Facultad de física

Universidad de Sevilla

Trabajo de Fin de Grado (Grado en Física)

Digitalizadores basados en radio cognitiva para aplicaciones IoT/5G: Etapa transmisora

Autor: Raúl Varela Ferrando

Tutores: Luis Alejandro Camuñas Mesa y Jose Manuel de la Rosa Utrera

17/11/2021

Agradecimientos

En agradecimiento a mis padres, a mi familia y a todo aquél que estuvo ahí cuando faltaban las fuerzas y ahogaba la presión. Especial mención a mis tutores, Luis Alejandro Camuñas Mesa y Jose Manuel de la Rosa Utrera, por ayudarme a subir este último escalón.

Mi presente es vuestro mérito, y mejorarlo mi responsabilidad.

Índice

1. Introducción

A día de hoy, uno de los problemas más significativos asociado a los sistemas de comunicaciones inalámbricos es el deficiente aprovechamiento del espectro electromagnético debido a las políticas de asignación fija de las bandas de frecuencia, lo que nos lleva a tener ciertas bandas totalmente congestinadas y muchas otras parcial o totalmente inutilizadas la mayor parte del tiempo. Sumando a esto la gran demanda que sufre el sector en estos últimos años debido a la digitalización de nuestro entorno, y el éxito de los servicios de las bandas de acceso libre, se ha motivado a desarrollar novedosas tecnologías que posibilitan el uso del espectro de forma inteligente, coordinada y oportunista sin perjudicar a los servicios restantes.

La radio cognitiva (RC) es una tecnología basada en cambiar la forma en que el espectro es utilizado actualmente, a la vez que incrementa su disponibilidad para otros sistemas de comunicaciones inalámbricos. Podemos interpretarla de distintas maneras, en un inicio se pensó como una ampliación del Radio Definido por Software (SDR), aunque después observaron que podía basarse en aplicaciones del SDR. Estas interpretaciones no son erróneas ya que partiendo del SDR podemos evolucionarla a una Radio Cognitiva, y por tanto basarse en sus aplicaciones. Esta idea se hizo pública oficialmente por primera vez en el artículo Joseph Mitola III y Gerald Q. Maguire, Jr. donde la definieron como "el punto en el cual las PDAs inalámbricas y las redes relacionadas son, en términos computacionales, lo suficientemente inteligentes con respecto a los recursos de radio y las correspondientes comunicaciones de terminal a terminal como para detectar las necesidades eventuales de comunicación del ususario como una función del contexto de uso y proporcionarle los recursos de radio y servicios inalámbricos más adecuados a sus necesidades", donde el concepto PDA hace referencia a un asistente digital personal, es decir, teléfonos móviles u ordenadores personales por ejemplo.

Como hemos mencionado anteriormente, una forma de implementar este paradigma sería a través del concepto SDR. Este concepto se resume en un sistema de radiocomunicaciones donde los componentes que antes eran normalmente implementados en hardware, como filtros, moduladores/demoduladores y demás; son ahora implementados vía software a través de computadoras u otros dispositivos.

Dentro de la SDR tenemos que al utilizar procesadores no especializados para gran parte del procesamiento de la señal, valga la redundancia, en lugar de un hardware de propósito específico, podemos cambiar los protocolos de transmisión y emisión de nuestras señales en tiempo real a través de los parámetros definidos en nuestro software.

Un ejemplo de sistema de SDR serían las placas ADALM-PLUTO del fabricante Analog Devices, ilustradas en la **Fig. 1**, con las cuáles se realizarán los experimentos del proyecto.

Figura 1: Placa ADALM-PLUTO de SDR.

Usando dos de estas placas ADALM-PLUTO, Javier Soler Medina y yo en colaboración, implementaremos un sistema de comunicaciones inalámbrico y demostraremos la capacidad de ajustar los parámetros de transmisión y recepción usando un esquema de modulación QPSK, el cual describiremos en detalle a continuación.

2. Fundamentos teóricos

2.1. Modulación de una onda

La modulación engloba todo el conjunto de procesos utilizados con el fin de introducir información en una onda portadora, lo que nos permite poder transmitir más información simultáneamente a la vez que nos protegemos contra posibles ruidos e interferencias. En resumen, consiste en hacer que un parámetro de la onda portadora varíe de acuerdo a las propiedades de la señal moduladora, siendo esta última la que contiene la información que queremos transmitir, obteniendo como resultado una onda modulada que es la que transmitimos.

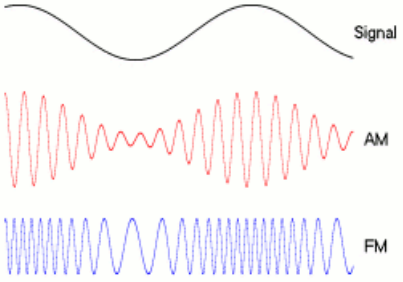


Figura 2: Ejemplos de modulacion en amplitud (AM) y en frecuencia (FM).

Dentro del concepto de modulacion, podemos encontrar varios tipos, como ya hemos podido intuir a través de la **Fig. 2**. A continuación enumeraremos algunos de estos tipos:

1. **Modulación en amplitud:** Consiste en cambiar la amplitud de la onda portadora en función

de las variaciones de la señal moduladora, como se muestra en la **Fig. 2**.

2. **Modulación en frecuencia:** Consiste en cambiar la frecuencia de la onda portadora en función de las variaciones de la señal moduladora, como también se muestra en la **Fig. 2**.

3. **Modulación en fase:** Es un tipo de modulación que se caracteriza porque la variación de la fase de la onda portadora es directamente proporcional a la señal moduladora.

Dentro de estos tipos, nos centraremos en un subtipo de modulación en fase llamado **modulación por desplazamiento de fase o PSK**. Esta es una forma de modulacion angular que consiste en hacer variar la fase de la portadora entre un número finito de valores. La gran diferencia de este tipo de modulación comparada con la modulación en fase convencional es que en este caso la señal moduladora es digital, por lo que el número de estados es finito.

Este tipo de modulación se resume en que la fase de la señal portadora representa cada símbolo de información de la señal moduladora, cuyo valor angular se elige entre un conjunto discreto de N valores posibes con la ayuda del modulador. Dado que estamos tratando señales digitales, el número de fases a tomar será potencia de dos ya que se suelen utilizar números enteros de bits. Partiendo de dos fases nos vamos encontrado los distintos tipos de modulación PSK, con dos fases sería la BPSK (Binary Phase-Shift Keying), con cuatro fases sería la QPSK (Quadrature Phase-Shift Keying), y así sucesivamente. Nosotros en este caso nos centraremos en la modulación QPSK.

En este tipo de modulación tenemos que las cuatro fases equiespaciadas representan los dígitos 00, 01, 10 y 11, transmitiendo dos bits en cada fase. Este tipo de modulación nos divide la señal en dos BPSK, una en fase (I) y otra en cuadratura (Q).

$$V_{QPSK}(t) = I \cdot cos(\omega t) + Q \cdot sin(\omega t) \tag{1}$$

Donde si los datos no cambian de un periodo al siguiente, la fase de la portadora permanece inalterada. Mientras que en el caso de que cambie un bit, la portadora es desfasada 90^o, y si cambian ambos bits es desfasada 180^o.

A través del gráfico de constelaciones (**Fig. 3**) podemos visualizar fácilmente los diferentes estados posibles, clasificando cada bit de la señal en uno de ellos, dependiendo del offset en fase asignado.

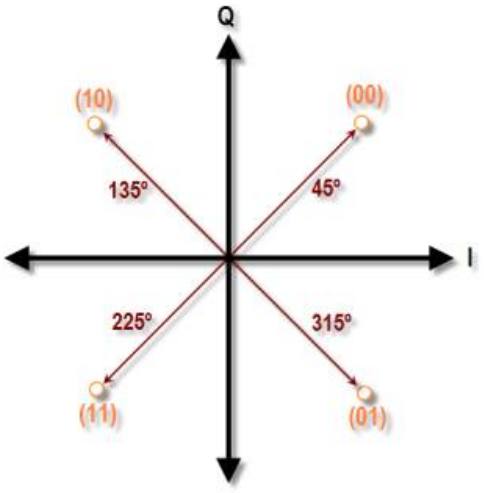


Figura 3: Gráfico de contelación para la modulación QPSK con desfase de $\pi/4$.

2.2. Sistema SDR (ADALM-PLUTO)

Como hemos dicho antes, un aparato SDR básico puede estar conformado por un conversor analógico-digital seguido de algún tipo de adaptador de radiofrecuencia (RF) y capacitado para cambiar en tiempo real los parámetros de transmisión y recepción de datos. En la **Fig. 4** tenemos el esquema de un equipo SDR básico, donde se pueden diferenciar tres secciones elementales: RF, FI y Banda-Base. Cada bloque cumplirá una función u otra dependiendo de si nuestro sistema se trata de un emisor o un receptor, siendo el emisor representado por la flecha que se dirige hacia la antena y el receptor por la flecha en sentido contrario.

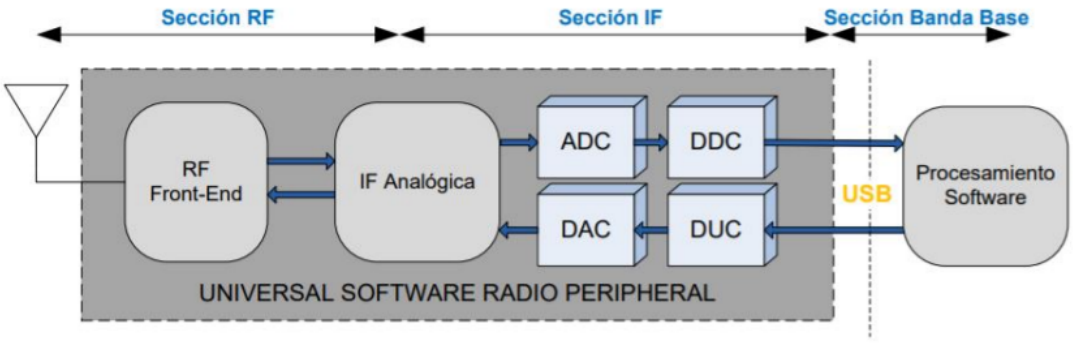


Figura 4: Esquema de un sistema SDR básico. Tomada de *Implementación de Software Defined Radio en sistemas de comunicaciones actuales*, Gutiérrez A. 2020, `https://idus.us.es/bitstream/handle/11441/109235/TFG-3378-GUTIERREZ\%20RIVERA.pdf?sequence=1&isAllowed=y`

La sección Banda-Base contiene un procesador que se encarga de tratar/extraer nuestra información, pudiendo ser una o varias señales compuestas permitiendo la transmisión/recepción simultánea de varios modos.

La sección FI es en la que se sitúa el filtrado, modulación/demodulación y la conversión analógico-digital o digital-analógico.

Por último, la sección RF en la transmisión es la responsable de convertir la señal a nuestra frecuencia de envío, para después amplificarla y aplicarla a la antena, mientras que en la recepción primero se amplifica para después convertirla a la frecuencia intermedia, aunque en algunas aplicaciones esto último no es necesario.

Como hemos mencionado anteriormente, el sistema SDR que utilizaremos en este proyecto es ADALM-PLUTO, ilustrado en la **Fig. 1**. Este sistema del fabricante Analog Devices es un transceptor, es decir, está formado por un transmisor y un receptor, y sus especificaciones se muestran a continuación:

- **Cobertura RF**: 325MHz - 3.8GHz
- **Ancho de banda**: 20MHz
- **Convertidores ADC/DCA**: 12 bits
- **Interfaz USB**: USB 2.0
- **Frecuencia de muestreo**: 61.44 Mbps
- **Velocidad de muestreo**: 65.1kSps - 61.44 MSps
- **Salida de potencia Tx**: 7dBm
- **Figura de Ruido Rx**: <3.5dBm
- **Precisión de Modulación de Rx y Tx**: -34dB
- **Núcleo**: ARM® Cortex-A9™ MPCore de 667MHz
- **Celdas lógicas FPGA**: 28000
- **Módulos DSP**: 80
- **DDR3L**: 4GB (512MB)
- **QSPI Flash**: 256MB (32MB)

El diagrama de bloques asociado a este sistema SDR es el siguiente:

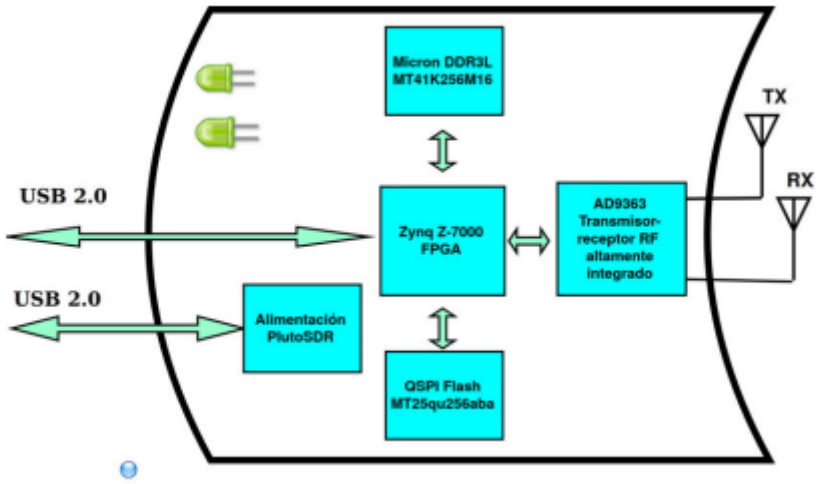


Figura 5: Diagrama de bloques del ADALM-PLUTO. Tomado de *Diseño de una estación remota de comunicaciones basada en radio software*, Herrador G. B., 2018, `https://oa.upm.es/67406/1/TFG_GUILLERMO_BARTOLOME_HERRADOR.pdf`

Como se puede observar, el Pluto SDR es un sistema compuesto principalmente por dos componentes hardware:

- Analog Devices AD9363 Highly Integrated RF Agile Transreceiver.
- Rlinx® Zynq Z-7000 FPGA.

El AD9363 es un transceptor de RF integrado de alto rendimiento empaquetado en un formato de 10mm x 10mm cuya capacidad de programación y ancho de banda lo hace muy últil para un amplio abanico de aplicaciones en radiofrecuencia. En este dispositivo se combinan una parte frontal de radiofrecuencia con una sección bandabase de señal mixta flexible, además de sintetizadores de frecuencia integrados. Este transmisor-receptor opera en la rango de frencuencias desde 325MHz a 3.8GHz, como señalan sus especificaciones, cubriendo así la mayoría de las bandas con y sin licencia, además de aportar un ancho instantáneo desde 200KHz hasta 20MHz. Incluye dos receptores independientes de conversión directa, cada uno de ellos incluye control de ganancia automático independiente o "Automatic Gain Control" (AGC), corrección de desplazamiento de CC, corrección de cuadratura y filtrado digital, aunque también dispone de modos de ganancia manual flexibles, dos ADC de alto rango por canal.

Los filtros de diezmado y el filtro FIR son los encargados de recibir las señales I/Q y producir una señal de salida de 12 bits. La arquitectura de conversión directa de los transmisores logra una alta precisión en la modulación. Podemos utilizar como detector de potencia el monitor de potencia de

tranmisión incorporado, permitiendo así una potencia de transmisión muy precisa. El PLL está totalmente integrado y proporciona una baja potencia en la síntesis de frencuencia para recibir y transmitir canales. Los osciladores (VCO) son los encargados de controlar el voltaje. El núcleo del AD9363 se puede alimentar desde 1,3V directamente, controlandolo a través de un puerto serie estándar de 4 hilos y 4 pines de control de entrada-salida en tiempo real.

Por otra lado, la familia Zynq-7000 nos ofrece la frexibilidad y la escalabilidad de una FPGA que porporciona potencia, rendimiento y facilidad de uso. Esta gama de dispositivos permite a los diseñadores apuntar aplicaciones sensibles al coste y de alto rendimiento desde una única plataforma. Pueden servir en una amplia gama de aplicaciones como:

- Asistencia al conductor en automoción.
- IP y cámara inteligente.
- LTE (Long Term Evolution).

La inclusión de un procesador de aplicaciones permite el soporte de sistemas operativos de alto nivel, por ejemplo GNU.

A continuación se muestran las tablas que contienen los parámetros más relevantes de ambos componentes hardware:

Parámetro	**Unidades**	**Estándar**
Pérdida de retorno de la entrada	dB	-10
Figura de ruido	dB	2.5
IP3 LNA	dBm	-18
IP2	dBm	40

Tabla 1: Parámetros en recepción del AD9363 HIghly Integrated RF Agile Transreceiver.

Parámetro	**Unidades**	**Estándar**
Pérdida de retorno de la salida	dB	-10
Potencia de salida máxima	dBm	2.5
OIP3 LNA	dBm	23

Tabla 2: Parámetros en transmisión del AD9363 Highly Integrated RF Agile Transreceiver.

