Universidad Autónoma de Madrid
Escuela Politécnica Superior

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Trabajo Fin de Grado

IMPLEMENTACIÓN DE UN SISTEMA DE COMUNICACIONES BASADO EN SDR MEDIANTE GNU RADIO

Autor: Jorge Carpio López de Castro
Tutor: Juan Córcoles Ortega
Ponente: Jorge Ruiz Cruz

Junio 2017

IMPLEMENTACIÓN DE UN SISTEMA DE COMUNICACIONES BASADO EN SDR MEDIANTE GNU RADIO

Autor: Jorge Carpio López de Castro
Tutor: Juan Córcoles Ortega
Ponente: Jorge Ruiz Cruz

Grupo de Radiofrecuencias: Circuitos, Antenas y Sistemas
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio 2017

Resumen

En el presente proyecto se ha desarrollado un sistema de comunicaciones inalámbrico utilizando tecnologías de Radio Definida por Software, haciendo especial énfasis en la modulaciones digitales. Se ha llevado a cabo en un entorno basado en el framework GNU Radio y el hardware USRP.

Se ha propuesto como objetivo principal el envío de ficheros entre un transmisor y un receptor, formados por dos ordenadores y las herramientas anteriormente mencionadas. Puede englobarse el trabajo realizado dentro de los niveles 1 y 2 de la pila protocolaria OSI, pues se han utilizado métodos básicos de paquetización de la información para facilitar la sincronización de símbolo.

Se han tratado algunos de los aspectos clave de los sistemas de comunicaciones inalámbricas, como la sincronización de portadora y símbolo, el bit error rate, la codificación de línea y, sobre todo, las modulaciones digitales. Se han implementado tres sistemas diferentes utilizando modulaciones diversas: modulación en amplitud (ASK), modulación de fase (PSK) y modulación de frecuencia (FSK). Utilizando estos sistemas se han realizado experimentos con el objetivo de estudiar su rendimiento.

Además, se han investigado los medios de trabajo utilizados (GNU Radio y USRP), sus características esenciales, las herramientas que proporcionan para la construcción de sistemas de comunicación y su potencial de cara al futuro, incluyendo algunos proyectos en desarrollo.

Palabras Clave

Radio Definida por Software, GNU Radio, comunicaciones inalámbricas, C++, Python, diagrama de flujo, Linux, código abierto, modulaciones digitales, USRP, Ettus Research, ASK, PSK, FSK

Abstract

This BSc thesis has been aimed to the design and implementation of a wireless communications system using Software Defined Radio technologies, using a set of different digital modulation schemes. The system has been built using the GNU Radio framework and the USRP hardware.

The main goal for this project is to achieve a successful communication between a transmitter and a receiver, which will be host-based SDR systems. The whole of the work carried out in this project sits within the first two levels of the OSI protocol stack, since a very basic packetizing scheme has been used to achieve symbol synchronizaton.

Some of the key aspects of the wireless communications systems have been studied, such as symbol and carrier synchronization, bit error rate, line coding but, mainly, digital modulation techniques. Three different systems have been implemented, each using one of the following modulations: Amplitude Shift Keying (ASK), Phase Shift Keying (PSK) and Frequency Shift Keying (FSK). Different tests have been carried out to validate the systems and measure its performance.

Furthermore, some research has been performed on the tools above mentioned (GNU Radio and USRP) and how they can be used for the development of SDR-based communication systems. Some future prospectives have been given out, mentioning some of the current research trends.

Key words

Software Defined Radio, GNU Radio, wireless communications, C++, Python, flowgraph, Linux, open source, digital modulations, USRP, Ettus Research, ASK, PSK, FSK

Agradecimientos

En primer lugar quiero agradecer a Juan su ayuda y su paciencia este año, además de darme la oportunidad de llevar a cabo este proyecto. Gracias también a aquellos profesores de la EPS que me han ayudado a orientar mi carrera. Gracias al personal de las administraciones de los centros por los que he pasado.

Gracias a mis amigos de siempre por seguir ahí y aceptarme como soy, y a Elisa por abrirme camino en las convalidaciones. Gracias a Paula y a su familia.

Por último, gracias a mi familia: a John por su ejemplo, a Fer por ser el hermano definitivo, a Lola por ser una bola peluda, a mi padre por apoyarme y a mi madre por su fé incondicional.

Índice general

Índice de Figuras

Índice de Cuadros

1

Introducción

1.1. Descripción del proyecto y objetivos

El sistema a construir en este proyecto está formado por una cadena de tratamiento de señal digital, implementada con GNU Radio, y un hardware transceptor para la emisión y recepción de las ondas electromagnéticas, el Universal Software Radio Peripheral (USRP). Se emplearán varios esquemas de modulación digital dentro de los sistemas desarrollados con el objetivo de contrastar su rendimiento en las pruebas realizadas en el capítulo 5.

La estructura del sistema de comunicaciones estará formada por dos subsistemas independientes, uno para la transmisión y otro para la recepción de los datos, compuestos cada uno por un PC[1] con un USRP N210 conectado a él. En cada host se ejecutará la aplicación GNU Radio correspondiente al transmisor o al receptor. El flujo de datos será exclusivamente unidireccional.

El objetivo principal es la correcta recepción de un fichero transmitido entre los hosts a través del sistema implementado. Como objetivo secundario, se van a estudiar en el sistema desarrollado el espectro en transmisión, las velocidades de bit alcanzables y las capacidades de transmisión y recepción. Se perseguirán estos objetivos para cada una de las modulaciones.

[1] Más comúnmente referido como host

1.2. Motivación

En el mundo de las comunicaciones móviles existe un amplio conjunto de estándares que definen el nivel físico de las comunicaciones. Por si fuera poco, la llegada de una nueva generación de estándares móviles cada 10 años ha obligado históricamente a los terminales móviles a dar soporte a multitud de modos de acceso radio a través de numerosos chips especializados diseñados de forma individual. Desde la llegada del 4G, las tecnologías SDR ha permitido ahorrar recursos hardware llevando al dominio software varias de las tareas realizadas por los circuitos integrados especializados. Con la implantación de la SDR se pretende llegar al receptor universal [5], con un único front-end de radiofrecuencias que dé soporte a todos los estándares de comunicación implementados en el software.

Al tiempo que la radio SDR se ha visto impulsada por la industria y las aplicaciones militares, se han ido formando conjuntos de investigadores que, atraídos por la flexibilidad de esta nueva tecnología, han desarrollado tanto herramientas software como hardware que han dado paso a los entusiastas y aprendices a formar la comunidad SDR. Hoy en día es posible, con un software gratuito y un dispositivo de 30€, construir receptores de radio y capturar señales de diversa naturaleza[2] en todo el espectro, desde los 20 hasta los 1700 MHz[3].

1.3. Plan de trabajo y material disponible

El plan de trabajo se ha dividido en las siguientes etapas:

1. Familiarización con el entorno de trabajo: GNU Radio – USRP. Realización de tutoriales [7] y pruebas básicas. Duración aproximada de 2 meses. Reflejado en los capítulos 2 y 3.
2. Implementación de la parte SDR de los sistemas en GNU Radio. Validación interna (sin el uso de los USRP). Duración 4 meses. Trabajo recogido en los capítulos 4 y 5.1.
3. Validación externa: pruebas de los sistemas con los USRP y transmisión de ficheros. Duración 1,5 meses. Contenido en el capítulo 5. Objetivo primario del proyecto.
4. Evaluación del rendimiento de los sistemas implementados y búsqueda de posibles mejoras. Duración: 2 semanas. Recogido en los capítulos 5 y 6. Da cobertura a los objetivos secundarios del proyecto.

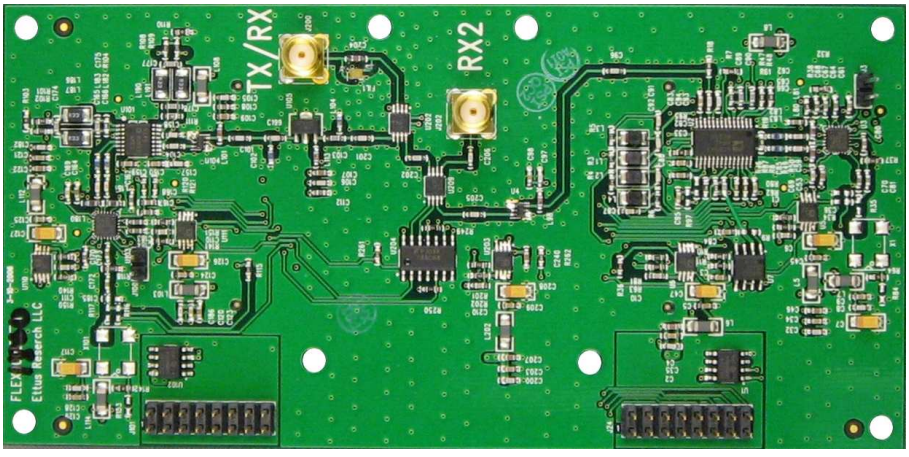

Figura 1.1: Daughterboard de la serie RFX

[2]Es ilegal en algunos países no sólo la emisión, sino la recepción de señales en bandas de frecuencia no ISM [6]
[3]ver RTL-SDR, capítulo 2.3.2

Las pruebas del sistema sobre hardware se han llevado a cabo en el laboratorio C-107 de la EPS. El material del que se dispone para la gestión del proyecto, implementación del sistema y realización de las pruebas queda recogido en la siguiente lista.

- 2 USRP N210 rev.4: este modelo de USRP tiene una interfaz Gigabit Ethernet para la comunicación con el host.
- 2 daughterboards RFX2400: placa transceptora operativa en la banda de 2.3 a 2.9 GHz [8].
- 2 daughterboards XCVR2450[4]: placa transceptora dual-band a 2.4 y 5 GHz [8].
- 2 PCs con Linux. Con interfaces de red cableada compatibles con Gigabit Ethernet.
- Cables Ethernet (Cat.6) para la conexión de los hosts con los USRP.
- Antenas monopolo.
- Cables coaxiales SMA-SMA para pruebas entre USRP.
- Atenuadores -20 dB para pruebas con cable coaxial.
- Analizadores de espectros.
- Dongle Wi-Fi para una red dedicada a la comunicación entre hosts, usado para tareas de gestión de los procesos GNU Radio.
- Radio RTL-SDR para familiarización con GNU Radio.
- Raspberry Pi para el repositorio Git del proyecto.

[4]No utilizada en la implementación y pruebas del sistema

2

Estado del Arte

ESTE capítulo se propone como introducción a las herramientas utilizadas, teniendo en cuenta su estado actual. En un principio se hablará del framework GNU Radio desde el punto de vista del usuario. Concluiremos haciendo un repaso de los dispositivos hardware compatibles con GNU Radio, haciendo especial énfasis en el USRP.

2.1. Radio Definida por Software

El término de Radio Definida por Software (SDR) lleva 30 años en el panorama tecnológico mundial [9]. Según el Wireless Innovation Forum, se define como "un sistema radio en el que algunas o todas las funciones de la capa física están definidas por software." Al contrario de la imagen que se suele formar, no tiene tanto que ver con el front-end RF como con la capa física de los sistemas de comunicaciones.

Los sistemas basados en SDR se caracterizan por su alta flexibilidad. Se puede modificar su funcionalidad mediante cambios en el software, sin necesidad de sustituir módulos hardware (salvando las limitaciones de la etapa front-end utilizada). La tendencia es la búsqueda de un receptor universal [5], capaz de operar en todos los modos y con todos los estándares de manera completamente configurable.

La radio SDR permite configurar la capa física de las comunicaciones de forma dinámica. Esto está dando paso al desarrollo de radios cognitivas, que son capaces de seleccionar de forma inteligente el canal de frecuencias a utilizar, modificar el esquema de modulaciones utilizado, cambiar la velocidad de las comunicaciones o incluso utilizar varios esquemas para información con diferentes requerimientos. Todo ello de forma autónoma y basándose en el entorno que les rodea.

Las industrias que han implantado la tecnología SDR incluyen la militar y la telefonía móvil. Su desarrollo ha estado marcado por los avances en los mundos del software y los semiconductores. Prácticamente todas las estaciones base 4G se han desarrollado utilizando FPGAs y RFIC [9]. Por parte del software, se aplican algoritmos matemáticos cada vez más complejos para realizar operaciones de forma eficiente. En la industria, la SDR se puede implementar utilizando ASICs o DSPs integrados en FPGAs. Los procesadores de propósito general, a pesar de haber sido útiles para el desarrollo de esta tecnología, no cubren los requerimientos de rendimiento para nuevas generaciones de estándares como 5G o IoT.

2.2. GNU Radio

GNU Radio es un entorno de desarrollo para sistemas de Radio Definida por Software. Es un software de código abierto[1] que está compuesto por una serie de módulos y librerías escritos en C++ y Python. Es posible tanto el uso directo de estas herramientas de forma programática como el indirecto, a través de una interfaz de usuario, conocida como GNU Radio Companion, la cual trataremos más adelante en su correspondiente sección.

Figura 2.1: Logo de GNU Radio [1]

El proyecto de GNU Radio surgió en 2001, iniciado por John Gilmore, fundador de la EFF (Electronic Frontier Foundation). Comenzó a desarrollarse a partir de un proyecto del MIT, llamado Pspectra, aunque actualmente nada de este código original permanece. Cuenta con una comunidad activa de usuarios y desarrolladores, que organizan una feria de conferencias anual (GRCon). El proyecto cuenta con un sitio web principal [1] y una wiki de referencia[2] para usuarios. Los desarrolladores pueden consultar la página de Doxygen[3] que sirve como referencia para el código C++ del proyecto. Además, existe una mailing list activa disponible para consultas, sugerencias y noticias. GNU Radio está desarrollado principalmente para su uso en entornos Linux.

GNU Radio está diseñado para funcionar mediante el uso de diagramas de flujo. Existe un código central de aplicación que conecta bloques de tratamiento de señal que realizan operaciones sobre flujos de muestras. Existen bloques especiales dedicados a interconectar procesos de GNU Radio a través del sistema operativo con diversos elementos de entrada/salida, como ALSA (gestor de E/S de audio de Linux), ficheros, la GUI (Qt), o las interfaces de red. Pero la pieza de comunicación más importante probablemente sean los drivers para el hardware SDR. Existen diversas alternativas dentro del hardware compatible con GNU Radio. Algunas de ellas se detallan en el cuadro 2.1.

Figura 2.2: Código de colores de los tipos más utilizados en el GRC

A continuación se definen algunos de los términos y conceptos utilizados en el entorno de GNU Radio:

Scheduler En GNU Radio, los diagramas de bloques se ejecutan de forma concurrente, cada bloque tiene su propio hilo. El scheduler es el motor interno de GNU Radio. Se ocupa de la coordinación de todos los hilos para que los recursos dedicados a cada uno estén equilibrados [10].

Puerto Punto de entrada o salida de un bloque. Los datos manejados pueden ser de varios tipos, de los cuales se muestran los más utilizados durante este trabajo en la figura 2.2

Bloque Representan un subsistema que realiza una determinada operación[4]. Pueden tener cualquier número de puertos entrada/salida. Pueden ser construidos de forma programática (en C++ o Python) o gráficamente mediante un diagrama de bloques jerárquico. Desde el punto de vista del código, un bloque no es más que un objeto de una clase con un formato determinado en el

[1] Código fuente de GNU Radio disponible en `https://github.com/gnuradio/gnuradio`
[2] Wiki GNU Radio: `https://wiki.gnuradio.org`
[3] Referencia para desarrolladores: `https://gnuradio.org/doc/doxygen`
[4] Serán representados en cursiva a lo largo de esta memoria.

que se definen las interfaces de entrada/salida y se implementan las operaciones de procesado de señal.

Diagrama de bloques Como su nombre indica, los también llamados *flowgraphs* interconectan bloques generando un sistema completo listo para ser ejecutado. Aunque existe la posiblidad de crearlo en C++, normalmente lo veremos como un script Python generado automáticamente si estamos utilizando el GRC.

Bloque jerárquico Son bloques especiales en cuanto a la forma de ser construidos. En vez de estar definidos por código directamente, se construyen mediante un diagrama de bloques en GRC. Por tanto, su comportamiento depende del de otros bloques. Ayuda a mantener más ordenados los flowgraphs.

