RTL-SDR

▼ **Sobre el RTL-SDR**

https://www.rtl-sdr.com/about-rtl-sdr/

Parámetro	Descripción
Case	Aluminio
Interface USB IC	RTL2832U
Tuner IC	R820T2
Rango de frecuencia	25 MHz a 1750 MHz
Ancho de banda	1 MHz a 3 MHz
Resolución	8 bits
Canales TX	0
Canales RX	1
TCXO clock	Sí
Tipo de conector de la antena	SMA Hembra

▼ Calibración

El receptor RTL-SDR, caracterizado por tener tolerancias de ±30 a ±50 ppm, este parámetro varía en función del fabricante y modelo de RTL-SDR por lo que es necesario calibrar.

El calibrado de un dispositivo SDR consiste en calcular la desviación en frecuencia medida en ppm de forma que se ajuste el oscilador para la sintonización a la frecuencia RF deseada. El receptor SDR se calienta a medida que aumenta el tiempo de trabajo por lo que es recomendable dejar en funcionamiento el dispositivo cierto tiempo antes de calibrarlo, por ejemplo unos 20 minutos.

Software kalibrate

El Software Open-Source kalibrate es una aplicación de consola disponible en los sistemas operativos GNU/Linux y Windows, la cual se puede descargar a partir de los siguietes comandos:

```
$ git clone https://github.com/steve-m/kalibrate-rtl.git
$ sudo apt-get install libtool libfftw3-dev
```

Debido a que se trata de una aplicación de consola se trabaja desde la terminal del sistema operativo utilizado, accediendo al directorio donde se descargo el archivo. Luego, ingresando el comando `kal -h` se despliega las diferentes acciones que se pueden realizar.

```
apavet@becarios206:~/kalibrate-rtl$ kal -h
kalibrate v0.4.1-rtl, Copyright (c) 2010, Joshua Lackey
modified for use with rtl-sdr devices, Copyright (c) 2012, Steve Markgraf
Usage:
  GSM Base Station Scan:
    kal <-s band indicator> [options]

  Clock Offset Calculation:
    kal <-f frequency | -c channel> [options]

Where options are:
  -s  band to scan (GSM850, GSM-R, GSM900, EGSM, DCS, PCS)
  -f  frequency of nearby GSM base station
  -c  channel of nearby GSM base station
  -b  band indicator (GSM850, GSM-R, GSM900, EGSM, DCS, PCS)
  -g  gain in dB
  -d  rtl-sdr device index
  -e  initial frequency error in ppm
  -E  manual frequency offset in hz
  -v  verbose
  -D  enable debug messages
  -h  help
```

Para calibrar un dispositivo SDR se escanea la banda de frecuencia GSM900 mediante el comando `kal -g 42 -e 22 -s GSM900`.

https://www.instructables.com/Calibrando-SDR-RTL/ (de donde salen los pasos)

https://qso365.co.uk/how-to-calibrate-the-rtl-sdr-dongle-for-use-with-an-aprs-rx-only-igate-raspbian-stretch-version/

https://www.pe2kmv.nl/wp/en/diversen-en/calibrating-an-rtl-sdr-dongle/

```
apavet@becarios206:~/kalibrate-rtl$ kal -g 42 -e 22 -s GSM900
Found 1 device(s):
  0:  Generic RTL2832U OEM

Using device 0: Generic RTL2832U OEM
Found Rafael Micro R820T tuner
Exact sample rate is: 270833.002142 Hz
[R82XX] PLL not locked!
Setting gain: 42.0 dB
kal: Scanning for GSM-900 base stations.
GSM-900:
    chan:   12 (937.4MHz - 22.781kHz)    power:  427823.64
    chan:   17 (938.4MHz + 38.017kHz)    power:  439039.85
    chan:   33 (941.6MHz - 23.407kHz)    power:  425968.08
    chan:   35 (942.0MHz + 9.555kHz)     power:  816800.53
```

