



Universidad Nacional de Mar del Plata

FACULTAD DE INGENIERÍA

**SISTEMAS DE COMUNICACIONES
IMPLEMENTADOS EN GNU-RADIO Y ADALM
PLUTO SDR**

Comunicaciones digitales

Autores:
Trigo, Nicolás Daniel
Vazquez, Leonardo David

Fecha de entrega: 6/12/2021

Índice

1. Introducción	2
1.1. GNU-Radio	2
1.2. Adalm Pluto SDR	3
1.2.1. Características principales	4
1.2.2. Conexión de la placa a una PC	5
2. Radioenlace Simplex	7
2.1. Esquema de comunicación	7
2.2. Radioenlace implementado con SDR	8
3. Comunicación mediante modulación QPSK	10
3.1. Fundamentos teóricos	10
3.2. Simulación en GNU-Radio	11
3.3. Implementación con SDR	17
4. Conclusiones	20

Resumen

En el presente trabajo se propone implementar sistemas de comunicaciones en el software GNU Radio a través de la simulación y de la experimentación mediante la utilización de la placa SDR Adalm Pluto. En primer lugar, se realiza un radioenlace simplex al transmitir y recibir un tono de frecuencia determinada. Luego, se pretende transmitir y recibir una imagen mediante modulación QPSK.

1. Introducción

1.1. GNU-Radio

Las comunicaciones digitales cada día están más inmersas en el mundo de la tecnología, tan así que es un campo muy estudiado en la actualidad. Con el avance del tiempo, se van diseñando programas que facilitan la tarea de simular sistemas de comunicaciones de manera sencilla, tal como el denominado software GNU-Radio [1].

GNU-Radio es un software de desarrollo libre para el usuario en modo de bloques de procesamiento de señales, cuya base es la programación en *Python* y en *C++*. La ventaja de este programa es proveer una herramienta a gente común para que puedan desarrollar la habilidad de entender a nivel técnico y en profundidad el espectro electromagnético de las comunicaciones. En la figura 1 se puede observar el entorno del GNU-Radio (versión 3.7).

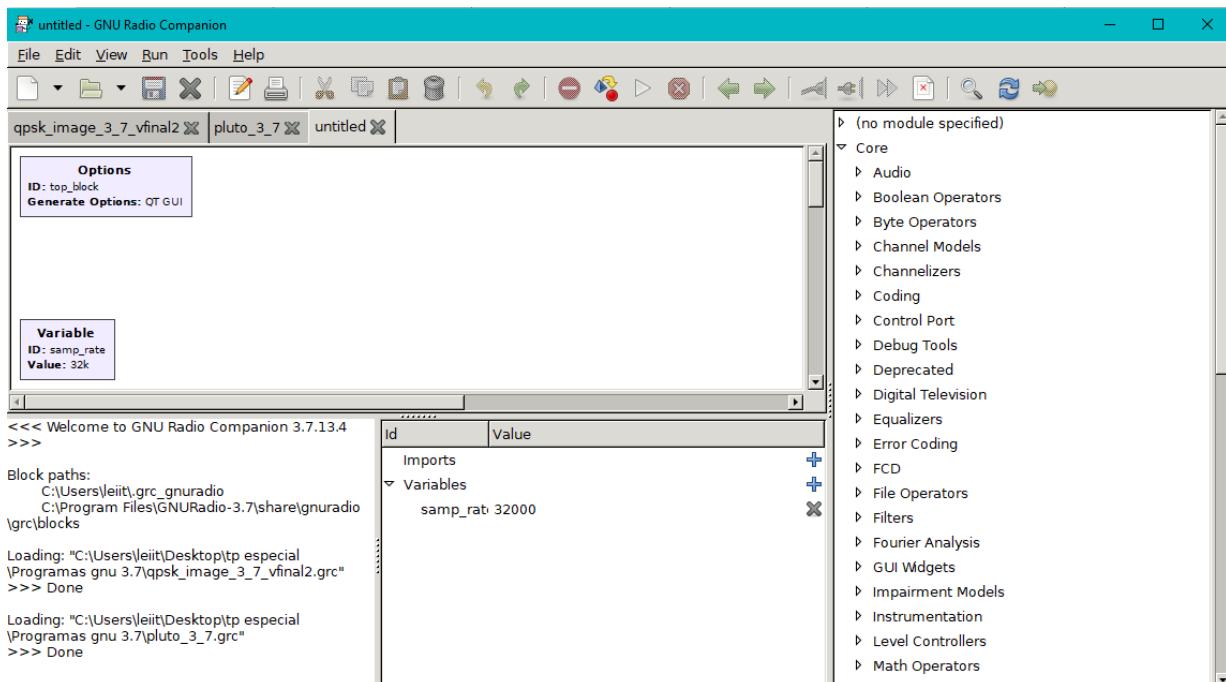


Figura 1: Entorno del software GNU-Radio (versión 3.7).

Es muy importante nombrar con qué tipos de formatos de información trabaja GNU-Radio, como bien se observa en la figura siguiente:

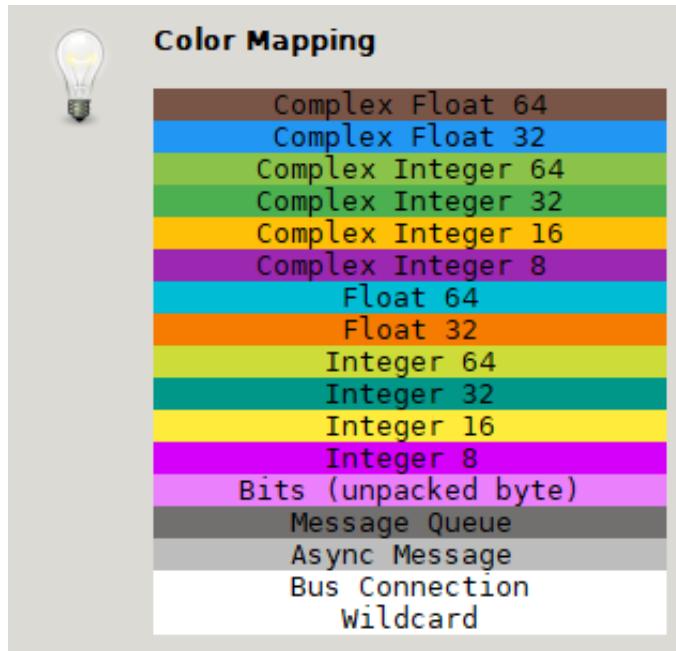


Figura 2: Tipos de datos que utiliza GNU-Radio (versión 3.7).

1.2. Adalm Pluto SDR

La radio definida por software (SDR) es un sistema de comunicación por radio en el que los componentes que tradicionalmente se han implementado en hardware (por ejemplo, mezcladores, filtros, amplificadores, moduladores/demoduladores, detectores, etc.) son implementados por medio de software en un ordenador personal o en un sistema integrado, es decir, un sistema embebido.

Adalm Pluto [2] es un dispositivo SDR que cuenta con un sistema embebido Linux y dos antenas: de transmisión y de recepción. Además, incluye una interfaz Ethernet y una interfaz USB que le permite conectarse con computadoras personales e interactuar con softwares como GNU-Radio. En la figura 3 se observa el dispositivo en cuestión.

