MODEM FM MEDIANTE RADIO DEFINIDO POR SOFTWARE (SDR), OCTAVE, GNU RADIO Y HACK RF ONE: UNA REVISIÓN DE SOFTWARE Y HARDWARE

FM MODEM VIA SOFTWARE DEFINED RADIO (SDR), OCTAVE, GNU RADIO AND RF ONE HACK: A SOFTWARE AND HARDWARE REVIEW

Eric Mario Silva Cruz
Tecnológico Nacional de México / IT de Oaxaca, México
eric.cruz@itoaxaca.edu.mx

Miguel de Jesús Sánchez Maldonado
Tecnológico Nacional de México / IT de Oaxaca, México
17160553@itoaxaca.edu.mx

Miguel Ángel Pérez Solano
Tecnológico Nacional de México / IT de Oaxaca, México
miguel.solano@itoaxaca.edu.mx

Roberto Tamar Castellanos Baltazar
Tecnológico Nacional de México / IT de Oaxaca, México
roberto.castellanos@itoaxaca.edu.mx

Víctor Manuel Jiménez Ramos
Tecnológico Nacional de México / IT de Oaxaca, México
victor.jimenezr@itoaxaca.edu.mx

Recepción: 13/noviembre/2022 **Aceptación:** 17/mayo/2023

Resumen

El presente artículo es una revisión del diseño de radio definido por software (*SDR*) para un sistema digital de frecuencia modulada (*FM*), considerando el software y hardware para su implementación, usando octave y una interfaz gráfica de usuario (*GUI*) en GNU radio. El diseño digital de *FM* se presenta considerando el tiempo y la frecuencia de muestreo, el tipo de datos para un módem de banda estrecha (*Narrowband*). Se aplica una metodología de diseño considerando los aspectos en el tiempo y el espectro en frecuencia, especificando el proceso de modulación y sus requisitos. Al aplicar la metodología a través de código en octave y en un diagrama de bloques en GNU radio existente en la literatura se adquieren

habilidades de programación en software de código abierto para diseños que permitan la implementación en la tarjeta de desarrollo *Hack rf one*.

Palabras Clave: FM, GNU radio, Hack rf one, octave, SDR.

Abstract

The present article shows the review of a software-defined radio (SDR) digital frequency modulated (FM) system considering software and hardware requirements for its implementation, in code using octave and a graphical user interface (GUI) in GNU Radio. The digital design of an FM modulator is presented considering the time, the sampling rate, the data type, for a narrow band modem (NB- FM). A design methodology is applied considering the aspects in time and spectrum frequency, specifying the modulation process and its requirements. By applying the methodology through code in octave and in block diagram in GNU radio, programming skills are acquired in open-source software for designs in software radio existing in the literature, and its implementation in the developer board Hack rf one.

***Keywords*:** *FM, GNU radio, Hack rf one, octave, SDR.*

1. Introducción

La presente investigación se enfoca al diseño e implementación de *SDR* mediante las tecnologías de acceso libre, octave, *GNU radio* y la tarjeta *Hack rf one*, las cuales permiten la integración de laboratorios de simulación, así como la implementación de redes de comunicaciones digitales inalámbricas. En el caso particular, el departamento de Ingeniería Electrónica del Instituto Tecnológico de Oaxaca requiere la conformación de una red de sensores sísmicos, en donde se realice la transmisión inalámbrica de datos de sensores con dispositivos de corto y largo alcance que sean reconfigurables, de bajo costo y escalables, como parte del proyecto de sismología y riesgo sísmico.

La modulación digital *FM* se basa en modelos de diseño en software, estas características hacen del radio definido por software una disciplina que permite hacer usos de sistemas de modulación en diferentes rangos de frecuencia e

implementados en tarjetas reconfigurables, haciendo uso de técnicas analógicas y digitales para transmitir datos que no requieran grandes anchos de banda. En [Xiao, 2018], se observa un algoritmo para modulación por código de pulsos con *FM* (*PCM-FM*) implementado en un campo de arreglo de compuertas programables (*FPGA*), con una tasa de modulación en banda base de 1kbps~10Mbps, el cual permite entender la necesidad de sistemas de hardware basados en *FPGA*, como lo es la tarjeta *Hack rf one*.

En [De la Cruz, 2018], se presentan elementos de sistemas basados en anchos de banda limitados basados en un análisis espectral de la frecuencia.

En [Zhang, 2016], se presentan elementos de diseño para SDR utilizando el software con licencia simulink de Matlab de la empresa Mathworks. El diseño presentado refleja la necesidad de utilizar herramientas de código abierto, debido a que el software con licencia requiere inversiones mayores para generar técnicas digitales, facilitando generar sistemas más elaborados como *FSK*, *GFSK*, o la multiplexación ortogonal por división de frecuencia *OFDM*, [Maziar, 2016], empleando la aplicación de algoritmos y técnicas de procesamiento digital de señales en 5G, [Luo, 2016].

