EE 504 Software Defined Radio

CAL POLY

Instructor: Professor Dunton

Jacquline Radding and Ben Duval

Spring 2024

ELECTRICAL ENGINEERING DEPARTMENT
California Polytechnic State University
San Luis Obispo

EE 504 Experiment #1

Transmit and Receive

Objective Learn how to operate and understand the Pluto radio.

Equipment: MATLAB, Pluto Radio, USB cable, antennas, loopback cable

Background:

RF Transceiver	LO Tuning Range	Bandwidth
AD9363 (factory default)	325 – 3800 MHz	20 MHz
AD9364	70 – 6000 MHz	56 MHz

Figure 1: Choose the 20MHz RF transceiver chip (AD9363) using the configurePlutoRadio

Procedure:

1. Connect the loopback SMA cable to the tx port and the rx port of your pluto.
2. Take a picture of your physical setup.
3. Run the example script and save the final state of the spectrum analyzer plot as a figure.
4. Modify the example script to transmit your chirp signal. Run the procedure again.
5. Replace the loopback cable with the antennas and run the procedure again.
6. Remove the antennas and run the procedure again.

Results:

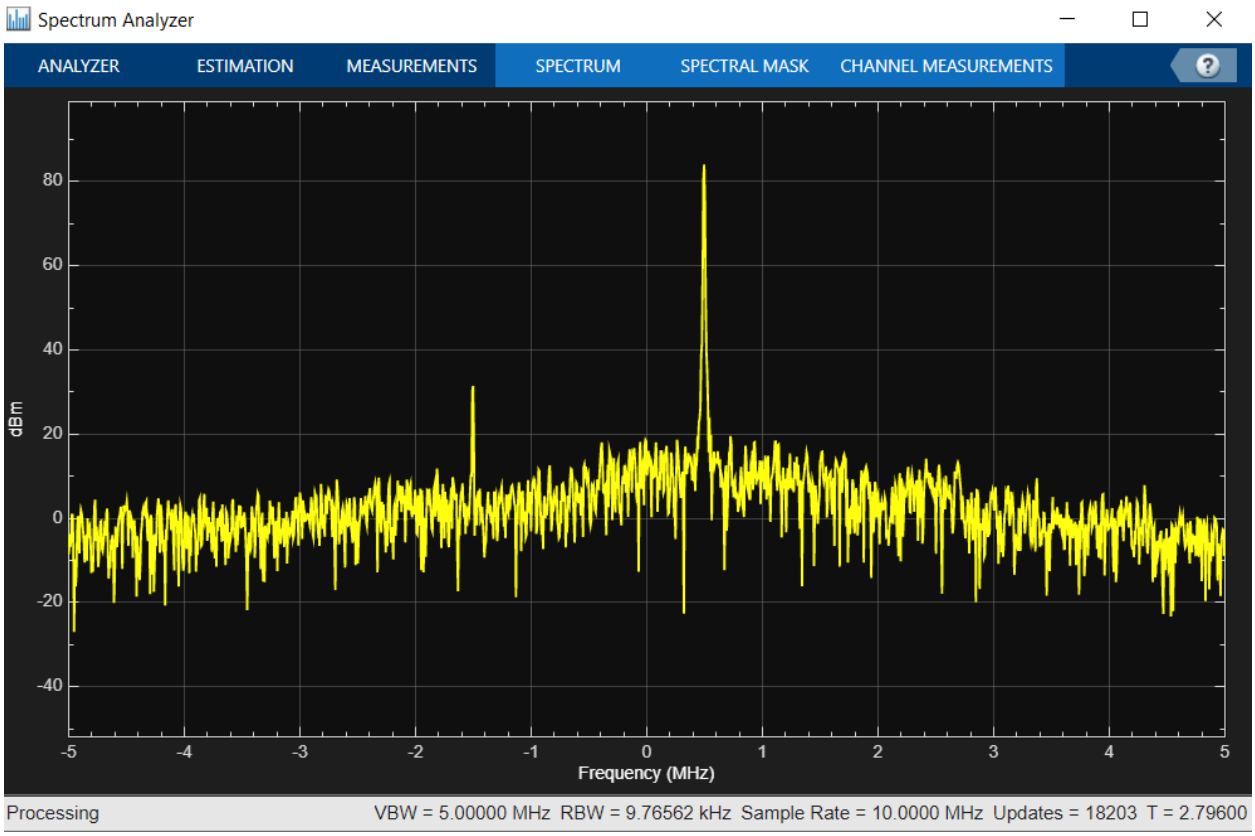


Figure 2: Bunny ears set up sine wave with fs/2

The antennas may have picked up noise from other FM signals.

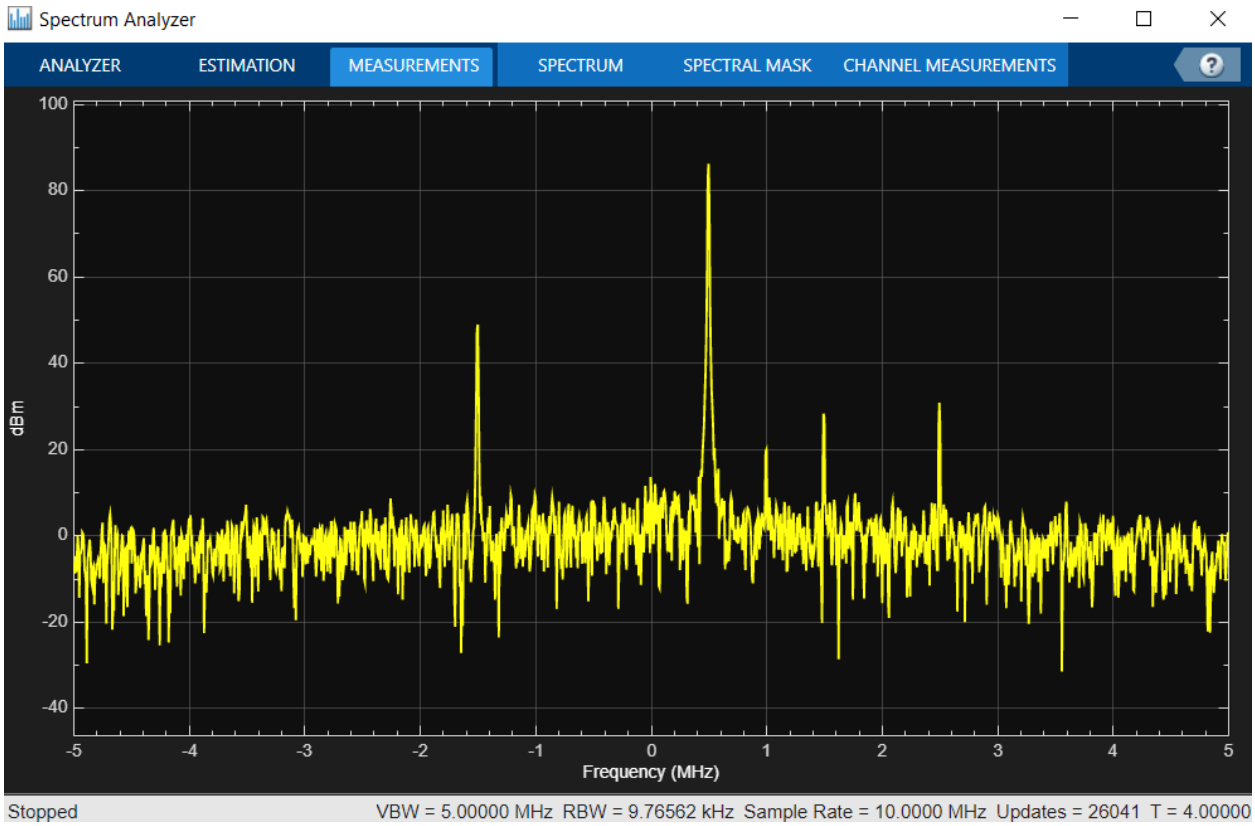


Figure 3: SMP loopback set up using sine wave. fs/2

Harmonics in figure 3 may be seen in the frequency space. The third harmonic is usually the most impactful on the signal. Harmonics may be seen in the figure above. The clock is 4MHz so the interference is most likely not the clock.