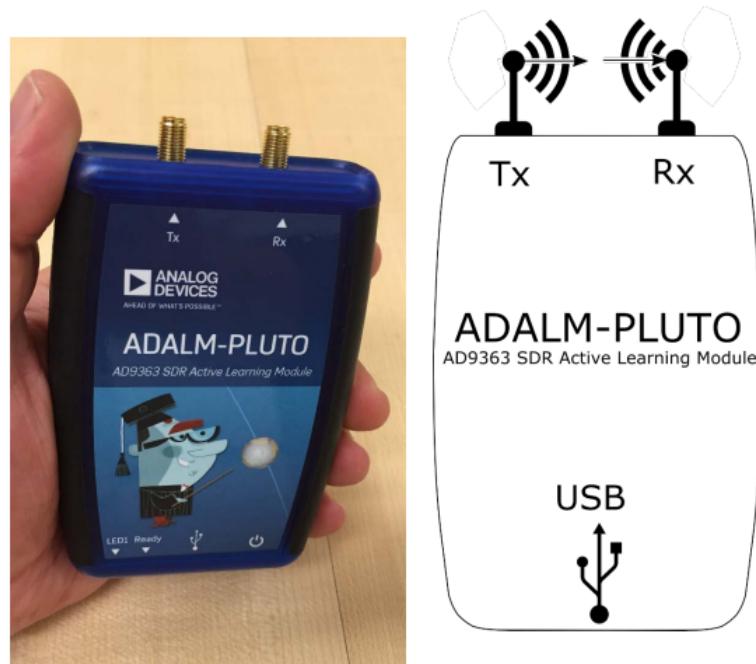


Figura 3: Dispositivo Adalm Pluto SDR.

1.2.1. Características principales

Un diagrama en bloques del dispositivo puede verse en la figura 4. En ella se notan la placa FPGA (donde está el sistema operativo), las interfaces anteriormente mencionadas, los dispositivos de potencia y RF, las antenas y las memorias.

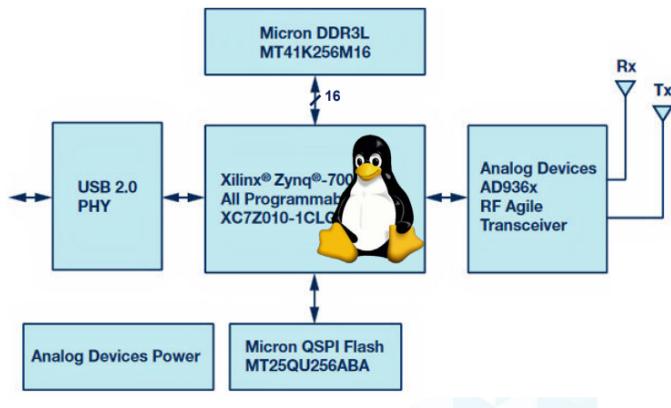


Figura 4: Diagrama en bloques del dispositivo Adalm Pluto SDR.

Entre sus características principales se encuentran:

- Captura de muestras en fase y cuadratura I/Q de 12 bits
- Velocidad de muestras entre $65.1kSps$ a $61.44MSps$
- Ancho de banda de señal entre $200kHz$ y $20MHz$
- Rango de sintonía de entre $325MHz$ y $3.8GHz$

El transceptor utilizado por Adalm Pluto es el *AD9361* [3]. En la figura 5 se observa un modelo en bloques del transceptor integrado, el cual se distinguen la sección transmisora y la sección receptora.

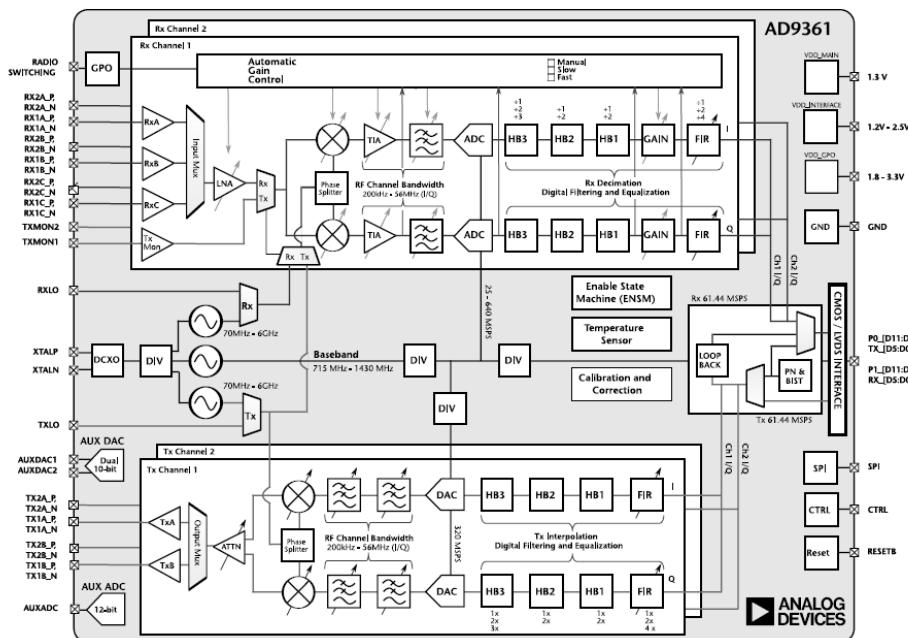


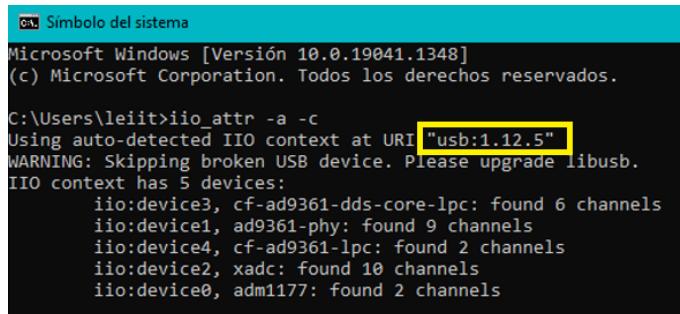
Figura 5: Diagrama en bloques del transceptor AD9361.

1.2.2. Conexión de la placa a una PC

Para que la placa Adalm Pluto pueda conectarse a una PC con sistema operativo Windows y luego a un software como GNU-Radio, se debe seguir una serie de pasos sencillos ([4] y [5]):

1. Instalar Drivers de Adalm Pluto para un sistema operativo Windows:
<https://wiki.analog.com/university/tools/pluto/drivers/windows>
2. Instalar GNU-Radio con librería IIO incluida:
<https://wiki.analog.com/resources/tools-software/linux-software/gnuradio>
3. Instalar librería IIO en Windows:
<https://github.com/analogdevicesinc/libiio/releases/tag/v0.23>

4. Conectar el Adam Pluto mediante USB a la Pc. Luego, abrir consola de Windows y escribir lo que se indica en la figura 6 para encontrar la ubicación del dispositivo