FM es una variante de modulación de ángulo y pertenecen a las técnicas de modulación no lineales, esta señal se caracteriza por su requerimiento de un ancho de banda grande y buen desempeño en la presencia de ruido, [Stremler, 1998], [Tomassi, 1996], actualmente se aplica en *broadcast FM* de alta fidelidad, *broadcast* de televisión y audio, modulación de portadora de microondas y sistemas de comunicación punto a punto.

El modulador – demodulador (*MODEM*) *FM* está constituido por el modulador en frecuencia que tiene una señal portadora sinusoidal, [Proakis, 2013], la cual varía su componente de frecuencia en función de la integración de la señal mensaje a transmitir, tal como se expresa en la ecuación 1.

$$u(t) = A_c \cos\left(2\pi f_c\, t + 2\pi k_f \int_{-\infty}^{t} m(\tau) d\tau\right) \qquad (1)$$

Donde:

$u(t)$: Señal modulada en *FM.*

A_c : Amplitud de la señal portadora.

f_c : Frecuencia de la señal portadora (*carrier*).

k_f : Constante de desviación del factor de modulación *FM.*

$m(\tau)$: Integración del mensaje a transmitir.

2. Métodos

La metodología a utilizar en el diseño del *MODEM* se presenta aplicando un código (script) en el software octave y el diseño de *FM Narrow band* en GNU radio, [GNU, 2020]:

- Analizar y definir los bloques de un sistema modulador *FM* mediante la aplicación de procesamiento digital de señales.
- Establecer las características de la señal a transmitir considerando la frecuencia de muestreo y la frecuencia de portadora de la señal.
- Sintetizar la señal del mensaje y la señal modulada con sus características en el dominio del tiempo y la frecuencia aplicando la transformada de *Fourier*.
- Comparar el ancho de banda de la señal del mensaje y la señal modulada en términos de su contenido espectral.
- Añadir ruido a la señal transmitida para valorar el desempeño del sistema de modulación *FM* aplicando ruido blanco gaussiano (*AWGN*) y observando la relación señal a ruido (*SNR*).
- Realizar el bloque demodulador de *FM* mediante el empleo de un detector de fase envolvente, utilizando un filtro Transformador de Hilbert para eliminar la portadora y obtener el mensaje transmitido.
- Implementar el modem *FM* definido por software mediante GNU radio y la tarjeta de desarrollo *Hack rf one*, [Balint, 2014].

3. Resultados

Resultados del diseño del MODEM FM en Octave

El diseño del *MODEM* se basa en el script en Matlab de [Proakis, 2013], con una señal de mensaje rectangular definido en la ecuación 2.

$$m(t) = \begin{cases} 1, & 0 \le t < \dfrac{t_0}{3} \\ -2, & \dfrac{t_0}{3} \le t < \dfrac{2t_0}{3} \\ 0, & otros\ valores \end{cases} \quad (2)$$

Donde:

$m(t)$: Mensaje a transmitir en el tiempo t.

t_0 : Tiempo de duración del mensaje.

Se considera la base del tiempo de duración del mensaje t_0, el cual define la relación del índice de modulación con respecto a la portadora. A continuación, se muestra en la tabla 1 las características para el proceso de diseño, así como el resultado de la simulación del script en octave definido en [Proakis, 2013], usando las librerías de *signal* en octave para procesar la señal digital.

Tabla 1 Características de la señal.

t_0=0.15	Duración de la señal en tiempo
t_s=0.0005	Intervalo de muestreo
$t = [0{:}\, ts{:}\, t_0]$	Vector de tiempo
f_c=200	Frecuencia portadora
k_f=50	Índice de modulación
$f_s = 1/ts$	Frecuencia de muestreo
df=0.25	Resolución en frecuencia (bins)
snr=5; $snr\_lin = 10^{(snr/10)}$	Relación señal a ruido del ruido blanco aditivo gausiano (*AWGN*)

Las características de la portadora y el índice de modulación son parámetros que pueden variar y están directamente relacionados a la frecuencia del mensaje.

El diseño del mensaje está delimitado por el intervalo de muestreo y definido en la ecuación 1, en donde un script en octave define $m = [-3 \cdot ones(1, t_0/(4t_s)), 3 \cdot ones(1, t_0/(4 * t_s)), 2 \cdot ones(1, t_0/(4 * t_s)), 2 \cdot ones(1, t_0/(4 * t_s) + 1)]$ como el mensaje transmitido.