Nombre del dispositivo	Z-7010
Número de pieza	XC7Z010
Núcleo del procesador	Dual-Core ARM Cortex-A9 MPCore con CoreSight
Extensiones del procesador	NEONTM Punto flotante de precisión simple o doble para cada procesador
Frecuencia del SoC	667MHz 766MHz 866MHz
Caché L1	Instrucción 32KB
Cache L2	512KB
Memoria On-Chip	256KB
Memoria externa de apoyo	DDR3, DDR3L, DDR2, LPDDR2
Memoria estática externa de apoyo	2x Quad-SPI, NAND, NOR
Canales DMA (Direct Memory Access)	8 (4 dedicados a la lógica programable)
Periféricos	2x UART, 2x CAN 2.bB, 2x I2C, 2x SPI, 4x 32bits GPIO

Tabla 3: Processing System (Zynq Z-7000 FPGA).

Nombre del dispositivo	Z-7010
Número de pieza	XC7Z010
Celdas lógicas programables	28000
LUTS	17600
Flip-Flops	35200
Bloques de memoria RAM (Bloques de 36KB)	2.1 MB (60)
Cache L2	512KB
Bloques DSP	80

Tabla 4: Programmable Logic (Zynq Z-7000 FPGA).

2.3. Transmisores. Tipos de transmisores.

Dentro de los transmisores tenemos varios tipos dependiendo de su arquitectura y topología, en este apartado aprenderemos un poco sobre ellos. Aunque a día de hoy se siguen utilizando transmisores puramente analógicos, los cuáles implementan modulaciones tipo AM o FM, en los sistemas de comunicaciones inalámbricos se utilizan otras tecnologías, en particular, transmisores de conversión directa o superheterodinos, los cuales explicaremos a continuación.

2.3.1. Transmisor de banda lateral única ("Single-Sideband")

La modulación AM resulta en dos imágenes espejo de nuestra banda, superior e inferior, como se muestra en la **Fig. 6**, las cuales son transmitidas en conjunto. La banda superior resulta de la suma de la frecuencia de la señal portadora (f_c) y la frecuencia de la señal moduladora (f_m), mientras que la inferior resulta de la resta de la frecuencia de la señal portadora y la frecuencia de la señal moduladora.

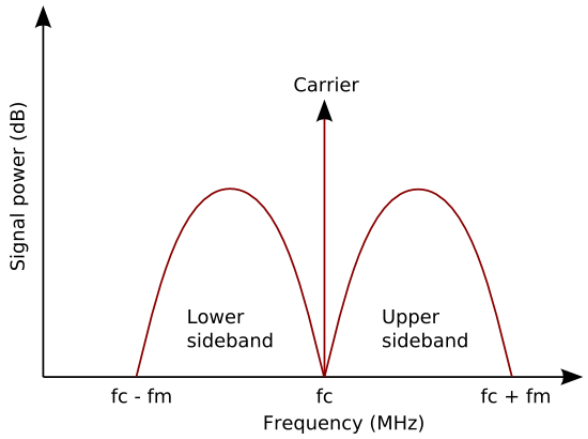


Figura 6: Espectro para la modulación AM. Tomada de *Wikipedia*, (s.f.), `https://upload.wikimedia.org/wikipedia/commons/f/fd/Am-sidebands.png`

Un transmisor de banda lateral única (SSB) difiere de un transmisor AM básicamente en que sólo transmite una de las bandas anteriormente mencionadas, la superior o la inferior, pero no ambas, por lo que este tipo de transmisor utiliza un menor ancho de banda que un transmisor AM convencional.

En la **Fig. 7** se muestra la implementación de un transmisor SSB. Un oscilador genera nuestra señal portadora, la cuál amplificamos antes de introducirla en nuestro modulador. Paralelamente, nuestra señal moduladora ("Audio signal") también es amplificada y procesada antes del modulador.

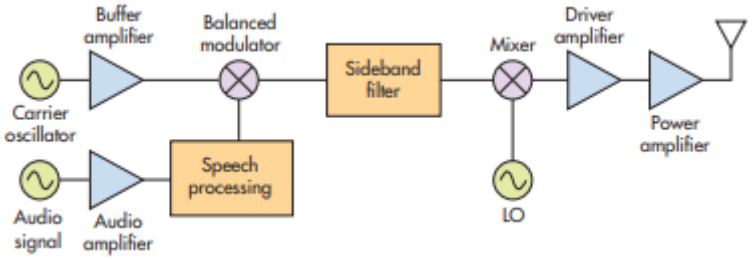


Figura 7: Esquema de un transmisor SSB. Tomada de *The Differences Between Transmitter Types*, Demartino C. (s.f.), `https://www.mwrf.com/technologies/systems/article/21848235/the-differences-between-transmitter-types-part-2`

Una vez introducimos las señales en el modulador, a la salida obtenemos nuestra señal ya modulada y preparada para introducirla en el filtro de banda lateral. En este filtro es donde decidimos qué banda se transmitirá y qué otra no lo hará. Una vez filtrada nuestra señal, la cual es ahora una señal SSB, pasa por el mezclador junto a una señal generada por un oscilador local (LO), obteniendo como resultado una señal de alta frecuencia la cuál está preparada para ser amplificada y enviada.

Este tipo de modulación es puramente analógica, al igual que la modulación FM, las cuáles están cayendo en desuso ya que los nuevos transmisores que se están fabricando están enfocados a las señales digitales. Estos transmisores hacen uso de la tecnología DSP ("digital-signal-processing")

para procesar la información antes de enviarla. Una de las técnicas más utilizada en la modulación de este tipo de señales es la I/Q (In-Phase/Quadrature), la cuál explicamos anteriomente. Este tipo de técnica es la que se usa en la modulación QPSK, que es la que utilizaremos en nuestro experimento. Un esquema de la implementación de estas señales se muestra a continuación:

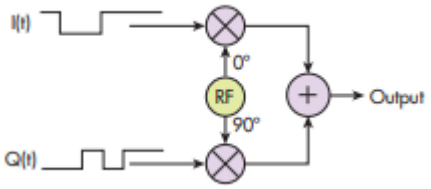


Figura 8: Representación simple de la modulación QPSK. Tomada de *The Differences Between Transmitter Types*, Demartino C. (s.f.), `https://www.mwrf.com/technologies/systems/article/21848235/the-differences-between-transmitter-types-part-2`

A partir de este tipo de técnicas se han implementado otras clases de transmisores, como son los transmisores de conversión directa o los transmisores superheterodinos.

2.3.2. Transmisor de conversión directa.

Este tipo de transmisores se usan frecuentemente debido a su simpleza a la hora de implementarlos y de su bajo coste de producción. En la **Fig. 9** se ilustra el esquema asociado a este tipo de transmisor.

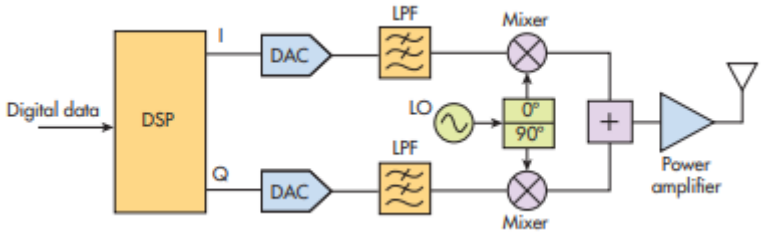


Figura 9: Esquema de un transmisor de conversión directa. Tomada de *The Differences Between Transmitter Types*, Demartino C. (s.f.), `https://www.mwrf.com/technologies/systems/article/21848235/the-differences-between-transmitter-types-part-2`

En este caso, la señal digital es la que contiene la información que queremos transmitir, la cuál es procesada con técnicas DSP resultando en dos señales I/Q en bandabase. Estas señales alimentan los respectivos convertidores digitales-analógicos o "Digital-to-Analog Converters" (DACs), cuya señal de salida entrará en nuestros filtros de paso bajo (LP).

A su vez generamos una señal RF con un LO, la cual será dividida en dos señales desfasadas 90º entre ellas. Cada una de estas señales se introduce en sus respectivos mezcladores junto a

nuestras señales en fase y en cuadratura. A la salida de los mezcladores combinamos ambas señales, obteniendo así nuestra señal modulada, y amplificándola antes de transmitirla.

2.3.3. Transmisores Superheterodinos

Este tipo de transmisores destacan por su complejidad sobre los de conversión directa. El proceso es muy parecido al usado en los transmisores de conversión directa, sólo que este proceso no termina una vez tenemos nuestra señal modulada a la salida de los mezcladores, como se muestra en la **Fig. 10**.

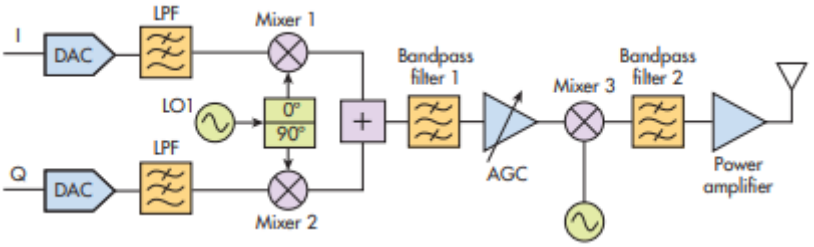


Figura 10: Esquema de un transmisor de superheterodino. Tomada de *The Differences Between Transmitter Types*, Demartino C. (s.f.), `https://www.mwrf.com/technologies/systems/article/21848235/the-differences-between-transmitter-types-part-2`

Nuestra señal, conocida como señal de frecuencia intermedia (IF), se introduce a continuación por un filtro de paso de banda. Una vez filtrada, esta se amplifica antes de ser convertida a la frecuencia de envío por el mezclador, para después ser filtrada de nuevo, amplificada y transmitida.

En nuestro experimento modulamos nuestra señal en bandabase, sin usar una frecuencia intermedia, por lo que podemos decir que se trata de un transmisor QPSK de conversión directa, y tanto este como los receptores que se han usado en este proyecto se explican en las secciones posteriores.

3. Implementación del sistema SDR en MATLAB.

Uno de los programas que nos permiten implementar este sistema de telecomunicaciones inalámbrico es MATLAB (MathWorks®, s.f.), a través del entorno SIMULINK, aunque también utilizaremos el Universal Radio Hacker (Programador Clic, s.f.) para ver algunas propiedades del mensaje un poco más adelante. A través de los siguientes bloques disponibles en el entorno SIMULINK podemos interactuar con la placa, permitiéndonos tanto transmitir como recibir señales (**Fig. 11 y 12**):

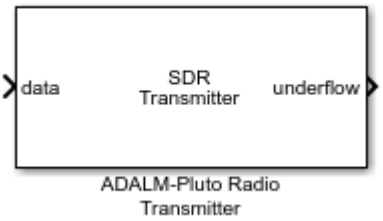


Figura 11: Bloque ADALM-PLUTO Radio transmitter.

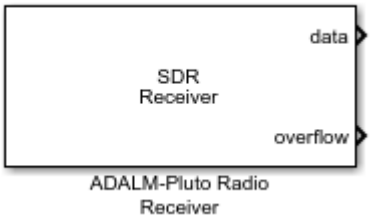


Figura 12: Bloque ADALM-PLUTO Radio receiver.

Como se puede observar, el bloque "ADALM-PLUTO Radio transmitter" tiene una entrada, "data", por la cuál enviaremos nuestra señal, que debe ser un vector columna de longitud fija, y una salida llamada "underflow", la cual a través de un scope nos permite visualizar en tiempo real los samples que se han perdido en la transmisión del mensaje, mostrándonos un 1 cuando estos se están perdiendo y 0 en caso contrario. Por otra parte, el bloque "ADALM-PLUTO Radio receiver" solo consta de dos salidas,"overflow", cuya función es la misma que la salida "underflow" del bloque transmisor, y "data", por donde se transmiten las señales recibidas a través de la placa a los demoduladores y demás componentes.

Estos bloques se comunican con la placa ADALM-PLUTO de la siguiente manera:

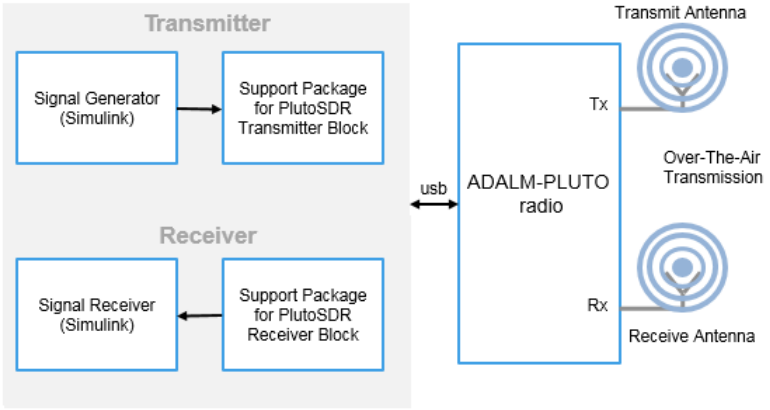


Figura 13: Esquema de funcionamiento de los bloques. Tomada de *Pluto transmitter*, Mathworks®, (s.f.), `https://es.mathworks.com/help/supportpkg/plutoradio/ref/plutotransmitter.html`

Como muestra la figura, partimos de la generación de una señal en el entorno de SIMULINK, la cuál introducimos en el bloque "ADALM-PLUTO Radio Transmitter" a través del puerto de entrada, "data". Esta señal viaja a través de la conexión USB a la placa y se transmite a través del aire con la ayuda de la antena transmisora Tx, como se muestra en la **Fig. 13**. Esta señal es percibida por la antena receptora Rx, la cual de nuevo a través de la conexión USB se transmite al bloque receptor permitiéndonos así decodificar nuestro mensaje en este entorno SIMULINK.

La ventana de diálogo de ambos bloques es muy parecida como se muestra en las siguientes figuras:

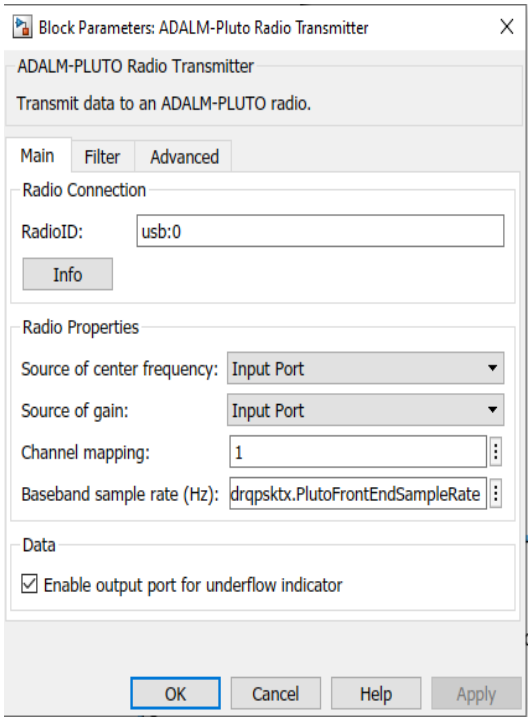


Figura 14: Ventana de diálogo del transmisor.

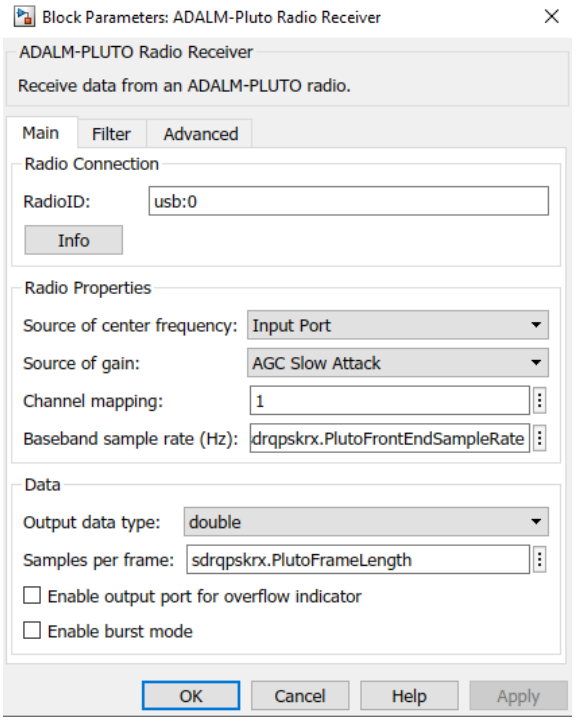


Figura 15: Ventana de diálogo del receptor.

A continuación enumeraremos los distintos parámetros que aparecen en estas ventanas de diálogo:

- **Radio ID:** Es el identificador del radio hardware especificado, es decir, el identificador de nuestra placa. Por defecto se usa "usb:0".
- **Center Frequency Source:** Nos permite especificar el modo de elección de la frecuencia de envío, ya mencionada anteriormente. Podemos elegir entre "Dialog", la cuál nos permite introducir el valor deseado directamente en la ventana de diálogo, o "Input Port", la cuál nos crea una nueva entrada, permitiendo así la elección del valor deseado a través de otro bloque conectado esta.
- **Center Frequency:** Este parámetro es la frecuencia de envío utilizada en la transmisión y recepción del mensaje, siendo un valor escalar desde 70MHz hasta 6GHz. Como podemos

observar es un rango mayor al establecido anteriormente para la placa, esto es debido a que el soporte en este entorno nos permite configurar el ADALM-PLUTO para usar el "AD9364 firmware" ampliando así el rango de frecuencias.

