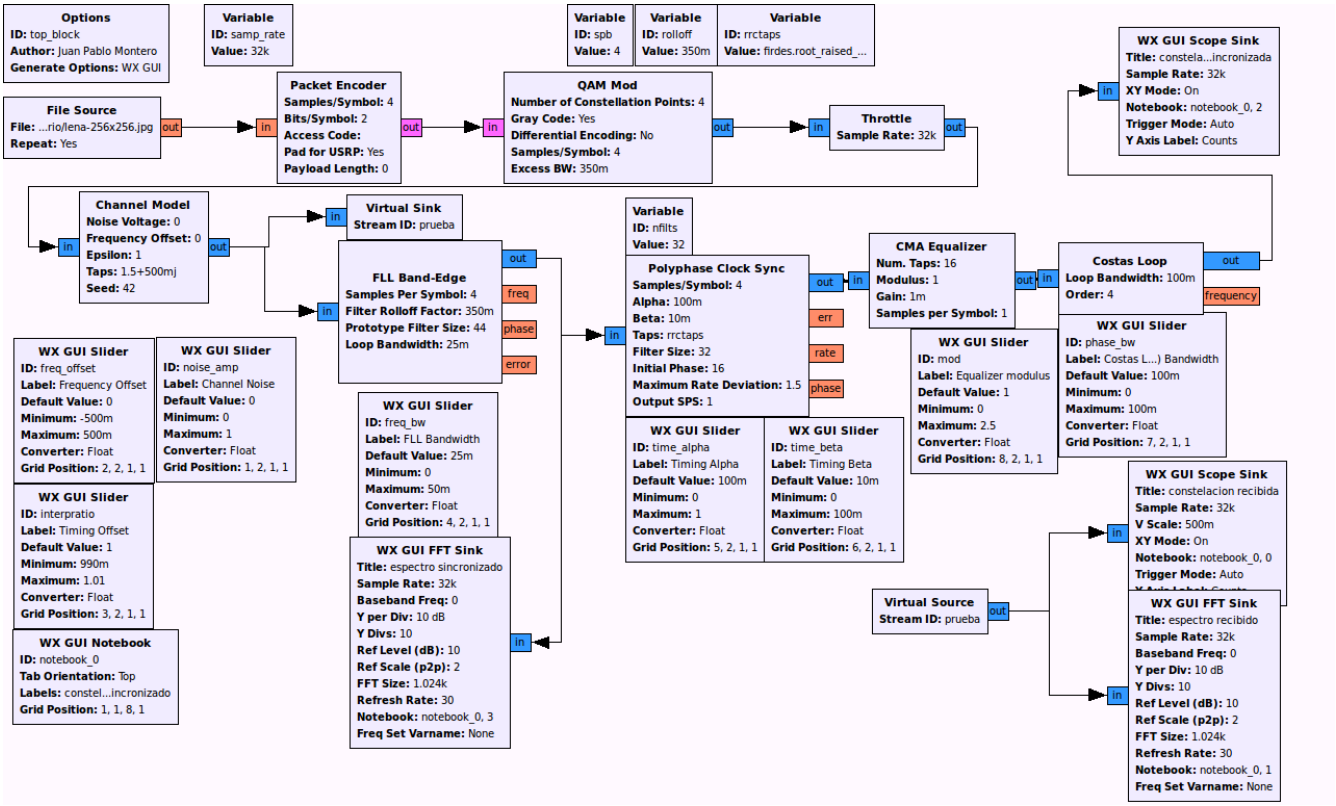


Fig 6.0.94: Diseño prueba de sincronismos

En este caso, tal y como muestra la imagen anterior, no interesa la demodulación de la señal por lo tanto se ha suprimido la última parte de la cadena quedando únicamente la parte en donde se realizan los sincronismos. En este apartado se utilizará una modulación 4-QAM para los casos de estudio.

6.2.1 Ruido

Este apartado tiene por objetivo mostrar los efectos que causa el ruido en la recepción de una señal. Para simular estos efectos se modificará el parámetro *Channel Noise*, el cual se encarga de establecer el valor del voltaje de la señal de ruido AWGN[46]. Con la intención de recalcar exclusivamente el efecto que produce el ruido en la señal recibida, se asumirá que la señal recibida está correctamente sincronizada de tal forma que sólo se mostrará su efecto sobre la señal sincronizada.

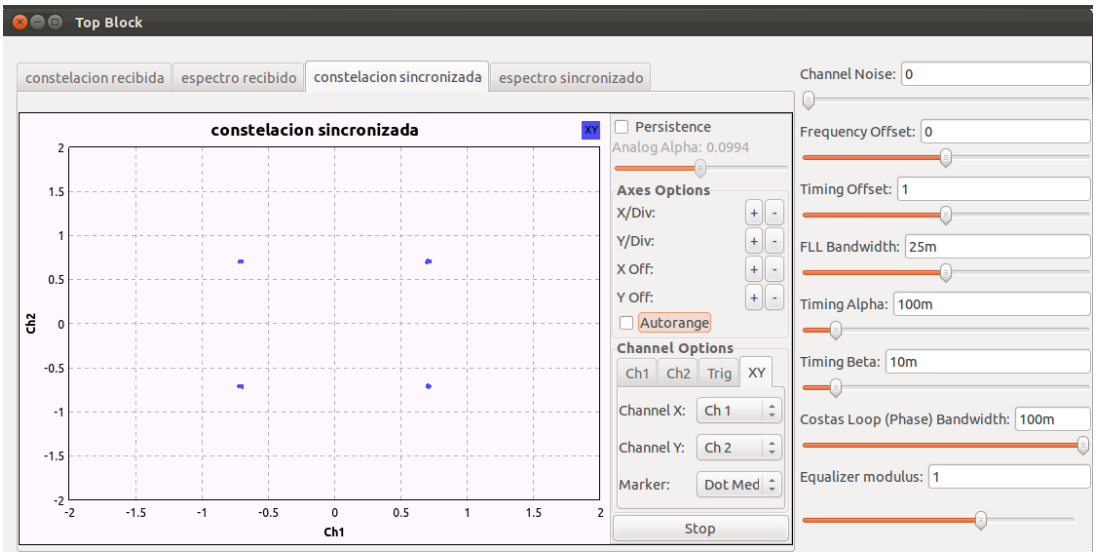


Fig 6.0.95: Constelación sincronizada sin ruido

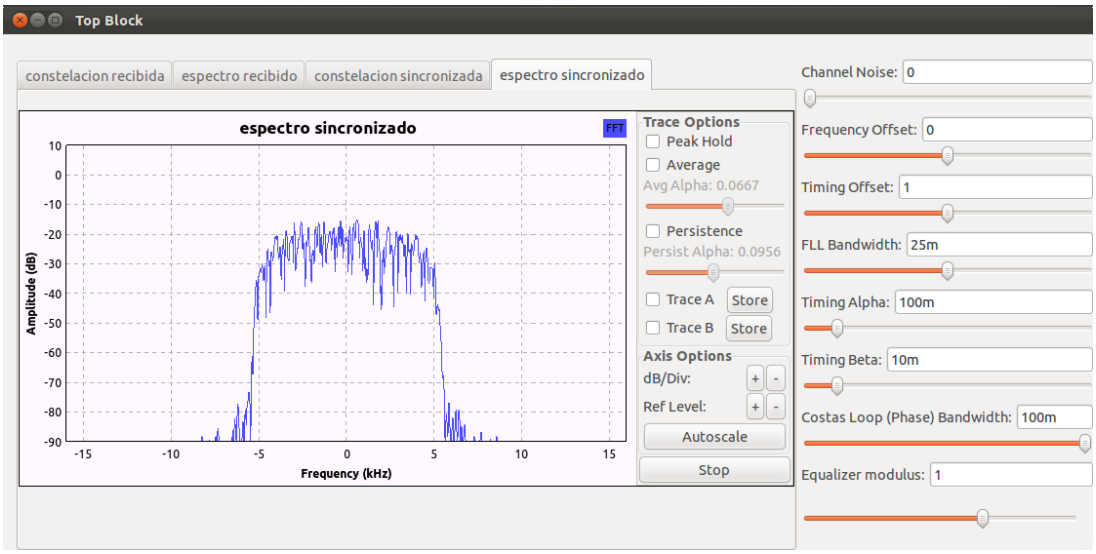


Fig 6.0.96: Espectro sincronizado sin ruido

[46] AWGN: Additive White Gaussian Noise

Las figuras mostradas, representan la señal (diagrama de constelación) y el espectro que se recibe en el ordenador en un enlace con una modulación 4-QAM, en donde no hay presencia de ruido.
A continuación se muestra el efecto que tiene la adición de ruido en la señal sincronizada:

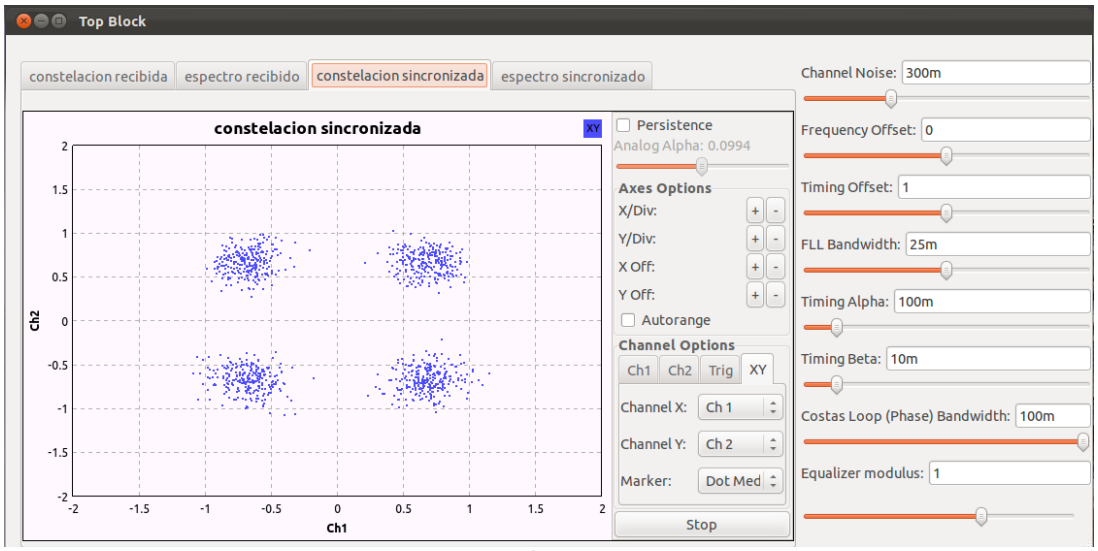


Fig 6.0.97: Constelación sincronizada con ruido

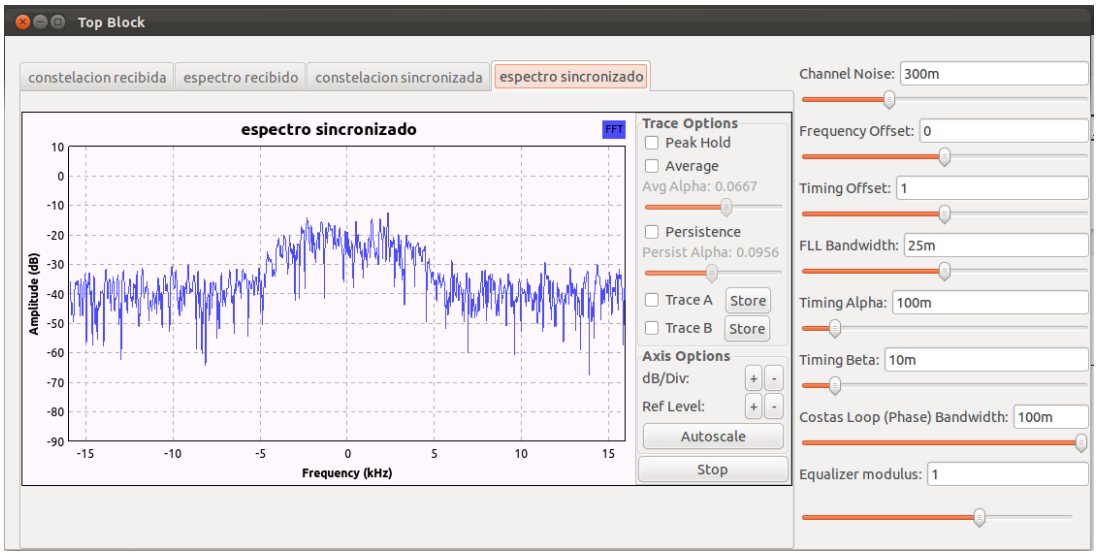


Fig 6.0.98: Espectro sincronizado con ruido

Tal y como muestran las imágenes anteriores, si el nivel de ruido es suficientemente elevado, producirá una mala interpretación de los símbolos y por consiguiente la pérdida de comunicación.

6.2.2 Sincronismos

Este apartado tiene por objetivo el estudio del sincronismo en los receptores digitales. Se distinguen tres tipos de sincronismos:

- Sincronización de paquete
- Sincronización de portadora

- Recuperación de reloj

6.2.2.1 Sincronización de paquete

La sincronización de paquete tiene por objetivo indicar al receptor cuando comienza la transmisión, para ello se utiliza un código prefijado llamado *preamble* al inicio de la transmisión.
El bloque encargado de esta función es el *packet encoder/decoder*.
Este bloque, introduce un preamble de 16 bits (0xA4F2) además de introducir un código de acceso mediante el cual se controla que no se propague la salida si ocurren más errores del prefijado, esto se realiza mediante el *threshold.* Por defecto, esto ocurrirá si ocurren más de 12 errores en el código de acceso cuya longitud es de 64 bits (0xACDDA4E2F28C20FC).
El paquete estará conformado por el *preamble* seguido del *access code*, de la longitud, del payload y finalmente del código crc[47] (aparte de los datos a transmitir)

6.2.2.2 Sincronización de portadora

La sincronización de portadora tiene por objetivo compensar las diferencias en cuanto a fase y frecuencia entre los osciladores del transmisor y receptor así como los desplazamientos que haya sufrido la propia portadora debido a su propagación por el medio. Por lo tanto este tipo de sincronismo juega un papel importante en la demodulación coherente. Con motivo de simplificar el estudio, se estudiará este tipo obviando el resto de sincronismos.

[47] Crc: Cyclic Redundancy Check, se trata de un código de detección de errores.

6.2.2.2.1 Sincronización de frecuencia

Este apartado tiene por objetivo mostrar los efectos que causa el desplazamiento en frecuencia en la recepción de una señal. Para simular estos efectos se modificará el parámetro *Frecuency Offset*, el cual se encarga de establecer el desplazamiento de la frecuencia:

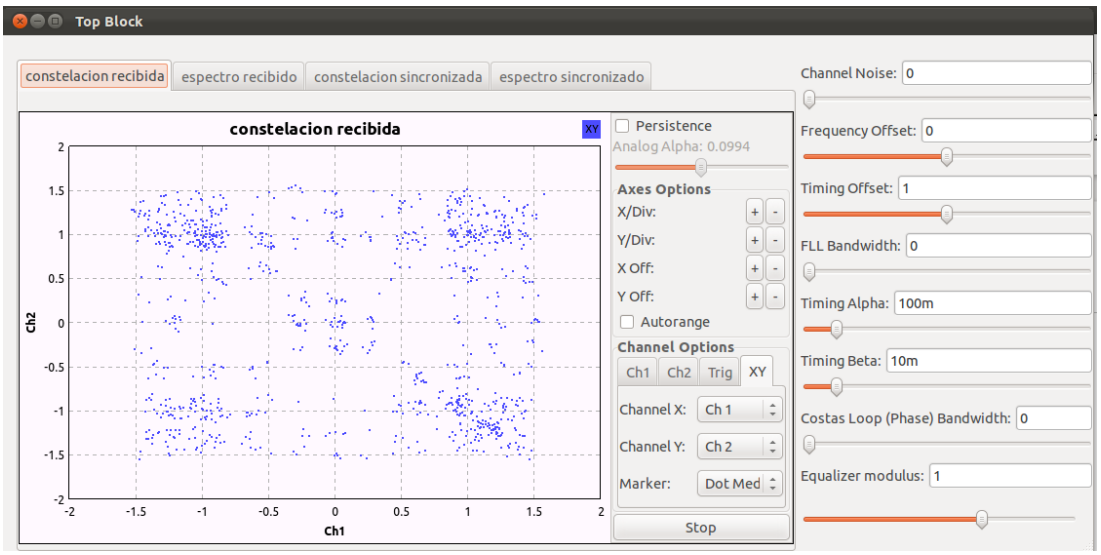


Fig 6.0.99: Constelación presincronizada en condiciones ideales

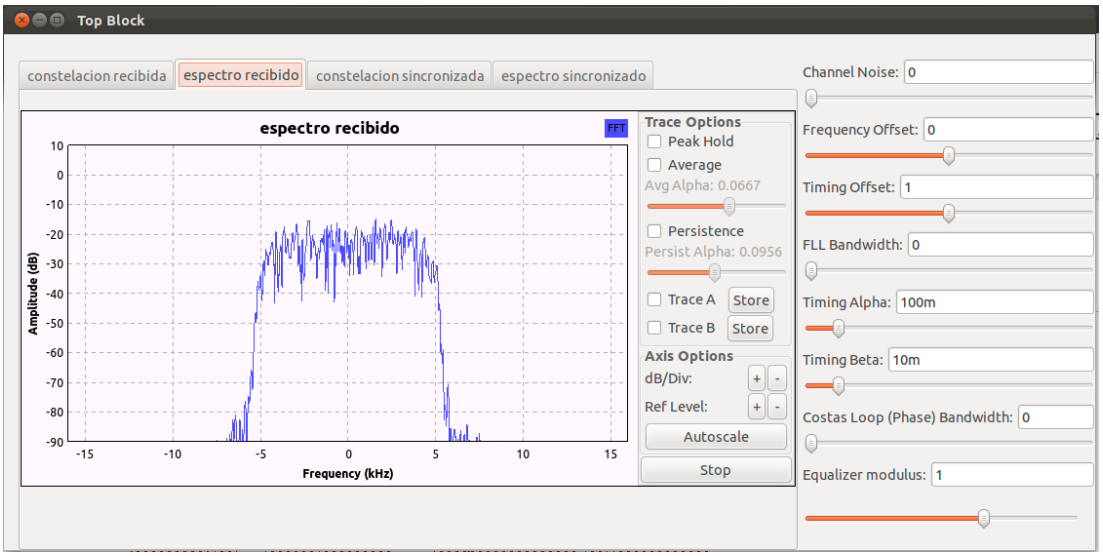


Fig 6.0.100: Espectro presincronizado en condiciones ideales

Las figuras mostradas, representan la señal (diagrama de constelación) y el espectro que llega al receptor antes de atravesar la cadena de sincronización. Cabe esperar, al darse condiciones ideales, que al no haber multitrayecto, ni problemas de sincronismo ni tan siquiera ruido la señal se recibiera correctamente sin cometer error alguno. Sin embargo, la figura 6.0.99 muestra una situación diferente a la esperada. Aunque bien es cierto, que el diagrama de constelación presentado se parece razonablemente al de una señal modulada en 4-QAM, el resultado no es en el que a priori se pensaba. Este efecto se debe al filtro adaptado, la inclusión de filtros adaptados en el sistema tiene por objetivo reducir la interferencia entre símbolos y por lo tanto hasta que no lo atraviese no se podrá apreciar el diagrama de constelación esperado.