Tras el escaneo de canales, se observa que el canal recibido con mayor potencia es el canal 35, con 816800.53 W por lo que se recibirá la señal de dicho canal para obtener la desviación en frecuencia con el comando `kal -e 41 -c 35 -v`

```
apavet@becarios206:~/kalibrate-rtl$ kal -e 41 -c 35 -v
Found 1 device(s):
  0:  Generic RTL2832U OEM
```

```
Using device 0: Generic RTL2832U OEM
Found Rafael Micro R820T tuner
Exact sample rate is: 270833.002142 Hz
[R82XX] PLL not locked!
kal: Calculating clock frequency offset.
Using GSM-900 channel 35 (942.0MHz)
Tuned to 942.000000MHz (reported tuner error: 0Hz)
  offset   1: 28108.20
  offset   2: 36089.25
  offset   3: 20156.07
  offset   4: 21548.76
  offset   5: 16052.42
  offset   6: 10672.83
  offset   7: 20692.28
  offset   8: 30518.53
  offset   9: 30326.36
  offset  10: -452.08
  offset  11: 17833.56
  offset  12: 21557.03
  offset  13: 24491.16
  offset  14: 16556.60
  offset  15: 15702.18
  offset  16: 11575.80
  offset  17: 14084.28
  offset  18: 27725.94
  offset  19: 22900.11
  offset  20: 14491.34
  offset  21: 15717.68
  offset  22: 38111.12
  offset  23: 23622.28
  offset  24: 27732.13
  offset  25: 16968.82
  offset  26: 13144.11
  offset  27: 21118.97
  offset  28: 27324.04
  offset  29: 13642.09
  offset  30: 23497.27
  offset  31: 14712.43
  offset  32: 26723.78
  offset  33: 31199.37
  offset  34: 20626.16
  offset  35: 14803.35
  offset  36: 20287.28
  offset  37: 22536.45
  offset  38: 18251.99
  offset  39: 11502.45
  offset  40: 12187.42
  offset  41: 39579.22
  offset  42: 10903.22
  offset  43: 16477.04
  offset  44: 26861.19
  offset  45: 10335.00
  offset  46: 29893.48
  offset  47: 14492.37
  offset  48: 20324.48
  offset  49: 9841.15
  offset  50: 27146.34
  offset  51: 15293.06
  offset  52: 15009.98
  offset  53: 11540.67
  offset  54: 28077.21
  offset  55: 8559.01
  offset  56: 23034.42
  offset  57: 28627.87
  offset  58: 30270.57
  offset  59: 28562.78
  offset  60: 16190.86
  offset  61: 28885.13
  offset  62: 23236.92
  offset  63: 22436.23
  offset  64: 21441.31
  offset  65: 15650.53
  offset  66: 15129.82
  offset  67: 15994.57
  offset  68: 6396.64
```

```
   offset  69: 22980.70
   offset  70: 21636.57
   offset  71: 23069.55
   offset  72: 24844.49
   offset  73: 22736.88
   offset  74: 14662.84
   offset  75: 22090.13
   offset  76: 8135.42
   offset  77: -4770.63
   offset  78: 22113.88
   offset  79: 18391.46
   offset  80: 9814.28
   offset  81: 26118.36
   offset  82: 22525.08
   offset  83: 13916.91
   offset  84: 5981.31
   offset  85: 28735.32
   offset  86: 16451.21
   offset  87: 24973.63
   offset  88: 23437.35
   offset  89: 23026.16
   offset  90: 27730.07
   offset  91: -24.36
   offset  92: 16359.27
   offset  93: 25898.30
   offset  94: 27256.88
   offset  95: 14533.70
   offset  96: 14843.64
   offset  97: 23017.89
   offset  98: 16614.46
   offset  99: 18300.55
   offset 100: 24542.81
average   [min, max]  (range, stddev)
+ 19.963kHz   [10673, 28628]  (17955, 5098.722656)
overruns: 0
not found: 1137
average absolute error: 19.808 ppm
```

De este modo, puede observarse que el receptor RTL-SDR tiene una desviación en frecuencia de 19.808 ppm.