- **Gain Source:** Al igual que la "Center Frequency Source", nos permite especificar el modo de elección de nuestra ganancia con las mismas dos posibilidades, "Dialog" o "Input Port".
- **Gain:** Es la ganancia en dB aplicada a nuestra señal. En el caso del bloque transmisor el valor de esta debe estar comprendido entre -50 y 0, dando a entender así que en la transmisión sólo nos es posible atenuar la señal, mientras que en el bloque receptor debe estar comprendido entre -4 y 50, dándonos la posibilidad de amplificar la señal en caso de que sea demasiado débil o atenuarla un poco en caso contrario.
- **Channel Mapping:** Este parámetro siempre tiene un valor especificado de 1.
- **Baseband Sample Rate:** Es el ratio de sampleo en banda-base en Hz, es decir, el ritmo al que sampleamos la señal en el rango de banda-base (bajas frecuencias).
- **Underflow/Overflow Port:** Nos permite elegir si queremos tener la salida "underflow" u "overflow" en nuestros bloque transmisor y receptor respectivamente.
- **Output Data Type:** Nos permite elegir entre int16 (16-bit), double o single.
- **Samples per frame:** Es el número de samples efectuados por frame, cuyo valor debe ser un número par entero comprendido entre 2 y $16.777216 \cdot 10^9$.

Como se puede observar, estos dos últimos parámetros son exclusivos del bloque receptor. Dado que en este proyecto usaremos dos placas ADALM-PLUTO, las cuales se comunicarán entre ellas, es necesario que el parámetro "Center Frequency" tenga el mismo valor tanto en el transmisor como en el receptor, ya que determina la frecuencia central de nuestro ancho de banda, en caso de que no sea así, el receptor no percibirá al completo la señal que le enviamos desde el emisor.

Otros de los bloques que vamos a utilizar en la implementación de este sistema SDR en el entorno de SIMULINK de MATLAB será el modulador y el demodulador QPSK en bandabase. Empezando por el modulador tenemos las siguientes figuras:

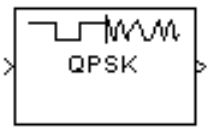


Figura 16: Bloque de SIMULINK del modulador QPSK en bandabase.

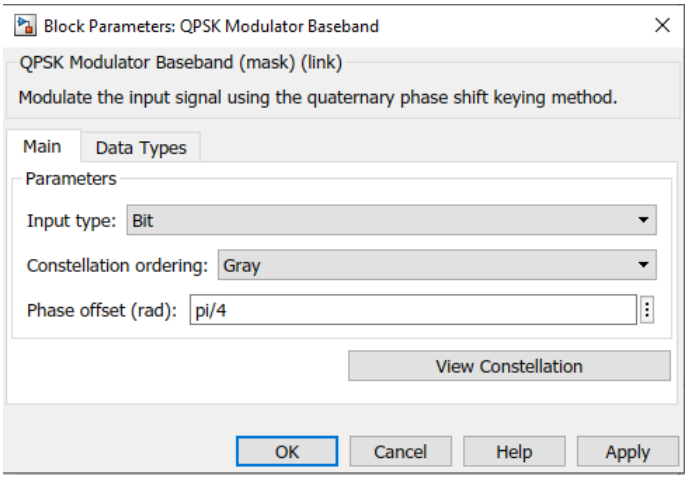


Figura 17: Ventana de diálogo del modulador QPSK en bandabase.

Para este bloque tenemos los siguientes parámetros:

- **Input Type:** Indica qué tipo de dígitos conforman la entrada, podemos elegir entre enteros ("Integers") o pares de bits ("Bit").
- **Constellation Ordering:** Determina la forma en la que el bloque nos mapea cada dígito, entero o par de bits, en el diagrama de constelaciones.
- **Phase Offset (rad):** La fase asignada al primer dígito de la constelación.

Su función, como su propio nombre indica, es modular nuestra señal usando el método "quadrature phase shift keying" obteniendo como salida una representación en bandabase de nuestra señal moduladora. Podemos elegir entre dos posibilidades del parámetro "Input Type", como hemos comentado anteriormente; la primera es que la entrada sean enteros, dandola por válida sólo si los valores son 0, 1, 2 o 3. Si a su vez elegimos la opción "Binary" del parámetro "Constellation Ordering" para una entrada $\boldsymbol{m}$, obtenemos como salida una exponencial compleja del siguiente tipo:

$$e^{j(\theta+m\frac{\pi}{2})} \tag{2}$$

donde θ representa el parámetro "Phase Offset". En este caso el bloque acepta como entrada un vector columna o un escalar. Sin embargo, si cambiamos el parámetro "Input Type" a "Bit", le estamos diciendo al bloque que la entrada estará formada por parejas de valores binarios (bits). En esta configuración el bloque acepta como entrada un vector columna de longitud par. Dando un valor para el "Phase Offset", $\pi/4$ por ejemplo, que será el valor que utilizaremos en nuestros

experimentos, el bloque nos mapea los dígitos de la señal según elijamos “Binary” o “Gray” para el parámetro “Contellation Ordering”, como se muestra a continuación:

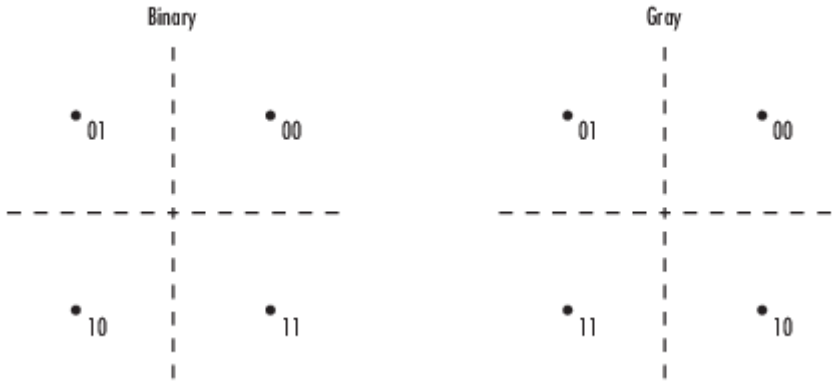


Figura 18: Diagrama de contelaciones. Tomada de *QPSK Modulator Baseband*, Mathworks®, (s.f.), `https://es.mathworks.com/help/comm/ref/qpskmodulatorbaseband.html`

Para el caso del demodulador, el bloque de SIMULINK y la ventana de diálogo son prácticamente iguales:

Figura 19: Bloque de SIMULINK del demodulador QPSK en bandabase.

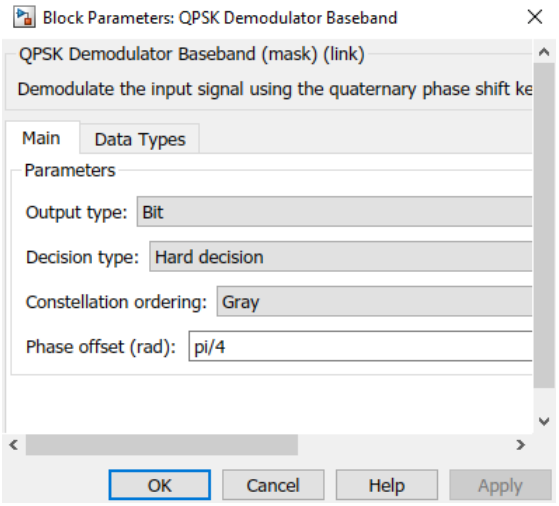


Figura 20: Ventana de diálogo del demodulador QPSK en bandabase.

Como se puede observar, en la ventana de diálogo sólo nos cambian dos parámetros con respecto al modulador, el parámetro “Input Type” es ahora “Output Type”, con el que elegimos el tipo de

salida que queremos obtener, enteros o parejas de bits, y el parámetro “Decision Type”, el cuál no aparecía en el modulador. Con este parámetro elegimos el algoritmo utilizado en la demodulación de la señal entre “Hard decision”, “LLR” o “Approximate LLR”. Básicamente, este bloque realiza la operación inversa del modulador, obteniendo así de nuevo una señal digital como salida. No entraremos en más detalle dado que este proyecto se enfoca en la etapa emisora del sistema.

También pondremos en uso los bloques “Raised Cosine Transmit Filter” y “Raised Cosine Receive Filter” en el transmisor y en el receptor respectivamente. Ambos están representados por bloques similares, aunque no tienen la misma ventana de diálogo.

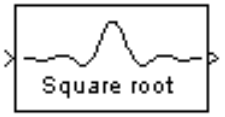


Figura 21: Bloque de SIMULINK del filtro.

Para el caso del filtro utilizado en el transmisor tenemos la siguiente ventana de diálogo:

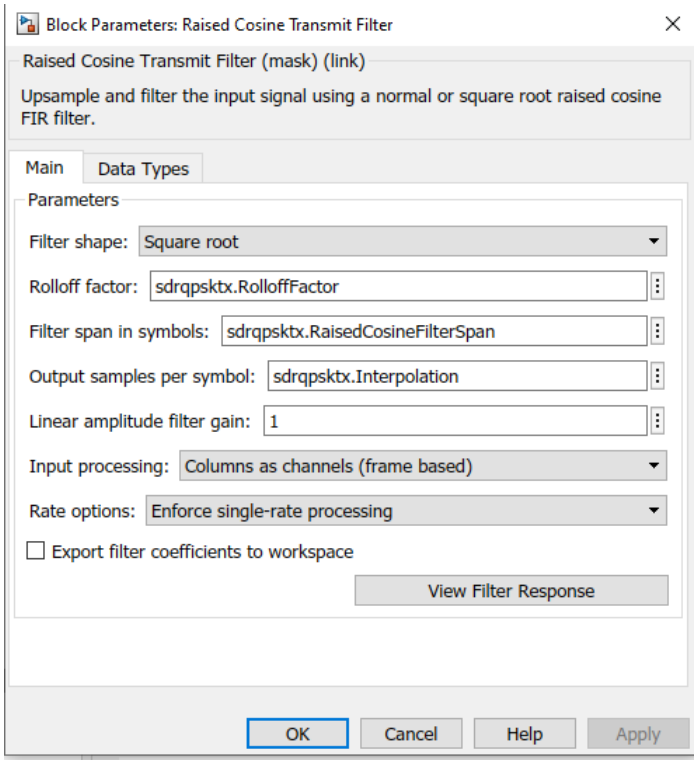


Figura 22: Ventana de diálogo del filtro del transmisor.

Este bloque sobremuestrea y filtra nuestra señal de entrada usando un filtro “raised cosine FIR” ilustrándose en el propio bloque la forma de la señal de salida. Podemos diferenciar los siguientes parámetros:

- **Filter Shape:** Especifica la forma del filtro aplicado como cuadrático ("Square Root") o normal ("Normal").
- **Rolloff Factor:** Determina el exceso de ancho de banda del filtro. Debe ser un número real entre 0 y 1.
- **Filter span in symbols:** Define el número de dígitos que abarca el filtro. Debe ser un número entero par y positivo.
- **Output samples per symbol:** Especifica el número de samples por cada dígito de la señal de salida (8 por defecto). Su valor debe ser entero y positivo.
- **Linear amplitude filter gain:** Establece un valor escalar positivo que se utiliza para escalar, valga la redundancia, los coeficientes del filtro.
- **Input Processing:** Especifica la manera en la que nuestro bloque procesa la señal de entrada. Se puede elegir entre "Columns as channels (frame based)" o "Elements as channels (sample based)".
- **Rate options:** Determina el método que utilizará el bloque para sobremuestrear y filtrar nuestra señal de entrada. Podemos elegir entre "Enforce single-rate processing" o "Allow multirate processing".

Usando la opción "Normal" para el parámetro "Filter Shape", con R como "Rolloff Factor" y T como periodo, obedece la siguiente ecuación:

$$h(t) = \frac{sin(\pi t/T)}{\pi t/T} \cdot \frac{cos(\pi Rt/T)}{(1 - 4R^2t^2/T^2)} \tag{3}$$

Mientras que si elegimos "Square root" obedecerá esta otra:

$$h(t) = 4R \cdot \frac{cos((1+R)\pi t/T) + \frac{sin((1-R)\pi t/T)}{4Rt/T}}{\pi\sqrt{T}(1 - (4Rt/T)^2)} \tag{4}$$

Dado que para un filtro ideal tenemos una respuesta impulso infinita, el bloque trunca la respuesta impulso hasta un número de dígitos que se especifica en el parámetro "Filter span in symbols", N. Este parámetro junto al "Output samples per symbol", L, determina la longitud de la respuesta impulso. El parámetro "Rolloff factor" determina el exceso de ancho de banda de nuestro filtro, como se dijo anteriormente; para el caso de un valor de 0.5, por ejemplo, que a su vez será el que utilicemos en el experimento, nos dice que el ancho de banda de nuestro filtro será 1.5 veces mayor que la frecuencia de muestreo de nuestra señal de entrada.

Para el caso del filtro usado en el receptor tenemos que el bloque es exactamente igual al utilizado en el transmisor, a diferencia del nombre como se expuso anteriormente. Sin embargo la ventana de diálogo asociada a este filtro es algo diferente:

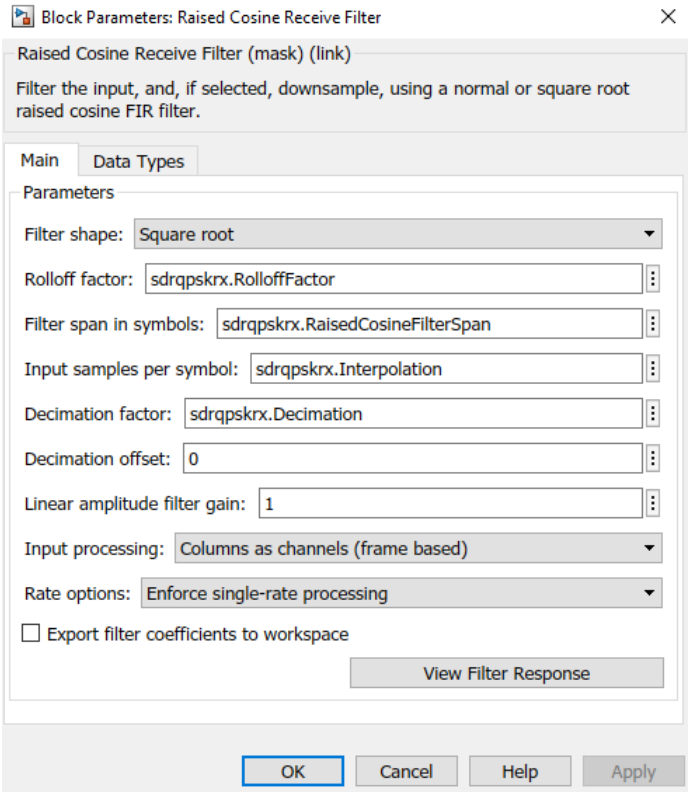


Figura 23: Ventana de diálogo del filtro del receptor.

Como se puede observar, la ventana de diálogo es muy parecida a la asociada al bloque usado en el transmisor pero con algunos parámetros diferentes. Los parámetros "Filter Shape", "Rolloff Factor", "Filter span in symbols", "Linear amplitude filter gain", "Input processing" y "Rate options" tienen las mismas funciones, pero además de estos, tenemos otros que no hemos definido anteriormente:

- **Input samples per symbol:** Especifica el número de samples por cada dígito de la señal de entrada (8 por defecto). Su valor debe ser entero y positivo.
- **Decimation factor:** Especifica el valor que el bloque aplicará a la señal. El valor del del parámetro "Output samples per symbol" de la señal de salida es el valor dado al parámetro "Input samples per symbol" dividido por este factor. Si el "Decimation factor" es igual a uno, sólo se aplica el filtrado y no la diezmación.
- **Decimation offset:** Especifica el offset en samples de la diezmación. Debe ser un valor entre cero y el valor del factor de diezmación menos uno.

3.1. Transmisor ADALM-PLUTO de modulación QPSK

En la siguiente figura puede visualizarse nuestro diagrama de bloques del emisor:

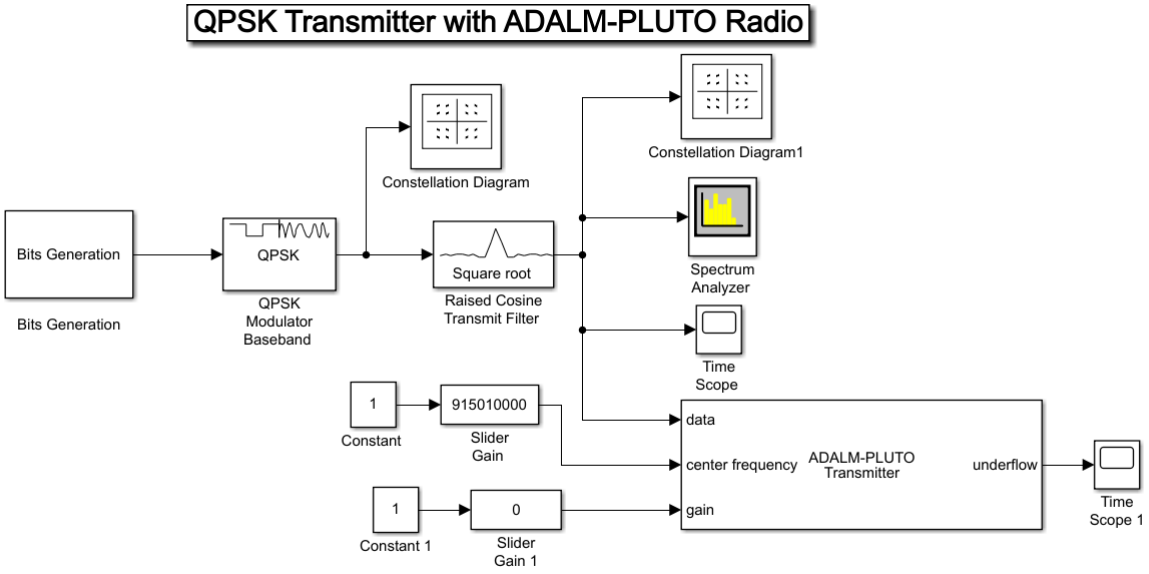


Figura 24: Modelo de SIMULINK del emisor (Sistema SDR).