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19041.1348]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\leiit>iio_attr -a -c
Using auto-detected IIO context at URI "usb:1.12.5"
WARNING: Skipping broken USB device. Please upgrade libusb.
IIO context has 5 devices:
    iio:device3, cf-ad9361-dds-core-lpc: found 6 channels
    iio:device1, ad9361-phy: found 9 channels
    iio:device4, cf-ad9361-lpc: found 2 channels
    iio:device2, xadc: found 10 channels
    iio:device0, adm1177: found 2 channels
```

Figura 6: Localización virtual del dispositivo Adam Pluto.

5. Abrir GNU-Radio. Las conexiones se realizan mediante los bloques "Pluto SDR Sink" (transmisión) y "Pluto SDR Source" (Recepción). Se observa en la figura 7 que utilizan una elevada velocidad de muestras, portadora en $2.4GHz$, Ancho de banda de $20MHz$, atenuación $1dB$, ganancia $15dB$ y método de retransmisión Cycle. El parámetro obtenido en el punto anterior se coloca en "Device Uri".

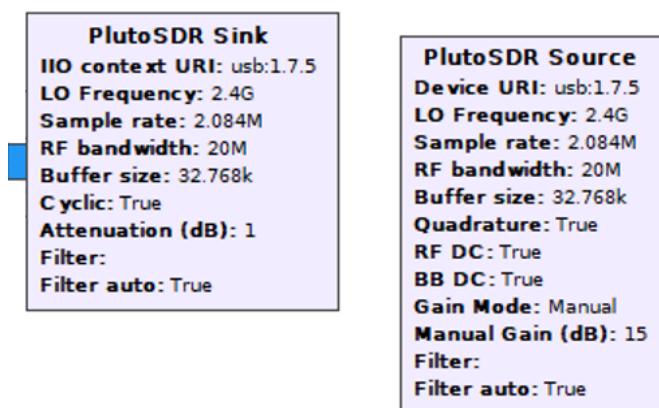


Figura 7: Conexión de Adam Pluto con GNU-Radio.

2. Radioenlace Simplex

2.1. Esquema de comunicación

Un esquema de comunicación sencillo para transmitir y recibir una señal puede observarse en la figura 8, en donde hay un transmisor quien se encarga de enviar el mensaje, un canal por donde este último viaja, y el receptor quien recibe la información.



Figura 8: Esquema de Comunicación Simplex

Por otro lado, un Radioenlace es un sistema de comunicación inalámbrica que permite la comunicación y el envío de información entre dos puntos, es decir, que implementa el esquema de comunicación de la figura anterior, pero en donde el canal es el aire y en donde se utilizan antenas tanto para la transmisión como para la recepción. Un ejemplo de aplicación de este tipo de sistemas se encuentra en las comunicaciones satelitales (figura 9). El satélite envía datos meteorológicos e incluso imágenes mediante una antena transmisora en cierta banda de frecuencias de RF para que la antena receptora en una base terrestre reciba la información y la procese.

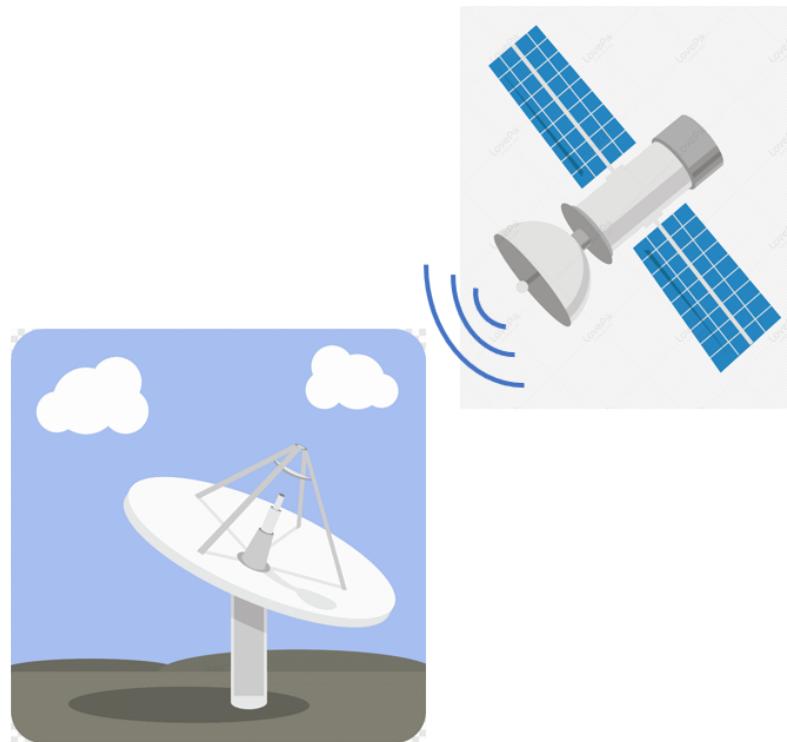


Figura 9: Radioenlace satelital

2.2. Radioenlace implementado con SDR

Se desea entonces implementar un Radioenlace mediante GNU-Radio y el Adalm Pluto SDR. El esquema de comunicación es sencillo: se transmite un tono de frecuencia $10kHz$ mediante la antena T_x del SDR y se recibe la información mediante la antena R_x , en el cual, se espera recibir el mismo tono transmitido. En la figura 10 se muestra el sistema en GNU-Radio.

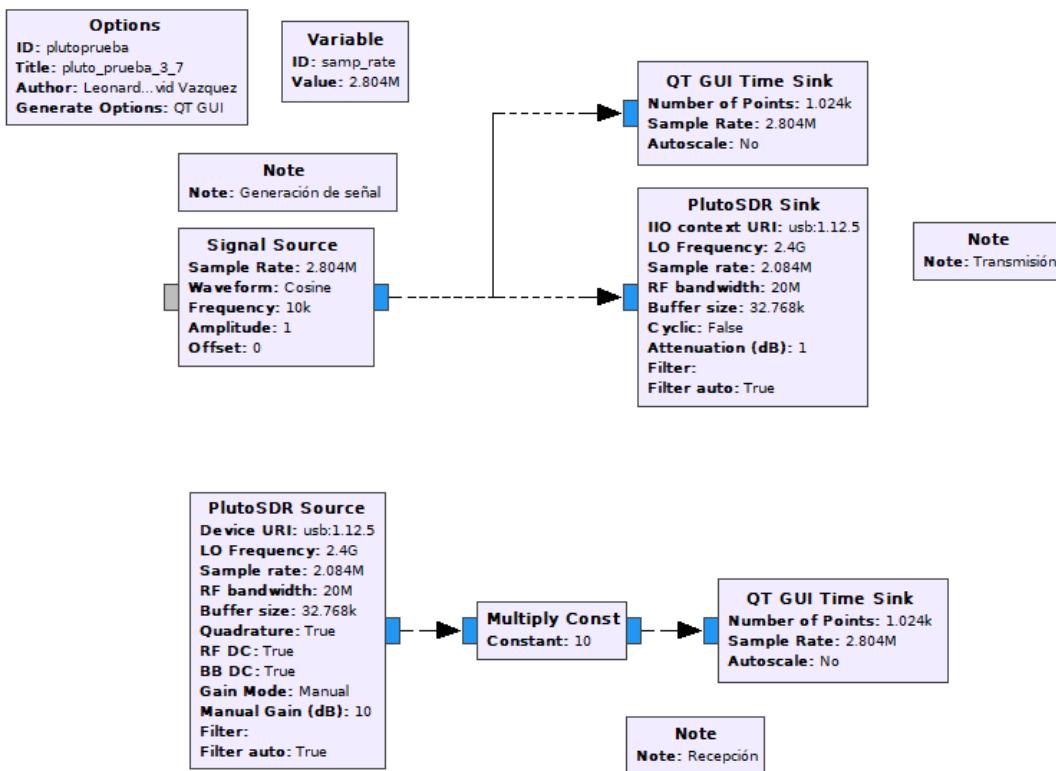


Figura 10: Radioenlace propuesto

