Comunicaciones (E311)
Julio 2021

LABORATORIO I
DEMODULACIÓN DE SEÑALES DE FM - SDR

Autor:
Pereyra Nehuen (878/6)

1. Interpretación del problema

El objetivo del trabajo es lograr demodular una señal FM. Para ello la cátedra brindó un Dongle USB el cual nos permitirá capturar las muestras para luego procesarlas.

1.1. Detalles del problema a resolver

El Dongle permite realizar lo que se conoce como Software Defined Radio (SDR), es un sistema de radiocomunicaciones donde varios de los componentes típicamente implementados en hardware (mezcladores, filtros, moduladores/demoduladores, detectores, etc) son implementados en software, utilizando un ordenador personal u otros dispositivos de computación embebidos. En la figura 1 se puede ver un diagrama en bloques simplificado de un sistema SDR.

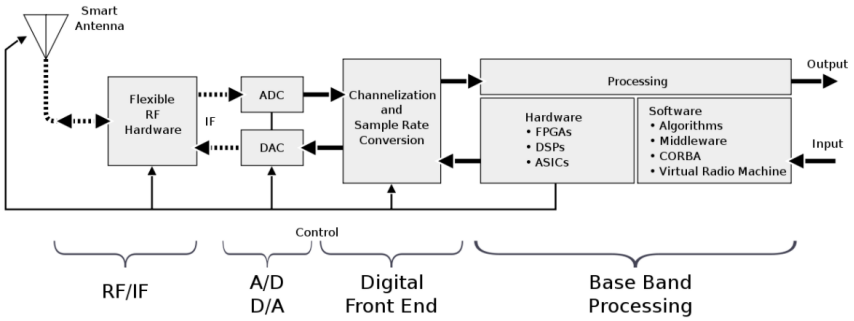

Figura 1: Diagrama en bloques de un sistema SDR.

Podemos destacar que el Dongle esta conformado por dos integrados principales, los cuales son:

- **R820T2**: Un sintonizador de señales que puede sintonizar un rango de frecuencias de 24 MHz a 1700MHz. Posee una figura de ruido de 3.5 dB y consume alrededor de 180 mA.
- **Controlador USB RTL2832U y DSP (Digital Signal Processor)**: Posee un ADC (Analog to Digital Converter) y un DSP que realiza la conversión de frecuencias intermedia a banda base a través de mezcladores en fase y cuadratura (I/Q), el filtrado pasa bajos, el re-muestreo en fase y cuadratura y el envío de las muestras al puerto USB.

En la figura 2 se puede observar estos dos integrados principales.

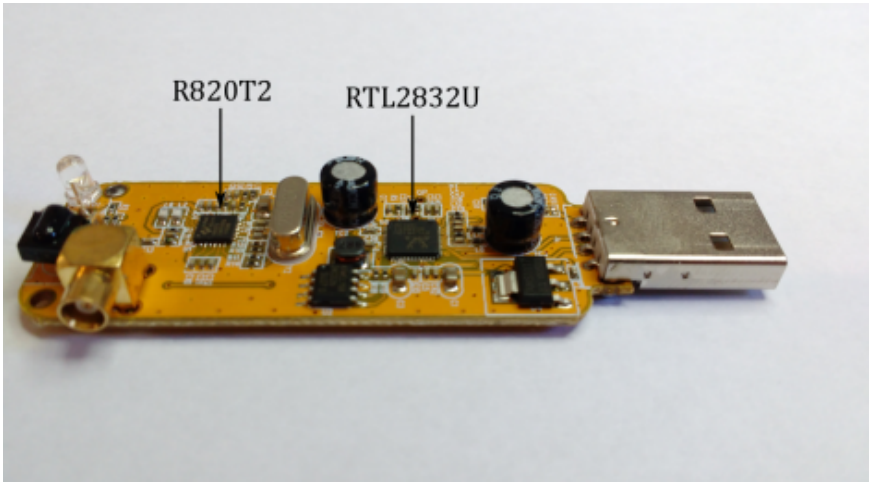

Figura 2: Integrados principales de un sistema SDR.

Para facilitarnos el uso del Dongle, se utiliza el software SDRSharp que nos brinda una interfaz gráfica para trabajar con él. En la figura 3 se puede ver como luce.