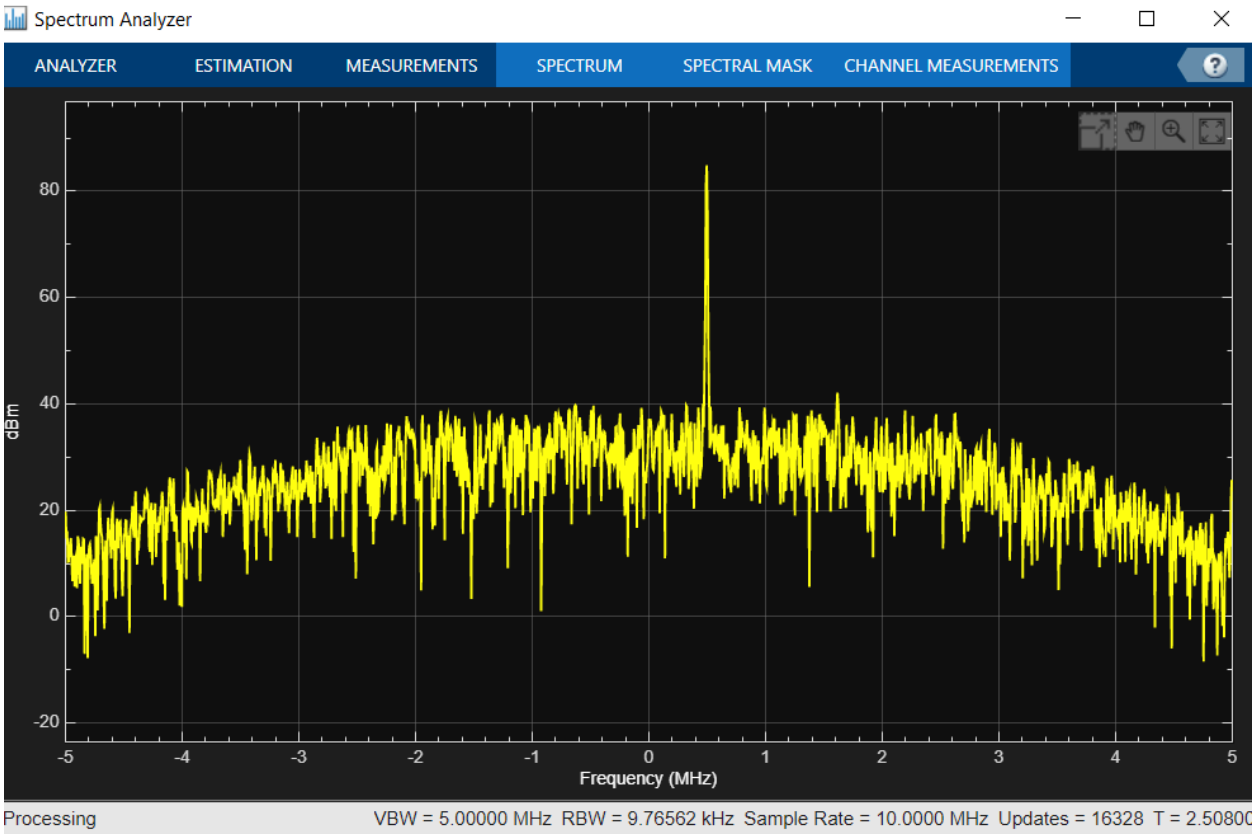


Figure 4: No antennas or SMP loopback with fs/2

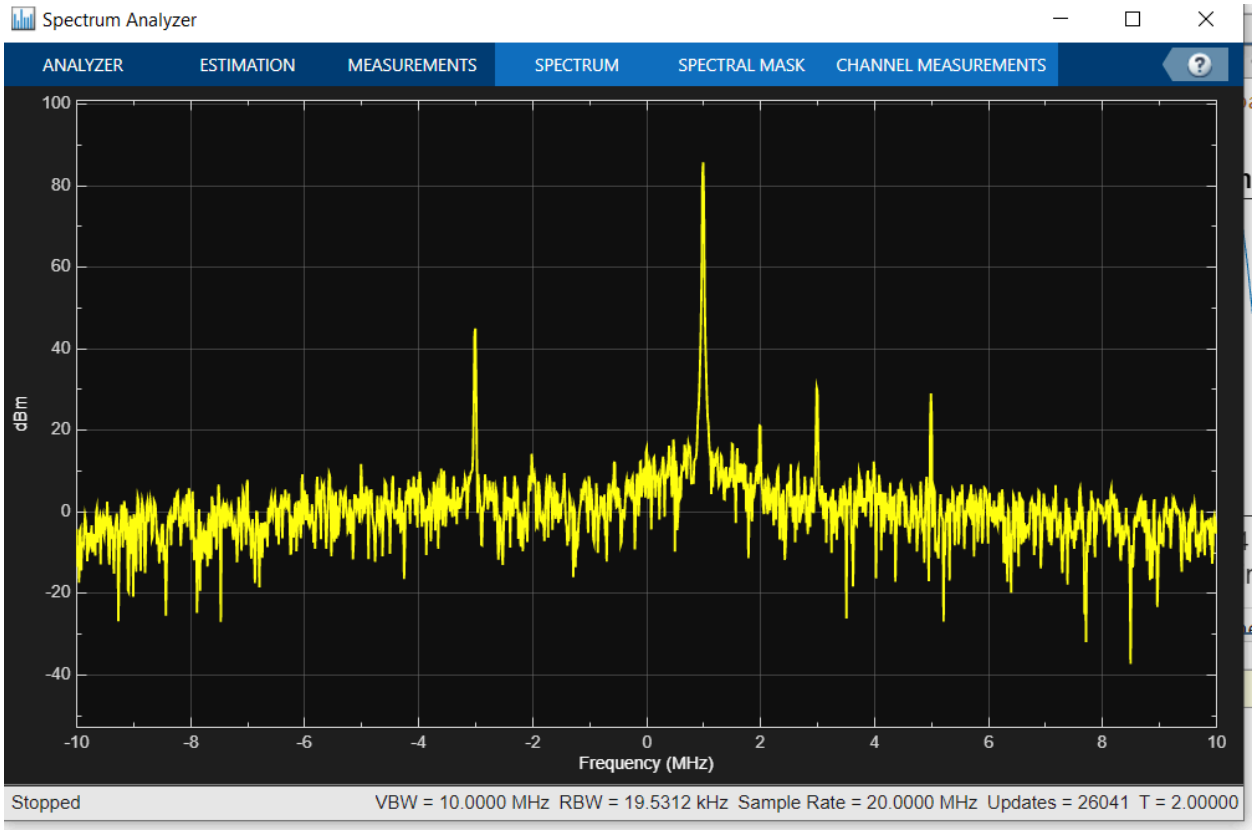


Figure 5: SMP loopback with fs

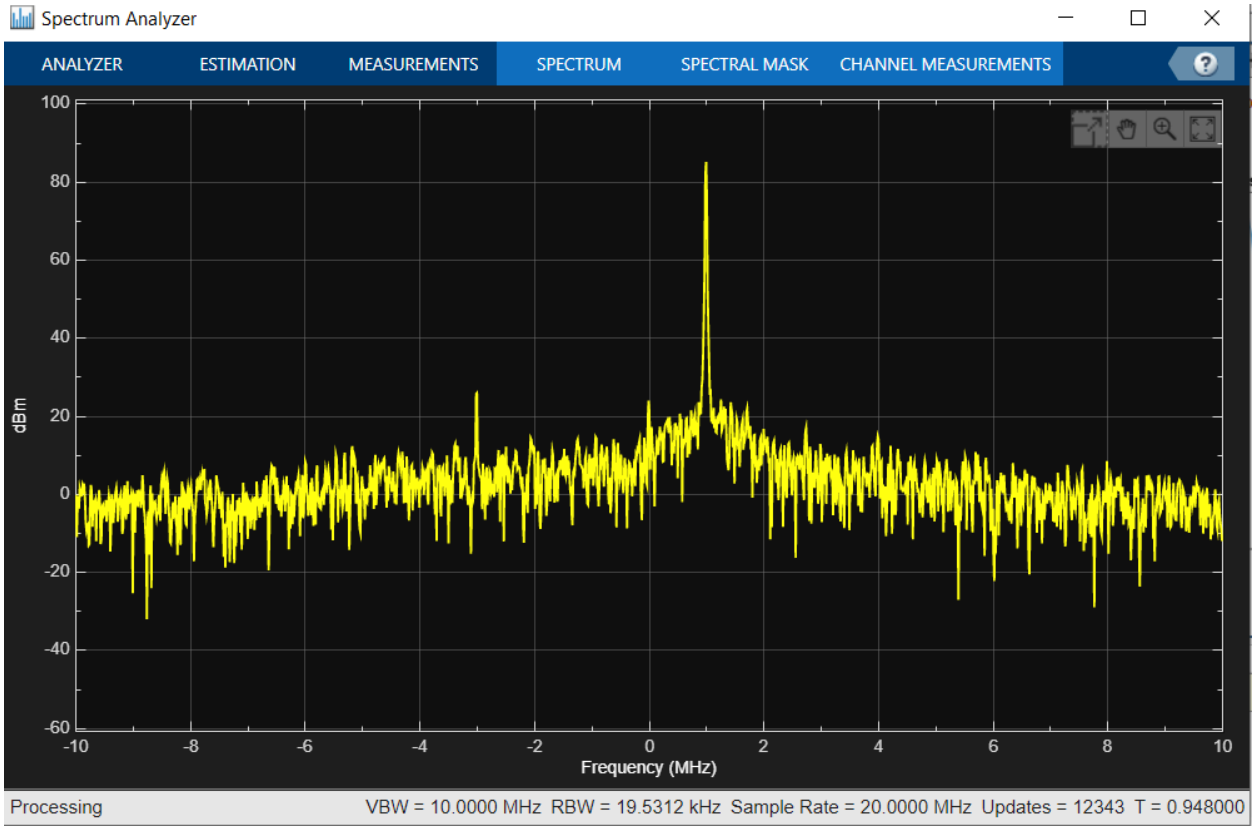


Figure 6: Bunny ears with fs

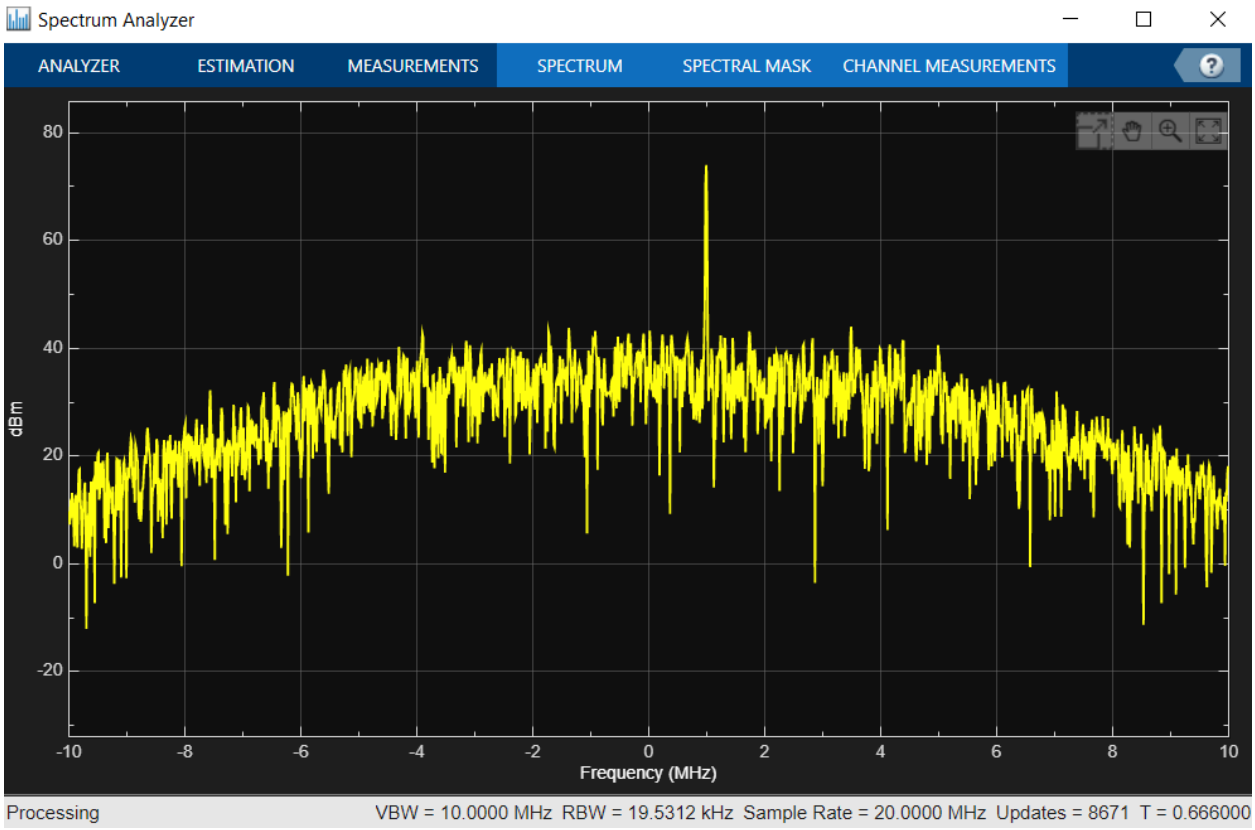


Figure 7: Nothing with fs

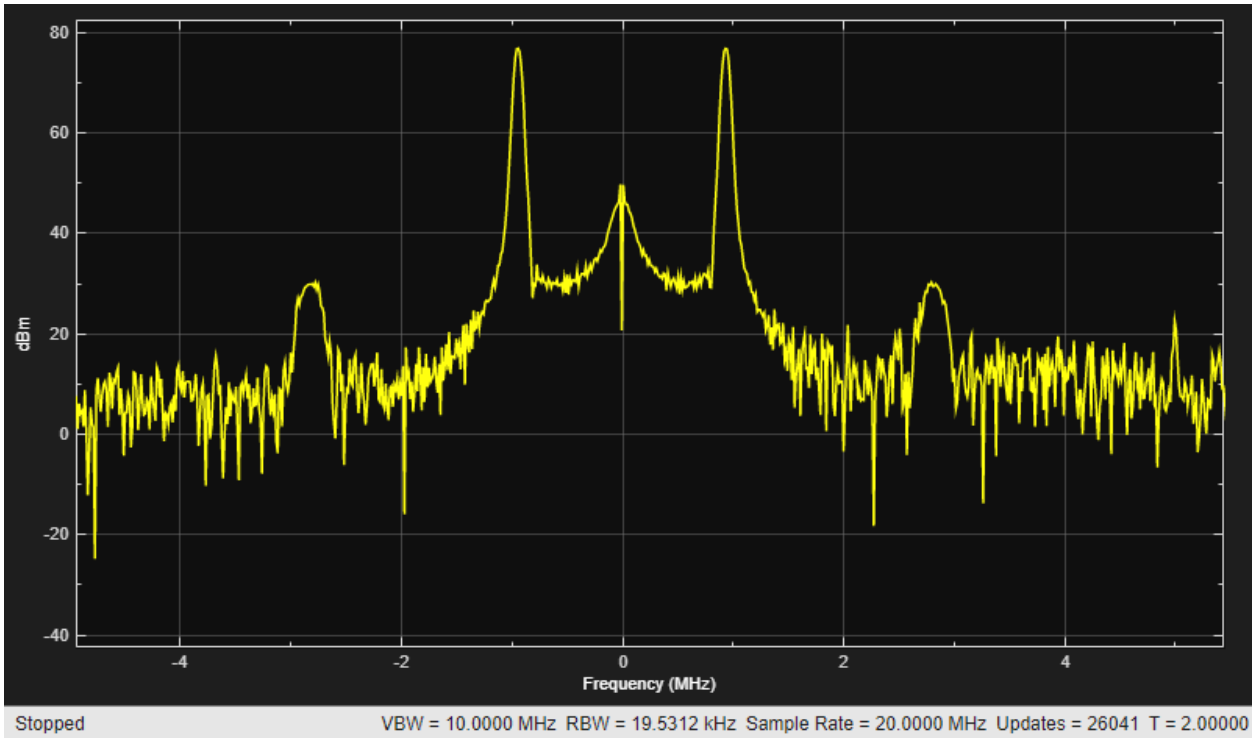


Figure 8: Bunny ears with chirp signal

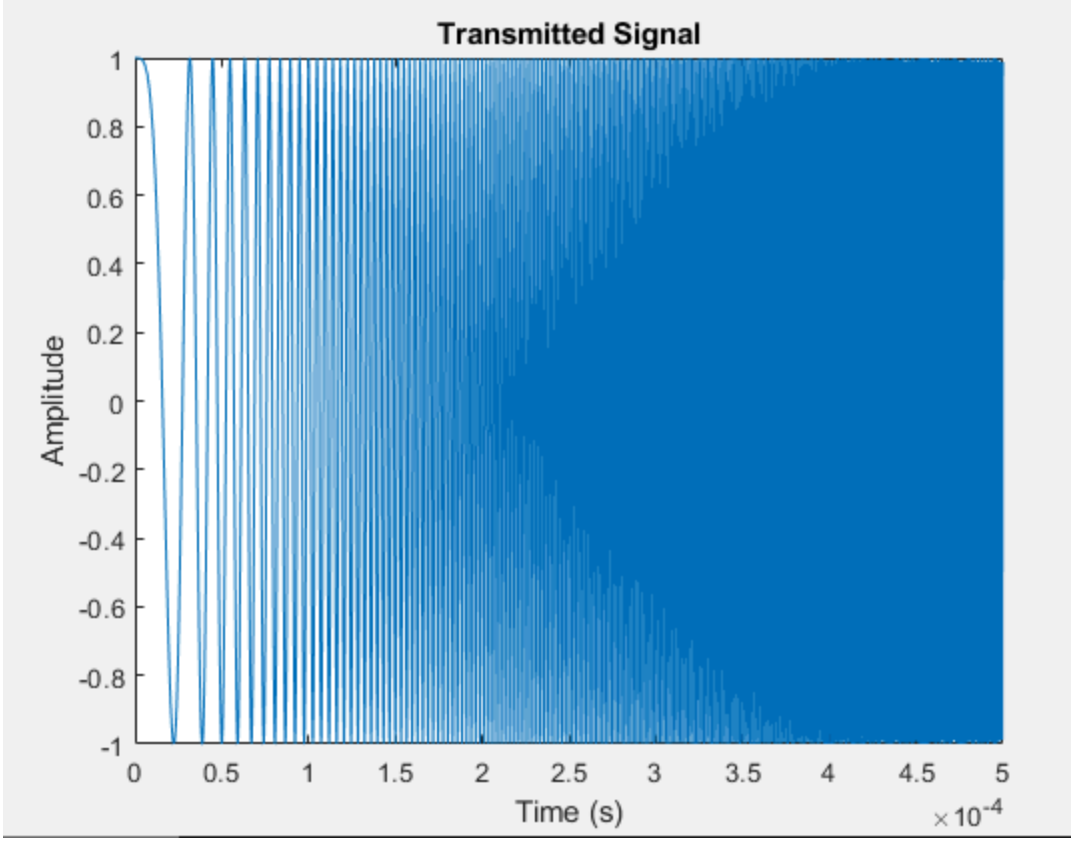


Figure 9: Chirp signal Transmission

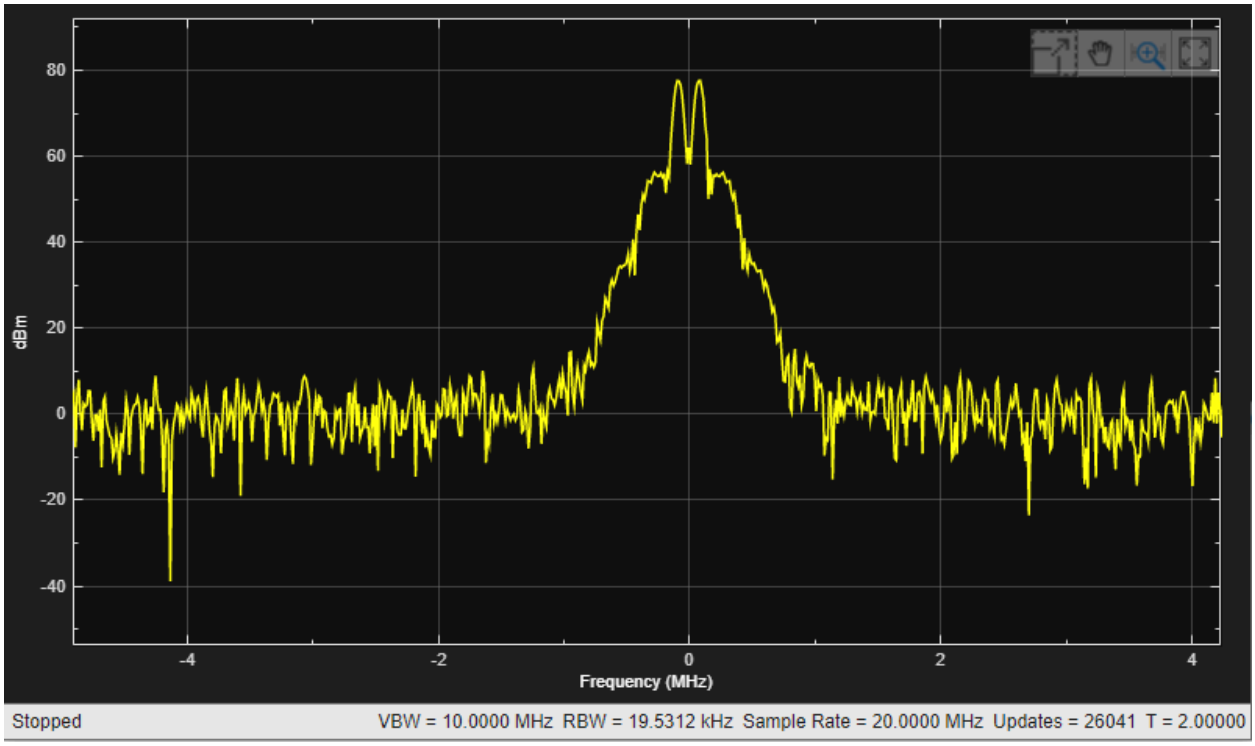


Figure 10: SMP loopback Chirp signal

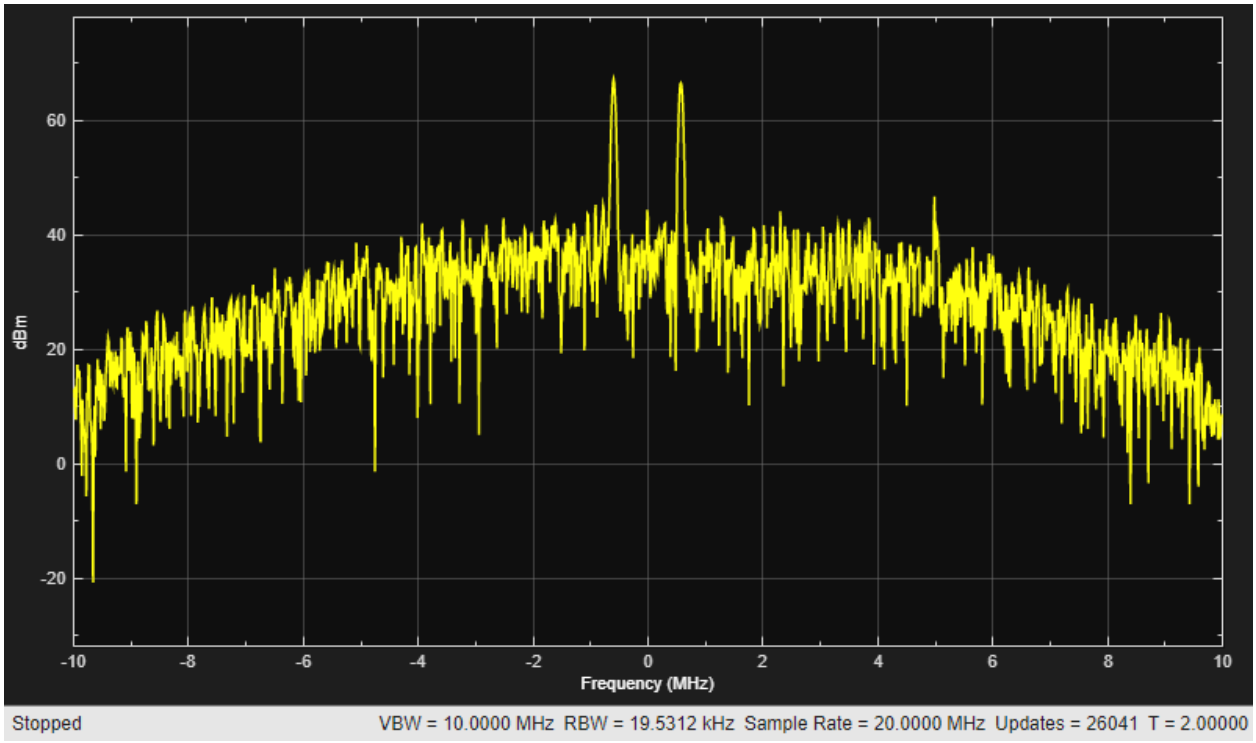


Figure 11: Nothing with chirp signal

Questions:

1. The noise floor is defined as the sum of measured noise power. What happens to the noise floor when switching from the loopback cable to the open ports.

- Looking at Figure 5, the noise floor seems relatively stable when using the SMA loopback cable. However, upon changing to leaving the ports open, in Figure 7, we can actually see the noise floor increase across all frequencies. This is indicative of noise being picked up from the surroundings.

2. A spur is an undesirable signal that may be generated at the receiver or from some interfering source. Do you notice any spurs in the loopback spectrum? Which frequencies if so?

- There are a few spurs in the loopback spectrum, in Figure 3, at 3 and 5 MHz. These may be harmonics, interference, or reflections.

3. The signal-to-noise ratio (SNR) is defined as the ratio of signal power to the noise power. Estimate the peak signal power and average noise power from each spectrum. Which system seems to have the best SNR?

- SNR = peak amplitude / rms noise, SNR = S - N for decibel
- Nothing from fig 11: SNR = 60dBm - (-8 dBm) = 68 dBm
- Loopback from fig 10 SNR =75- (-20) = 95 dBm
- Antenna from fig 8 75-(-10) = 85 dBm
- The loopback has the best dBm. This makes sense due to less susceptibility to noise interference due to a direct closed loop.

4. The bandwidth of a signal describes the range of frequencies present. What is the bandwidth of your chirp signal?

- The bandwidth is 0.4 MHz according to graphs 10 and 11.

5. How would the received signal spectrum change if we transmitted a single tone signal instead of a chirp signal?

- A signal tone would be an impulse-like function at a certain frequency in the frequency space.

6. Is the radio, as used in the experiment, operating in simplex, half duplex, or full duplex?

- The radio is full duplex since it sends and receives simultaneously.

Conclusion:

It is important to understand what loopback, disconnected, and antenna signals look like when troubleshooting. Some interference could be seen in the antenna signal due to other transmissions such as radio. The loopback may have reflections or harmonics as interference. Overall, the introduction to the Pluto radio improved communication understanding.

