Dual FM Receiver Report
ECE 447 Communications
November 1, 2024

C1C D Benjamin Cometto

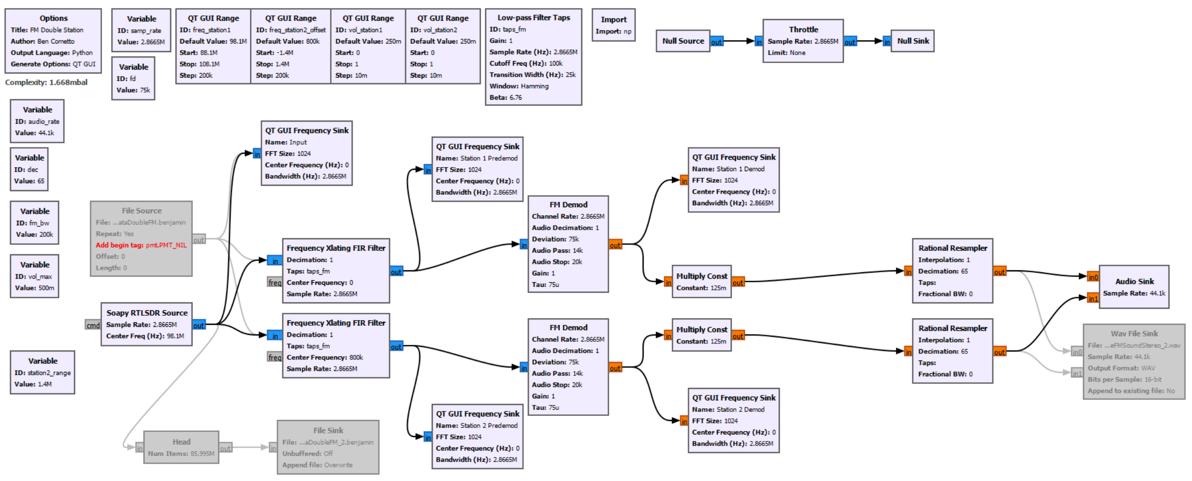
Introduction

While it is common to receive one FM radio station, it is often desired to listen to multiple songs (or, more likely, advertisements) simultaneously. This need motivated the development of the dual FM receiver. This receiver can be used to monitor two FM channels at the same time, a vital feature of any radio to be used in a combat environment.

This dual FM receiver is implemented in GNU Radio, and uses the RTL-SDR software defined radio. It can listen to any any channel in the FM radio band ($88\,\text{MHz}$ to $108\,\text{MHz}$) simultaneously with a second channel within $1.4\,\text{MHz}$ of the first. Each channel has individual volume controls. Additionally, when the capabilities exist, the receiver operates with stereo audio: one station is sent to the left channel and one is sent to the right channel, allowing better monitoring of disparate audio.

1 Flowgraph Description

The GNU Radio flowgraph for the dual FM receiver can be seen in Figure 1. The first channel (station 1) is set using `freq_station1`, which can tune to any FM frequency. The second channel (station 2) is set using an offset from the first. The sample rate limits the range of this second channel to one-half of `samp_rate`, allowing an offset of $\pm 1.4\,\text{MHz}$, set in increments of $200\,\text{kHz}$ for useability.


Options
Title: FM Double Station
Author: Ben Cometto
Output Language: Python
Generate Options: QT GUI
Complexity: 1.668mbal

Variable
ID: samp_rate
Value: 2.8665M

Variable
ID: fd
Value: 75k

QT GUI Range
ID: freq_station1
Default Value: 98.1M
Start: 88.1M
Stop: 108.1M
Step: 200k

QT GUI Range
ID: freq_station2_offset
Default Value: 800k
Start: -1.4M
Stop: 1.4M
Step: 200k

QT GUI Range
ID: vol_station1
Default Value: 250m
Start: 0
Stop: 1
Step: 10m

QT GUI Range
ID: vol_station2
Default Value: 250m
Start: 0
Stop: 1
Step: 10m

Low-pass Filter Taps
ID: taps_fm
Gain: 1
Sample Rate (Hz): 2.8665M
Cutoff Freq (Hz): 100k
Transition Width (Hz): 25k
Window: Hamming
Beta: 6.76

Import
Import: np

Null Source → Throttle (Sample Rate: 2.8665M, Limit: None) → Null Sink

Variable
ID: audio_rate
Value: 44.1k

Variable
ID: dec
Value: 65

Variable
ID: fm_bw
Value: 200k

Variable
ID: vol_max
Value: 500m

Variable
ID: station2_range
Value: 1.4M

File Source
File: ...ataDoubleFM.benjamin
Repeat: Yes
Add begin tag: pmt.PMT_NIL
Offset: 0
Length: 0

Soapy RTLSDR Source
Sample Rate: 2.8665M
Center Freq (Hz): 98.1M

Head
Num Items: 85.995M

File Sink
File: ...aDoubleFM_2.benjamin
Unbuffered: Off
Append file: Overwrite

QT GUI Frequency Sink
Name: Input
FFT Size: 1024
Center Frequency (Hz): 0
Bandwidth (Hz): 2.8665M

Frequency Xlating FIR Filter
Decimation: 1
Taps: taps_fm
Center Frequency: 0
Sample Rate: 2.8665M

Frequency Xlating FIR Filter
Decimation: 1
Taps: taps_fm
Center Frequency: 800k
Sample Rate: 2.8665M

