Lab 4 - Analog Modulation with SDR

ECE531 – Software Defined Radio

Spring 2022

Rahe Gerson, rmizrahi@email.arizona.edu

Table of Contents

Table of Figures

1 Introduction.

In this laboratory, we examine how analog modulation is implemented with software defined radio. We primarily investigate FM modulation because the Pluto cannot receive broadcast AM signals; the frequency bands of broadcast AM are below what the Pluto can receive. Throughout the reception process, it is important to make sure that sample rates match throughout the blocks. We also need to pay careful attention to filter widths to avoid interference.

1.1 Amplitude Modulation – AM

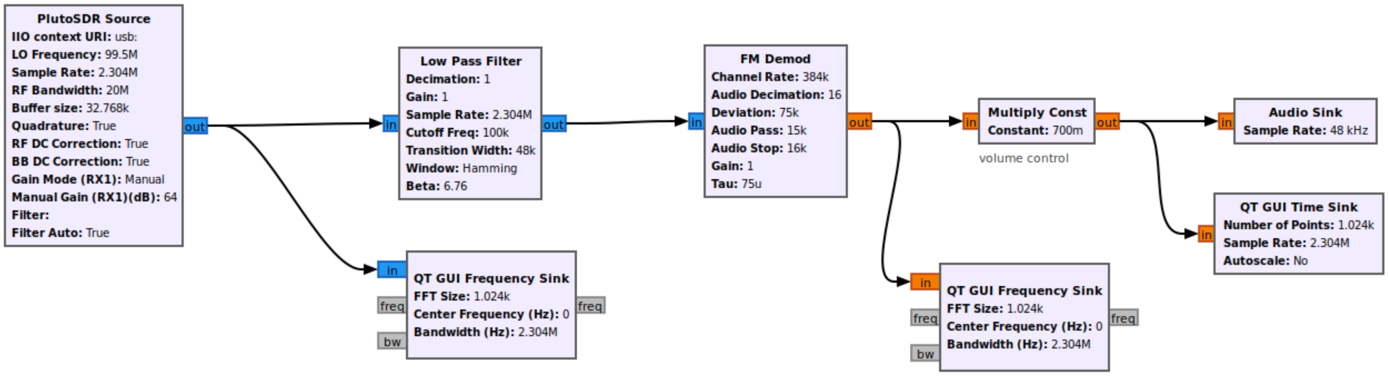
In amplitude modulation, information is encoded through *changes* in amplitude. The change in amplitude is proportional to the message amplitude. An amplitude-modulated signal can be demodulated in one of two ways. The first is called coherent modulation. It works by multiplying the modulated signal with a cosine at exactly the same frequency as the transmitter, then passing the output of that through a low pass filter. This method is extremely expensive and unpopular. The second method is called envelope detection, and works by passing the signal AM modulated signal through a rectifier, then through a low pass filter. In software, the rectifier circuit can be implemented using the "Complex to Mag" block; the low pass filter using a FIR filter block or rolling average.

1.2 Angle Modulation – PM & FM

Another common analog modulation standard is FM. This is a modulation scheme where a carrier wave is modulated with an information signal in a way that causes the frequency to fluctuate as the amplitude of the information signal changes. Unlike AM, the amplitude of the modulated carrier remains constant. The modulation standard is commonly used for commercial radio stations due to its high resilience to additive noise. In this experiment, we picked the station 99.5 FM to listen to.

2 FM Demodulation with GNU Radio

In this section, we listened to 99.5 FM by implementing FM modulation. We first built our graph without the custom buffer. However, the audio kept cutting out and was basically incoherent. Once we added the custom buffer, the audio quality was almost as good as a commercial radio. The reason for this is that samples were being skipped by the audio sink. Once those samples were held in memory, the sink sampled from them whenever it was ready. The resulting sound was smooth and pleasant.