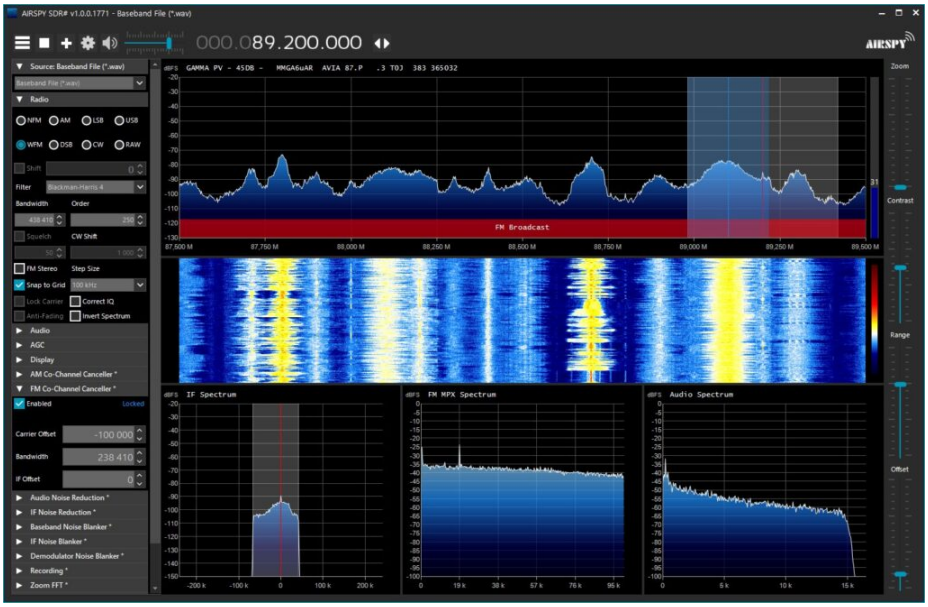

Figura 3: Interfaz gráfica del software SDRSharp.

Para realizar la captura de las muestras, utilizo el plugin “IF Recorder” para así almacenar una porción del espectro de FM en un archivo wav. En la figura 4 se puede ver cuando abrimos el archivo wav con las muestras almacenadas. Se puede observar que se lograron capturar varias estaciones FM.

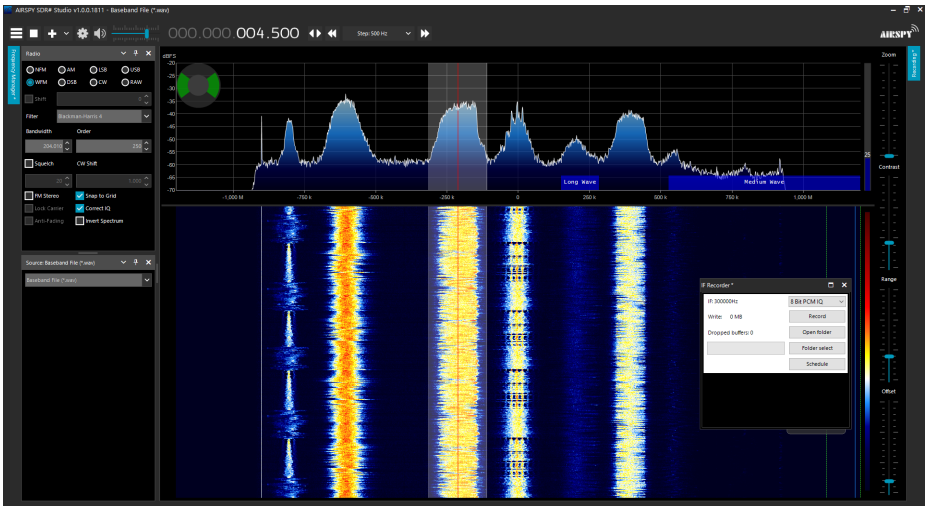

Figura 4: Estaciones capturadas con SDRSharp.

Una vez obtenidas las muestras se utiliza Octave, que es un software libre que permite realizar cálculo numérico, para procesar esas muestras y demodular la señal. En la figura 5 se puede ver su logo.

Figura 5: Logo del software Octave.