Las siguientes imágenes muestran el efecto que tiene en la recepción la presencia de offset en frecuencia:

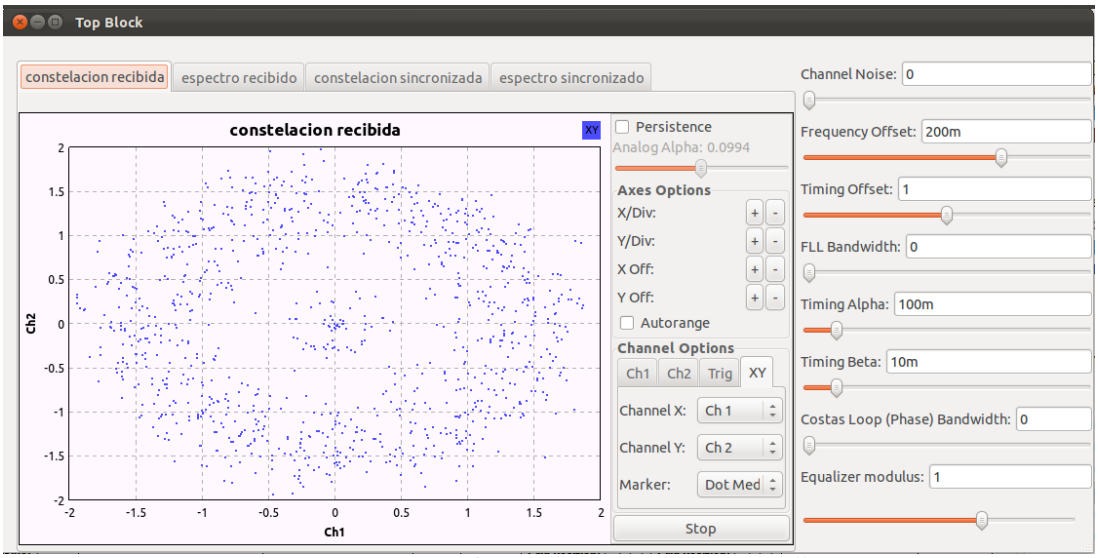


Fig 6.0.101: Constelación presincronizada con frecuency offset

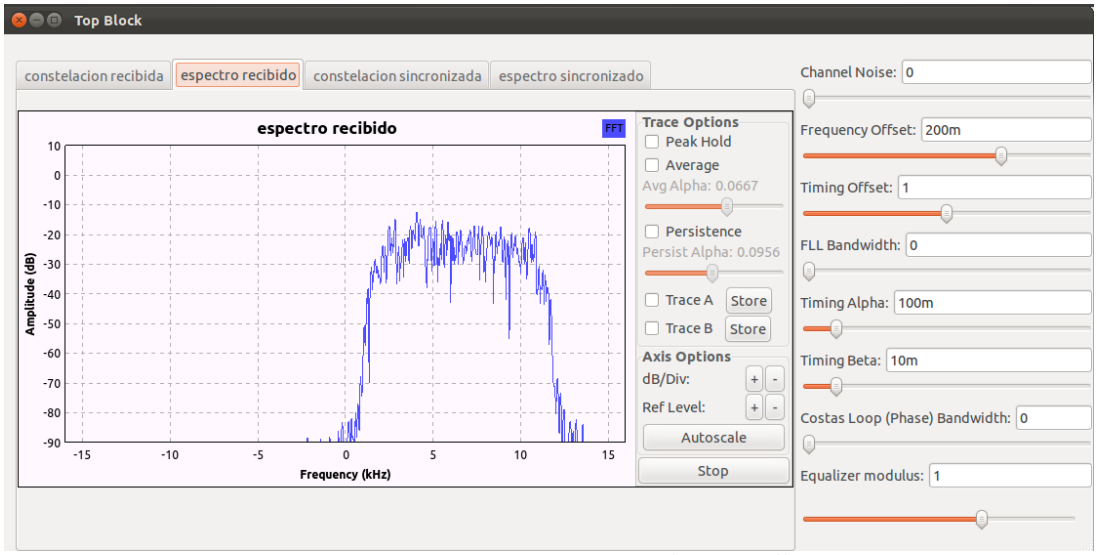


Fig 6.0.102: Espectro presincronizado con frecuency offset

Como se puede apreciar en las imágenes anteriores, al introducir un offset en frecuencia los símbolos se demodularán erróneamente (dependerá de la magnitud del desplazamiento). La manera más intuitiva de apreciar este efecto, es fijarse en el espectro recibido, el cual se puede apreciar cómo está desplazando. Para corregir este offset se ha de introducir un bloque llamado FLL Band-Edge, el cual implementa un *Frecuency Lock Loop*.

Por lo tanto ajustando el parámetro del ancho de banda de este se obtiene los siguientes resultados:

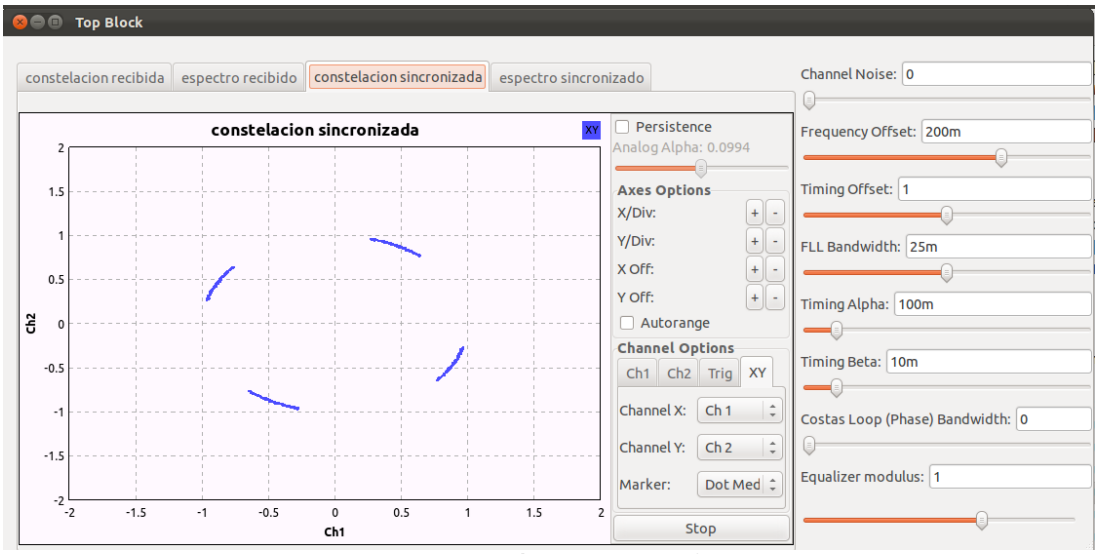


Fig 6.0.103: Constelación sincronizada en frecuencia

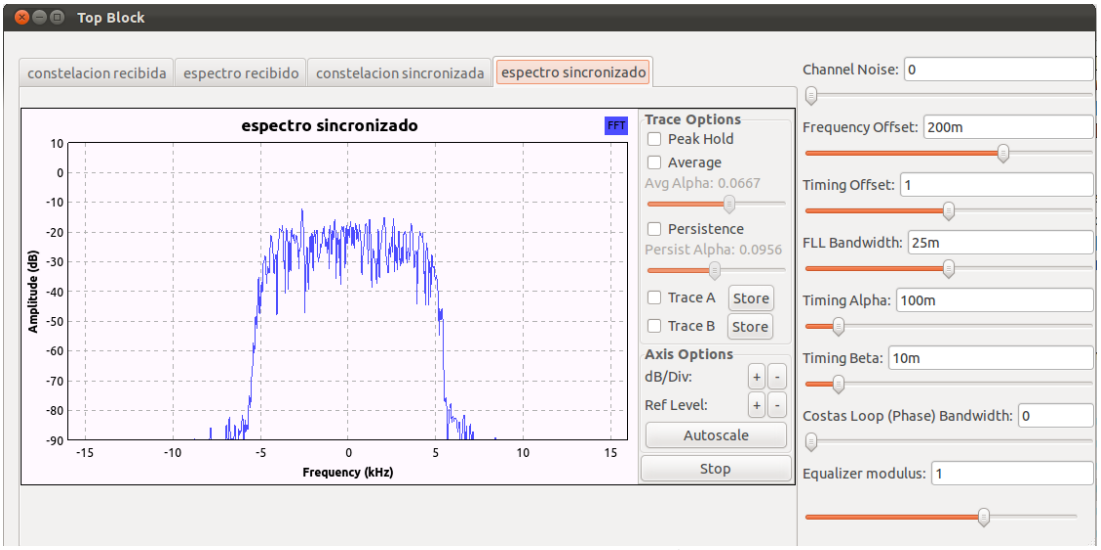


Fig 6.0.104: Espectro sincronizado en frecuencia

Las figuras anteriores muestran los efectos de introducir el bloque que implementa al *Frecuecy Lock Loop*, la figura 6.0.104 muestra como el *offset* en frecuencia se ha corregido. Sin embargo, la figura 6.0.103 muestra como la constelación rota sobre sí misma, esto se debe a un problema con las fases.

6.2.2.2.1 Sincronización de fase

Este apartado tiene por objetivo mostrar cómo se puede corregir los efectos que causa el desplazamiento en fase en la recepción de una señal, como se ha podido apreciar, el desplazamiento en fase produce rotaciones en los diagramas de constelación impidiendo una correcta demodulación de la señal.

Para subsanar este error, se introduce un bloque, el cual está basado en el algoritmo de *Costas Loop* obteniendo así los siguientes resultados:

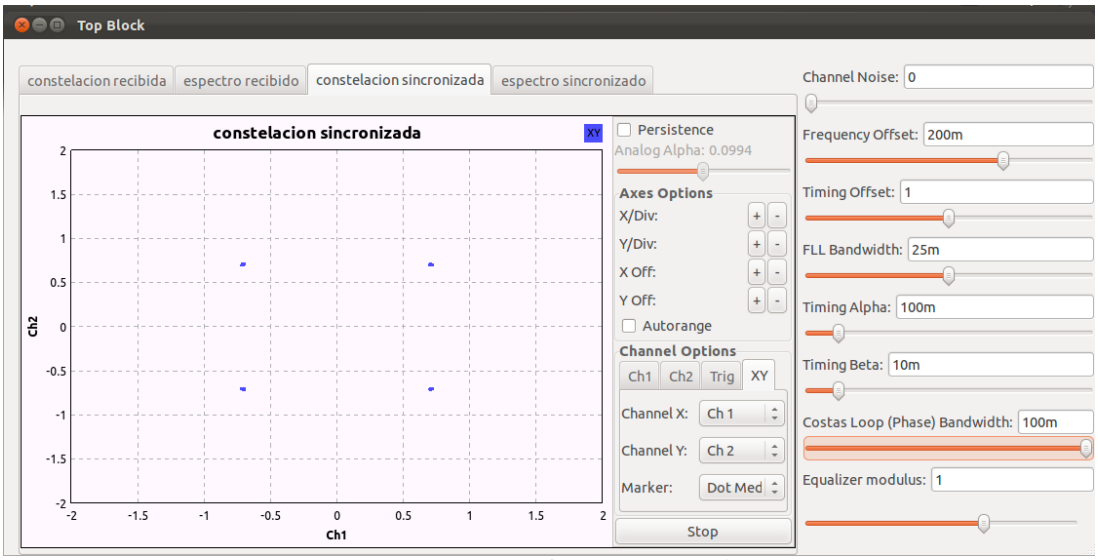


Fig 6.0.105: Constelación sincronizada en fase

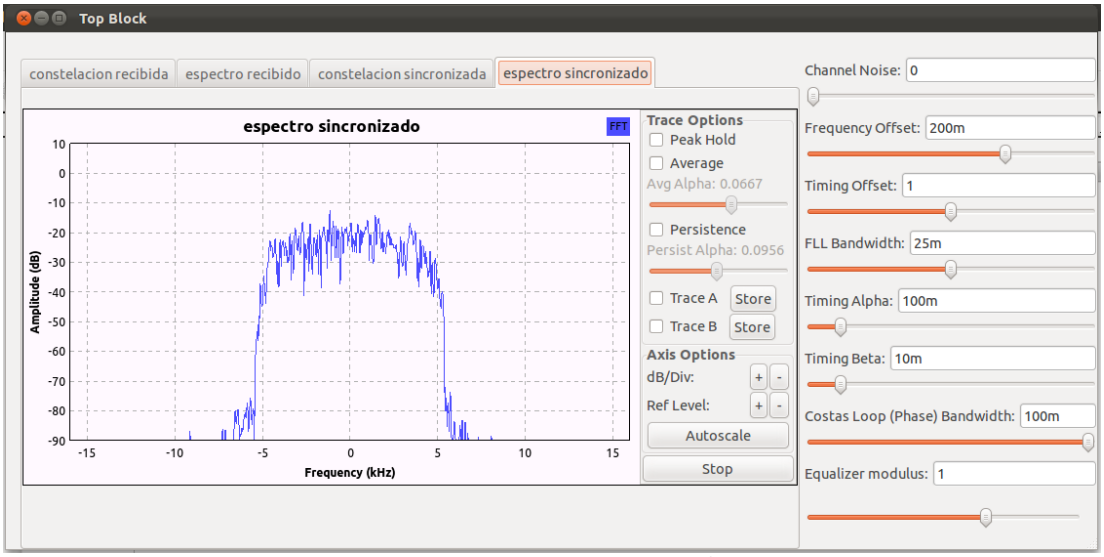


Fig 6.0.106: Espectro sincronizado en fase

Tal y como muestra la figura 6.0.105, al sincronizar la fase, se soluciona el efecto hablado en la última parte del apartado anterior, pudiendo recibir correctamente la señal. No obstante, para llegar a este resultado, se ha supuesto sincronización de reloj.

6.2.2.3 Recuperación de reloj

Este apartado tiene por objetivo mostrar los efectos que causa la diferencia en los tiempos de muestreo. Para simular estos efectos se modificará el parámetro *Epsilon*, el cual se encarga de establecer la diferencia de tiempos, con motivo de simplificar el estudio se supondrá que no hay ningún desplazamiento de fase ni frecuencia.

Inicialmente se tenía (caso ideal):

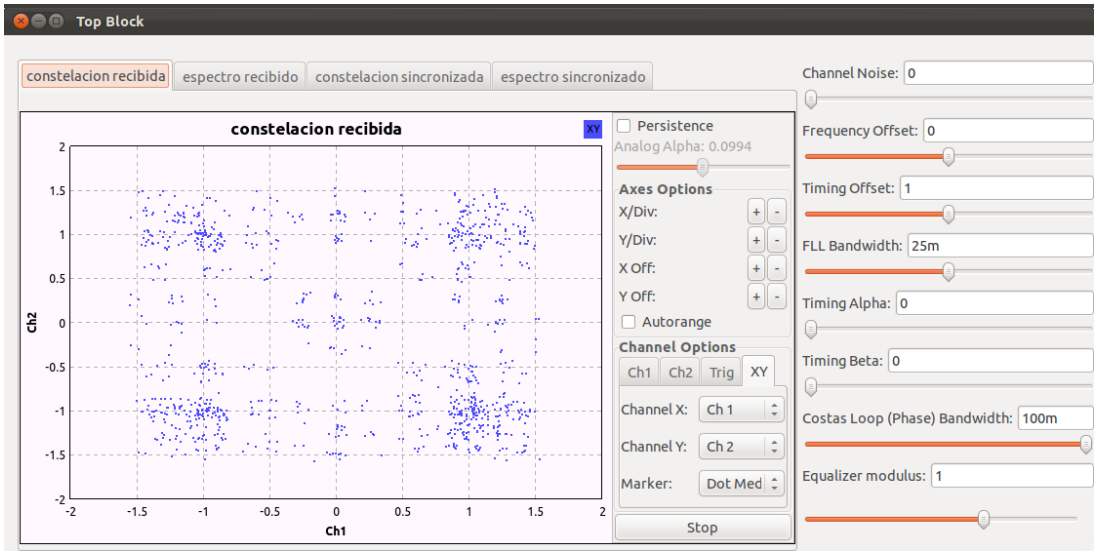


Fig 6.0.107: Constelación presoncronizada en condiciones ideales

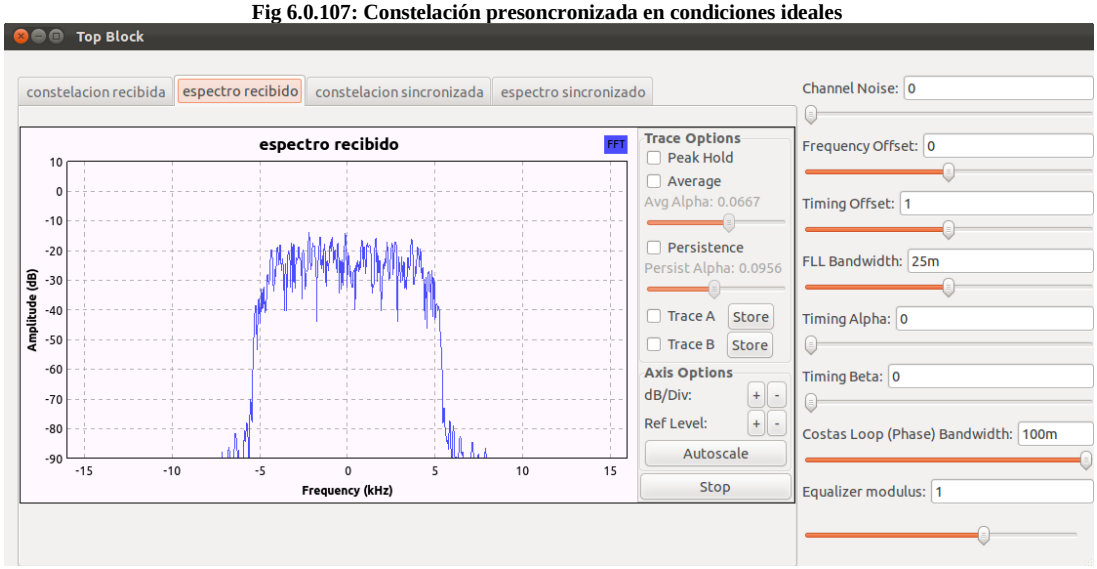


Fig 6.0.108: Espectro presincronizada en condiciones ideales

Con la intención de una mejor visualización y comprensión de este fenómeno, se ha vuelto a plasmar la constelación y el espectro recibido (antes de la cadena de sincronización).

Las siguientes imágenes muestran el efecto que tiene la diferencia en los tiempos de muestreo:

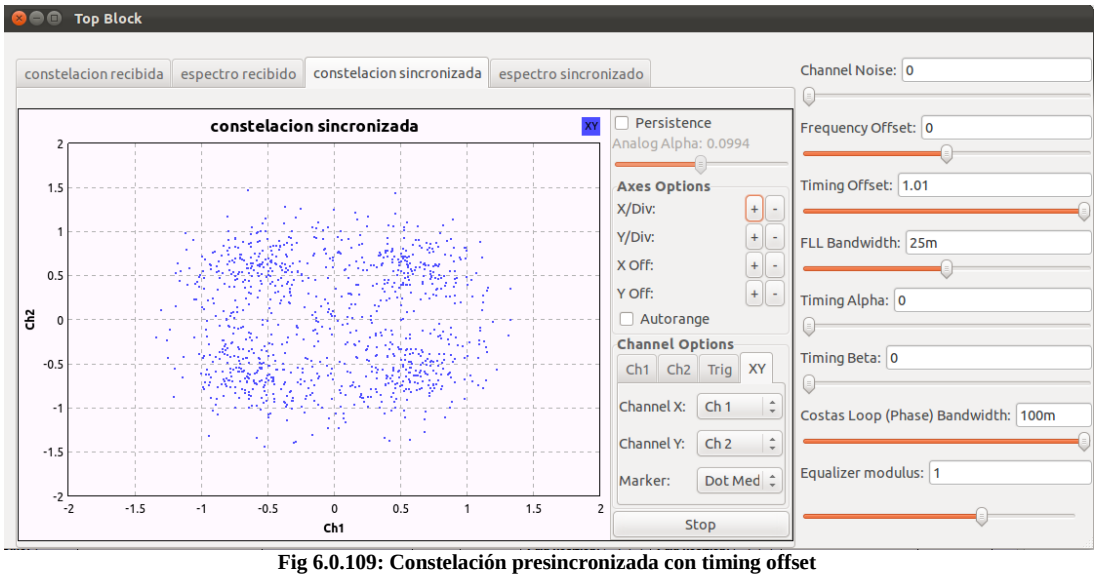


Fig 6.0.109: Constelación presincronizada con timing offset

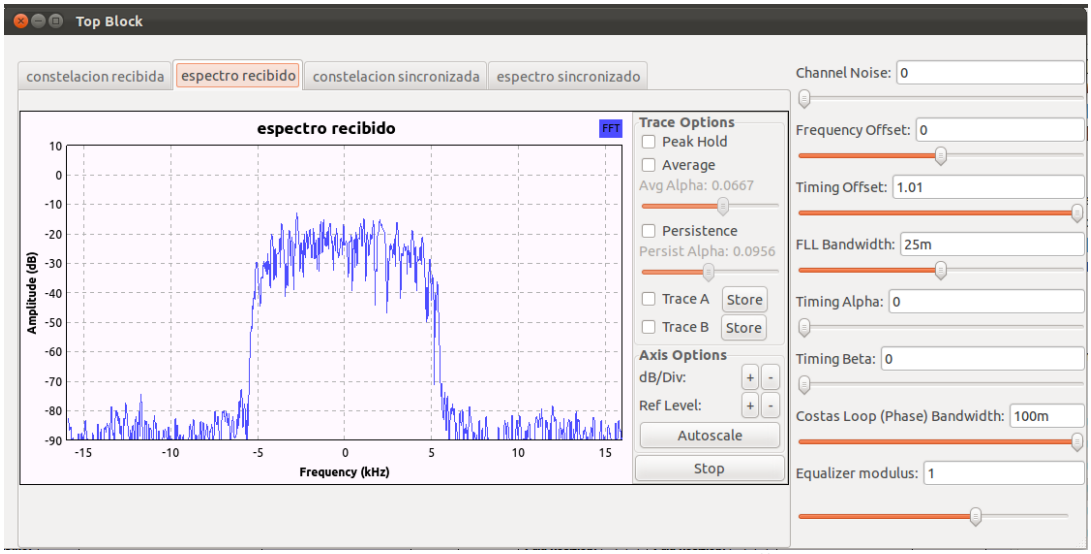


Fig 6.0.110: Espectro presincronizado con timing offset

Tal y como se muestra en las imágenes anteriores, una diferencia en los tiempos de muestreo provocan que los símbolos se demodulen erróneamente.

