Week 3 Lab: FM Radio Receiver

In this weeks lab we'll use Software Defined Radio (SDR) and GNU Radio Companion (GRC) to make our own FM radio receiver and explore important concepts in digital signal processing. Before completing this lab, please review the pre-lab reading which will provide detailed background on the topics covered, as well as example GRC flowgraphs.

Creating the GRC Flowgraph

1. Open GNU Radio Companion (GRC) on your computer and ensure that a new flowgraph is open.

2. Using the search feature (magnifying glass icon on the top bar), find the **Variable** block and add four of these blocks to your workspace. You'll use these variables later in the flowgraph so it's good to have them ready. Double click on the block to open the properties window. Make the following modifications to the variable blocks:

   (a) ID: samp_rate, Value: 2200000 or 2.2e6

   (b) ID: cutoff, Value: 100000 or 1e5

   (c) ID: transition, Value: 1000000 or 1e6

   (d) ID: audio_dec, Value: 10

   The Variable blocks allow us to input values under an ID that can be used in other blocks. You could simply hard code the values you want in each block, but using variables makes it easier to change a value, as we only have to change the value in one place instead of finding every place it's used in the flowgraph.

3. Search for the **QT GUI Range** block and add two of these blocks to your workspace. Make the following modifications:

   (a) ID: freq, Label: Center Frequency, Default Value: 90e6, Start: 85e6, Stop: 110e6, Step: 1e5

(b) ID: volume, Label: Volume, Default Value: 0.3, Start: 0, Stop: 1, Step: 0.01

These blocks will allow us to adjust the frequency we're tuned to, and the volume, while the flowgraph is running.

4. Now you're ready to add the block that will receive data from the RTL-SDR. There are two options for how to do this. If you or a small group each has their own RTL-SDR, then add the **RTL-SDR Source** block. If your instructor has the RTL-SDR, the data from it can be sent through the internet and pickup by the **ZMQ SUB Source** block. Make the following modifications based on which block you're using.

   (a) **RTL-SDR Source:** Device Arguments: "rtl=0", Sample Rate: samp_rate, Ch0 Frequency: freq, Ch0 Gain Mode: False, Ch0 RF Gain: 20, Ch0 IF Gain: 20, Ch0 BB Gain: 20

   This block interfaces with the RTL-SDR device to capture the FM signal as an IQ (In-phase and Quadrature) stream.

   (b) **ZMQ SUB Source:** IO Type: Complex, Vector Length: 1, Address: ask your instructor. If you use this block then you will not be able to adjust the center frequency, you're instructor will do this on their computer.

5. Add a **Low Pass Filter** block to the right of your source block. Connect the source and Low Pass Filter blocks together either by clicking and holding on the out port and dragging to the in port, or by clicking on the out port then clicking on the in port. The connecting arrow should be black, if it is red then the program is indicating an error, usually that the two blocks are using different data types. A list of data types and their corresponding colors can be found in the top menu under Help → Types. Make the following modifications to the Low Pass Filter block:

   (a) FIR Type: Complex→ Complex (Decimating), Decimation: int(samp_rate/5e5), Gain: 1, Sample Rate: samp_rate, Cutoff Freq: cutoff, Transition Width: transition, Window: Hamming, Beta: 6.76

   The Low Pass Filter block removes high frequency components of the signal, such as the carrier frequency, as well as other noise, which isolates the baseband modulated signal.

6. Add a **WBFM Receive** block and connect it with the Low Pass Filter block. Make the following modifications:

   (a) Quadrature Rate: 5e5, Audio Decimation: audio_dec

   In signal processing, one common method for the FM demodulation procedure,

as shown in the pre-lab reading, is called a phase-locked loop (PLL). An analog PLL circuit consists of a phase detector, a loop filter, and a voltage-controlled oscillator (VCO). The phase detector compares the phase of the received signal with a reference signal. The loop filter filters the phase error signal to produce a control voltage for the VCO. The VCO generates an output signal whose frequency is proportional to the control voltage. The output of the VCO is the demodulated baseband signal $m(t)$. The WBFM Receive block does the digital version of this, all in a single block! This block does most of the heavy lifting involved in extracting the modulating signal $m(t)$ from the received signal.

Note that the WBFM Receive block has a complex data type input (blue port) and a float data type output (orange port). This is what we want and is not an error.