ELECTRICAL ENGINEERING DEPARTMENT

California Polytechnic State University
San Luis Obispo

EE 504 Experiment #2

FM Radio

Objective Learn how to operate and understand the FM radio.

Equipment: MATLAB, Pluto Radio, USB cable, SMA Antenna

Results:

Measuring out the quarter wavelength of 91.3 MHz, ~82 cm would serve as a functional antenna. $\lambda/4 = c / 4 \times f$. A copper wire was cut and soldered onto an SMA connection that went to the PLUTO radio. A ground wire was not used.

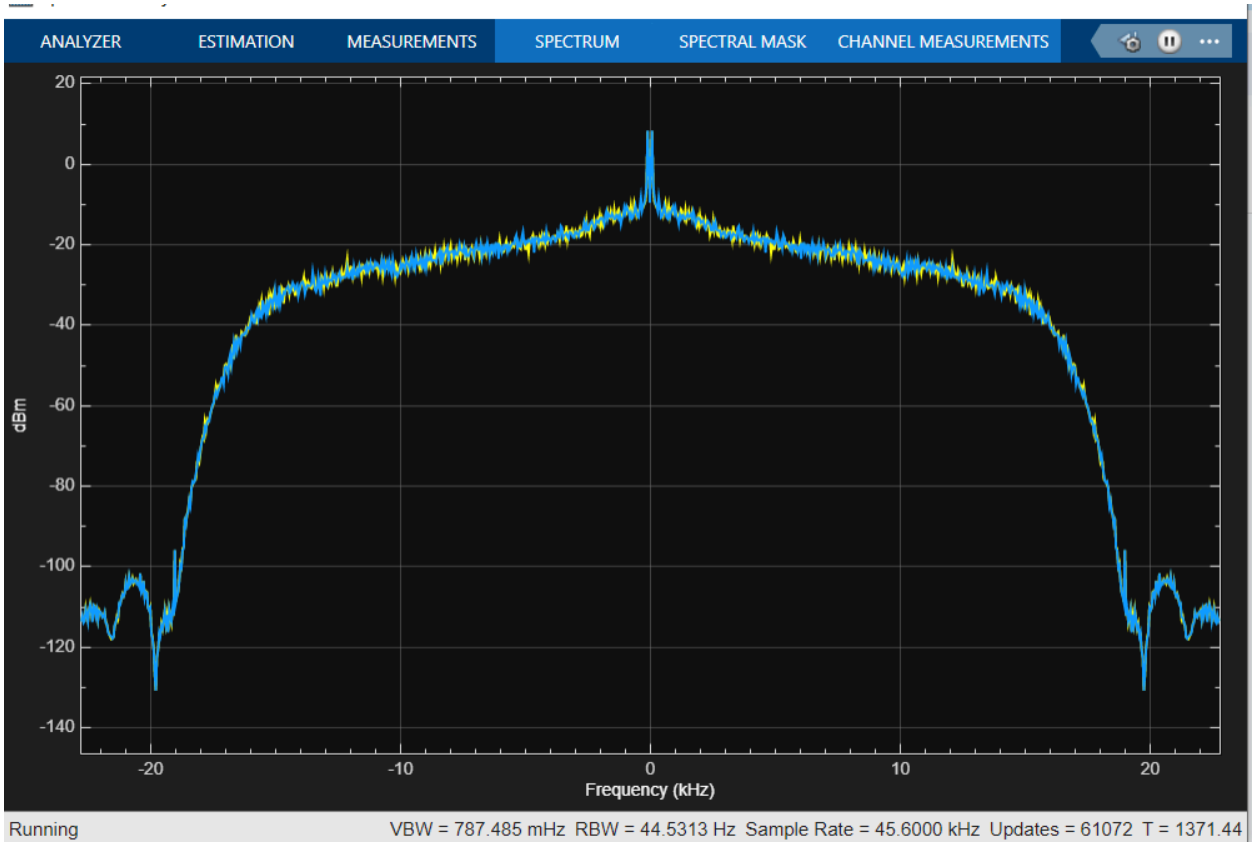


Figure 2-1: Using the oscilloscope, the FM signal after the ZOH

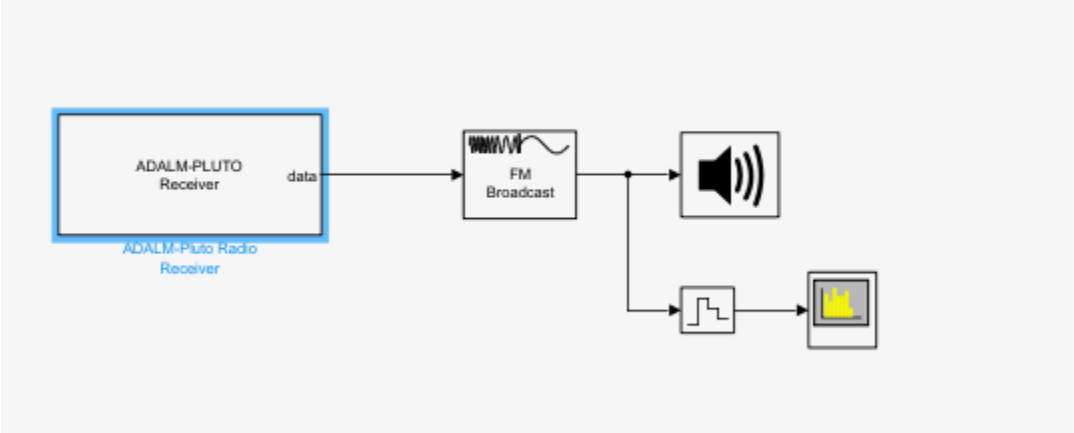


Figure 2-2: The Simulink FM broadcast setup.

Questions:

1. How did the audio fidelity compare between the kit antenna and the λ/4 monopole?

The kit antenna had poor audio quality and some distortion. The monopole was clear. This was due to the fact that the kit was the wavelength divided by a large number, it was not at the resonant frequency.

2. Why do we need a ZOH block between the demodulated signal and the spectrum analyzer?

We need a ZOH block so the signal is consistent going to the oscilloscope. The ZOH is also used to reconstruct the signal so it can be read.

3. What did you notice about the two baseband spectra?

The two baseband spectra have a lot of noise. The frequency measured is in kHz

4. Recall that the procedure has defined baseband sample rate, samples per frame, frequency deviation, de-emphasis filter, and audio sample rate. These were extracted from an example provided by MathWorks. Explain why each of these settings ought to work.

These settings must work because the sample rate is above the Nyquist. All the values line up and are consistent from block to block. The frequency sampled has a rate that can obtain voice. The audio sample rate is the same as how CDs and recording devices are sampled, at 44.1KHz.

Conclusion:

It is important to understand how antennas are important to software that is receiving radio. The proper hardware is necessary to interpret signals. In a software focused world, troubleshooting software and hardware together is needed to get to the root of most issues.

ELECTRICAL ENGINEERING DEPARTMENT

California Polytechnic State University
San Luis Obispo

EE 504 Experiment #3

Digital Modulation and Noise

Objective Learn how to interact with modulation and counteract noise

Equipment: MATLAB, Pluto Radio, USB cable, SMA Antenna

Results:

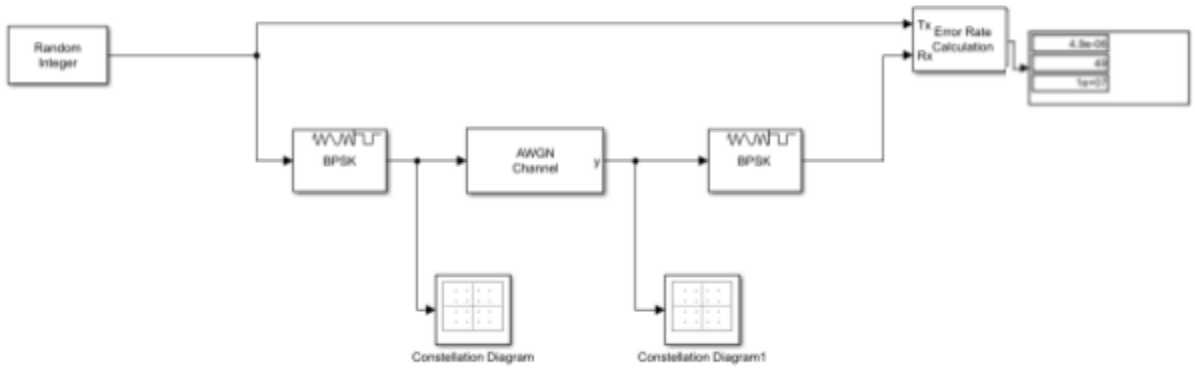


Figure 3-1: Simulink QAM schematic

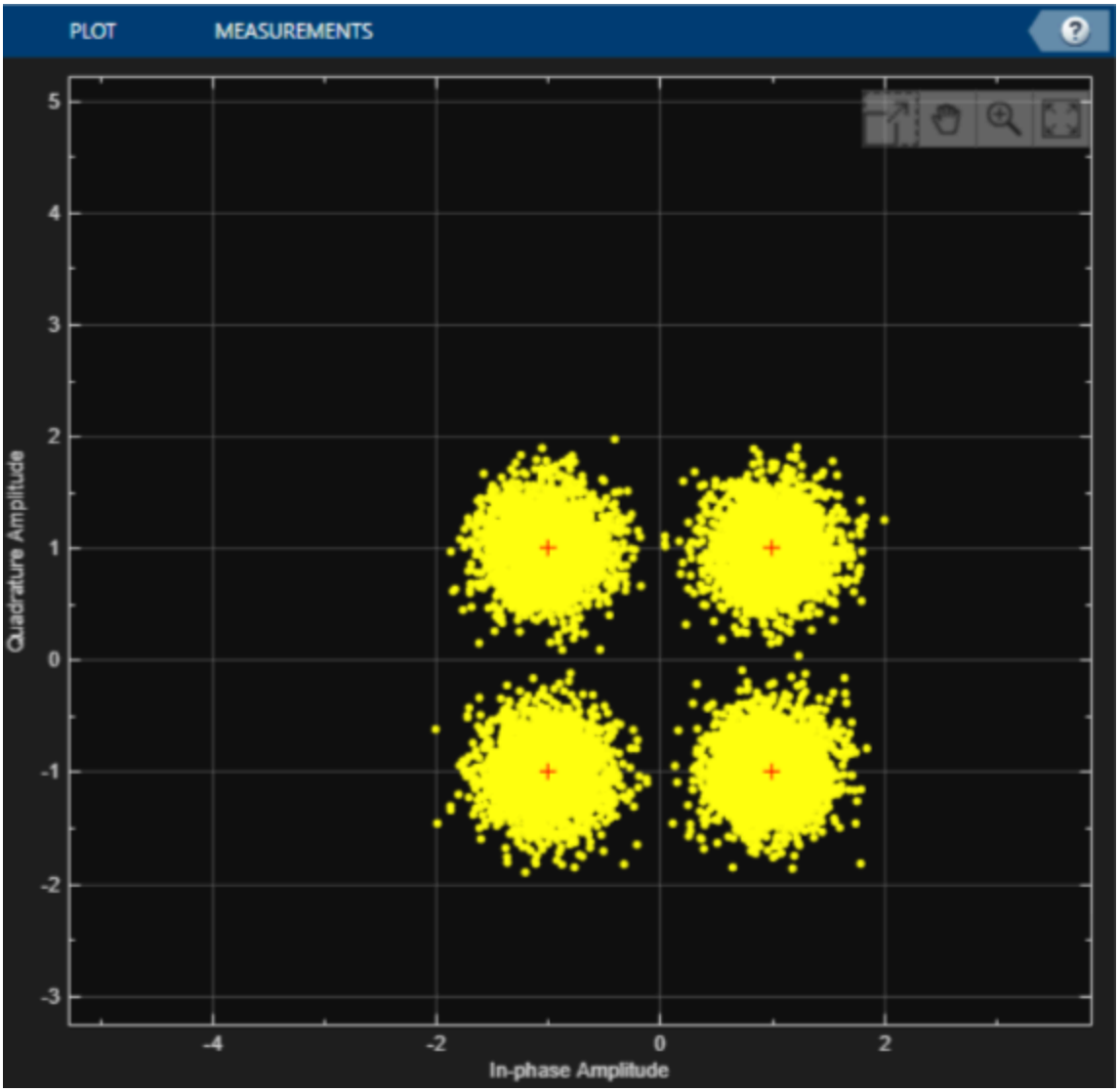


Figure 3-2: Constellation Diagram of QAM modulation

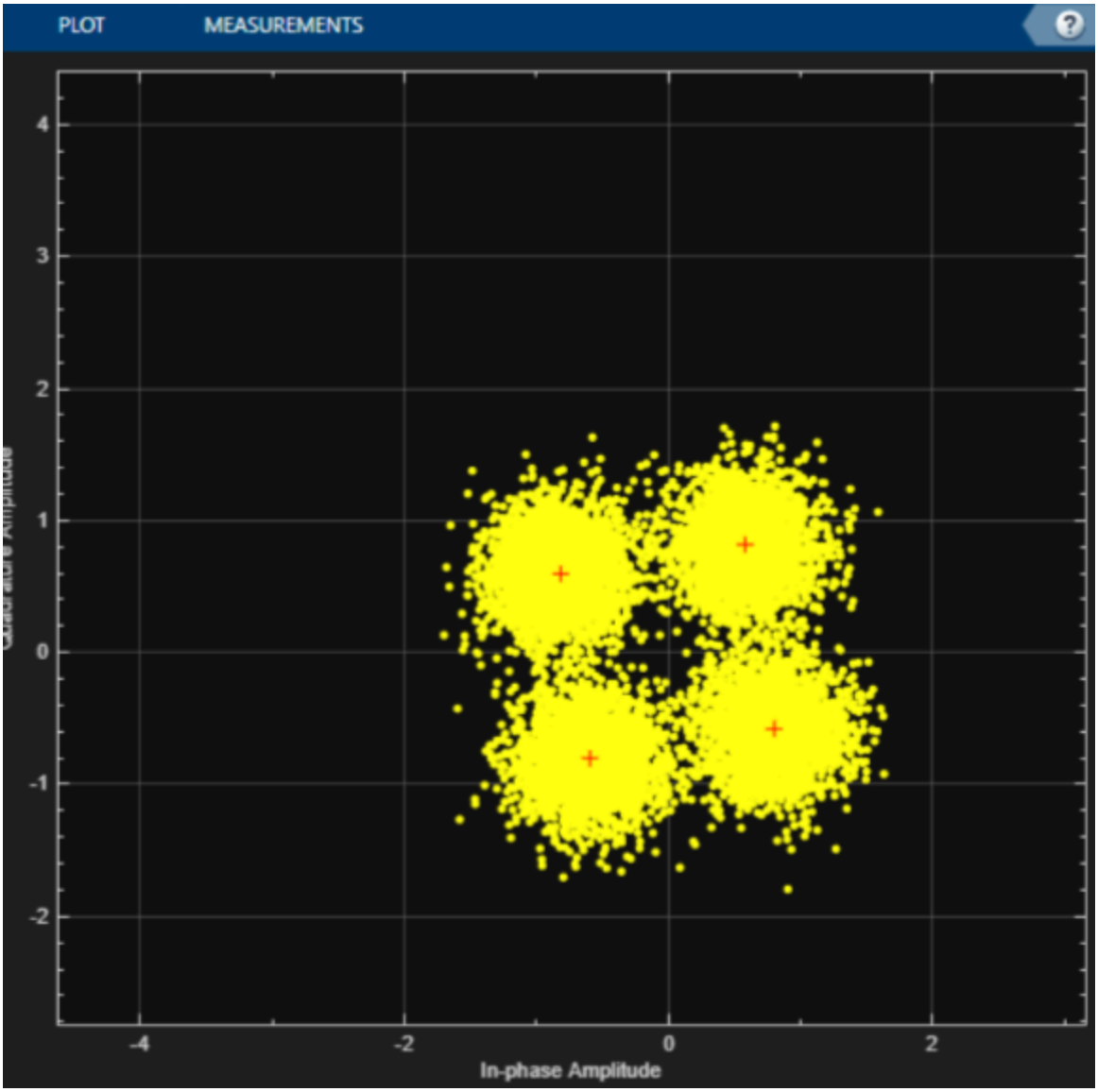


Figure 3-3: Constellation diagram of QPSK modulation

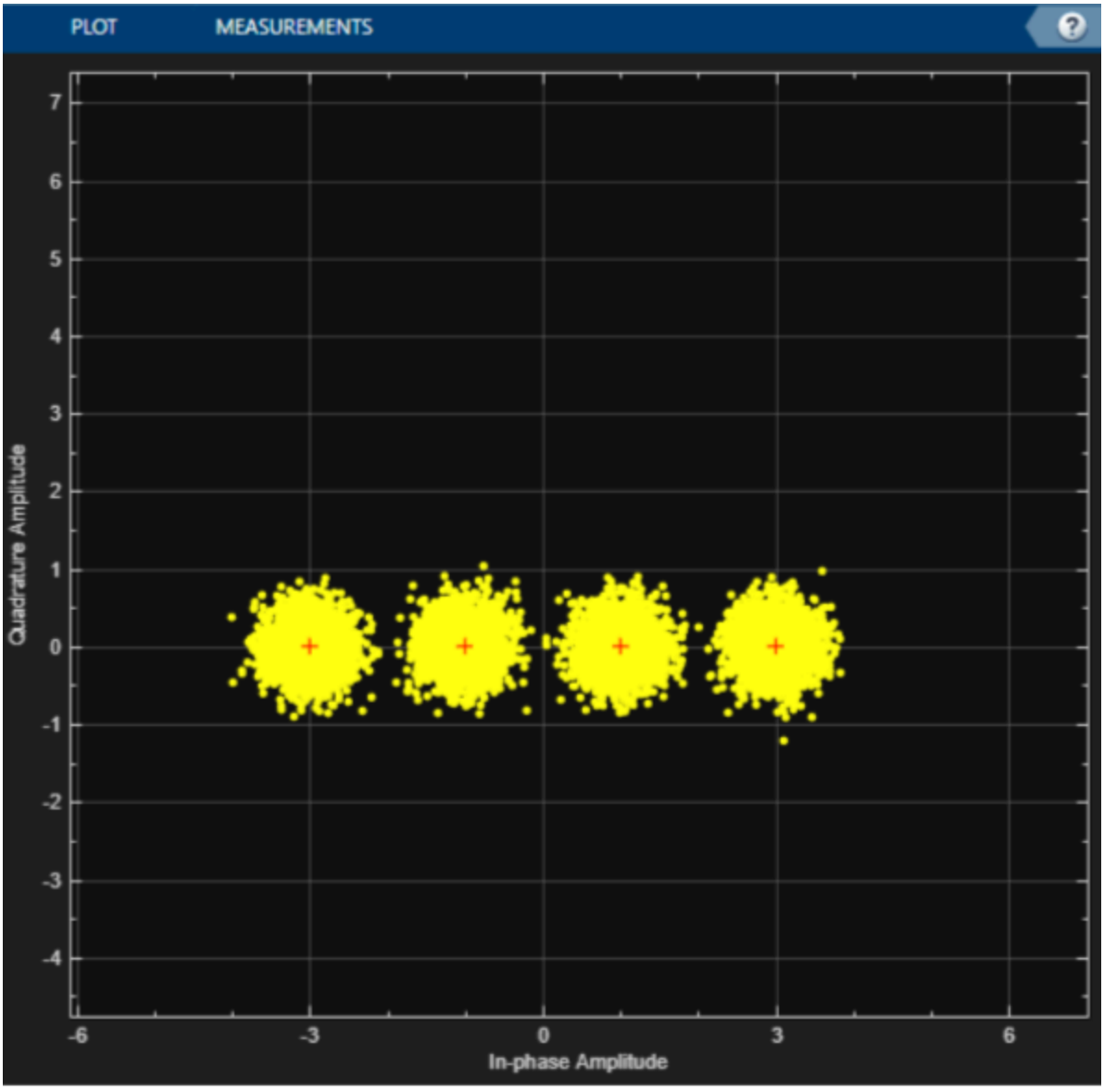


Figure 3-4: Constellation diagram of PAM modulation

The cluster spread represents the noise in the measurement. The QPSK modulation has more noise in the system as the spread is hard to read. From figure 3-5, QPSK has the highest error rate which puts the consistency of the signal into question. The PAM modulation has linear clusters.

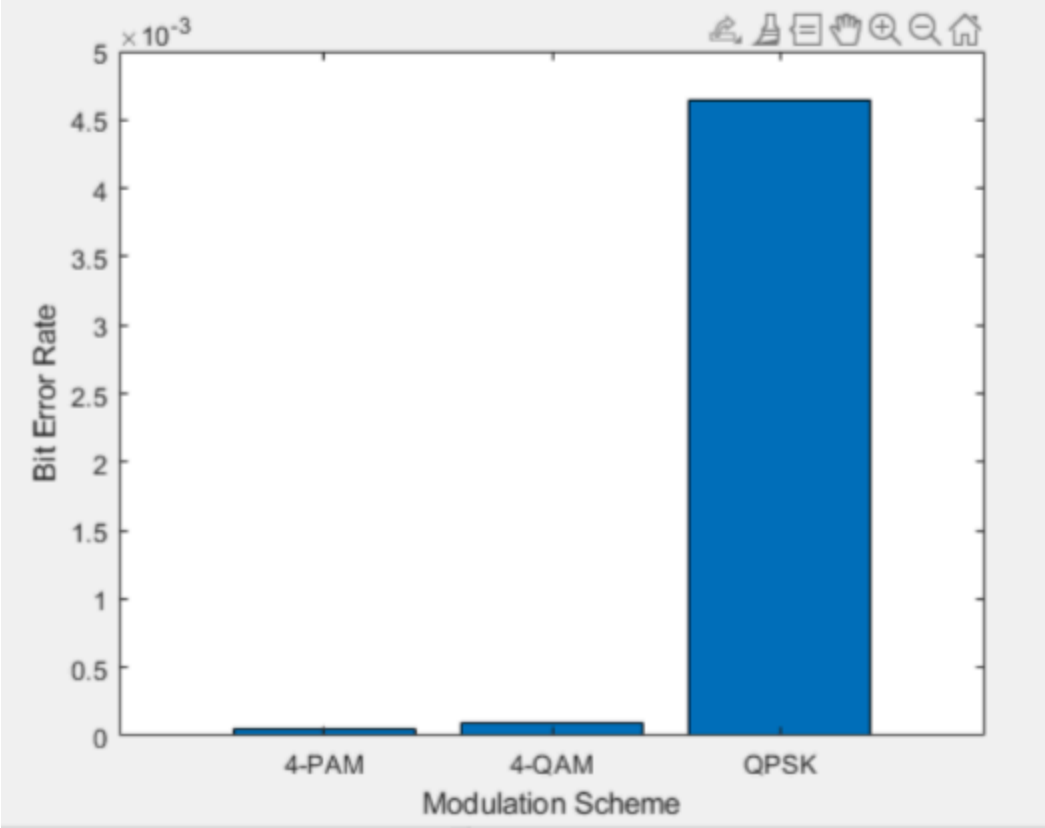


Figure 3-5 Plot histogram of bit error rate

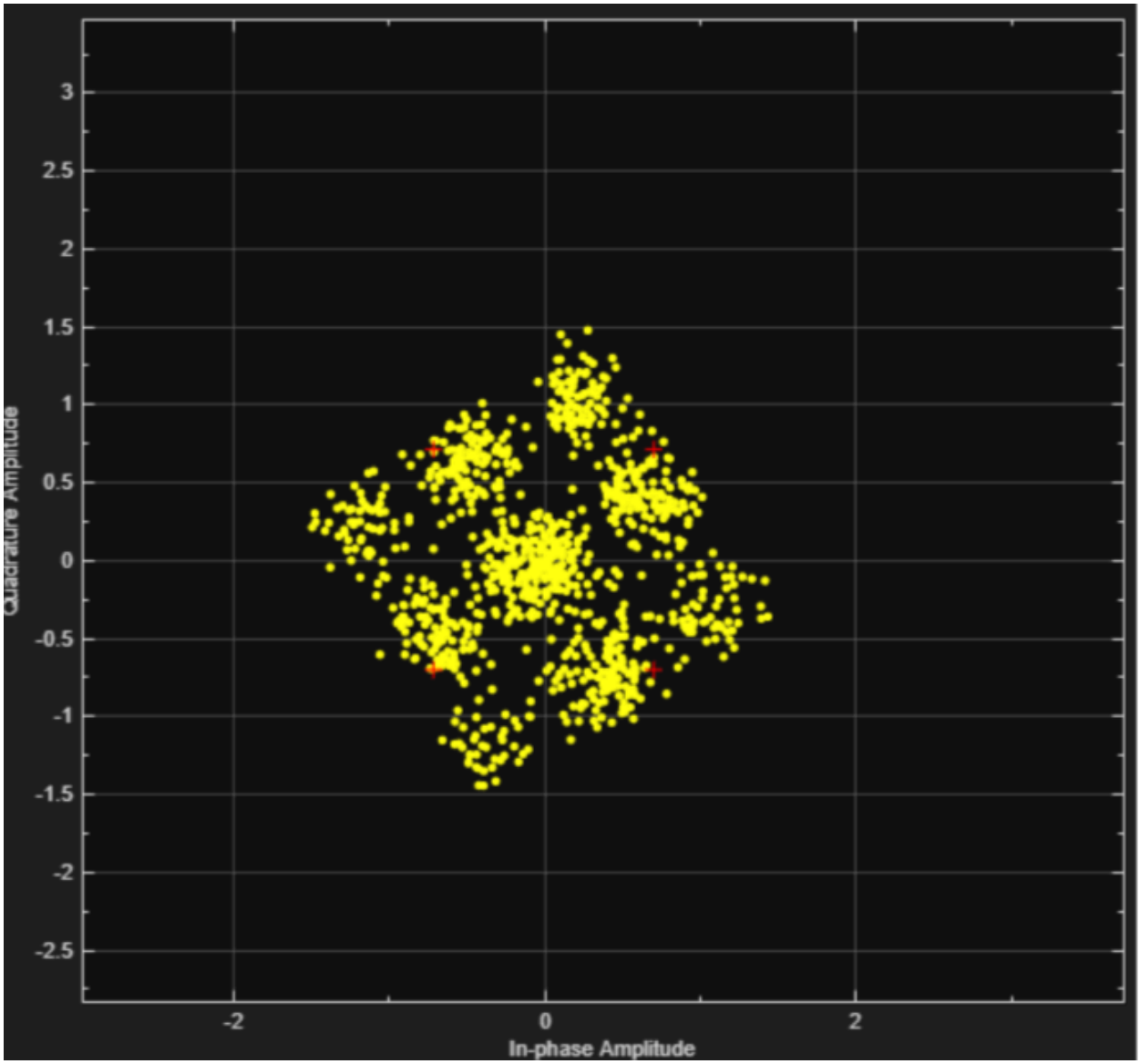


Figure 3-6 QPSK put through SRRC.. the clusters do not interfere with each other from the previous QPSK figure.

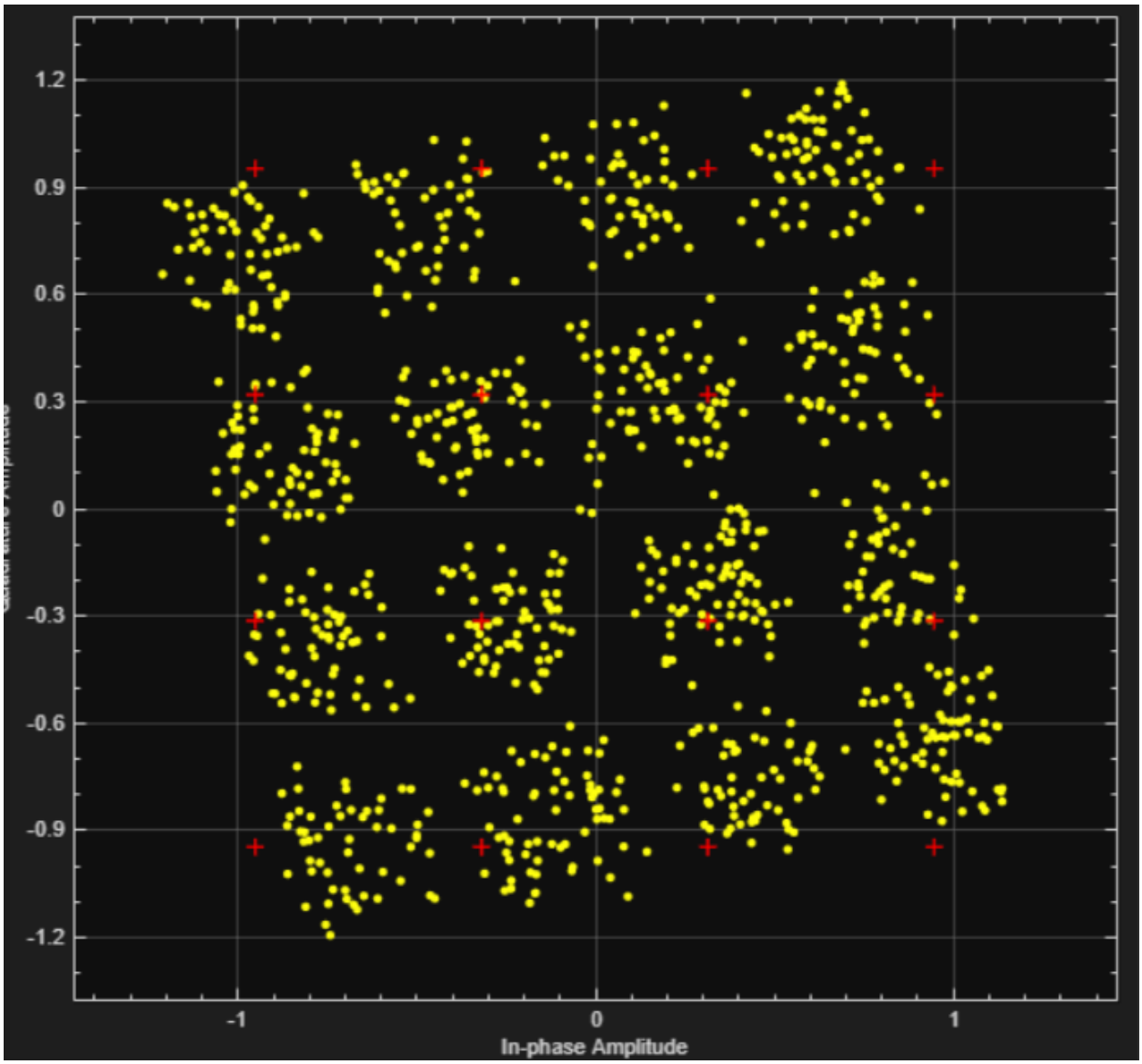


Figure 3-7 QAM timing using plutoLoopback16QAM.m

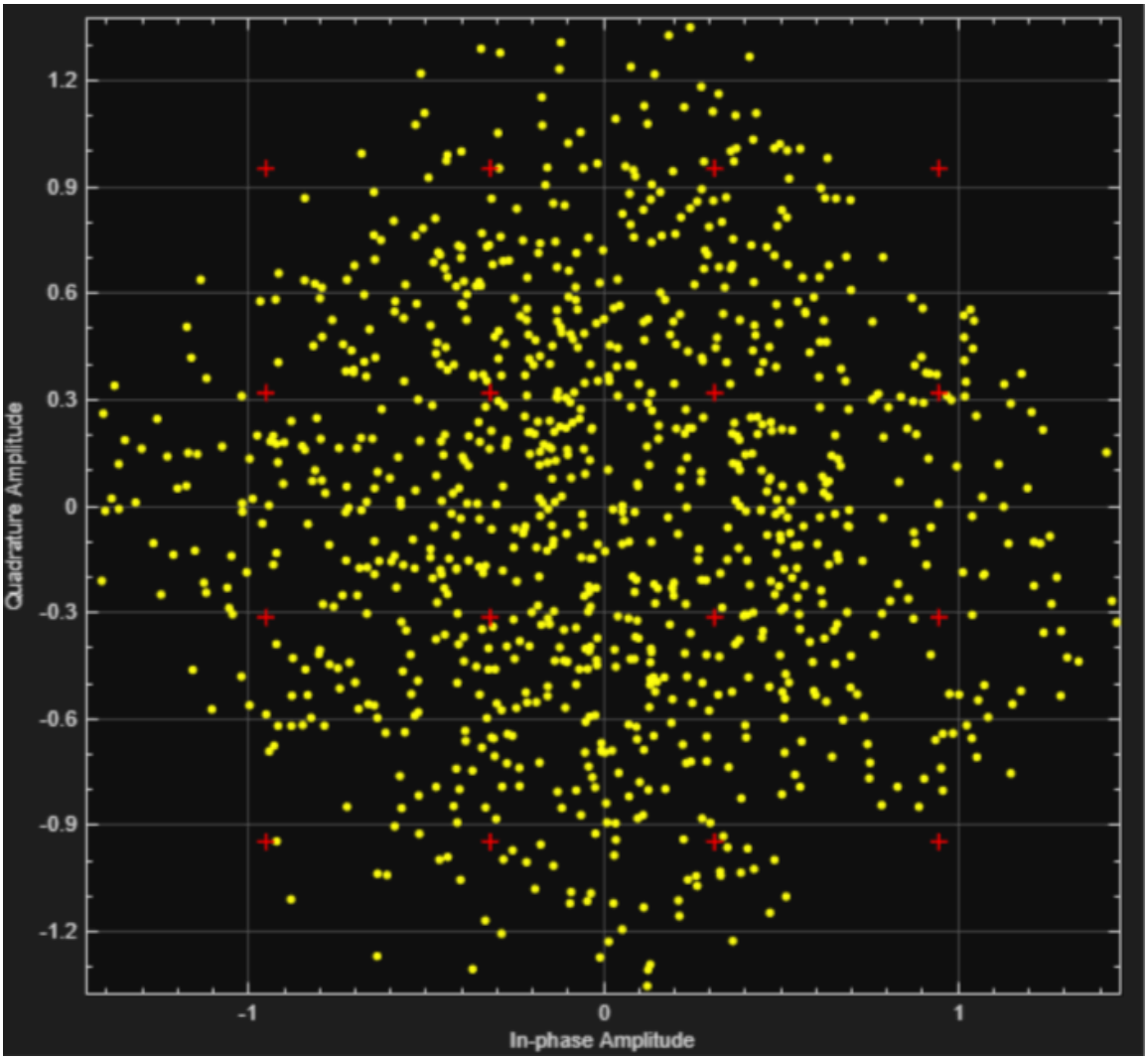


Figure 3-8 QAM modulation using plutoLoopback16QAM.m live signal.

The low signal spread demonstrates low noise. The end of the transmission in 3-8 shows low signal integrity.