Este modelo es un ejemplo expuesto en la página web de MathWorks al que se puede acceder escribiendo **"plutoradioQPSKTransmitterSimulinkExample"** en la ventana de comandos de MATLAB. Hemos introducido algunos bloques como pueden ser los diagramas de constelación, para visualizar el mensaje que estamos generando, el "Time Scope" y el "Spectrum Analyzer", para ilustrar la señal en el dominio del tiempo y de la frecuencia respectivamente, y los bloques "Constant" y "Slider Gain", los cuales nos permiten controlar en tiempo real los valores asignados a la frecuencia central y a la ganancia de nuestro bloque transmisor del ADALM-PLUTO.

Los tres primeros bloques que aparecen ("Bits Generation", "QPSK Modulator Baseband" y "Raised Cosine Transmit Filter") son los que nos generan, modulan y dan forma respectivamente a la onda que contiene nuestro mensaje. El bloque "Bits Generation" es un subsistema cuya estructura se muestra en la **Fig. 25**.

Este subsistema utiliza las variables definidas en el "workspace" de MATLAB como soporte de cada frame, que contienen una serie de cien "Hello world ###" y un encabezado, esto último por motivos de sincronización. Este encabezado está formado por 26 bits, creados a partir de un código Barker de 13 bits sobremuestreado por dos con la finalidad de tener 13 símbolos QPSK (una serie de 13 bits para cada BPSK, en fase y en cuadratura). Los demás bits corresponden a nuestro mensaje, el cual no es más que una representación ASCII de "Hello world ###" donde las almohadillas representan la secuencia '001', '002',...,'099'. Este código se introduce en el bloque

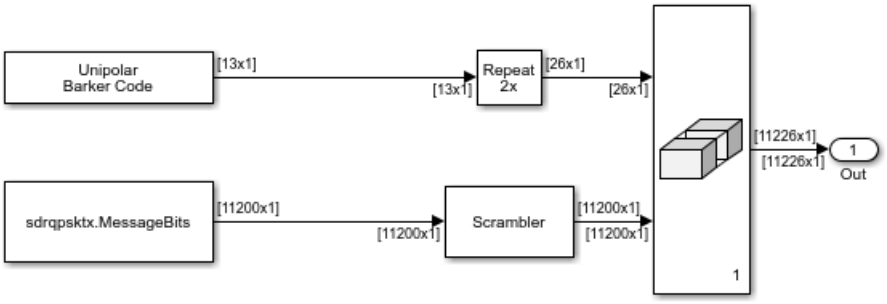


Figura 25: Subsistema “Bits Generation”.

“Scrambler” para asegurarnos una distribución equilibrada de unos y ceros con la intención de dar margen al receptor dado de tiene un tiempo de recuperación (“time recovering”) asociado.

La salida es modulada en banda-base por el modulador QPSK en código Gray (“Gray mapping”), el cuál viene representado en el diagrama de constelaciones de la **Fig. 18**, cuyo offset en fase es de $\pi/4$. Esta onda ya modulada, es sobremuestreada por el bloque “Raised Cosine Transmit Filter” con un valor para el factor roll-off de 0.5. El output rate de este bloque es aumentado hasta 400k samples por segundo con un ratio de 200k símbolos por segundo, el cual debe ser el mismo que utilicemos en el receptor.

Por último, esta señal es introducida en el bloque transmisor de nuestro ADALM-PLUTO aumentando la frencuencia hasta el valor elegido para el envío con una ganancia determinada, siendo transmitida a través de la antena ‘Tx’ de nuestra placa.

Para iniciar la transmisión hacemos uso del script que se muestra en el apéndice **A.1**. Este script genera una estructura a la que le ponemos el nombre “prmQPSKTransmitter”, que contiene los parámetros utilizados en todos los bloques de nuestro modelo. Para generar esta estructura hace llamamiento a la función “SimParams”, la cual está dividida en varias partes.

En primer lugar nos encontramos con los parámetros generales de simulación, como el orden de modulación (en nuestro caso es de orden 4 ya que estamos utilizando modulación QPSK), la frecuencia de muestreo o el tiempo entre símbolos. A continuación tenemos las especificaciones de nuestros frames, donde generamos el código Barker que usaremos como encabezado de nuestro mensaje y especificamos el contenido de nuestro mensaje, el número de mensajes que contiene cada frame y el número de bits que definen cada dígito de nuestro mensaje entre otros.

El apartado “Tx parameters” hace referencia a todos los parámetros necesarios para el bloque “Scrambler” y para el filtro que hemos utilizado en el modelo (Raised Cosine Transmit Filter). Justo después encontramos la parte donde generamos nuestro mensaje (“Message generation”), es decir, la secuencia ‘Hello world ###’ donde las almohadillas son tres números que van desde 000

hasta 099, como dijimos anteriormente; seguido del apartado donde definimos los valores de todos los parámetros de nuestro bloque ADALM-PLUTO del modelo de SIMULINK. Por último tenemos los parámetros de simulación, donde tenemos el tiempo asignado a cada frame y el tiempo que dura nuestra simulación. En este caso se ha elegido un tiempo lo suficientemente alto como para poder ir variando nuestros parámetros en tiempo real antes de que termine la simulación.

Una vez cargado el script, el modelo utiliza la estructura "prmQPSKTransmitter" generada para transmitir el mensaje a través del aire con la ayuda de nuestra placa siguiendo el proceso que hemos comentado anteriormente.

3.2. Receptor ADALM-PLUTO de modulación QPSK

Aunque este proyecto se centra en la parte emisora, he considerado de interés explicar el funcionamiento del receptor implementado por mi compañero, Javier Soler Medina, ya que así entenderemos mejor el funcionamiento del sistema de telecomunicaciones. Para ello se ha utilizado un modelo que también aparece como ejemplo en la página web de MathWorks el cual se abre escribiendo **"plutoradioQPSKReceiverSimulinkExample"** en la ventana de comandos de MATLAB, y al que le ha aplicado unas pequeñas modificaciones para poder variar los parámetros que nos interesan en tiempo real y observar el comportamiento de la onda recibida, como hicimos con el transmisor. Este modelo se muestra en la siguiente figura:

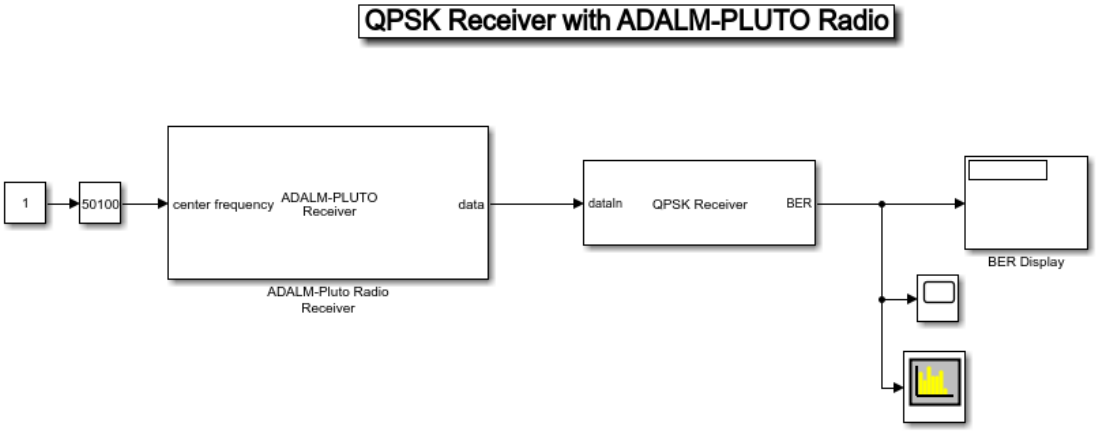


Figura 26: Modelo de SIMULINK del receptor (Sistema SDR).

Como se puede observar en este modelo también se ha utilizado los bloques "Constant" y "Slider Gain" para controlar en tiempo real el valor asignado a la frecuencia central del bloque emisor. La salida de este bloque será la señal recibida a través de la antena 'Rx' de la segunda placa ADALM-PLUTO ya transferida a bandabase y con un determinado factor de ganancia. Esta señal

es introducida en el subsistema "QPSK Receiver", cuya estructura se representa en la siguiente figura, y donde tenemos varios diagramas de constelación para visualizar nuestro mensaje, además del analizador de espectro, que nos permite visualizar tanto señal recibida como la filtrada en el dominio de la frecuencia.

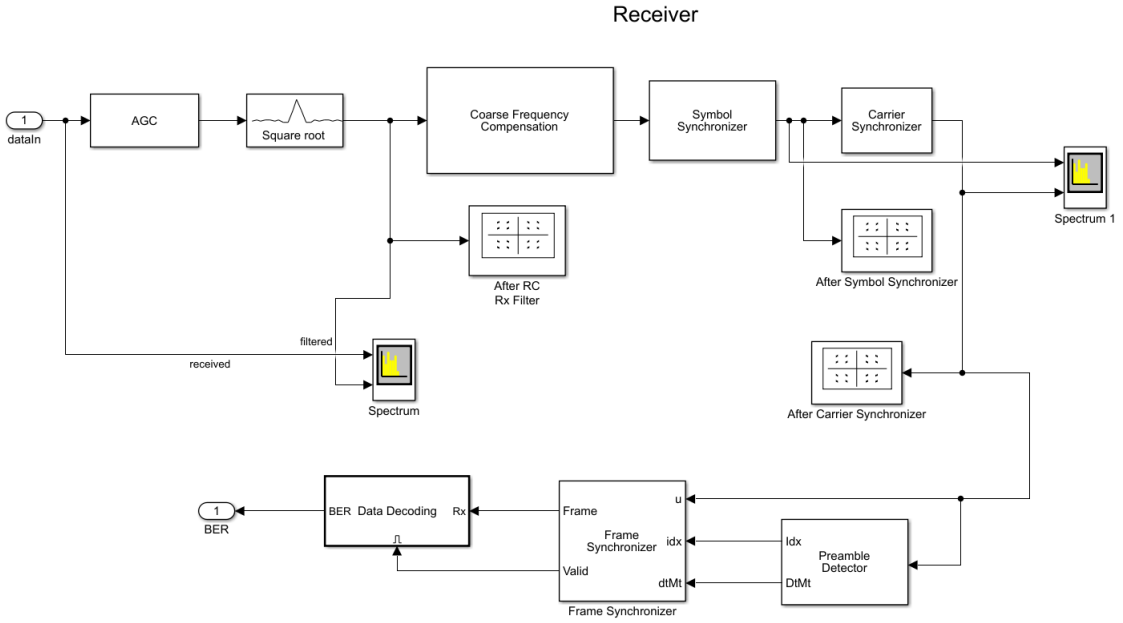


Figura 27: Subsistema "QPSK Receiver".

El primer bloque que nos encontramos es el AGC (Automatic Gain Control). Dado que la amplitud de la señal recibida puede afectar a la precisión de los bloques "Carrier Synchronizer" y "Symbol Synchronizer", debemos estabilizarla para asegurar que nuestro sistema funciona correctamente, y esto es precisamente lo que nos permite el AGC. Determina el valor de la potencia de salida asegurando así que la ganancia de fase y el timing error permanencen constantes a lo largo del tiempo. Este bloque se sitúa justo antes del filtro "Raised Cosine Receive Filter", cuyo factor roll-off es de 0.5, con intención de poder sobremuestrear la amplitud con un factor de dos, mejorando así la exactitud de la estimación.

A continuación, tenemos el subsistema "Coarse Frequency Compensation" (**Fig. 28**), cuya función es estimar el offset en frecuencia debido a la transmisión y recepción del mensaje con la ayuda del bloque "Coarse Frequency Compensator", usando algoritmos de correlación; y compensarlo, tarea que recae sobre el bloque "Phase/Frequency Offset". Aun después de la compensación seguirá existiendo un offset residual que desembocará en una rotación del diagrama de constelación.

El bloque "Symbol Synchronizer" es el que nos reduce el mismatch asociado al ratio de muestreo entre el transmisor y el receptor. Con la ayuda de un PLL ("Phase-Locked Loop") corrije el error asociado al timing de nuestros símbolos en la señal recibida.

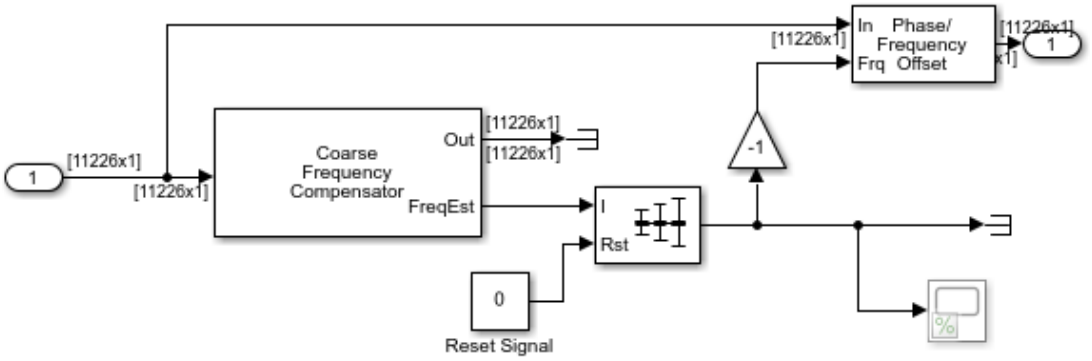


Figura 28: Subsistema "Coarse Frequency Compensation".

Su salida llega al "Carrier Synchronizer" donde se realiza una compensación en frecuencia más precisa. Este bloque también implementa un PLL (Phase-Locked Loop) para determinar el offset en frecuencia y en fase residuales. El PLL usa un DDS (Direct Digital Synthesizer) para generar esta compensación, la cual se estima a través del error en fase de la salida de un Loop Filter. Justo después de esto entramos en el bloque "Preamble Detector", el cual detecta la localización del código Barker que usamos como encabezado y así obtener la secuencia de símbolos de nuestro mensaje, seguido de una última sincronización de nuestros frames de cara a la decodificación.

Por último tenemos el subsistema "Data Decoding" donde se sitúa el demodulador QPSK. También está el modelo usado para la generación de nuestro mesanje en el transmisor pudiendo así comparar ambas señales y obtener el BER, una estrucutura de tres valores que corresponden al ratio de errores, el número de errores detectados, y el número de samples comparados.

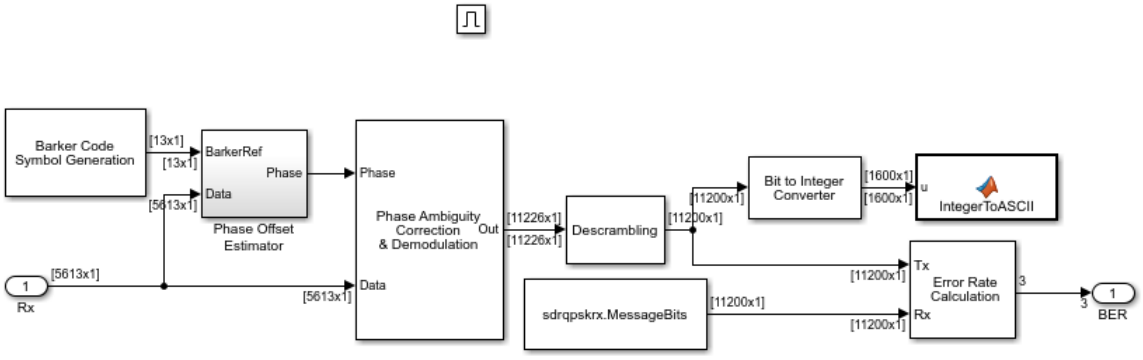


Figura 29: Subsistema "Data Decoding".

Para poner en fucionamiento el modelo del receptor es necesario ejecutar el script detallado en el apéndice **A.2**, que como se puede observar es muy parecido al utilizado en el transmisor excepto por los demás parámetros necesarios para los bloques utilizados en el control de ganacia y la compensación en frencuecia. Además contiene un apartado adicional donde generamos la estructura BER con los resultados de nuestra simulación (porcentaje de errores, número de errores detectados

y número de samples comparados). Una vez ejecutado, el mensaje recibido será decodificado y se mostrará en la ventana de comandos de MATLAB mientras corremos la simulación. La estructura BER también se mostrará una vez termine esta.