2. Implementación

Se intentó modularizar el problema para hacerlo más independiente y reutilizable. Se separaron en 3 archivos los módulos que se utilizarán:

- **FM_DEMOD_PEREYRA**: Esta función contiene la demodulación como tal.
- **calc_DEP**: Permite estimar la densidad espectral de potencia (DEP) de la señal pasada por parámetros. Además permite definir si se mostrará el gráfico de la DEP con frecuencia normalizada o no.

- **filtro_digital**: Esta función permite aplicar un filtro digital a la señal enviada por parámetro. Además grafica el filtro construido en módulo y fase.

El archivo **main** se encarga de leer las muestras e inicializar algunas variables para luego ejecutar FM_DEMOD_PEREYRA(). Por último reproduce el audio. El pseudocódigo se puede ver a continuación:

```
1  Se realiza la lectura de las muestras
2  Me quedo con uno de los canales de audio
3  Se aproxima la DEP
4  Se establece la frecuencia de corte normalizada de la estación a filtrar
5  Se llama a la función FM_DEMOD_PEREYRA
6  Se reproduce el audio
```

Listing 1: Pseudocódigo del main.

Para leer las muestras, se utiliza la función audioread() la cual retorna un vector con las muestras y la frecuencia de muestreo (f_s). En mi caso, trabajo con una frecuencia de muestreo de 2400 KHz por lo tanto se puede ver en la DEP que el rango de frecuencias va de 0 a 1200 KHz.

Luego se llama a la función calc_DEP() para graficar el espectro, en la figura 6 se puede ver el espectro obtenido con frecuencia normalizada, mientras en la figura 7 podemos verla sin normalizar.

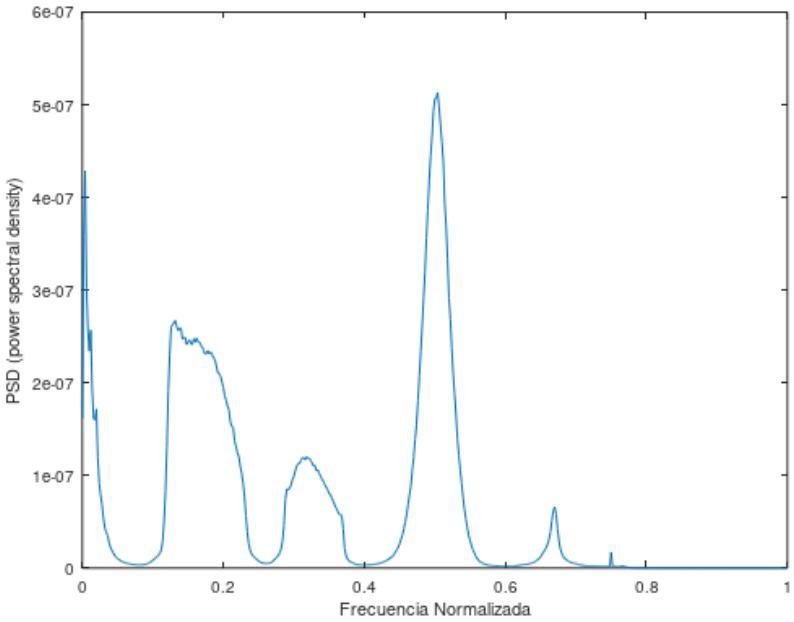

Figura 6: DEP con frecuencia normalizada.