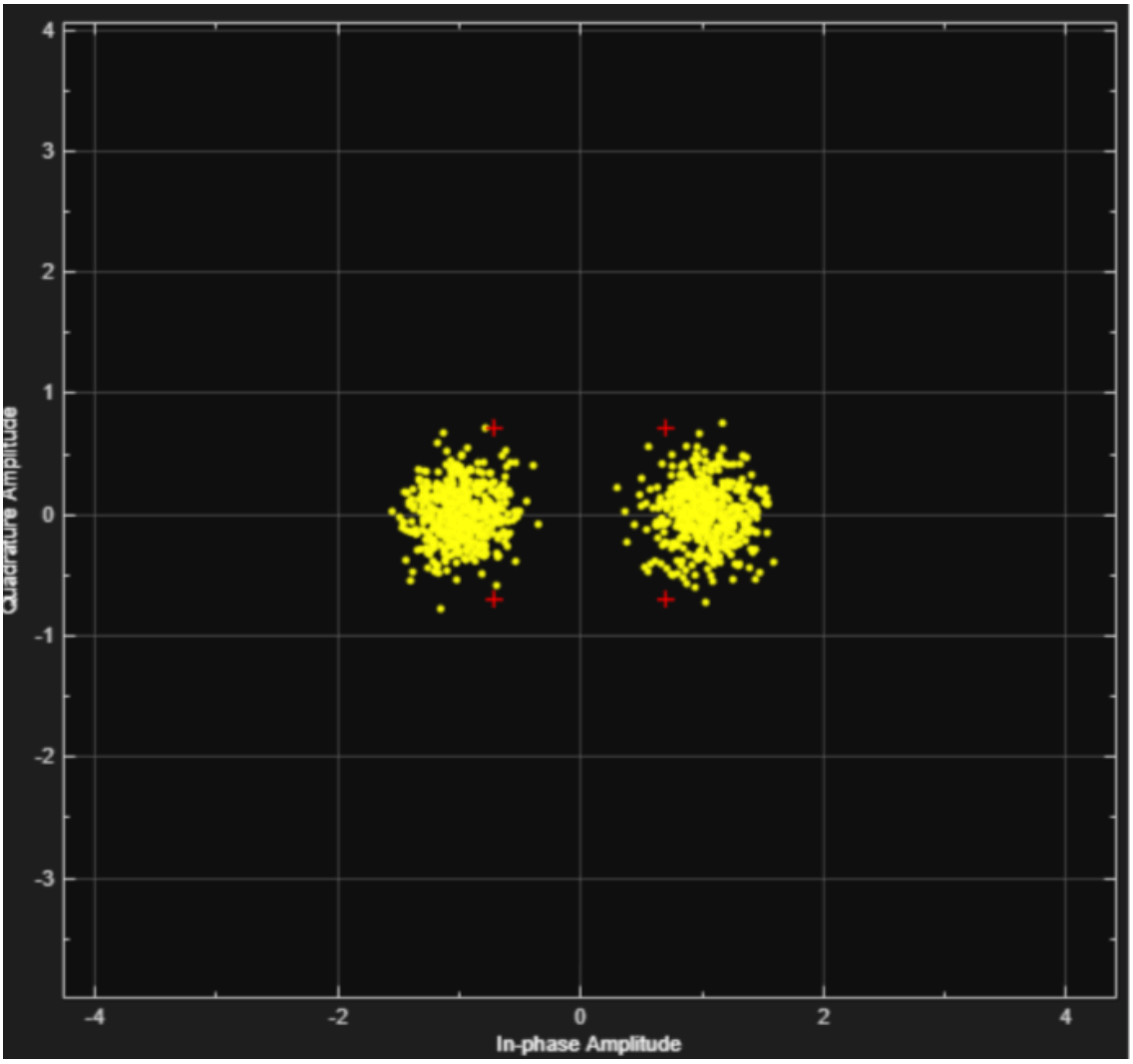
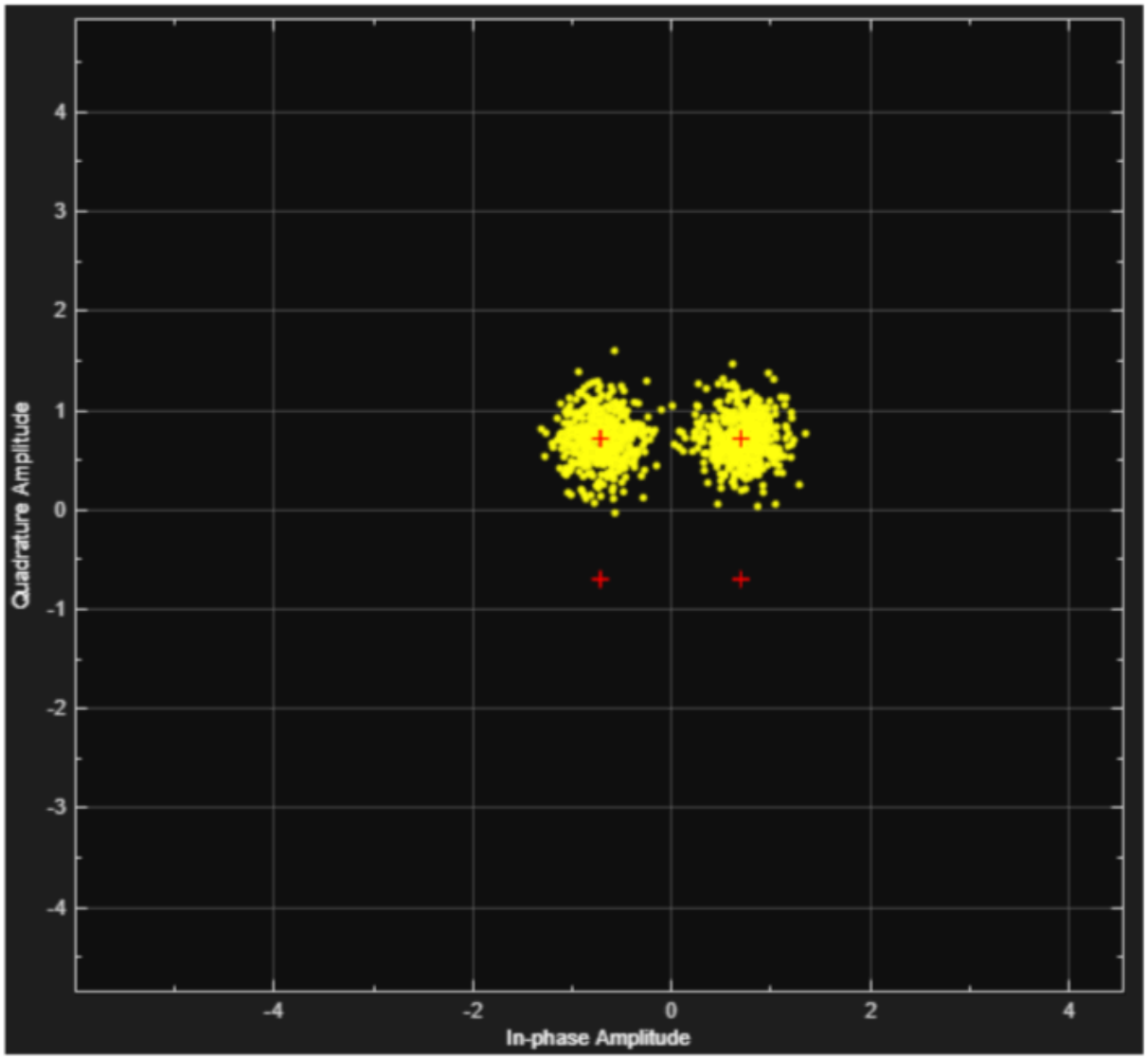


