

# Sistema de comunicación de voz y datos en tiempo real utilizando radios definidos por software

Luis Herasme

Ingeniería electrónica y de comunicaciones  
Instituto tecnológico de Santo Domingo  
Santo Domingo, República Dominicana  
1088668@est.intec.edu.do

**Abstract**—En este documento se describe el diseño y funcionamiento de un sistema de comunicación implementado en radios definidos por software, capaz de transmitir voz y datos en tiempo real.

**Index Terms**—GFSK, Sockets, UDP, Transmisión de voz, GNU Radio

## I. INTRODUCCIÓN

Este sistema fue diseñado con la finalidad de lograr la transmisión y recepción inalámbrica de voz en tiempo real, además debido a su versatilidad tiene la capacidad de enviar datos. El sistema cuenta con cuantificación logarítmica de audio utilizando  $\mu$ -law, modulación GFSK y codificación de paquetes donde se lleva a cabo un proceso de sincronización de trama.

## II. CONFIGURACIÓN

El sistema de comunicación fue implementado utilizando radios definidos por software o SDR. El SDR utilizado es el LimeSDR-Mini que opera en el rango de 10 MHz - 3.5 GHz y tiene un ancho de banda en radiofrecuencia de 30.72 MHz además es capaz de realizar 30.72 MSPS donde cada muestra es de 12 bits.

El procesamiento de señales fue realizado sobre GNU Radio, GNU Radio es un kit de herramientas de desarrollo de software de código abierto que proporciona bloques de procesamiento de señales. Se puede utilizar con hardware de RF externo, o sin hardware en un entorno similar a una simulación [1].

Para la transmisión de audio todo el procesamiento fue realizado sobre la herramienta GNU radio, sin embargo para la transmisión de datos fue utilizado un pequeño programa en Python para la recepción de los PDU.

Debido a que solo se contaba con un SDR para llevar a cabo el proyecto, este fue utilizado para realizar tanto la transmisión como la recepción, esto es posible ya que el LimeSDR tiene dos antenas y nos permite enviar en una de ellas y recibir en la otra (también enviar por las dos o recibir por las dos). De hecho en el proceso de diseño del sistema se había pensado en aprovechar ambas antenas ya que en cada una de ellas hay un oscilador de radiofrecuencia que podemos modificar a nuestro antojo solo cambiando un parámetro. Al tener dos osciladores podríamos realizar una duplexación por división de frecuencia (FDD) para lograr hacer que la comunicación sea full duplex. Sin embargo, solo tendrían que realizarse pequeños cambios

de valores en variables si posteriormente se quiere realizar este experimento.



Fig. 1. Radio definido por software utilizado en este experimento (LimeSDR-Mini).

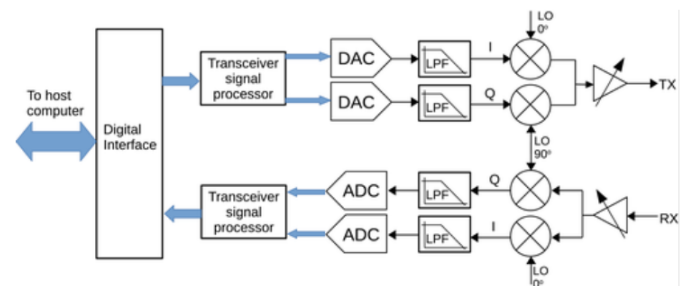


Fig. 2. Diagrama simplificado del funcionamiento del LimeSDR-Mini. [5]

## III. FUNCIONAMIENTO DEL SISTEMA

Como se mencionó anteriormente el sistema tiene dos variantes, una para la transmisión de voz y otra para la de datos. En ambas versiones cada usuario debe tener tanto la estructura del receptor como la del emisor. Para los experimentos realizados, el SDR se envía a sí mismo, pero por estar usando ambas antenas parecería un sistema simplex, pero no lo es, ya que se supone que una de las antenas es para enviar y la otra

para recibir por lo tanto podrían realizarse la transmisión y la recepción mismo tiempo.

Ahora veamos el funcionamiento la versión de voz del sistema:

#### A. Transmisor de voz

Para capturar el audio se utiliza un bloque llamado "Audio Source" de GNU radio, este bloque actúa como entrada de micrófono. Luego esta señal es amplificada al multiplicarla por una variable que es editable durante la ejecución que puede variar entre 1 y 10, esto se debe a que al convertirse en bytes (se cuantifica sin compansión) y luego ser cuantificado por el bloque de  $\mu$ -law, se observó que hay un mejor desempeño al amplificar la señal (por la cuantificación no logarítmica debido al cambio de tipo de datos).