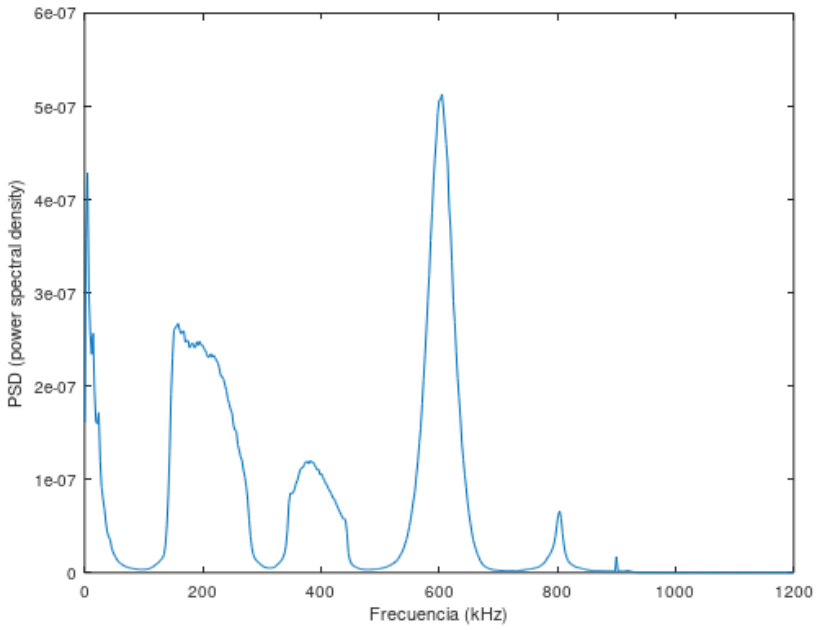

Figura 7: DEP sin frecuencia normalizada.

Podemos distinguir en el espectro, que hay varias estaciones transmitiendo. En este caso tomaremos la estación que se encuentra entre 0.12 y 0.2 en frecuencia normalizada. La frecuencia normalizada se calcula en base a la frecuencia de muestreo, con la siguiente fórmula:

$$f_n = \frac{B}{\frac{f_s}{2}}$$

Donde B es el ancho de banda en Hz y f_s es la frecuencia de muestreo.

Luego llamamos a la función FM_DEMOD_PEREYRA() a la cual le pasamos como parámetros: la señal, la frecuencia de muestreo, la frecuencia de corte normalizada para la estación y para el filtrado en banda base además del factor de diezmado que se explicará luego. A continuación se puede ver el pseudocódigo de esta función:

```
1  Se establece el orden del filtro
2  Se aplica el filtro digital para filtrar la estación deseada
3  Se realiza la demodulación
4  Se aplica un filtrado en banda base
5  Se grafica la señal de audio demodulada
6  Se aplica un diezmado
7  Se retorna la señal diezmada y la nueva frecuencia de muestreo
```

Listing 2: Pseudocódigo de la función FM_DEMOD_PEREYRA().

Cuando se llama a la función de filtro digital, el cual aplica el filtro y gráfica el filtro aplicado. La función del filtro, debe recibir la señal a filtrar, el orden del filtro y la frecuencia de corte normalizada.

Se utiliza un filtro tipo butterworth, el cual es diseñado para producir la respuesta más plana que sea posible hasta la frecuencia de corte. El orden del filtro establece el grado de aceptación o rechazo de frecuencias por arriba o por debajo, de la respectiva frecuencia de corte. La frecuencia de corte normalizada es el rango que establecimos anteriormente (es un filtro pasa bandas para el filtrado de la estación). En la figura 8 podemos ver el módulo y fase del filtro generado. Mientras que en la figura 9 podemos ver el espectro de la señal filtrada.

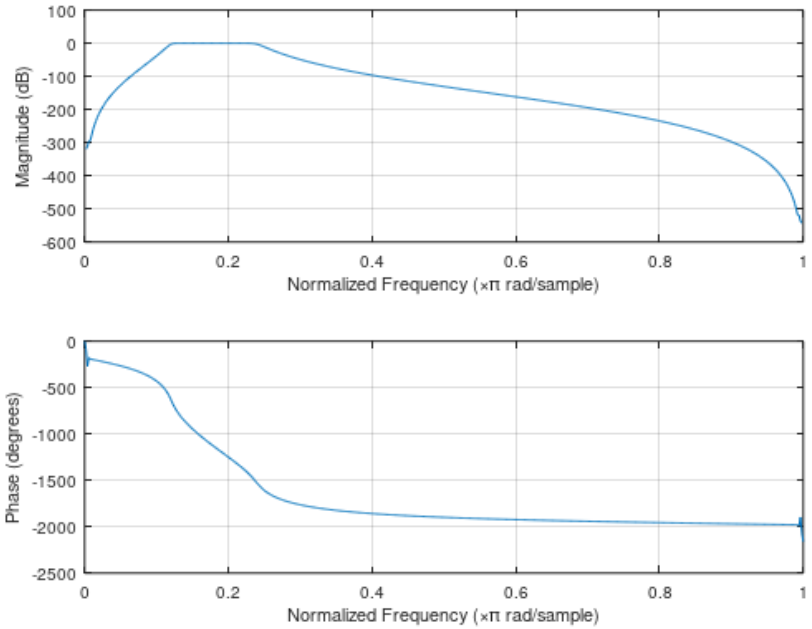

Figura 8: Módulo y fase del filtro generado.