El bloque **Signal Source** se encarga de generar un tono digital y complejo de frecuencia $10kHz$. El bloque **PlutoSDR Sink** envía la señal digital mediante una conexión USB al Adalm Pluto SDR con una velocidad de muestras de $2.804MSps$. Se desconoce el tipo de modulación que utiliza la placa pero sí se conocen los parámetros típicos de la transmisión: portadora en $3.2GHz$, ancho de banda $20MHz$, atenuación $1dB$, entre otros.

Para la recepción se utiliza el bloque **PlutoSDR Source** que es el encargado de recibir los datos procesados desde la antena y mediante la conexión USB. Los parámetros que se utilizan son los mismos que para la recepción con algunos agregados: Demodulación por cuadratura, ganancia, entre otros. Luego se utiliza como opcional un bloque de multiplicación constante **Multiply Const** para elegir una amplitud determinada en la salida. Los bloques **QT GUI Time Sink** se utilizan para visualizar temporalmente las señales transmitida y recibida.

En la figura 11 se ve el banco experimental: el Adalm Pluto SDR conectado mediante USB a una computadora portatil con sistema operativo Windows.



Figura 11: Banco experimental

Una vez transmitida la señal, se visualiza el tono recibido y transmitido, como se observa en la figura 12. Al tratarse de tonos complejos, ambas señales poseen parte imaginaria y parte real. Como era de esperarse, la señal recibida se ve un poco distorsionada debido al ruido del canal y, quizás también, debido a ruidos de cuantificación y de decodificación de la señal, ya que, recordemos que el transceptor tiene bloques tanto DAC como ADC.

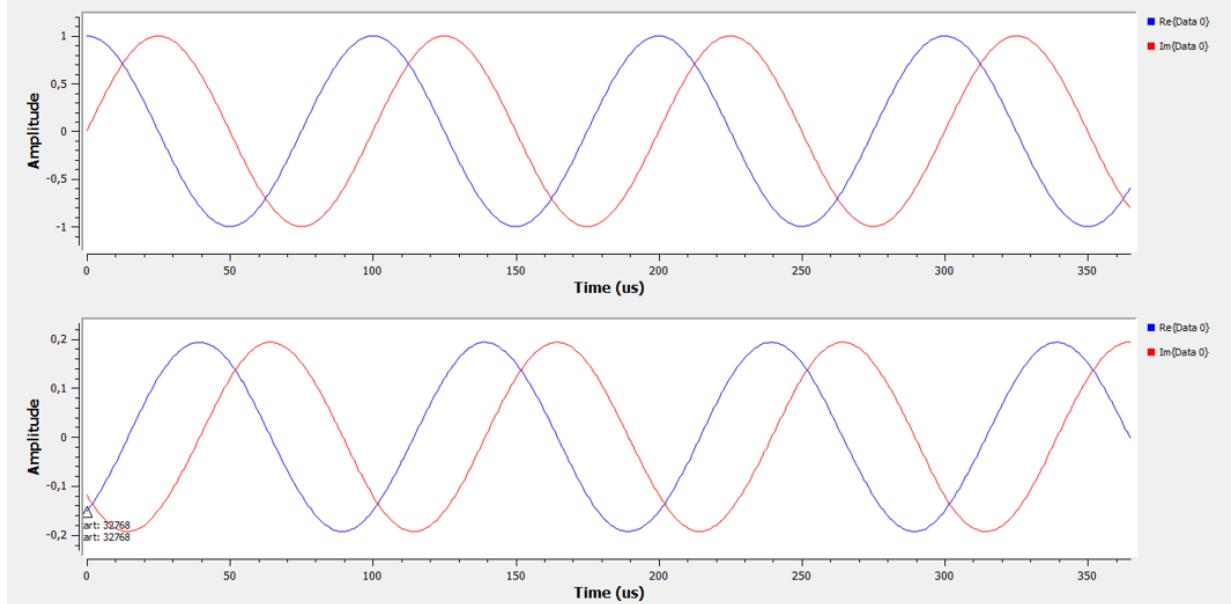


Figura 12: Visualización temporal en GNU-Radio de las señales transmitida (superior) y recibida (inferior).

Puede estimarse también el retardo de transmisión, el cual es $\tau = 40\mu s$.

3. Comunicación mediante modulación QPSK

3.1. Fundamentos teóricos

La Transmisión digital se basa en mensajes que son símbolos ordenados secuencialmente y producidos por una fuente de información discreta [6]. Esta fuente se basa a su vez en un alfabeto de M símbolos con una velocidad promedio r .

La modulación por desplazamiento de fase (PSK) [7] es un esquema de modulación digital que transmite datos cambiando o modulando la fase de una señal de referencia (es decir, la onda portadora). PSK utiliza un número finito de fases, cada una de las cuales tiene asignado un patrón único de dígitos binarios. Normalmente, cada fase codifica un número igual de bits. Cada patrón de bits forma el símbolo representado por la fase en cuestión. La forma de onda de modulación PSK puede ser representada en sus formas de componentes en fase y cuadratura:

$$x_c(t) = s_i(t - kD) - s_q(t - kD) \quad (1)$$

Donde:

$$s_i(t - kD) = \sum_k \cos \varphi_k \cdot p_D(t - kD) = \sum_k I_k \cdot p_D(t - kD) \quad (2)$$

$$s_q(t - kD) = \sum_k \sin \varphi_k \cdot p_D(t - kD) = \sum_k Q_k \cdot p_D(t - kD) \quad (3)$$

En el cual:

$$I_k = \cos \varphi_k \quad (4)$$

$$Q_k = \sin \varphi_k \quad (5)$$

Los valores a_k de la forma de onda M-aria en modo unipolar, se corresponden con las diferentes fases φ_k :

$$\varphi_k = \frac{2\pi a_k}{M} \quad a_k = 0, 1, 2, \dots, M-1 \quad (6)$$

Como caso particular, y como se define en la bibliografía citada [7] y en el Software GNU-Radio [1], en la modulación QPSK $M = 4$ y se describen a las diferentes fases como:

$$\varphi_k = \frac{2\pi(a_k + \frac{1}{2})}{4} \quad a_k = 0, 1, 2, 3 \quad (7)$$

Además, para normalizar a I_k y a Q_k

$$I_k = \cos \varphi_k \cdot \frac{2}{\sqrt{2}} \quad (8)$$

$$Q_k = \sin \varphi_k \cdot \frac{2}{\sqrt{2}} \quad (9)$$

Obteniendo así:

$$\varphi_k = \frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4} \quad (10)$$

$$I_k = 1, -1, -1, 1 \quad (11)$$

$$Q_k = 1, 1, -1, -1 \quad (12)$$

La constelación QPSK puede observarse en la figura 13:

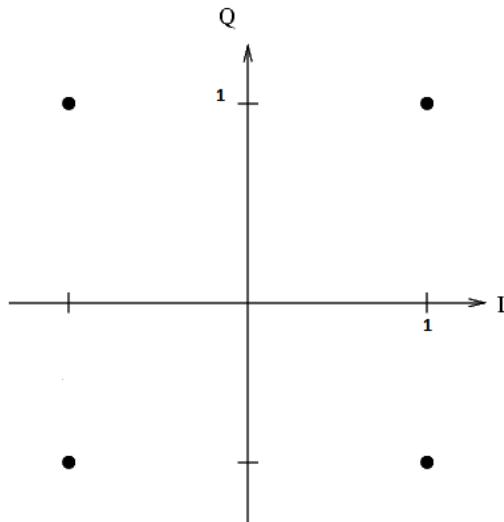


Figura 13: Constelación QPSK

3.2. Simulación en GNU-Radio

Se propone entonces implementar el siguiente esquema de comunicación (figura 14) en el cual, se desea transmitir y recibir una imagen con formato .jpg codificada y modulada en QPSK.

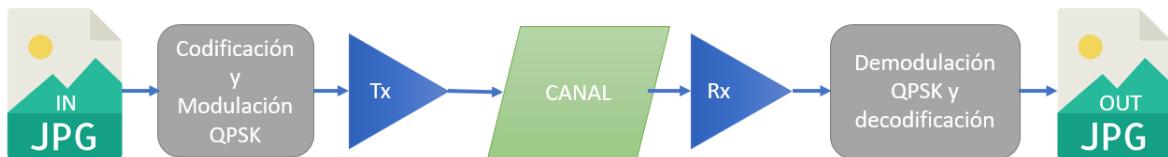


Figura 14: Esquema de comunicación propuesto

El esquema de la figura anterior se basa en transmitir una imagen con formato jpg: Primero se levanta la imagen y se la codifica. Luego, se modula en QPSK para obtener una constelación a transmitir por el canal. El receptor recibe dicha constelación para posteriormente demodularla y decodificarla. Este decodificador también se encarga de reconstruir la información para entregar una imagen con el mismo formato.

Se implementa entonces el esquema propuesto en el Software GNU-Radio como se observa en la figura 15.

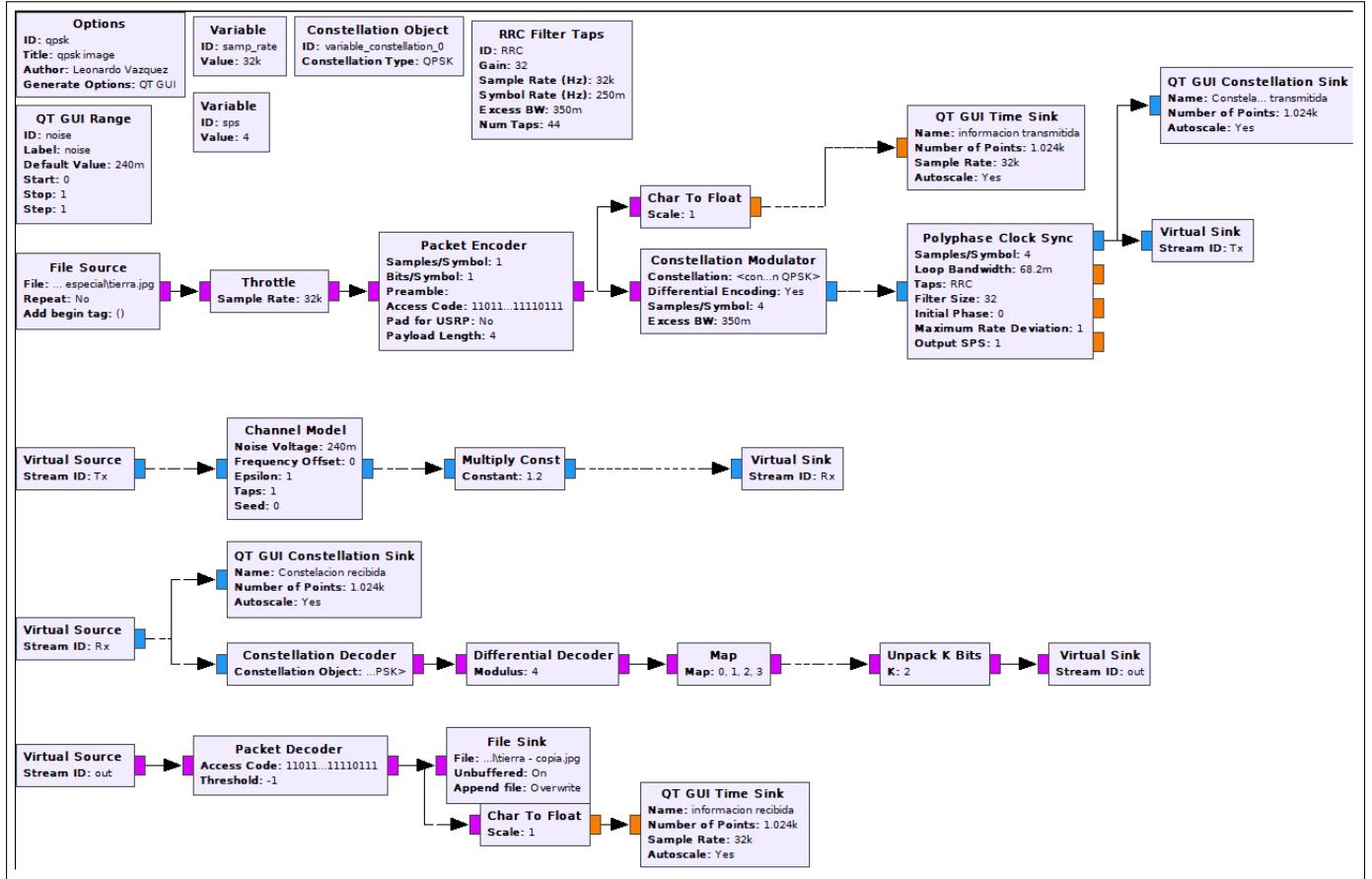


Figura 15: Esquema de comunicación propuesto implementado en GNU-Radio

El bloque **File Source** se encarga de levantar la imagen original que se desea transmitir. Luego, se limita la velocidad de muestreo en bytes con **Throttle**, a unos $32kSps$. El bloque **Packet Encoder** genera tramas de bits permitiendo que la información se organice de manera adecuada y contribuye a la reducción de errores tanto en la transmisión como en la recepción. Utiliza un código de acceso y un estándar CRC32 para verificar la información.



Figura 16: Estructura de una trama codificada por el bloque Packet Encoder