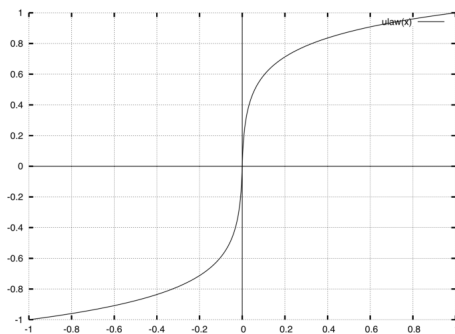


Fig. 3. Entra y salida de la función  $\mu$ -law [4].

El bloque de  $\mu$ -law realiza una cuantificación logarítmica de nuestro audio, esta hace pasar la señal por un compresor logarítmico y luego la cuantifica.

En este cuantificador tendremos pequeños pasos de cuantificación para amplitudes pequeñas y pasos de cuantificación grandes para los valores grandes de amplitud, lo que significa menor resolución en señales amplitud grande.

Luego de que la voz se cuantifica, esta es codificada utilizando un bloque llamado "Packet Encoder", este bloque tiene un mecanismo de sincronización de trama lo que permite que cuando los bits de audio lleguen al receptor estos no sean decodificados con un orden incorrecto provocando que la data se corrompa. Luego de que la información es empaquetada, es enviada a un modulador GFSK, con dos muestras por símbolo y un BT de 0.35, donde el BT es utilizado para mejorar la eficiencia espectral mediante un filtrado paso bajo, como puede verse en la siguiente figura 4.

Luego de que la información es modulada se envía al bloque "LimeSuite Sink (TX)", donde es muestreada a 2MSPS. Este bloque lleva la señal a paso banda multiplicándola por una señal generada por oscilador local de frecuencia modificable, en este sistema se utilizó 440MHz ya que es una porción del espectro de radio atribuida internacionalmente a la radioafición [3]. Luego la señal es amplificada con un amplificador de bajo ruido (LNA) con ganancia configurable, para este experimento se utilizó una ganancia de 40 dBm, aunque esta ganancia no

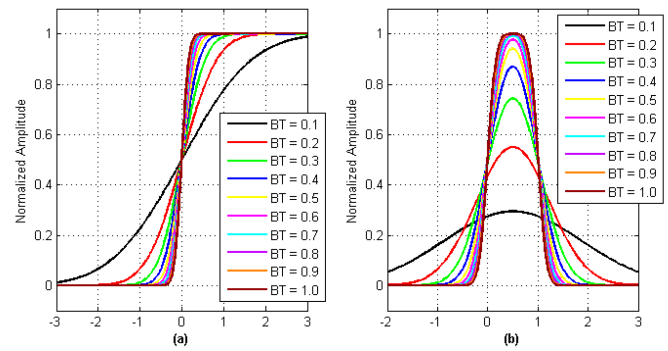


Fig. 4. Relación entre la respuesta unitaria del filtro gaussiano y el parámetro BT. (a) Respuesta escalonada unitaria del filtro gaussiano; y (b) respuesta al impulso de la unidad de filtro gaussiano [2] .

es muy trascendente porque estamos transmitiendo de una antena de SDR a la otra. Sin embargo, como se mencionó anteriormente para que el sistema funcione de una SDR a otro solo tendríamos que cambiar unos pocos parámetros y esta potencia y la frecuencia del oscilador son parte de ellos.

Luego la señal es transmitida para ser recibida por el otro SDR o en nuestro caso por la otra antena.

#### B. Receptor de voz

Cuando llega la señal al receptor pasa por un filtro análogo de 5MHz y un filtro digital de 1MHz, se amplifica la señal con un LNA para posteriormente pasar por dos ramas donde hay dos osciladores en cuadratura. Esto nos produce una señal con parte real e imaginaria, que va al conversor análogo digital y luego a nuestra estructura de receptor en GNU radio (Ver anexo). Luego de que está en el receptor de GNU radio, la