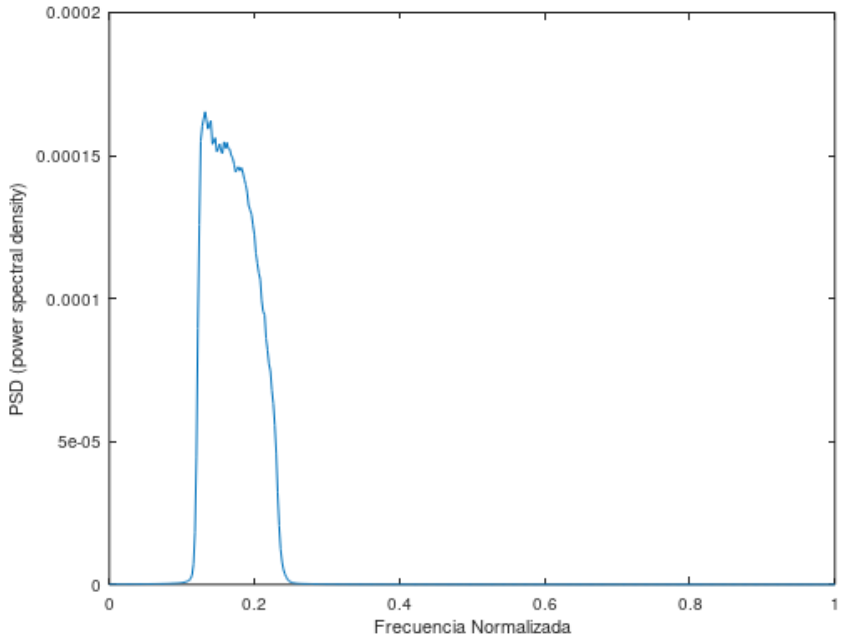

Figura 9: DEP de la señal filtrada.

Para la parte de la demodulación, se realizó el diagrama en bloques que se muestra en la figura 10.

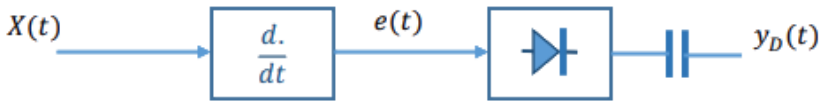

Figura 10: Diagrama en bloques para la demodulación.

Donde $X(t)$ es la señal modulada y $y_D(t)$ es la señal demodulada.

Como tenemos una señal modulada en frecuencia para extraer el mensaje de esa señal, primero derivamos la señal a la entrada obtenemos la siguiente expresión:

$$e(t) = -A(2\pi f_p + \frac{d\phi}{dt})sen(2\pi f_p t + \phi(t))$$

Si distribuimos podemos observar que tenemos una constante (si f_p permanece constante) multiplicada por una señal sinusoidal de frecuencia f_p en donde la fase varía muy lentamente, más la desviación de frecuencia multiplicado por una señal sinusoidal de frecuencia f_p. Como tiene la forma de una señal modulada en AM, entonces con un detector de envolvente y un capacitor (para filtrar el valor de continua) se podría obtener el mensaje. Esto en octave se puede implementar primero derivando con la función diff(), luego el detector de envolvente se realiza con la función abs(), y quitamos la continua utilizando la función mean(), en donde al valor de la continua y se lo restamos al mensaje.

Si aplicamos esto, con una entrada modulada en FM como es en este caso, la salida del mensaje $y_D(t)$ es:

$$y_D(t) = 2\pi A k_f M(t)$$

Por lo tanto la ganancia del detector es:

$$k_D = 2\pi A$$

Esto no es bueno ya que la constante del discriminador en frecuencia varía con la amplitud de la señal, ya que si el transmisor y el receptor se alejan va cambiar el valor de la pendiente del discriminador. Para solucionar esto se puede colocar un limitador, pero no es la única solución posible. Para este caso no voy a considerar esto en el código de octave, pero en la realidad es algo importante a tener en cuenta.