4. Implementación del receptor en Universal Radio Hacker

Además de MATLAB, también usaremos el software Universal Radio Hacker (URH) para el análisis espectral de nuestra onda. Este software es capaz de cooperar con distintos tipos de equipos SDR, entre los cuales está nuestra placa ADALM-PLUTO. Nos permite utilizar funciones de demodulación de señal y decodificación personalizable, aunque esto concretamente nos resultará más fácil estudiarlo en MATLAB. En general URH nos divide nuestro estudio en tres etapas, análisis de señales, generación de señales y simulación de señales, aunque como hemos comentado nosotros nos centraremos en la primera.

URH recibe señales de muestreo inalámbrizas mediante el control de equipos SDR y las convierte en secuencias de parejas de bits. Específicamente, este software tiene los siguientes objetivos:

- Proveer de una interfaz de análisis ituitiva y que no requira a los ususarios tener conocimiento profundo de radiofrecuencia (RF).
- Proporcionar cobertura de todas las operaciones involucradas en el ataque a un protocolo privado inalámbrico.
- Facilitar una interfaz que permita trabajar con otro software.

Como hemos comentado, dentro de las distintas herramientas que nos proporciona URH tenemos el "Spectrum Analyzer". Esta herramienta nos permite visualizar nuestra señal en el dominio de la frecuencia además de proveer de un diagrama de potencia, pudiendo así obtener una idea más precisa del estado de nuestra onda. En la **Fig. 30** se muestra la interfaz de este analizador de espectro, el cual se ha usado para estudiar un tono de 315MHz.

A la izquierda de esa ventana tenemos los distintos parámetros a elegir para nuestro análisis, el dispositivo que se usará, la frecuencia central de la onda a estudiar, la frecuencia de muestreo, la ganancia y el ancho de banda entre otros. En la esquina superior derecha se puede visualizar el espectro de nuestro tono donde la banda negra es la onda visualizada instantáneamente y la banda roja representa los máximos obtenidos hasta ese momento, mientras que en la esquina inferior podemos ver el diagrama de potencia.

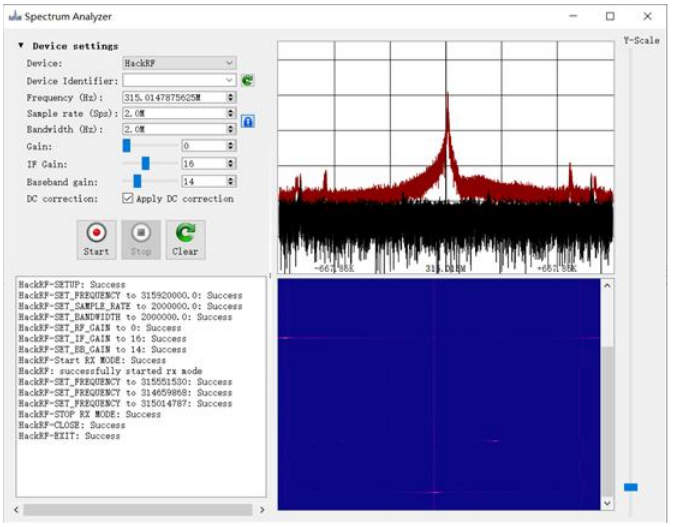


Figura 30: Espectro de un tono de 315MHz.

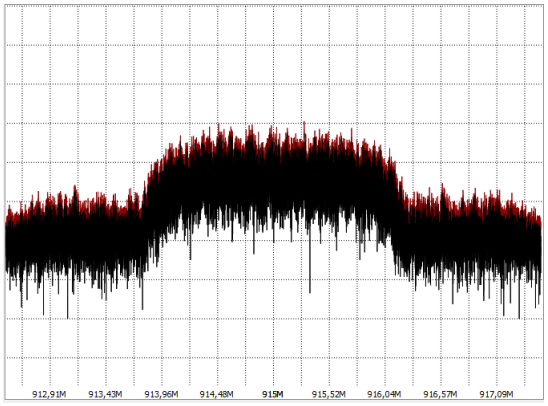
Con la ayuda de esta herramienta llevaremos a cabo el análisis espectral de nuestra onda transmitida. Para el transmisor utilizaremos el modelo de MATLAB expuesto en el punto anterior, e introduciendo la frecuencia central seleccionada en el analizador de espectro del URH podremos visualizar en tiempo real nuestra onda recibida a través de la placa ADALM-PLUTO y su comportamiento en distintas situaciones.

5. Resultados

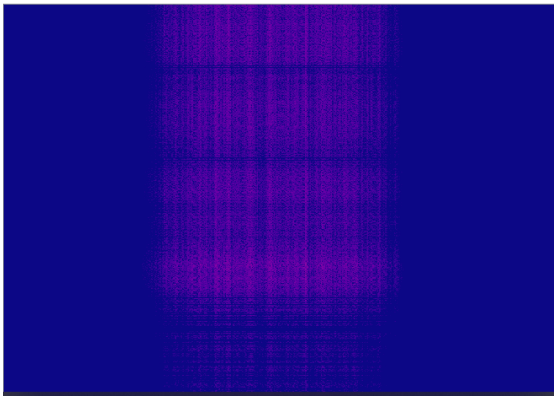
A continucación estudiaremos el comportamiento del sistema SDR anteriormente expuestos en las distintas plataformas (MATLAB y Universal Radio Hacker), visualizando así las propiedades del mensaje en las distintas condiciones de transmisión y recepción.

5.1. Estudio del espectro a través del software Universal Radio Hacker

Para empezar hemos utilizado este software para la placa receptora ya que nos permite visualizar un ancho de banda de 56MHz a través de su herramienta “Spectrum Analyzer”, un valor bastante más grande que el permitido en el entorno SIMULINK a través del bloque de mismo nombre (“Spectrum Analyzer”). En este caso estudiaremos el espectro de frecuencias para distintos valores de la frecuencia de transmisión, de forma que podemos viasualizar nuestra onda tanto para la frecuencia definida en el ejemplo (915MHz) o canales de uso libre, los cuales están desocupados, como para bandas de frecuencia asignadas a cadenas de radio o televisión.

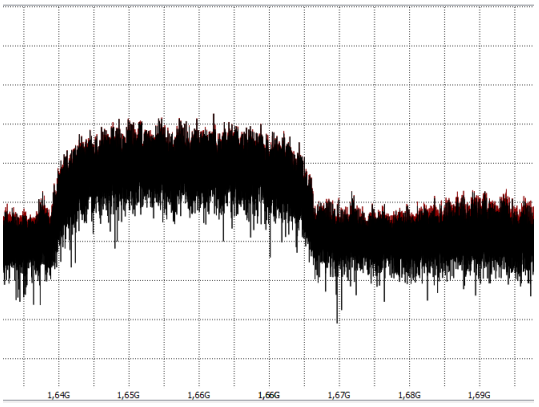


(a) Espectro de frecuencias.

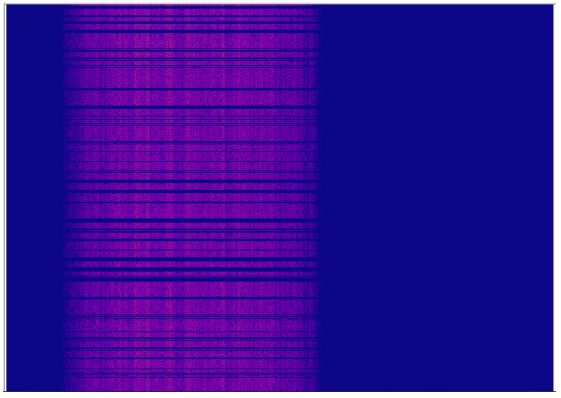


(b) Diagrama de potencia.

Figura 31: Frecuencia ejemplo (915MHz).



(a) Espectro de frecuencias.

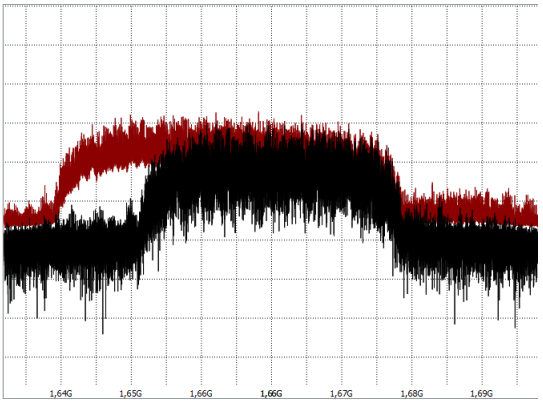


(b) Diagrama de potencia.

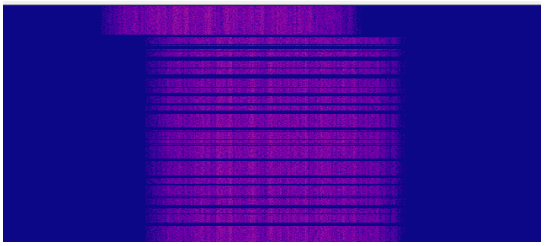
Figura 32: Banda de uso libre (1655MHz).

Con la ayuda del modelo anteriormente expuesto para el emisor (**Fig. 24**) y el script que nos define la estructura que contiene los valores de las distintas variables necesarias, iniciamos la transmisión, y a través del bloque "Slider Gain" le vamos asignando distintos valores a la frecuencia de transmisión ("Central Frequency") de nuestro bloque ADALM-PLUTO transmisor en tiempo real, de forma que visualicemos esta variación en Universal Radio Hacker.

En estas figuras se muestran los espectros obtenidos, además de una gráfica a color que representa la evolución en el tiempo de la potencia de nuestra señal, donde el color azul se da para valores más bajos de la potencia y el color rojo los valores más altos. El eje de abscisas representaría a la frecuencia mientras que el de ordenadas sería el tiempo.

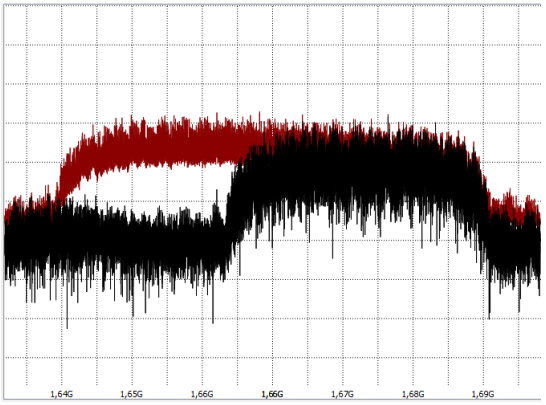


(a) Espectro de frecuencias.

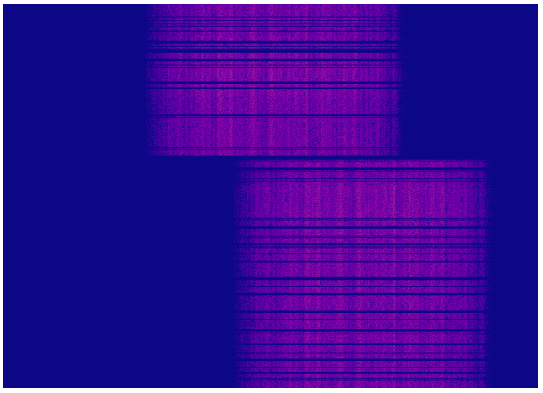


(b) Diagrama de potencia.

Figura 33: Banda de uso libre (1663MHz).



(a) Espectro de frecuencias.



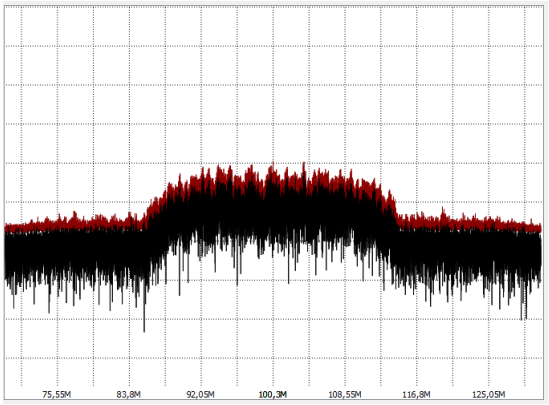
(b) Diagrama de potencia.

Figura 34: Banda de uso libre (1670MHz).

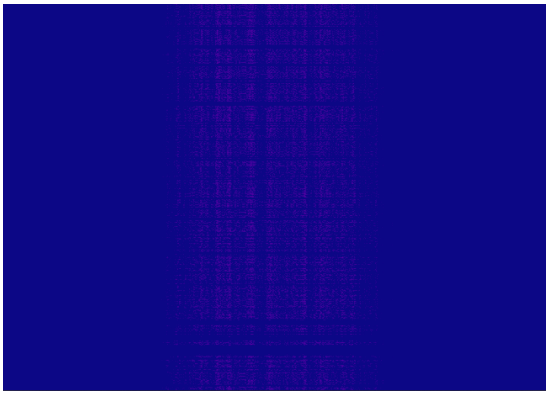
En este caso vemos como al elegir bandas de frecuencia que están mayormente desocupadas, nuestra señal destaca en el espectro distinguiéndose bien del ruido, de forma que no perderemos parte de nuestro mensaje dado que apenas hay efecto de apantallamiento, aunque siempre exista un ruido residual.

Estos canales corresponden a la banda de frecuencia de uso libre asignada por el gobierno la cual comprende desde 1660.5MHz hasta 1670MHz (Subdirección General de Planificación y Gestión del Espectro Radioeléctrico. Secretaría de Estado de Telecomunicaciones e Infraestructuras Digitales. Ministerio de Asuntos Económicos y Transformación Digital. Abril, 2011), y donde se puede visualizar en los diagramas a color que la señal recibida en esta banda tiene mayor potencia, esto es debido a que la banda de 915MHz aunque aparece mayormente desocupada también está destinada a usos industriales, por lo que podemos ver que el color rojo es algo más débil debido al apantallamiento con otras señales.

En las siguientes figuras se muestran los espectros y los diagramas de potencia asociados a bandas

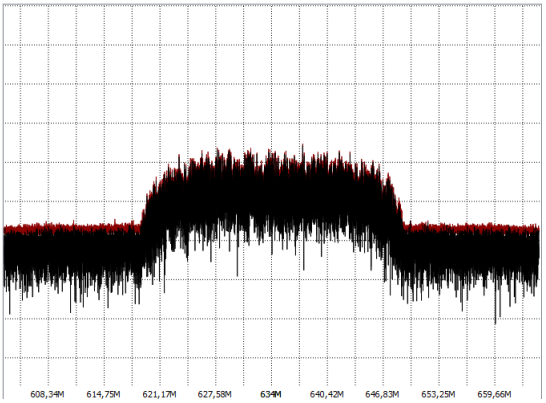


(a) Espectro de frecuencias.

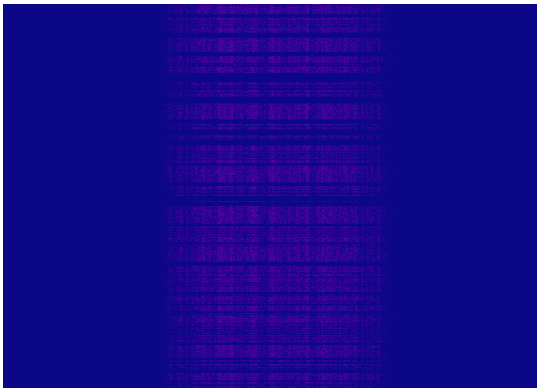


(b) Diagrama de potencia.

Figura 35: KISS FM (100.3MHz).



(a) Espectro de frecuencias.



(b) Diagrama de potencia.

Figura 36: Antena 3 HD (634MHz).

de frecuencia que sí están ocupadas, concretamente a las frecuencias asignadas en Sevilla a la cadena de radio KISS FM (100.3MHz) y al canal de televisión Antena 3 HD (634MHz).

Como se puede observar, aunque reconozcamos la forma de nuestra onda en el espectro de frecuencias (**Fig. 35.a y 36.a**), si nos fijamos en los diagramas de potencia que tenemos a la derecha (**Fig. 35.b y 36.b**) la señal es casi indistinguible del ruido (peor relación señal-ruido), debido a al ampantallamiento que resulta por la emisión de estas empresas en esta banda de frecuencias. En este caso parte de nuestro mensaje se está perdiendo, lo que nos llevará a errores de decodificación del mensaje que obtendremos posteriormente en MATLAB.

Además, es conveniente señalar que hemos tenido que aumentar el valor de la ganancia aplicada a la señal recibida en el caso de 634MHz porque ni siquiera llegábamos a visualizar la potencia de la señal en el diagrama.

5.2. Estudio de la variación de los parámetros en el entorno SIMULINK de MATLAB

Iniciamos la transmisión del mensaje a través del script que hemos expuesto en el apéndice **A.1**, y utilizando los distintos diagramas de constelaciones y los bloques "Time Scope" y "Spectrum Analyzer" observamos las propiedades de nuestra onda en las distintas etapas de nuestro transmisor. En las siguientes figuras se muestran las distintas representaciones obtenidas a través de estos bloques para una frecuencia de transmisión de 915MHz.