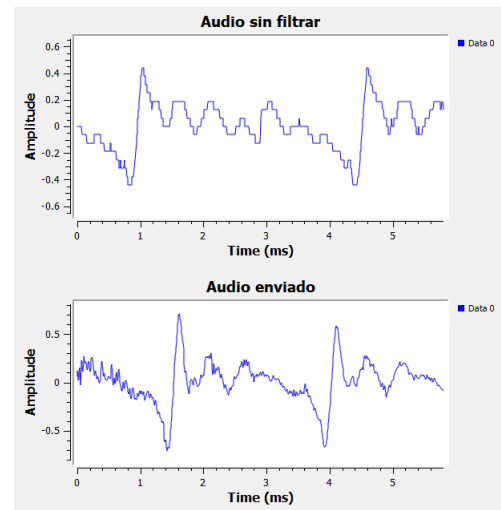


Fig. 5. Audio enviado tomado del micrófono y audio recibido sin filtrar.

señal pasa a un bloque llamado "Costas Loop", este bloque se usa para estimar y compensar las diferencias de frecuencia y fase entre la onda portadora de una señal recibida y el oscilador local del receptor convirtiendo la señal recibida a banda base. Luego de este proceso la señal es apta para pasar

por el demodulador GFSK, este demodula la señal llevándola a bits y los envía al bloque llamado "Packet Decoder", el que hace la sincronización de trama.

Ahora que tenemos los bits sincronizados debemos convertirlos nuevamente en audio, por lo tanto los pasamos por el decodificador  $\mu$ -law donde se realiza la función inversa al compresor logarítmico y luego el resultado se convierte en audio (El procedimiento conjunto de compresión y expansión se denomina companding).

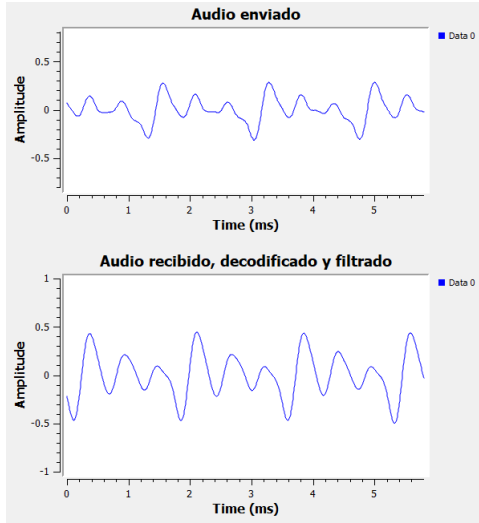


Fig. 6. Audio recibido al filtrarse.

Por el proceso de cuantificación a la señal se le agregaron altas frecuencias indeseadas por lo tanto es filtrada y finalmente enviada al bloque "Audio Sink" que nos permite reproducir la señal a través algún dispositivo de audio.

#### IV. VELOCIDAD

En el flujo primero se muestrea el audio del micrófono a 44.1kHz, luego el codificador genera 8 bits en serie por cada muestra. Podemos saber cuantos bits por segundo transmitimos de esta forma, no estoy tomando en cuenta el codificador ni el empaquetador ya que a estos se le hará la operación contraria en el receptor y se van a cancelar.

$$R_b = \left(44.1k \frac{\text{muestras}}{\text{segundo}}\right) \left(8 \frac{\text{bit}}{\text{muestras}}\right)$$

$$R_b = 352.8kbps$$

#### V. VERSIÓN DE DATOS

Para la versión de datos se usaron los mismos bloques hasta el sincronizador de trama ya que lo único que cambia es la fuente de información. Esto es posible porque en la versión de voz el audio era convertido en bits para luego ser modulado utilizando dos símbolos en GFSK, por esta razón se puede utilizar la misma interfaz de radio para transmitir cualquier tipo de dato si previamente es convertido a bits.

Aprovechando esta propiedad del sistema se usa un bloque para que reciba la información de un puerto UDP, entonces

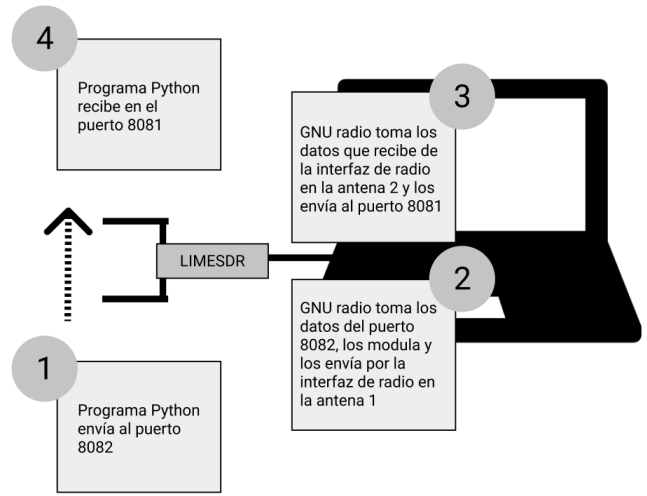


Fig. 7. Flujo de la información en los experimentos realizados durante el desarrollo del sistema.