Se puede observar en la figura 11 la DEP de la señal demodulada (con frecuencia normalizada) utilizando el método anteriormente descrito.

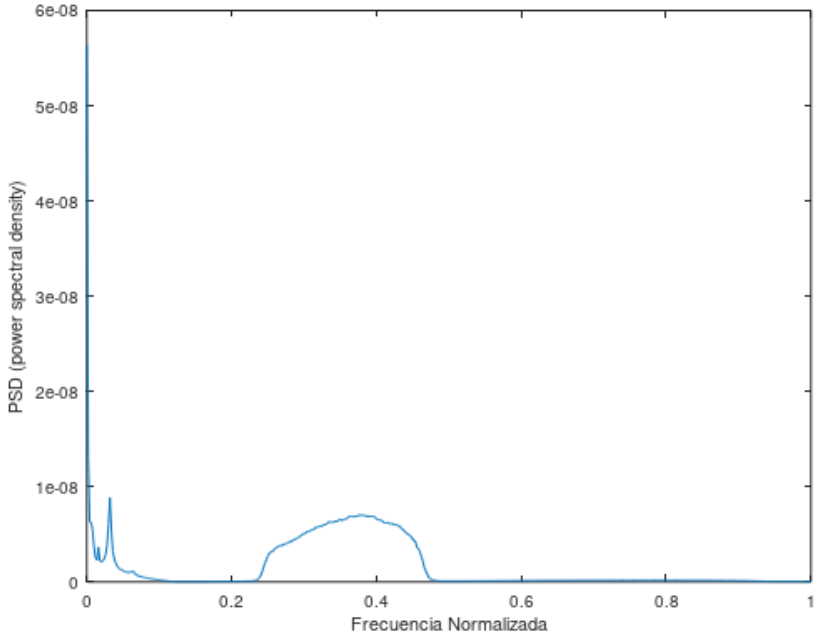

Figura 11: DEP de la señal demodulada con frecuencia normalizada.

Luego de demodular la señal, aplicamos nuevamente un filtro digital para el filtrado en banda base, se puede ver en la figura 12 el filtro aplicado y en la figura 13 la señal filtrada.

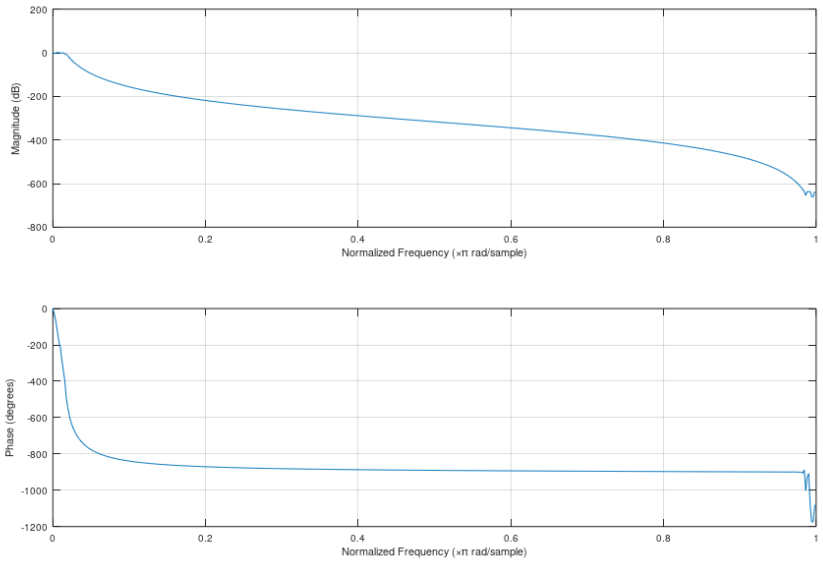

Figura 12: Módulo y fase del filtro para banda base.