Módulo Los bloques en GNU Radio están organizados en paquetes software o módulos, normalmente según las aplicaciones a las que estén destinados, y desarrollados por diferetes agentes. Los módulos son fácilmente identificables por su nombre: todos ellos comienzan por "gr-". Ejemplos de módulos son gr-osmocom y gr-digital.

Existe una gran variedad de bloques disponibles. Los incluidos en GNU Radio por defecto son, a fecha de este trabajo, 521. Sin embargo, existe toda una red de desarrolladores que comparten sus propios módulos con la comunidad de GNU Radio. Pueden ser encontrados en el directorio de CGRAN[5]. Entre los módulos comunitarios más populares se encuentra *gr-ieee802-11*, que implementa la capa física de la familia de estándares IEEE 802.11, extensamente utilizada en redes Wi-Fi.

2.2.1. La interfaz gráfica: GNU Radio Companion

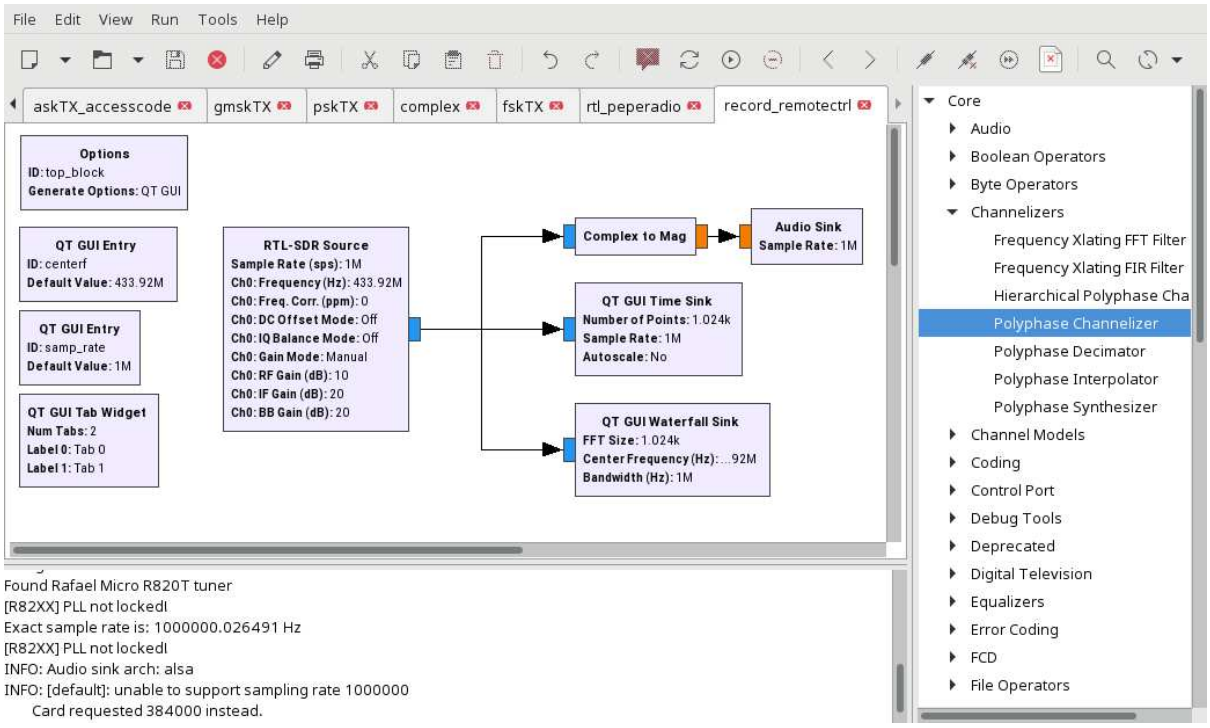

Figura 2.3: Interfaz GNU Radio Companion

GNU Radio cuenta con una interfaz gráfica (figura 2.3) que nos permite diseñar de forma interactiva los diagramas de flujo para manejar el tratamiento de las señales. El proceso para la creación de un diagrama puede ser resumido en los siguientes puntos:

[5]Sitio web CGRAN: `http://www.cgran.org/index.html#projects`

1. Se crea un nuevo flowgraph vacío, que contiene por defecto la variable *samp_rate*, mediante la cual podemos dar valor la frecuencia de muestreo del diagrama.
2. Se instancian y conectan los bloques, incluyendo al menos un bloque generador y un sumidero.
3. Se da valor a los parámetros relevantes de los bloques, creando variables si así se desea.
4. En el caso de que el diagrama no esté destinado a usarse con ningún hardware, es necesario añadir un bloque *Throttle*, que se encargará de limitar el uso de recursos del SO en la ejecución. Si el diagrama incluye ya un bloque hardware, este se encargará establecer los límites.
5. Finalmente, se genera automáticamente un script de Python correspondiente al diagrama pinchando un botón en la GUI. Es posible también ejecutar el diagrama sin necesidad de generarlo antes. Puede configurarse la generación del script como bloque con GUI, sin GUI o como bloque jerárquico.

La interfaz está construida utilizando el framework Qt. Además de permitirnos construir diagramas de manera interactiva, existen bloques específicos para la interacción del usuario con los flowgraph. Con ellos es posible variar parámetros (ej: *QT GUI Range*) o para visualizar el comportamiento del sistema (ej: *QT GUI Frequency Sink*).

2.3. Hardware SDR compatible

La tecnología SDR es posible entre otros factores gracias a los avances en la electrónica digital [9], que han permitido que implementar el tratamiento de señal en software sea lo suficientemente eficente para ser usada en aplicaciones reales. Los módulos que proporcionan los drivers para el hardware SDR más común en GNU Radio son:

1. *gr-uhd*: proporciona soporte para el USRP, desarrollado por Ettus.
2. *gr-osmocom*: desarrollado por Osmocom, soporta RTL-SDR, hackRf, bladeRF, UmTRX, Funcube y USRP.
3. *gr-baz*: otro driver para el aclamado RTL-SDR.

Existen dos dispositivos de considerable relevancia en el mundo de GNU Radio y la SDR que vale la pena destacar. Estos son el USRP y el RTL-SDR. El primero es el dispositivo más usado por profesionales, investigadores y entusiastas de GNU Radio y la SDR. El segundo es una familia de receptores de distintos fabricantes, inicialmente diseñados para servir de receptor para DVB-T TV, que han sido adoptados en el mundo SDR por su bajo coste.

Pero el hardware SDR no se limita a estas dos opciones. Existe una oferta cada vez más variada, que cubre un amplio espectro de aplicaciones. Llama la atención la fuerte presencia de proyectos de "Open Hardware", que ponen a disposición de la comunidad los esquemáticos y diagramas hardware. En el cuadro 2.1 se muestran algunas de las radios SDR más exitosas.

2.3.1. El hardware USRP

Se puede decir que el hardware SDR por excelencia en lo que respecta a GNU Radio es el USRP, desarrollado por Ettus Research. Esta organización nació de la mano de Martin Ettus, un antiguo colaborador de GNU Radio. El USRP1, que fue el primer modelo, data de 2003.

Todos los USRP contienen una FPGA[6] para manejar el procesamiento de las muestras. Están pensados para ser complementados por placas secundarias o *daughterboards*, que implementan el hardware de radio frecuencias más crítico. Esta etapa es también llamada RF front-end.

[6] Actualmente los USRP usan FPGAs de Xilinx, a pesar de haber comenzado con Altera en el USRP1

	RTL-SDR*	HackRF	BladeRF	USRP*
Tipo	RX	TX-RX, half duplex	TX-RX full duplex	TX-RX full duplex
Precisión	8 bits	8 bits	12 bits	ADC 14 bits, DAC 16 bits
Rango de frecuencias	22 a 2200 MHz	1 MHz a 6 GHz	300 MHz a 3.8 GHz	DC – 6 GHz
Velocidad de muestreo	2,56 MSps	20 MSps	40 MSps	200 MSps
Interfaz	USB 2	USB 2 HS	USB 3	Variable
Coste	30€	275€	375€	767 a 5511€

Cuadro 2.1: Comparativa de hardware SDR

Las especificaciones de para los dispositivos marcados con (*), se han extraído de todos los modelos disponibles.

En el anexo A.3 se exponen las especificaciones del dispositivo utilizado para este proyecto, el USRP N210 (figura 2.4). Se ha observado en experimentos que la potencia máxima que es capaz de generar es del orden de los 7 dBm. Ettus recomienda en las notas de aplicación del USRP no entregar más de -10 dBm [11] al dispositivo a riesgo de dañar la cadena de RF de entrada. Por ello, es necesario el uso de atenuadores al hacer uso de la conexión directa entre terminales.

Figura 2.4: USRP N210

UHD: el driver para USRP

La interacción de los hosts con el USRP la posibilita el driver UHD [12], escrito en C++. Está disponible para Linux, Mac OS y Windows y es utilizado no sólo por GNU Radio, sino también por LabVIEW[7], y Matlab/Simulink. Otros proyectos se han construido directamente sobre la API de UHD, como OpenBTS2.3.1.

El driver UHD incluye varias aplicaciones para la configuración o el testeo del USRP [11] a través de la línea de comandos que incluyen, entre otras:

1. `uhd_find_devices`, `uhd_usrp_probe`: testeo del reconocimiento del USRP por parte del host.
2. `uhd_image_loader`, `uhd_images_downloader`: configuración de las imágenes FPGA y firmware del USRP.
3. `uhd_siggen`, `uhd_fft`, `uhd_rx_nogui`: para la generación o recepción de señales con el USRP.
4. `uhd_cal_rx_iq_balance`, `uhd_cal_tx_iq_balance`: aplicaciones para la generación de tablas de calibración del USRP.

Estructura interna del USRP

El diagrama de bloques de la figura 2.5 representa la estructura interna genérica del conjunto USRP – daughterboard. La ruta que siguen los datos en transmisión[8] es:

[7] Software de ingeniería de sistemas desarrollado por National Instruments, propietaria de Ettus Research desde 2010

[8] La cadena de recepción es casi idéntica, pero a la inversa y con una capacidad menor de conversión A/D (100 MSps en vez de las 400 MSps que proporciona el DAC)

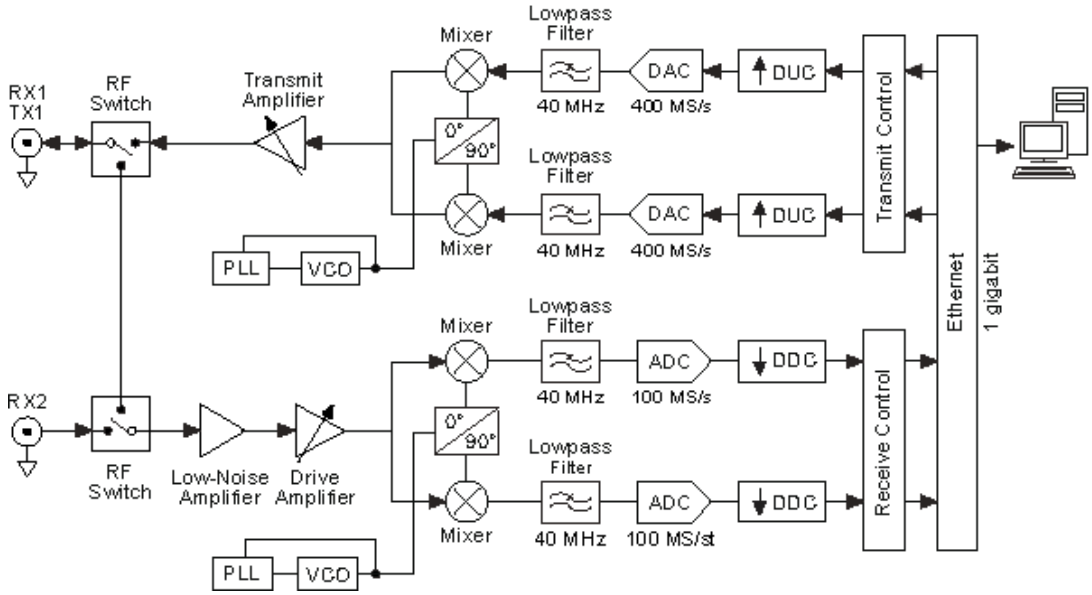

Figura 2.5: Diagrama del sistema USRP + daugtherboard [2]

1. A través de la interfaz con el host, el USRP recibe las muestras complejas fase/cuadratura de la señal en banda base, con una velocidad máxima de 25 MSps[9] para el USRP N210.
2. En la FPGA, las muestras pasan por un banco de filtros interpoladores que las adapta a la frecuencia de muestreo del reloj de referencia utilizado[10].
3. A continuación pasan por la etapa de conversión de frecuencia digital (DUC), también contenida en la FPGA.
4. Un par de conversores DAC convierten las muestras I/Q al dominio analógico.
5. En la daughterboard se efectúa el filtrado, modulación de cuadratura y amplificación. Se mezclan las señales y se transmiten por el puerto de salida, normalmente conectado a una antena. Cada daughterboard tiene un ancho de banda y una frecuencia central de funcionamiento. En [2] se resumen las características de algunas de ellas.

Sintonización de frecuencias en el USRP

El proceso de sintonización para la conversión en frecuencia se divide en dos etapas [13]: una etapa RF implementada en la daughterboard y otra controlada por la FPGA. La conversión RF es de tipo discreta y normalmente no proporciona suficiente precisión para la sintonización de las frecuencias requeridas por el usuario, lo que crea la necesidad de una segunda etapa.

En un proceso normal de sintonización, el oscilador RF se ajustará a la frecuencia más cercana a la requerida. Será entonces el DSP quien se encargue del ajuste fino de la sintonización. Es posible también el ajuste de cada una de las etapas por separado haciendo uso del objeto `uhd::tune_request_t`. Esta alternativa permite reducir la componente DC y aumentar el aislamiento entre etapas a diferentes frecuencias, por lo que es la aconsejable.

Algunas daughterboards, como la RFX2400 utilizada en este proyecto, incluyen dos osciladores independientes para las cadenas de transmisión y recepción. Esto posibilita el uso de diferentes bandas de frecuencia para links de comunicaciones full-duplex.

[9] 25 MSps utilizando muestras I/Q con 16 bits para componente. 50 MSps si se utilizan muestras de 8 bits

[10] Por defecto, 100 MHz en el USRP N210. Otras frecuencias configurables son 200 MHz y 184,32 MHz. También se puede conectar un reloj de referencia.

Notas sobre el rendimiento en los hosts

En los sistemas SDR basados en host, el punto crítico para el rendimiento de los sistemas suele ser la interfaz entre el dispositivo SDR y el PC. Debido a ello, las diferentes generaciones de USRP[11] han variado de interfaz física. Se han utilizado estándares desde USB 2.0, pasando por PCIe, hasta Thunderbolt, la tecnología que actualmente propicia el mayor throughput.

Por ser de propósito general, los PC no suelen estar configurados para poder interactuar con el USRP de forma óptima. Podemos aumentar el rendimiento del sistema siguiendo algunos de los métodos explicados en el anexo A.2.

Investigaciones en curso y proyectos amigos

Actualmente, gran parte de la investigación en el campo del USRP está destinada al desarrollo de RFNoC (RF Network on Chip [14]). Esta tecnología trata de implementar los bloques SDR actualmente ejecutados en GPPs en hardware programable. Investigaciones se están llevando a cabo también para la integración de RFNoC con GNU Radio. De esta forma parte del tratamiento de señal se haría dentro de la FPGA, aprovechando el potencial del hardware programable y reduciendo el impacto computacional en el host, que vería reducida su carga de trabajo. Estos bloques aparecerán como un bloque normal en GNU Radio, pero en tiempo de ejecución, en vez de ser manejados por el scheduler, correrían en la imagen de la FPGA. Las imágenes por defecto para las FPGAs llevarán dentro un set de bloques RFNoC por defecto.

Un proyecto de interés ligado a los USRP es el llamado OpenBTS. Se trata de una aplicación de código abierto para plataformas Linux que permite construir redes móviles 3GPP utilizando tecnologías SDR. Únicamente son necesarios un hardware SDR, una antena para la cobertura de los usuarios, una máquina que sirva de servidor y una conexión a Internet que sirva de red de retorno o backhaul. Aunque hoy este estándar móvil está algo desfasado, ha sido de gran utilidad en situaciones que han requerido aumentar la cobertura móvil en una zona de forma económica y rápida, como eventos masivos en lugares con poca población.