Una vez diseñado el mensaje, se debe aplicar el proceso de integración, considerando el período de muestreo t_s y la longitud del mensaje, como se muestra el código de programación en octave:

```
for i=1: length(t)-1
inLm(i+1)= inLm(i)+m(i)*ts;
end
```

Una vez obtenido la integración del mensaje, este valor se adiciona al valor de la frecuencia de la portadora f_c para obtener la frecuencia modulada (u), como se observa en la ecuación 3 y su resultado en la figura 1.

$$u = cos\left(2\pi f_c t + 2\pi k_f \cdot inLm\right) \tag{3}$$

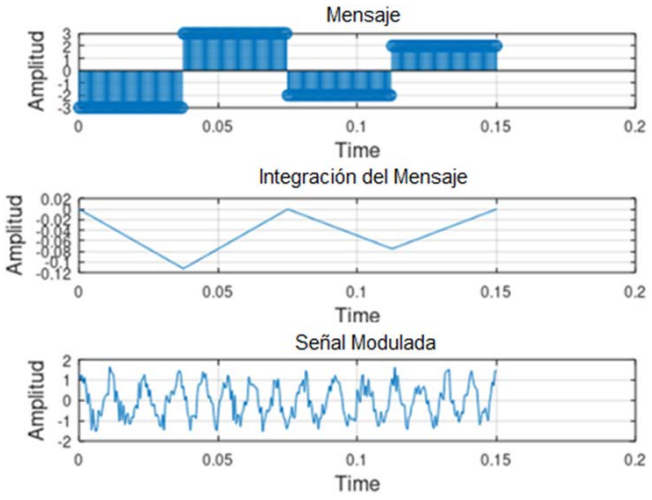


Figura 1 *FM* mediante integración $t_0 = 0.15$, $t_s = 0.0005$.

El resultado de simulación mostrado en la figura 1 de la modulación *FM* es la aproximación a una respuesta de modulación analógica, sin embargo, el proceso de muestreo de las señales origina la variación en amplitud de las muestras debido al ruido de cuantización, [Proakis, 2013]. En la figura 2 se observa un mensaje con menor cambio en suamplitud y con un ruido de cuantización menor, en donde el mensaje está definido por el mismo intervalo, pero con diferentes tiempos en el tiempo de duración de cada símbolo de la señal con $m = [ones(1, t_0/(3t_s)), -2 \cdot ones(1, t_0/(3 * t_s)),\ zeros(1, t_0/(3t_s) + 1)]$.Este tiempo de duración del símbolo en relación con el muestreo se debe considerar en una implementación a nivel hardware debido al ruido de cuantización originado en el conversor analógico digital (*ADC*).

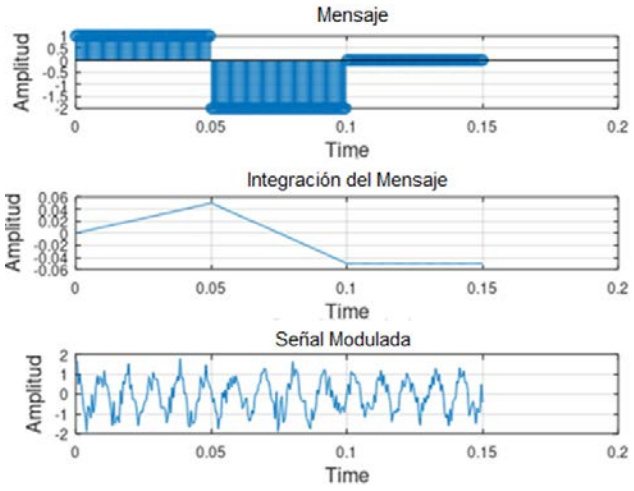


Figura 2 *FM* con $t_0 = 0.15$, $t_s = 0.0005$.

Siguiendo con la metodología se procede a obtener los componentes en frecuencia de las señales, esto se logra mediante la aplicación de la transformada de *Fourier* a las secuencias en tiempo, en el caso del diseño se ocupa la función *fftseq*, a la cual se aplica el mensaje, el período de muestreo y la resolución en frecuencia df, la señal obtenida requiere un escalamiento en magnitud y para graficarse es necesario generar un vector de frecuencias f para el espectro del mensaje, [Proakis, 2013]:

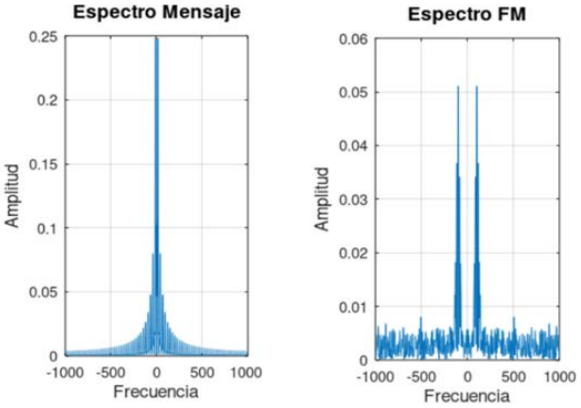
```
[M,m,dfl]=fftseq(m,ts,df);          %Transformada de Fourier
M=M/fs                               %Escalamiento
f=[0:dfl:dfl*(length(m)-1)]-fs/2;    %Vector de frecuencia
```

Al realizar el proceso de obtención del espectro del mensaje y la señal modulada, mediante la aplicación de la transformada de *Fourier*, figuras 3 y 4, se define el contenido espectral del mensaje original en figuras 3a y 4a, así mismo se observa el espectro modulado de la señal de *FM* en figuras 3b y 4b, con dos símbolos en el mensaje respectivamente.