▼ Recepción con algoritmo (Py)

Respecto a la herramienta Software utilizada para el diseño del algoritmo de decodificación se ha escogido trabajar en el lenguaje Python puesto que incorpora gran cantidad de funciones que permiten el análisis, procesado y extracción de características de señales con muy buenas prestaciones.

En primera instancia se deben instalar bibliotecas y herramientas de desarrollo necesarias desde la terminal:

1. Instalar el paquete de Python `sudo apt install python3`
2. Instalar las dependencias del sistema `sudo apt-get install libusb-1.0-0-dev`
3. Instalar las herramientos de RTL-SDR `sudo apt-get install rtl-sdr`
   Si surgen problemas de compilacion se puede revisar: rtl-sdr installation
4. Instalar pyrtlsdr `pip install pyrtlsdr`

La IDE que se va a utilizar para desarrollar el codigo es Visual Studio Code, que para su instalacion debe descargarse el pacage de VSC y ejecutar en la terminal: `sudo dpkg -i archivo.deb`

Aqui se incluye el código en Python implementado para la recepcion de muestras del RTL-SDR utilizando la libreria rtlsdr.

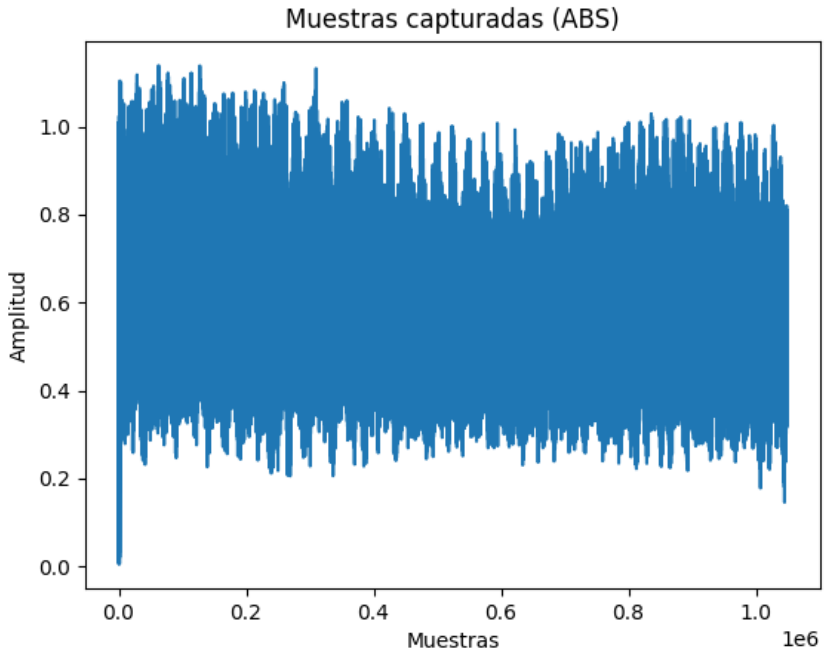
Se centra la frecuencia en la emisora 99.9 FM para trabajar en una señal conocida.