Figure 3-9 VPSK, Var = 0.1



’Figure 3-10 QPSK Var = 0,1

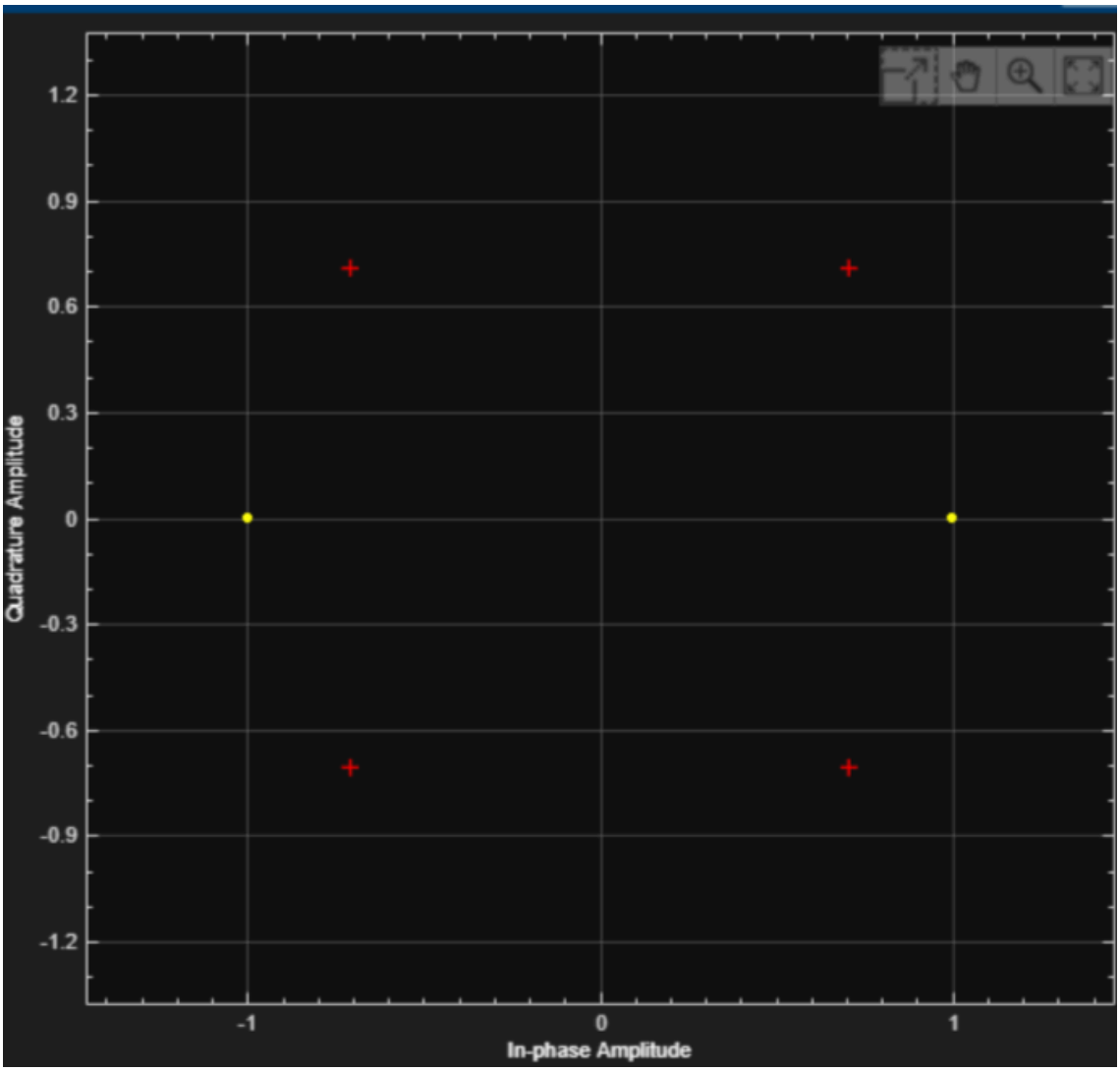


Figure 3-11 BPSK Var = 0.1

Error Rate		
BPSK 4.7e-6	47	1e7
APSK 0.001585	1.6e4	1e7

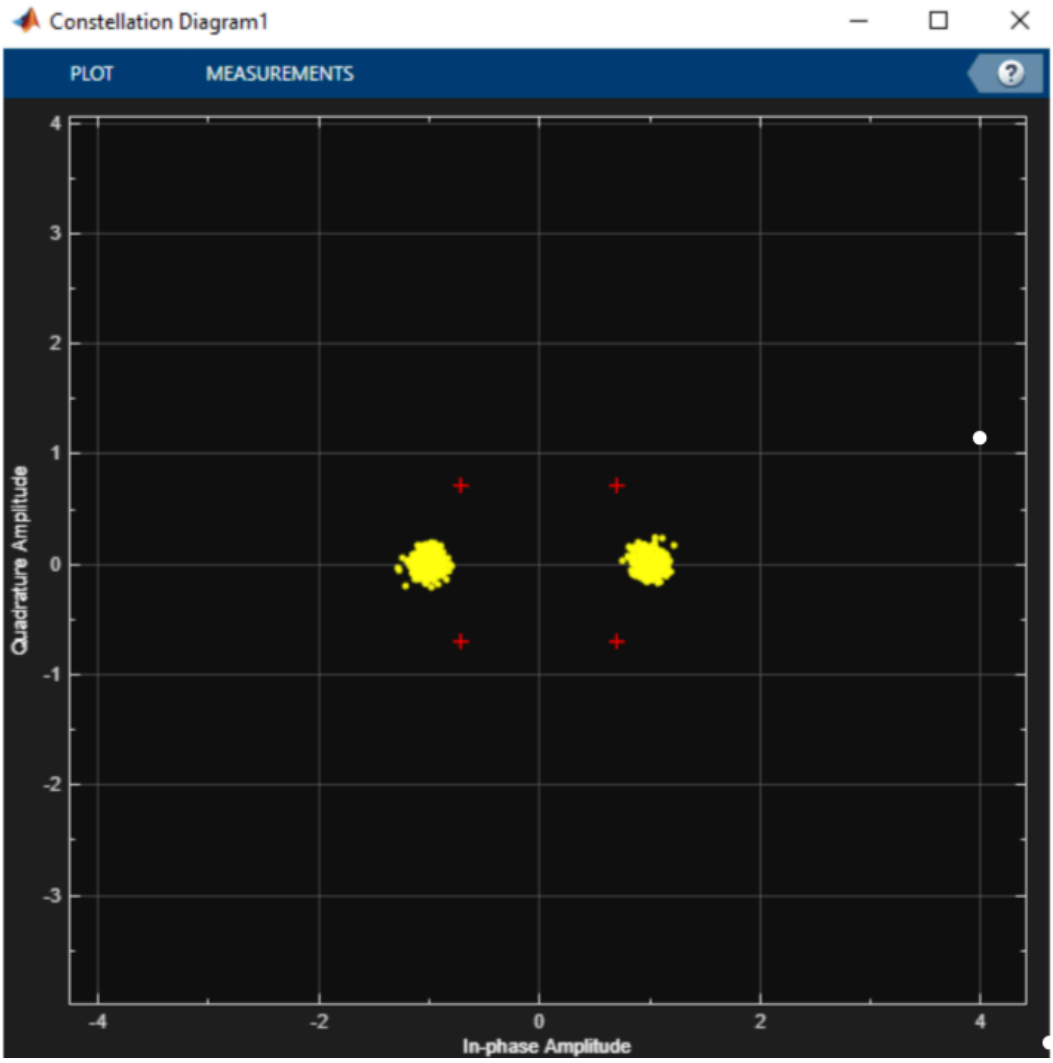


Figure 3-12 BPSK Var = 0.01

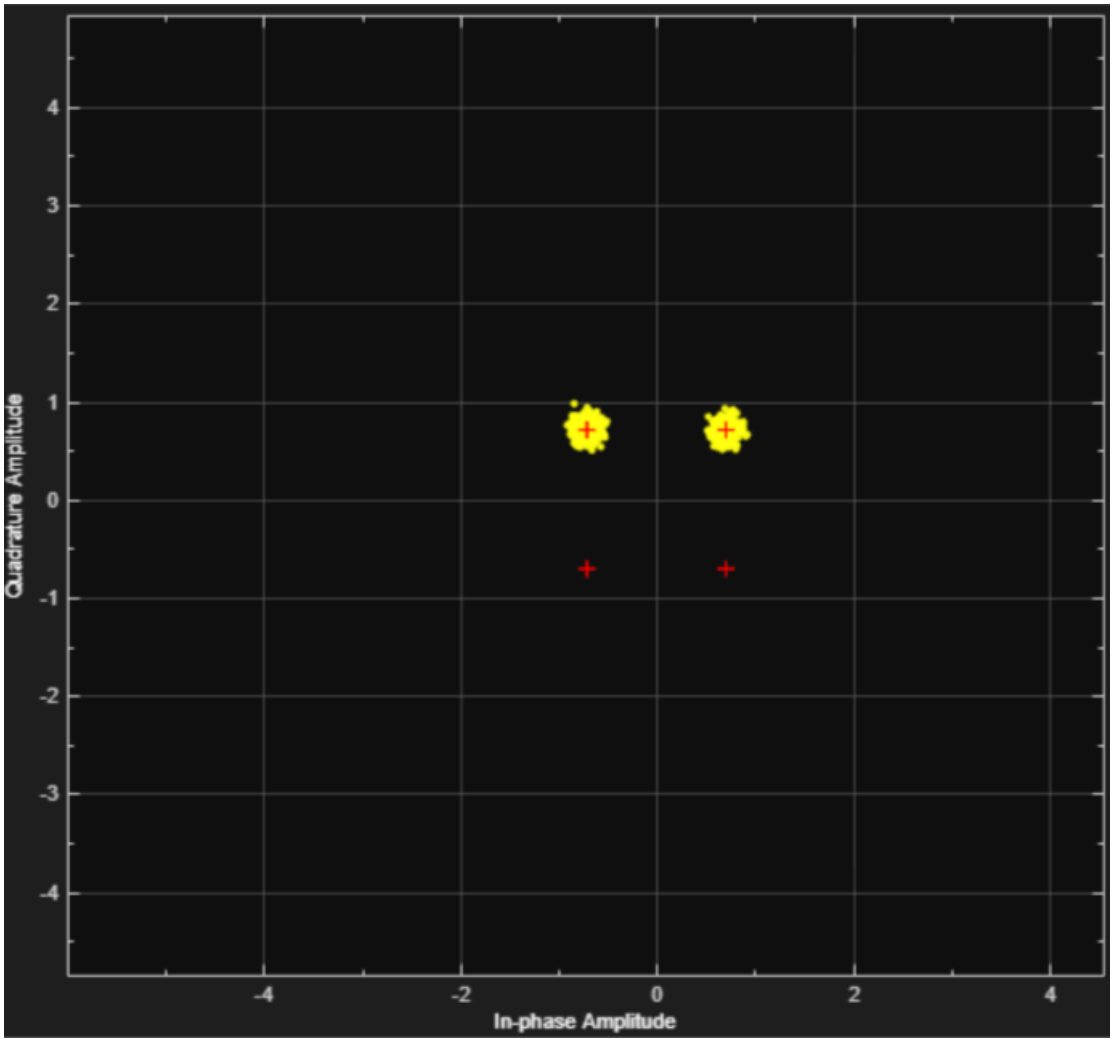


Fig 3-12 QPSK Var = 0.01

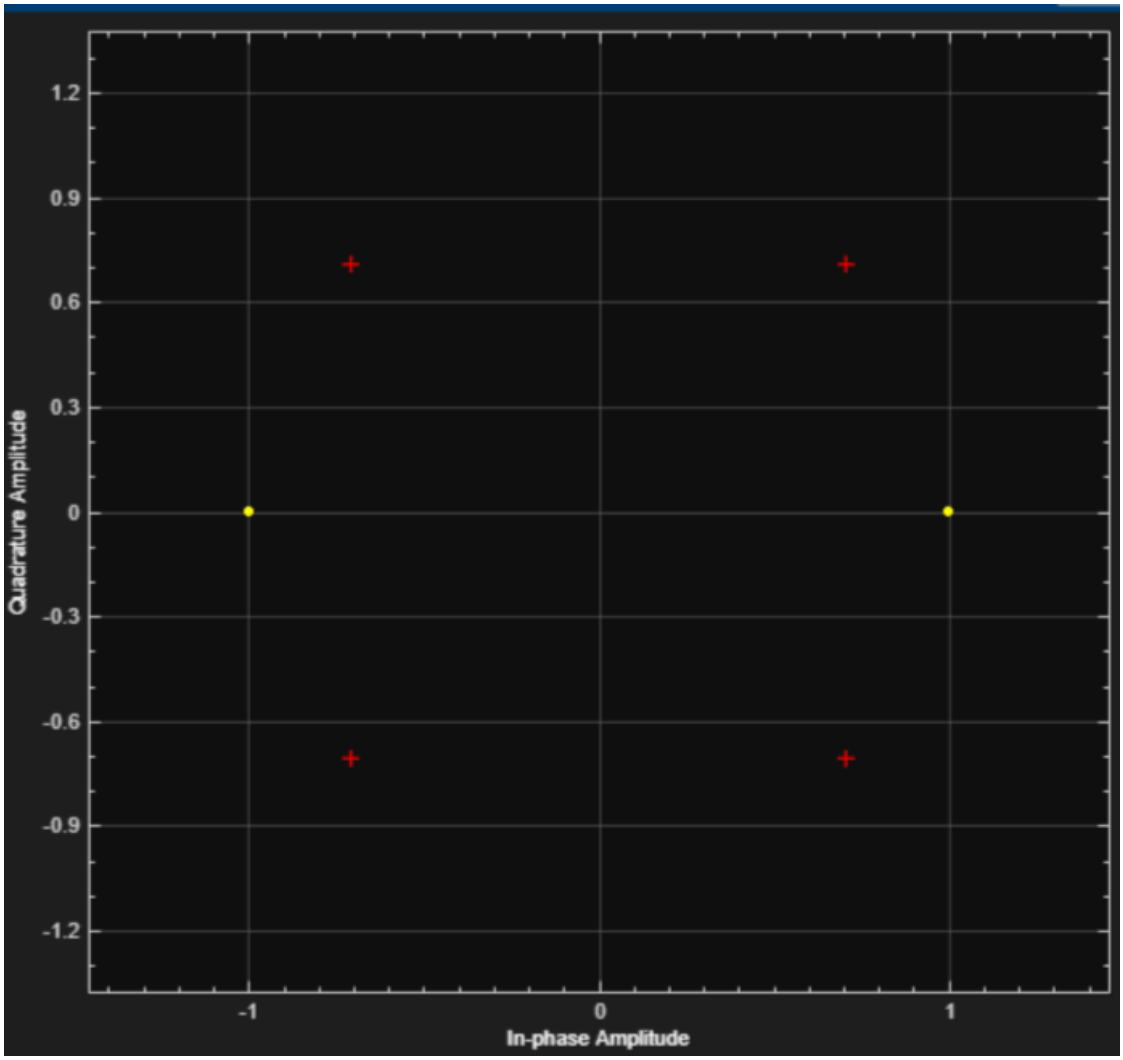


Figure 3-13 BPSK Variance = 0.01

Error Rate

BPSK 1e7

QPSK 1e7

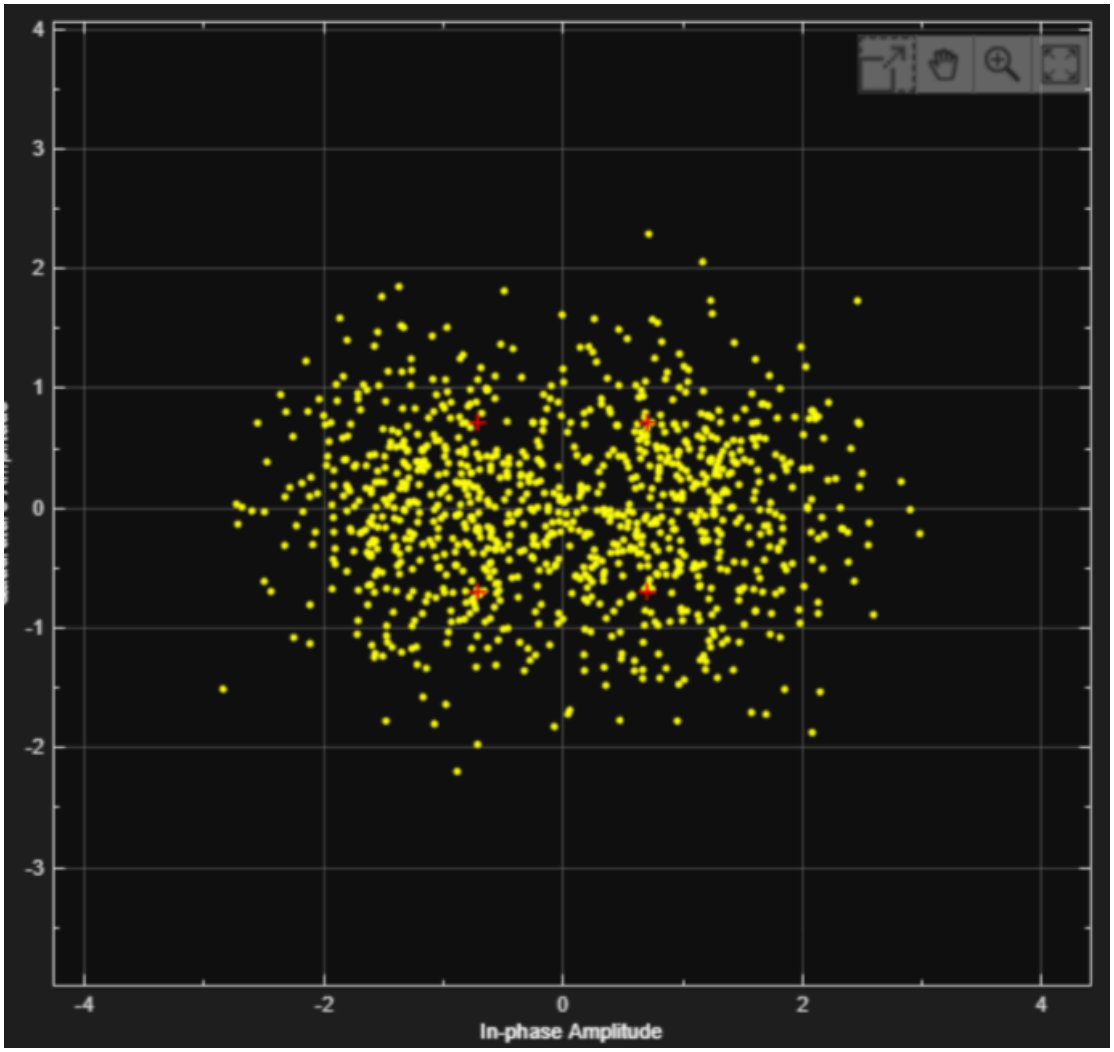


Figure 3-14 BPSK Var = 1

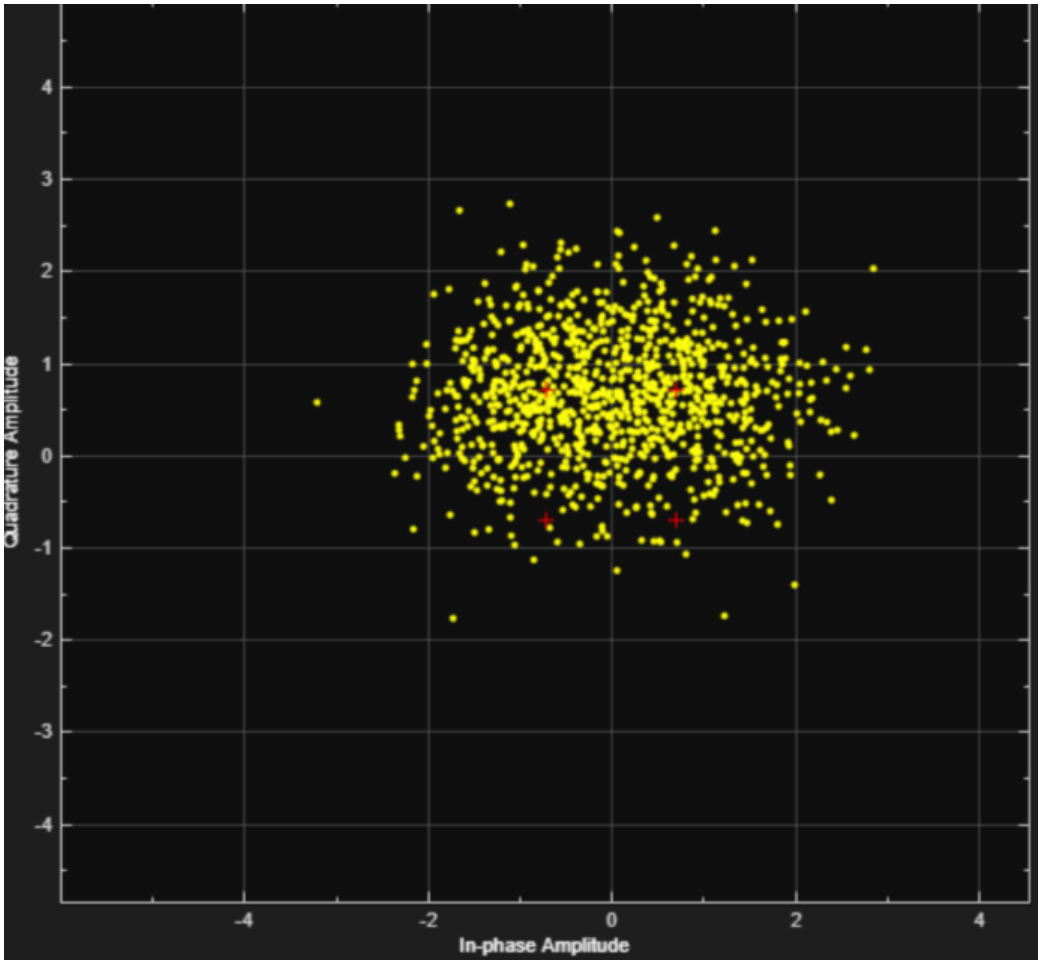


Fig 3-15 Var = 1 QPSK

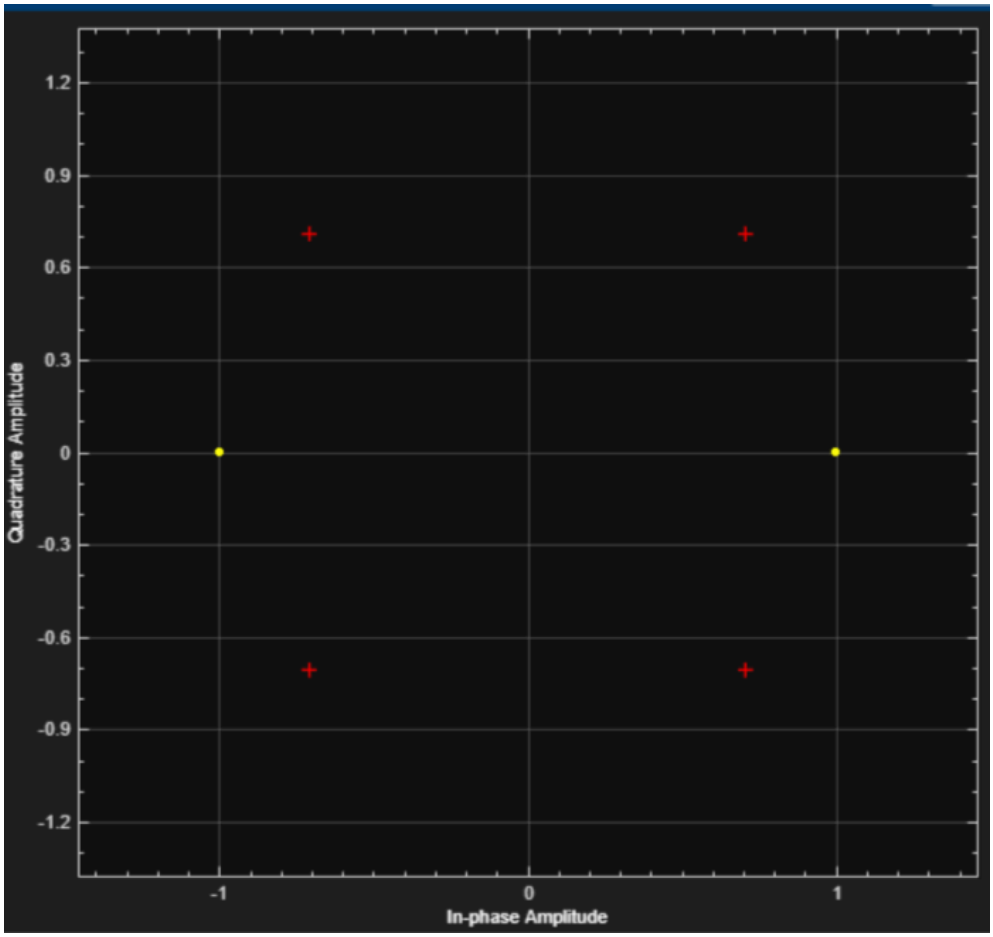
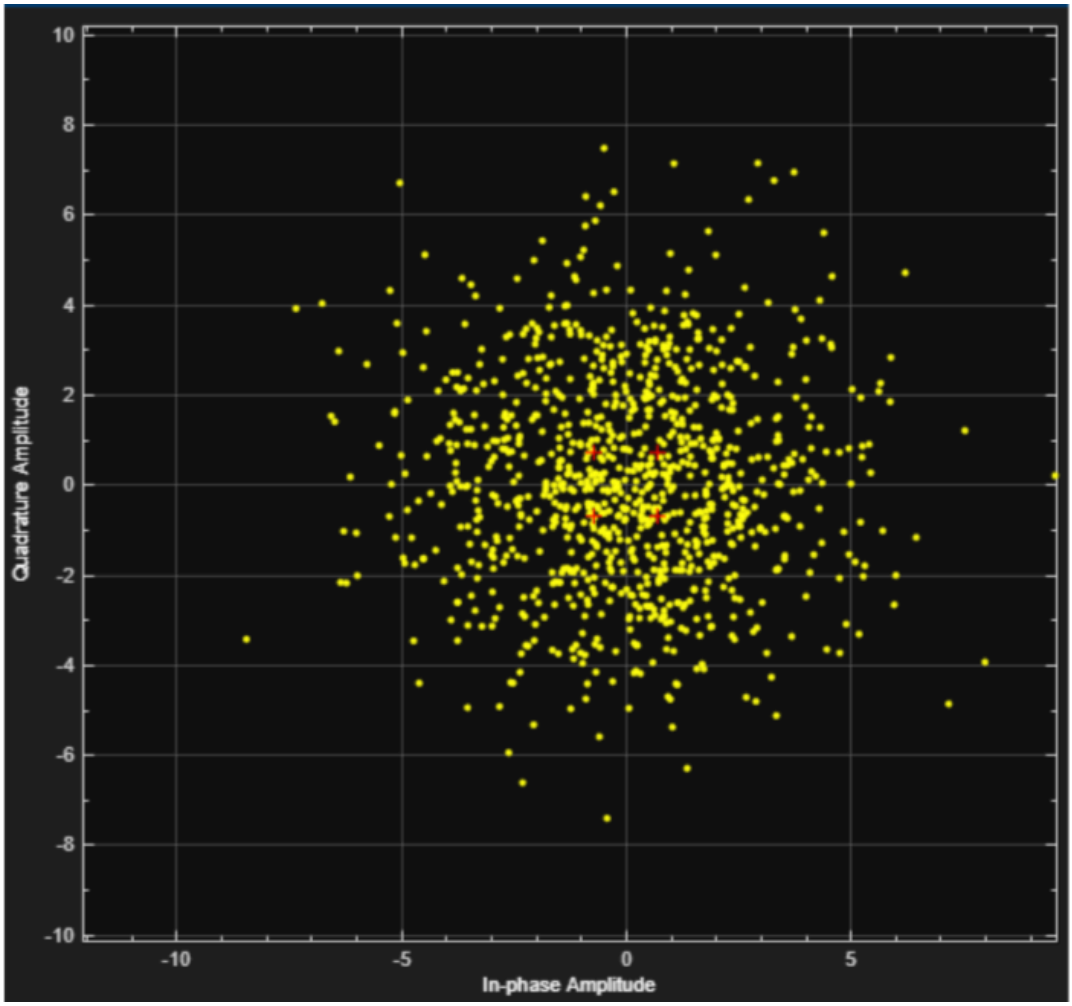