Una vez codificada la información, se modula en QPSK con 4 muestras por símbolo mediante el bloque **Constellation Modulator**. Luego, para lograr una sincronización en la constelación se utiliza el bloque **Polyphase Clock Sync** que se basa en un banco de filtros digitales. En la figura siguiente se observa el efecto de incorporar estos bloques:

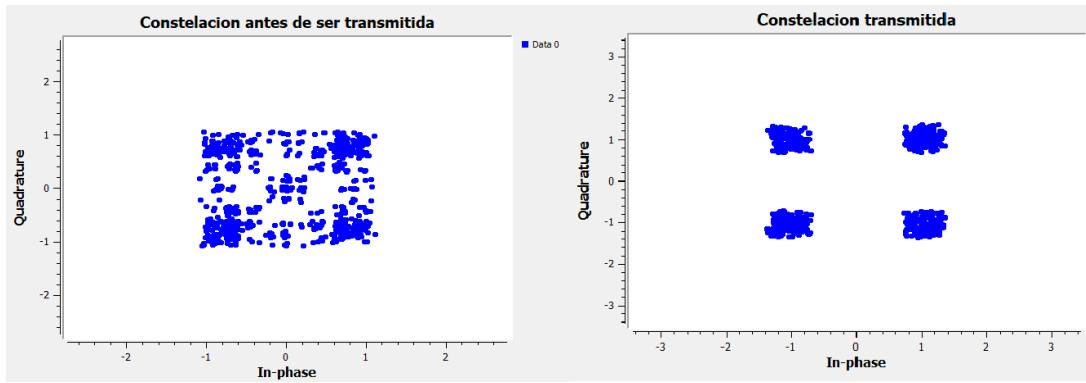


Figura 17: Efecto de la Sincronización

El paso siguiente es simular el Canal. Con la ayuda de **Virtual Source** y **Virtual Sink** sepáramos las etapas y utilizamos el bloque **Channel Model** para simular la adición de ruido Gaussiano. En la figura siguiente se observa el efecto del ruido al notar como los puntos de la constelación se van dispersando conforme aumenta el ruido.

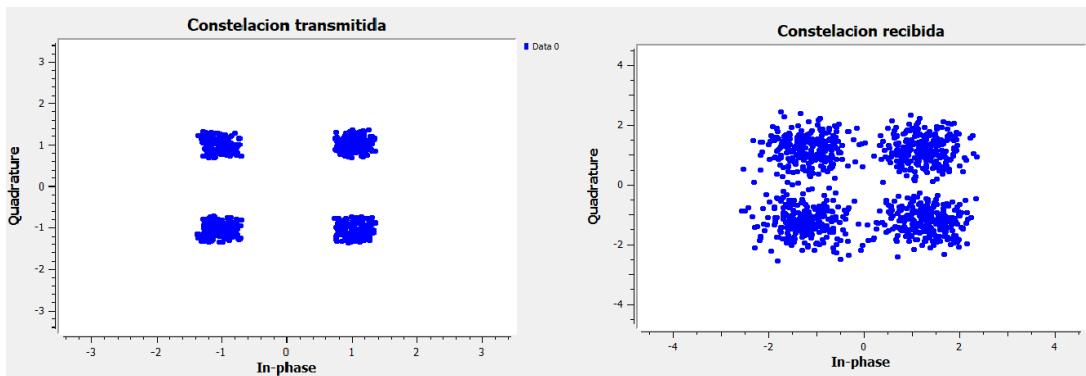


Figura 18: Efecto del Ruido

Ya en la etapa de la recepción, se utiliza el bloque **Constellation Decoder**, que decodifica los puntos del espacio complejo en bits basados en el mapeo que se haya hecho, es decir, QPSK. Luego, se recurre a **Map** y a **Diferencial Decoder** para traducir los símbolos obtenidos de manera diferencial (diferencias de fase) a sus valores absolutos. Por último, se desempaquetan los símbolos con el bloque **Unpack K bits** y se recurre al bloque **Packet Decoder** que hace lo contrario al **Packet Encoder**, es decir, decodifica toda la información basándose en el código de acceso y el estándar CRC32 para verificar la información. En la figura 19 se observa, luego de una conversión con el bloque **Char to Float**, la señal transmitida y recibida.

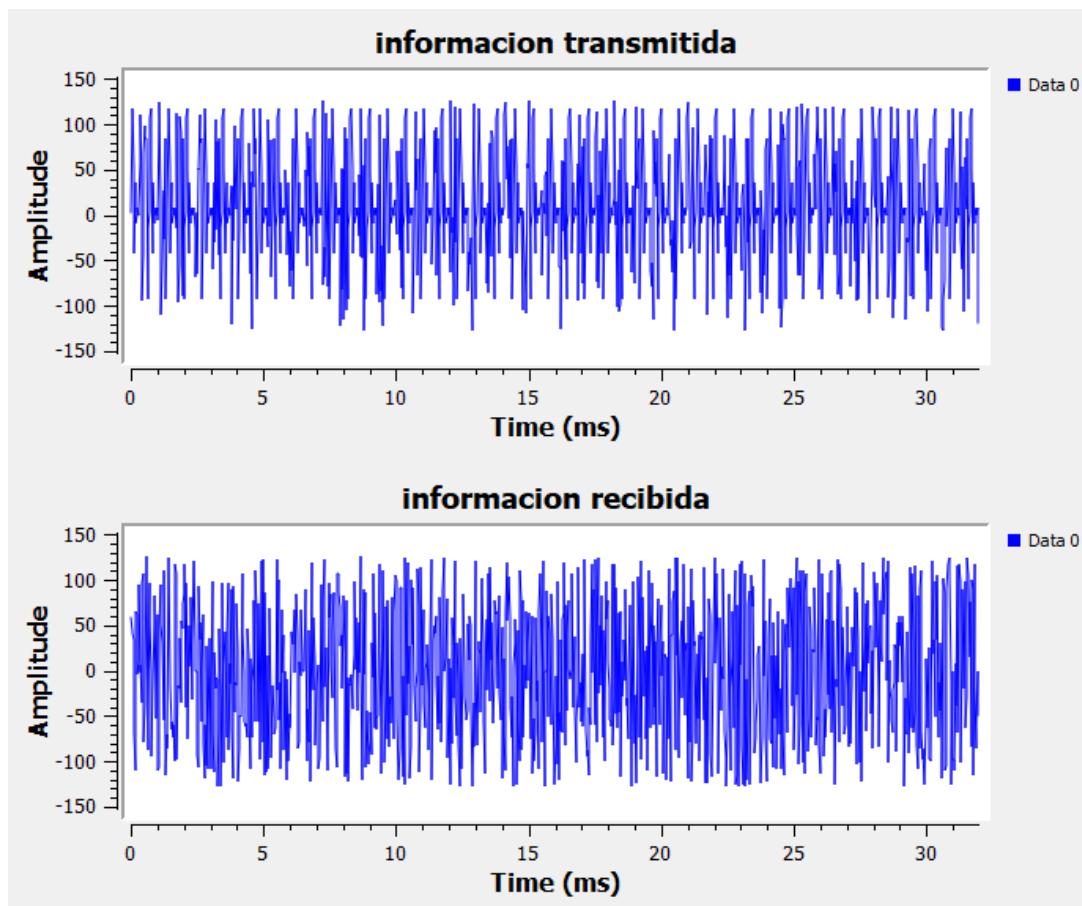


Figura 19: Información recibida y transmitida

A modo de ejemplo, se muestran algunas imágenes transmitidas y recibidas, tanto con ruido como sin ruido.



Figura 20: Imagen recibida y transmitida. Sin ruido

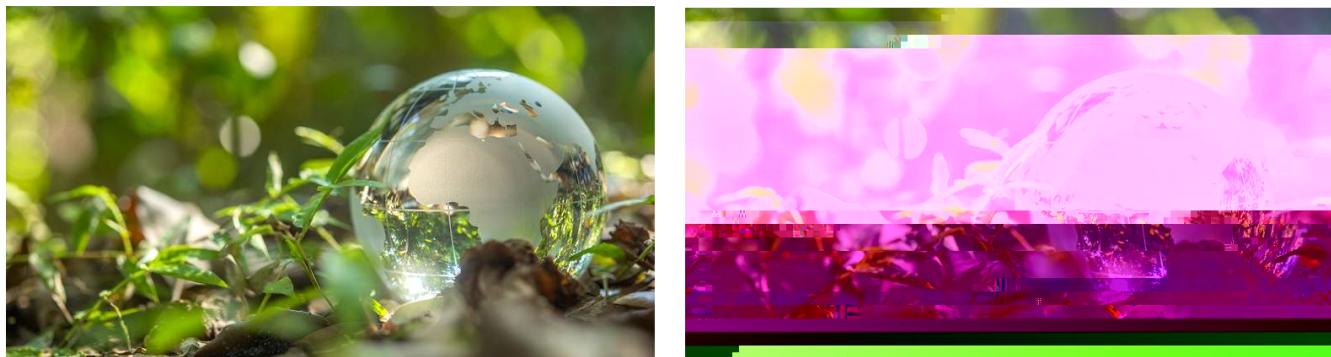


Figura 21: Imagen recibida y transmitida. Con ruido



Figura 22: Imagen recibida y transmitida. Sin ruido. En este ejemplo se observa un error en la imagen que seguramente se debió a un error de decodificación.



Figura 23: Imagen recibida y transmitida. Con ruido

No obstante, en muchos casos en donde el canal posee ruido, la decodificación es eficaz debido a la corrección y detección de errores al empaquetar la información, obteniendo así la imagen deseada. La siguiente tabla muestra el efecto del ruido en la pérdida de bytes de la imagen original:

Ruido [mV]	Tamaño transmitido en bytes	Tamaño recibido en bytes	Diferencia en bytes
0	40728	40724	4
100	40728	40724	4
200	40728	40724	4
300	40728	40648	80
350	40728	40248	480
400	40728	38728	2000
450	40728	35216	5512
500	40728	29256	11472
550	40728	21364	19364
600	40728	13660	27068