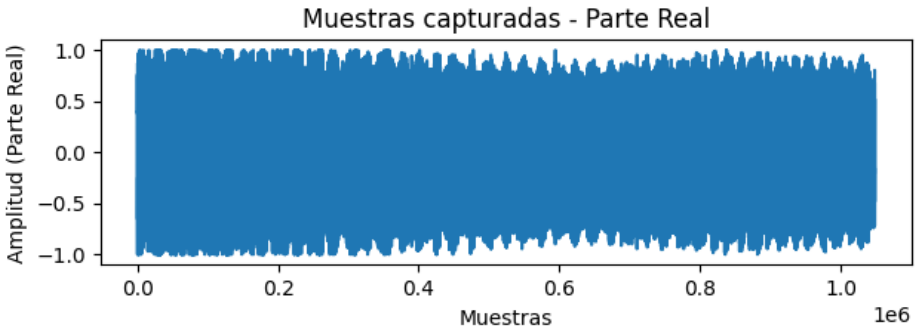
▼ Código

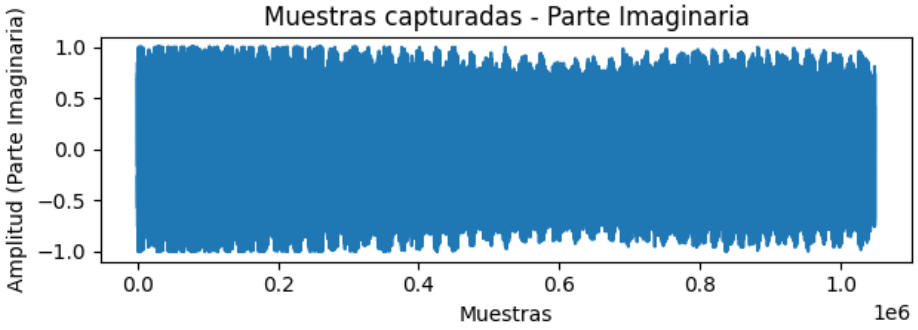
```
import numpy as np
import rtlsdr
import matplotlib.pyplot as plt

# -------------------------- RTL-SDR Configuration ----------------------------------------

sdr = rtlsdr.RtlSdr()                       # Inicializar el dispositivo RTL-SDR

sample_rate = 2.4e6                         # Tasa de muestreo en muestras por segundo (3MHz)
                                            # The maximum sample rate is 3.2 MS/s (mega samples per second).
                                            # However, the RTL-SDR is unstable at this rate and may drop samples.
                                            # The maximum sample rate that does not drop samples is 2.56 MS/s.

# Configurar parámetros
# Rafael Micro R820T/2/R860   24 – 1766 MHz (Can be improved to ~13 - 1864 MHz with experimental drivers)

sdr.sample_rate = sample_rate               # Tasa de muestreo en muestras por segundo
sdr.center_freq = 99900000                  # Frecuencia central en Hz (99.9 MHz)
sdr.gain = 42                               # Ganancia del receptor

output_file = "fm999.bin"                   # Nombre del archivo de salida con las muestras capturadas

block_size = 1024 * 1024                    # Tamaño de cada bloque de muestras
samples_buffer = []                         # Buffer para almacenar las muestras

# ------------------------------------------------------------------------------------------------------

# ------------------------ Captura y procesamiento de muestras en bloques ---------------------------------

num_blocks = 1                                          # Capturar 1 bloques de muestras

for _ in range(num_blocks):

    samples = sdr.read_samples(block_size)              # Capturar bloque de muestras
    np.save(output_file, samples, allow_pickle=False)   # Guardar las muestras en el archivo binario
    samples_buffer.extend(samples)                      # Agregar las muestras al buffer

    # Mantener el buffer dentro del tamaño deseado
    if len(samples_buffer) > block_size * num_blocks:
        samples_buffer = samples_buffer[-block_size * num_blocks:]


sdr.close()                                 # Cerrar el dispositivo RTL-SDR

signal_energy = np.sum(np.abs(samples_buffer)**2) / len(samples_buffer)    # Calcular la energía de la señal

threshold = 1e-8                            # Umbral de energía de la señal

information_present = signal_energy > threshold # Verificar si hay información en las muestras

if information_present:
    print("Se detectó información en las muestras.")
else:
    print("No se detectó información en las muestras.")

# ------------------------------------- Ploteos ------------------------------------------------------

# Muestras capturadas
plt.plot(np.abs(samples_buffer))
plt.xlabel('Muestras')
plt.ylabel('Amplitud')
plt.title('Muestras capturadas (ABS)')
plt.show()

# Muestras capturadas (Parte Real)
plt.subplot(2, 1, 1)
plt.plot(np.real(samples_buffer))
plt.xlabel('Muestras')
plt.ylabel('Amplitud (Parte Real)')
plt.title('Muestras capturadas - Parte Real')

# Muestras capturadas (Parte Imaginaria)
plt.subplot(2, 1, 2)
plt.plot(np.imag(samples_buffer))
```