escribí un pequeño programa en Python que envía datos a este puerto y así le paso la información a GNU radio. GNU radio envía la información recibida en el puerto utilizando la misma interfaz de radio que usa para la voz. Luego de que se recibe y la información, esta es procesada por GNU radio igual que en la versión de voz hasta la sincronización de trama, luego los datos son enviados a un bloque llamado "UDP Sink" que se encarga de enviar la información recibida por otro puerto. Escribí un segundo programa en Python que lee de dicho puerto e imprime la información que recibe cada vez que se llena un buffer de 32 bits.

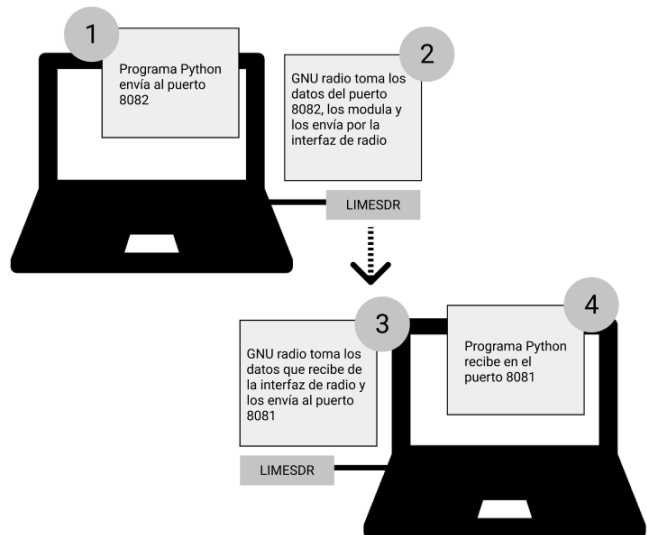


Fig. 8. Flujo normal de la información.

El diagrama presentado en la figura 7, solo aplica para las pruebas realizadas durante el desarrollo de este sistema, como se mencionó anteriormente el propósito es lograr una

comunicación full duplex mediante FDD y al igual que para el caso de voz los cambios de configuración serían mínimos para lograr la comunicación de una computadora a otra. Para el funcionamiento normal de este sistema también elabore un diagrama que puede verse en la figura 8.

## VI. ERROR DE CUANTIFICACIÓN

Al enviar el audio este fue primero cuantificado para así utilizar su representación binaria para poder modularlo y llevarlo a paso banda. Hay una pérdida de información en el cuantificador; los datos entran al programa por el bloque "Audio source" son muestras de audio de 32 bits, sin embargo, el cuantificador uLaw solo genera valores de 8 bits en serie, por lo tanto hay una evidente pérdida de información en el proceso de cuantificación.

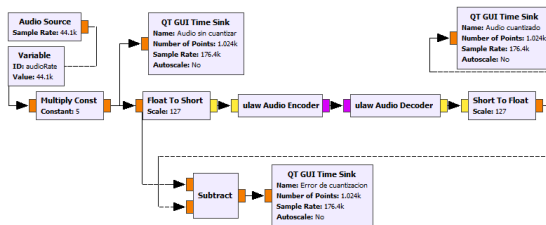


Fig. 9. Experimento realizado para determinar el error de cuantificación de sistema.

Como se puede ver en la figura 9, para determinar el error lo que se hizo fue cuantizar el audio y luego al audio original restarle el audio cuantizado, dándonos como resultado el error que se puede ver en la figura 10.

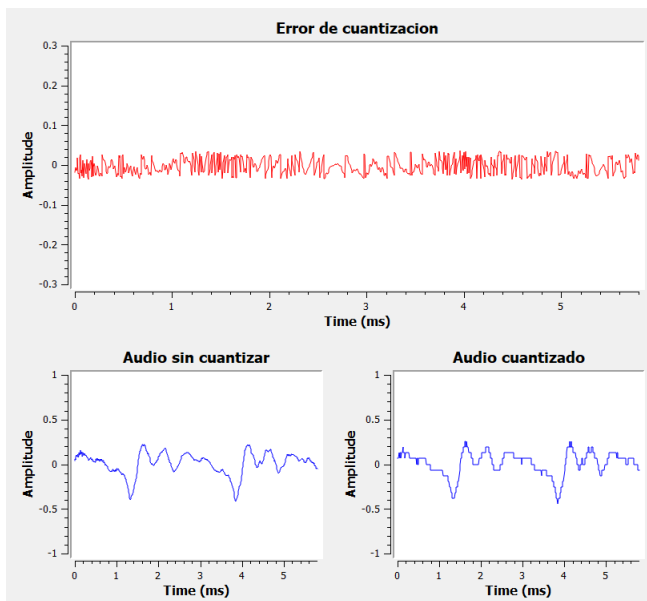


Fig. 10. Error de cuantificación.