650	40728	7720	33008
700	40728	3700	37028
750	40728	1568	39160

Tabla 1: Pérdida de la información en función del ruido.

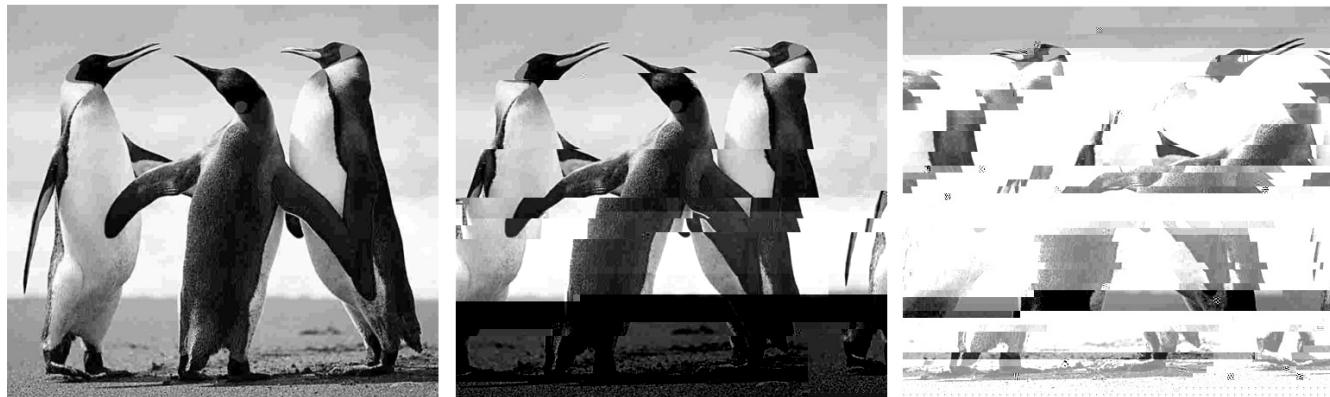


Figura 24: Efecto del incremento en el ruido del canal

3.3. Implementación con SDR

Se propone implementar con el Adalm Pluto SDR el esquema de la figura 14 y el diagrama en bloques en GNU-Radio de la figura 15. Para ello, se modifica tal diagrama para realizar la conexión con la placa SDR.

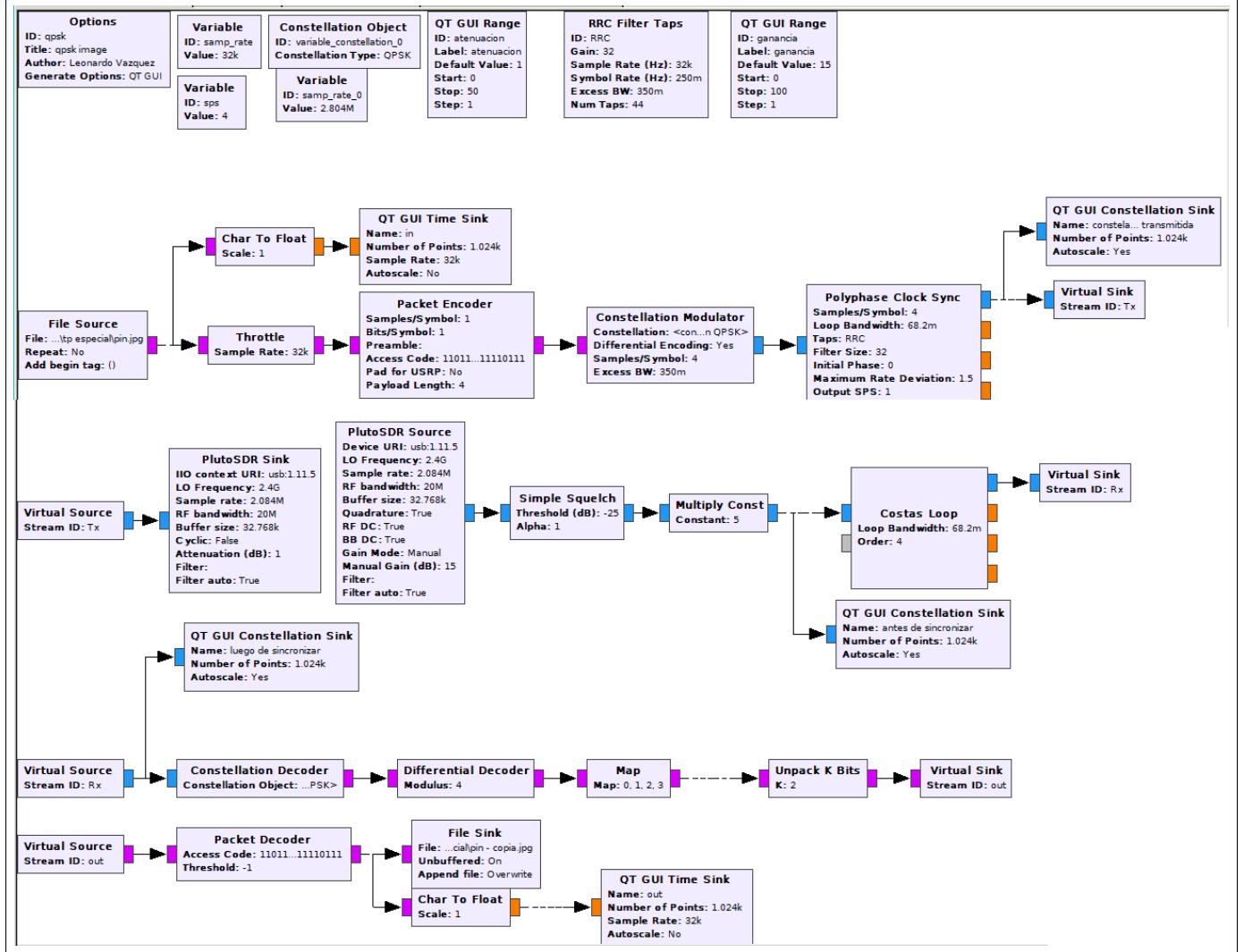


Figura 25: Diagrama en bloques para realizar la conexión con el SDR

Se puede observar que los principales cambios se encuentran en el canal. Se encuentran los bloques **PlutoSDR Sink** y **PlutoSDR Source** para transmitir y recibir a la señal. Como un intento de disminuir la influencia del ruido, se emplea el bloque **Simple Squelch**, utilizado para eliminar toda componente que no supere un determinado nivel. Por otro lado, el bloque **Costas Loop** se utiliza para sincronizar la constelación obtenida en una constelación que corresponda a QPSK.

En la figura siguiente se observa la implementación del sistema con la placa Adalm-Pluto y GNU-Radio.



Figura 26: Banco experimental

Los resultados obtenidos fueron los siguientes:

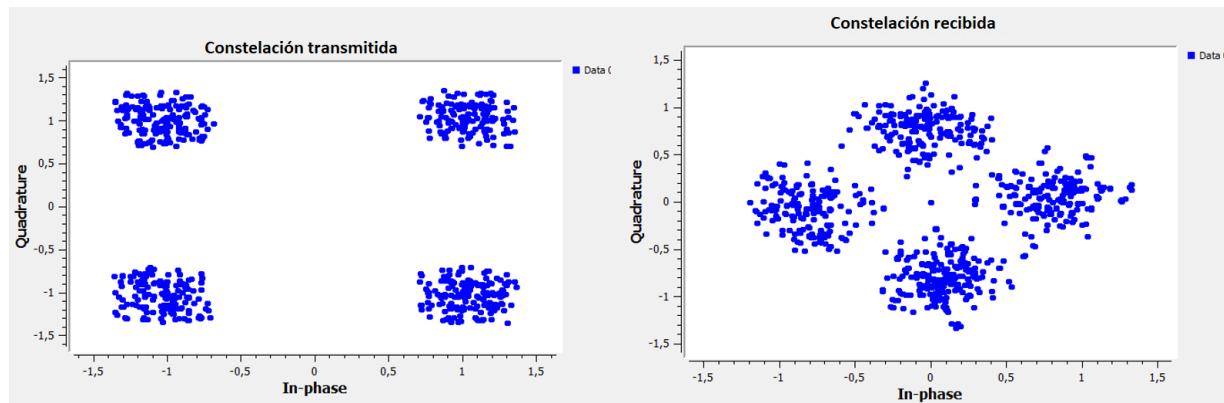


Figura 27: Constelaciones transmitida y recibida

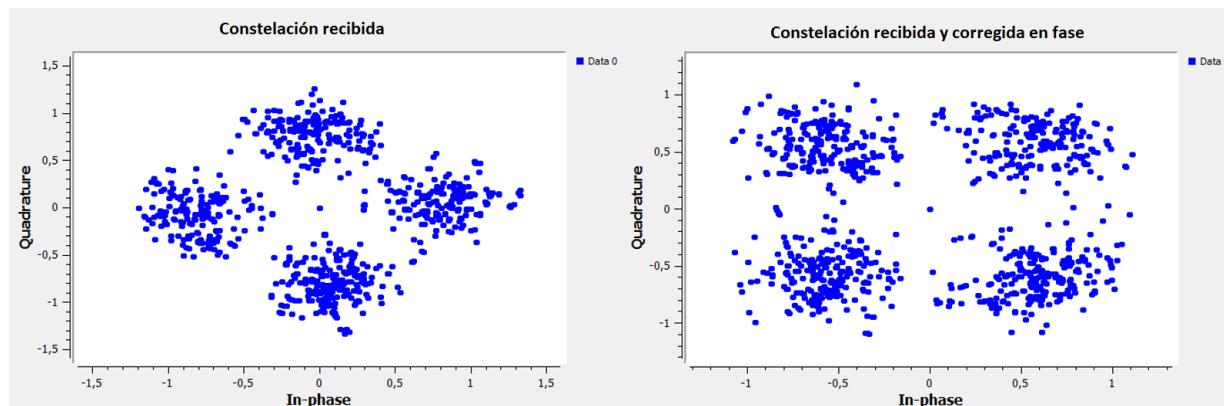


Figura 28: Constelaciones recibida y corregida en fase

En la figuras anteriores se puede notar la influencia del ruido, ya sea del canal como de decodificación dentro de la placa SDR. Y dado que las nubes de las constelaciones están sumergidas en ruido, no fue posible obtener una salida apropiada por lo que no se obtuvo ninguna imagen como respuesta.

No obstante, se realizó otra prueba experimental: enviar la misma imagen pero repetidas veces. Esto se realiza para disminuir la probabilidad de que una imagen enviada llegue con errores debido al ruido. El resultado temporal fue el siguiente:

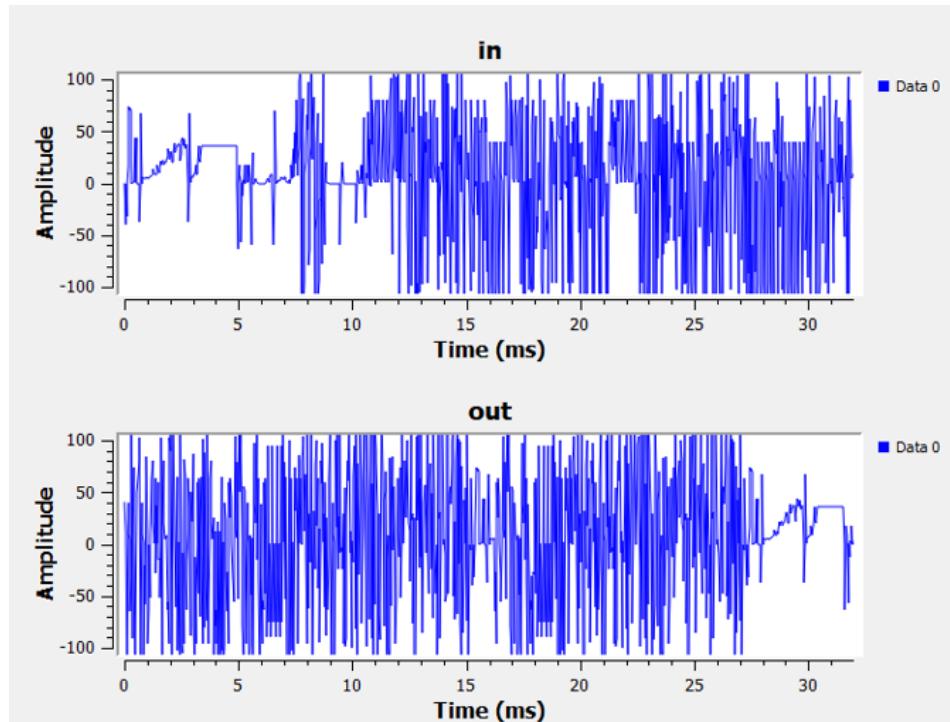


Figura 29: Señal transmitida y señal recibida (suma secuencial de imágenes)

Claramente la información llegaba pero no era posible visualizarla en el archivo resultante. Esto es porque el archivo resultante posee la suma de información que llegaba de manera secuencial, es decir, una imagen tras otra, debido a la repetición en la transmisión. La mayoría de las transmisiones utiliza la repetición para mejorar la comunicación o incluso para promediar la información y así disminuir la probabilidad de error de una sola imagen transmitida.

4. Conclusiones