PlutoSDR Source
IIO context URI: usb:
LO Frequency: 99.5M
Sample Rate: 2.304M
RF Bandwidth: 20M
Buffer size: 32.768k
Quadrature: True
RF DC Correction: True
BB DC Correction: True
Gain Mode (RX1): Manual
Manual Gain (RX1)(dB): 64
Filter:
Filter Auto: True

Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 2.304M
Cutoff Freq: 100k
Transition Width: 48k
Window: Hamming
Beta: 6.76

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 2.304M

FM Demod
Channel Rate: 384k
Audio Decimation: 16
Deviation: 75k
Audio Pass: 15k
Audio Stop: 16k
Gain: 1
Tau: 75u

Multiply Const
Constant: 700m

Audio Sink
Sample Rate: 48 kHz

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 2.304M
Autoscale: No

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 2.304M

Figure 1. FM modulation flowgraph

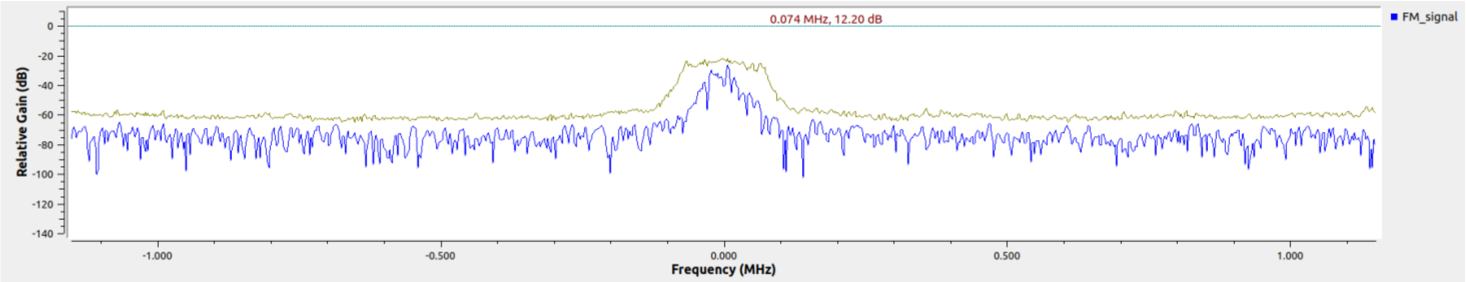


Figure 2. Signal before demodulation

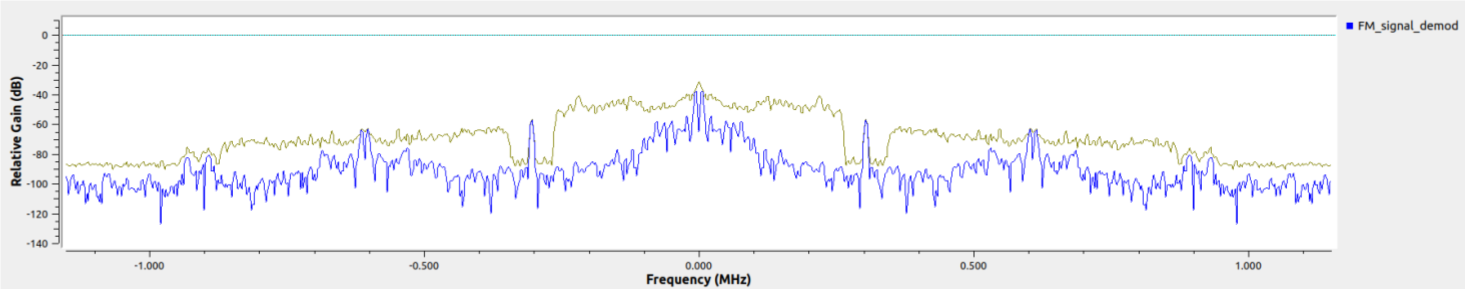
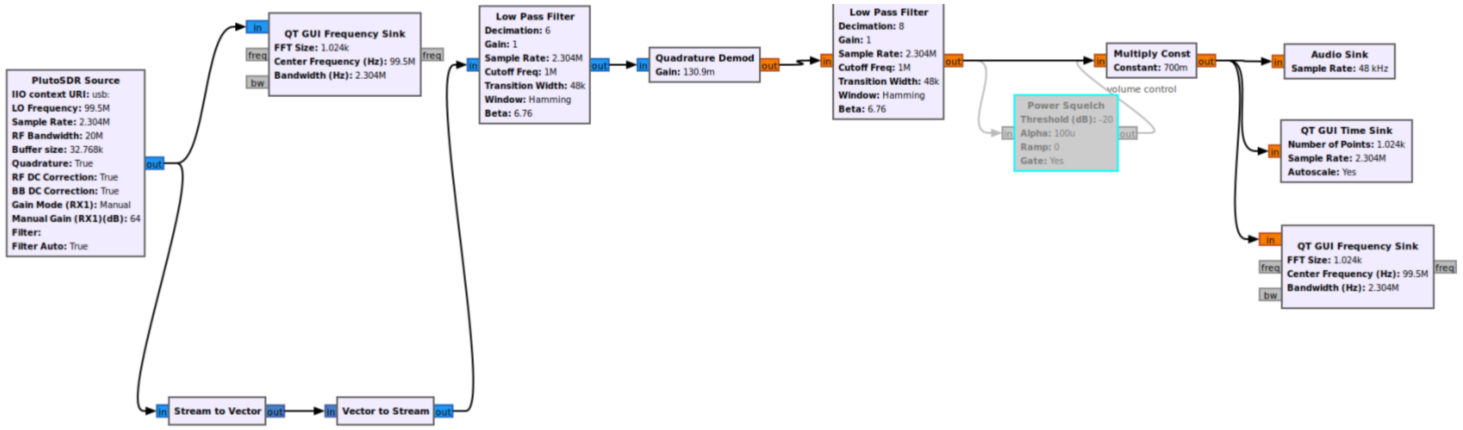


Figure 3. FM signal after demodulation.

Our decimation and interpolation approach was based on the fact that the Pluto SDR samples at a different rate than the audio sink, but they must sample at the same rate. This can be achieved in 2 ways. The first way uses a rational resampler block, which outputs a new sampling frequency
$f_{s_{out}} = \ f_{s_{in}} * interpolation/\ decimation.$