2.3.2. RTL-SDR

Los RTL-SDR eran originalmente receptores destinados al mundo de la televisión digital. Todos ellos tienen en común el chipset RTL2832U [15]. En marzo de 2010 se descubrió que era posible acceder a las muestras raw I/Q de estos receptores [16], lo que significaba información RF en crudo, propicia para ser procesada por un softwrare SDR.

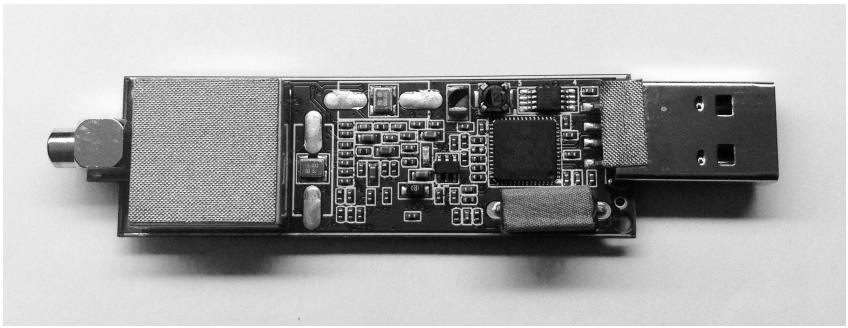

Figura 2.6: Dongle RTL-SDR

[11] Los USRP E310 y E312 suponen una excepción, pues están orientados a el desarrollo de aplicaciones SDR embebidas y no ejecutadas en un host aparte

Debido a su bajo coste y pequeño tamaño, tuvieron una calurosa acogida por la comunidad SDR. Aunque su aplicabilidad queda reducida por el hecho de no soportar la transmisión de ondas, se han encontrado aplicaciones para las que ha demostrado ser una solución apropiada. Algunas de las aplicaciones desarrolladas para este dispositivo son: receptor GNSS [17], receptor de imágenes de satélites meteorológicos NOAA o triangulación de señales desconocidas. Los bloques driver usados para RTL-SDR en GNU Radio han sido desarrollados por el proyecto Osmocom.

La banda de frecuencias soportada por los RTL-SDR depende del chip sintonizador. Los más comunes se muestran en el cuadro 2.2

Sintonizador	Rango de frecuencias
Elonics E4000	52 – 2200 MHz (gap 1100 – 1250 MHz)
Rafael Micro R820T/2	24 – 1766 MHz
Fitipower FC0013	22 – 1100 MHz
Fitipower FC0012	22 – 948.6 MHz
FCI FC2580	146 – 308 MHz y 438 – 924 MHz

Cuadro 2.2: Sintonizadores y frecuencias en dongles RTL-SDR [4]

3

Modelo de Sistema de Comunicaciones

En este capítulo se pretende describir el sistema implementado a partir de un modelo general para los sistemas de comunicación. Se comenzará tratando un modelo de sistema transmisor-receptor, continuando con el modelado específico del transmisor y receptor. Intentaremos mantener un hilo relacional entre el modelo y el sistema a implementar haciendo referencia a algunas características de GNU Radio y el USRP N210.

3.1. Introducción

Antes de introducir el modelo, cabe caracterizar el sistema que nos ocupa. Se trata de un sistema de comunicaciones digitales inalámbrico que se ha simplificado con respecto a los sistemas reales. Esto es debido a que se pretende hacer especial énfasis en las diferentes modulaciones a utilizar. Algunas de las simplificaciones realizadas son:

- Es unidireccional: existen un emisor y un receptor definidos y no intercambiables.
- No se establecen requerimientos con respecto al ancho de banda del canal.
- No se hace uso de codificación de canal.
- No se busca la eficiencia, pudiendo establecer las ganancias necesarias[1].
- No se contemplan interferencias de otros sistemas[2].
- Todas las modulaciones utilizadas son binarias, no se implementará ninguna multisímbolo.

Generalmente, los modelos de comunicaciones pueden separar los sistemas en tres agentes independientes, que se corresponden con las tres filas de bloques de la figura 3.1: transmisor, canal y receptor. Las simplificaciones mencionadas nos permiten obviar el modelado del canal. Únicamente se estudiarán los sistemas de transmisión y recepción.

Todos los bloques visibles en el diagrama de la figura 3.1 serían implementables por software en nuestro sistema, puesto que la interfaz con el USRP es en banda base. Sin embargo, nos centraremos en lo esencial para la implementación de diferentes modulaciones: el modulador/demodulador y, en ciertos casos, la codificación de línea.

[1] Aunque el conjunto USRP N210 – RFX2400 no nos permite manejar la ganancia de transmisión

[2] En caso de detectarlas, se evitarán cambiando la frecuencia de operación. La experiencia, de todas maneras, ha demostrado que la comunicación entre USRP hace más daño a la conexión Wi-Fi que a la inversa.

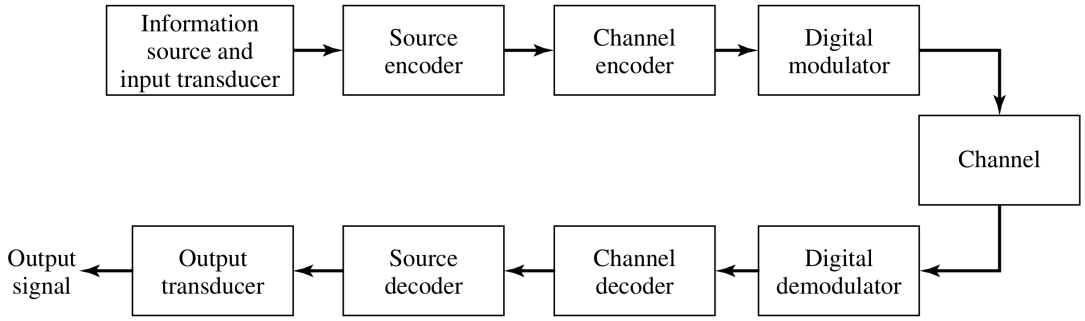

Figura 3.1: Sistema de comunicaciones digitales en banda base [3]

3.1.1. Modulaciones

El objetivo de una modulación es el de transportar la información utilizando frecuencias altas, que suponen una eficiencia mayor. Las modulaciones utilizan como base una señal portadora de la forma:

$$s(t) = A\cos(\omega t + \phi) \tag{3.1}$$

La función de la onda portadora presenta tres grados de libertad: amplitud, frecuencia y fase. Las modulaciones digitales combinan estas magnitudes para formar símbolos[3], que se mapean de forma unívoca a un set de M mensajes digitales, cada uno de longitud $log_2(M)$. Una forma común de clasificar las modulaciones digitales responde a estas tres magnitudes variadas [18]:

- Modulaciones lineales: varían los aspectos lineales del fasor de onda, que son la amplitud y la fase. Se pueden dividir a su vez en:
  - Modulaciones de amplitud: ASK o AM (analógica)
  - Modulaciones de fase: como BPSK, QPSK...
  - Modulaciones amplitud-fase, que varían ambas magnitudes formando un conjunto de M símbolos: MQAM, APSK, etc.
- Modulaciones no lineales. Utilizan la frecuencia de la portadora para transmitir información: ejemplos son FSK, DTMF y MSK. Estas modulaciones suponen una mayor complicación del sistema por la necesidad de uso de elementos no lineales.

Se han seleccionado para su implementación en este trabajo las modulaciones más básicas, pertenecientes a las diferentes clases existentes y que sirven de base para modulaciones más complicadas, de mayor aplicación en el mundo real. Estas son: **ASK**, **BPSK** y **FSK**.

3.2. Modelo de transmisor

El sistema transmisor está formado por:

1. Subsistema modulador: totalmente definido en software. Contiene, además de la lógica para la modulación, otros bloques de tratamiento de señal necesarios para el buen funcionamiento del sistema, como filtros, conversores de formato, etc. Se explicarán más detalladamente en el capítulo 4.
2. Interpolador: conversión entre las frecuencias de muestreo del host y la parte digital del USRP.
3. Conversión fina a frecuencia IF (DUC).
4. Conversión D/A de las muestras.
5. Filtrado y conversión hardware a frecuencia RF con un modulador de cuadratura.
6. Amplificación y emisión por la antena.

[3] Comúnmente representados como constelaciones de puntos en coordenadas cartesianas

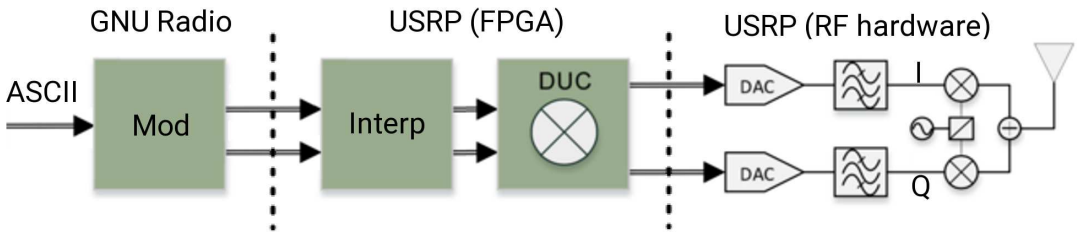

Figura 3.2: Esquema del sistema transmisor

3.3. Modelo de receptor

El receptor es el elemento crítico en un sistema de comunicaciones. Esto se debe a que el proceso para la extracción de información de las ondas electromagnéticas es complejo. Un buen receptor debe presentar las siguientes características [19]:

- Alta ganancia: para recuperar la señal del medio adecuadamente. UHD tiene controles de ganancia para la cadena de recepción y transmisión. La daughterboard RFX2400 proporciona una ganancia de recepción de hasta 70 dB [20].
- Figura de ruido: es importante para no estropear la calidad de la señal. El elemento crítico es el amplificador LNA, que suele ser el primer elemento de la cadena de recepción. La figura de ruido del USRP depende de la daughterboard utilizada, siendo de 8 dB para la RFX2400 [20].
- Selectividad. Es importante, sobre todo en entornos ruidosos e interferentes, que el receptor adapte la banda de recepción al canal que contiene la información y rechaze frecuencias indeseables.
- Conversión a una frecuencia intermedia para su tratamiento. El proceso de selección de frecuencia intermedia del USRP se trata en el apartado 2.3.1.
- Detección de la información: el receptor debe ser capaz de recuperar la información embebida en la señal. El principal responsable de esto es el demodulador. Trataremos con detalle diferentes demoduladores en el capítulo 4.
- Aislamiento: este aspecto es sólo relevante en los sistemas transceptores.[4]. El switch del USRP proporciona aproximadamente 30 dB de aislamiento.

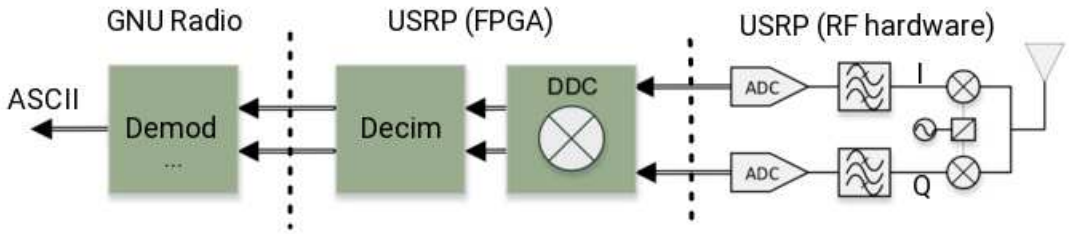

Figura 3.3: Esquema del sistema receptor

Particularizando para el sistema implementado, podemos resumir el camino de la señal recibida dividiéndolo en tres etapas según el contexto en el que se implementan:

1. Daughterboard: amplificación, primera conversión en frecuencia (demodulador de cuadratura) y filtrado de la señal IF.

[4]Cabe mencionar que, aunque el proyecto se ha realizado con dos USRPs (uno para la transmisión y otro para la recepción de señales), hubo un tiempo en el que las pruebas se realizaban sobre un solo USRP. Se abandonó esta práctica cuando se hizo una prueba de aislamiento del USRP, que dio malos resultados. Más información sobre esta prueba en la sección 5.3

2. FPGA: Segundo desplazamiento en frecuencia. Operaciones de diezmado para adaptación a la frecuencia de muestreo del host.
3. GNU Radio. Contiene varios elementos:
   a) Filtrado paso banda: proporciona la selectividad en frecuencia.
   b) Umbral de potencia: impide la demodulación del ruido de fondo.
   c) AGC: control automático de ganancia para compensar las pérdidas.
   d) Demodulador: mapeo de la amplitud, frecuencia o fase instantánea de la señal al símbolo más cercano.
   e) Sincronización de los símbolos para la recuperación de la información en formato nativo (bytes).

Debido a la posibilidad de control separado de las dos conversiones de frecuencia que efectúa el USRP, puede considerarse como un sistema receptor de tipo superheterodino de doble conversión. Sin embargo, se utilizará como sistema de conversión única dejando que el conjunto UHD/USRP seleccione de forma automática la frecuencia intermedia.

4

Implementación en GNU Radio

ESTE capítulo recoge los detalles de la implementación en GNU Radio de los subsistemas moduladores[1] (para ASK, BPSK y FSK) y de los bloques para la sincronización de símbolo. Todas las imágenes han sido capturadas de el GNU Radio Companion.

Todos los moduladores han sido desarrollados como bloques jerárquicos en el GRC. Para simplificar su diseño y utilización, se ha definido un formato estándar para los puertos de entrada/salida:

- Los bloques moduladores deben aceptar como entrada un stream de bytes con información desempaquetada[2]. Su salida serán las muestras I/Q en banda base.
- La entrada y la salida principal de los bloques demoduladores serán inversas a las del modulador: como entrada un flujo de muestras complejas, y como salida símbolos (bytes desempaquetados), que en modulaciones binarias contienen 1 bit.

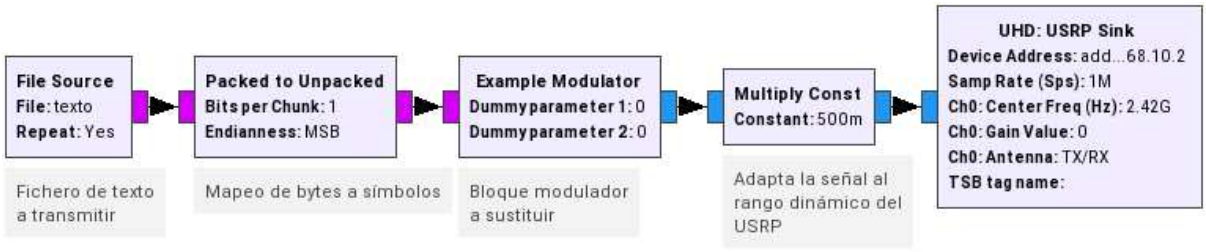

Figura 4.1: Diagrama de bloques básico para usar con diferentes moduladores

Una vez desarrollados los bloques jerárquicos, se integrarán en el diagrama base[3] correspondiente (transmisor o receptor) para su uso. Se ilustra el diagrama base del transmisor en la figura 4.1.