De acuerdo con la metodología se agrega ruido utilizando un modelo de canal en banda base con ruido blanco aditivo gaussiano (*AWGN*), mediante la aplicación del código de la función *spower*, y el cálculo del ruido:

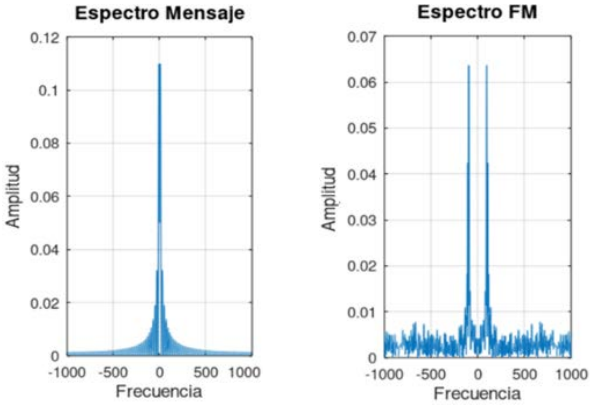
```
signaLpower=spower(u(1:length(t)));      % potencia de la señal modulada
```

```
noise_power=signaLpower/snr_lin      % potencia de la señal
noise_std=sqrt(noise_power);
noise=noise_std*randn(1,length(u));  % ruido AWGN
r=u+noise;                           % mensaje modulado más ruido
```



a) Mensaje *b) Mensaje modulado*

Figura 3 Espectro con un símbolo en el mensaje.



a) Mensaje *b) Mensaje modulado*

Figura 4 Espectro con dos símbolos en el mensaje.

La etapa del demodulador se diseña considerando un transformador de Hilbert, el cual está diseñado en [Proakis, 2013] mediante la función *env_phas*, y es la etapa del demodulador el cual realiza la supresión de la portadora:

```
[v,phase]=env_phas(u2,ts,fc);     % envolvente de fase de la señal
phi=unwrap(phase);                % restaurar fase original
dem=(1/(2*pi*kf))*(diff(phi)/ts); % diferencial de la fase
```

La demodulación *FM* se observa al obtener la señal recibida en la figura 5, la cual se obtiene aplicando el transformador de Hilbert y se obtienen las afectaciones de la señal en el dominio del tiempo con respecto al mensaje original.

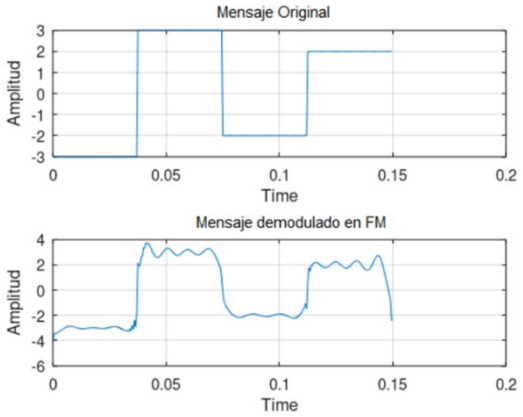


Figura 5 Mensaje transmitido y recibido.

Resultados del diseño del Modem FM en GNU Radio

Para diseñar el sistema de modulación *FM* en *SDR* mediante GNU radio, se propone utilizar el diagrama a bloques del modem en banda base de la figura 6, en donde es necesario conocer los tipos de datos que se puedan manejar, como son; flotante (*float*), entero o complejo (*complex*), esto con la finalidad de poder programar en el ambiente de GNU radio, [Balint, 2014], [GNU, 2020], [Clark, 2015].

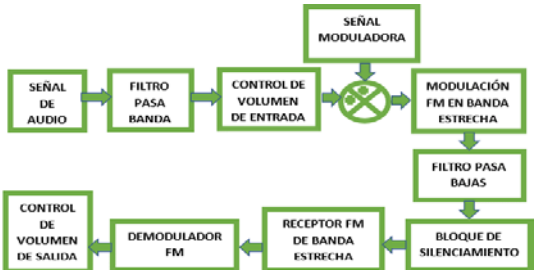


Figura 6 Diagrama a bloques del Modulador *FM Narrow Band*.

La entrada de datos del modulador *FM* consiste de un bloque *wav file source*, para datos de formato de audio *wav* de tipo flotante, puede ser monoaural o estéreo, figura 7. El bloque *variable*, define una tasa de muestreo (*sample rate*) con el identificador (*Id*) *samp_rate*, figura 8, de 48 kHz.

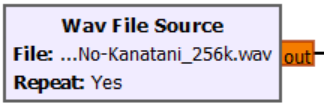


Figura 7 Bloque de señal de audio a transmitir.

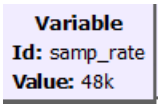


Figura 8 Bloque que define la tasa de muestreo.

Se aplica un filtro pasa banda (*band pass filter*) de 200 Hz, a la señal para limitar el ancho de banda, figura 9, una decimación de 1, frecuencia de muestreo de 48 kHz, con frecuencia de corte en la banda baja de 100 Hz y alta de 20 kHz, y una ventana Hamming con factor beta de 6.76. En el bloque *Qt gui range* de la figura 10, que tiene el *Id* volumen y con etiqueta volumen de entrada de tipo flotante (*float*) inicializado en 0 y terminando en 5, con incrementos de 0.25, en el parámetro *widget counter + slider* es posible modificar los valores de la simulación.