Fig 3-16 Var =1 BPSK



3-17 Var =10 BPSK

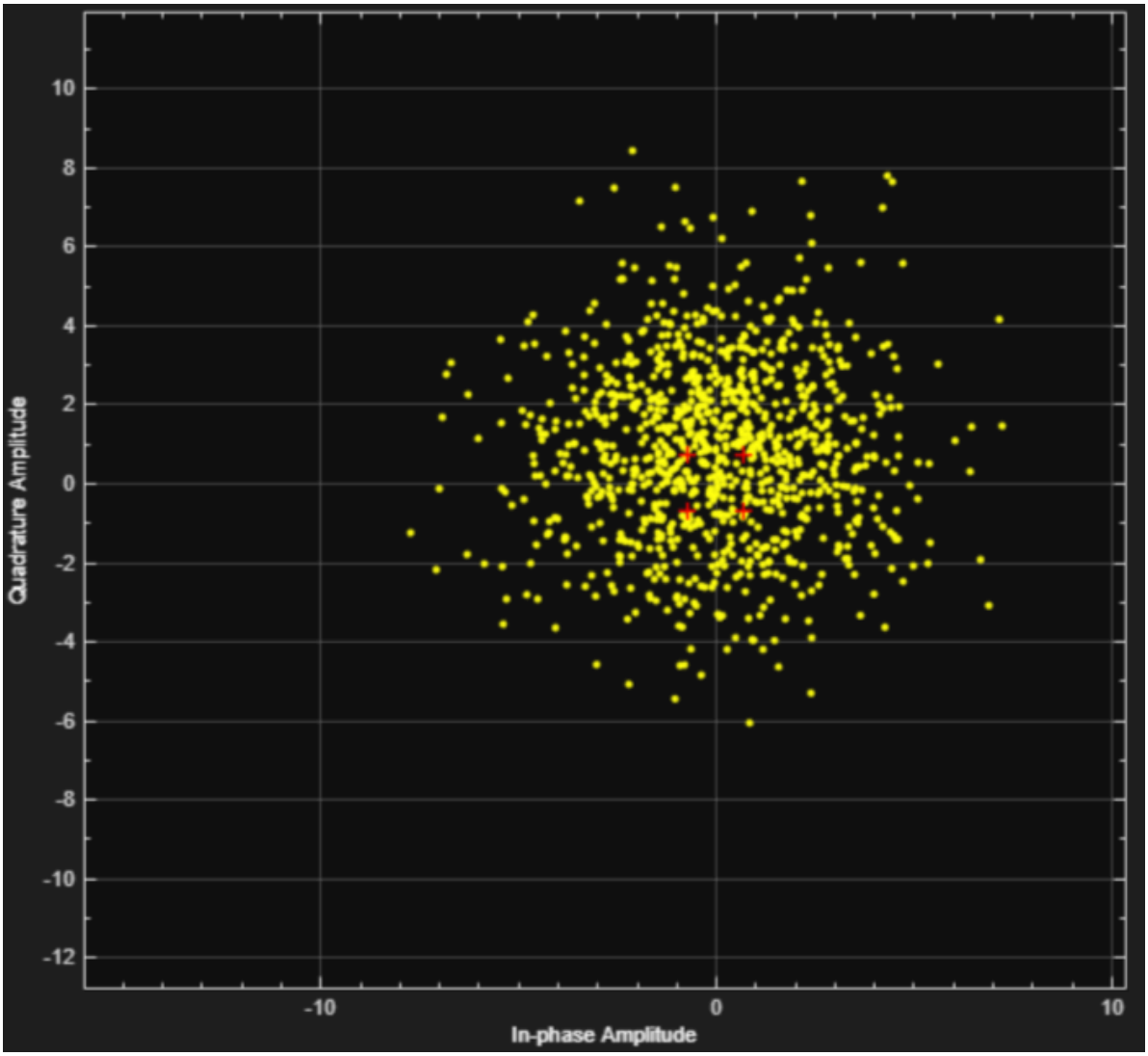


Fig 3-18 QPSK Var =10

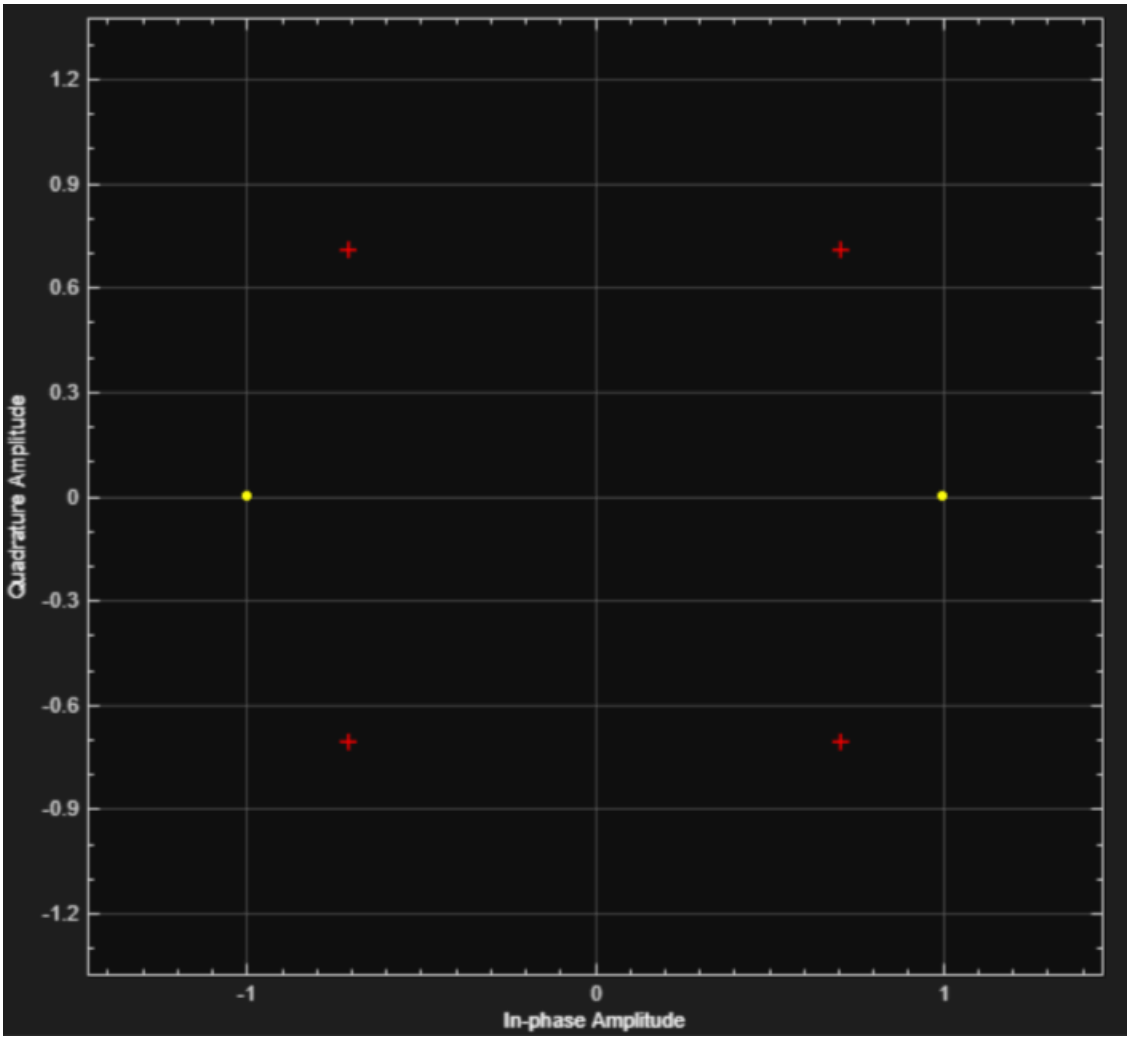


Fig 3-19 BPSK Var =10

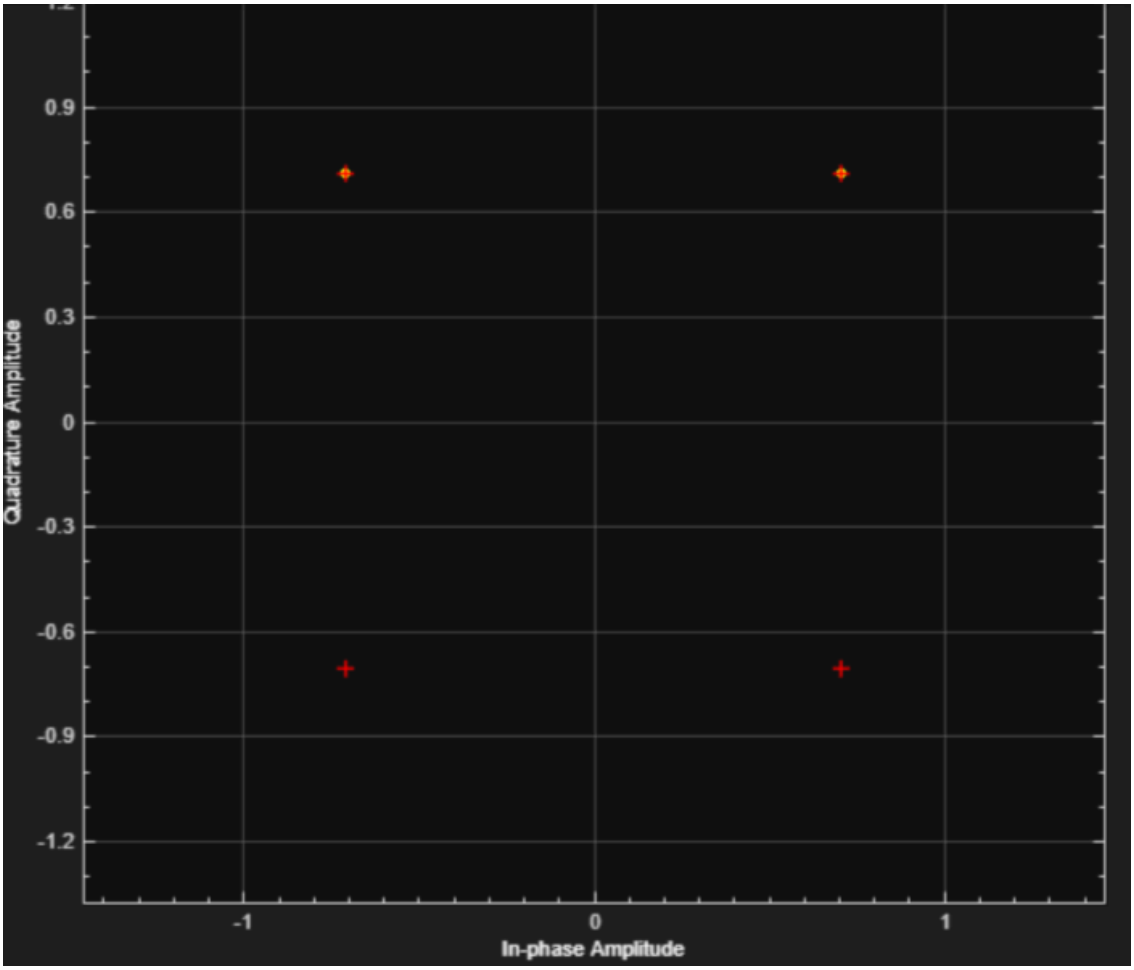


Fig 3-20 QPSK Var =10

Error Rate: BPSK 0.33

QPSK = 0.61

Questions:

What is a reference constellation?

- A reference constellation is a way to demonstrate the quality of the sigal in communications.QAM changes the amplitude and phase while PAM changes the amplitude and PSK changes phase.

What happens in the constellation as the variance increases.

- The variation makes the diagram spread leading to more noise and a distorted signal.

What happens to the error rate as the variance increases?

- Higher noise and an increased error rate

When the number of symbols, the distance between adjacent symbols becomes less. What implication does this have on the noise susceptibility and error rate?

- Less symbols means more means for noise to interfere with the signal. The error rate goes up and lets neighboring points interfere with each other.

What would be a similar quantitative measure to BER for analog signals.

- Noise figure NF and SNR are similar to BER as they can measure the signal noise and integrity.

ELECTRICAL ENGINEERING DEPARTMENT
California Polytechnic State University
San Luis Obispo

EE 504 Experiment #4

Transmit and Receive

Objective Learn how to operate and understand the Raspberry PI 5.

Equipment: Raspberry PI 5, Mouse, Keyboard

Password: sdr

Username: sdr

Results:

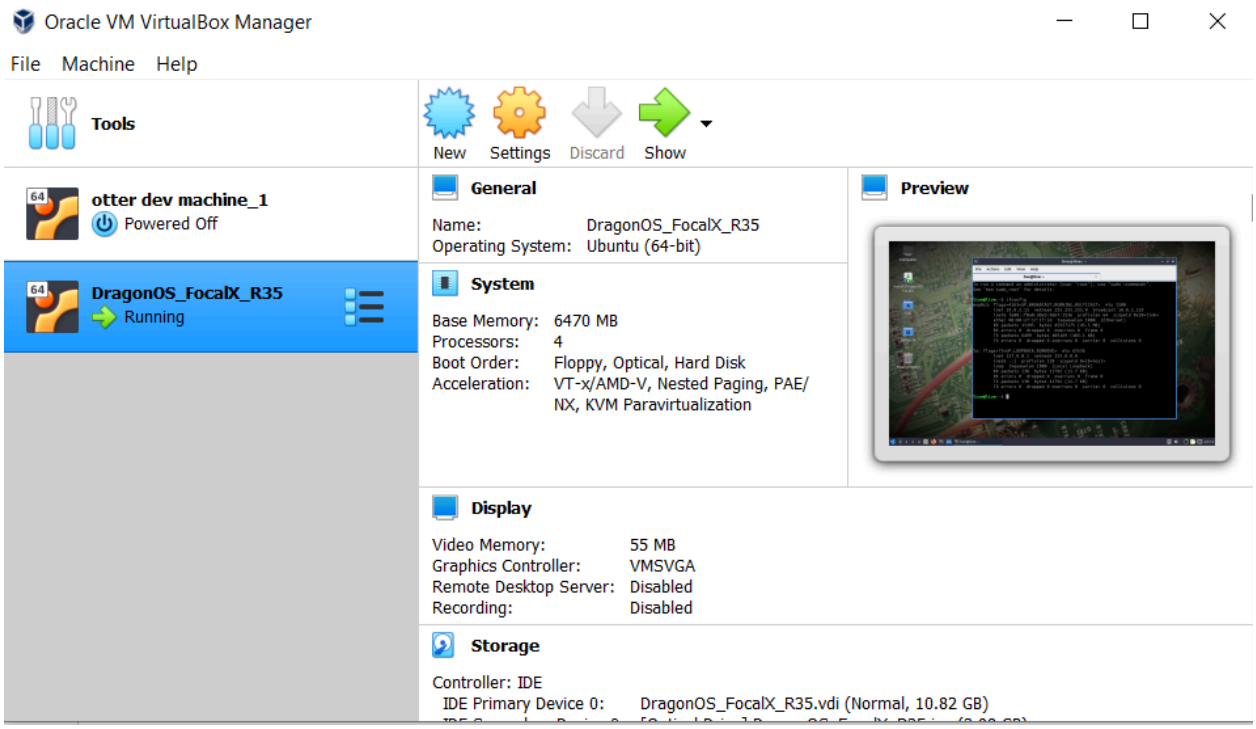


Figure 4-1: Download the dragon OS and upload to the Oracle VirtualBox

Ensure to allocate at least 4 cores, 10+GB of storage and as much memory as your laptop comfortably allows.