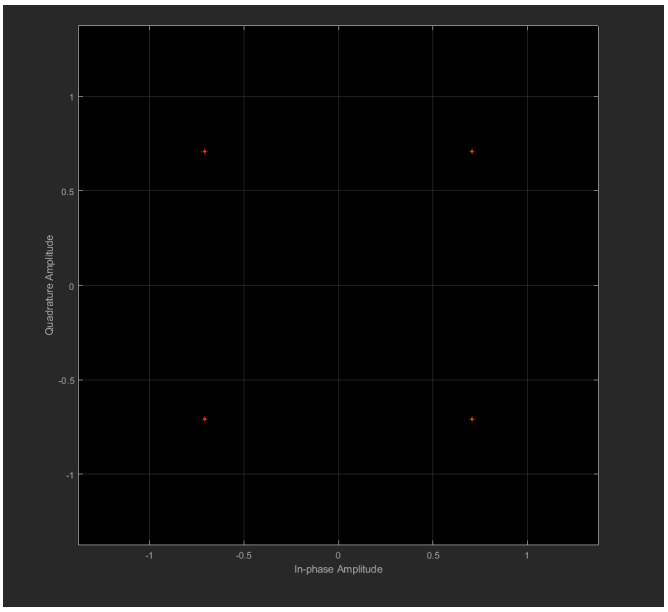


Figura 37: Diagrama de constelaciones después del modulador QPSK (915MHz).

En primer lugar tenemos el diagrama de constelaciones que como habíamos explicado anteriormente, representan los distintos dígitos posibles, esto es, 00, 01, 10 y 11 en las fases $\pi/4$, $3\pi/4$, $5\pi/4$ y $7\pi/4$ respectivamente, aunque no se distingan del todo en la figura. En nuestro eje de abcisas tenemos nuestras componentes en fase y en el eje de ordenadas nuestras componentes en cuadratura, de forma que tenemos una superposición de puntos de las diferentes combinaciones de bits. No observamos ningún tipo de distorsión debido a que todavía no ha tenido lugar la transmisión a través del aire del mensaje.

Tras pasar por el filtro podemos visualizar nuestra señal en el dominio de la frecuencia y del tiempo (**Fig. 38 y 39**).

Con esta representación ya tenemos una idea de la forma de nuestra onda en el dominio de la frecuencia, esto nos servirá de guía a la hora de comparar la distorsión producida en la onda durante la transmisión y recepción de la onda.

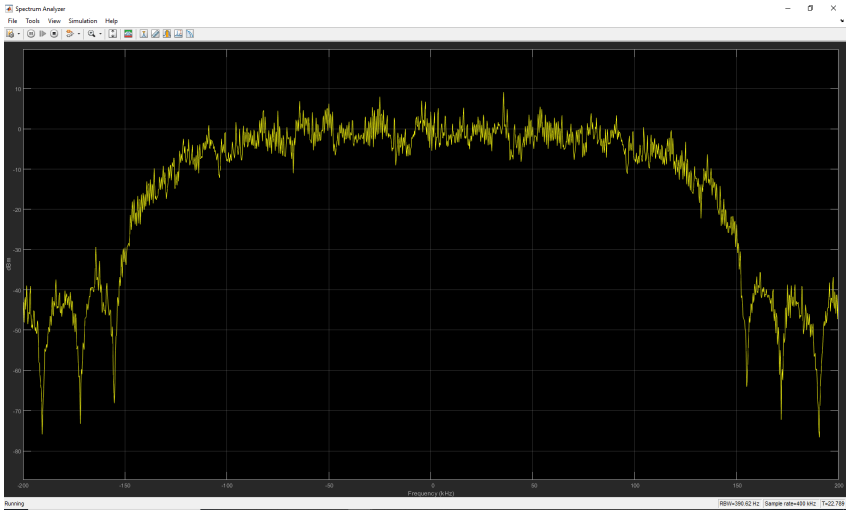
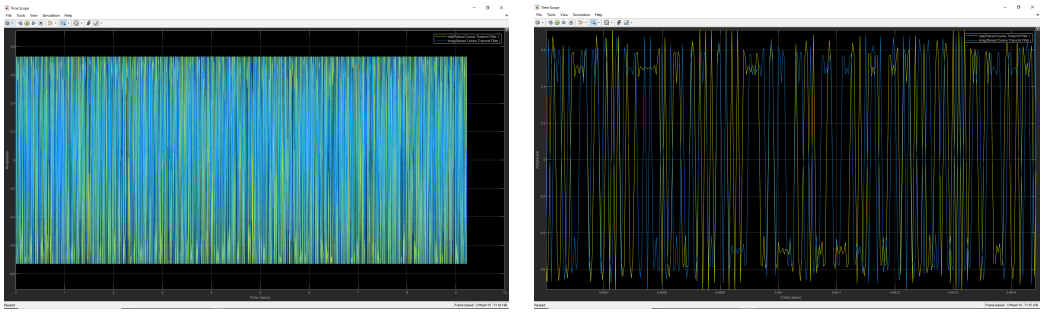


Figura 38: Representación de la señal en el dominio de la frecuencia.



(a) Escala normal. (b) Ampliada.

Figura 39: Representación de la señal en el dominio del tiempo.

Por último mostraremos los valores que nos da el puerto "underflow" para visualizar que no se está perdiendo parte de los samples que contienen el mensaje (**Fig. 40**). Como se puede observar, en determinadas ocasiones obtenemos el valor unidad como muestra de que en ese momento se han perdido samples del mensaje, aunque dado que en comparación con el total son muy pocos los samples perdidos podemos achacarlo a la propia electrónica del sistema.

Una vez en este punto, sabemos la forma que tiene nuestra onda y que la transmisión se está realizando correctamente, por lo que iniciaremos nuestro modelo del receptor para ver las propiedades de la onda recibida y la decodificación del mensaje.

A continuación ejecutamos el script que contiene los valores de las variables de simulación para el receptor, de forma que mediante la placa ADALM-PLUTO recibimos la onda y la decodificamos siguiendo el modelo anteriormente expuesto (**Fig. 26**), obteniendo los diagramas de coonstelaciones ilustrados en la **Fig. 41**.

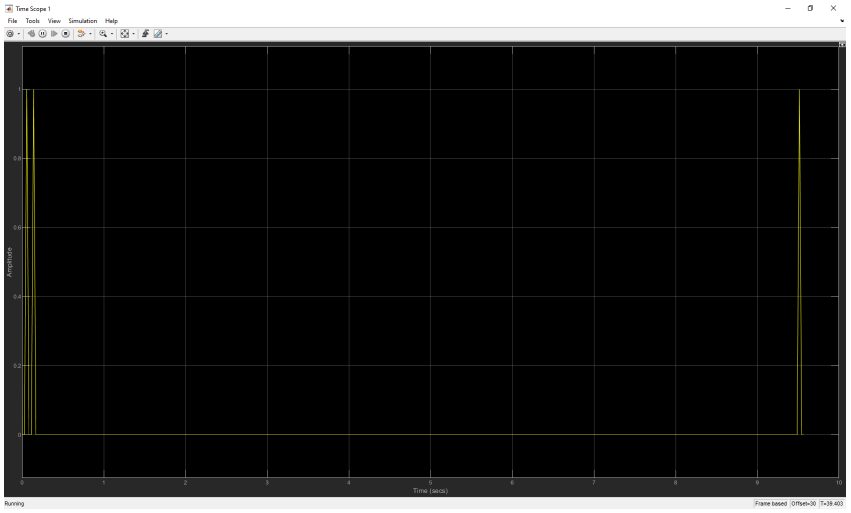
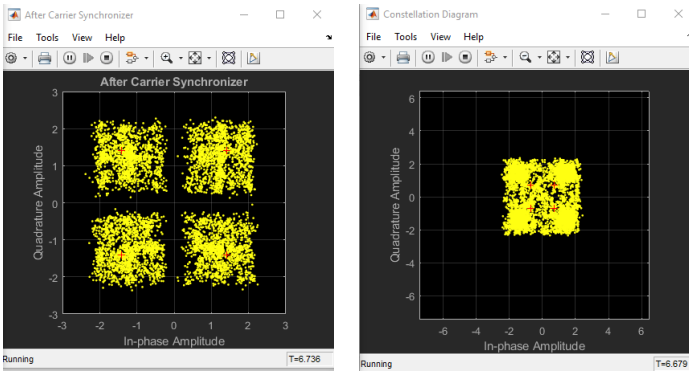


Figura 40: Representación del puerto “underflow”.



(a) Después del “Carrier Symchronizer” (b) Antes del demodulador QPSK

Figura 41: Diagramas de constelaciones del receptor (915MHz).

El primero lo obtenemos una vez se ha llevado a acabo la corrección del offset en frecuencia asociado a la transmisión como habíamos comentado anteriormente, de forma que se visualizan fácilmente los cuatro grupos asociados a cada combinación de los bits, y el segundo es justo antes de pasar por el demodulador QPSK, donde también podemos diferenciar estos grupos. Es conveniente señalar que estos gráficos van cambiando a lo largo de la realización del experimento por lo que nos ha sido dificil capturarlos en el momento que el mensaje se estaba transmitiendo y así ver los cuatro grupos perfectamente separados los unos de los otros, debido a esto el último diagrama se ve algo más difuso.

También podemos visualizar nuestra onda en el dominio de la frecuencia con la ayuda de analizador de espectro que hemos colocado en el modelo. Este nos permite visualizar la onda tanto tras la operación de recuperación ("timing recovery") como tras la compensación del offset en frecuencia, como se muestra en la **Fig. 42**. Comparando este espectro con el obtenido en el transmisor podemos decir que ambas operaciones se están realizando correctamente y nuestra onda recibida tiene una gran similitud con la generada.

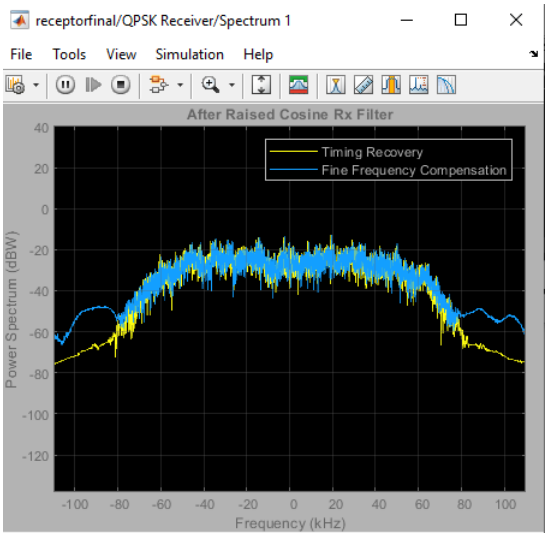
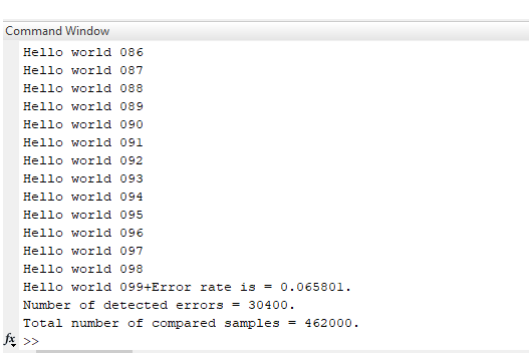


Figura 42: Espectro de frecuencias para la señal recibida.

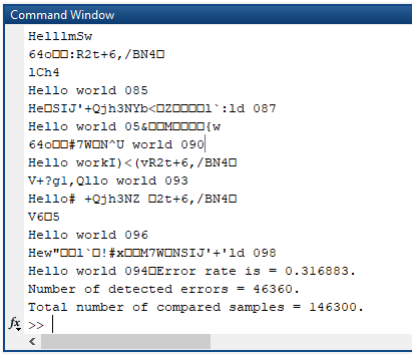
Una vez terminada la simulación se imprime el mensaje obtenido en la ventana de comandos de MATLAB junto a los valores que contiene la estructura BER, que como hemos dicho antes coinciden con el ratio de errores, el número errores detectados y el número total de samples comparados. Esto se ilustra en la siguiente figura:

```
Command Window
  Hello world 085
  Hello world 086
  Hello world 087
  Hello world 088
  Hello world 089
  Hello world 090
  Hello world 091
  Hello world 092
  Hello world 093
  Hello world 094
  Hello world 095
  Hello world 096
  Hello world 097
  Hello world 098
  Hello world 099Error rate is = 0.000000.
  Number of detected errors = 0.
  Total number of compared samples = 1309000.
fx >>
```

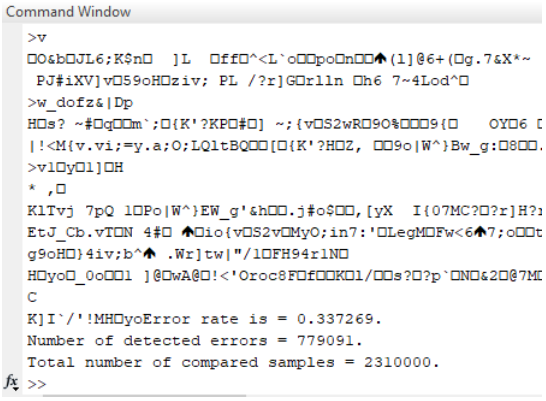
Figura 43: Mensaje decodificado y valores de la estructura BER.



(a) 915.0008MHz-915MHz



(b) 915.00095MHz-915MHz



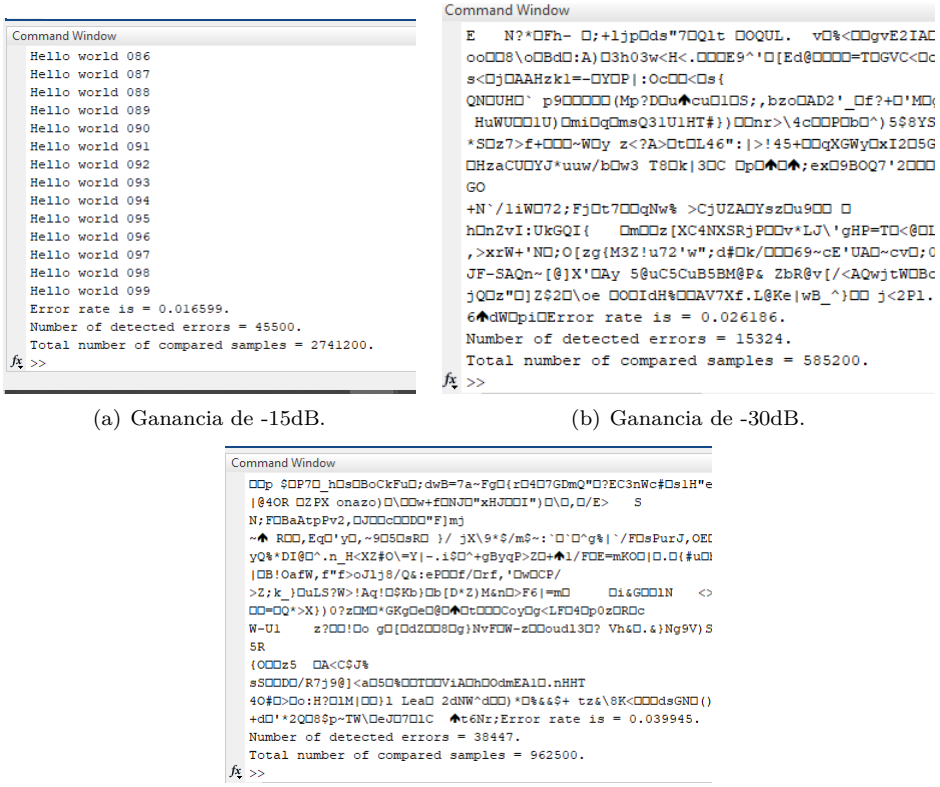
(c) 915.001MHz-915MHz

Figura 44: Decodificación del mensaje para valores diferentes de la frecuencia central en el transmisor y en el receptor.

Como se puede observar, tenemos un 0 % de errores en la recepción del mensaje, por lo que está completamente decodificado y no se ha perdido parte de él durante el proceso de transmisión-recepción (**Fig. 43**).

Sabiendo que nuestro sistema funciona correctamente, empezamos a variar los distintos parámetros de nuestro sistema con la intención de visualizar los límites de este. En primer lugar variamos lentamente el valor asignado a la frecuencia central del transmisor dejando fijo el del receptor en 915MHz, y así ver el efecto del ancho de banda en nuestro sistema, ilustrado en la **Fig. 44**.

Se observa que a medida que distanciamos más el valor de la frecuencia central en el transmisor con respecto al receptor vamos obteniendo un ratio de errores cada vez mayor, partiendo de un 6 % aproximadamente en el primer caso, hasta prácticamente un 34 % en el último. De esta forma reafirmamos la necesidad de sincronización en términos de frecuencia central entre transmisor y receptor para que podamos decodificar íntegramente nuestro mensaje.



(a) Ganancia de -15dB.

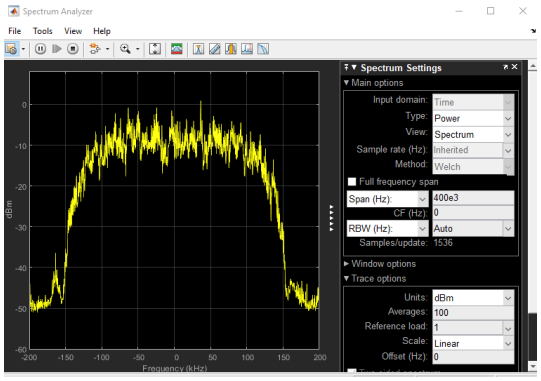
(b) Ganancia de -30dB.