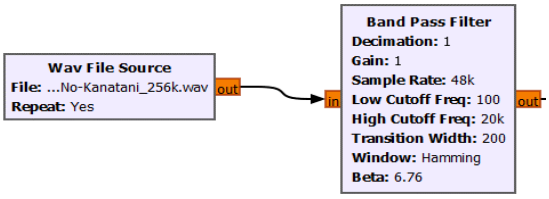


Figura 9 Bloque de filtrado para limitar ancho de banda.

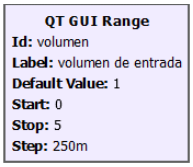


Figura 10 Bloque QT para modificar parámetros de la señal.

El bloque del filtro limitador se conecta al bloque *multiply const* (Figura 11), de tipo *float* y con *Id volumen*, cuyo bloque permite amplificar el volumen de entrada del archivo *wav*. La señal moduladora es senoidal de tipo *float*, amplitud de 0.15 y una frecuencia de muestreo con el valor *samp_rate*, en el bloque *signal source*, figura 12.

Figura 11 Bloque QT para modificar una constante.

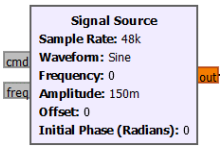


Figura 12 Bloque para generar la señal moduladora.

Las señales *multiply const* y *signal source* se suman en el bloque sumador (*Add*), figura 13, que son el mensaje de audio y la portadora, respectivamente. El bloque transmisor narrowband *FM NBFM transmit*, figura 14, define valores de entrada flotante en un rango [-1, +1] y salida de banda base compleja con modulación *FM*.

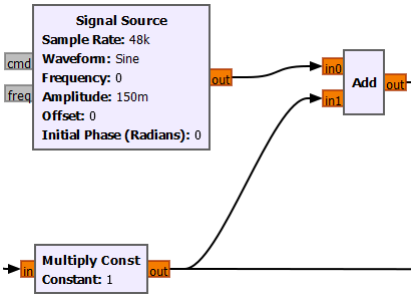


Figura 13 Bloque sumador.

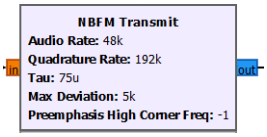


Figura 14 Bloque transmisor *NBFM*.

La tasa de audio (*audio rate*) a una frecuencia de muestreo del flujo mayor a 16k y debe ser un valor entero, definido en el parámetro *samp_rate*.

La tasa de cuadratura debe ser un múltiplo de la tasa de audio con un valor de 4 veces (192 kHz). El parámetro *tau* es una pre-acentuación de tipo flotante con un valor en el transmisor de *FM* de 0.000075, una desviación media (*max deviation*) de 5 kHz y un preénfasis de frecuencia (*preemphasis high comer freq*) de -1.

La entrada para el transmisor es de tipo *float* y su la salida es de tipo compleja (*complex*). La señal modulada se filtra con un filtro pasa bajas (*low pass filter*), de respuesta finita al impulso (*FIR*) complejo, figura 15, con una decimation de 1, una ganancia unitaria y un *sample rate* definido por el *Id if_rate* a 192 kHz. La frecuencia de corte (*cutoff freq*) es de 5 kHz con un ancho de transición (*transition width*) entre la banda de corte y la banda de paso de 2 kHz, utilizando una ventana tipo Hamming con 6.76 de valor de beta.

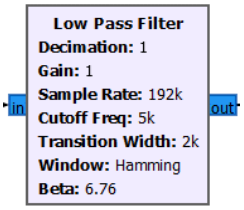


Figura 15 Filtro Pasabajas.

Mediante un bloque *Qt gui range* (Figura 16), se configura el bloque *simple squelch* con el *Id nivel_del_umbral* y con etiqueta *squelch*. El valor de defecto (*default value*) de este bloque en GNU radio es de -50 iniciando con -100 y avanzando en pasos de 5, con un bloque *widget* tipo *counter* + *slider* para poder modificar el valor durante la simulación.

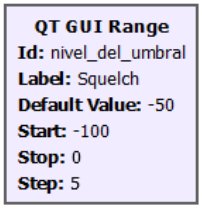


Figura 16 Bloque *Qt GUI range*.

Se conecta la salida del filtro pasa bajas con el bloque *simple squelch* de la figura 17. Este bloque está basado en la potencia de señal promedio y el umbral en decibelios (*dB*), el cual ayuda a reducir la interferencia que se escucha cuando la señal es recibida, el bloque utiliza un identificador *Id nivel_de_umbral* como parámetro para el bloque y una ganancia unitaria denominada *alpha*. La salida del filtro pasa bajas se conecta a un bloque *complex to mag*, figura 18. Para observar

la señal en el dominio del tiempo se requiere un bloque *Qt GUI time sink*, el cual toma un conjunto de valores flotantes y la gráfica en tiempo real. Para esto, se agrega un bloque variable de la figura 19 y se le asigna el *Id usrp_rate* con un valor de 576 kHz.