[1] Nótese que, por simplicidad, nos utilizaremos la palabra "moduladores" para referirnos tanto a los bloques moduladores como a los demoduladores. Cuando se quiera hablar de los moduladores excluyendo a los demoduladores se utilizará el sustantivo singular "modulador"

[2] Los moduladores utilizan como unidad básica de información los símbolos, mientras que los procesadores usan bytes. Para poder procesar los símbolos por separado con rapidez, cada uno de ellos debe residir en un byte individual. Es necesario convertir la información de un paradigma a otro reempaquetando la información. Se denominan bytes empaquetados aquellos que contienen información. Un byte desempaquetado contiene $log_2(M)$ bits de información, siendo M el número de símbolos

[3] Diagrama top-level

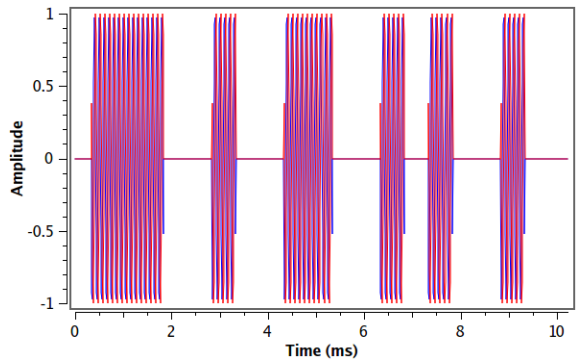

Figura 4.2: Señal ASK

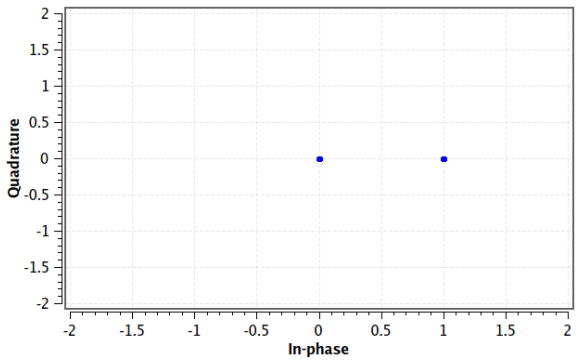

Figura 4.3: Constelación ASK

Notas sobre las expresiones matemáticas

Por coherencia, se usará un conjunto de letras específicas para la representación de diferentes elementos identificables en las expresiones teóricas utilizadas en el resto de este capítulo. A pesar de que los moduladores están implementados en software, la onda modulada es analógica. Por tanto, se utilizará la variable de tiempo continuo para caracterizar las modulaciones. Las expresiones relativas al funcionamiento de los bloques en GNU Radio se escribirán según la notación para el tiempo discreto.

T_s	Periodo de muestreo de la señal digital
ω_c	Frecuencia angular central de la modulación: $\omega_c = 2\pi f_c$
ϕ	Fase de la portadora
$m(t)$	Señal moduladora, información en banda base. Puede tomar dos valores: 0 ó 1
$s(t)$	Señal modulada, resultado de modular de la información

4.1. Modulación ASK

La modulación ASK (Amplitude Shift Keying) es la modulación equivalente a la AM utilizada para en el mundo analógico. La información en banda base modula la amplitud de una señal portadora. La figura 4.2 muestra la modulación ASK de una secuencia de bits en GRC. La expresión general de la señal modulada es la siguiente:

$$s(t) = m(t)\cos(\omega_c t), \quad m(t) \in \{0, 1\} \tag{4.1}$$

El espectro de una señal ASK es de doble banda lateral (DBL), normalmente sin la inclusión de la portadora. Es posible su demodulación de forma coherente (con la recuperación de la portadora), pero lo más común es ver demoduladores incoherentes. La demodulación coherente puede llevarse a cabo añadiendo la portadora a la señal modulada o implementando un circuito específico para la extracción de la portadora en el receptor, comúnmente un PLL.

Aunque se pueden definir varios niveles de amplitud para convertir la ASK en una modulación multisímbolo, la dependencia del decisor de la potencia de entrada complica la definición de varias regiones de decisión. Por tanto, y al ser la ASK utilizada principalmente por su sencillez, no suele verse como modulación multisímbolo. Sin embargo, a pesar de su susceptibilidad al ruido, distancia y otros factores que afectan a la amplitud de la señal, siempre queda la ventaja de ser la modulación más sencilla de implementar a nivel hardware.

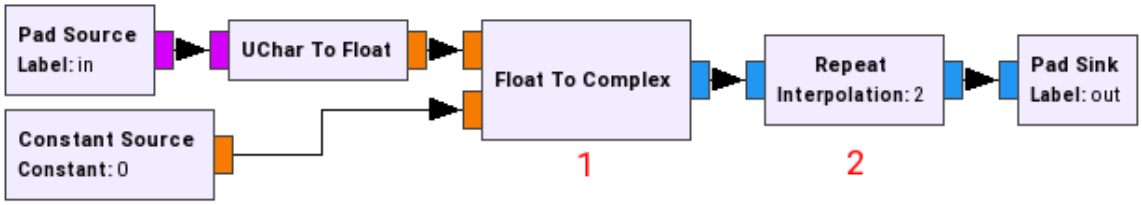

Figura 4.4: Modulador ASK implementado

Nota sobre la construcción de bloques jerárquicos: los puertos de entrada y salida que el bloque exhibirá una vez generado son representados en el diagrama de la implementación por el *Pad Source* y el *Pad Sink*, respectivamente. Los bloques de la implementación son parametrizables desde el bloque jerárquico a través de un *Parameter*.

4.1.1. Implementación del modulador ASK

El modulador puede ser implementado como un oscilador local y un mezclador alimentados por la señal binaria NRZ, en la que el '1' binario se representa mediante un voltaje V, mientras el '0' se representa con voltaje nulo. Como el USRP contiene ya un modulador de cuadratura, el bloque modulador de GNU Radio no necesitará más que un interpolador y algunos bloques para la conversión de tipos. Nótese que la señal mostrada en la figura 4.2 ha sido simulada en un diagrama aparte, añadiendo la mezcla con una portadora.

1. Al variar la ASK la amplitud pero no la fase, no necesitamos emplear para la modulación las dos componentes I/Q representadas por la señal compleja a la salida del modulador. Una señal real habría sido suficiente, pero el bloque *UHD Sink* requiere muestras complejas. Generamos las muestras complejas con su parte imaginaria a cero.
2. Repetición de muestras para alcanzar el número de muestras por símbolo deseado.

4.1.2. Implementación del demodulador ASK

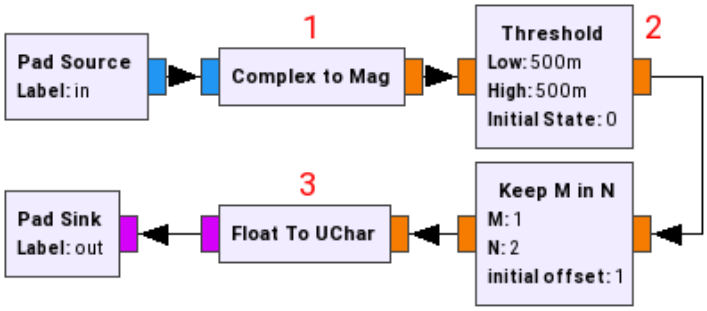

Figura 4.5: Demodulador ASK implementado

Se ha elegido para el diseño un demodulador de tipo incoherente, sin recuperación de la portadora. El demodulador implementado (fig. 4.5) presenta los siguientes bloques:

1. La detección de envolvente se lleva a cabo computando el módulo de las muestras complejas entrantes.[4]
2. El bloque *Threshold* pone a su salida un 1 o un 0 según la señal de entrada esté por encima o por debajo del umbral especificado. Se utiliza como decisor.
3. Diezmado de las muestras. Se pasa de tener N muestras por símbolo a 1 muestra por símbolo.

[4]Sería deseable un filtrado previo de la señal para eliminar posibles interferencias y reducir el ruido.

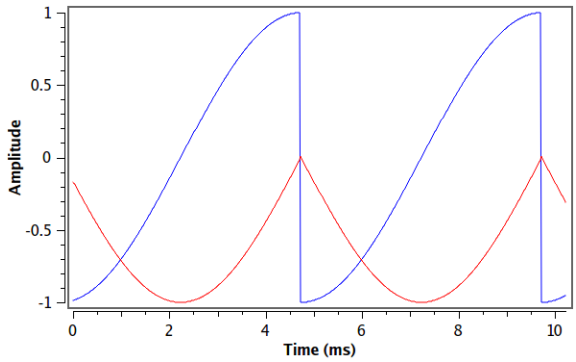

Figura 4.6: Señal BPSK

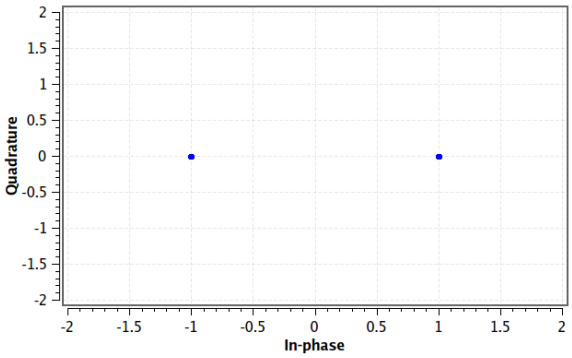

Figura 4.7: Constelación BPSK

4.2. Modulación BPSK

La modulación PSK emplea cambios en la fase de la portadora para modular la información. En su versión binaria (BPSK), implementada en este proyecto, la fase toma valores de 0 y 180°. Puede representarse el funcionamiento de la PSK según la ecuación 4.2.

$$s(t) = \cos(\omega_c t + \phi(t)), \quad \phi(t) = \begin{cases} 0, & \text{si } m(t) = 0 \\ \pi, & \text{si } m(t) = 1 \end{cases} \tag{4.2}$$

Debido a los cambios bruscos de fase, la modulación PSK tiene un espectro muy amplio, a pesar de poder ser reducido mediante filtros. La demodulación debe ser coherente, debido a la necesidad de una fase de referencia. Dada la amplitud constante que tiene la PSK, esta modulación tiene la ventaja de ser menos sensible a la atenuación de la de señal.

4.2.1. Implementación del modulador BPSK

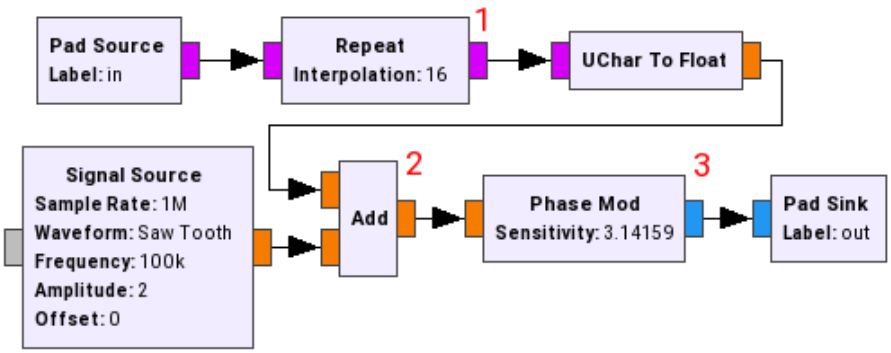

Figura 4.8: Modulador BPSK implementado

El modulador BPSK implementado con el GRC se muestra en la figura 4.8. Se ha aprovechado el bloque *Phase Mod* para añadir la funcionalidad de poder especificar una frecuencia intermedia[5]. El modulador tiene la siguiente estructura:

1. Interpolación de muestras entrantes [6] para alcanzar el número de muestras por símbolo deseado.
2. El valor de los símbolos (1 o 0) se suma a una señal tipo rampa de amplitud 2 y frecuencia igual a la de la frecuencia intermedia, en el caso de haberla.

[5] La cual se dejará a 0 en todas las pruebas con este bloque

[6] bytes desempaquetados, por tanto conteniendo un símbolo por muestra

3. A través del bloque *Phase Mod*, la señal anterior actúa como fase de la señal final con un factor de escala (Sensitivity) $S = \pi$. La señal $y[n]$ generada tiene la forma de la ecuación 4.3.

$$y[n] = \cos(Sx[n]) + j\,\sin(Sx[n]) \tag{4.3}$$

4.2.2. Implementación del demodulador BPSK

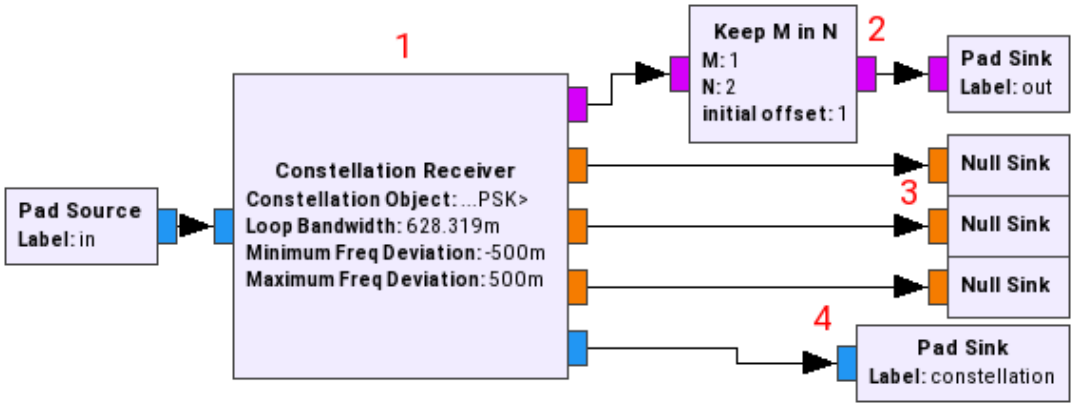

Figura 4.9: Demodulador BPSK implementado

Para la implementación del demodulador se han usado algunos bloques de GNU Radio más complejos, por lo que aquí trabajamos a un nivel mayor. Su estructura es la siguiente (ver figura 4.9):

1. Se ha utilizado el bloque *Constellation Receiver* para extraer los símbolos. Este bloque es un demodulador genérico para modulaciones lineales. Sus parámetros son un objeto *Constellation Object*, que contiene la información relativa a la modulación utilizada (en este caso BPSK), y las frecuencias angulares para el Costas Loop que usa internamente para la recuperación de la fase de la portadora. Este bloque extrae los símbolos (en este caso ceros y unos) y los presenta a la salida en forma de byte desempaquetado.

2. Se diezma desechando N-1 muestras de cada N, donde N es el número de muestras por símbolo. A pesar de que este método para el diezmado es arriesgado, dada la aleatoriedad de la posición de la muestra elegida, la falta de una secuencia de sincronización imposibilita la elección de las muestras más estables dentro de la transmisión del símbolo.

3. Los *Null Sink* desechan la información innecesaria. Son necesarios al no poder dejar estos puertos desconectados.

4. En su último puerto, el *Constellation Receiver* extrae los símbolos demodulados. Como esta información puede ser interesante más adelante, se ha redirigido a un puerto de salida del bloque jerárquico.

4.3. Modulación FSK

La modulación FSK alterna la señal portadora entre dos frecuencias, f_1 y f_2, para representar información binaria. Se llama *desviación de frecuencia* (Δf) a la diferencia entre ambas. La FSK puede expresarse como:

$$s(t) = \cos(\omega(t)t), \quad \omega(t) = \begin{cases} 2\pi f_1, & \text{si } m(t) = 0 \\ 2\pi f_2, & \text{si } m(t) = 1 \end{cases} \tag{4.4}$$

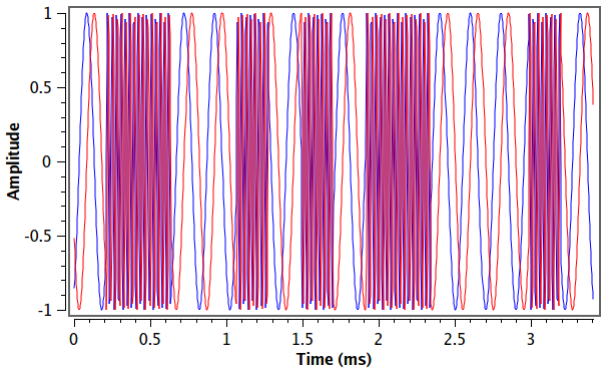

Figura 4.10: Señal FSK

El espectro de la señal FSK es especialmente complejo debido a los cambios de frecuencia. Según [21], en una modulación FSK donde f_{sym} es la velocidad de símbolo (baud rate), el ancho de banda efectivo es:

$$B = 2(\Delta f + 2f_{sym}) \tag{4.5}$$

Es una práctica común la selección de f_1 y f_2 como $f_c - \Delta f$ y $f_c + \Delta f$ respectivamente, donde f_c representa la frecuencia del oscilador local. Esto permite el uso de una codificación de línea tipo NRZ polar[7], que simplifica el decisor en el demodulador, como se verá en la implementación.