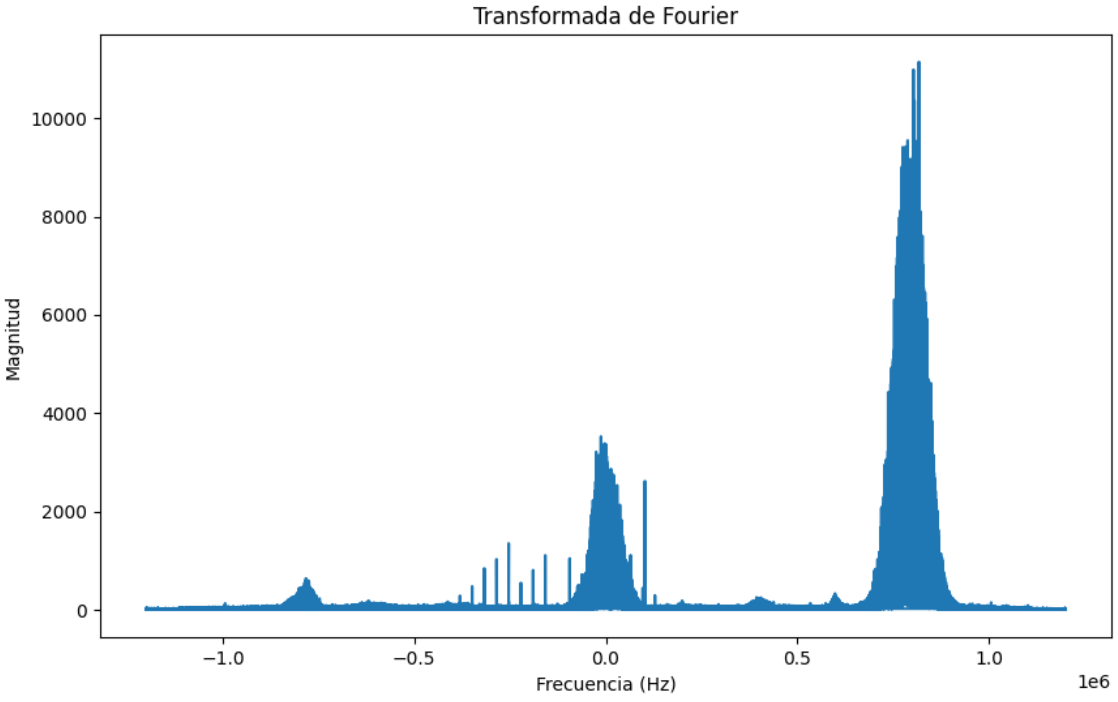
```
plt.xlabel('Muestras')
plt.ylabel('Amplitud (Parte Imaginaria)')
plt.title('Muestras capturadas - Parte Imaginaria')

plt.tight_layout()
plt.show()

# Transformada de Fourier y frecuencias correspondiente de las muestras en el dominio de la frecuencia
fft_result = np.fft.fft(samples_buffer)
magnitud_fft = np.abs(fft_result)
frecuencias = np.fft.fftfreq(len(samples_buffer), d=1/sample_rate)

# Grafica de la transformada de Fourier
plt.figure(figsize=(10, 6))
plt.plot(frecuencias, magnitud_fft)
plt.title('Transformada de Fourier')
plt.xlabel('Frecuencia (Hz)')
plt.ylabel('Magnitud')
plt.show()

# Espectro de frecuencia
frequencies, spectrum = plt.psd(samples_buffer, NFFT=1024, Fs=sample_rate, scale_by_freq=True)
plt.xlabel('Frecuencia (Hz)')
plt.ylabel('Densidad espectral de potencia')
plt.title('Espectro de frecuencia')
plt.show()
```