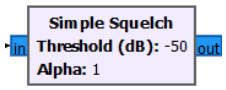


Figura 17 Bloque *simple squelch*.

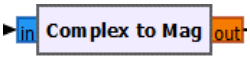


Figura 18 Bloque c*omplex to mag*.

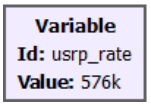


Figura 19 Bloque Variable.

En el bloque *Qt GUI time sink* de la figura 20, se parametriza el valor de visualización con una tasa de muestreo definida en el *Id usrp_rate*. Además, se asigna un número de puntos de 2048 sin auto escala y con un valor de 0.1 de periodo de actualización.

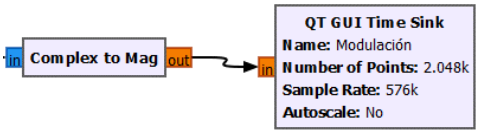


Figura 20 Bloque *Qt GUI time sink*.

El bloque *NBFM Receive* es el receptor demodulador de *FM Narrow band*, figura 21, con un valor de datos entrada de banda base complejo y una salida de audio en el rango [-1, +1]. Con respecto a la configuración de los datos, se tiene una tasa de audio (*audio rate*) perteneciente al valor del bloque con el *Id samp_rate* (48 kHz), una tasa de cuadratura (*quadrature rate*) de 192 kHz, una *tau* de 0.75 y una desviación máxima (*max deviation*) de 5 kHz.

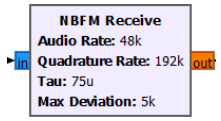


Figura 21 Bloque *NBFM receive*.

Se conecta un bloque *Qt GUI range* de tipo *float*, figura 22, con el *Id volumen_de_salida* con valor por defecto de 0.250, con un intervalo de 0 a 5 con pasos de 0.25, adicionando un *widget* tipo *counter* + *slider* para poder modificar el valor durante la simulación. La salida del bloque *NBFM receive* se conecta a un bloque *multiply const* para la ganancia del volumen de salida, figura 23, con entrada y salida de tipo *float*. Para graficar la señal demodulada en el dominio del tiempo se utiliza el bloque *Qt GUI time sink* de dos entradas con un denominador con número de puntos de 2048 y una frecuencia de muestreo correspondiente al valor del *Id usrp_rate* (576 kHz), sin auto escala y con dos entradas de tipo *float*, figura 24.

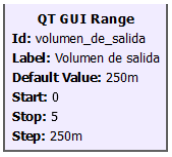


Figura 22 Bloque *Qt GUI range*.

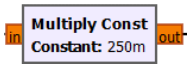


Figura 23 Bloque Multiply Const.

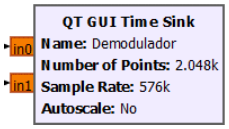


Figura 24 Bloque Qt GUI time sink.

Con un bloque *audio sink*, figura 25, se reproduce la señal a través de los altavoces de la computadora, utilizando una frecuencia de muestreo definida por *Id samp_rate* y como nombre del dispositivo (*device name*) se coloca el nombre correspondiente a las bocinas del computador, por ejemplo, altavoces (*conexant smart audio HD*), con una frecuencia de muestreo de 48 kHz.

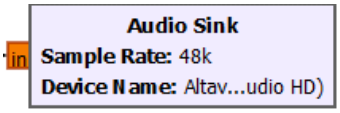


Figura 25 Bloque *audio sink*.

El bloque final sirve para graficar la señal original y demodulada mediante un bloque *Qt GUI time sink* de dos entradas. La primera entrada proviene de la salida del bloque *multiply const* ubicado después del *NBFM receive* y la segunda corresponde a la salida del bloque *multiply const* ubicado antes del bloque *add*. Se observa en la figura 26 el bloque completo de la simulación en GNU radio.