Para solucionar este problema, se utiliza el bloque *Poliphase Clock Sync*. Este bloque utiliza un algoritmo que implementa un lazo de control para encontrar el tiempo de muestreo exacto y así arreglar las diferencias cometidas entre transmisor y receptor. Como ya se ha mencionado anteriormente, este bloque incluye también un filtro en coseno alzado que se encargará de la ISI.

Con la correcta parametrización de los factores de ganancia del bucle de control se obtienen los siguientes resultados:

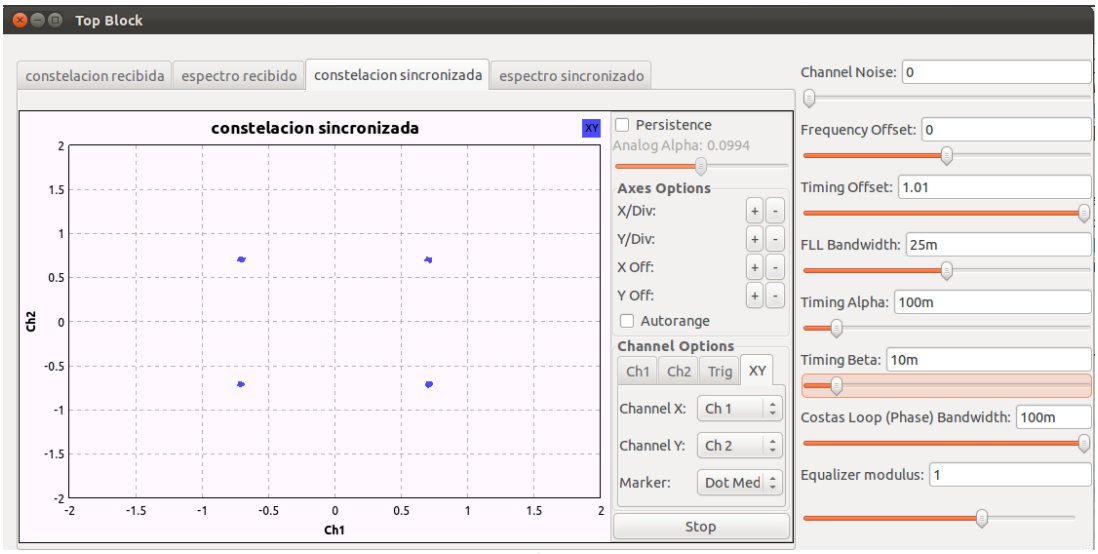


Fig 6.0.111: Constelación sincronizada en tiempo

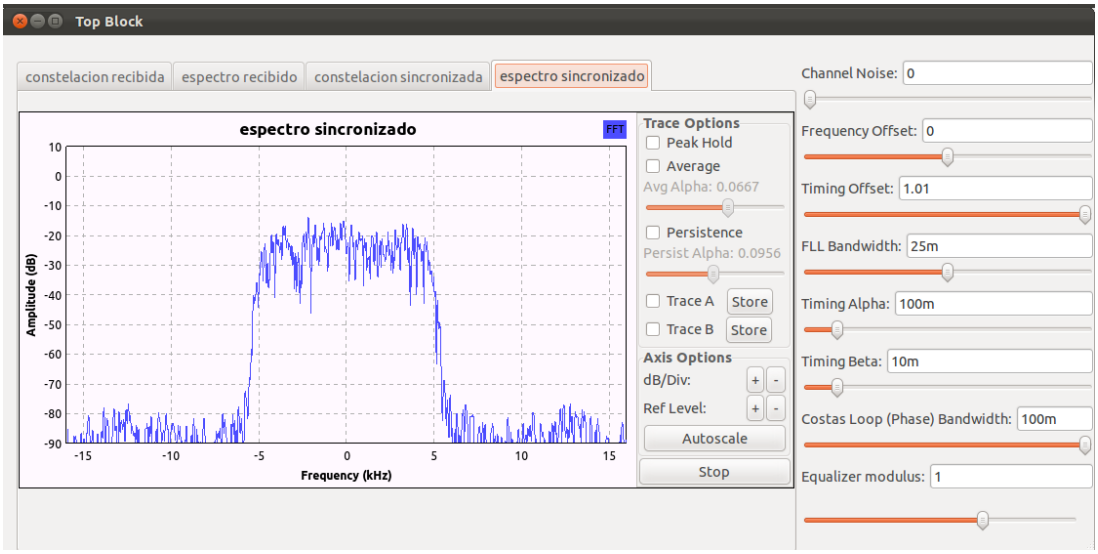


Fig 6.0.112: Espectro sincronizado en tiempo

Tal y como se puede apreciar, al introducir el bloque de procesado mencionado, las ligeras diferencias en los tiempos de muestreo son corregidas recuperando de manera correcta la señal recibida.

6.2.3 Multitrayecto

Este apartado tiene por objetivo mostrar los efectos que causa el multitrayecto. Para simular estos efectos se modificará el parámetro *Taps*, el cual se encarga de establecer los rayos.

Las siguientes imágenes han sido obtenidas para un valor del parámetro *Taps* = 1.5+0.6j.

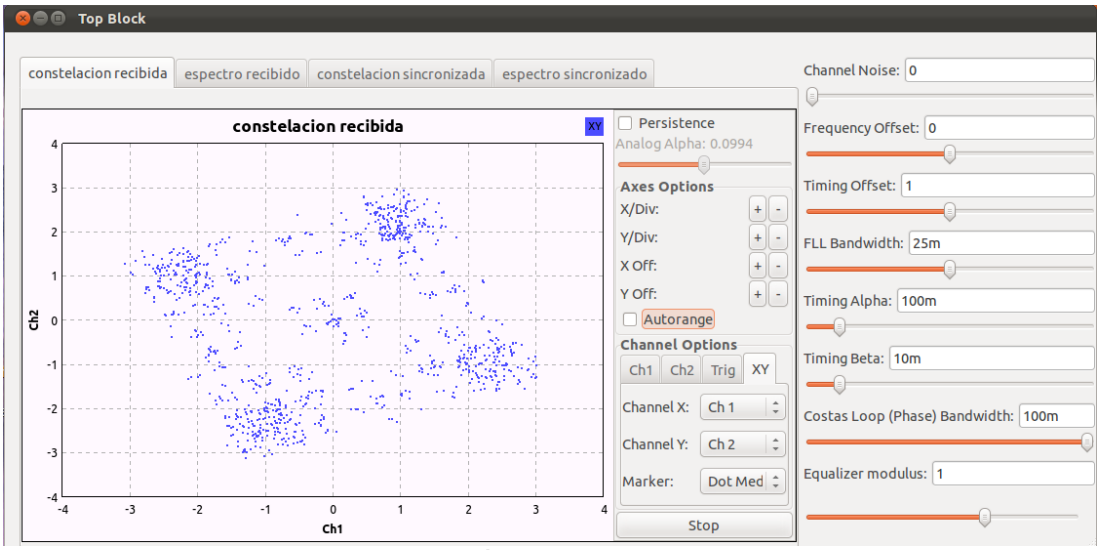


Fig 6.0.113: Constelación presincronizada con multitrayecto

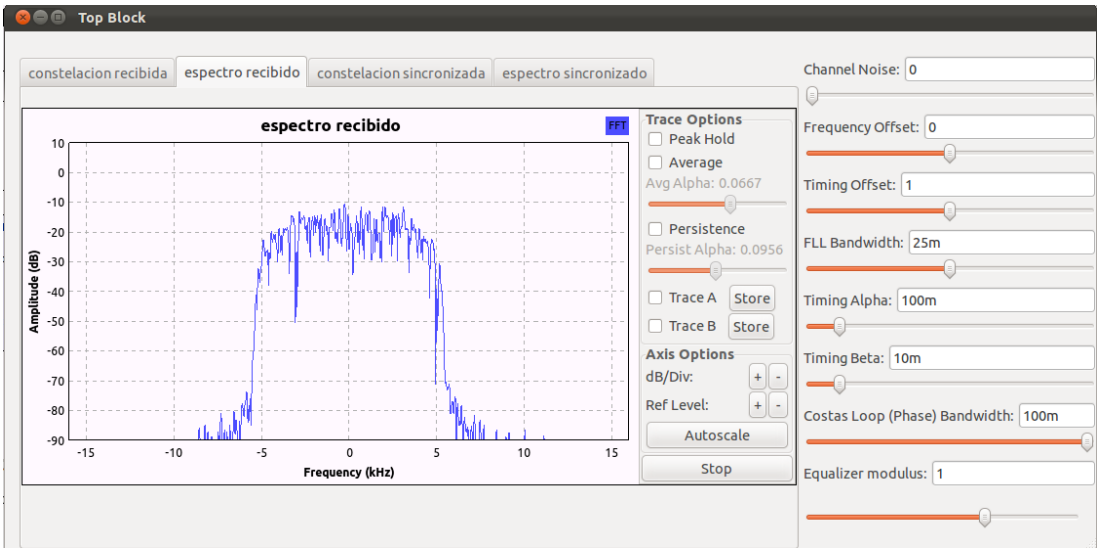


Fig 6.0.114: Espectro presincronizado con multitrayecto

Al llegar varios rayos al receptor, esto afecta directamente a la amplitud y a la fase (a no ser que estén que los rayos este en fase y/o contrafase) de la señal recibida. Por este motivo, se puede apreciar como la constelación aparece rotada y en este caso con una amplitud superior.

A continuación se muestra el efecto de los bloques *Costas Loop* y del CMA Equalizer que ocasiona a la señal afectada por el efecto del multitrayecto:

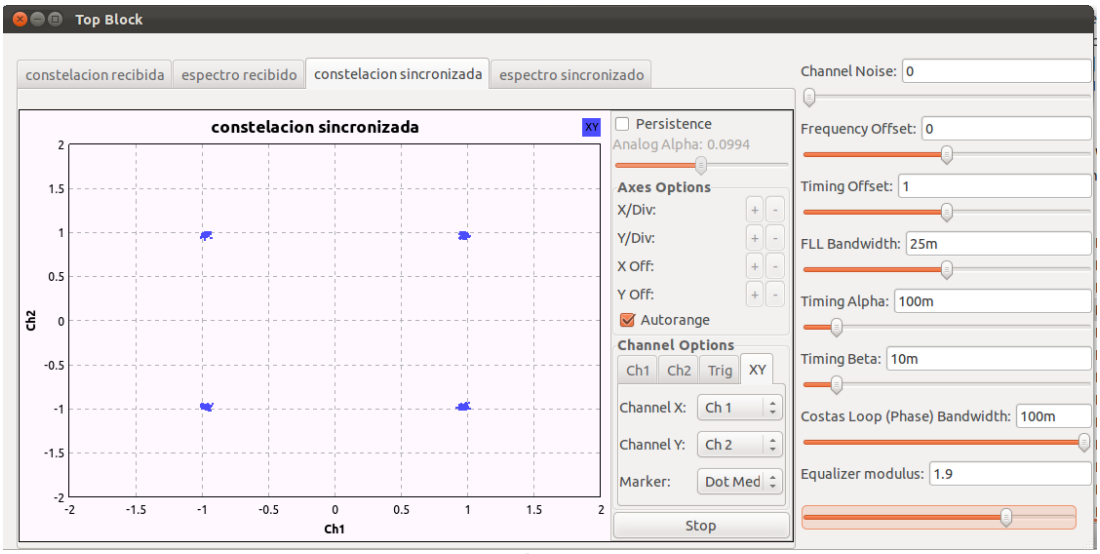


Fig 6.0.115: Constelación sincronizada y ecualizada

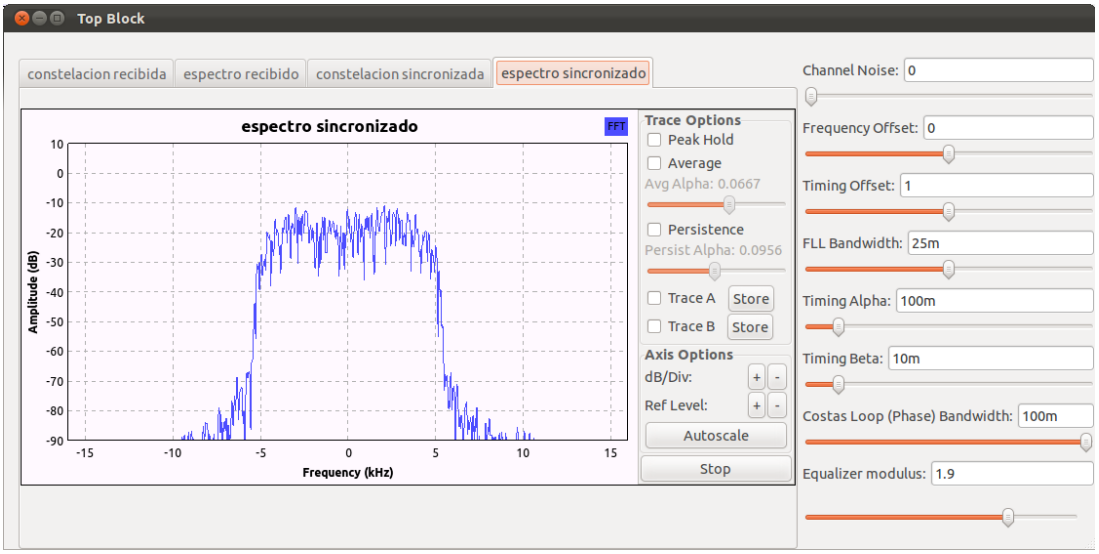


Fig 6.0.116: Espectro sincronizado y ecualizado

Adaptando adecuadamente los parámetros (modulus y la ganancia) se puede apreciar cómo se recibe adecuadamente la señal y se ha dado solución a este fenómeno.

6.3 Calculo de *Bit Error Rate*

Una vez se ha presentado varios sistemas de comunicación digitales, se ha de caracterizar la calidad de estos. Para ello, se utilizan los parámetros conocidos como *Bit Error Rate* y *Signal To Noise Ratio (SNR,* en comunicaciones digitales: *EbN0)* entre otros.

Este apartado tiene por objetivo presentar un método válido para realizar la estimación de error de bit. Para ello, el método utilizado queda definido por la siguiente fórmula:

$$BER\ (\%) = \frac{1}{N_{bits}} \sum_{n=1}^{N_{bits}} (x_{tx}[n] - x_{rx}[n]) \cdot 100$$

Ec 6.0.4: Cálculo BER

Para llevar a cabo esta prueba, se ha empleado una modulación QPSK y la transmisión de la imagen lena.jpg. Una vez se ha recibido la imagen, se ejecuta el código que se muestra en el apéndice E. Tal y como se puede apreciar en dicho apéndice, la estimación de la tasa de error de bit se ha realizado en octave debido a la facilidad que provee esta herramienta en cuanto a tratamiento de imágenes.

A continuación se muestra el esquema utilizado para la simulación del sistema, para ello únicamente se supondrá la presencia de ruido en el sistema:

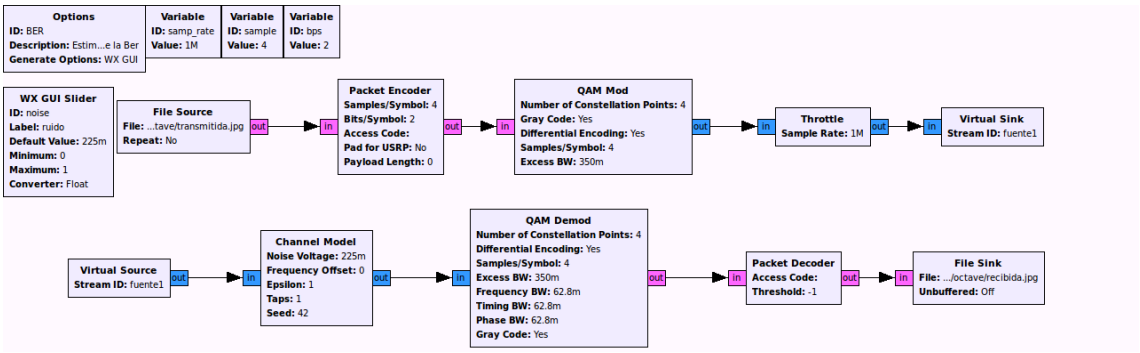


Fig 6.0.117: Esquemático para evaluación de la BER

El resultado obtenido tras la ejecución de esta simulación, servirá junto con la imagen original como parámetros de entrada del script de octave, encargándose de estimar la tasa de error. Dicho script muestra como salida las siguientes imágenes:

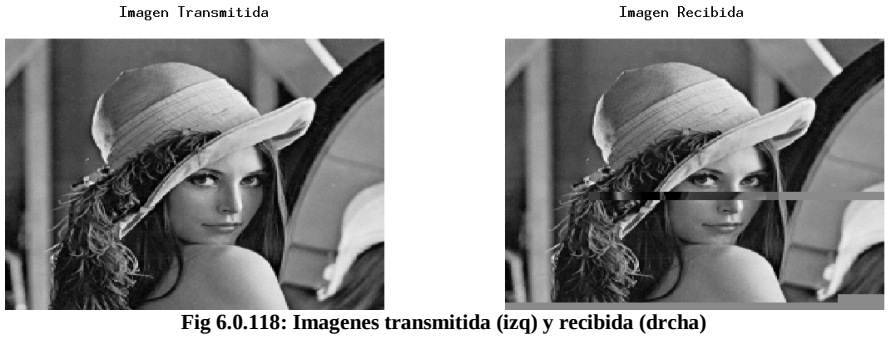


Fig 6.0.118: Imagenes transmitida (izq) y recibida (drcha)

Para visualizar el error cometido en la transmisión se muestra la imagen error como la resta entre ambas imágenes en valor absoluto:

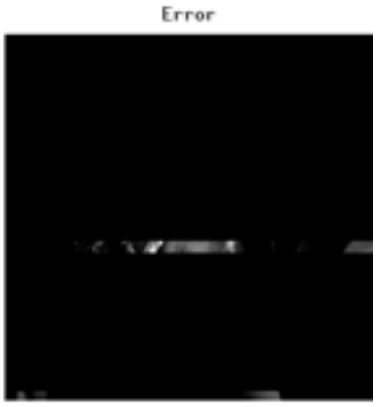


Fig 6.0.119: Imagen de error

La figura 6.0.119 muestra el error cometido en la recepción de la imagen transmitida con un valor de ruido de 0.225. Sobre esta imagen se calcula la tasa de error de bit:

```
octave:176> printf("BER: %e%%\n",biterrorrate)
BER: 8.771896e-01%
```

Fig 6.0.120: Valor de la BER

Como se puede apreciar, el error cometido en el sistema es entorno al 0.877%. Cabe destacar, que las imágenes utilizadas son definidas del tipo uint8, sus valores estarán comprendidos entre 0 y 255, y por lo tanto podrán representarse como byte en el esquema (8 bits), para realizar una estimación válida hay que descomponer el error cometido (figura 6.0.120) en valores de bits.
Por último, mencionar que este método no siempre será válido para estimar la BER. Esto es a consecuencia del bloque *packet decoder*, el cual, dejará de mostrar la salida alcanzado un determinado número de bits erróneos en el *acces code* (Véase la codificación de paquete Cap:6.2.2.1)

7
Experimentos y resultados con USRP

Este capítulo tiene por objetivo mostrar el proceso de migración del diseño de un transmisor digital en entorno de simulación a uno en donde se utilice plataformas hardware, para ello se han utilizado las placas xcvr2450 siguiendo el esquema de interconexión mostrado en la imagen 5.0.33.

Por último, con la intención de acercar la tecnología al usuario, se ha decidido desarrollar un script de tal forma, que el uso de la plataforma software quede en segundo plano y se permita realizar determinadas aplicaciones de manera muy intuitiva e interactiva con el usuario final.

7.1 Sistema genérico

En este apartado se llevará a cabo el diseño de un sistema digital formado por tres tipos de modulaciones a elegir: 4-QAM, 4-PSK y GFSK. A diferencia del anterior apartado, para realizar las pruebas se han utilizado plataformas hardware (USRP), por lo tanto se tendrá que tener en cuenta en el diseño ciertos aspectos como son los valores de las amplitudes y la tasa binaria. Con respecto a la amplitud, tal y como ya se introducía en el capítulo de las pruebas iniciales, los valores de las amplitudes impactan directamente en el voltaje de la señal enviada hacia el DAC que presenta un rango desde -1 a +1 V y por lo tanto un valor que exceda a esta magnitud (0 dB) saturará al DAC. Sin embargo, dependiendo de la placa RF utilizada, valores inferiores (pero próximos) a esta magnitud puede producir que la señal se comprima y se trabaje en regiones no lineales, efecto indeseable se corregirá añadiendo un multiplicador a la señal modulada que se envíe a la *daughterboard*.