## VII. MAS DETALLES SOBRE EL LIMESDR-MINI

Como se puede ver en la figura 11, el LimeSDR-Mini cuenta con dos conectores de RF, y como se puede ver aún mejor

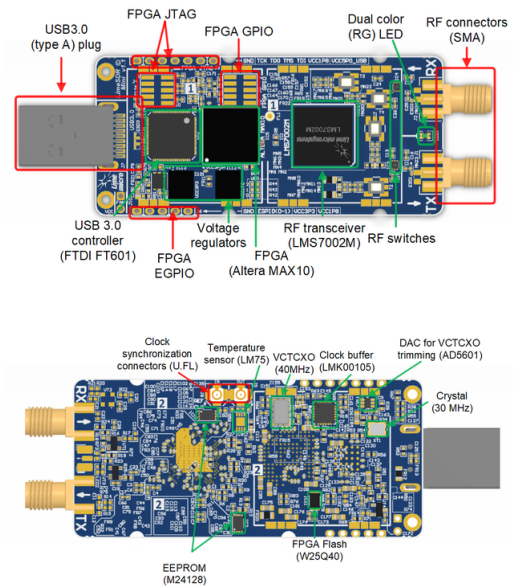


Fig. 11. Diagrama interno del LimeSDR-Mini.

en la figura 2, el SDR nos permite enviar por una antena y recibir por otra, enviar por las dos antenas o recibir por las dos antenas, entonces con dos LimeSDR-Mini se podría hacer MIMO 2x2. Para enviar por las dos antenas o recibir por las dos antenas solo hay que cambiar una configuración en el bloque de LimeSDR, de SISO a MIMO y se habilitarán dos entradas al bloque del transmisor o si esta configuración fue establecida en el bloque del receptor a este le aparecerán dos salidas, por tanto podemos utilizar el conjunto de señales para mejorar la capacidad de nuestro enlace.

**LimeSuite Sink (TX)**  
Device serial: 1D4C2A4CC49084  
Device type: LimeSDR-Mini  
Chip mode: SISO  
Load settings: NO  
RF frequency: 440M  
Sample rate: 2M  
Oversamples: Default  
CH0:INCO frequency: 0  
CH0:CHIX mode: UPONVERT  
CH0:Calibration: OFF  
CH0:PA path: Band1  
CH0:Analog filter: ON  
CH0:Analog filter bandwidth: 5M  
CH0:Digital filter: OFF  
CH0:Gain: 40

**LimeSuite Sink (TX)**  
Device serial: 1D4C2A4CC49084  
Device type: LimeSDR-Mini  
Chip mode: MIMO  
Load settings: NO  
RF frequency: 440M  
Sample rate: 2M  
Oversamples: Default  
CH0:INCO frequency: 0  
CH0:CHIX mode: UPONVERT  
CH0:Calibration: OFF  
CH0:PA path: Band1  
CH0:Analog filter: ON  
CH0:Analog filter bandwidth: 5M  
CH0:Digital filter: OFF  
CH0:Gain: 40

Fig. 12. Bloque configurado de SISO a MIMO.

## VIII. CONCLUSIÓN

En este proyecto se diseñó e implemento un sistema de comunicación de forma exitosa, algo que podría mejorarse para versiones futuras de este proyecto es la calidad del audio que al ser recibida y decodificada no se escucha con total nitidez. Además, durante del desarrollo del proyecto por la escasa documentación del software utilizado (GNU Radio), no se tuvo mucha elección con respecto al uso de algunos bloques lo que produjo que haya una menor libertad creativa en el proceso de diseño. Una versión futura de este sistema de comunicación probablemente estará implementada en otro software, además, para la versión de datos propondría el uso de un sistema de detección errores. En una futura versión más robusta podría utilizarse un esquema de modulación con una eficiencia espectral más alta como OFDM, y mediante la interfaz de datos realizar transmisión de vídeo.