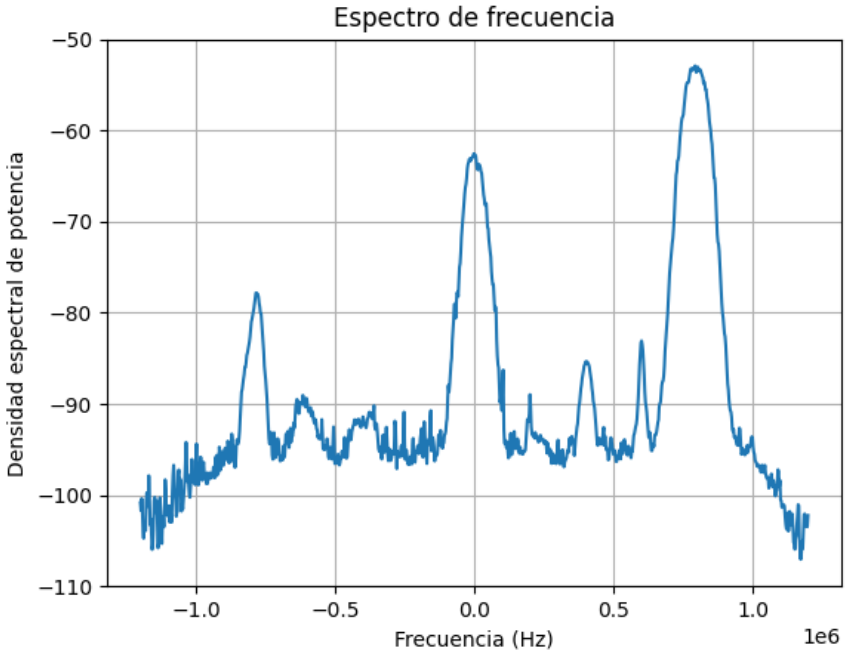
https://tech.integratekmexico.com/sincategoria/transformada-de-fourier-como-analizar-las-senales/




Muestras capturadas - Parte Real
Amplitud (Parte Real)
1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
1.0
Muestras
1e6

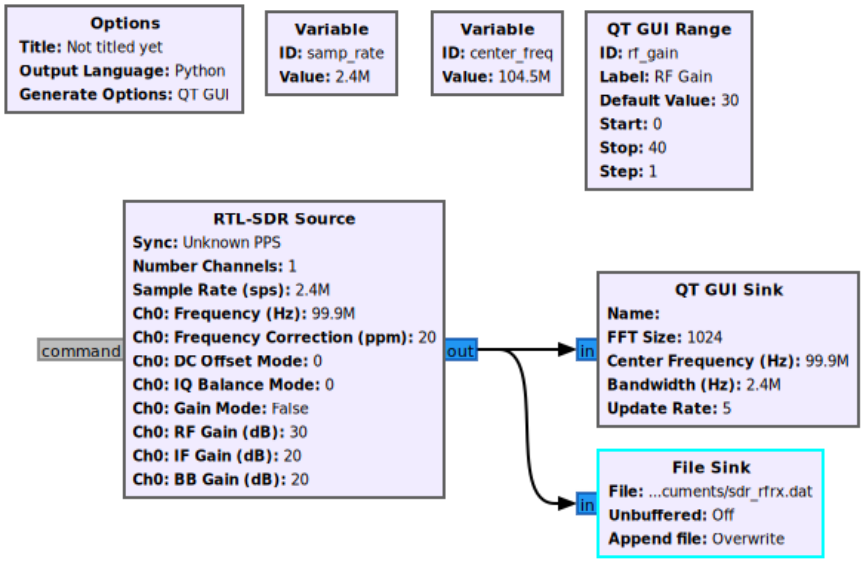

Muestras capturadas - Parte Imaginaria
Amplitud (Parte Imaginaria)
1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
1.0
Muestras
1e6


Transformada de Fourier
Magnitud
10000
8000
6000
4000
2000
0
−1.0
−0.5
0.0
0.5
1.0
Frecuencia (Hz)
1e6



▼ Recepción con GNU Radio

Se centra la frecuencia en la emisora 99.9 FM para trabajar en una señal conocida.