Por otra parte, se tendrá que tener en cuenta el factor del *sample rate* Según el fabricante, el máximo *sample rate* soportado es de 25 Msps o de 50Msps según el número de bits utilizados para la representación de cada muestra (32 o 16).
En base a esto, habrá una velocidad de transmisión que tendrá que ser soportada por la conexión GbE, para ello se emplea la siguiente fórmula:

$$V\ (bps) = \mathrm{N_{bits}} \left(\frac{bits}{Sample}\right) \cdot sample\ rate\ \left(\frac{Samples}{second}\right) \leq 1 \cdot 10^9$$

Ec 7.0.1: Velocidad de transmisión para el bus de comunicación

Como previamente se ha comentado, el máximo *sample rate* estará fijado a 25MSps o a 50MSps, esto generará una velocidad de transmisión (para el caso de 32 bits por muestra) de:

$$V\ (bps) = 16\left(\frac{\text{bits}}{\text{I Sample}}\right) \cdot +16\left(\frac{\text{bits}}{\text{Q Sample}}\right) \cdot 25^6\left(\frac{Samples}{second}\right) = 8 \cdot 10^6 bps \leq 1 \cdot 10^9$$

Ec 7.0.2: Cálculo velocidad máxima de conexión Ethenet

Velocidad soportada por el estándar GbE al ser inferior al Gbps, cabe destacar que este cálculo se ha realizado para el caso de un enlace en modo de operación *half-duplex*.

De acuerdo con la tasa de Nyquist, la frecuencia de muestreo (f_s) ha de ser el doble del ancho de banda de la señal a muestrear (parte positiva del espectro). Esto significa, que la señal modulada (banda base), como máximo, tendrá un ancho de banda fs/2 (parte positiva del espectro. En realidad, al estar en banda base, estará centrada en 0 y su espectro estará comprendido entre -fs/2 y fs/2). Por otra parte, a la hora de calcular el MBW[48] de la señal, hay que tener en cuenta el número de muestras utilizadas por período (el del símbolo) de la siguiente manera:

$$f_{bb}\ (Hz)\ = \frac{fs}{2 \cdot N\ (Spsym)}$$

Ec 7.0.3: Ancho de banda de la señal a muestrear

Por lo tanto, el máximo ancho de banda de la señal (el mínimo número de muestras por símbolo es 2) será fs/4. Esto se traduce en un máximo ancho de banda de 6.25MHz para el caso de 32 bits y 12.5MHz para el caso de 16 bits (el ancho de banda mencionado describe la parte positiva del espectro).

El ancho de banda de la señal está íntimamente ligado a la velocidad de transmisión:

$$R_b\ (bps) = 2f_{bb} \cdot log_2\, M$$

Ec 7.0.4: Tasa binaria en función del ancho de banda

Donde Rb es la tasa binaria y M el índice de modulación. Sustituyendo el parámetro f_{bb} por el obtenido en la ecuación 7.03 se obtiene:

$$R_b\ (bps) = \frac{fs}{N\ (Spsym)} \cdot log_2\, M$$

Ec 7.0.5: Tasa binaria en función del parámetro sample rate

Por último, si se quiere obtener la velocidad de símbolo, la ecuación 7.05 quedará como:

$$R_s\ (baudios) = \frac{fs}{N\ (Spsym)}$$

Ec 7.0.6: Velocidad de símbolo en función del parámetro sample rate

[48] MBW: Modulation Bandwith

Con respecto al diseño del sistema, se utilizará como fuente de información una señal del tipo diente de sierra.

Por último se utilizará como *sink* un sumidero gráfico que se encargará de representar el espectro de la señal que le llega, así como sumideros gráficos auxiliares para mostrar el flujo de la señal en diversos puntos del sistema.

7.1.1 Transmisor genérico

La siguiente figura muestra el diseño llevado a cabo para un transmisor genérico, dando la posibilidad de modular la señal de información mediante M-QAM, M-PSK, BFSK. Este esquema es fruto de la migración de la figura 6.0.73 a un entorno en donde se trabaja con las plataformas hardware USRP:

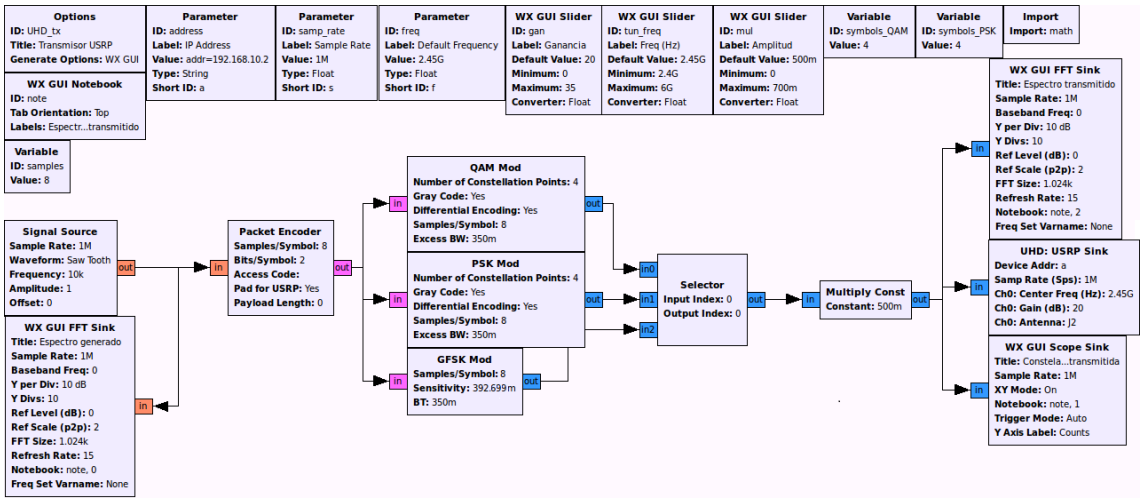


Fig 7.0.1: Diseño transmisor genérico con USRP

Estableciendo el parámetro de número de símbolos, se elegirá el índice de modulación deseado para las modulaciones QAM y PSK, mientras que para la modulación de tipo FSK el número de bits estará fijado a 1 (BFSK).

La siguiente figura muestra la configuración realizada para el bloque USRP cuando el sistema se caracteriza como a un transmisor. En este sistema, el USRP actuará como receptor con respecto al ordenador:

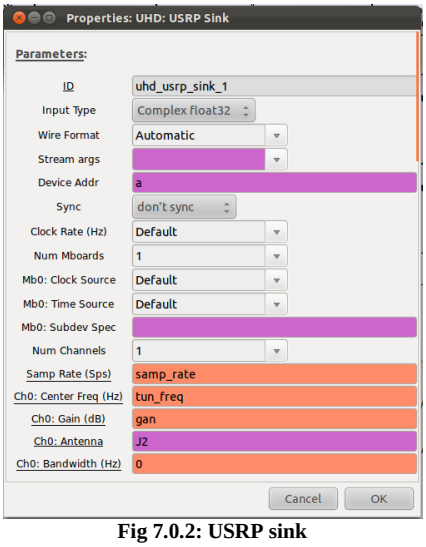


Fig 7.0.2: USRP sink

Para realizar la comunicación con periférico USRP, se ha de indicar en el diagrama del transmisor el tipo de dato que va a manejar (32 bits por muestra o en su defecto 16), y la IP del dispositivo. En caso de no introducir una dirección IP, esta empleará una por defecto (192.168.10.2). El hardware soporta la tecnología MIMO por lo tanto se puede utilizar más de un USRP a la vez, para ello se deberá establecer el número de placas madres.

Los siguientes parámetros sirven para especificar si se empleará un fuente externa o por defecto la interna para llevar a cabo la sincronización.

El parámetro *Subdev Spec* sirve para indicar el camino de la antena al conversor, este parámetro se podrá obviar siempre y cuando se trabaje con canales complejos, mientras que de trabajar con canales reales se ha de especificar (Basic RX, Basic Tx). Este último parámetro está relacionado con el número de canales, al disponer de conversores de doble canal si se opera con canales reales (Basic RX) se podrá transmitir y/o recibir dos canales.

Al periférico USRP se le ha de indicar la *sample rate* deseada, la frecuencia de operación y la antena. El parámetro de ganancia es el encargado de establecerla en la cadena transmisora (*daughterboard*). Por último, también se le ha de indicar el ancho de banda de los filtros si los hubiere, encargándose de programar el ancho de banda de los filtros.

7.1.2 Receptor genérico

La siguiente figura muestra el diseño llevado a cabo para un receptor genérico. Este esquema es fruto de la migración de la figura 6.0.77 a un entorno en donde se trabaja con las plataformas hardware USRP:

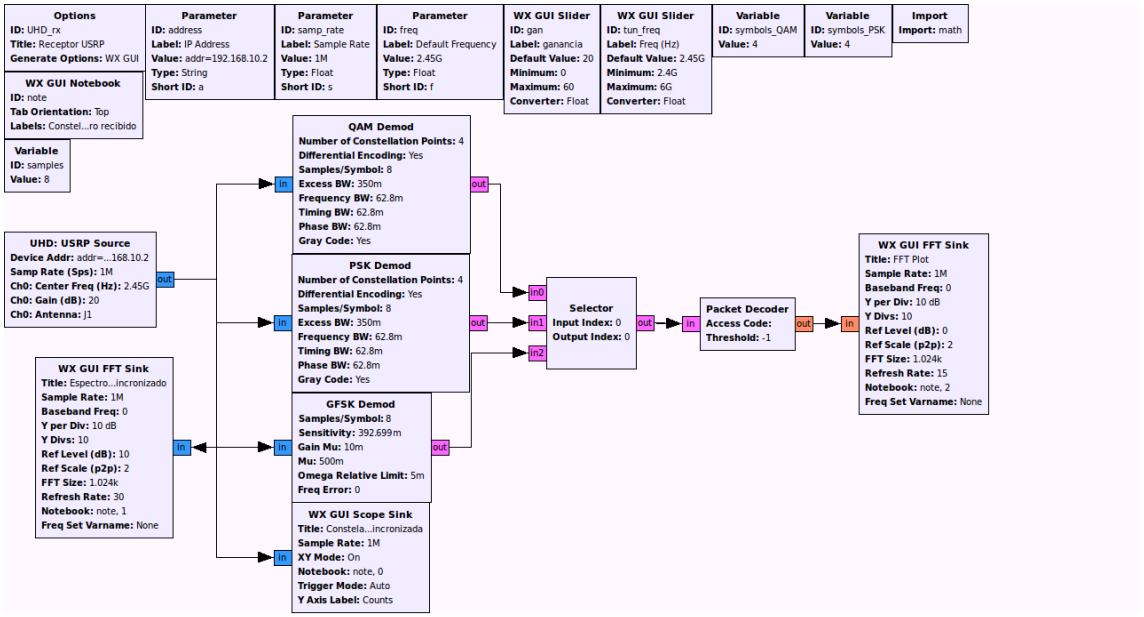


Fig 7.0.3: Diseño receptor genérico con USRP

La siguiente figura muestra la configuración realizada para el bloque USRP cuando el sistema se caracteriza como a un receptor. En este sistema, el USRP actuará como transmisor con respecto al ordenador:

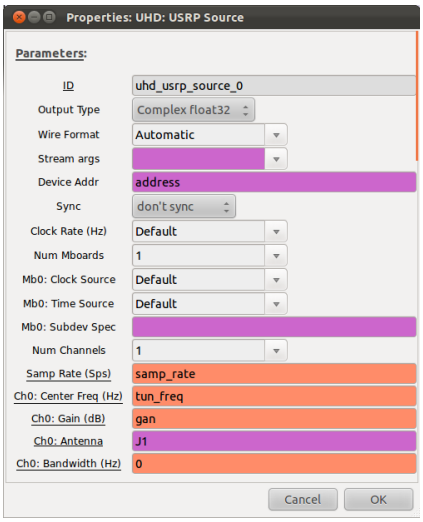


Fig 7.0.4: USRP source

Tal y como se aprecia, la configuración del bloque USRP caracterizando a un receptor (USRP *source*) es idéntica a la llevada a cabo para el transmisor (USRP *sink*)

7.1.3 Resultados obtenidos

Para las pruebas realizadas, a diferencia del capítulo anterior, ya no se podrá asumir una situación ideal, si no que aparecerán los efectos descritos en el apartado 6.2. Como previamente se ha mencionado, para hacer posible la visualización de las figuras que representan la señal transmitida/recibida en distintos puntos de la cadena, se ha recurrido al uso sumideros gráficos a lo largo del flujo los cuales se encargarán de mostrar el espectro o bien la constelación.

7.1.3.1 Sistema 4-QAM

Inicialmente se muestra el espectro de la señal generada, esta imagen se corresponde al espectro de una señal triangular de frecuencia 10KHz:

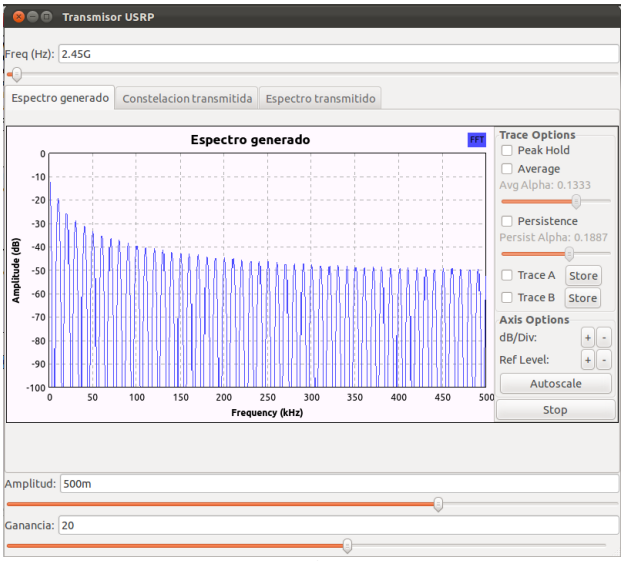


Fig 7.0.5: Espectro de la señal generada para 4-QAM

Una vez la señal atraviesa la cadena moduladora, la señal que se obtiene a la salida es la siguiente:

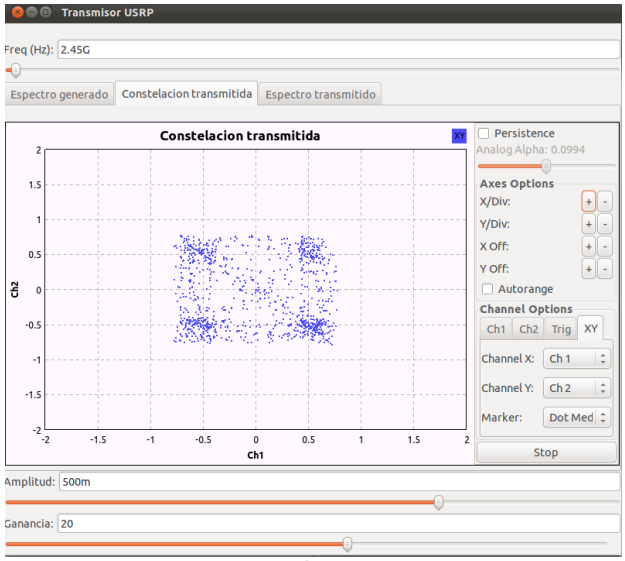


Fig 7.0.6: Constelación 4-QAM transmitida

Tal y como muestra la figura anterior, la constelación transmitida coincide con la figura 6.21 que se muestra en el apartado 6.1.1.3.1 tal y como era de esperar.

A continuación se mostrará el espectro de la señal transmitida:

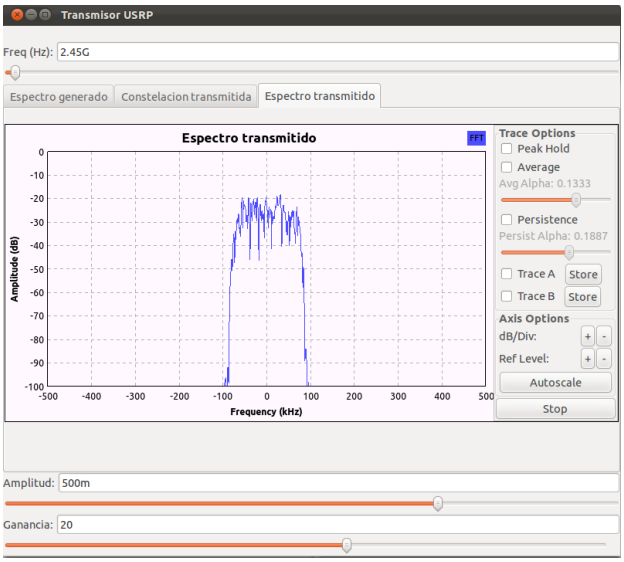


Fig 7.0.7: Espectro transmitido 4-QAM

Tal y como previamente se había comentado, la modulación es llevada a cabo en banda base y luego posteriormente es desplazada en frecuencia mediante el uso del *mixer* situado en la *daughterboard* y del DUC (si procede).

La constelación recibida que le llega al modulador presentará los efectos descritos en el apartado 6.2 tal y como muestra la siguiente figura:

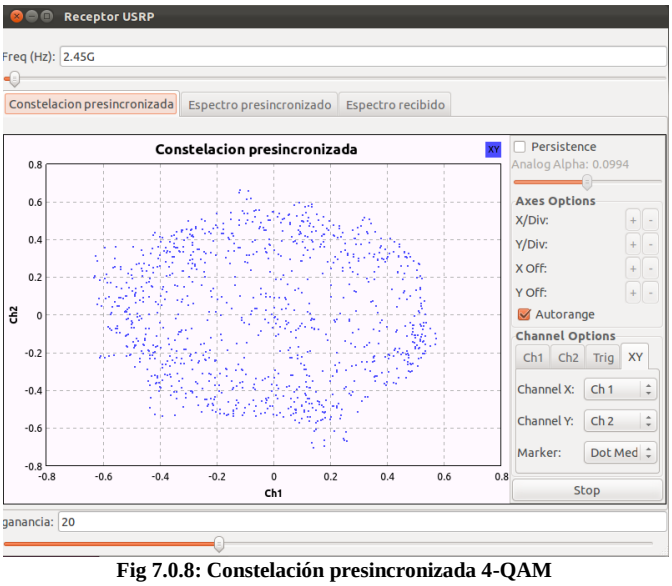


Fig 7.0.8: Constelación presincronizada 4-QAM

Como se había anunciado anteriormente, la señal no podrá ser correctamente demodulada si no se sincroniza previamente en tiempo, fase y frecuencia.

A continuación se muestra el espectro de la señal recibida justo antes de atravesar la cadena demoduladora :

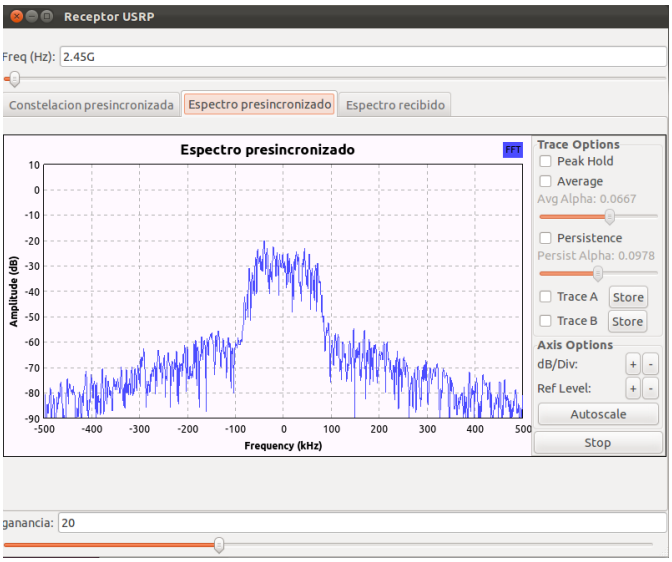


Fig 7.0.9: Espectro recibido 4-QAM

Finalmente, una vez realizado el proceso de compensado y de demodulación descrito en el apartado 6.1 se obtiene:

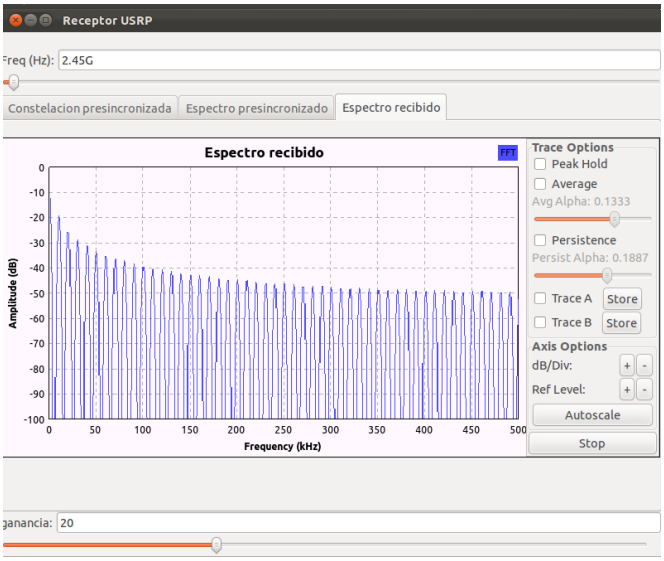


Fig 7.0.10: Espectro de la señal recibida 4-QAM

Donde la imagen anterior, muestra que el proceso de demodulación se ha llevado correctamente.