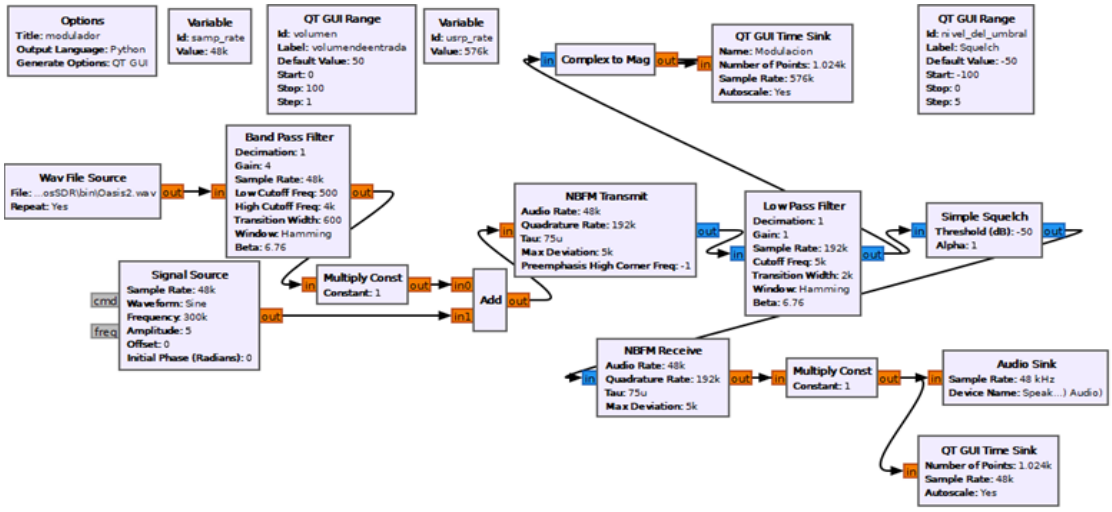
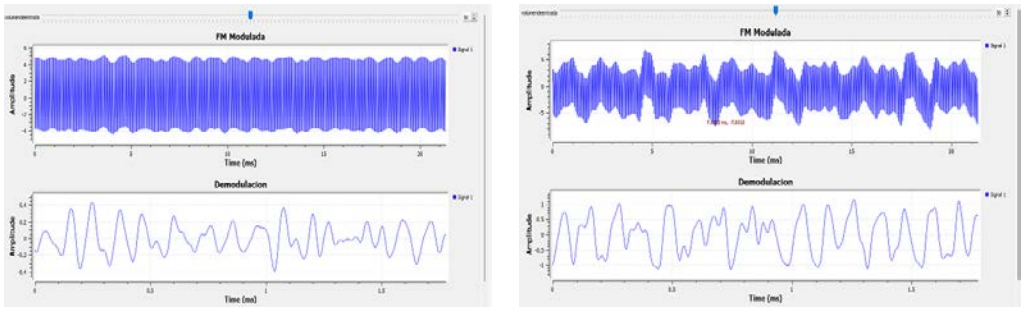


Figura 26 *MODEM FM* con etapas.

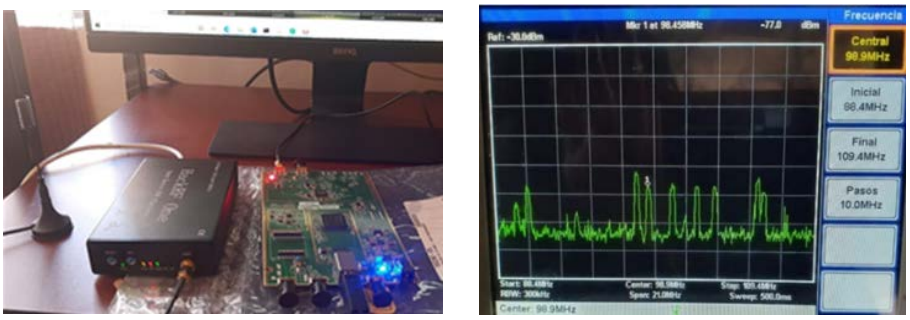
El sistema diseñado es reconfigurable, por lo cual se puede ajustar la frecuencia de la señal portadora, en la figura 27 se presenta la simulación de la portadora con una frecuencia de 400 kHz y 800 kHz, con cada modificación de la portadora se debe ajustar la frecuencia de muestreo y la amplitud de la señal para evitar la sobre modulación con respecto a la señal de audio transmitida.

El modem *FM* diseñado en SDR se implementa con la tarjeta *Hack rf one*, figura 28a, la cual se emplea mediante una selección de frecuencias de radio conocidas, con las restricciones de transmisión en potencia, con una cobertura no mayor a 10 metros para evitar la interferencia de emisores locales, tal como se observa la transmisión en el osciloscopio en la banda de 88.4 MHz a 109 MHz, figura 28b.



a) 400 kHz *b) 800 kHz*

Figura 27 *Modem FM* GNU radio.



a) Hack RF One *b) Osciloscopio*

Figura 28 Transmisión en *Hack rf one*.

4. Discusión

Se presentan las etapas de análisis y síntesis de un modem *FM* mediante las plataformas Octave y GNU Radio, utilizando scripts basados en C y mediante bloques prediseñados en Python usando la librería Qt, respectivamente.

Al seguir una base metodológica de diseño se requiere matemática aplicada y los algoritmos para la modulación y realizar el radio definido por software, para lo cual es indispensable tener conocimientos de sumas, multiplicaciones, diferenciación, integración, mediante técnicas de procesamiento digital de señales, utilizando formatos de datos tipo flotante y complejo.

En cuanto a la simulación en GNU radio, este software posee bloques prediseñados para realizar los diferentes sistemas de comunicación, y sus características facilitan el diseño, siempre y cuando se tenga conocimiento de los distintos bloques y su programación mediante la variación de sus parámetros.

La ventaja de este tipo de software es la posibilidad de modificar los diseños y crear códigos o scripts propios mediante octave, y así incluir bloques personalizados, en

donde se origina la necesidad de conocer a detalle el lenguaje de programación *python* y su librería *Qt*, las cuales resultan herramientas útiles para desarrollar nuevos diseños.