(c) Ganancia de -50dB.

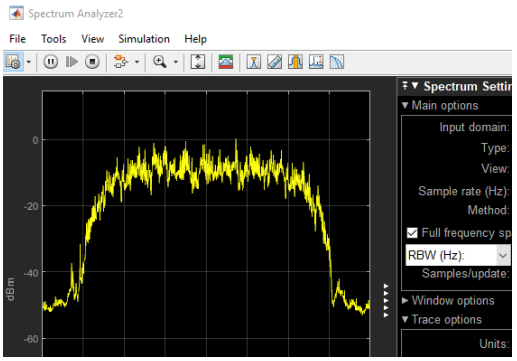
Figura 45: Decodificación del mensaje para 915MHz.

A continuación variamos el otro parámetro dentro del bloque transmisor dejando fija la frecuencia central en 915MHz, que no es más que la ganancia en decibelios aplicada. La intención de este experimento fue visualizar el efecto que tiene la atenuación de la señal en la decodificación de esta, de modo que se espera que a mayor atenuación tendremos un efecto de apantallamiento más notable, por lo que se obtiene un mayor ratio de errores.

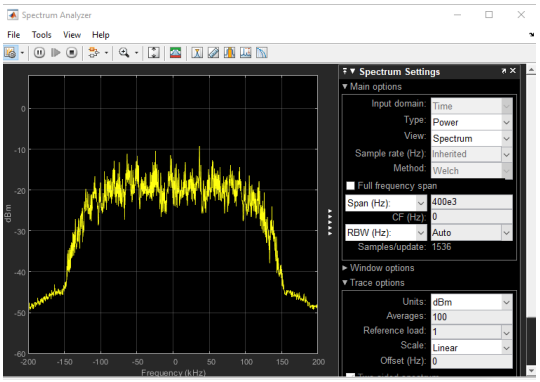
Partiendo del valor nulo de la ganancia (no hay atenuación) con el que hemos obtenido un ratio de errores de 0 %, como se ilustraba en la **Fig. 43**, aumentamos su valor, concretamente hemos elegido los valores -15dB, -30dB y -50dB de forma que veamos el comportamiento de la señal a lo largo del intervalo permitido por el bloque transmisor. En la (**Fig. 45**) se ilustran los resultados.



(a) Ganancia de 0dB.



(b) Ganancia de -25dB.

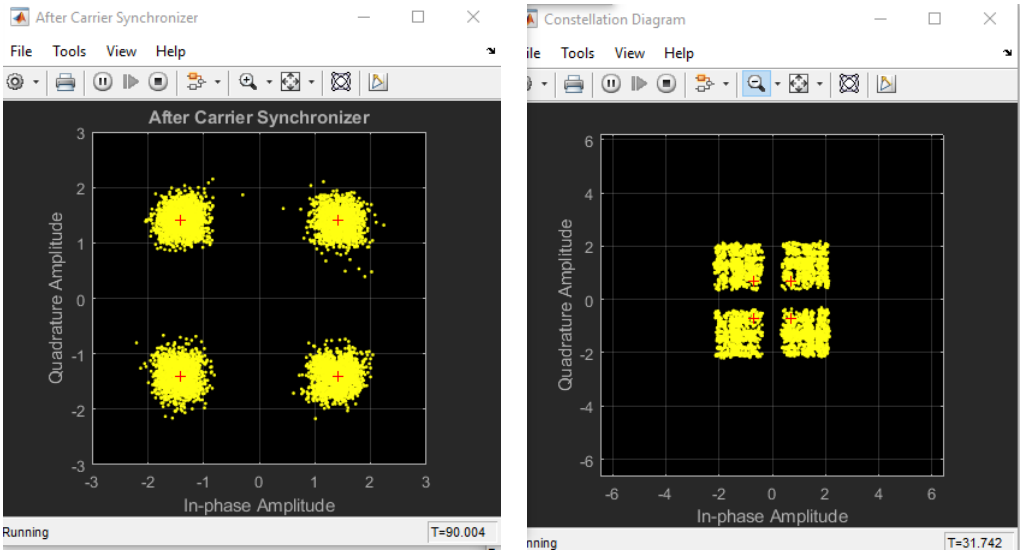


(c) Ganancia de -50dB.

Figura 46: Espectros de frecuencia para 915MHz.

Como se puede observar a medida que aumentamos la atenuación se cumple lo esperado, en el primer caso obtenemos un ratio de errores de 1.6 %, en el segundo 2.6 % y en el tercero 4 % aproximadamente (**Fig. 45**) confirmando así lo dicho anteriormente. Aun así, como se puede comprobar no son valores demasiado altos y aunque en las **Fig. 45.b y 45.c** no se vea reflejado dado que los fallos solían ser al final de la simulación, la mayor parte del mensaje se decodifica correctamente, por lo que podemos concluir que la ganancia no es un factor demasiado influyente aunque tiene su efecto al reducir el ratio señal-ruido.

También hemos visualizado el efecto de la atenuación por parte del transmisor en el espectro de frecuencias (**Fig. 46**), aunque en este caso hemos elegido valores de la ganancia de 0dB, -25dB y -50dB para observarlo sólo de forma cualitativa. Es fácil observar cómo el máximo va disminuyendo a medida que aumenta la atenuación como es lógico.



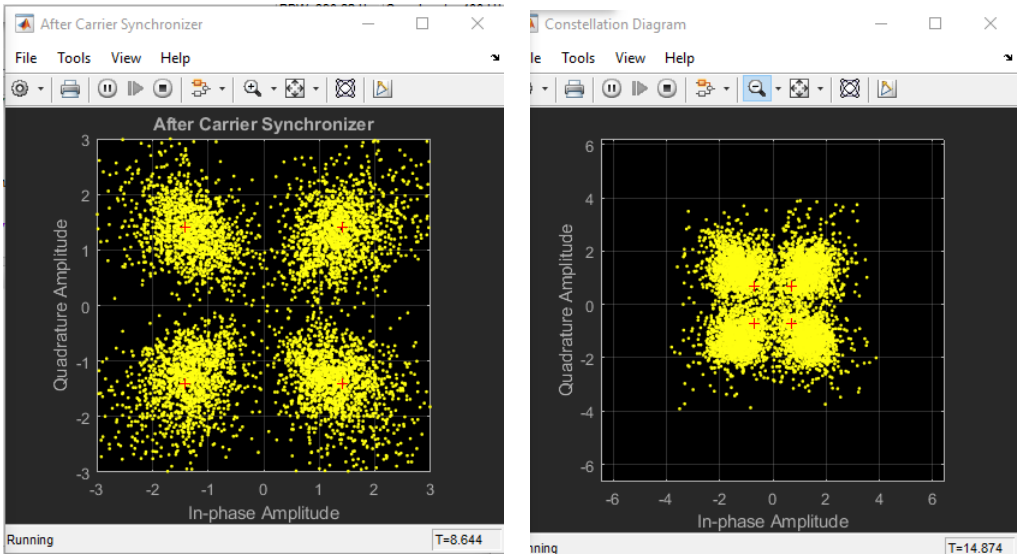
(a) Después del "Carrier Symchronizer" (b) Antes del demodulador QPSK

Figura 47: Diagramas de constelaciones para 1.6631GHz (canal libre).

Una vez visto el efecto de la ganancia en nuestro sistema, estudiaremos a través de los diagramas de constelaciones la distribución de bits del mensaje recibido dependiendo del canal que utilicemos, de forma que se espera obtener una distribución más difusa para los canales que están ocupados y otra más definida para los canales desocupados. Hemos elegido los mismos valores para la frecuencia central que habíamos elegido anteriormente, es decir, la banda de uso libre según el Gobierno de España (1660.5MHz - 1670MHz), el canal correspondiente a KISS FM y el correspodiente a Antena 3 HD ya que el diagrama de constelaciones asociado a la frecuencia 915MHz se había mostrado anteriormente.

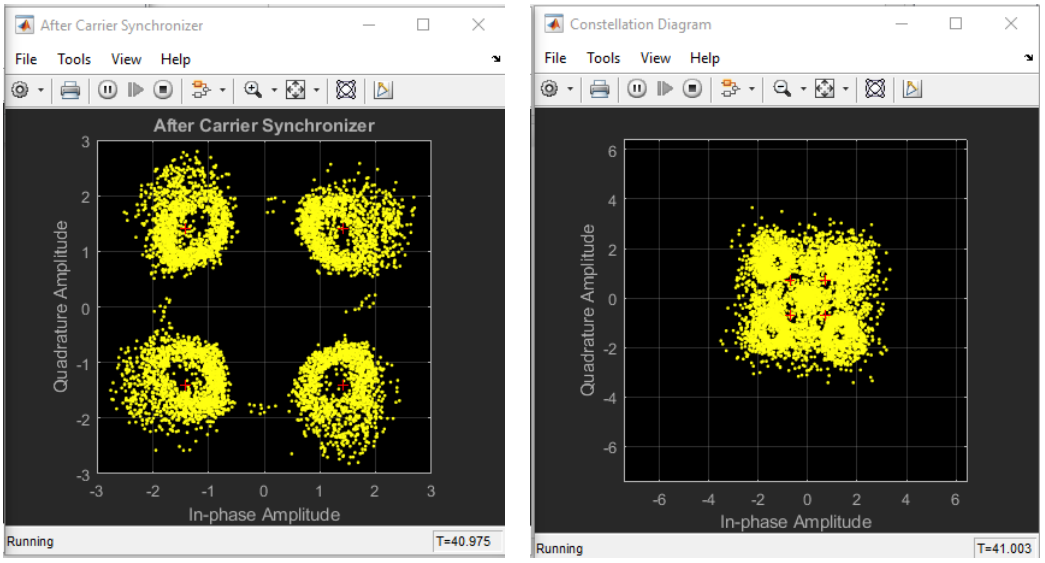
Como se puede observar obtenemos unos diagramas bastante definidos cuando emitimos en la banda de uso libre (**Fig. 47**), más definidos aún que en el caso de 915MHz debido a que esta banda también se utiliza con fines industriales como habíamos comentado anteriormente. Sin embargo, para los canales ocupados (KISS FM y Antena 3 HD) obtenemos un espectro bastante más difuso (**Fig. 48 y 49**), resultando en más errores a la hora de decodificar el mensaje debido a las interferencias con las demás ondas.

Es conveniente comentar que para el caso de los canales libres sólo ilustraremos un caso (1.6631GHz) ya que son similares para toda la banda, por lo que a efectos prácticos toda la información útil se encuentra en estas capturas. Para el caso de los canales ocupados sí hemos visto necesario mostrar ambos casos ya que los diagramas obtenidos no son exactamente iguales.



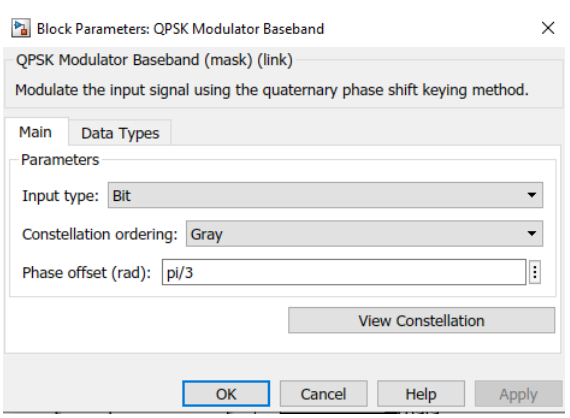
(a) Después del "Carrier Symchronizer" (b) Antes del demodulador QPSK

Figura 48: Diagramas de constelaciones para KISS FM (100.3MHz).

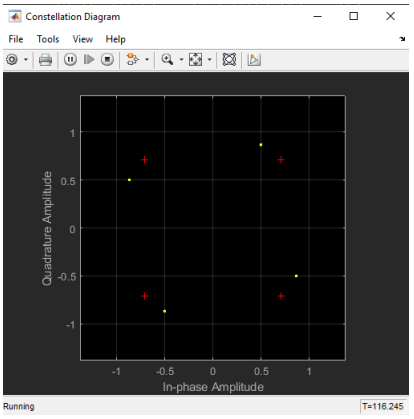


(a) Después del "Carrier Symchronizer" (b) Antes del demodulador QPSK

Figura 49: Diagramas de constelaciones para Antena 3 HD (634MHz).

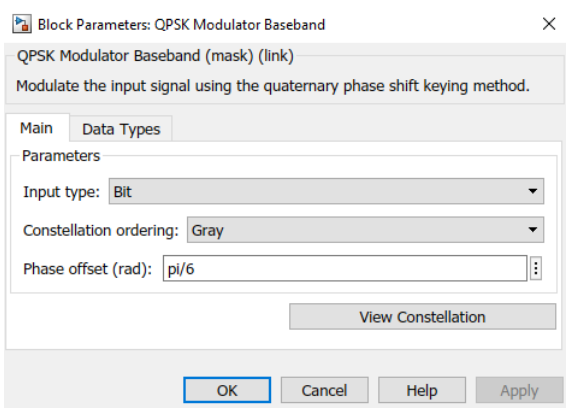


(a) Ventana de diálogo.

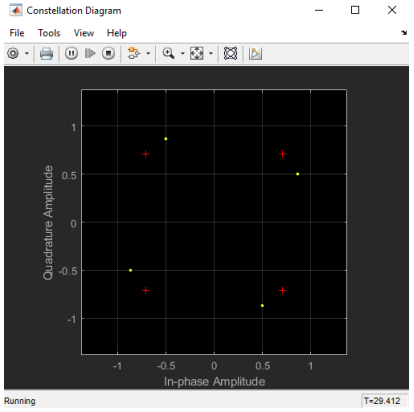


(b) Diagrama de constelación.

Figura 50: Resultados para $\pi/3$.



(a) Ventana de diálogo.

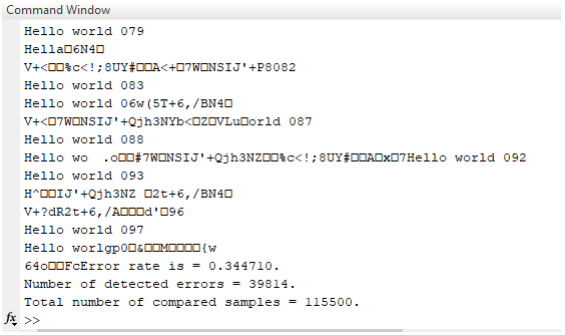


(b) Diagrama de constelación.

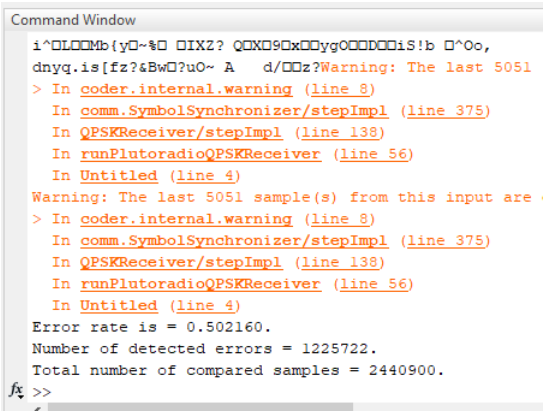
Figura 51: Resultados para $\pi/6$.

Por último hemos decidido ilustrar el efecto que causa la fase asignada a cada combinación de bits en el modulador QPSK, en los diagramas de constelación y en la decodificación del mensaje. En las **Fig. 50 y 51** se muestran los diagramas de constelación obtenidos para las fases $\pi/3$ y $\pi/6$ de forma que obtenemos los mismos cuatro puntos que obteníamos en la **Fig. 37** sólo que girados con el ángulo correspondiente a cada caso.

Esto a su vez también afecta en la decodificación del mensaje, ya que si no adaptamos nuestro receptor a este cambio obtenemos errores significativos. Parte del mensaje podrá decodificarse debido a los fenómenos aleatorios que toman lugar en la transmisión del mensaje, obteneniendo así una nube de puntos que al estar cercanos a las fases de $\pi/4$, $3\pi/4$, $5\pi/4$ y $7\pi/4$; le permite al receptor obtener ciertos dígitos correctamente, aunque no los suficientes como para catalogar el experimento como exitoso. Los resultados (BER) se muestran a continuación.



(a) $\pi/3$



(b) $\pi/6$

Figura 52: Resultados de la decodificación del mensaje (BER).

Como se puede observar en la figura para el caso de $\pi/3$ obtenemos un ratio de errores del 35 %, un porcentaje bastante elevado como para considerar que se ha decodificado correctamente. En el caso de $\pi/6$ se obtiene un porcentaje incluso mayor, del 50 % aproximadamente, el mismo que se asocia al lanzamiento de una moneda al aire por lo que podemos visualizar aquí los fenómenos aleatorios que comentamos anteriormente. Por lo tanto es necesaria la sincronización de ambos sistemas, transmisor y receptor, para que la decodificación del mensaje se haga correctamente ya que aunque estemos recibiendo una señal esta no es la deseada.