The other way to get the Pluto sampling rate to the audio sink's is by adjusting the decimation in the properties of the FM Demod and Low Pass Filter Blocks so that their product equals 24. This is what we did in this experiment, and our audio sounded very good (once we implemented a custom buffer). The custom buffer was implemented using the stream to vector and vector stream blocks. Together, these blocks essentially held the sampled audio until the audio sink was ready to sample. We chose numitems to be f_s * 8 , where 8 is the number of seconds of sampled audio, and vector length to be 1.
In the provided flowgraph example (not shown), the low pass filter had a cutoff frequency of 100 KHz. With this value, the low pass filter attenuates the signal from an adjacent FM channel from entering demodulator as interference.

2.1 Manual RF Demodulation

The NBFM, WBFM, and FM Demod blocks in GNU Radio perform a multiple signal processing steps for you internally such as calculating frequency deviation from the carrier and filtering. To get a better appreciation for how FM demodulation works, we demodulated the FM signal using quadrature modulation. In quadrature demodulation, we get information of the message signal by taking the imaginary and real parts of 2 consecutive received samples and calculating the phase difference between them. The phase is calculated using the equation $\phi = re\{x\}/im\{x\}$, where x is a complex sample. This gives us information about the message signal because FM modulation encodes information through changes in frequency.
In this experiment, we repeated the flowgraph from the previous section up until and following the FM demodulation block. We replaced the FM demodulator block with the quadrature demodulator block and low-pass filter, as shown below.