QT GUI Frequency Sink
Name: Station 1 Predemod
FFT Size: 1024
Center Frequency (Hz): 0
Bandwidth (Hz): 2.8665M

QT GUI Frequency Sink
Name: Station 2 Predemod
FFT Size: 1024
Center Frequency (Hz): 0
Bandwidth (Hz): 2.8665M

FM Demod
Channel Rate: 2.8665M
Audio Decimation: 1
Deviation: 75k
Audio Pass: 14k
Audio Stop: 20k
Gain: 1
Tau: 75u

FM Demod
Channel Rate: 2.8665M
Audio Decimation: 1
Deviation: 75k
Audio Pass: 14k
Audio Stop: 20k
Gain: 1
Tau: 75u

QT GUI Frequency Sink
Name: Station 1 Demod
FFT Size: 1024
Center Frequency (Hz): 0
Bandwidth (Hz): 2.8665M

QT GUI Frequency Sink
Name: Station 2 Demod
FFT Size: 1024
Center Frequency (Hz): 0
Bandwidth (Hz): 2.8665M

Multiply Const
Constant: 125m

Multiply Const
Constant: 125m

Rational Resampler
Interpolation: 1
Decimation: 65
Taps:
Fractional BW: 0

Rational Resampler
Interpolation: 1
Decimation: 65
Taps:
Fractional BW: 0

Audio Sink
Sample Rate: 44.1k

Wav File Sink
File: ...eFMSoundStereo_2.wav
Sample Rate: 44.1k
Output Format: WAV
Bits per Sample: 16-bit
Append to existing file: No

Figure 1: Dual FM Receiver Flowgraph

After receiving the raw data from the RTL-SDR, the flowgraph selects each wanted signal using a translating low pass filter. The first filter (for station 1) is centered around $0\,\text{Hz}$, the center frequency of the RTL-SDR source. The second filter is centered around the chosen offset.

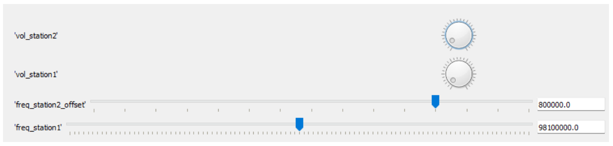
Then, each signal is demodulated, using the standard FM frequency deviation f_d of $75\,\text{kHz}$. At this point, each signal is multiplied by the volume control, a constant between 0 and 0.5. Finally, the signals are decimated down to the audio rate of $44.1\,\text{kHz}$ and played through a stereo audio sink.

Notice additionally that the flowgraph has the capability to record the raw RTL-SDR data, playback raw RTL-SDR data, and record the output audio to a `.wav` file.

2 Features

A main consideration was a desire to be able to listen to any two FM stations. However, the maximum sampling frequency f_s of the RTL-SDR is $3.2\,\text{Msps}$. This limits the ability of the second station to be at any frequency. Thus, only the first station is unconstrained, while the second is limited to be within approximately $f_s/2$ of the first station.

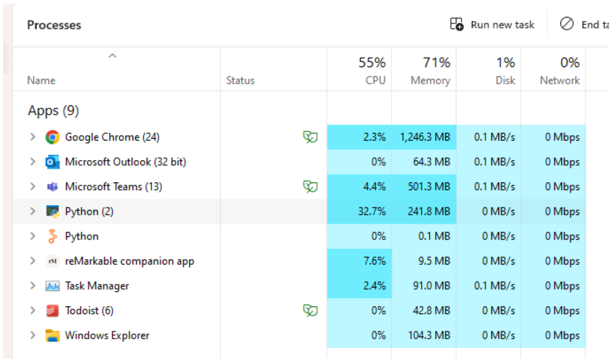
The system is easily controlled using the QT GUI. The GUI features a slider/entry for station 1's frequency and the offset for station 2 and volume knobs for each channel. These can be seen in Figure 2. Additionally, the GUI features several analytical graphs, including the frequency spectrums of the RTL-SDR data and each station before and after modulation.



'vol_station2'
'vol_station1'
'freq_station2_offset' 800000.0
'freq_station1' 98100000.0

Figure 2: The receiver's QT GUI control knobs

A problem with the current iteration of the system is the large CPU requirements. Running the RTL-SDR with $f_s = 2.8\,\text{MHz}$ requires significant CPU computing power, which can be seen in Figure 3.



Name	Status	55% CPU	71% Memory	1% Disk	0% Network
Apps (9)					
Google Chrome (24)		2.3%	1,246.3 MB	0.1 MB/s	0 Mbps
Microsoft Outlook (32 bit)		0%	64.3 MB	0.1 MB/s	0 Mbps
Microsoft Teams (13)		4.4%	501.3 MB	0.1 MB/s	0 Mbps
Python (2)		32.7%	241.8 MB	0 MB/s	0 Mbps
Python		0%	0.1 MB	0 MB/s	0 Mbps
reMarkable companion app		7.6%	9.5 MB	0 MB/s	0 Mbps
Task Manager		2.4%	91.0 MB	0.1 MB/s	0 Mbps
Todoist (6)		0%	42.8 MB	0 MB/s	0 Mbps
Windows Explorer		0%	104.3 MB	0 MB/s	0 Mbps

Figure 3: Windows Task Manager showing GNU Radio (Python)'s CPU usage

This leads to occasional computer-wide slowdowns and audio skips/lag.

Documentation

I did my own work, using the GNU Radio wiki and a few google searches related to commercial FM radio, such as for the frequency deviation and common stations in Colorado Springs.