4.3.1. Implementación del modulador FSK

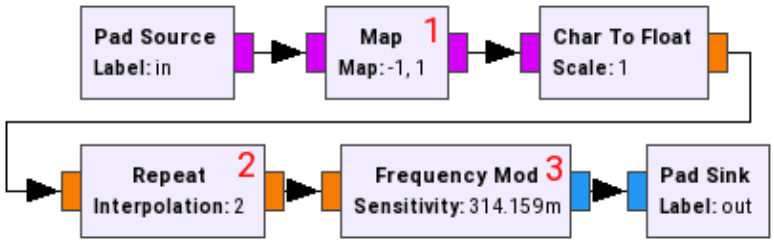

Figura 4.11: Modulador FSK implementado

El modulador implementado puede observarse en la figura 4.11 y está formado por los siguientes bloques básicos:

1. El bloque *Map* es el responsable de la codificación NRZ polar: traduce los 0 y 1 en -1 y 1 respectivamente.
2. Repetición de símbolos para generar el número de muestras por símbolo deseado.

[7] '1' y '0' binarios representados respectivamente por los voltajes V y -V

3. El *Frequency Mod* genera un pulso complejo según la ecuación 4.6, de frecuencia angular normalizada igual a la sensibilidad (Sensitivity) multiplicada por el valor de entrada. La sensibilidad se ha calculado como $S = \Delta\Omega = 2\pi\Delta f T_s$, siendo $\Delta\Omega$ la desviación de frecuencia angular normalizada. De esta forma, este bloque conmuta la frecuencia de su salida $y[n]$ entre $\Delta\Omega$ y $-\Delta\Omega$.

$$y[n] = e^{j2\pi x[n]\Delta f/f_s}, \quad x[n] \in \{-1, 1\} \tag{4.6}$$

4.3.2. Implementación del demodulador FSK

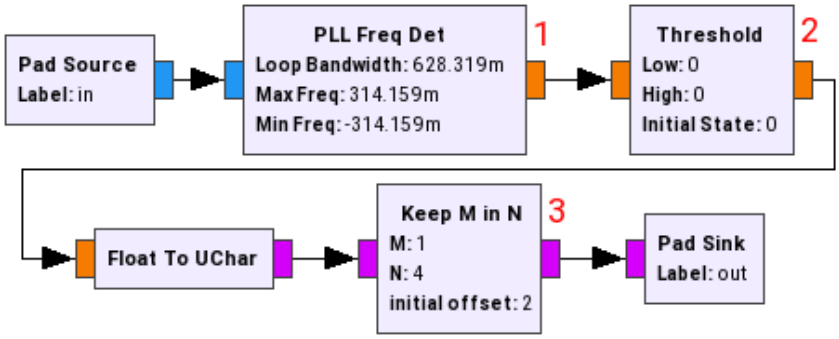

Figura 4.12: Demodulador FSK implementado

El demodulador FSK ha sido implementado siguiendo la siguiente estructura:

1. El bloque *PLL Freq Det* se engancha a la fase de la señal entrante y extrae su frecuencia. Los parámetros introducidos definen las características del PLL. A la salida de este bloque tendremos, idealmente, muestras con valor $\pm\Delta\Omega$, es decir, se ha recuperado la señal NRZ polar, salvo un factor de escala.
2. El decisor queda simplificado debido a que la frontera de decisión para recuperar la información binaria original es 0. El bloque *Threshold* genera un 1 o un 0 en función de si el valor de la muestra entrante está por encima o por debajo de 0.
3. Finalmente, se desechan muestras las muestras repetidas por cada símbolo de la misma manera que en el resto de demoduladores.

La ventaja de la codificación polar NRZ es que el decisor es independiente de la desviación de frecuencia. Únicamente sería necesario asegurarse de que su valor es suficientemente grande para protegerse de los posibles errores de detección generados por el ruido de fase. Sin embargo, al utilizar osciladores definidos por software, la sintetización de frecuencias es muy estable.

4.4. Sincronización de símbolo

Uno de los problemas a que deben resolver los sistemas receptores es la sincronización de símbolo, es decir, la capacidad de determinar la llegada del primer símbolo de una transmisión para así alinear la información y reconstruir los bytes enviados.

Con el fin de conseguir esto, es común el uso de una palabra de sincronización ("access code" en inglés). Esta palabra consiste en una secuencia de bits añadida como prefijo a la información que va a ser transmitida, de forma similar a una cabecera. El receptor realiza correlaciones de forma continua entre los bits demodulados y la palabra de sincronización. Cuando esta es detectada, la información a continuación se considera información relevante. La longitud de la información relevante puede estar definida a priori en el sistema receptor o estar embebida en ella utilizando métodos de paquetización.

Durante las primeras fases de realización de este proyecto no se ha utilizado método alguno para la sincronización de símbolo. Se recibía información desalineada que era sometida a un procesado a posteriori utilizando un script que detectaba y extraía el fichero enviado. Pero este sistema requería conocer, al menos, los primeros bits del fichero y su longitud.

Más adelante, se ha incluido la sincronización de símbolo en los diagramas de GNU Radio. Para ello se han implementado por el método jerárquico dos bloques (figuras 4.13 y 4.14): el primero para la encapsulación de los datos en paquetes con prólogo de sincronización y el segundo para la detección de este y la extracción de los datos encapsulados. Como palabra de sincronización se ha utilizado el access code por defecto proporcionado por GNU Radio[8]. Se han utilizado técnicas de comunicaciones asíncronas y señalización propias de GNU Radio[9].

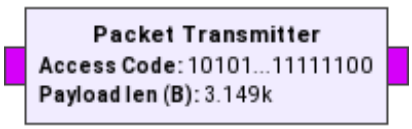

Figura 4.13: Bloque paquetizador que añade la palabra de sincronización de símbolo en el transmisor

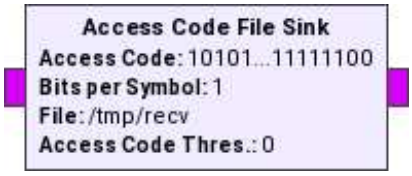

Figura 4.14: Bloque para la sincronización de símbolo en recepción

El bloque de la figura 4.13 es el responsable de prefijar el código de acceso a la información transmitida. Además de ello, añade una pequeña cabecera a la información con un campo de 32 bits conteniendo la longitud de los datos del pàquete. El bloque *Access Code File Sink* a su derecha detecta el código de acceso en el stream de bits entrante, extrae la longitud del paquete y añade sus datos al fichero especificado, además de sacarlos por el puerto de salida. Ambos bloques se han construido siguiendo el método jerárquico con GRC según se muestra en las figuras 4.15 y 4.16.

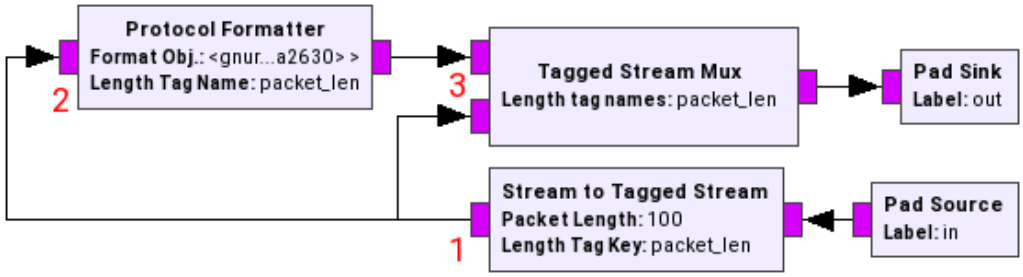

Figura 4.15: Implementación del bloque *Packet Transmitter*

En la figura 4.15 se muestra la implementación en GRC del bloque *Packet Transmitter*. Su funcionamiento se resume en los siguientes puntos:

1. El bloque *Correlate Access Code - Tag* añade una etiqueta[10] a una muestra de cada X muestras, donde X viene indicado por el parámetro "Packet Length".
2. El *Protocol Formatter* genera la cabecera de los paquetes, según el formato especificado en el parámetro "Format Obj.", que en nuestro caso es un objeto `digital.header_format_default`,

[8] `gnuradio.digital.packet_utils.default_access_code: 1010 1100 1101 1101 1010 0100 1110 0010 1111 0010 1000 1100 0010 0000 1111 1100`

[9] Respectivamente, PDUs y tags. Un buen documento de referencia es [22]

[10] Las etiquetas [23] o tags en GNU Radio son metadatos asociados a una muestra en concreto. Son utilizadas para añadir señalización a los flujos de datos. Tienen formato clave-valor.

el más simple de los formatos de cabecera definidos por GNU Radio[11]. Esta cabecera está formada por el código de acceso y un campo de 32 bits con la longitud de los datos del payload. Este bloque genera únicamente las muestras de la cabecera del paquete añadiendo una etiqueta con la longitud del paquete a su primera muestra, pero no copia a su salida los datos entrantes. Para añadir el contenido del paquete a la cabecera es necesario el siguiente bloque.

3. El *Tagged Stream Mux* es utilizado para juntar la cabecera del paquete y los datos. Este bloque combina en un mismo flujo dos streams entrantes de datos etiquetados: si llegan a ambos puertos sendas etiquetas con la clave "packet_len" y valores L_1 y L_2, se copian a la salida, del primer puerto de entrada (el superior), L_1 muestras, y del segundo puerto L_2 muestras. Además, la primera muestra de los paquetes combinados se etiqueta con una clave "packet_len" de valor $L_1 + L_2$.

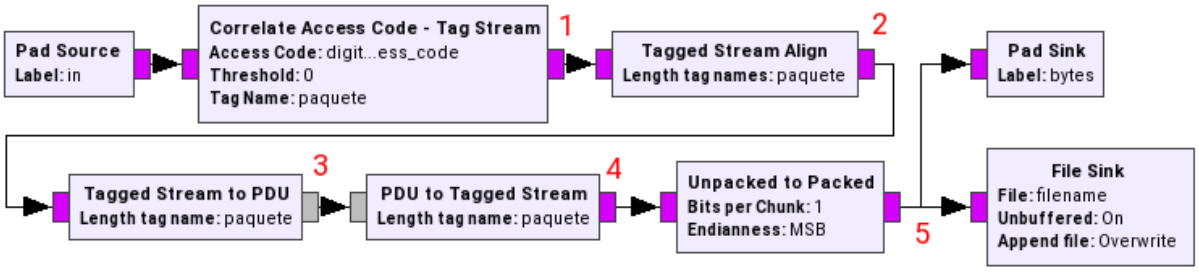

Figura 4.16: Implementación del bloque *Access Code File Sink*

El funcionamiento del bloque (figura 4.16) creado para la sincronización de símbolo en el receptor es el siguiente:

1. El bloque *Correlate Access Code - Tag Stream* es el encargado de la correlación de las muestras entrantes y el código de acceso especificado. El parámetro "Threshold" indica el número de errores de bit permitidos para la correlación. Cuando se detecta el código de acceso, se sacan por el puerto de salida los datos del paquete, con una etiqueta en la primera muestra de clave "paquete" y valor igual a la longitud del paquete extraída de la cabecera.
2. El bloque *Tagged Stream Align* actúa como filtro basado en etiquetas. Copia a su salida de forma alineada los datos entrantes únicamente cuando forman parte de un paquete (señalizado por una etiqueta "paquete" con valor igual a la longitud del paquete). Este bloque es necesario para el correcto funcionamiento de el siguiente.
3. Se crea una PDU[12] por cada paquete entrante.
4. Las PDUs se convierten de nuevo a un stream de datos. Este bloque genera muestras únicamente cuando estas son recibidas como contenido de una PDU. Este es un detalle importante, y es la razón por la que los bloques 2 3 y 4 son necesarios. De esta manera, el siguiente bloque (*Unpacked to Packed*) recibe los bits correctamente alineados para su empaquetamiento.
5. Se empaquetan los bits (bytes desempaquetados) y se escriben en un fichero, a la vez que se envían por el puerto de salida del bloque jerárquico.

[11] Otros incluyen números de secuencia, código de detección de errores u otros campos extra

[12] La PDU en GNU Radio es un tipo de mensaje [24], utlizado como unidad de información transmitida de forma asíncrona, al contrario de las muestras generadas conforme a una frecuencia de muestreo.

5

Experimentos Realizados

Se han realizado pruebas tanto de las unidades individuales como del sistema completo diseñado, en diversos escenarios que se detallan en el apartado descriptivo de cada experimento. Se han utilizado en todos los casos las daughterboards RFX2400.

Con el propósito de elegir frecuencia libre de interferencias se ha llevado a cabo un escaneo de los puntos de acceso Wi-Fi cercanos al entorno de experimentación. Los resultados han mostrado que ningún punto de acceso utiliza los canales 2, 3 y 4. En consecuencia, se ha elegido como frecuencia central para los experimentos 2.42 GHz.

Para la realización de estos experimentos se han optimizado ambos hosts siguiendo algunas de las recomendaciones en el anexo A.2, como el redimensionamiento de buffers y la prioridad de hilos.

Experimento 5.1: Validación de los moduladores en GNU Radio

5.1.1 Descripción del experimento

Se ha simulado en GNU Radio el funcionamiento del sistema con las tres modulaciones implementadas. Para ello, se han preparado sendos diagramas como el de la figura 5.1 para cada una de las modulaciones disponibles. Se han añadido los bloques para la sincronización de símbolo. Los diagramas se han simulado hasta alcanzar una frecuencia de muestreo de 10 MSps utilizando 2 muestras por símbolo.

Para probar el correcto funcionamiento de los sistemas se ha utilizado el bloque *BER*, que compara las muestras demoduladas con las producidas por el *File Source*. La salida de este bloque es el Bit Error Rate (BER), en formato logarítmico. Además, la información demodulada se escribe por pantalla para asegurar la llegada de muestras.

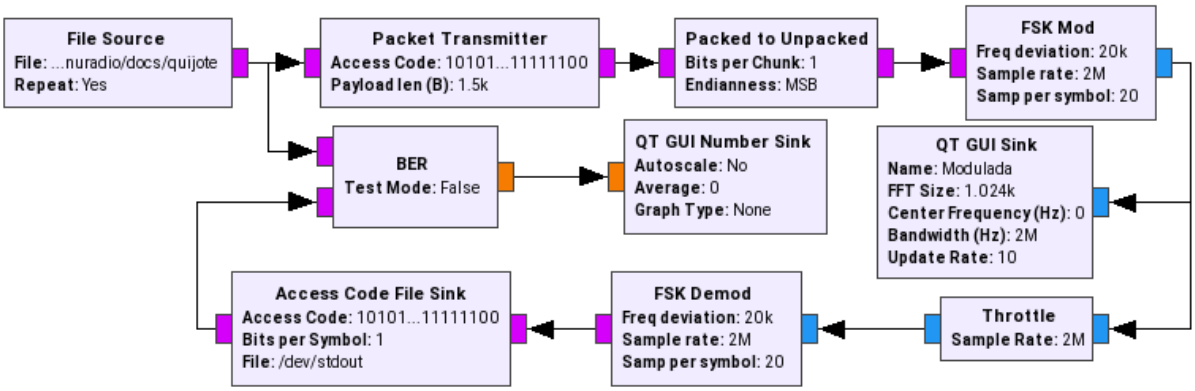

Figura 5.1: Diagrama para validación interna

5.1.2 Resultados

Todos los diagramas han producido un bit error rate nulo hasta la frecuencia de muestreo mencionada. Podemos, por ello, considerar los moduladores y demoduladores diseñados como válidos para ser empleados en el sistema real, que operará como máximo a 10 MSps para evitar underruns y overflows excesivos.

Experimento 5.2: Análisis del espectro de las modulaciones

Se ha observado en un analizador el espectro de la transmisión con las diferentes modulaciones con el objetivo de:

1. Testear la transmisión de señales con el USRP N210
2. Comparar el espectro de las señales moduladas con el de las señales en banda base observadas en GNU Radio.

5.2.1 Descripción del experimento

Estas pruebas han consistido en el uso de un analizador de espectros[1] para la visualización del espectro de cada modulación. Además, se ha visualizado un pulso simple para obtener una idea de la máxima potencia alcanzable en transmisión.

Se ha conectado el USRP con un cable coaxial SMA-N al analizador ajustado a la frecuencia central de 2.42 GHz, visualizando un ancho de banda igual a la frecuencia de muestreo utilizada: 5 MHz. Se ha utilizado un número bajo de muestras por símbolo: 2 para ASK y PSK, 4 para FSK, cercano al que utilizaremos para las comunicaciones en los experimentos sucesivos.

5.2.2 Resultados

Pulso continuo

El pulso de la figura 5.2 ha sido generado enviando una constante con valor 0,5 al USRP a través de GNU Radio. Lo que se puede observar es el espectro del oscilador local utilizado para la modulación de cuadratura. La potencia de salida es de 7 dBm.