PlutoSDR Source
IIO context URI: usb:
LO Frequency: 99.5M
Sample Rate: 2.304M
RF Bandwidth: 20M
Buffer size: 32.768k
Quadrature: True
RF DC Correction: True
BB DC Correction: True
Gain Mode (RX1): Manual
Manual Gain (RX1)(dB): 64
Filter:
Filter Auto: True

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 99.5M
Bandwidth (Hz): 2.304M

Low Pass Filter
Decimation: 6
Gain: 1
Sample Rate: 2.304M
Cutoff Freq: 1M
Transition Width: 48k
Window: Hamming
Beta: 6.76

Quadrature Demod
Gain: 130.9m

Low Pass Filter
Decimation: 8
Gain: 1
Sample Rate: 2.304M
Cutoff Freq: 1M
Transition Width: 48k
Window: Hamming
Beta: 6.76

Power Squelch
Threshold (dB): -20
Alpha: 100u
Ramp: 0
Gate: Yes

Multiply Const
Constant: 700m

Audio Sink
Sample Rate: 48 kHz

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 2.304M
Autoscale: Yes

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 99.5M
Bandwidth (Hz): 2.304M

Stream to Vector

Vector to Stream

Figure 4. Quadrature demodulation

Repeat the Narrow Band FM reception example from the section above, but perform the individual demodulation steps. This is accomplished by replacing the FM demodulator block.
with the quadrature demodulator and audio low-pass filter. The signal before and after demodulation is shown below:

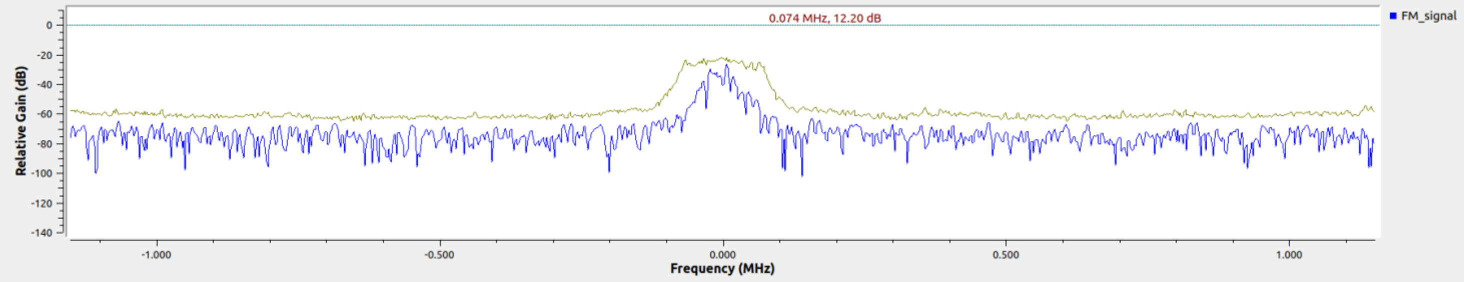


Figure 5. Signal before demodulation

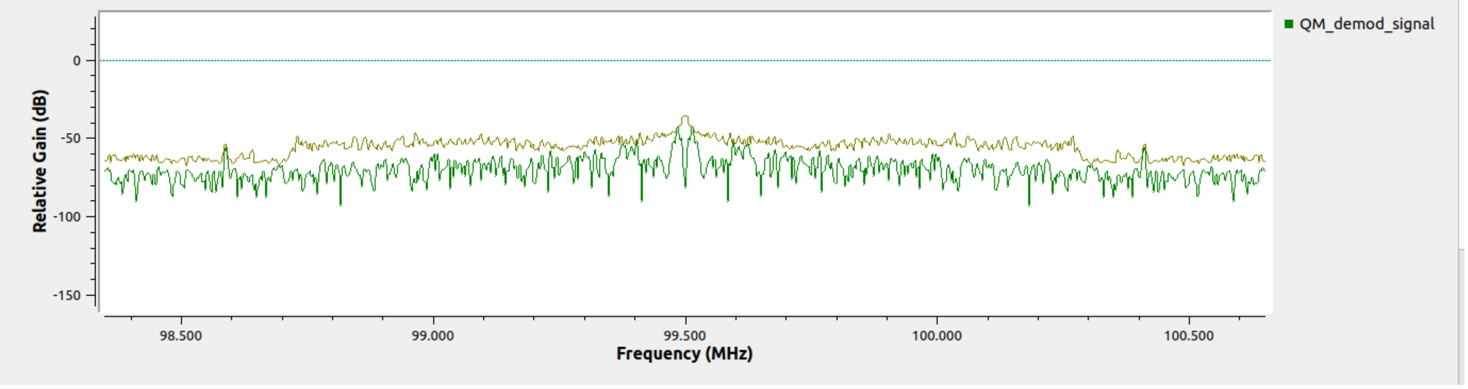


Figure 6. QM demodulated signal

\To ensure consistent sample rates throughout receive chain, we decimated by a factor 24. This time the decimation was part of the second low pass filter.
How does your audio sound compared to the previous FM flowgraph? The quality of the audio sounded the same as in the previous section. However, the audio would stop for briefly and then continue.

For the “Gain” parameter in the quadrature demodulator, we created a range slider and dynamically adjusted it until audio quality reached a peak. We couldn’t set the value statically or determine it analytically because frequency deviation is transmitter dependent. Thus, we used the following formula to set the frequency deviation:
2*math.pi*deviation)/audio_rate, where audio_rate was 48KHz.

3 Questions

1. Describe how to demodulate an amplitude modulated signal without simply using the “AM Demod” block in GNU Radio? There are two main ways to demodulate an AM modulated signal. The first is called coherent or synchronous detection. In this method, the AM modulated signal is multiplied by a cosine with the same carrier frequency as the modulating carrier. This signal is then passed through a LPF, which yields the original message signal at half the amplitude. The main con of coherent demodulation is that the receiver must be able to generate a local carrier at the exact frequency and phase as transmitter’s modulating carrier. The second method to demodulate an AM modulated signal is through envelope detection, also called non-coherent demodulation. This method is far cheaper (both financially and computationally) than coherent demodulation. In this method, the AM modulated signal is passed through a rectifier and then through a low pass filter. The rectifier circuit outputs only the positive cycles, which gives us the original signal. The rectifier is implemented by with by a simple diode circuit, as a diodes will turn on only during the cycles where the voltage is positive. We need a low pass filter to filter out any undesired frequencies. The envelope detection circuit is shown below

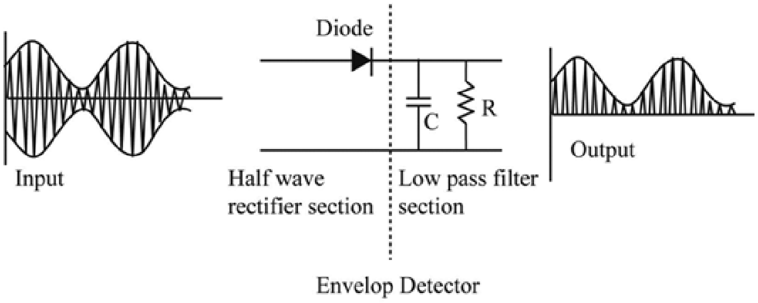


Figure 1

2. Our car stereos are able to receive broadcast AM radio signals. Why are we unable to demodulate these signals on the PlutoSDR without upconversion? The frequency band of AM radio is 535 KHz to 1605 KHz, but our Pluto cannot detect frequencies below 70 MHz.

Conclusions

This lab explored how analog demodulation is implemented in software defined radio. All the circuitry that is implemented in hardware has a software equivalent, which is why were able to listen FM radio stations. Important considerations to keep in mind throughout the receive chain are sample rates and filter widths. Sample rates must be consistent between source and sink blocks in order for the signal to be accurately received. Filter widths can be determined using Carson’s bandwidth rule.