7.1.3.2 Sistema 4-PSK

Inicialmente se muestra el espectro de la señal generada, esta imagen se corresponde al espectro de una señal de tipo *saw tooth* de frecuencia 10KHz:

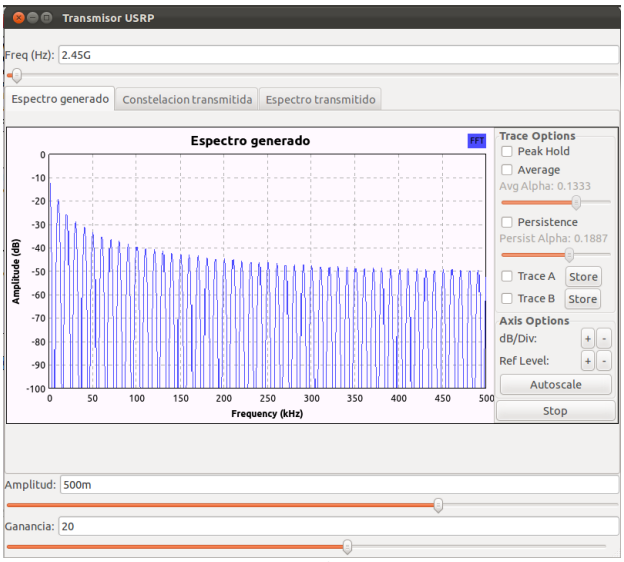


Fig 7.0.11: Espectro de la señal generada para 4-PSK

Una vez la señal atraviesa la cadena moduladora, la señal que se obtiene a la salida es la siguiente:

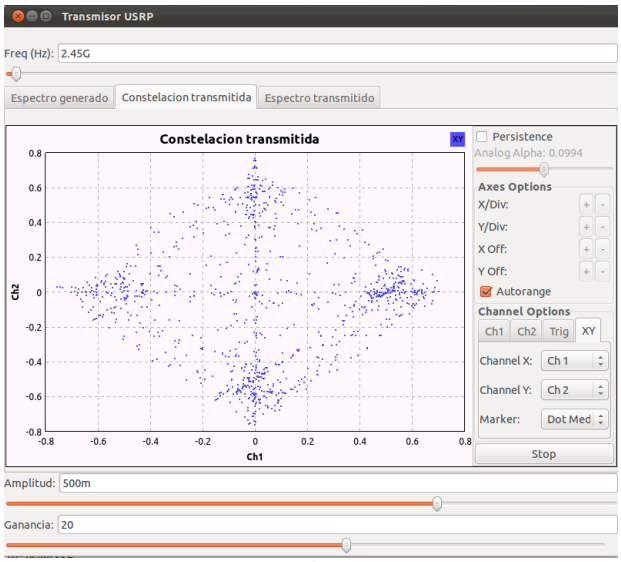


Fig 7.0.12: Constelación 4-PSK transmitida

Tal y como muestra la figura anterior, la constelación transmitida coincide con la figura 6.49 que se muestra en el apartado 6.1.2.3.1 tal y como era de esperar.

A continuación se mostrará el espectro de la señal transmitida:

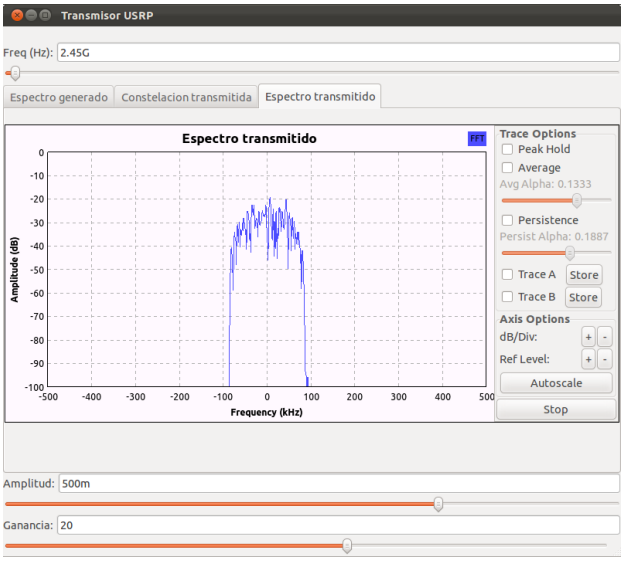


Fig 7.0.13: Espectro transmitido 4-PSK

La constelación recibida que le llega al modulador presentará los efectos descritos en el apartado 6.2 tal y como muestra la siguiente figura:

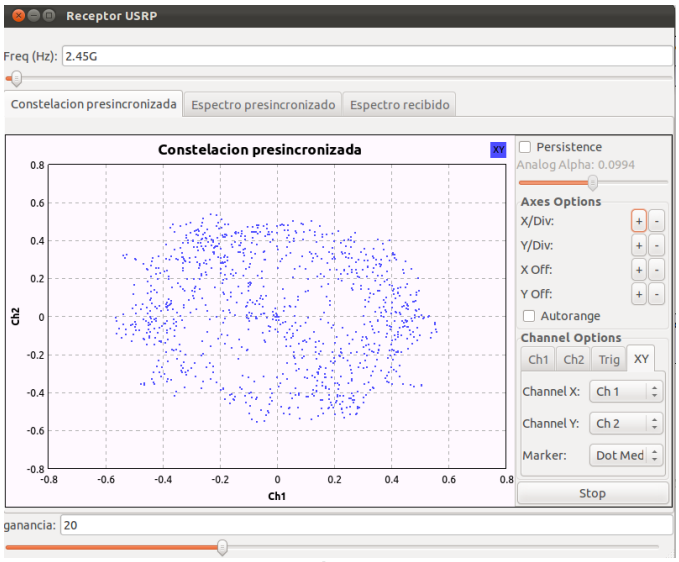


Fig 7.0.14: Constelación presincronizada 4-PSK

Tal y como se puede apreciar en la imagen anterior, la señal no podrá ser correctamente demodulada si no se sincroniza previamente en tiempo, fase y frecuencia.

A continuación se muestra el espectro de la señal recibida justo antes de atravesar la cadena demoduladora:

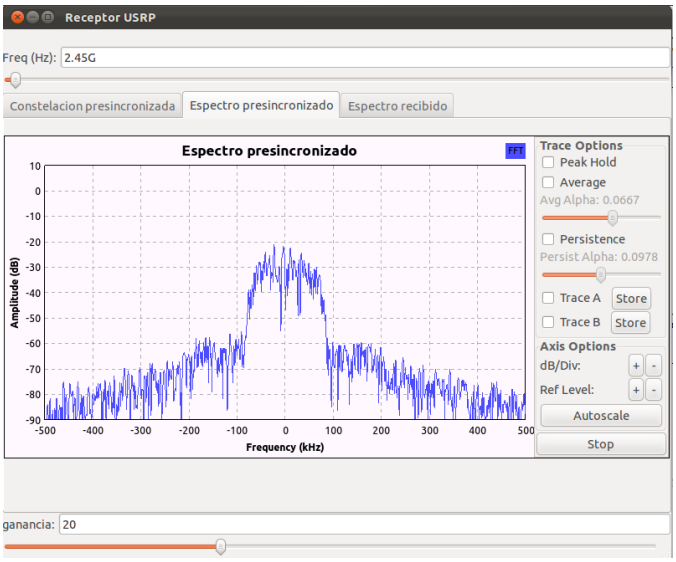


Fig 7.0.15: Espectro recibido 4-PSK

Tal y como previamente se había comentado, la modulación es llevada a cabo en banda base.

Finalmente, la señal recibida se muestra a continuación:

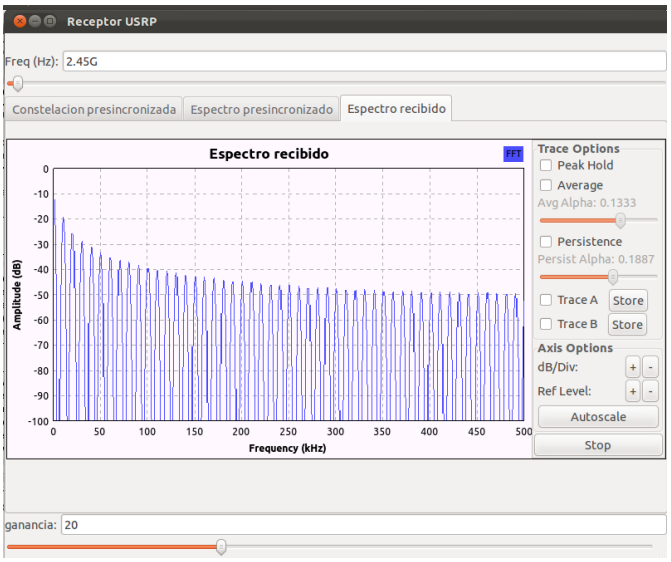


Fig 7.0.16: Espectro de la señal recibida 4-PSK

Esta es la figura que ilustra que el proceso se ha llevado correctamente, de no haber sido así, si se hubiera producido errores el bloque *packet decoder*, no hubiera devuelto datos y esta imagen quedaría completamente en blanco. Cabe destacar, que si existe alguna diferencia entre los espectros de la señal generada y recibida se debe a la variabilidad de la señal en el tiempo y a los diferentes momentos de capturas.

7.1.3.3 Sistema FSK

Inicialmente se muestra el espectro de la señal generada, esta imagen se corresponde al espectro de una señal triangular de frecuencia 10KHz:

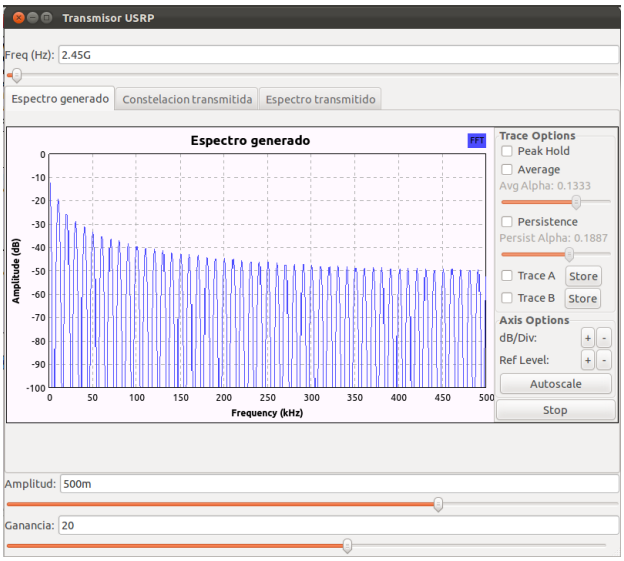


Fig 7.0.17: Espectro de la señal generada para FSK

Una vez la señal atraviesa la cadena moduladora, la señal que se obtiene a la salida es la siguiente:

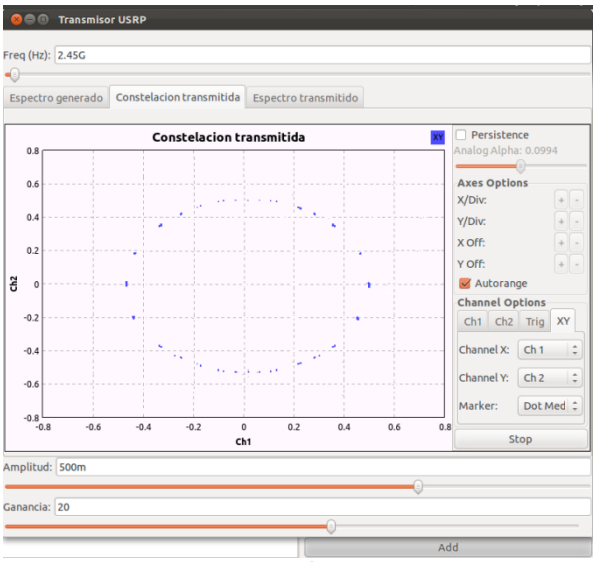


Fig 7.0.18: Constelación FSK transmitida

A continuación se mostrará el espectro de la señal transmitida:

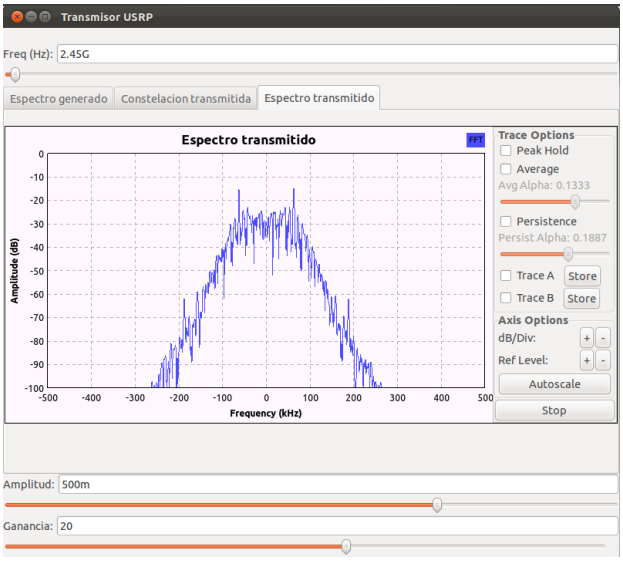


Fig 7.0.19: Espectro transmitido FSK

La constelación recibida que le llega al modulador presentará los efectos descritos en el apartado 6.2 tal y como muestra la siguiente figura:

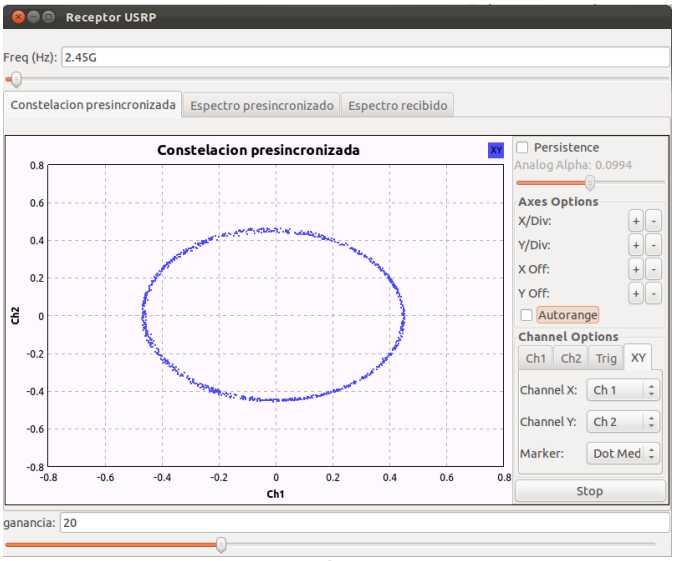


Fig 7.0.20: Constelación presincronizada FSK

A continuación se muestra el espectro de la señal recibida justo antes de atravesar la cadena demoduladora :

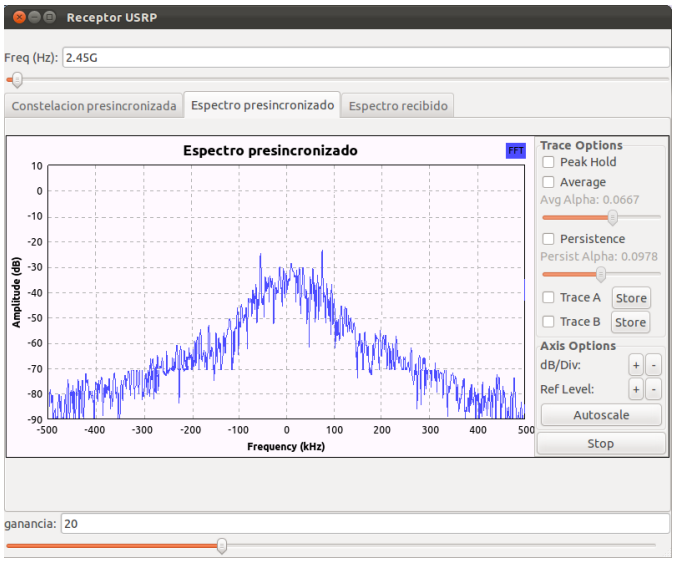


Fig 7.0.21: Espectro recibido FSK

Tal y como previamente se había comentado, la modulación es llevada a cabo en banda base.

Finalmente, una vez la señal atraviesa el demodulador se obtiene la siguiente figura:

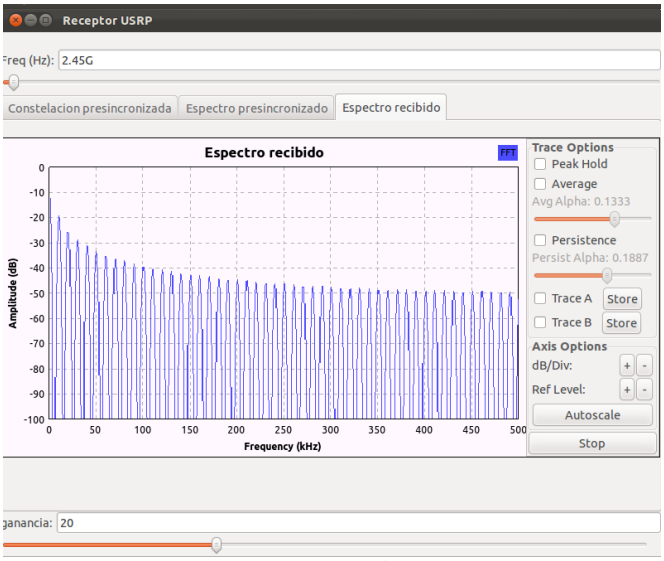


Fig 7.0.22: Espectro de la señal recibida FSK

Donde la imagen anterior, muestra que el proceso de demodulación se ha llevado correctamente.

7.2 Script desarrollado

Los apartados hasta ahora presentados (capítulo 5 y apartado 7.1), se basan en la interacción de la herramienta GNU Radio con la plataforma hardware disponible. Hasta ahora, se ha demostrado cómo es posible configurar las plataformas en cuanto a frecuencia, ganancia, modulación utilizada…

Este script se desarrolló con el propósito de facilitar la tarea al desarrollador evitando la necesidad de programar la aplicación. Consecuentemente, con la intención de reducir el nivel de conocimiento requerido del hardware a utilizar, se aprovecha la información almacenada en las EEPROM para brindar una configuración personalizada según el tipo de *daughterboard* utilizada en cuanto a rangos de ganancia, rangos de frecuencia de operación y modo de operación (*half-duplex* o *full-duplex*)

Dicho script, constará de las siguientes aplicaciones:

- Transmisor:
    - Dos tonos: El usuario podrá emitir dos tonos donde podrá manipular ciertos parámetros de forma gráfica.
    - Barrido de frecuencia: El usuario podrá realizar un barrido en frecuencia con un tono, para ello tendrá la posibilidad de modificar ciertos parámetros de forma gráfica.
- Receptor:
    - Analizador de espectro: El usuario podrá implementar un analizador de espectros donde podrá manipular de manera gráfica ciertos parámetros.

Como en capítulos anteriores se ha mostrado la transmisión de tonos, sólo se presentará la interfaz gráfica que muestra dicho script.

La siguiente imagen muestra la bienvenida al programa:

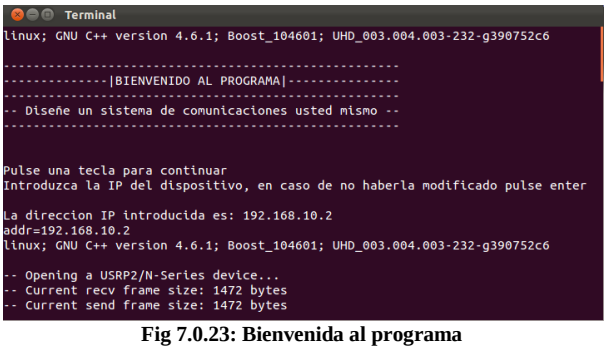


Fig 7.0.23: Bienvenida al programa

Inicialmente el programa te pide introducir la IP del dispositivo. Ésta, por defecto está fijada a 192.168.10.2 y si no se ha modificado no hace falta introducirla.

En este momento se iniciará la comunicación con el USRP, de tal forma que si la IP introducida no sigue un patrón de dirección IP v4, el programa solicitará que se introduzca dirección válida. En el caso en el que la dirección introducida siga el patrón

establecido pero sin embargo sea incorrecta (se introduce otra que no es) o el USRP no está conectado al ordenador, el programa informará que se conecte el USRP dando un tiempo de 25 segundos tratando de iniciar la comunicación con el dispositivo cada 4 segundos. Finalmente, si no se ha podido iniciar la comunicación con el USRP el programa finalizará la ejecución.