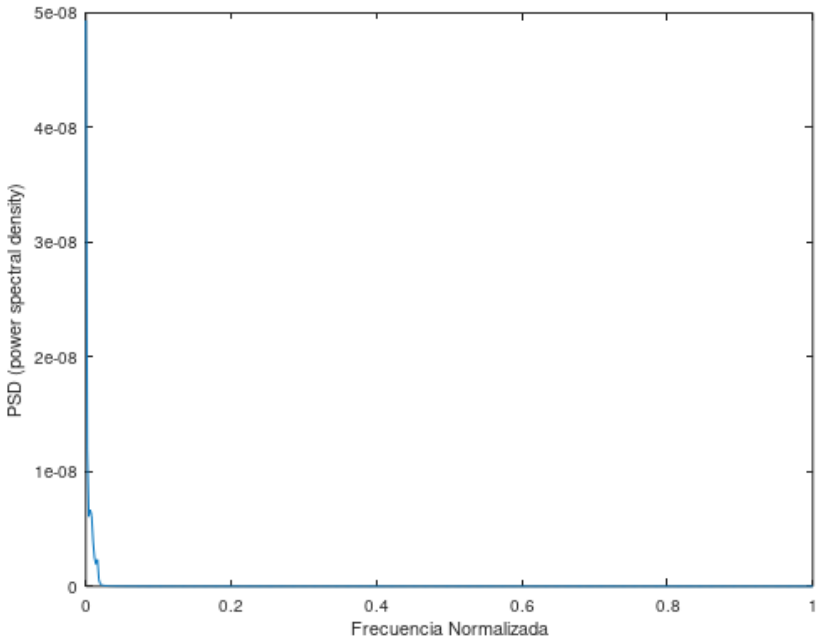

Figura 13: DEP de la señal demodulada filtrada.

Por último aplicamos un diezmado, ya que muchas placas de audio no reproducen cualquier frecuencia de muestreo. El diezmado consiste en reducir ancho de banda (filtrado) y la reducción de

la frecuencia de muestreo. Se puede ver en la figura 14 como trabaja el diezmado, en donde hay una señal $Y_{B1}[n]$ que entra al sistema de diezmado y sale una señal $Y_{N1}[m]$ en donde esta última está muestreada a una $f_{sYN1} = f_s/(2 * N_1)$.

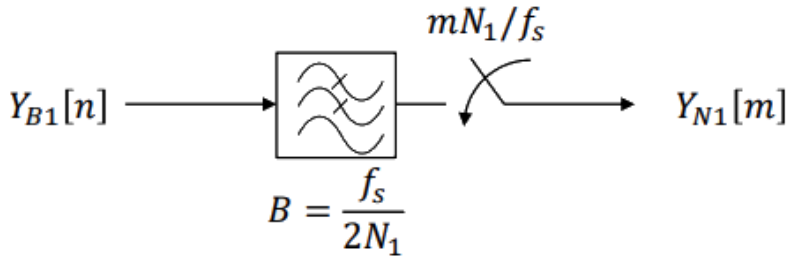

Figura 14: Diagrama en bloques funcionamiento diezmado.

Lo primero que hace el sistema de diezmado es filtrar la señal a un ancho de banda B y luego toma una muestra cada N_1. El filtro se piensa como un filtro antialiasing.

Por lo tanto si las placas de sonido trabajan bien para una frecuencia de muestreo de 48 kHz, teniendo una frecuencia de muestreo de 2.4 MHz debemos aplicar un diezmado de 25 para lograr llegar a esas magnitudes. En la figura 15 se ve el bloque de diezmado aplicado y en la figura 16 se puede ver la DEP de la señal de salida, después de aplicar el diezmado.

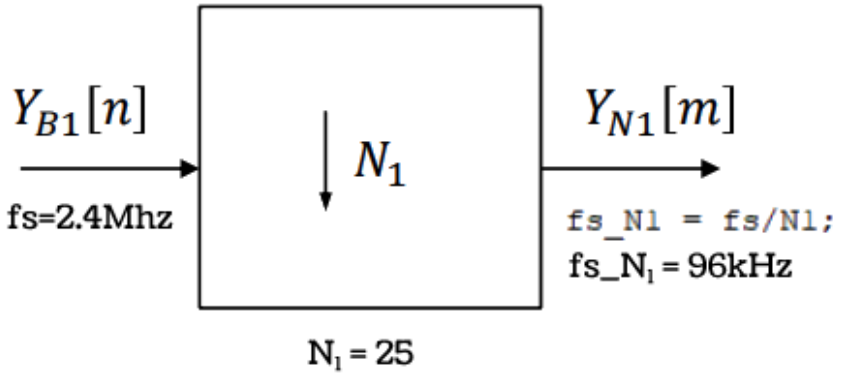

Figura 15: Bloque de diezmado aplicado.