A pesar de las complicaciones de Drivers y compatibilidad con los sistemas operativos, se implementó de manera efectiva el radioenlace mediante el Software GNU-Radio y el dispositivo Adalm Pluto SDR. Esta prueba fue muy importante ya que nos permitió visualizar una señal que se transmitió, asegurando el funcionamiento del sistema de comunicación. Esto da hincapié a implementar otros sistemas y modelos de comunicación en dispositivos SDR. Aún así, al estar utilizando el puerto USB para la comunicación, podría incluso limitar la velocidad a la que se transmite la información. También podría utilizarse el puerto Ethernet de la placa SDR para enviar y recibir datos por redes.

Las simulaciones dieron resultados muy favorables. Se pudo estudiar de manera cualitativa y cuantitativa el efecto del ruido del canal, tanto en el aumento de la nube de la constelación como en la distorsión visible y la pérdida de datos en las imágenes recibidas. También se pudo corroborar la pérdida de datos mediante un análisis cuantitativo en el tamaño de las imágenes recibidas.

A pesar de que no se recibieron imágenes en el experimento, los resultados fueron muy favorables. Se concluye que no se recibió información debido a la inmensa cantidad de ruido que se observó en la constelación recibida. Una solución que se pudo optar fue cambiar el modelo de modulación. En este caso, al utilizar una modulación especial llamada GMSK(Modulación por desplazamiento mínimo Gaussiano), sí se recibió una imagen distorsionada. No obstante, podrían implementarse herramientas de filtrado o ecualización para mejorar la constelación resultante. Incluso podría cambiarse el método de codificación y de decodificación para mejorar la manera en que se detectan y corrigen errores.

Como trabajo posterior, se podría plantear mejorar el sistema de comunicación para la transmisión y recepción de imágenes, tanto en la utilización de distintos tipos de modulaciones como de un algoritmo que sea capaz de promediar la recepción de imágenes transmitidas con repetición, es decir, que sea capaz de generar la imagen deseada a partir de muchas imágenes recibidas con error. Esto quizás requiera de modelos y técnicas de Inteligencia Artificial con redes neuronales que apliquen algoritmos para la reconstrucción de imágenes.

Referencias

- [1] GNU-Radio <https://wiki.gnuradio.org/>
- [2] Analog Devices https://wiki.analog.com/_media/plutoworkshop.pdf
- [3] AD9361 Datasheet <https://www.analog.com/en/products/ad9361.html>
- [4] Analog Devices <https://wiki.analog.com/university/tools/pluto/users/quickstart>
- [5] Github
<https://github.com/sdrforengineers/LabGuides/blob/master/grcon2020/Intro-To-libIIO-and-IIOScope.pdf>
- [6] Communication Systems. Carlson, A. 4° Edition.
- [7] Software Defined Radio for Engineers. Collins, T. Analog Devices.