A continuación, una vez establecida la comunicación con el periférico, el programa mostrará un submenú donde se pregunta si se desea trabajar en modo *half-duplex* o *full-duplex*. En caso de que la *daughterboard* no soporte el modo de operación *full-duplex* (XCVR2450) y se seleccione dicho modo, el programa lanzará un aviso de que la placa utilizada no soporta dicho modo y se solicita que elija el modo de operación *half-duplex*.

En caso de que el modo de operación sea *half-duplex*, se mostrará un nuevo menú en el que se preguntará si quiere implementar un transmisor o por el contrario un receptor redirigiendo a otro submenú donde indicará las aplicaciones disponibles. En caso de que el modo de operación haya sido *full-duplex* conducirá en primer lugar al submenú de aplicaciones de un transmisor y posteriormente al del receptor. La siguiente imagen muestra el submenú para un transmisor:

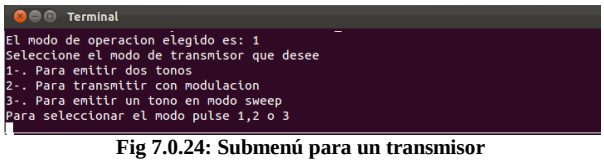


Fig 7.0.24: Submenú para un transmisor

Tal y como se mencionaba, la imagen anterior muestra el menú para un transmisor, a diferencia de lo que se mencionaba al inicio del apartado, éste, muestra tres modos de operación, sin embargo, el segundo modo, aún no está desarrollado. No obstante, se ha tenido en cuenta para que sea escalable y en el futuro se pueda implementar esta modalidad.

Si en vez de caracterizar un transmisor, lo que se eligió fue caracterizar un receptor en modo *half-duplex*, el programa mostrará por pantalla la siguiente imagen:

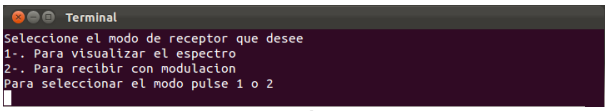


Fig 7.0.25: Submenú para un receptor

Nuevamente, tal y como ocurría en el caso del transmisor, la imagen mostrada difiere en cuanto a la funcionalidad descrita en el inicio del apartado. El motivo de ello, facilitar la extensión del script sin cambiar su estructura.

Las siguientes imágenes muestran la apariencia y los parámetros de configuración en las distintas aplicaciones de las que consta el script. Para ello, puesto que ya se realizaron pruebas muy similares en capítulos anteriores, no se mostrará la comunicación entre dos USRP, si no que se ha utilizado un único USRP con las distintas placas hijas.

En el caso en el que se desee probar la aplicación que emitirá dos tonos, se seleccionará modo como modo de operación *half-duplex*, y en el menú, se elegirá que se desea caracterizar a un transmisor implementando esta aplicación, mostrándose así la siguiente imagen:

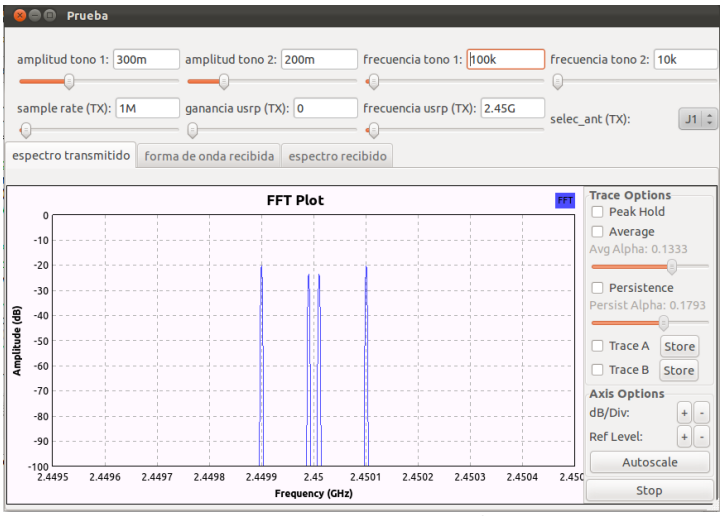


Fig 7.0.26: Two tones en modo half-duplex

En este caso, se está transmitiendo dos tonos de distinta frecuencia y distinta amplitud, la señal será transmitida por el puerto J1 puesto que se está utilizando la placa XCVR2450 donde el único modo de operación permitido es el *half-duplex*.

Como se eligió el modo de operación *half-duplex*, no hay diseñado ningún receptor tal y como muestran las siguientes imágenes:

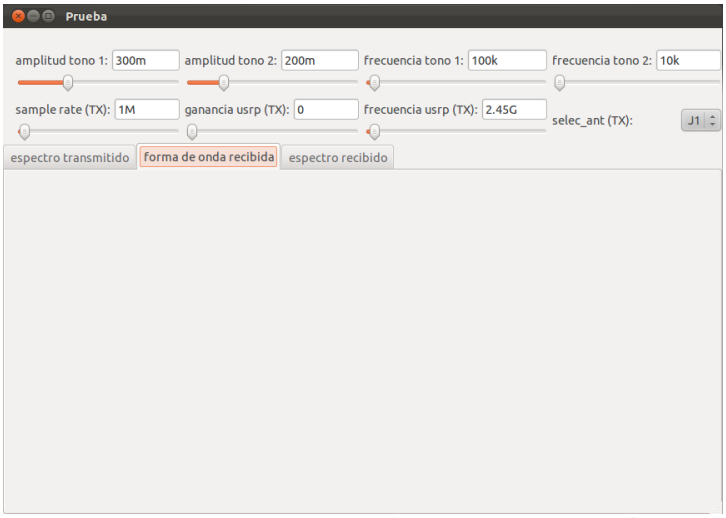


Fig 7.0.27: Forma de onda habiendo diseñado un transmisor en half-duplex

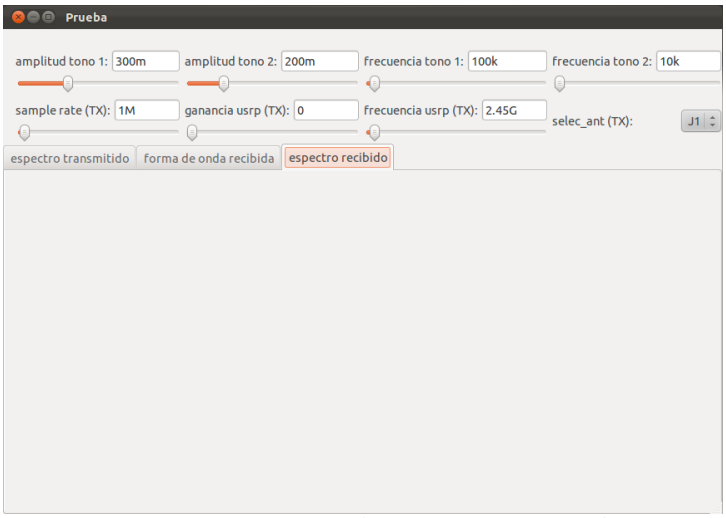


Fig 7.0.28: Espectro habiendo diseñado un transmisor en half-duplex

La otra aplicación disponible para caracterizar a un transmisor, se basa en realizar un barrido en frecuencia donde se puede variar el ancho del barrido y la velocidad de este. Para ello se elegirá como aplicación la tercera disponible en el menú que muestra al haber seleccionado que se desea caracterizar a un transmisor:

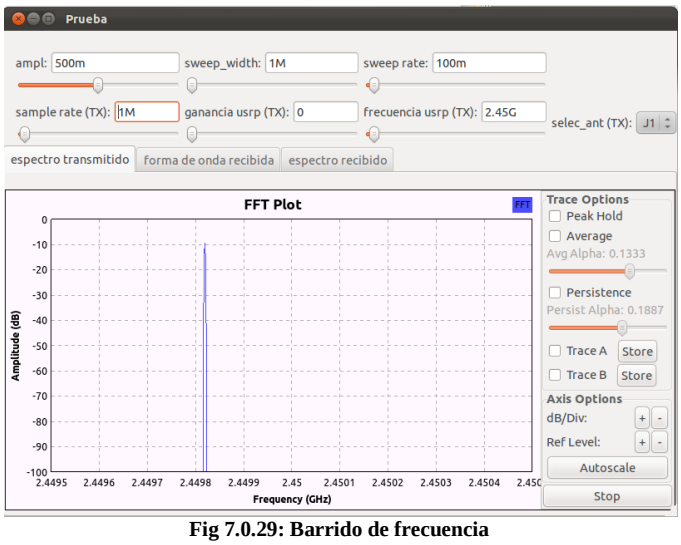


Fig 7.0.29: Barrido de frecuencia

En esta aplicación, se puede establecer el parámetro que indicará la velocidad de transmisión al USRP así como el *sample rate* Como parámetros configurables disponibles se encuentran la amplitud del tono, la velocidad de transmisión, la ganancia transmisora, la frecuencia de transmisión, la rapidez con la que se realiza el barrido (*sweep rate*) y el ancho del barrido. Con respecto al ancho del barrido, será siempre la mitad de lo que el usuario indique en pantalla. Para implementar esta aplicación se ha utilizado un bloque que modula en frecuencia a la señal y al que se le indica como parámetro el valor de la tasa de barrido, mientras que en la misma generación de la señal se le indicará como parámetro la frecuencia del barrido (Nótese el factor $\frac{1}{2}$ mencionado).

En caso de haber elegido caracterizar a un receptor en modo *half-duplex*, algo similar hubiera ocurrido, se mostraría el espectro de la señal recibida y su forma de onda mientras que no aparecería señal generada.

Cuando el modo de operación elegido es *full-duplex*, se estará caracterizando a un transmisor y a un receptor simultáneamente por lo que se le deberá indicar la aplicación que se quiere implementar tanto para el transmisor como para el receptor.

La siguiente imagen muestra los parámetros de configuración para la aplicación de transmisión dos tonos en dicho modo de operación:

Fig 7.0.30: Two tones en modo full-duplex

Tal y como muestra la figura anterior, los parámetros configurables en este tipo de aplicación deberán coincidir prácticamente con los expuestos en la figura 7.0.26. Sin embargo, al estar caracterizando a un receptor a la vez se le han de añadir la configuración disponible para este. Siendo así configurable la velocidad de transmisión del USRP, la ganancia que presenta en la cadena de recepción y la frecuencia de recepción.

Al haber establecido el modo de operación que habilita la transmisión y la recepción simultánea, en este caso, también estará disponible el receptor tal y como muestra la siguiente imagen:

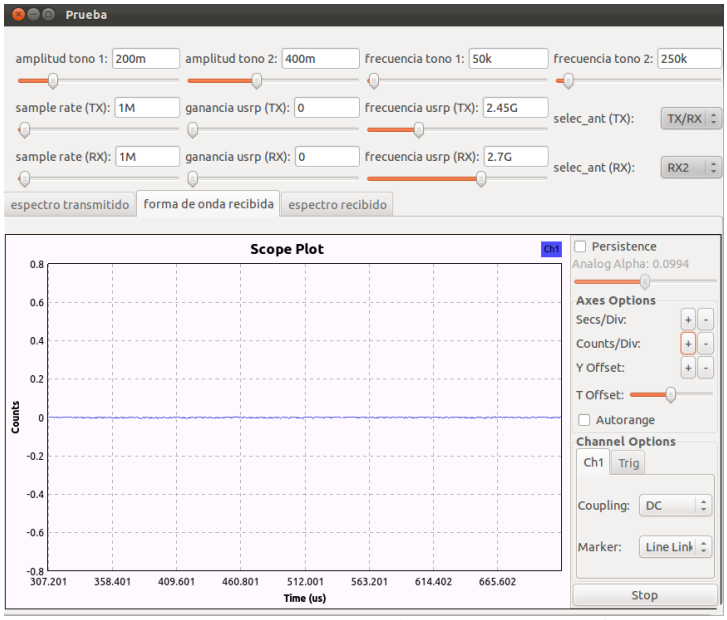


Fig 7.0.31: Forma de onda habiendo diseñado un tranmisor en full-duplex

Como se puede apreciar, el USRP también está configurado como receptor y está transmitiendo datos al PC. En este caso tal y como se ha mencionado, al utilizar un único USRP no le está llegando señal de información.

Cuando se caracteriza al receptor también es importante la información que se puede apreciar en el dominio frecuencial por ello se muestra también el espectro de la señal recibida:

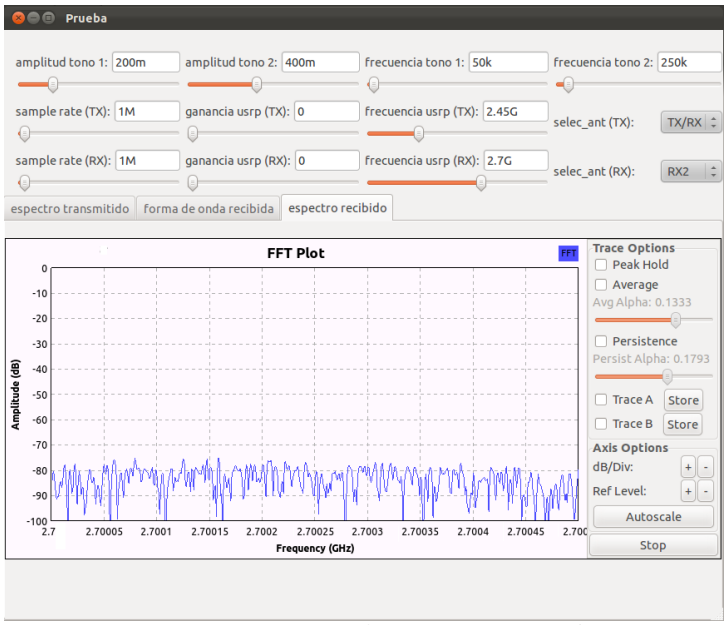


Fig 7.0.32: Espectro habiendo diseñado un transmisor en full-duplex

Una vez más se puede apreciar cómo no se está recibiendo ninguna señal de información puesto que no se distingue ninguna componente frecuencial, simplemente señal de ruido.

8
Conclusiones y trabajo futuro

8.1 Conclusiones

En este PFC, se plantea el uso del SDR como una plataforma donde realizar experimentos prácticos con gran versatilidad tanto en entorno de simulación como con plataformas hardware.

Los sistemas basados en *Software Radio* tienen la finalidad de implementar sistemas radio, aglutinando gran variedad de estándares existentes y servir como base a nuevas tecnologías emergentes como es *Cognitive Radio*. La finalidad de CR es la personalización del enlace radio.

Tal como se ha descrito en este documento, existen dos partes bien diferenciadas en un sistema basado en SDR, la parte hardware y la parte software. La primera realiza el paso de desde radiofrecuencia a banda base o a frecuencia intermedia (low IF) y es llevada a cabo mediante un periférico encargado de operar en un rango amplio de frecuencia (en función de la *daughterboard* empleada). Mientras que la segunda, se encarga en líneas generales de todo el procesado de señal.
Existe gran variedad en cuanto a periféricos S*oftware Radio*, tanto a fabricantes como a modelos.

La conexión entre periférico y ordenador es llevada a cabo mediante cable de red (USRPN210) El nexo entre el ordenador y el USRPN210 es el driver que proporciona Ettus. Existen librerías para Python que lo incorporan, para funcionar en Linux, Mac e incluso en Windows. Para evitar programar en Python existe un entorno de ventanas y bloques, GNU Radio Companion, facilitando la tarea al diseñador.

Para alcanzar el objetivo de implementar un sistema de comunicaciones con plataformas hardware, el primer paso a realizar es la caracterización de dichos dispositivos. Para ello, se ha utilizado dispositivos de medición externos tal que verifiquen que se controla la transmisión y recepción de señales RF.

Una quedan caracterizados los dispositivos como transmisor y receptor de RF, el siguiente paso es controlar la herramienta software con la que se diseñan las aplicaciones, para ello el paso más sensato es operar en un entorno de simulación con la herramienta software (GNU Radio) tal que se diseñe el sistema a implementar sin recurrir al uso de ningún hardware adicional.

Por último, una vez se han realizado los pasos previos y se maneja la herramienta software, el siguiente paso a dar, es llevar a la práctica todos los conocimientos adquiridos migrando el diseño en el entorno de simulación para ahora sí, implementar un sistema de comunicación basado en la tecnología SDR haciendo uso de las plataformas hardware.

8.2 Trabajo futuro

Como trabajo futuro, con el aporte de los conocimientos necesarios para poder implementar sistemas, se podría proseguir el trabajo comenzado en el script desarrollado añadiendo más funcionalidades. Además se podría estudiar otros sistemas OFDM que han quedado fuera del alcance de este proyecto apoyándose en el trabajo ya realizado en este campo por la comunidad de GNU Radio.

Otra posible continuación del trabajo expuesto sería incorporar antenas para realizar un sistema de comunicaciones *wireless* empleando la tecnología MIMO soportada por los periféricos disponibles.

Otra línea de trabajo futuro sería trabajar en el proyecto Open BTS, el cuál combina la interfaz GSM del usuario con tecnologías VoIP mediante SDR.

En términos generales, como línea de futuro, se propone trabajar de manera colaborativa en cualquier proyecto de los mencionados en el apartado 1.4. Para implicarse en alguno de los proyectos mencionados o de cualquier otro que surja, se recomienda empezar buscando los proyectos en la misma página de GNU Radio, siendo ahí donde detallan más información de cada proyecto.

Referencias

[1] Wireless Innovation Forum:
http://www.wirelessinnovation.org/page/Introduction_to_SDR

[2] J Mitola, "The Software Radio", IEEE National Telesystems Conference, 1992 - Digital Object Identifier 10.1109/ NTC.1992.267870

[3] Software Defined Radio:
http://www.wirelessinnovation.org/assets/documents/SoftwareDefinedRadio.pdf

[4] R.I. Lackey, D. W. Upmal, "*SpeakEasy:The Military Software Radio*", IEEE Communications Magazine, Vol. 33, New York, pp. 5-6-61, May. 1995.

[5] The Joint Tactical Radio System (JTRS) and the Army's Future Combat System (FCS): Issues for Congress:
http://digital.library.unt.edu/ark:/67531/metacrs7941/m1/1/high_res_d/RL33161_2005Nov17.pdf

[6] Tevfik Yücek, and Hüseyin Arslan., "A Survey of Spectrum Sensing Algorithms for Cognitive Radio Applications", IEEE Communications Surveys & Tutorials, vol. 11, no. 1, first quarter 2009.

[7] J. Mitola, Cognitive radio – model-based competence for software radios, Licentiate Thesis, KTH, Stockholm (September 1999).

[8] J. Mitola, J. and J. Maguire, G. Q., "Cognitive radio: making software radios more personal," IEEE Personal Commun. Mag., vol. 6, no. 4, pp. 13–18, Aug. 1999.

[9] National Telecommunications and Information Administration, on FCC ET Docket No. 03-108: "Facilitating Opportunities for Flexible, Efficient and Reliable Spectrum Use Employing Cognitive Radio Technologies", Feb. 2005.

[10] Understanding the Issues in Software Defined Cognitive Radio (Jeffrey H. Reed, Charles W. Bostian. Virginia Tech Bradley Dept. of Electrical and Computer Engineering):
http://wireless.vt.edu/research/Cognitive_Radios_Networks/Presentations/IEEE_San_Diego_CR_Talk.pdf

[11] Communication Systems Engineering (2nd Edition) [John G. Proakis, Masoud Salehi].

[12] Apuntes Teoría de la comunicación EPS UAM (2007-08) Temas: III, IV.

[13] eBook "Electrical Communication "is based on the printed copy of the book "Electrical Communication" by A.L. Albert (http://www.vias.org/albert_ecomm/).

[14] www.ettus.com

[15] http://code.ettus.com/redmine/ettus/projects/public/documents

[16] http://gnuradio.org/redmine/projects/gnuradio/wiki/UsrpDBoardRFX2400

[17] www.gnuradio.org

[18] http://www.csun.edu/~skatz/katzpage/sdr_project/sdr/plotting_with_octave.pdf

[19] http://gnuradio.org/doc/doxygen/

[20] http://gnuradio.org/redmine/projects/gnuradio/wiki/FAQ

[21] http://gnuradio.4.n7.nabble.com/

[22] https://www.ruby-forum.com/forum/gnuradio

[23] http://stackoverflow.com/questions

[24] http://lists.ettus.com/mailman/listinfo

[25] http://www.ettus.com/kb/detail/frequently-asked-questions

Apéndice A: Puesta en marcha del software necesario:

En este proyecto se indicará los pasos a seguir para el correcto uso de las herramientas software necesarias para poder desarrollar un sistema de comunicaciones basado en plataformas software, en particular las herramientas que se instalarán son GNU Radio + UHD tanto en Windows como en Linux, así como la instalación de UHD + Simulink para poder utilizar esta herramienta en Matlab.

Instalación y configuración de GNU Radio + UHD

Las herramientas software GNU Radio y UHD pueden ser instaladas en plataformas como LINUX, Mac y Windows.
Para instalar dichas herramientas hay dos posibilidades:

- Instalar el software mediante paquetes pre-compilados.
- Compilarlo manualmente.