TABLE I  
ALGUNOS PARÁMETROS DEL SISTEMA

|                                 |            |
|---------------------------------|------------|
| <i>Modulación</i>               | GFSK       |
| <i>Tasa de bits</i>             | 352.8kbps  |
| <i>Método de cuantificación</i> | $\mu$ -law |
| <i>BT filtro Gaussiano</i>      | 0.35       |
| <i>Frecuencia</i>               | 440MHz     |
| <i>Amplificación</i>            | 40dBm      |
| <i>Filtrado de RF análogo</i>   | 5MHz       |
| <i>Filtrado de digital</i>      | 1MHz       |
| <i>Filtrado de audio</i>        | 44Hz       |
| <i>Duplexación</i>              | FDD        |

## REFERENCES

- [1] G. Radio, "About GNU Radio · GNU Radio", GNU Radio, 2021. [Online]. Available: <https://www.gnuradio.org/about/>. [Accessed: 23-Apr- 2021].
- [2] Zhang, J.; Zhang, S.; Wang, J. Pseudorange Measurement Method Based on AIS Signals. Sensors 2017, 17, 1183. <https://doi.org/10.3390/s17051183>.
- [3] Wikipedia, "70-centimeter band - Wikipedia", En.wikipedia.org, 2021. [Online]. Available: [https://en.wikipedia.org/wiki/70-centimeter\\_band](https://en.wikipedia.org/wiki/70-centimeter_band). [Accessed: 23- Apr- 2021].
- [4] "Ley Mu - Wikipedia, la enciclopedia libre", Es.wikipedia.org, 2021. [Online]. Available: [https://es.wikipedia.org/wiki/Ley\\_Mu](https://es.wikipedia.org/wiki/Ley_Mu). [Accessed: 23- Apr- 2021].
- [5] Michal, CA. A low-cost multi-channel software-defined radio-based NMR spectrometer and ultra-affordable digital pulse programmer. Concepts Magn Reson Part B. 2018; 48B:e21401.

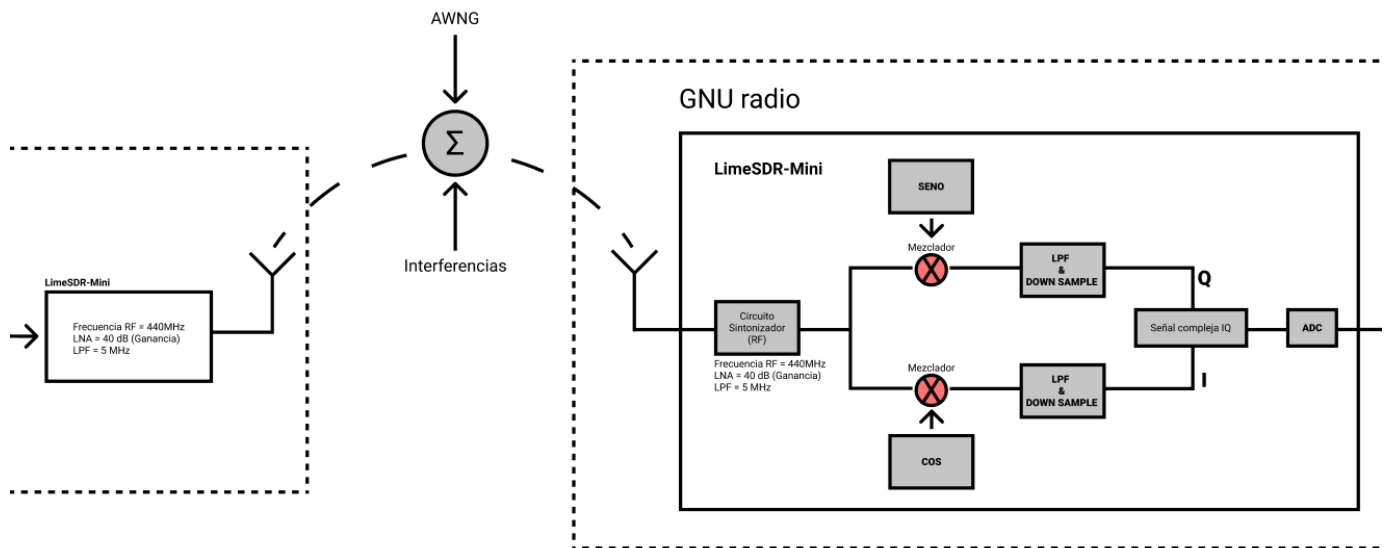


Fig. 13. Modulación paso banda en el LimeSDR.