[1] Rohde & Schwartz FSM

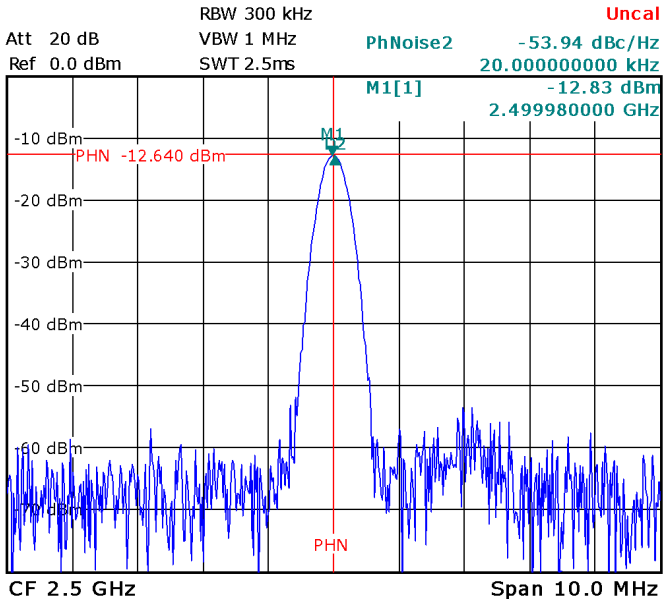

Figura 5.2: Espectro de pulso a 2.5 GHz emitido con el USRP

Modulación ASK

El espectro de la ASK presenta la forma esperada a excepción de dos picos laterales, que podrían ser frecuencias espúreas originadas por la saturación de un componente. Alternativamente podrían ser debidos a la sintetización de frecuencias en el DAC o a la conversión en frecuencia de la señal (por una selección automática de la frecuencia intermedia demasiado cercana a la frecuencia final).

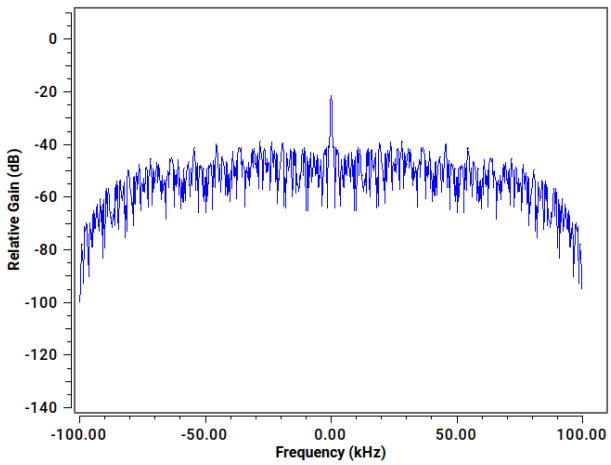

Figura 5.3: Espectro ASK en GNU Radio

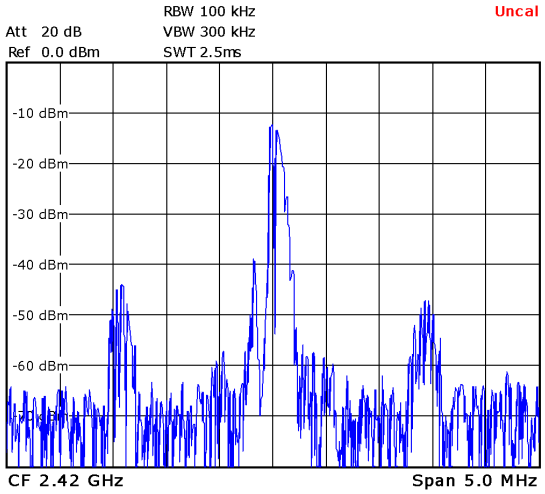

Figura 5.4: Espectro ASK en el analizador de espectros

Modulación PSK

La modulación PSK muestra un espectro con alto contenido frecuencial, como ya se adelantó en el capítulo 4.2. Las frecuencias centrales del espectro se levantan según se aumenta el número de muestras por símbolo.

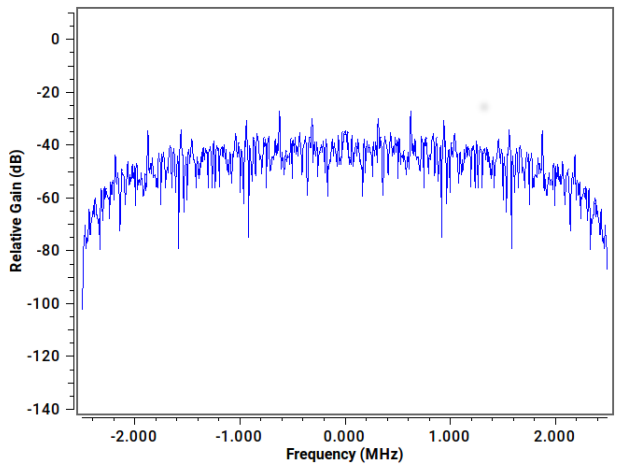

Figura 5.5: Espectro PSK en GNU Radio

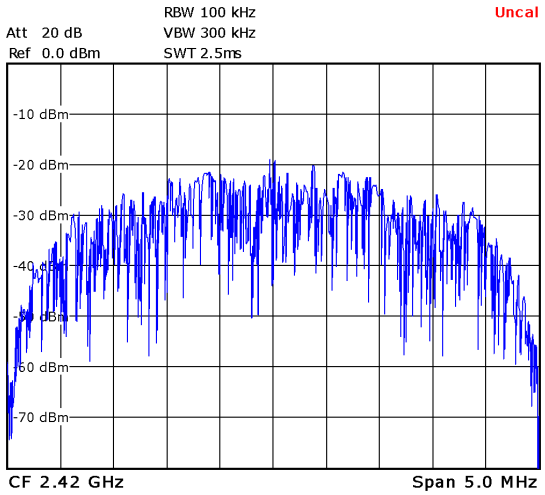

Figura 5.6: Espectro PSK en el analizador de espectros

Modulación FSK

La desviación de frecuencia utilizada para la generación de la señal FSK ha sido de 500 KHz. El contenido frecuencial del espectro se eleva entre las dos frecuencias principales con respecto al resto.

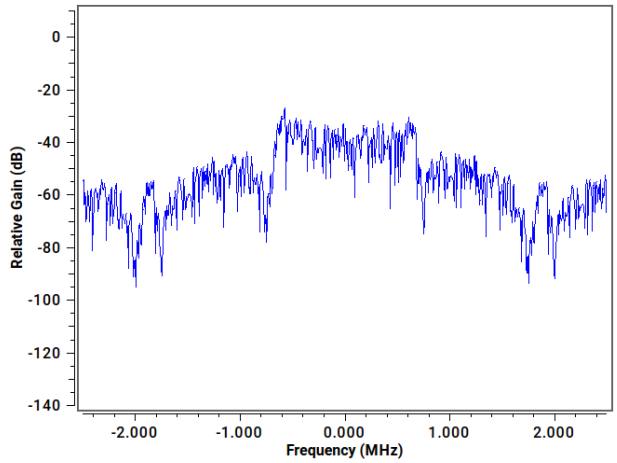

Figura 5.7: Espectro FSK en GNU Radio

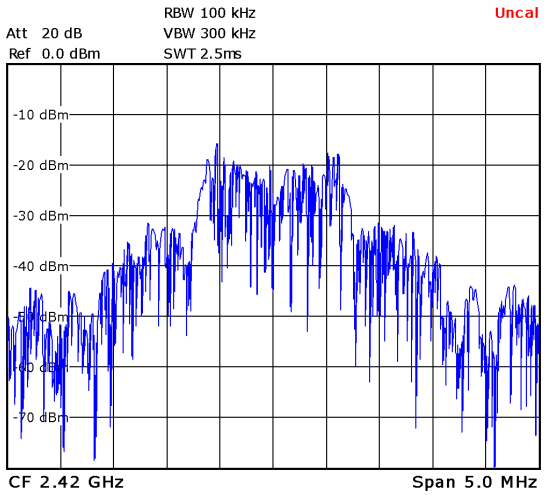

Figura 5.8: Espectro FSK en el analizador de espectros

Conclusiones

El conjunto del USRP N210 y la daughterboard RFX2400 muestra un nivel de salida máximo del orden de los 5 mW (7 dBm), no controlable desde UHD en GNU Radio. Según los datos del fabricante [8], esta daughterboard debería proporcionar una potencia de hasta 20 dBm.

Se ha observado durante la ejecución de los experimentos que el número de muestras por símbolo afecta en gran medida a la forma del espectro de la señal. En la modulación PSK se existe una relación inversa entre el número de Sps utilizadas y el ancho de banda de la modulación. Esto puede ser explicado por la mayor frecuencia con la que la señal cambia de fase, que puede inducir frecuencias más altas. En la señal FSK quedan más definidos las dos frecuencias que representan los símbolos. En la ASK, la señal portadora también se ve con mayor claridad.

Experimento 5.3: Desventajas del uso de un único USRP como sistema transceptor

Como se ha mencionado previamente, en las etapas iniciales del proyecto se ha hecho uso de un único USRP para la transmisión y recepción desde un mismo host. Este método se eligió por facilitar el control del sistema completo. Sin embargo, se dejó de utilizar por los siguientes motivos:

1. El aislamiento entre cadena de entrada y de salida no es lo suficientemente bueno para simular un sistema real.
2. Al haber mayor tráfico en el link host–USRP, aumenta el riesgo de overflow/underflow.

5.3.1 Descripción del experimento

Se ha testeado el aislamiento interno del USRP ejecutando un flowgraph que transmite un pulso a la frecuencia de trabajo y muestra en el GRC el espectro recibido por el mismo USRP. Para la ejecución de esta prueba se han conectado sendas cargas de 50 Ω a los puertos del USRP.

5.3.2 Resultados y conclusiones

Se ha observado que el conjunto USRP–RFX2400 proporciona un aislamiento de aproximadamente 25 dB en esta banda de frecuencias. Puede verse en la figura 5.9 que el pulso emitido por GNU Radio tiene un nivel de -40 dB. A este valor se le ha añadido el nivel de la señal original enviada al USRP, que es de -15 dB, obteniendo los 25 dB de aislamiento.

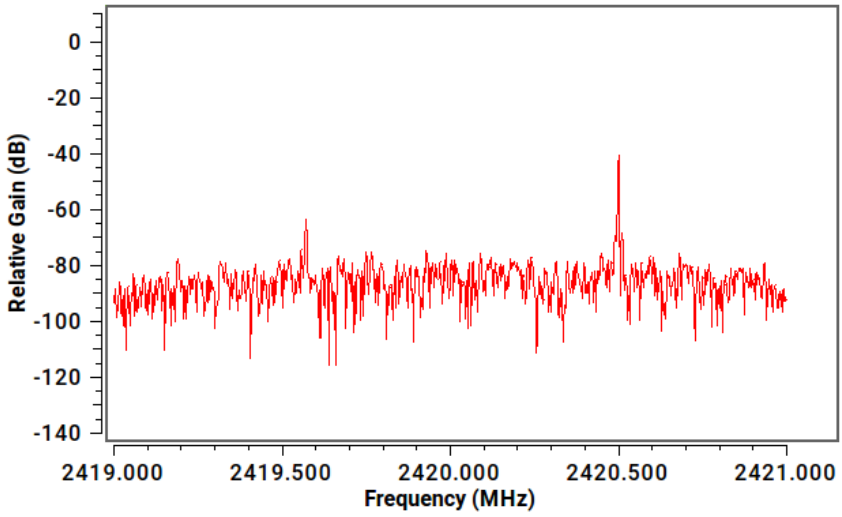

Figura 5.9: Aislamiento RF del USRP

Cabe destacar que la falta de un aislamiento completo (casi imposible de alcanzar) entre las cadenas de TX y RX puede ser aprovechada para tareas de calibración del conversor IQ. Los programas `uhd_cal_tx_iq_balance`, `uhd_cal_rx_iq_balance` y `uhd_cal_tx_dc_offset` hacen uso de los acoplamientos internos del USRP con este propósito.

Para facilitar el control de ambos hosts al utilizar la configuración transmisor-receptor, se ha hecho uso en algunos casos, además del acceso por SSH, del protocolo XML-RPC mediante el uso del bloque *XMLRPC Server* y un cliente Python. Este protocolo permite el control de la ejecución de flowgraphs en GNU Radio, así como del valor de las variables, de forma remota.

Experimento 5.4: Underrun en el transmisor

En vista de los problemas generados en el experimento 5.5 debido al underrun (ver anexo A.1) en el host transmisor[2], se van a perfilar las pérdidas de muestras en relación a la frecuencia de muestreo y el tamaño de muestras utilizadas. Para ello, se ejecutará repetidas veces un diagrama en GNU Radio cambiando la frecuencia de muestreo y el formato de muestra, durante periodos de un minuto. Al final de cada prueba se realizará un recuento de los flags "U" generados por UHD. Aunque esto no corresponde al número total de muestras perdidas, nos servirá para valorar las frecuencias de muestreo óptimas.

Las características del host utilizado para el sistema transmisor son: CPU: AMD A8-3530MX quad-core, 1900 MHz; Tarjeta de red: Realtek Semiconductor Co., Ltd. RTL8111/8168/8411 PCI Express Gigabit Ethernet Controller (rev 06); SO: Linux (OpenSUSE Tumbleweed, kernel 4.11.6-1-default #1 SMP PREEMPT).

5.4.1 Resultados y conclusiones

En la figura 5.10 se pueden observar los resultados de las pruebas realizadas. Ambos formatos (8 y 16 bits) tienen un comportamiento parecido. LLama la atención la reducción de las pérdidas entre 7 y 8 MSps. Para evitar los problemas generados por los underruns, que resultan en los datos no enviados y que pueden ser causantes de la corrupción de paquetes en el receptor, se utilizará la velocidad de muestreo de 1 MSps en los experimentos sensibles a fallos.

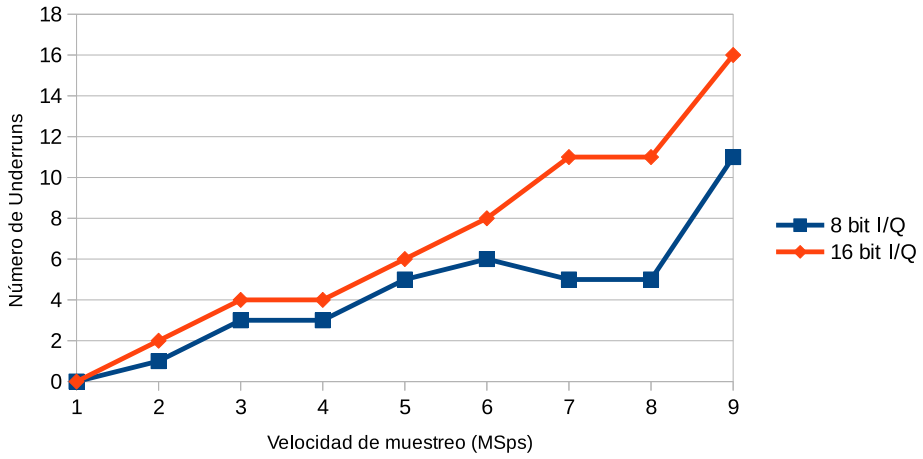

Figura 5.10: Relación entre underruns y frecuencia de muestreo, con muestras de 8 y 16 bits

Experimento 5.5: Validación del sistema completo

Una vez comprobado el correcto funcionamiento de la parte GNU Radio de los sistemas, se va a dar paso a las pruebas en el espacio libre. A lo largo de este experimento se consolida la consecución del objetivo primario del proyecto, la transmisión de ficheros entre hosts. Además, se hará un breve estudio del rendimiento que se consigue con cada modulación.

[2]No se han detectado problemas de overflow en el receptor, por lo que este no se ha perfilado.

5.5.1 Descripción del experimento

Se han separado los diagramas para la validación interna (figura 5.1) en dos, uno para el transmisor y otro para el receptor. Estas dos partes formarán dos aplicaciones GNU Radio separadas que serán ejecutadas en dos hosts diferentes. Cada host mantiene una conexión por red con uno de los USRP N210 disponibles, con las daughterboards RFX2400 montadas. Los terminales RF estarán conectados, bien a antenas, bien a cargas de 50 Ω. Puede verse una imagen del montaje en la figura 5.11.