Una vez adaptamos nuestro receptor a las fases asignadas en el modulador podemos decodificar correctamente nuestro mensaje, mostrando así que nuestras premisas eran correctas. Para este ejemplo sólo mostraremos una captura ya que ambos casos son similares (ratio de errores del 0 %).

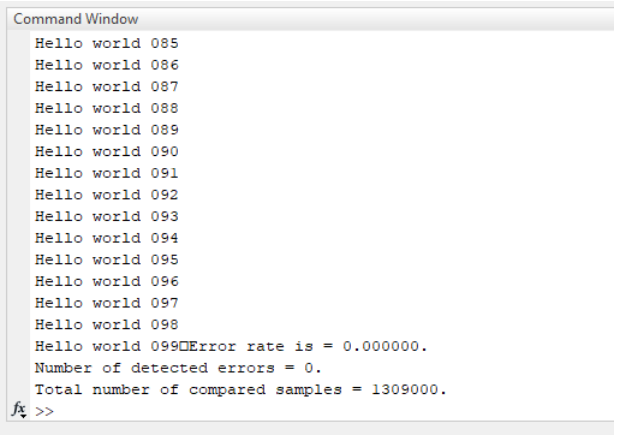


Figura 53: Resultados obtenidos para ambos valores de la fase asignada.

6. Conclusiones

Tras la realización de esta serie de experimentos podemos decir que hemos llevado a cabo los objetivos porpuestos para este proyecto. En primer lugar hemos implementado correctamente nuestro sistema SDR con la ayuda de distintos softwares (Universal Radio Hacker y MATLAB), de forma que nos permite cambiar en tiempo real los valores asignados a los distintos parámetros de transmisión y recepción.

Hemos estudiado la onda que queremos transmitir desde distintos puntos de vista (dominio del tiempo y de la frecuencia, diagramas de constelación, etc.), de forma que podemos compararla con la recibida y estudiar los fenómenos resultantes de la transmisión de este mensaje. Al variar los distintos parámetros posibles en la transmisión tales como la ganancia o la frecuencia central de nuestro bloque transmisor del ADALM-PLUTO, se han visualizado los efectos que causan en la recepción y decodificación del mensaje, confirmando así los problemas planteados en la introducción y a los que pone solución el paradigma de la radio cognitiva, respaldado por los mejores resultados que hemos obtenido en bandas de frecuencia que estaban parcial o completamente desocupadas. También hemos visualizado la importancia de que tanto el transmisor como el receptor estén sincronizados en términos de frecuencia y modulación para que nuestro mensaje se reciba y decodifique correctamente, dando a entender así que no son sistemas independientes y que esos parámetros deben variar simultáneamente en ambos.

Estas conclusiones nos llevan a la aceptación de este paradigma (radio cognitiva) como una mejor forma de implementar nuestros sistemas de comunicaciones y de los problemas planteados por la actual forma de usarlos, por lo que consideramos que este proyecto ha cumplido los objetivos de forma satisfactoria.

A. Apéndice

A.1. Código de inicialización del transmisor

```
% Transmitter parameter structure
prmQPSKTransmitter = plutoradioqpsktransmitter_init;
% Specify Radio ID
prmQPSKTransmitter.Address = 'usb:0'

function SimParams = plutoradioqpsktransmitter_init

%% General simulation parameters

SimParams.Rsym = 0.2e6;             % Symbol rate in Hertz
SimParams.ModulationOrder = 4;      % QPSK alphabet size
SimParams.Interpolation = 2;        % Interpolation factor
SimParams.Decimation = 1;           % Decimation factor
SimParams.Tsym = 1/SimParams.Rsym;  % Symbol time in sec
SimParams.Fs   = SimParams.Rsym * SimParams.Interpolation; % Sample rate

%% Frame Specifications

% [BarkerCode*2 | 'Hello world 001\n' ... | 'Hello world 099\n' |];
% Bipolar Barker Code
SimParams.BarkerCode      = [+1 +1 +1 +1 +1 -1 -1 +1 +1 -1 +1 -1 +1];
SimParams.BarkerLength    = length(SimParams.BarkerCode);
% Duplicate 2 Barker codes to be as a header
SimParams.HeaderLength    = SimParams.BarkerLength * 2;
SimParams.Message         = 'Hello world';
% 'Hello world 000\n'...
SimParams.MessageLength   = length(SimParams.Message) + 5;
% Number of messages in a frame
SimParams.NumberOfMessage = 100;
% 7 bits per characters
SimParams.PayloadLength   = SimParams.NumberOfMessage ...
* SimParams.MessageLength * 7;
```

```
% Frame size in symbols
SimParams.FrameSize       = (SimParams.HeaderLength ...
+ SimParams.PayloadLength) / log2(SimParams.ModulationOrder);
SimParams.FrameTime       = SimParams.Tsym*SimParams.FrameSize;

%% Tx parameters

SimParams.RolloffFactor     = 0.5;   % Rolloff Factor of Raised Cosine Filter
SimParams.ScramblerBase     = 2;
SimParams.ScramblerPolynomial           = [1 1 1 0 1];
SimParams.ScramblerInitialConditions    = [0 0 0 0];
% Filter span of Raised Cosine Tx Rx filters (in symbols)
SimParams.RaisedCosineFilterSpan = 10;

%% Message generation

msgSet = zeros(100 * SimParams.MessageLength, 1);
for msgCnt = 0 : 99
    msgSet(msgCnt * SimParams.MessageLength ...
    + (1 : SimParams.MessageLength)) ...
    = sprintf('%s %03d\n', SimParams.Message, msgCnt);
end

integerToBit = comm.IntegerToBit(7, 'OutputDataType', 'double');
SimParams.MessageBits = integerToBit(msgSet);

% Pluto transmitter parameters

SimParams.PlutoCenterFrequency      = 915e6;
SimParams.PlutoGain                 = 0;
SimParams.PlutoFrontEndSampleRate   = SimParams.Fs;
SimParams.PlutoFrameLength          = SimParams.Interpolation ...
* SimParams.FrameSize;
```

```
% Simulation Parameters

SimParams.FrameTime = SimParams.PlutoFrameLength ...
/SimParams.PlutoFrontEndSampleRate;
SimParams.StopTime  = 1000;
```

A.2. Código de inicialización del receptor

```
printReceivedData = true;    % true if the received data is to be printed

BER = runPlutoradioQPSKReceiver(prmQPSKReceiver, printReceivedData);

fprintf('Error rate is = %f.\n',BER(1));
fprintf('Number of detected errors = %d.\n',BER(2));
fprintf('Total number of compared samples = %d.\n',BER(3));



function SimParams = plutoradioqpskreceiver_init

%% General simulation parameters

SimParams.Rsym = 0.2e6;             % Symbol rate in Hertz
SimParams.ModulationOrder = 4;      % QPSK alphabet size
SimParams.Interpolation = 2;        % Interpolation factor
SimParams.Decimation = 1;           % Decimation factor
SimParams.Tsym = 1/SimParams.Rsym;  % Symbol time in sec
SimParams.Fs   = SimParams.Rsym * SimParams.Interpolation; % Sample rate

%% Frame Specifications

% [BarkerCode*2 | 'Hello world 000\n' ... | 'Hello world 099\n'];
% Bipolar Barker Code
SimParams.BarkerCode      = [+1 +1 +1 +1 +1 -1 -1 +1 +1 -1 +1 -1 +1];
SimParams.BarkerLength    = length(SimParams.BarkerCode);
% Duplicate 2 Barker codes to be as a header
SimParams.HeaderLength    = SimParams.BarkerLength * 2;
SimParams.Message         = 'Hello world';
```

```
% 'Hello world 000\n'...
SimParams.MessageLength   = length(SimParams.Message) + 5;
% Number of messages in a frame
SimParams.NumberOfMessage = 100;
% 7 bits per characters
SimParams.PayloadLength   = SimParams.NumberOfMessage ...
* SimParams.MessageLength * 7;
% Frame size in symbols
SimParams.FrameSize       = (SimParams.HeaderLength ...
+ SimParams.PayloadLength) / log2(SimParams.ModulationOrder);
SimParams.FrameTime       = SimParams.Tsym*SimParams.FrameSize;

%% Rx parameters

% Rolloff Factor of Raised Cosine Filter
SimParams.RolloffFactor     = 0.5;
SimParams.ScramblerBase     = 2;
SimParams.ScramblerPolynomial           = [1 1 1 0 1];
SimParams.ScramblerInitialConditions    = [0 0 0 0];
% Filter span of Raised Cosine Tx Rx filters (in symbols)
SimParams.RaisedCosineFilterSpan = 10;
% AGC desired output power (in watts)
SimParams.DesiredPower                  = 2;
% AGC averaging length
SimParams.AveragingLength               = 50;
% AGC maximum output power gain
SimParams.MaxPowerGain                  = 60;
SimParams.MaximumFrequencyOffset        = 6e3;
% Look into model for details for details of PLL parameter choice.
% Refer equation 7.30 of
% "Digital Communications - A Discrete-Time Approach" by Michael Rice.
K = 1;
A = 1/sqrt(2);
% Normalized loop bandwidth for fine frequency compensation
SimParams.PhaseRecoveryLoopBandwidth    = 0.01;
% Damping Factor for fine frequency compensation
SimParams.PhaseRecoveryDampingFactor    = 1;
```

```
 % Normalized loop bandwidth for timing recovery
SimParams.TimingRecoveryLoopBandwidth   = 0.01;
% Damping Factor for timing recovery
SimParams.TimingRecoveryDampingFactor   = 1;
% K_p for Timing Recovery PLL, determined by 2KA^2*2.7 (for binary PAM),
% QPSK could be treated as two individual binary PAM,
% 2.7 is for raised cosine filter with roll-off factor 0.5
SimParams.TimingErrorDetectorGain       = 2.7*2*K*A^2+2.7*2*K*A^2;
SimParams.PreambleDetectorThreshold     = 0.8;

%% Message generation and BER calculation parameters

msgSet = zeros(100 * SimParams.MessageLength, 1);
for msgCnt = 0 : 99
    msgSet(msgCnt * SimParams.MessageLength + ...
    (1 : SimParams.MessageLength)) = ...
    sprintf('%s %03d\n', SimParams.Message, msgCnt);
end
integerToBit = comm.IntegerToBit(7, 'OutputDataType', 'double');
SimParams.MessageBits = integerToBit(msgSet);

% For BER calculation masks

SimParams.BerMask = zeros(SimParams.NumberOfMessage ...
* length(SimParams.Message) * 7, 1);
for i = 1 : SimParams.NumberOfMessage
    SimParams.BerMask( (i-1) * length(SimParams.Message) * 7 ...
    + ( 1: length(SimParams.Message) * 7) ) ...
    = (i-1) * SimParams.MessageLength * 7 + (1: length(SimParams.Message) * 7);
end

% Pluto receiver parameters

SimParams.PlutoCenterFrequency      = 915e6;
SimParams.PlutoGain                 = 50;
SimParams.PlutoFrontEndSampleRate   = SimParams.Fs;
SimParams.PlutoFrameLength          = SimParams.Interpolation ...
```

```
* SimParams.FrameSize;

% Experiment parameters

SimParams.PlutoFrameTime = SimParams.PlutoFrameLength ...
/ SimParams.PlutoFrontEndSampleRate;
SimParams.StopTime = 10;
```

Bibliografía

Mitola J. (8 de Mayo del 2000) *Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio*. Stockholm, Sweden: Royal Institute of Technology (KTH). ISSN 1403 - 5286.

Frenzel L. (2016). *Fundamentals of Frequency Modulation. Principles of electronic communication systems (4 edición)*. Nueva York, EE.UU.: McGraw Hill. p. 159. ISBN 978-0-07-337385-0.

Tomasi W. (2003). *Transmisión por modulación angular.* En Guillermo Trujano Mendoza, ed. *Sistemas de Comunicaciones Electrónicas.* Naucalpan de Juárez, Edo. de México, México: Pearson Educación de México, S.A. de C.V. pp. 228-237. ISBN 970-26-0316-1.

Perrins E., Schober R., Rice M., Simon M. (junio de 2006). *Shaped-Offset QPSK with Multiple-Bit Differential Detection [QPSK de compensación continua con detección diferencial de varios bits.]* 3. pp. 1212 - 1218. ISBN 1-4244-0355-3.

Borwein P. Mossinghoff M. J. (2008). *Barker sequences and flat polynomials.* James McKee; Chris Smyth (eds.). *Number Theory and Polynomials.* LMS Lecture Notes. 352. Cambridge University Press. pp. 71–88. ISBN 978-0-521-71467-9.

Subdirección General de Planificación y Gestión del Espectro Radioeléctrico. Secretaría de Estado de Telecomunicaciones e Infraestructuras Digitales. Ministerio de Asuntos Económicos y Transformación Digital. (Abril, 2011). *Bandas y canalizaciones disponibles en el Servicio Fijo de banda ancha.* https://avancedigital.mineco.gob.es/espectro/Documents/SF-Banda-Ancha/SF-banda-ancha.pdf

Gonzalez S., Montes M. (Junio de 2009). *Medios de comunicación: Modulación de una onda.* http://recursostic.educacion.es/eda/web/eda2009/newton/galicia/materiales/gonzalez_esantiago_p3/EGMiramontes_p3_v3/tecnologias_comunicacion/Ondas.html

Zapata M. Recuperado el 28 de Octubre del 2021. *Modulación QAM.* https://soporte.syscom.mx/es/articles/2334170-modulacion-qam

Universitat de València. (s.f.). *Sistemas de Telecomunicación: Modulación Digital.* https://www.uv.es/~hertz/hertz/Docencia/teoria/Trasmdigital.pdf

Lupera P., Solano N. (10 de Noviembre del 2019). *Diseño e implementación de un demodulador QPSK utilizando una técnica de tendencia central.* http://scielo.senescyt.gob.ec/scielo.php?script=sci_arttext&pid=S1390-67122019000200051

Progamador Clic. (s.f.) *Universal Radio Hacker.* https://programmerclick.com/article/55471064267/

The Weekly Geekly. (s.f.). *Universal Radio Hacker: una forma fácil de explorar los protocolos de radio digital.* https://weekly-geekly-es.imtqy.com/articles/es434634/index.html

Phl J., Adamski K. (4 de Octubre del 2021). *Universal Radio Hacker: Investigate Wireless Protocols Like a Boss.* https://github.com/jopohl/urh#Installation

Demartino C. (s.f.). *The Differences Between Transmitter Types.* https://www.mwrf.com/technologies/systems/article/21848235/the-differences-between-transmitter-types-part-2

Arif Sobhan M., Islam Badal T. (Enero de 2019). *Design Architectures of the 2.4GHz CMOS Transmitters for RF Devices.* https://www.researchgate.net/publication/330061993_CMOS_Transmitters_for_24-GHz_RF_Devices_Design_Architectures_of_the_24-GHz_CMOS_Transmitter_for_RF_Devices

Huidobro J. M. (20 de Abril de 2011). *Radio Cognitiva. La radio se vuelve inteligente.* https://www2.coitt.es/res/revistas/04d%20Radio%20cognitiva.pdf

Weisstein E. (Septiembre de 2018). *Barker Code.* https://mathworld.wolfram.com/BarkerCode.html

MathWorks®. (s.f.). *MATLAB.* https://es.mathworks.com/products/matlab.html

MathWorks®. (s.f.). *Generate bipolar Barker Code.* https://es.mathworks.com/help/comm/ref/comm.barkercode-system-object.html

MathWorks®. (s.f.). *Scramble input signal.* https://es.mathworks.com/help/comm/ref/scrambler.html

MathWorks®. (s.f.). *QPSK Demodulator Baseband.* https://es.mathworks.com/help/comm/ref/qpskdemodulatorbaseband.html

MathWorks®. (s.f.). *QPSK Modulator Baseband.* https://es.mathworks.com/help/comm/ref/qpskmodulatorbaseband.html

MathWorks®. (s.f.). *Raised Cosine Filtering.* https://es.mathworks.com/help/comm/ug/raised-cosine-filtering.html#:~:text=A%20typical%20use%20of%20raised%20cosine%20filtering%20is,raised%20cosine%20filter%2C%20which%20results%20in%20minimum%20ISI.

MathWorks®. (s.f.). *Raised Cosine Transmit Filter.* https://es.mathworks.com/help/comm/ref/raisedcosinetransmitfilter.html

MathWorks®. (s.f.). *Raised Cosine Receive Filter.* https://es.mathworks.com/help/comm/ref/raisedcosinereceivefilter.html

MathWorks®. (s.f.). *Pluto Transmitter.* https://es.mathworks.com/help//supportpkg/plutoradio/ref/plutotransmitter.html

MathWorks®. (s.f.). *Pluto Receiver.* https://es.mathworks.com/help/supportpkg/plutoradio/ref/plutoreceiver.html

MathWorks®. (s.f.). *QPSK Transmitter with ADALM-PLUTO Radio.* https://es.mathworks.com/help/supportpkg/plutoradio/ug/qpsk-transmitter-with-adalm-pluto-radio.html

MathWorks®. (s.f.). *QPSK Receiver with ADALM-PLUTO Radio.* https://es.mathworks.com/help/supportpkg/plutoradio/ug/qpsk-receiver-with-adalm-pluto-radio.html