Con respecto a cada MODEM diseñado se puede observar en script y en bloques las características de transmisión considerando aspectos de las señales en tiempo y frecuencia, lo cual se observa realizando codificación en scripts, por ejemplo, en octave se generó la transmisión de una señal rectangular, la cual representa una señal digital, variando los parámetros de la señal portadora y su muestreo. En lo que respecta al diseño en GNU radio la señal transmitida es una señal de audio tipo *wav* que representa una señal analógica, parametrizando cada uno de los bloques en el transmisor y receptor para poder reproducir el audio original en el receptor.

El diseño del modulador *SDR FM* se comprobó mediante su implementación en la tarjeta *Hack rf one* siguiendo las características de diseño de un sistema de comunicaciones aplicando algoritmos usando procesamiento digital de señales.

5. Conclusiones

Las conclusiones de esta investigación ofrecen un panorama general de desarrollo de los sistemas de radio definidos por software mediante plataformas de código abierto, como lo son octave y GNU radio.

Se presentaron las características de diseño de un sistema de modulación *FM*, con un *MODEM FM* en banda base mediante scripts de código en octave, en donde se observan las afectaciones de la señal debido al proceso del muestreo, especificándose las carencias de una señal con una baja frecuencia de muestreo, tal como se observa en las figuras 1 y 2, con una transmisión de dos mensajes distintos definidos por la duración de cada símbolo, mismos que son importantes para realizar la integración de la señal que modificará a la señal portadora.

A pesar del muestreo reducido, es posible obtener adecuadamente la señal transmitida en el demodulador tal como se observa en las figuras 3 y 4, en donde se presentan la señal en frecuencia y se observa el espectro de la señal de *FM*.

Se caracteriza de manera correcta los bloques en GNU radio de los elementos que conforman un *MODEM FM Narrow band*, considerando los tipos de datos entre

bloques, la parametrización de los bloques, así como el uso correcto de las herramientas gráficas de la librería Qt que están diseñadas mediante *widgets*.

En cuanto a la metodología aplicada se estructura correctamente a cada una de las situaciones empleadas, tanto en código en octave como en los diseños a bloques de GNU radio, la presente revisión de hardware y software permite observar las simulaciones de las señales transmitidas y recibidas, corroborando una correcta modulación y demodulación digital.

La experiencia en el diseño con software radio representa un punto de partida para quienes tengan el interés en desarrollar sistemas *SDR*, en donde sea posible la transmisión de datos con diversas tasas de muestreo, distintos anchos de banda, el diseño de filtros, la implementación de *MODEM*s mediante tarjetas de hardware reconfigurable como lo son las *FPGA*, *Hack rf one* o bien las *USRP*, lo cual permite comprobar distintas técnicas de comunicaciones digitales y en diversas bandas de frecuencia.

6. Bibliografía y Referencias

[1] Balint, S., GNU Radio Tutorials Lab 1-5, Ettus Research, version 1.0, https://files.ettus.com/tutorials/, 2014.

[2] Clark, P., & Clark, D., Field Expedient SDR: Introduction to Software Defined Radio, Meadow Registry Press, second edition, 2015.

[3] De la Cruz, P. A., & Moreira, S. R., Diseño e implementación de un prototipo de Transmisión de Mensajes de Alerta en FM sobre cinco grupos de tres frecuencias portadora para su uso en vehículos de emergencia utilizando USRP y GNU Radio, Tesis, Universidad Politécnica Salesiana, Ecuador, http://dspace.ups.edu.ec/handle/123456789/15475, 2018.

[4] GNU, R., Wikipedia, La enciclopedia libre, https://es.wikipedia.org/w/index.php?title=GNU_Radio&oldid=131784095, 2020.

[5] Luo, F. L., & Zhang, C., (editores) Signal Processing for 5G: Algorithms and Implementations, First Edition, John Wiley & Sons, Ltd, 2016.

[6] Proakis, J., Contemporary Communication Systems Using MATLAB®, Third Edition, Cengage Learning, 2013.

[7] Maziar, N., & Yue, W., et al. Overview of 5G modulation and waveforms candidates. J. Commun. Inf. Netw. 1, 44–60, https://doi.org/10.1007/BF03391545, 2016.

[8] Stremler, F., Introducción a los Sistemas de Comunicación. 3ra ed. México, editorial Addison Wesley Longman, 1998.

[9] Tomassi, W., Sistemas de Comunicación Electrónicas. 2da ed. México, editorial Prentice Hall Hispanoamericana S.A, 1996.

[10] Xiao, H., & Liu, X., & et al., Design of PCM/FM Baseband Modulation Transmitter Based on Software Radio, 2018 Eighth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC), Harbin, China, pp. 714-717, doi: 10.1109/IMCCC.2018.00154, 2018.

[11] Zhang, K., Implementación de un demodulador de radio FM definido por Software. Tesis, Universidad Politécnica de Valencia, https://riunet.upv.es/handle/10251/68767?show=full, 2016.