7. Add a **Rational Resampler** block to your workspace and connect it to the WBFM Receive block. Make the following modifications:

   (a) Type: Float→Float (Real Taps), Interpolation: 48, Decimation: 50, Taps: [ ], Fractional BW: 0

   This block adjusts the sample rate of the demodulated signal to match the desired output sample rate.

8. Add a **Multiply Const** block to the workspace and connect it to the Rational Resampler block. Make the following modifications:

   (a) IO Type: float, Constant: volume, Vector Length: 1

   This block allows you to control the volume of the audio output by multiplying the signal by a scaling factor. Recall in the third step, we added a QT GUI Range block with the ID volume, which creates a slider we can use to adjust the value of the scaling factor while the flowgraph is running.

   The next three block can be added to the workspace vertically, directly above or below each other. These blocks are input only and are the end of the flowgraph.

9. Add an **Audio Sink** block and connect it to the Multiply Const block. Make the following modifications:

   (a) Sample Rate: 48 kHz, Device name: for most devices this can be left blank, OK to Block: Yes.

   This block outputs the audio signal to your computers sound card or audio

device for playback.

10. Add a **QT GUI Waterfall Sink** block and connect it to the Multiply Const block. Make the following modifications:

    (a) FFT Size: 2048, Center Frequency: freq, Bandwidth: samp_rate

    This block allows us to visualize the audio signal. The x-axis is frequency, centered on your chosen center frequency, and the y-axis is time. The color coding represents the amplitude of the signal at each frequency, with blue being a small amplitude and red being a large amplitude.

11. Add a **QT GUI Frequency Sink** block and connect it to the Multiply Const block. Make the following modifications:

    (a) FFT Size: 2048, Center Frequency: freq, Bandwidth: samp_rate

    This block also allows you to visualize the real time audio signal. It essentially computes and displays the fast Fourier transform (FFT) of the incoming signal. The x-axis is frequency and the y-axis is a proxy of amplitude called relative gain, in units of decibels (dB).

12. Your flowgraph is now complete and should look like this:

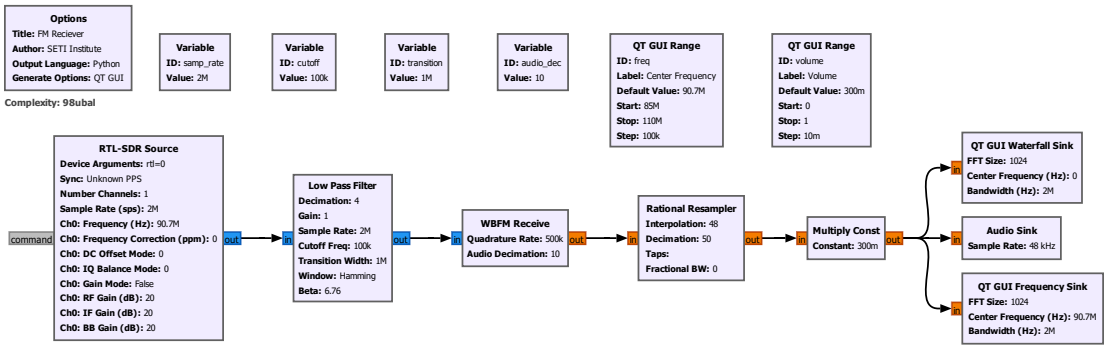

Figure 1: FM receiver flowgraph in GNU Radio

Run the flowgraph using the play button on the top menu. Look up FM radio stations near you and set the center frequency to the desired station. Try to find a station with 10-15 miles of your location for clearer audio and less noise. Remember the FM broadcast band is from roughly 85MHz to 110MHz. For example, if you're tuning into the 96.9 FM station, your center frequency will be 96.9e6Hz or 96,900,000Hz.

Post Lab Questions

1. Recall that in step 2, we set the sample rate of the received signal to 2.2MHz. Given the FM broadcast band and what you know of the Nyquist theorem, why does this sample rate make sense? What do you think will happen if the sample rate was decreased or increased? Try this in your flowgraph and record your findings below.

2. Human hearing can detect frequencies as high as about 20khZ. Given this, what sample rate should be used to process speech and music? Why does the sample rate we chose in the Audio Sink block make sense?

3. With regards to the carrier wave frequency and the modulating frequencies that go into the transmitted signal, what is the Low Pass Filter used for?