Se realizarán pruebas variando la velocidad binaria de los sistemas, en las que se tratará de medir la calidad de las comunicaciones. Estas pruebas se realizarán para cada una de las modulaciones, utilizando para la transmisión dos antenas a 1 metro de distancia. Antes de realizar las pruebas, se optimizarán manualmente en el transmisor la ganancia del USRP y el umbral del decisor (si existe).

A falta de un método estable y eficiente para la medida del BER en directo[3], se utilizará como métrica para la calidad de la comunicación el ratio de paquetes recibidos con éxito. Se llevará la cuenta de los paquetes enviados y recibidos en los sistemas transmisor y receptor registrando el número de etiquetas[4] generadas. Cada paquete transportará una copia del fichero "articulo.txt"[5] completo. Este contenido de los paquetes recibidos se imprimirá por pantalla en el receptor para poder aportar una evaluación cualitativa de la calidad de la comunicación.

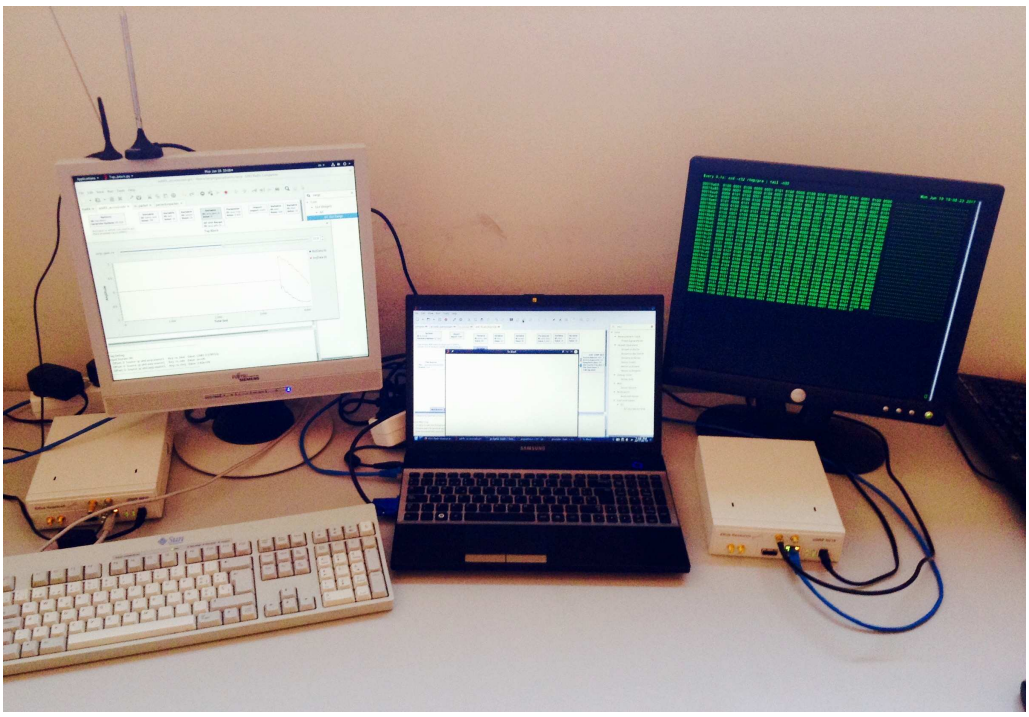

Figura 5.11: Sistema transmisor – receptor con dos USRP

En la figura 5.12 se muestra como ejemplo la pareja de diagramas utilizados para la modulación ASK.

Los resultados directos de las pruebas realizadas (número de paquetes enviados y recibidos) se van a presentar en tablas junto al valor de algunos parámetros. En orden, por columna, estos parámetros son: frecuencia de muestreo, número de muestras por símbolo, velocidad de símbolo, número de

[3] Se ha desarrollado un conjunto de programas y scripts para trabajar con la alineación y el bit error rate de ficheros capturados, pero se ha desechado el método por ser poco eficiente

[4] Las etiquetas (tags) generadas en los bloques para la sincronización de símbolo de la sección 4.4

[5] Artículo de prensa, 3149 bytes

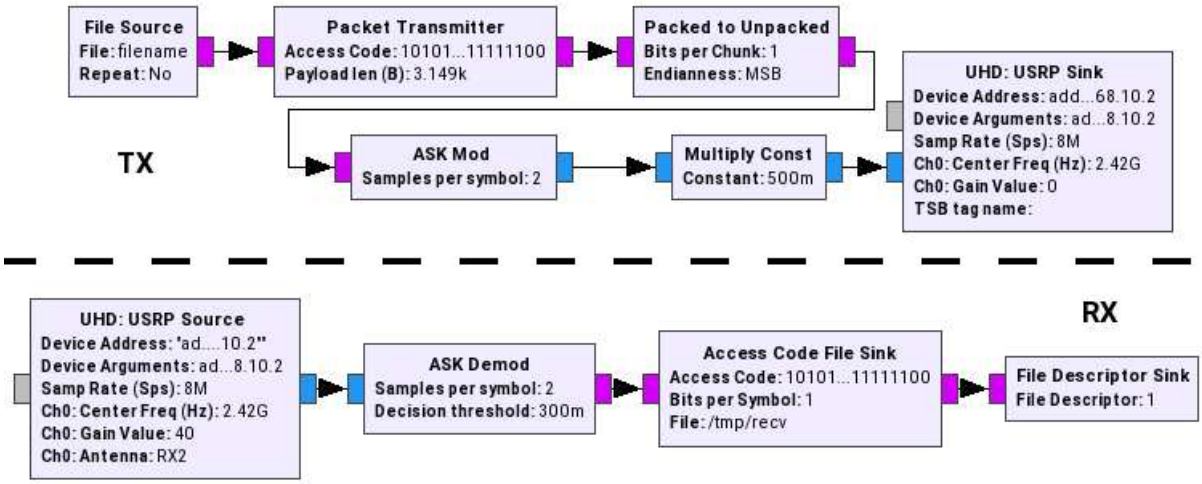

Figura 5.12: Ejemplo de diagrama utilizado para las pruebas transmisor – receptor

paquetes transmitidos, número de paquetes recibidos y porcentaje de éxito de las comunicaciones. Se han utilizado muestras de 16 bits. Las velocidades de muestreo han incurrido en underruns según los resultados del experimento 5.4.

5.5.2 Resultados

ASK

Se han hecho pruebas con el sistema ASK configurado para 2 muestras por símbolo y una frecuencia de muestreo desde 1 hasta 20 MSps. El contenido de los paquetes recibidos contenía pocos errores en relación al resto de modulaciones. Esto queda reflejado en los porcentajes de éxito de detección de paquetes, que dependen en última instancia de la integridad del código de acceso.

Los errores en el receptor se traducen en cambios de caracteres. Suelen estar aislados, de forma que el texto enviado sigue siendo legible. Puntualmente, aunque con mayor frecuencia según aumenta la frecuencia de muestreo, se pierde parte del contenido de un paquete. Esto puede ser debido a casos de underrun en el transmisor.

MSps	S/symb	kbps	TX	RX	%
1	2	500	1249	1149	92
2	2	1000	2264	2076	91,7
3	2	1500	4063	3604	88,7
4	2	2000	2765	2420	87,5
5	2	2500	6217	5567	89,5
6	2	3000	5838	4831	82,8
10	2	5000	8417	7605	90,4
20	2	10000	8862	6456	72,9

Cuadro 5.1: Resultados de pruebas de transmisión de ficheros con ASK

FSK

Para el sistema FSK se ha utilizado una desviación de frecuencia igual a un décimo de la frecuencia de muestreo. Los resultados de paquetes demodulados con éxito dejan a la modulación FSK en un lugar intermedio entre la ASK y la PSK. Los errores son de la misma naturaleza que en la ASK.

MSps	S/symb	kbps	TX	RX	%
1	2	500	898	776	86,4
2	2	1000	1691	1469	86,9
4	2	2000	5492	5049	91,9
8	2	4000	5307	4785	90,2
10	2	5000	5308	4585	86,4
20	2	1000	8248	5234	63,5

Cuadro 5.2: Resultados de pruebas de transmisión de ficheros con FSK

PSK

El sistema basado en la modulación PSK ha dado los peores resultados de las tres modulaciones. Aunque se han conseguido transmitir ficheros completos y sin errores, existe una probabilidad de error superior al 50 % incluso en el mejor de los casos testeados (20 muestras por símbolo a 1 MSps), en el que no existe apenas underrun en el transmisor[6]. El aumento de las muestras por símbolo no mejora las condiciones.

Los errores que han tenido lugar en la modulación PSK han sido de una naturaleza diferente a los de la ASK y la FSK. Normalmente, una parte del paquete (o fichero) se demodula correctamente, hasta que ocurre un error y se corrompe el resto de la información. La frecuencia de estos errores, incluso en el caso de 20 S/symb @ 1 MSps, elimina la opción de que sean causados únicamente por underruns del transmisor.

MSps	S/symb	kbps	TX	RX	%
1	20	50	281	177	63
2	10	200	333	173	51,9
2	8	250	789	494	62,6
2	4	500	1995	784	39,3
3	4	750	1194	355	29,7
4	4	1000	1502	974	64,8
4	2	2000	1886	554	29,4
5	2	2500	1662	944	56,8

Cuadro 5.3: Resultados de pruebas de transmisión de ficheros con PSK

Nota: debido al mal funcionamiento de la modulación PSK, estas pruebas se han realizado sobre cable coaxial. De esta forma se ha aumentado la relación SNR. Sin embargo, el rendimiento ha seguido siendo bajo (inferior al 60 % de paquetes recibidos).

[6] Aproximadamente 1 flag de underrun por cada 3 minutos

5.5.3 Conclusiones

En la gráfica 5.13 puede verse una comparación de las medidas tomadas para las tres modulaciones. En la implementación realizada, las modulaciones ASK y FSK han resultado ser las mejores en términos de throughput de información. Los resultados del experimento 5.6 coinciden en esto.

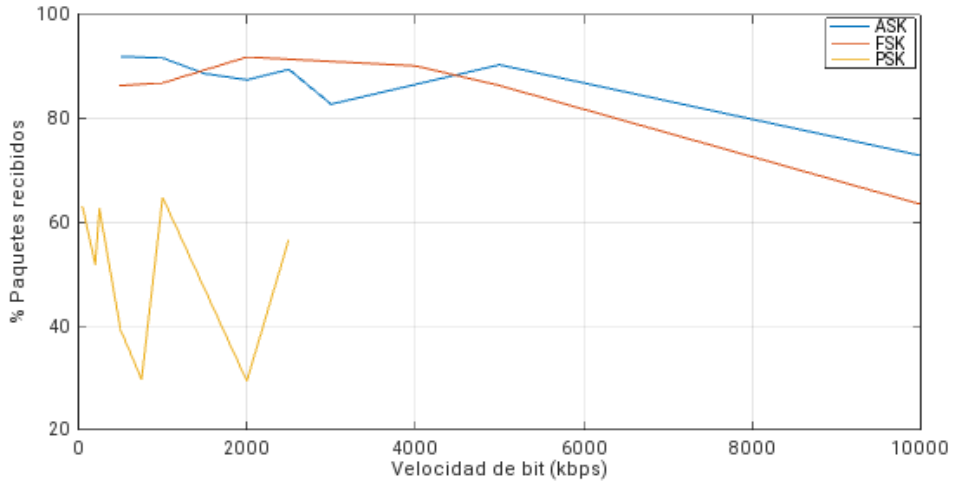

Figura 5.13: Comparación del rendimiento de las modulaciones implementadas

A parte de los resultados arriba mostrados, se ha conseguido transmitir un fichero con cada una de las tres modulaciones sin errores, uilizando 1 MSps y 2 muestras por símbolo, excepto para la PSK, en la que se han empleado 4 muestras por símbolo. Es posible la comunicación a mayores velocidades binarias, pero aumenta la probabilidad de error. Con esta prueba se ha cumplido el objetivo principal del proyecto.

Experimento 5.6: Capacidad de envío del sistema

Se va a tratar de encontrar el tamaño máximo de fichero que se puede enviar y recibir sin errores. Se determinará además el tamaño máximo de los paquetes[7] para la segmentación de la información a transmitir.

5.6.1 Descripción

En este experimento se han enviado ficheros encapsulados en paquetes (sin segmentar) entre los hosts a través de los USRP, aumentando su tamaño hasta que se han encontrado problemas para la comunicación, sea en el transmisor o en el receptor. En concreto, los ficheros enviados han sido extraídos de los primeros N bytes del fichero "quijote.txt"[8]. De este modo se ha encontrado el tamaño máximo de los paquetes soportado.

A continuación se han realizado pruebas con la segmentación de ficheros grandes en paquetes de tamaño mediano. Se han enviado varios ficheros, también aumentando su tamaño progresivamente, hasta que la comunicación ha sufrido fallos o han hecho falta demasiados intentos para llevarla a cabo. Para estas pruebas se han utilizado imágenes como ficheros a enviar, lo que facilita la detección de errores.

[7] Refiriéndonos a la encapsulación definida para habilitar la sincronización de símbolo con GNU Radio en 4.4
[8] Don Quijote de la Mancha en ASCII (304 kB)

5.6.2 Resultados y conclusiones

Siguiendo el método descrito se ha descubierto que el transmisor soporta hasta 32.000 bytes de datos en los paquetes, que coincide con la configuración para tamaño máximo de colas del kernel de Linux en el sistema transmisor[9]. Sin embargo, el receptor no es capaz de demodular paquetes de contenido superior a 26624 bits (26 kB). Cabe mencionar que el tamaño de la cabecera es de únicamente 12 bytes.

Se ha establecido un tamaño de 3200 bytes para los paquetes en la transmisión de ficheros segmentados. Se han obtenido resultados diferentes para cada modulación:

- ASK (2 S/symb @ 1 MSps): se han transmitido imágenes de hasta 1,5 MB con éxito. La siguiente prueba (5 MB) ha fracasado.
- FSK (4 S/symb @ 1 MSps): se ha recibido una imagen de 118 kB. Las pruebas con una imagen de 500 kB han resultado fallidas.
- PSK: los intentos con la imagen más pequeña (118 kB) han resultado en imágenes corruptas.

Al igual que en el experimento 5.5, los peores resultados se han conseguido con la modulación PSK. Sin embargo, las modulaciones ASK y FSK, que habían conseguido porcentajes de éxito parecidos, se han distanciado en esta prueba. Esto puede ser producido por la ligera ventaja que ofrecía la la ASK sobre la FSK a 1 MSps, que en transmisiones largas puede significar una gran diferencia.

[9] Comprobado ejecutando el comando `ipcs -l`

6

Conclusiones y Trabajo Futuro

UTILIZANDO tecnologías SDR basadas en host como las utilizadas en este proyecto es posible construir sistemas de comunicaciones de bajo coste y alta configurabilidad. Es de entender que se puedan desarrollar con los mismos métodos sistemas para ser integrados con tecnologías y sistemas ya existentes, como receptores GNSS o estaciones base móviles. Pueden implementarse incluso sistemas para la radioastronomía.

Con el sistema diseñado e implementado para este proyecto se han realizado comunicaciones unidireccionales con éxito. Se han diseñado moduladores y demoduladores definidos por software para las modulaciones ASK, PSK y FSK, con los que se han conseguido rendimientos dispares.

A lo largo de las pruebas realizadas se ha visto que la modulación que proporciona la mayor calidad a las comunicaciones es la ASK. Además, exhibe un espectro estrecho en comparación con la PSK y la FSK. No obstante, no es una modulación apropiada para las comunicaciones en entornos ruidosos o susceptibles a propagaciones multicamino y atenuaciones. Sería más conveniente el uso de la FSK en estos casos. La modulación PSK ha sido la que peores resultados ha dado.

Ha llamado la atención el contraste entre la alta capacidad del USRP N210 en términos de velocidad de muestras (25 MSps con muestras de 16 bits) y la baja capacidad de los hosts utilizados, que derivaba en pérdidas de muestras (sobre todo en el sistema transmisor). Aunque podría ser debido al hardware del ordenador transmisor, con una vida de 5 años a fecha de este proyecto, sufrir underruns a 2 MSps parece un poco excesivo.