La primera de ellas es más cómoda para el usuario, sin embargo, presenta claros inconvenientes entre ellos la alta probabilidad de instalar versiones obsoletas de GNU Radio, o la dificultad que implica modificar parte del proyecto GNU Radio en este método de instalación, por el contrario la otra opción presenta una mayor flexibilidad aunque la dificultad en el proceso de instalación aumenta considerablemente.
En este proyecto se detallará minuciosamente los pasos a seguir para la correcta instalación y configuración de las herramientas software en Ubuntu y en Windows7.

- **Instalación y configuración de GNU Radio y UHD en Ubuntu:**

  Con motivo de simplificar el proceso de instalar (compilando manualmente) la herramienta software, Marcus Leech desarrolló un script de instalación, el cual se encarga de descargar e instalar todo el software necesario. En este proyecto se decidió emplear este método, siendo éste el método recomendado para la mayoría de los usuarios.

  Los pasos a seguir para la puesta a punto del software necesario son los siguientes:

  - Abrir la terminal y escribir:
    - $ wget http://www.sbrac.org/files/build-gnuradio && chmod a+x ./build-gnuradio && ./build-gnuradio
  - Una vez haya finalizado el proceso de instalación se escribe en la terminal:
    - $ sudo gedit .bashrc, añadiendo:
      - export PYTHONPATH=/usr/local/lib/python2.7/dist-packages
  - Instalación de los siguientes paquetes:
    - python-matplotlib
    - python-scipy
    - python-numpy

    Para ello abrir terminal y escribir:
    - $ sudo apt-get install python-matplotlib

    - $ sudo apt-get install python-scipy
    - $ sudo apt-get install python-numpy
- Configuración de la red, se escribe en la terminal:
    - $ sudo gedit /etc/network/interfaces, añadiendo:

```
auto eth0
iface eth0 inet static
address 192.168.10.1
netmask 255.255.255.0
```

- Una vez se haya realizado los pasos anteriores, será posible hacer un ping al USRP, este por defecto tiene la IP 192.168.10.2, para ello escribir en la terminal:
    - $ ping 192.168.10.2
- En caso de no obtener respuesta del dispositivo, se deberá desactivar el firewall, para ello escribir en la terminal:
    - $ sudo ufw disable

- **Instalación y configuración de GNURadio y UHD en Windows7:**

  La instalación del software GNU Radio y UHD en Windows se ha llevado a cabo mediante paquetes pre-compilados.
  Para este proceso se dispone de dos opciones:

  - Instaladores estables: versiones que solamente incluyen correcciones de errores desde la última versión.
  - Instaladores inestables: versiones que además de incluir correcciones también incluye nuevas características.

  En este caso se eligió utilizar instaladores estables.
  Los pasos a seguir son los siguientes:

  - Instalación de UHD (UHD_003.005.000-release_Win32.exe)
    - En el proceso de instalación seleccionamos añadir al PATH del sistema
  - Instalación de GNU Radio (GNURadio_3.6.2-19_Win32)
    - En el proceso de instalación seleccionamos añadir al PATH del sistema
    - Añadir a la variable de entorno PYTHONPATH:
      - <dir-gnuradio>\lib\site-packages
  - Instalación de Python (v2.7.3)
  - Instalación de Numpy (numpy-1.6.2-win32-superpack-python27.exe)
  - Instalación de Scipy (scipy-0.11.0.win32-py2.7.exe)
  - Instalación de Matlplotlib (matplotlib-1.2.0.win32-py2.7.exe)
  - Instalación de PyGTK (pygtk-all-in-one-2.24.win32-py27)
  - Instalación de PyQT (PyQt-Py2.7-x86-gpl-4.9.6-1)
  - Instalación PyQWT (PyQt4.Qwt5-5.2.1.win32-py27)
  - Instalación de wxPython (wxPython2.8-win32-ansi-2.8.12.1-py27)
  - Instalación de setup_tools (setuptools-0.6c11.win32-py2.7)
    - Instalación de Cheetah (desde la terminal de windows):
      - $Cd <dir-Python>\Script easy_install cheetah

    - Instalación de lxml (desde la terminal de windows):
        - $Cd <dir-Python>\Script easy_install lxml==2.3
- Instalación de MSVC (vcredist_x86)
- Configuración de la red:
    - Se accede a Panel de control\Redes e Internet\Conexiones de red
    - Botón derecho→ propiedades sobre conexión de área local
    - Se accede a las propiedades del protocolo de internet versión 4 (TCP/IPv4)
    - Elegir añadir manualmente la IP:
        - Dirección IP: 192.168.10.1
        - Máscara de subred: 255.255.255.0
- Una vez se haya realizado los pasos anteriores, será posible hacer un ping al USRP, este por defecto tiene la IP 192.168.10.2, para ello escribir en la terminal:
    - $ ping 192.168.10.2
- En caso de no obtener respuesta del dispositivo, se deberá desactivar el firewall.

El método anteriormente expuesto presenta ciertos errores documentados:

- Añadir bloques: Este problema está relacionado con el paquete PyGTK, en la interfaz GNU Radio Companion, los bloques no se podrán añadir dando doble-click.
- Durante la instalación de Cheetah dará warnings, sin embargo no afectará al correcto funcionamiento del programa.
- El QT-GUI slider no funcionará adecuadamente, este problema está relacionado con el paquete PyQWT empleado.
- Si se emplea instaladores estables para instalar GNU Radio también se tendrá que utilizar un instalador estable para UHD. De no ser así, no se podrá hacer uso de las funciones desarrolladas en el módulo gr-uhd.

Se acaba de presentar una forma probada de llevar a cabo la correcta instalación de las herramientas GNU Radio + UHD en Windows. Sin embargo, la manera recomendada de instalar dichas herramientas en Windows, es llevar a cabo la instalación del software bajo un entorno Linux, como puede ser Cygwin.

- **Instalación y configuración del soporte para USRP en Matlab+Simulink en Windows7:**

Matlab y Simulink provee un paquete para dar soporte al USRP, este paquete llamado Comunications System Toolbox Support Package for USRP Radio funciona con versiones superiores a R2011a, dando soporte al USRP N210 desde la versión R2011b, en este caso se procede a la instalación de esta herramienta en la versión de matlab R2012b, para ello se realizan los siguientes pasos:

- Abrir matlab como administrador y escribir en el Command Window:
    - $ targetinstaller
- Seleccionar la opción de internet cuando se pregunte desde donde se quiere instalar y dar a siguiente.

- Seleccionar el paquete USRP(R) Radio, el cual es compatible con versiones de Windows de 32 bits y de 64bits así como Linux y dar a siguiente.
- Para las versiones anteriores de matlab (R2012a y R2011b) ir a: http://www.mathworks.es/discovery/sdr/usrp.html y clicar en el paquete a instalar (Comunications System Toolbox Support Package for USRP Radio).
- Completar el cuestionario y dar a registrar
- Clicar en la versión del paquete previsto para la versión de matlab que se disponga.
- Una vez descargado el paquete descomprimirlo en una carpeta independiente de matlab.
- Abrir matlab y navegar hasta la carpeta descomprimida y escribir en el Command Window:
  - $ install.
- Una vez instalado el toolbox necesario, cada vez que se inicie una nueva sesión habrá que clicar en Add SDRu situado en el SHORTCUTS, si no está esta opción habrá que escribir setupsdru en el Command Window.
- Configuración de la red:
  - Se accede a Panel de control\Redes e Internet\Conexiones de red
  - Botón derecho→ propiedades sobre conexión de área local
  - Se accede a las propiedades del protocolo de internet versión 4 (TCP/IPv4)
  - Elegir añadir manualmente la IP:
    - Dirección IP: 192.168.10.1
    - Máscara de subred: 255.255.255.0
- Una vez se haya realizado los pasos anteriores, será posible hacer un ping al USRP, este por defecto tiene la IP 192.168.10.2, para ello escribir en la terminal:
  - $ ping 192.168.10.2
- En caso de no obtener respuesta del dispositivo, se deberá desactivar el firewall.

Cargar las imágenes en la placa del USRP:

La primera vez que se ejecute un programa que utilice el hardware USRP, es muy probable que el programa desde el cual se está ejecutando, lance por pantalla un error indicando que no es posible la sincronización con el dispositivo, o bien que directamente hay que actualizar las imágenes para el firmware y para la FPGA de la placa. Estas imágenes serán imprescindibles para permitir la comunicación entre el host y el periférico. Este apartado tiene por objetivo detallar los pasos a seguir para su correcta actualización en el dispositivo USRP N210.
A la hora de actualizar las imágenes ya mencionadas, hay que tener en cuenta que herramienta software se va a emplear, puesto que hay algunas herramientas que utilizan versiones específicas como es el caso de Matlab y Simulink (Release 003.002.003), mientras que para GNU Radio se aconseja utilizar la última versión, también hay que considerar el número de revisión del hardware, puesto que diferentes periféricos utilizarán diferentes revisiones. Para saber cuál es necesaria hay que mirar la pegatina de la parte trasera del USRP (en nuestro caso es r4.0), una vez se sepa cuál se va a

utilizar hay que proceder a su descarga desde el siguiente enlace: http://files.ettus.com/binaries/uhd_stable/releases.
La forma más cómoda y rápida de realizar la actualización de las imágenes es mediante Python, para ello si se dispone previamente de un intérprete simplemente se tiene que descargar los scripts usrp_n2xx_net_burner.py y usrp_n2xx_net_burner_gui.py desde el siguiente enlace:
http://code.ettus.com/redmine/ettus/projects/uhd/repository/revisions/cab1746e1c3582f8d44f2141cfcbb948aa6dd10c/show/host/utils.
Como el segundo script necesita del primero, ambos han de ser visibles para Python (incluidos en el PYTHONPATH). Una forma sería copiar éstos en la carpeta Scripts que incluye Python, si se ha procedido a la instalación de GNU Radio y UHD en Linux mediante el script de instalación, no hace falta la descarga puesto que ya los incorpora.
Para llevar a cabo la actualización de las imágenes, la forma más cómoda es mediante la interfaz gráfica de usuario (usrp_n2xx_net_burner_gui.py), aunque es posible hacerlo en Linux mediante código.
Mediante la interfaz gráfica:
Para ello hay que ir hasta la carpeta donde se encuentra este script, típicamente en Linux su ubicación será: home/user/uhd/host/utils/usrp_n2xx_net_burner_gui.py, mientras que en Windows si se ha descargado tal y como se ha indicado la ubicación será: C:\Python_xx\Scripts, los pasos a seguir son:

- Ejecutar el script usrp_n2xx_net_burner_gui.py.
- Indicar las ubicaciones de las imágenes.
- Clicar en rescan for devices e indicar la Network Address del dispositivo (típicamente será 192.168.10.2 a menos que previamente se haya modificado).
- Clicar en Burn Images.
- Cuando pregunte si desea resetear el dispositivo darle aceptar.

Mediante código (sólo en Linux):

- $ cd home/user/uhd/host/utils.
- $ ./usrp_n2xx_net_burner.py --addr=<ip address> --fw=<ubicación de la imagen del firmware>
- $ ./usrp_n2xx_net_burner.py --addr=<ip address> --fpga=<ubicación de la imagen de la fpga>

Reseteo del periférico en modo seguro:

Esta opción está disponible en los Networks-Series del USRP, es posible recuperar el dispositivo en modo seguro (salido de fábrica) accionando el pulsador (S2) situado en la *motherboard* y próximo a la FPGA durante el arranque del dispositivo y manteniéndolo hasta que los *leds* del panel frontal dejen de parpadear. Este es el paso que hay que realizar si por algún motivo se cargan mal las imágenes del paso previo.
Al reconfigurar el dispositivo en modo seguro, la nueva IP del dispositivo es la que viene por defecto (192.168.10.2).

Apéndice B: GNU Radio companion-Python-C++

Este documento tiene por objetivo explicar el porqué del uso de diferentes lenguajes en una aplicación GNU Radio.

Como previamente ya se ha mencionado GNU Radio es una herramienta basado en tres capas o niveles de abstracción: GNU Radio companion, Python y C++.

Los descriptores xml es el precio que se tiene que pagar por disponer de los bloques en la interfaz gráfica y Swig el precio de embeber C++ en Python.

Tal y como se ha mencionado, el núcleo de GNU Radio, los bloques de procesado, están desarrollados en C++. La pregunta a formular en este punto es ¿por qué utilizar C++ como lenguaje *core*?
La respuesta reside en las ventajas que aporta este lenguaje, se trata de un lenguaje de nivel intermedio ya que comprende características tanto de alto nivel como de bajo presentando un buen rendimiento, además de permitir la programación orientada a objetos.

Siendo así, la siguiente pregunta que surge es ¿por qué utilizar Python?.
Para contestar a esta pregunta, primero se ha de mencionar los aspectos negativos que presenta C++. Se trata de un lenguaje que no presenta buenas prestaciones a la hora de realizar interfaces para interactuar con el usuario. Además, no es buen lenguaje para la integración.
Python es un lenguaje de alto nivel de *scripting* y por tanto no se necesita compilado, que aporta grandes ventajas como es la integración (*glue*) y la interacción con el usuario.

Por último, la última pregunta que hay que formular es ¿Por qué utilizar GNU Radio companion?
GNU Radio companion (GRC) es la interfaz gráfica de la que el desarrollador puede beneficiarse para escribir una aplicación GNU Radio completa sin escribir código, se trata por lo tanto de una herramienta gráfica opcional para simplificar el nivel de complejidad y de conocimientos en lenguajes de programación.

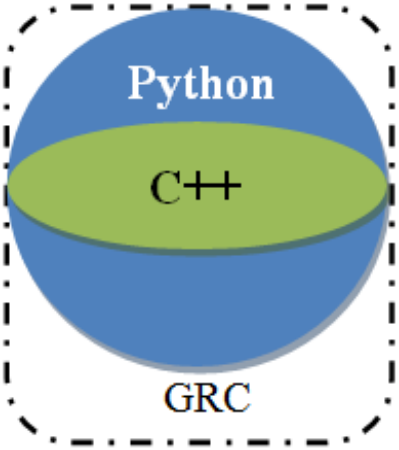


Fig D.0.1: Niveles de abstracción GNU Radio

Apéndice C: HOW TO ADD A BLOCK

Este documento tiene por objetivo indicar los pasos a seguir a modo ejemplo para llevar a cabo la inclusión de nuevos módulos (out-of-tree) y bloques en el proyecto GNU Radio.
Este proceso se lleva a cabo mediante el uso del script gr-modtool.py, el cual permite crear, modificar y eliminar módulos y/o bloques de procesado.

- **Generación de un nuevo módulo (out of tree):**

  Para crear un nuevo módulo bastará con escribir en pantalla el siguiente comando:

  ```
  $gr_modtool.py newmod nombre_del_módulo.
  ```

  Con este simple comando, el script genera todo el esqueleto de un módulo anteriormente descrito, en este caso se ha tomado como nombre del módulo prueba.

- **Generación de un nuevo bloque de procesado:**

  Para generar un nuevo bloque de procesado hay que escribir el siguiente comando una vez se haya posicionado dentro de la carpeta del módulo que queremos que lo incluya, en nuestro caso queremos que el bloque se encuentre dentro del nuevo módulo creado:

  ```
  $cd gr-nombre_del_módulo
  $gr_modtool add nombre_del_bloque
  ```

  Una vez se haya escrito esto preguntará por el tipo de bloque que deseas añadir, estos son: source, sink, hier, sync, decimator, interpolator, general, noblock.
  Sin adentrarse en las características de cada uno, se escogerá el bloque de tipo general, una vez escrito general en la pantalla, lo siguiente que se pide es que se introduzca es la lista de argumentos, esto no hace falta todavía puesto que se indicará más tarde, al pulsar la tecla Intro preguntará si se quiere añadir Python QA code, este código es utilizado a modo banco de pruebas para asegurarnos de que el bloque realiza su función correctamente, inmediatamente después se vuelve a preguntar por si se quiere añadir un C++ QA code, otra forma de comprobar el correcto funcionamiento es visualizar el resultado y ver que coincide con el esperado por lo que no hace falta implementar estos test. Una vez se ha contestado a las preguntas anteriores, el bloque se habrá añadido al módulo, y quedaría por realizar la propia implementación del bloque.

- **Implementación del bloque de procesado:**

  En este caso se ha decidido implementar un multiplexor 2:1 a modo ejemplo, se realizarán todos los pasos necesarios para disponer de este nuevo bloque en la interfaz GNU Radio-companion, donde se comprobará su correcto funcionamiento.

Lo primero que se ha de hacer, es escribir el código fuente del nuevo bloque, este se escribirá en el archivo mux_impl.cc, a continuación se muestra el código modificado:

```
/* The private constructor */
prueba_multiplexor::prueba_multiplexor()
: gr_block("multiplexor",
        gr_make_io_signature(3,3,sizeof(char),sizeof(char),sizeof(char)),
    //indicamos que son tres entradas; dos de datos y una de control. Las
    entradas //de datos son de tipo float, mientras que la de control es short
        gr_make_io_signature(1,1,sizeof(char)))
    // la salida será del mismo tipo que la entrada.


int prueba_mux::general_work (int noutput_items,
                gr_vector_int &ninput_items,
                gr_vector_const_void_star &input_items,
                gr_vector_void_star &output_items)
{
// se asignan las entradas y la salida
    const char *in0=(char *)  input_items[0];
    const char *in1=(char *)  input_items[1];
    const char *ctrl=(char *)  input_items[2];
    const char *out=(char*) output_items[0];
    for (int i = 0; i<noutput_items; i++){
        if (ctrl[i] ==1){
            out[i]=in0[i];
        }
        else{
            out[i]=in1[i];
        }
    }
return noutput_items ;
```

Si además se quiere disponer del nuevo bloque en la interfaz GNU Radio-companion, se ha de escribir manualmente el archivo xml situado en la carpeta gr-nombre_del_módulo/grc quedando de la siguiente manera:

```
<?xml version="1.0"?>
<block>
  <name>multiplexor</name>
  <key>prueba_multiplexor</key>
  <category>prueba</category>
  <import>import prueba</import>
  <make>prueba.multiplexor()</make>
  <sink>
    <name>in</name>
    <type>byte</type>
  </sink>
  <sink>
    <name>in</name>
```

```
    <type>byte</type>
  </sink>
  <sink>
    <name>in</name>
    <type>byte</type>
  </sink>
  <source>
    <name>out</name>
    <type>byte</type>
  </source>
</block>
```

Una vez creada la estructura del nuevo módulo e implementado el bloque de procesado, queda por instalar para que esté disponible en el proyecto GNU Radio. Para ello nos situamos en la carpeta raíz del módulo y escribimos en la terminal:

```
$ mkdir build
$ cd build
$ cmake ../
$ make
$ sudo make install
$ sudo ldconfig
```

El uso del script presenta diferentes bugs o problemas en el desarrollo de proyectos, tales como:

- Al momento de ejecutar el comando "cmake ../" se genera un error de que no encuentra el archivo "gruel_common.i" dando el siguiente aviso: "/usr/local/include/gnuradio/swig/gnuradio.i:31: Error: Unable to find 'gruel_common.i…". Este error se genera por el uso de swig con los parámetro de configuración obtenidos a través del archivo "CmakeLists.txt" en el folder "swig". Este error se corrige agregando al archivo "CmakeLists.txt" el siguiente código en la línea 38

  ```
  foreach(incdir ${GRUEL_INCLUDE_DIRS})
          list(APPEND GR_SWIG_INCLUDE_DIRS ${incdir}/gruel/swig)
  endforeach(incdir)
  ```

  Así también otro método para corregir el error es sustituyendo el archivo "CmakeLists.txt" por el que se ubica en el ejemplo "gr-howto-write-a-block" reemplazando los "howto" por el nombre del módulo del proyecto propio, puede usarse el "CmakeLists.txt" generado por gr-modtool.py para revisar las líneas que deben modificarse.

  Este error sólo se genera al momento de crear la estructura de directorios del módulo out-of-tree. Por lo que una vez corregido no se presentará de nuevo.