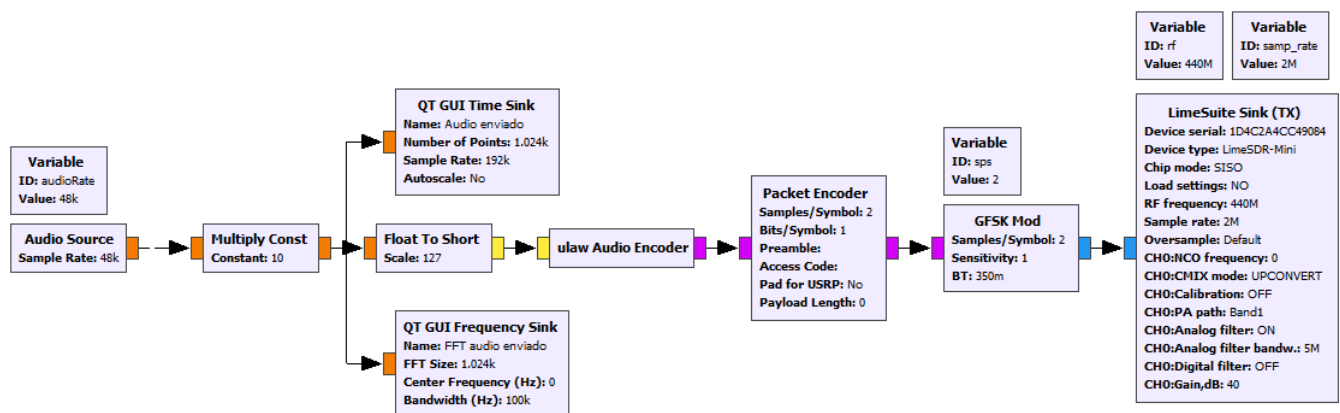


Fig. 14. Estructura del emisor.

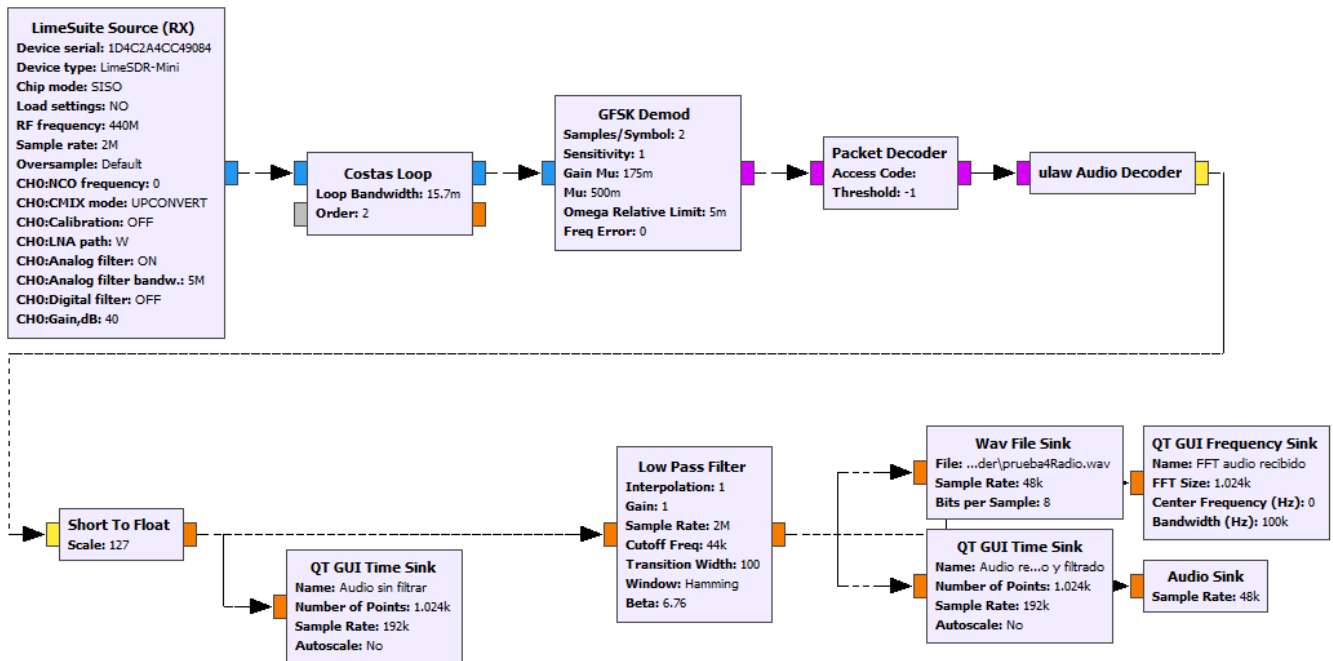


Fig. 15. Estructura del receptor.