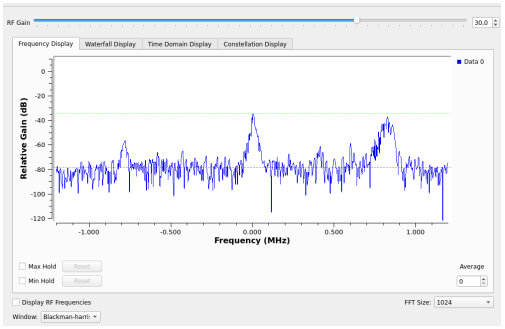


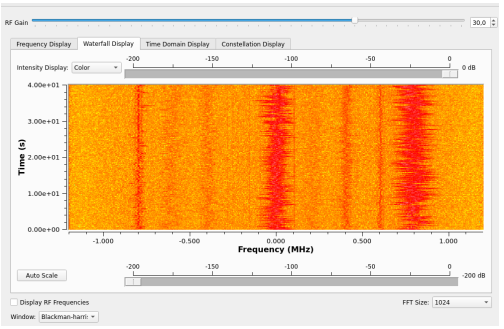
Esquemático de receptor.

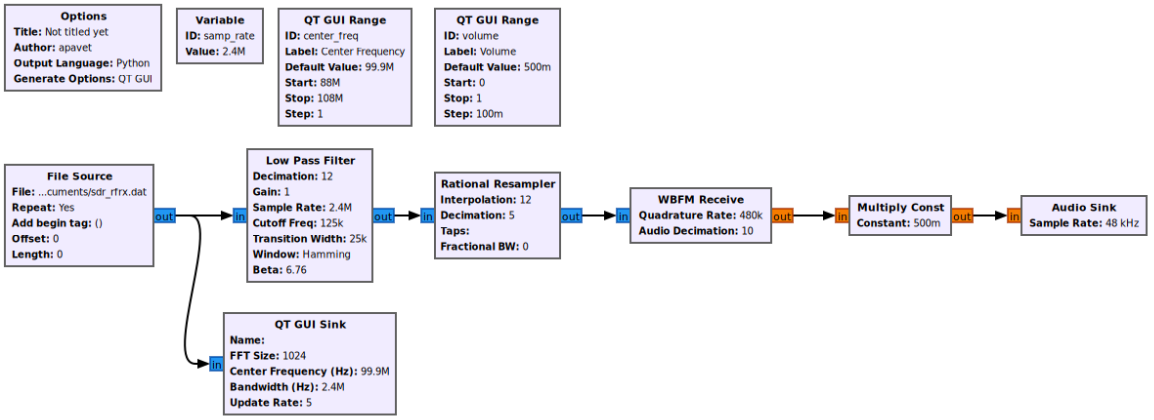
Cada muestra que se toma, son datos flotantes de 32 bits, por lo que el rtl produce 8 fragmentos de 8 bits; 2 bytes de i y 2 bytes de q, un total de 4 bytes por muestra, y en total 32 bytes.

Frecuencia central variable para que sea el centro de la señal que se desea medir. Ademas configure un rango de interfaz gráfica de usuario variable para la ganancia.

Luego el bloque de sincronizacion de archivos del tipo c de 32 bits.







Esquema de lectura de la recepción.

Se trabaja sobre la misma frecuencia de muestreo, misma frecuencia central y se toma el archivo con los datos capturados. Ya que es una FM de banda ancha, entonces es necesarion un modulador FMD para escuchar lo capturado.

https://m.youtube.com/watch?v=iDA8MShO6Uo analizador de espectro