- Al terminar de compilar e instalar el nuevo módulo con sus bloques se genera el error "ImportError: /usr/local/lib… swig.so: undefined symbol: …". Esto se debe a que no se declaró las funciones del bloque como exportaciones API. El error se corrige dentro de la librería del bloque a implementar (TuMódulo_bloque.h) que se ubica dentro del directorio "include". Se agrega el código que incluye a la libería API del módulo a desarrollar en la línea 24, esto es arriba de "#include <gr_block.h>":

24 #include <TuMódulo_api.h>
Además de agregar la invocación de la librería, TUMODULO_API en las líneas 30, 36 y 38 quedando de la siguiente manera:

30 TUMODULO_API TuMódulo_TuBloque_sptr TuMódulo_make_TuBloque ();
36 class TUMODULO_API TuMódulo_TuBloque : public gr_block
38 friend TUMODULO_API TuMódulo_TuBloque_sptr TuMódulo_make_TuBloque ();

Este procedimiento se deberá realizar cada vez que se agregue un nuevo bloque al módulo *out-of-tree.*

Al modificar el módulo, se requiere volver a ejecutar los comandos instalación del bloque.
Finalmente ya se dispone del bloque operativo en la interfaz GNU Radio companion:

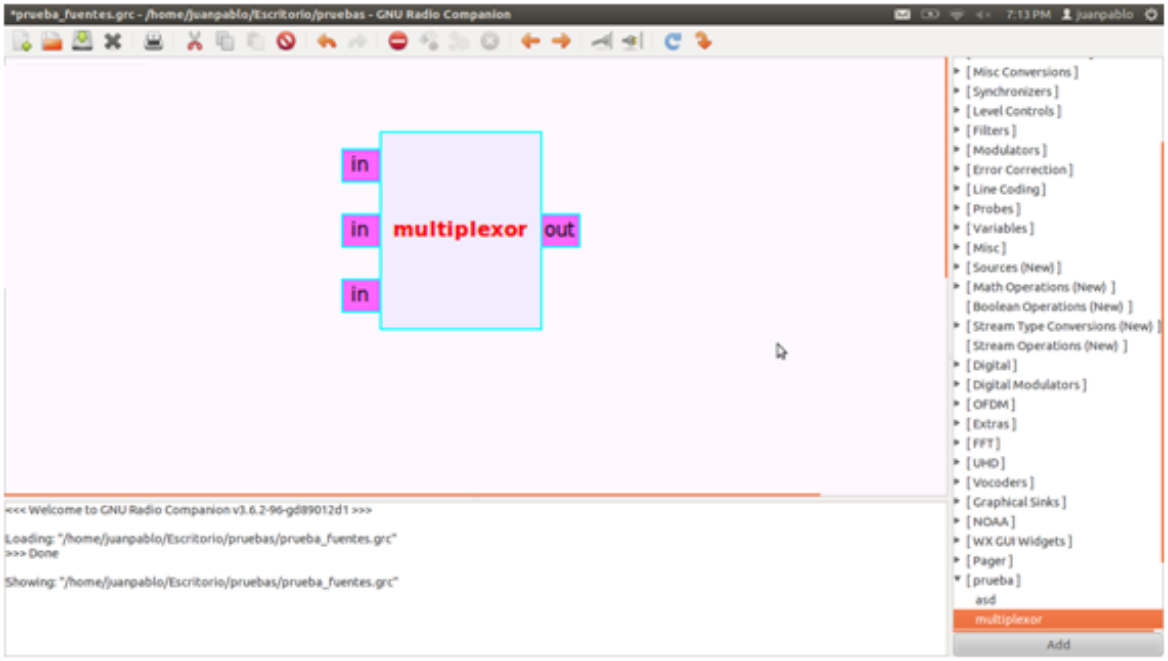


Fig B.0.1: multiplexor

Apéndice D: DIAL TONE

Este apéndice mostrará y explicará el código empleado para llevar a cabo una aplicación programada directamente en Python, en la que dos señales (cosenos) de frecuencias y amplitudes diferentes serán sumadas y conducidas a la tarjeta de audio.

```
#!/usr/bin/env python
from gnuradio import audio, eng_notation, gr
from gnuradio.gr import firdes
class dos_tonos(gr.top_block):

def __init__(self):

        gr.top_block.__init__(self, "Dos Tonos")
        fuente1 = gr.sig_source_f(32e3, gr.GR_COS_WAVE, 700, .3, 0)
        fuente2 = gr.sig_source_f(32e3, gr.GR_COS_WAVE, 300, .7, 0)
        suma = gr.add_vff(1)
        sumidero= audio.sink(32000, "plughw:0,0", True)
        self.connect((fuente1, 0), (suma, 0))
        self.connect((fuente2, 0), (suma, 1))
        self.connect((suma, 0), (sumidero, 0))

if __name__=’__main__’:
        tb = dos_tonos()
        tb.start()
        raw_input('Pulse enter para finalizar: ')
        tb.stop()
```

La primera línea a comentar, comienza con #, este carácter en el lenguaje Python corresponde a un comentario, la sentencia #! está indicando a la maquina el programa con el que se va a ejecutar el código.
Para comprender las siguientes líneas, es necesario hablar sobre los módulos y paquetes en lenguaje Python.
Un módulo es un programa realizado en Python, estos archivos tienen extensión “.py” y contiene el código de un programa que cumple alguna función específica. Un paquete es un conjunto de módulos que realizan funciones similares.

Una vez explicado a grandes rasgos la estructura de los módulos y paquetes en Python, se prosigue explicando el código anterior, al principio del programa se importan los módulos gr, eng_notation y audio.

Posteriormente se definirá la clase dos_tonos la cuál hereda todos los métodos de gr.top_block, esta clase contendrá el método __init__ el cuál será ejecutado cuando se realice la instancia de la clase, para ello hace uso del parámetro self que es la forma que tiene Python de tratar los objetos. El método __init__ contendrá las sentencias necesarias para definir los bloques de procesado:

```
fuente1 = gr.sig_source_f(32e3, gr.GR_COS_WAVE, 700, .3, 0)
fuente2 = gr.sig_source_f(32e3, gr.GR_COS_WAVE, 300, .7, 0)
suma = gr.add_vff(1)
```

```
sumidero= audio.sink(32000, "plughw:0,0", True)
```

Así como la declaración de las conexiones de los bloques:

```
self.connect((fuente1, 0), (suma, 0))
self.connect((fuente2, 0), (suma, 1))
self.connect((suma, 0), (sumidero, 0))
```

En primer lugar, se definen dos fuentes de tipo coseno (gr.GR_COS_WAVE), con amplitud 0.7 y 0.3 y frecuencia 700 y 300 respectivamente. Los otros parámetros se corresponden al offset deseado del coseno y al *sample rate*. Una vez definidas las fuentes del programa, se define el bloque el cuál se encarga de realizar la suma de las dos señales y posteriormente se define el sumidero, en este caso la tarjeta de sonido. La frecuencia de muestreo soportada es de 32e3Hz y la interfaz utilizada por ALSA[49] para manejar la tarjeta de sonido es el plughw:0,0.

Una vez queden definidos los bloques a utilizar, se define la conexión lógica uniendo los puertos de salida a los de entrada.

La última parte del código se corresponde con la primera parte a ejecutar, la línea "if __name__==__main__"[50] es la utilizada para ejecutar el contenido del condicional, esta expresión es utilizada para que pueda reutilizarse el código y así llamarse desde otro módulo. Las siguientes líneas crean una instancia de la clase dos_tonos y ejecutan el método start. En caso de pulsar intro, se llamará al método stop parándose la ejecución del programa.

Se ha mostrado el código de un simple programa típico a la hora de empezar con GNU Radio. Se recomienda que para facilitar la tarea al desarrollador se emplee la interfaz GNU Radio companion, puesto que permite una programación de la aplicación mucha más intuitiva y de manera directa.

[49] ALSA: Audio Linux Sound Architecture
[50] http://stackoverflow.com/questions/419163/what-does-if-name-main-do

Apéndice E: Preguntas frecuentes

Este capítulo tiene por objetivo mostrar gran parte de las dudas resueltas que han surgido a lo largo del desempeño del proyecto final de carrera, también se ha recogido respuestas a preguntas típicas de la comunidad GNU Radio en la sección FAQ[51][20],[21],[22],[23] y preguntas frecuentes relacionadas con el hardware [22].

Uno de los principales problemas que se ha encontrado al desarrollar el proyecto, es la búsqueda de documentación por lo tanto se recomienda encarecidamente en el momento de consultar alguna duda, comenzar por alguno de los foros así como leer este apéndice.

- **1-.Pregunta**: Cuando conecto el dispositivo se aprecia en la terminal (desde donde se esté ejecutando) el siguiente mensaje, ¿qué significa?, ¿para qué sirve?:

```
-- Opening a USRP2/N-Series device...
>>> -- Current recv frame size: 1472 bytes
>>> -- Current send frame size: 1472 bytes
>>>
>>> UHD Warning:
>>>     The recv buffer could not be resized sufficiently.
>>>     Target sock buff size: 50000000 bytes.
>>>     Actual sock buff size: 1000000 bytes.
>>>     See the transport application notes on buffer resizing.
>>>     Please run: sudo sysctl -w net.core.rmem_max=50000000
>>>
>>> UHD Warning:
>>>     The send buffer could not be resized sufficiently.
>>>     Target sock buff size: 1048576 bytes.
>>>     Actual sock buff size: 1000000 bytes.
>>>     See the transport application notes on buffer resizing.
>>>     Please run: sudo sysctl -w net.core.wmem_max=1048576
```

- **1-.Respuesta**: Tal y como indica el *warning*, para solucionar el problema potencial hay que escribir desde la terminal de Linux los siguientes comandos:

```
$sudo sysctl -w net.core.rmem_max=50000000
$sudo sysctl -w net.core.wmem_max=1048576
```

  Estos comandos se encargan de manejar a nivel sistema operativo el tamaño de los *buffers* para la transmisión y recepción cuando se interactúa con dispositivos que trabajan a una determinada tasa binaria. Esto se debe a que el ordenador es de propósito general y todos los procesos y/o hilos activos, comparten recursos (procesador/es). Por lo tanto, el planificador es el encargado de gestionar esos recursos cuando estime oportuno, por lo tanto las muestras generadas y/o recibidas se han de almacenar en *buffers*.

- **2-.Pregunta:** Cuando ejecuto aplicaciones GNU radio en la pantalla aparecen "O", "U", "a" y "u", ¿Qué significa?, ¿Cómo se soluciona?

[51] FAQ: Frequently Asked Questions

- **2-.Respuesta:** Estos mensajes tienen que ver con el dispositivo hardware que se está utilizando y aparecen cuando hay pérdida de muestras:

  "u" = USRP
  "a" = tarjeta de audio
  "U" = *underun*
  "O" = *overun*

  Cuando el ordenador no provee las muestras que están demandando los dispositivos se produce el *underun.* Por el contrario, el *overun* se produce cuando se pierden muestras porque el ordenador no es capaz de procesar la información entregada por los dispositivos.

  Las formas de solucionar este problema reside desde aumentar el tamaño de los *buffers* del SO[52], ajustar la tasa binaria o emplear una máquina más potente.

- **3-.Pregunta:** ¿Cuál es el ancho de banda soportado por el USRP N210?

- **3-.Respuesta:** El ancho de banda máximo que soporta el USRP depende del número de bits utilizados para representar las muestras, en caso de que se utilicen 16 bits por componente (32 bits por muestra) el ancho de banda máximo es de 25MHz, mientras que si se utilizan muestras de 16 bits (8 por componente) el ancho de banda máximo es de 50MHz.

- **4-.Pregunta:** La información correspondiente a fase y a cuadratura son enviadas de la forma $I_1Q_1I_2Q_2\cdots I_NQ_N$ (en caso de enviar más de un canal) en la trama. En caso de enviar una modulación que no tenga componente de cuadratura (BPSK, M-ASK). ¿Cómo se envía la información?:

- **4-.Respuesta:** Las modulaciones tales como BPSK y M-ASK no tienen componente en cuadratura (solo necesitan un vector para definir la base del subespacio vectorial), en la trama Ethernet la información de fase y cuadratura se entrelaza y llegado al USRP hay un multiplexor que se encarga de desentrelazar dichas componentes para separar los recorridos de las componentes. En caso de transmitir una constelación que quede definida con la componente en fase la información transmitida será de la forma $I_10I_20\cdots I_N0$

- **5-.Pregunta: ¿**Se puede implementar *phase arrays* con el software disponible?

- **5-.Respuesta:** Si, para implementar un *phase array* se utilizará la tecnología MIMO soportada por la familia *Network Series* de Ettus.

- **6-.Pregunta:** ¿Se puede programar una aplicación directamente en C++?

- **6-.Respuesta:** Si, se puede programar directamente en C++, aunque Python puede incrementar el rendimiento de la aplicación

[52] SO: Sistema Operativo

- **7-.Pregunta:** ¿Se puede programar directamente en Python?

- **7-.Respuesta:** Si, se puede emplear Python para el procesado de señal si se utilizan las librerías como numpy y scipy.

- **8-.Pregunta:** ¿Se puede utilizar un único USRP para transmitir y recibir?

- **8-.Respuesta:** Si y solo si la *daughterboard* soporta el modo de operación *full dúplex*. Dicho esto, hay que tener en mente el factores tales como el aislamiento entre el transmisor y el receptor y la potencia máxima de entrada (-10dBm).

- **9-.Pregunta:** Al ejecutar un programa en entorno de simulación la pantalla se queda congelada. ¿Cómo se soluciona esto?

- **9-.Respuesta:** La solución a este problema es introducir un bloque llamado *throttle*. Este bloque se tendrá que añadir siempre que no se utilice ningún hardware externo (USRP, tarjeta de sonido).

- **10-.Pregunta:** ¿Cuál es la función del bloque *throttle* de GNU Radio?

- **10-.Respuesta:** El bloque *throttle* de GNU Radio es el bloque que utiliza la herramienta para limitar el consumo de CPU cuando no hay una fuente externa que limita la *sample rate* del sistema.

- **11-. Pregunta:** ¿Cómo se distribuye la ganancia?

- **11-. Respuesta:** La ganancia en la cadena receptora se establece lo más próximo a la antena, primero en la *daughterboard* y luego en el ADC.
  Mientras que en el transmisor, la ganancia se establece primero en el DAC (si posee ganancia variable) y posteriormente en la *daughterboard*. Este proceso es llevado a cabo en el multi-usrp.cpp que invoca al método set_value() del fichero gain_group situado en la ruta /uhd/host/lib/utils.

- **12-. Pregunta:** ¿Qué significa el parámetro *sample rate* en GNU Radio?

- **12-. Respuesta:** *Digital signals are only meaningful in relation to the sampling rate. When we sample signals, we do so at a particular speed with, generally, uniform timing between samples. For our purposes here, we will assume uniform sampling in time to prevent any confusion.*
  When a real signal is sampled, we take a 'snapshot' of the signal at a particular time. We then convert the amplitude of that snapshot into a digital value, represented by some number of bits. Going the other way, we convert the digital representation at a point in a vector to an analog value at an equivalent point in time. These processes are known as analog to digital and digital to analog conversion, respectively.
  GNU Radio works on digital samples of signals. We pull samples in from a hardware source like a radio receiver or an audio microphone. Likewise, GNU Radio can send signals out to hardware sinks like a radio transmitter or an audio system. These hardware devices operate on a specific sample rate.

GNU Radio, however, generally works inside a general purpose processor (GPP) on a standard OS like Linux. Neither the OS nor the GPP has any really accurate concept of time, and so sampling rate as a measure of samples/second is rather meaningless. In fact, GNU Radio doesn't really understand sampling rates at all. It simply processes samples as fast as it can. So does sampling rate matter? Yes, but we have to be careful about how we understand how things are happening. Inside of GNU Radio blocks, most algorithms work off normalized sampling rates. They only understand that they have a set of samples that need to be processed. For something like a sine wave generated from a gr::analog::sig_source_X block, the sine wave is specified at a certain frequency and a given sample rate. The actual frequency of the sine generator only knows that it has to produce a full rotation around the unit circle at "frequency/sample rate". We normalize the frequency by the sample rate.
We can always think of GNU Radio running off a normalized sampling rate of 1.0 and everything must be set in relation to that value. We offer the concept of the sample rate in many blocks like the signal sources or the filter design functions so that we can more easily specify values of frequency and bandwidth in more natural terms like samples/second and Hertz.
But don't be confused: we need to match the sampling rates at the hardware ends. Think of it as an impedance matching problem. Take for example the following flowgraph:

`Rx (` *fs_0) --> block0 (D_0) --> block1 (D_1) --> block2 (I_0) --> Tx (* `fs_1)`

We receive a signal at a real sampling rate of fs_1. We have three blocks that do three stages of sample rate changes. We down sample in block0 by D_0; down sample again in block1 by D_1; and then up sampling in block2 by I_0. The signal is then passed to the transmitter, which operates at a rate of fs_0. What we have to make sure is that at each stage the rates are properly matched so that by the time we send the signal out of block2 the sampling rate of the signal is fs_1. That is:
*fs_1 = I_0 * fs_0/D_0/D_1*
If we're given fs_0 and fs_1, we have to make sure our rate-changing stages in the flowgraph match the above equation properly.

Apéndice F: Código para la estimación de la BER

Este documento tiene por objetivo proveer un método válido para la estimación de la *BER*, para ello la herramienta utilizada ha sido octave. El código empleado será mostrado y explicado a continuación:

```
close all;
clear
clc
addpath("/home/juanpablo/Escritorio/octave")
transmitida=imread("transmitida.jpg");
recibida=imread("recibida.jpg");
figure(1);
subplot(1,2,1);imshow(transmitida)
title("Imagen Transmitida","FontSize",20)
set([gca; findall(gca, "Type","text")], "FontSize", 20);
subplot(1,2,2);imshow(recibida)
title("Imagen Recibida","FontSize",20)
error = abs(transmitida-recibida);
figure(2);
imshow(error)
title("Error","FontSize",20)
dim=size(error);
colum=dim(2);
row=dim(1);
numBitsRepresentacion=8;
N_BITS= colum*row*numBitsRepresentacion;
errorv= reshape(error',1,row*colum);
errorbits=dec2bin(double(errorv),8);
tam=size(errorbits);
colum=tam(2);
row=tam(1);
bits=reshape(errorbits',1,row*colum);
biterrorrate=(100*sum((logical(bits-'0'))))/N_BITS;
printf("BER: %e%%\n",biterrorrate)
```

Para poder ejecutar el código anterior, primero se ha de abrir el editor de octave (escribir “$octave” en la terminal).
Las primeras líneas posteriores a realizar el borrado de las variables almacenadas y cierre de ventanas, sirven para poder cargar las imágenes en las variables transmitida y recibida (para que estén disponibles se ha de añadir la ruta donde se encuentran dichas imágenes):

```
addpath("/home/juanpablo/Escritorio/octave")
transmitida=imread("transmitida.jpg");
recibida=imread("recibida.jpg");
```

Las imágenes son representadas en un array de dimensión 256x256 de tipo uint8 (sus valores estarán comprendidos entre 0 y 255), el código muestra las imágenes transmitida y recibida así como la imagen definida mediante la resta de ambas:

```
error = abs(transmitida-recibida);
```

El número de bits totales será definido como el número de pixeles por el número de bits utilizados para representar el valor del pixel, 8 bits en este caso:

```
dim=size(error);
colum=dim(2);
row=dim(1);
numBitsRepresentacion=8;
N_BITS= colum*row*numBitsRepresentacion;
```

Para poder realizar la estimación de la BER, se transforma la matriz donde se almacena la imagen de error en una matriz fila, posteriormente se representa los valores de pixel en bits:

```
errorv= reshape(error',1,row*colum);
errorbits=dec2bin(double(errorv),8);
bits=reshape(errorbits',1,row*colum);
```

Finalmente, la BER, se define como el sumatorio del vector error dividido entre el número total de bits, para poder realizar esta operación se ha de transformar el tipo de dato en el cual está definido la variable bits, este valor calculado será mostrado por pantalla en notación científica:

```
biterrorrate=(100*sum((logical(bits-'0'))))/N_BITS;
printf("BER: %e%%\n",biterrorrate)
```

Apéndice G: Presupuesto

1) Ejecución Material

- Compra de ordenador personal (2x)..... 2000 €
- Compra USRP N210 (2x) ..... 2980 €
- Compra placa XCVR2450 (2x) ..... 700 €
- Compra placa RFX2400 (2x) ..... 490 €
- Compra de fuente de alimentación..... 22 €
- Alquiler de equipos de medida (4 meses)..... 4000 €
- Compra de impresora láser + toners..... 200 €
- Material de oficina..... 100 €
- Total de ejecución material..... 10692 €

2) Gastos generales

- 16 % sobre Ejecución Material ..... 1710.72 €

3) Beneficio Industrial

- 6 % sobre Ejecución Material ..... 641.52 €

4) Honorarios Proyecto

- 800 horas a 15 € / hora ..... 12000 €

5) Material fungible

- Gastos de impresión ..... 200 €
- Encuadernación ..... 15 €

6) Subtotal del presupuesto

- Subtotal Presupuesto ..... 25259.24 €

7) I.V.A. aplicable

- 21% Subtotal Presupuesto .....4304,44 €

8) Total presupuesto

- Total Presupuesto .....30563,68 €

Madrid, Enero 2014

El Ingeniero Jefe de Proyecto

Fdo. Juan Pablo Montero Hidalgo

Ingeniero Superior de Telecomunicación

Apéndice H: Pliego de condiciones

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, **"Implementación de un sistema de comunicaciones basado en Software Radio"**. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.
Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4% del presupuesto y la provisional del 2 %.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad “Presupuesto de Ejecución de Contrata” y anteriormente llamado ”Presupuesto de Ejecución Material” que hoy designa otro concepto.

Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente

proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.