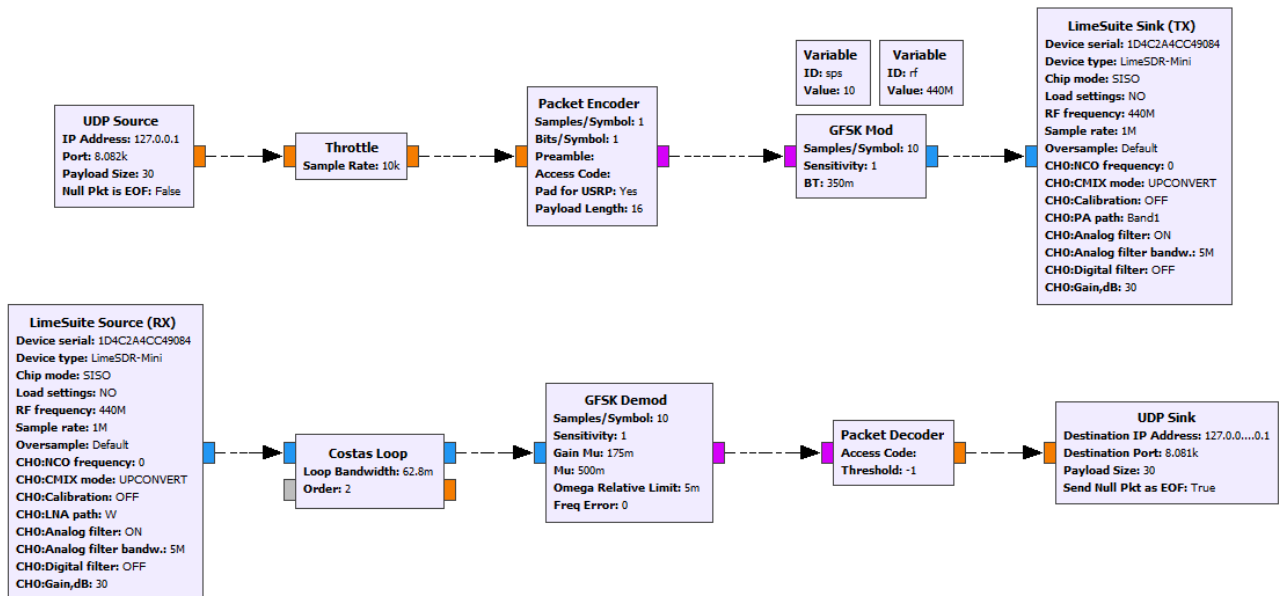
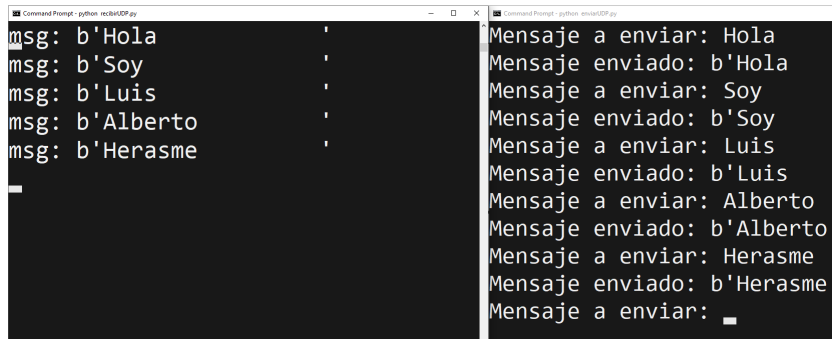


Fig. 16. Estructura del emisor y receptor en la versión para datos.



```
Command Prompt: python recbinUDP.py
msg: b'Hola'
msg: b'Soy'
msg: b'Luis'
msg: b'Alberto'
msg: b'Herasme'
_

Command Prompt: python enviosUDP.py
Mensaje a enviar: Hola
Mensaje enviado: b'Hola'
Mensaje a enviar: Soy
Mensaje enviado: b'Soy'
Mensaje a enviar: Luis
Mensaje enviado: b'Luis'
Mensaje a enviar: Alberto
Mensaje enviado: b'Alberto'
Mensaje a enviar: Herasme
Mensaje enviado: b'Herasme'
Mensaje a enviar: _
```

Fig. 17. Demostración de flujo de datos.