Este proyecto puede ser extendido usando las mismas herramientas (GNU Radio – USRP N210) para implementar la capa física de un sistema de comunicaciones punto a punto completamente operativo. Para ello se proponen las siguientes mejoras:

- Resistencia al shadowing y multipath en el receptor. Inclusión de métodos de control de ganancia automática[1] (AGC).
- Habilitación de las comunicaciones bidireccionales, pudiendo alcanzar una comunicación full-duplex.
- Uso de un protocolo MAC para la detección y corrección de errores (FEC).
- Conexión upstream con el host para soportar comunicaciones TCP/IP. Puede conseguirse utilizando el bloque *TUNTAP PDU* en GNU Radio.

Posteriormente podrían realizarse investigaciones en el campo de la radio cognitiva, en métodos de selección inteligente de frecuencia, escáneres para detección de transmisiones o detección automática de modulaciones.

Tanto este proyecto de fin de grado como este documento se han desarrollado utilizando software de código abierto. Memoria finalizada el 27 de junio del 2017.

[1] Aunque GNU Radio proporciona bloques para la AGC digital, lo óptimo sería aprovechar la ganancia de recepción que proporciona la daughterboard del USRP, controlable a través del bloque *UHD Sink*

Glosario de acrónimos

- **ADC**: Analog to Digital Converter
- **AGC**: Automatic Gain Control
- **ALSA**: Advanced Linux Sound Architecture
- **AM**: Amplitude Modulation
- **API**: Application User Interface
- **APSK**: Amplitude-Phase Shift Keying
- **ASIC**: Application Specific Integrated Chip
- **ASK**: Amplitude Shift Keying
- **BER**: Bit Error Rate
- **BPSK**: Binary Phase Shift Keying
- **CGRAN**: The Comprehensive GNU Radio Archive Network
- **CPU**: Central Processing Unit
- **DAC**: Digital to Analog Converter
- **DBL**: Doble Banda Lateral
- **DC**: Direct Current, o corriente continua
- **DDC**: Digital Down Converter
- **DSP**: Digital Signal Processor
- **DTMF**: Dual-tone Multi-frequency
- **DUC**: Digital Up Converter
- **DVB-T**: Digital Video Broadcasting - Terrestrial
- **FEC**: Forward Error Correction
- **FPGA**: Field Programmable Gate Array
- **FSK**: Frequency Shift Keying
- **GNSS**: Global Navigation Satellite System
- **GNU**: GNU's Not Unix!
- **GPP**: General Purpose Processor
- **GPSDO**: GPS Disciplined Oscillator
- **GPS**: Global Positioning System
- **GRC**: GNU Radio Companion, interfaz gráfica de GNU Radio
- **GUI**: Graphical User Interface, o interfaz gráfica
- **I/Q**: Hardware
- **IEEE**: Institute of Electrical and Electronics Engineers
- **IF**: Intermediate Frequency, frecuencia intermedia
- **ISM**: Industrial, Scientific and Medical radio bands
- **ITU**: International Telecommunication Union
- **IoT**: Internet of Things
- **LNA**: Low Noise Amplifier
- **LabVIEW**: Laboratory Virtual Instrument Engineering
- **MAC**: Media Access Control
- **MIT**: Massachusets Institute of Technology
- **MSK**: Minimal Shift Keying
- **MTU**: Maximum Transmission Unit

- **NRZ**: Not Return to Zero
- **OFDM**: Orthogonal Frequency-Division Modulation
- **OpenBTS**: Open Base Transceiver Station
- **Osmocom**: Open source mobile communications
- **PC**: Personal Computer
- **PDU**: Protocol Data Unit
- **PLL**: Phase Locked Loop
- **PLL**: Phase-Lock Circuit
- **PSK**: Phase Shift Keying
- **QAM**: Quadrature Amplitude Modulation
- **RFIC**: Radio Frequency Integrated Chip
- **RFNoC**: RF Network on Chip
- **RF**: Radio Frequency, referido al campo de las radiofrecuencias
- **RTL-SDR**: Pequeño dispositivo SDR de bajo sin capacidad de transmitir datos
- **SDR**: Software Defined Radio
- **SNR**: Signal to Noise Ratio
- **SO**: Sistema Operativo
- **USRP**: Universal Software Defined Peripheral

Bibliografía

[1] Página web principal de GNU Radio. https://www.gnuradio.org/.

[2] Nate Temple Neel Pandeya. *About USRP Bandwidths and Sampling Rates*. Ettus Research, mayo 2016. https://kb.ettus.com/About_USRP_Bandwidths_and_Sampling_Rates.

[3] John G. Proakis. *Digital Communications*, chapter 1, page 22. Mc Graw Hill, 2 edition, agosto 2001.

[4] Osmocom. RTL-SDR reference wiki. http://osmocom.org/projects/sdr/wiki/rtl-sdr.

[5] Silvian Spiridon. *Toward 5G Software Defined Radio Receiver Front-Ends*. Springer Briefs in Electrical and Computer Engineering. Springer, 2016.

[6] ITU. Radio regulations of the international telecommunication union, diciembre 1992. Article 5 – Frequency allocations, Section IV – Table of Frequency Allocations.

[7] Tutoriales GNU Radio, marzo 2017. https://wiki.gnuradio.org/index.php/Guided_Tutorials.

[8] Ettus Research. Transceiver Daughterboards for the USRP Software Radio System. Disponible online. http://www.olifantasia.com/gnuradio/usrp/files/datasheets/ds_transceiver.pdf.

[9] Software Defined Radio: Past, Present and Future. Technical report, National Instruments, mayo 2017.

[10] Tom Rondeau. Scheduler details. septiembre 2013. Disponible online en: https://static.squarespace.com/static/543ae9afe4b0c3b808d72acd/543aee1fe4b09162d0863397/543aee20e4b09162d0863578/1380223973117/gr_scheduler_overview.pdf.

[11] Ettus Research. *Application note: examples provided with the UHD hardware driver*. Disponible online en: https://www.ettus.com/content/files/kb/application_note_uhd_examples.pdf.

[12] Ettus Research. UHD development manual. Doxygen documentation. http://files.ettus.com/manual/page_uhd.html.

[13] Ettus Research. *UHD and USRP Manual: General*. http://files.ettus.com/manual/page_general.html.

[14] UHD and USRP manual: RFNoC, abril 2017. https://kb.ettus.com/RFNoC.

[15] Realtek. RTL2832U product description. http://www.realtek.com.tw/products/productsView.aspx?Langid=1&PFid=35&Level=4&Conn=3&ProdID=257.

[16] RTL-SDR history. http://rtlsdr.org/#history_and_discovery_of_rtlsdr.

[17] An open source global navigation satellite systems software-defined receiver. `http://gnss-sdr.org/`.

[18] Andrea Goldsmith. *Wireless Communications*. Cambridge University Press, 2005.

[19] David M. Pozar. *Microwave and RF design of Wireless Systems*. John Wiley & Sons, Inc., 2001.

[20] Ettus Research. USRP application notes: selecting a daughterboard. Disponible online. `https://www.ettus.com/content/files/kb/Selecting_an_RF_Daughterboard.pdf`.

[21] David M. Pozar. *Microwave and RF design of Wireless Systems*, chapter 9, page 306. John Wiley & Sons, Inc., 2001.

[22] Tim O'Shea. Eficient processing of bursty information streams with GNU Radio. septiembre 2006. `http://static1.1.sqspcdn.com/static/f/679473/25468326/1411400880130/Sep18_06_Oshea_+BURSTY.pdf?token=OezNy0Q3fL4t1UWZeEqK7cu7aQ4%3D`.

[23] *GNU Radio and C++ API Reference: Stream Tags*, agosto 2016. `https://gnuradio.org/doc/doxygen/page_stream_tags.html`.

[24] *GNU Radio and C++ API Reference: Message Passing*, agosto 2016. `https://gnuradio.org/doc/doxygen/page_msg_passing.html`.

[25] How to achieve Gigabit speeds with Linux. `http://datatag.web.cern.ch/datatag/howto/tcp.html`.

[26] Ettus Research, Mountain View, CA 94043. *USRP N210 Datasheet*, septiembre 2012. `https://www.ettus.com/content/files/07495_Ettus_N200-210_DS_Flyer_HR_1.pdf`.

A

Notas USRP N210

Experimento A.1: Overflow y underrun

UHD define varios flags que se presentan al usuario en el caso de que haya algún problema en las comunicaciones entre el USRP y el host. Los flags que aplican para el N210 (y el resto de USRPs con interfaz red) son los siguientes:

- **D (overflow)**: el USRP genera muestras continuamente. Cuando el host no es capaz de procesarlas al mismo ritmo, el búfer de entrada del socket se llena y se descartan muestras. En este caso, se imprime el caracter D por la salida estándar del sistema.
- **U (underrun)**: al transmitir, el USRP requiere también un flujo continuo de muestras. En el caso en el que el host sea incapaz de generarlas a tiempo, el USRP lo señaliza y UHD imprime el caracter U.

Experimento A.2: Optimización del rendimiento en el host

A.2.1 Redimensionamiento de búferes

Existe un tamaño máximo alocable para los búferes al crear un socket, definido por el sistema operativo. Como se acaba de mencionar en A.1, su tamaño juega un papel en el rendimiento. Aunque su sobredimensionamiento puede perjudicar la velocidad de transmisión, es posible que aumentarlos ligeramente sea útil para evitar a UHD ciertos trabajos, como gestionar su propio búfer para la transmisión. También se puede aumentar el tamaño para el búfer de recepción reduciendo así el riesgo de pérdida de paquetes, aunque si la aplicación no es capaz de procesar las muestras a tiempo no nos supodrá ninguna mejora. Para redimensionar el tamaño máximo de los búferes en Linux, se ejecutan los siguientes comandos:

```
sysctl -w net.core.rmem\_max=<new value>
sysctl -w net.core.wmem\_max=<new value>
```

El tamaño óptimo para los búferes viene dado por la expresión: tamaño = 2 * ancho_de_banda * latencia [25]

A.2.2 Número de descriptores de la interfaz

En algunos casos se puede aumentar dramáticamente el rendimiento cambiando el número de descriptores de la interfaz de red:

```
ethtool -G <interface> tx <N> rx <N>
```

A.2.3 Prioridad de hilos

UHD hace uso de diferentes hilos para realizar varias tareas de forma concurrente. Es frecuente que al crear un nuevo hilo intente aumentar su nivel de prioridad para mejorar su rendimiento, lo que genera un warning en la mayoría de sistemas por la falta de privilegios apropiados. Para permitir a UHD manejar la prioridad de sus hilos se añade la siguiente línea al fichero /etc/security/limits.conf:

```
@GROUP   --       rtprio   99
```

El usuario con el que se ejecute el programa que utilice UHD debe pertenecer al grupo GROUP.

A.2.4 MTU

La comunicación por red entre UHD, ejecutado en el host, y el USRP se realiza con un protocolo encapsulado en paquetes TCP. Los dispositivos tardan en llenar un paquete de muestras un tiempo inversamente proporcional a la frecuencia de muestreo, y luego lo envían. Reduciendo el tamaño de los paquetes se reduce este tiempo y, en consecuencia la latencia.

A.2.5 Coalesencia de la interfaz de red

En las aplicaciones de baja latencia es útil modificar la coalesencia de la interfaz de red (si esta la soporta). De esta manera podemos variar los tiempos entre las interrupciones generadas por la interfaz de red para notificar al kernel la llegada de nuevos paquetes. Una mayor frecuencia de interrupciones reduce la latencia, pero también el throughput total del sistema, lo que supone una menor capacidad de procesamiento de muestras.

En el caso en el que la interfaz lo soporte, en Linux se puede modificar la coalesencia cambiando alguno de los siguientes parámetros con ethtool:

```
ethtool -C|--coalesce devname [adaptive-rx on|off] [adaptive-tx on
   |off] [rx-usecs N] [rx-frames N] [tx-usecs N] [tx-frames N]
```

Experimento A.3: Hoja de datos

Las figuras de este apéndice han sido directamente extraídas de la hoja de datos del USRP N210 [26], disponible en el sitio web de Ettus.

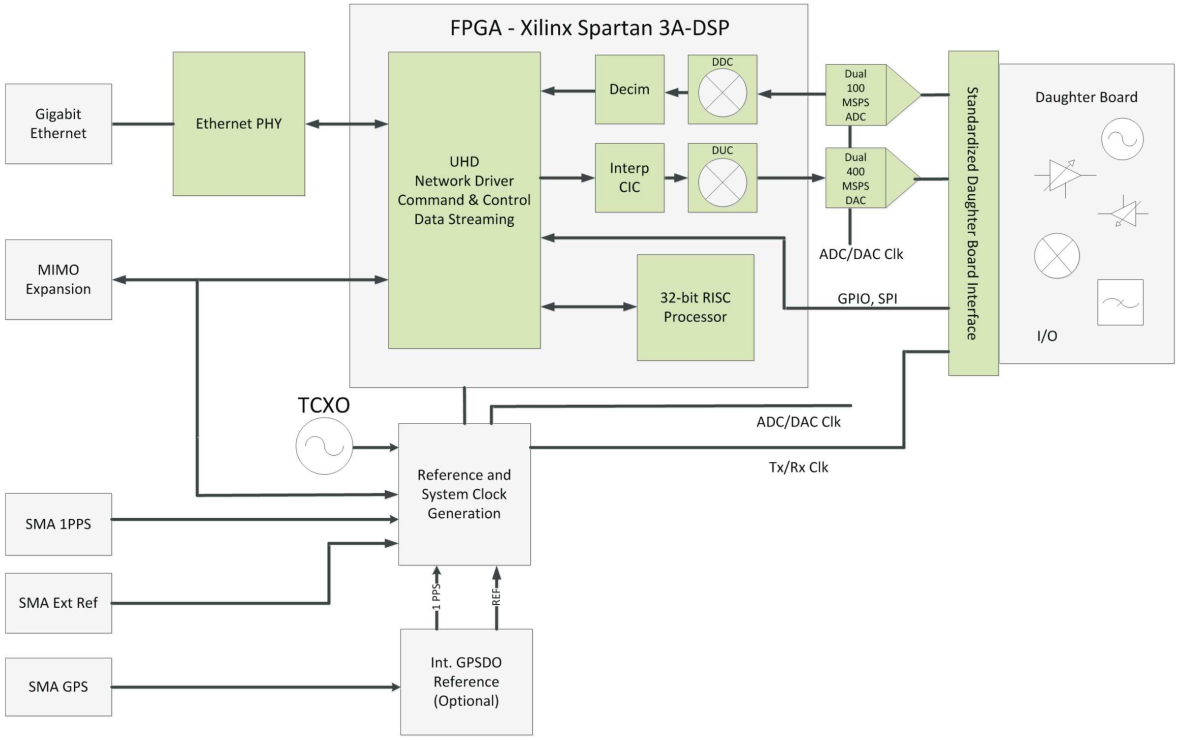

Figura A.1: Diagrama de bloques del USRP N210

SPECIFICATIONS

Spec	Typ.	Unit
POWER		
DC Input	6	V
Current Consumption	1.3	A
w/ WBX Daughterboard	2.3	A
CONVERSION PERFORMANCE AND CLOCKS		
ADC Sample Rate	100	MS/s
ADC Resolution	14	bits
ADC Wideband SFDR	88	dBc
DAC Sample Rate	400	MS/s
DAC Resolution	16	bits
DAC Wideband SFDR	80	dBc
Host Sample Rate (8b/16b)	50/25	MS/s
Frequency Accuracy	2.5	ppm
w/ GPSDO Reference	0.01	ppm

Spec	Typ.	Unit
RF PERFORMANCE (W/ WBX)		
SSB/LO Suppression	35/50	dBc
Phase Noise (1.8 Ghz)		
10 kHz	-80	dBc/Hz
100 kHz	-100	dBc/Hz
1 MHz	-137	dBc/Hz
Power Output	15	dBm
IIP3	0	dBm
Receive Noise Figure	5	dB
PHYSICAL		
Operating Temperature	0 to 55°	C
Dimensions (l x w x h)	22x16x5	cm
Weight	1.2	kg

* All specifications are subject to change without notice.

Figura A.2: Especificaciones del USRP N210