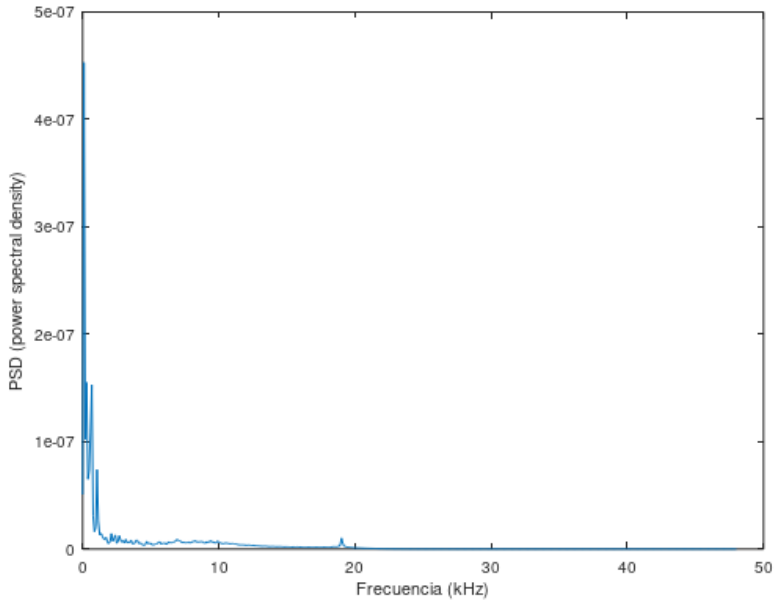

Figura 16: DEP de la señal con el diezmado aplicado.

Si graficamos la señal de audio demodulada obtenemos la gráfica de la figura 17.

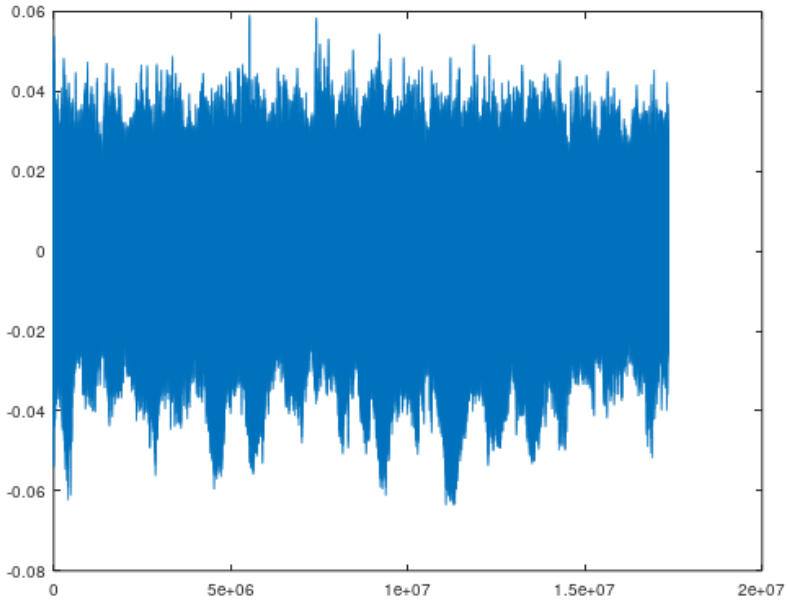

Figura 17: Señal de audio demodulada.

3. Validación

En la carpetas audios se encuentran los siguientes archivos:

- **señal_original**: este audio se obtuvo de grabar la transmisión de una estación específica con el software de Audacity en el SDRSharp.
- **señal_modulada_fm**: son las muestras capturadas utilizando el plugin "IF Recorder".
- **señal_demodulada**: es la señal demodulada que se obtiene de ejecutar FM_DEMOD_PEREYRA() y la que se reproduce en el archivo main en octave.

Con la función de octave audioplayer() podemos escuchar la señal demodulada, con un poco de ruido, pero se logra distinguir que es una de las transmisiones capturadas con el Dongle.