Open the terminal and extract the MAC address

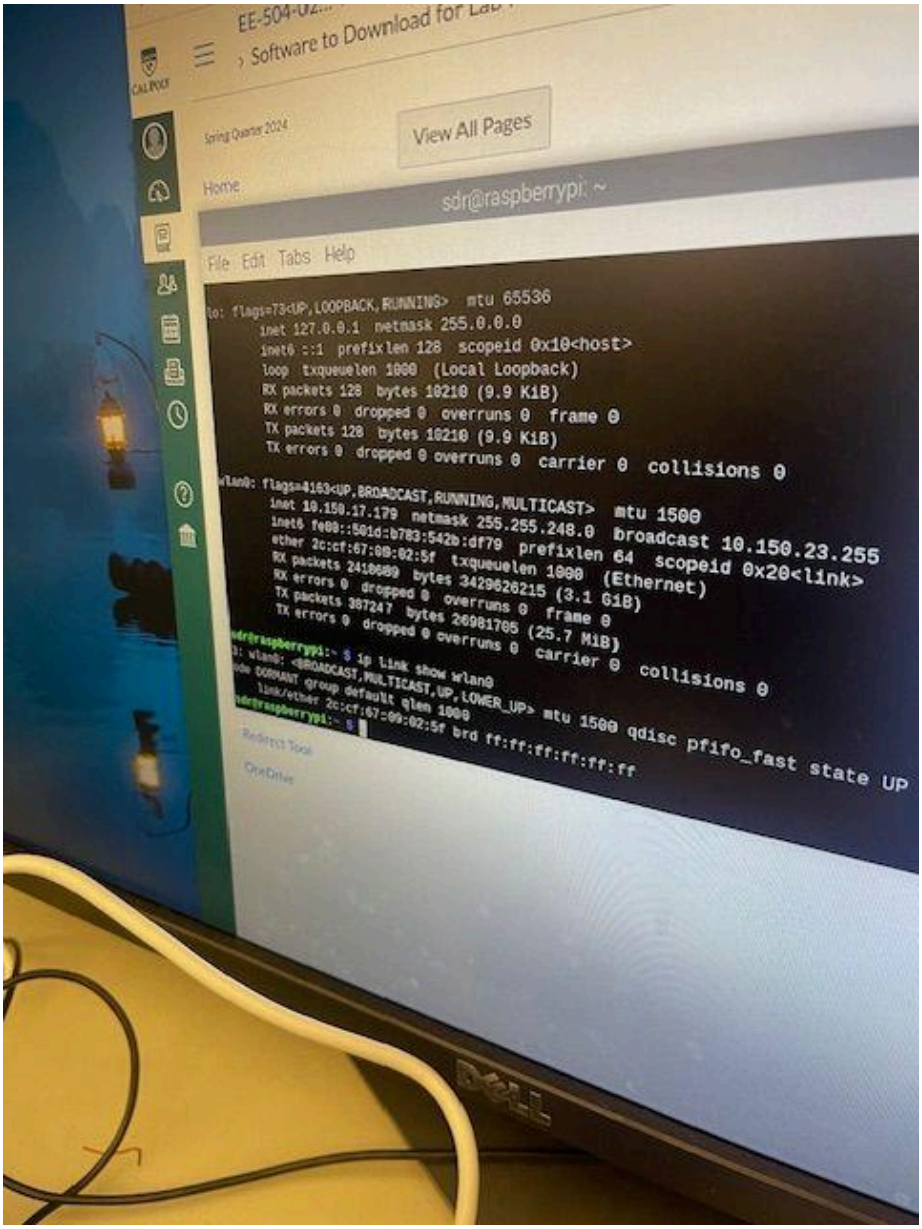


Figure 3-2 Extract the MAC address.

Flash Dragon OS to the raspberry pi SD card.

New Dragon Default Username = ubuntu

Default Password = dragon

ELECTRICAL ENGINEERING DEPARTMENT

California Polytechnic State University
San Luis Obispo

EE 504 Experiment #5

Multiplexing and Orthogonal Frequency

Objective

Equipment: Pluto SDR, MATLAB, limeSDR

Results:

Exploration:

Figure 5-1

Figure 5-2

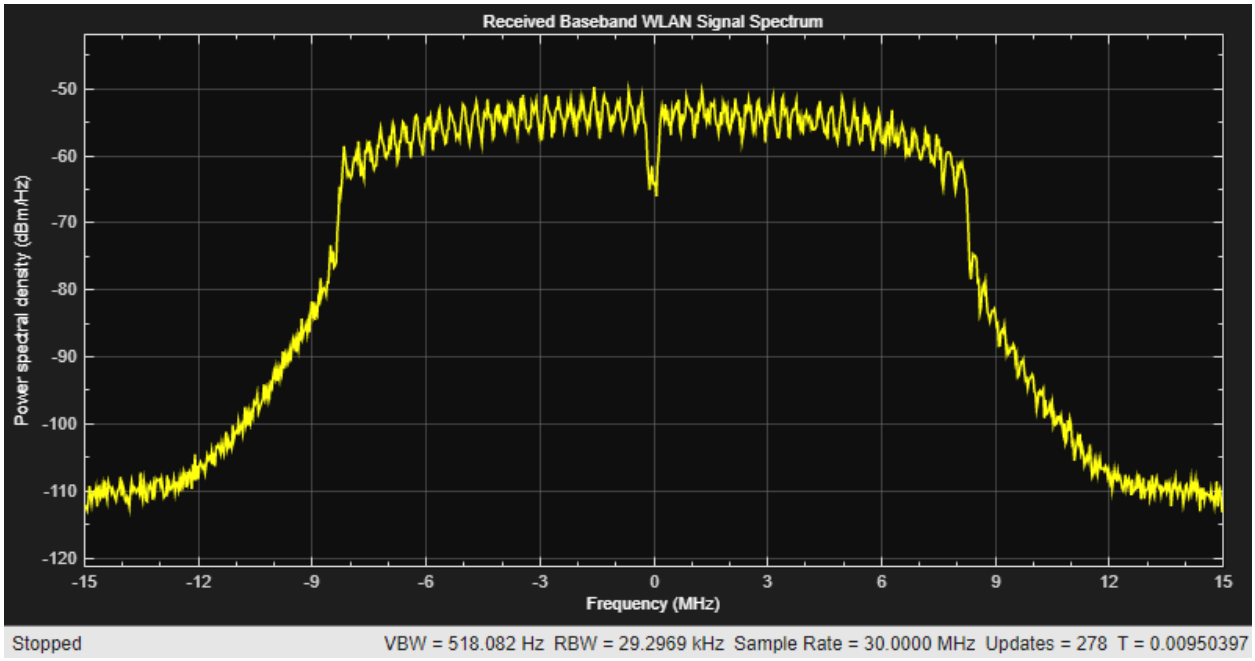


Figure 5-3

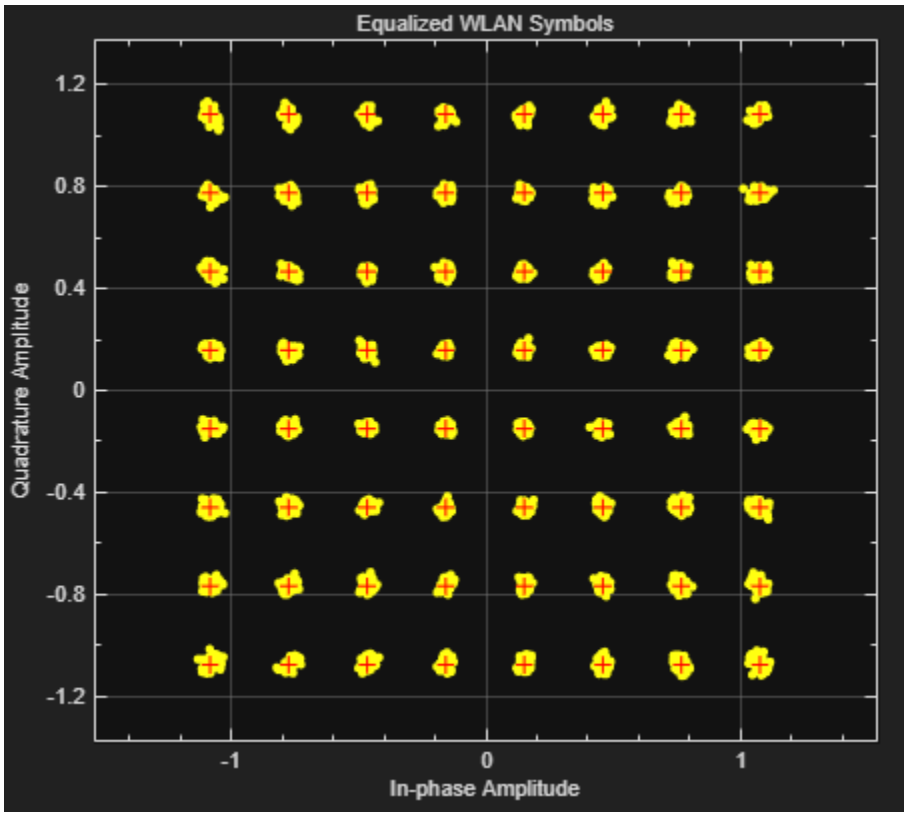


Figure 5-4

Figure 5-5 Using the kit antenna

Figure 5-6 Using the kit antenna

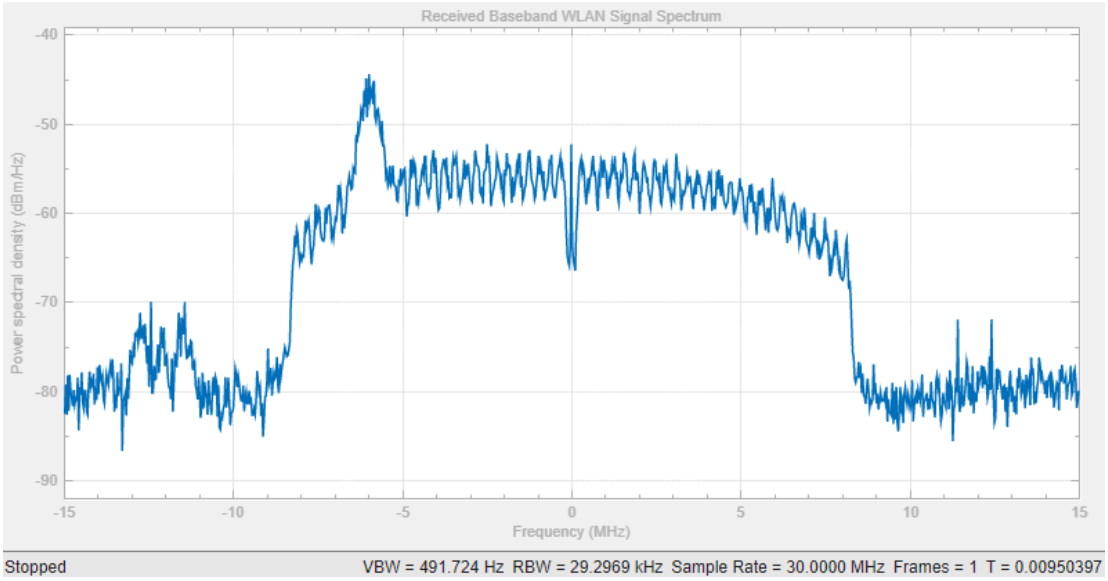


Figure 5-7 Using the kit antenna

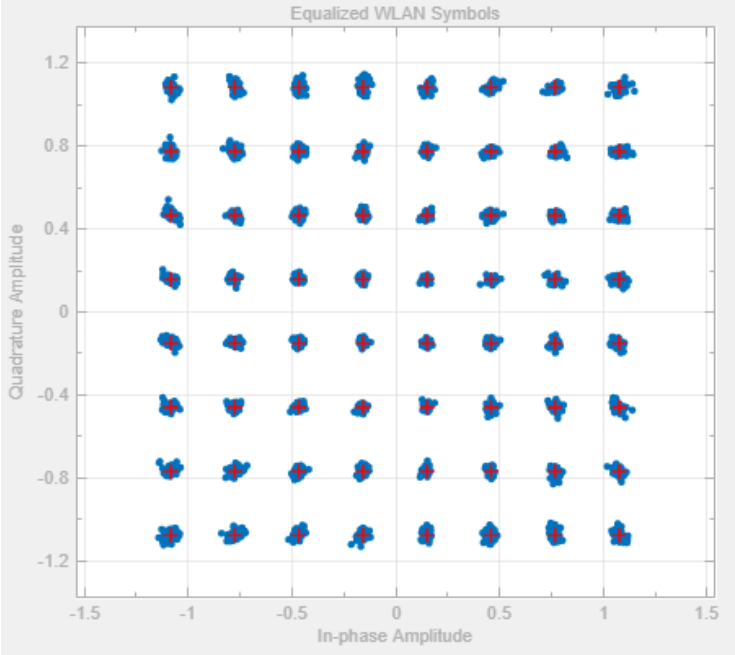


Figure 5-8 Using the kit antenna

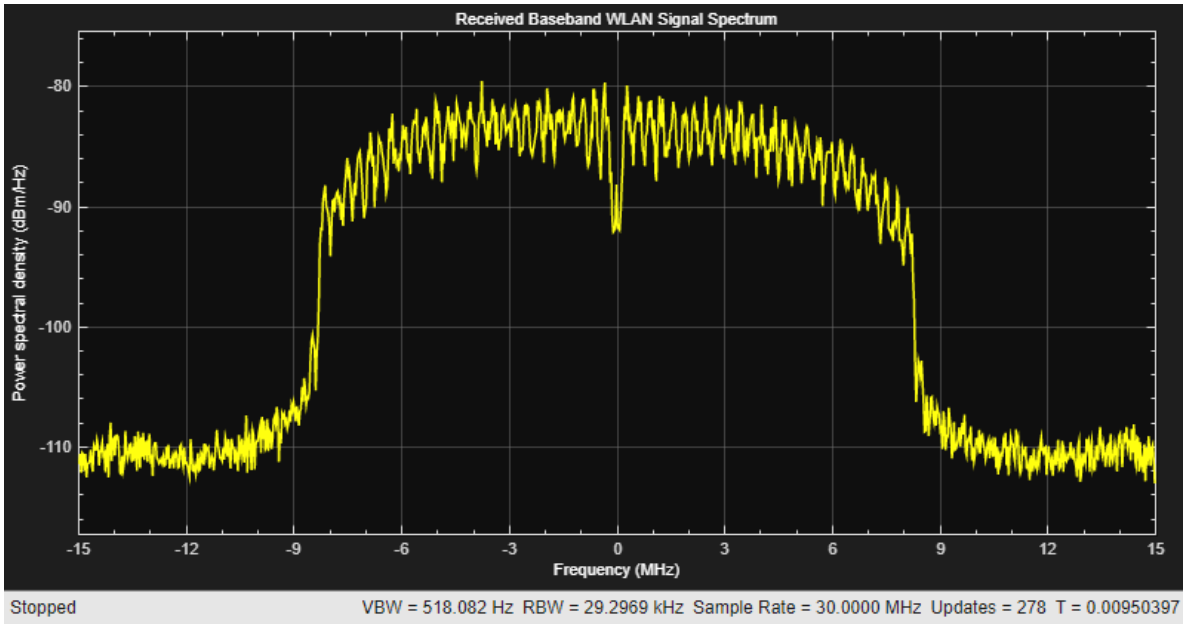


Figure 5-9 Using the kit antenna

- I (Ben Duval) took another kit antenna capture because I didn't really like the inverted color I went with when I originally ran this simulation back home. Doing it in my apartment, I seem to have gotten a lot more interference as I had to simulate it multiple times for it to give me a fairly accurate result.
- Jacqueline would like to add the wire antenna created during lab 2 at some point to improve results.

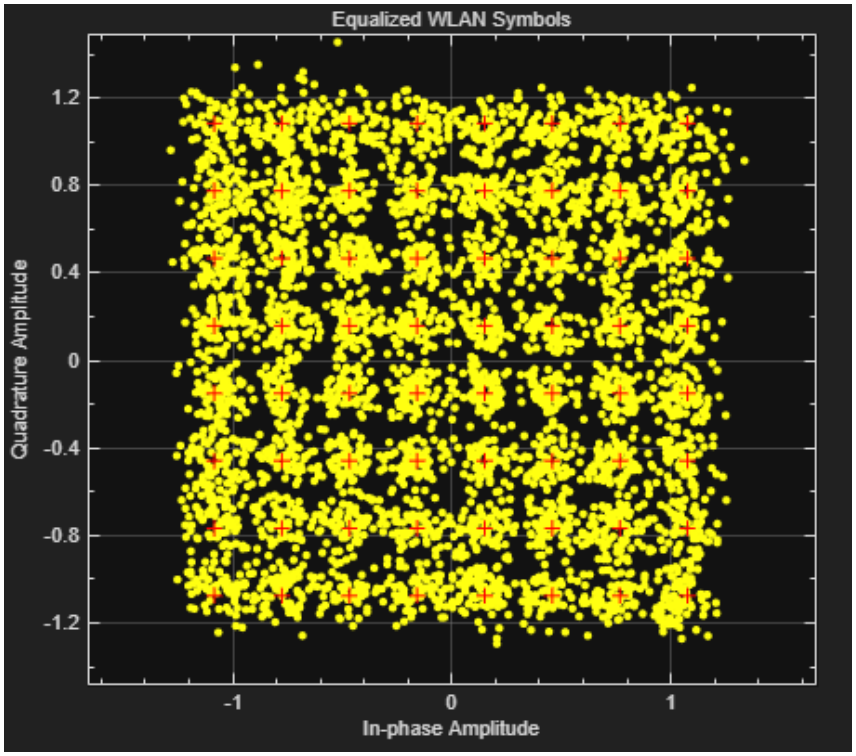


Figure 5-10 Using the kit antenna

Simulink:

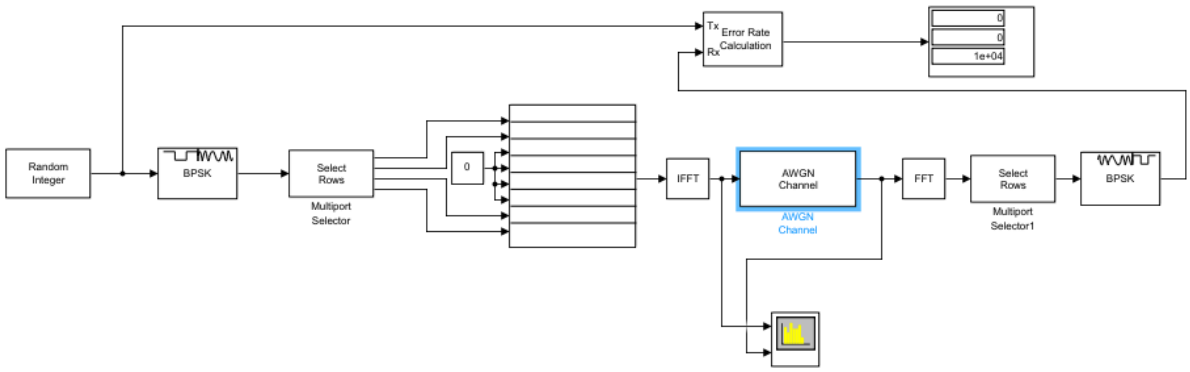


Figure 5-11

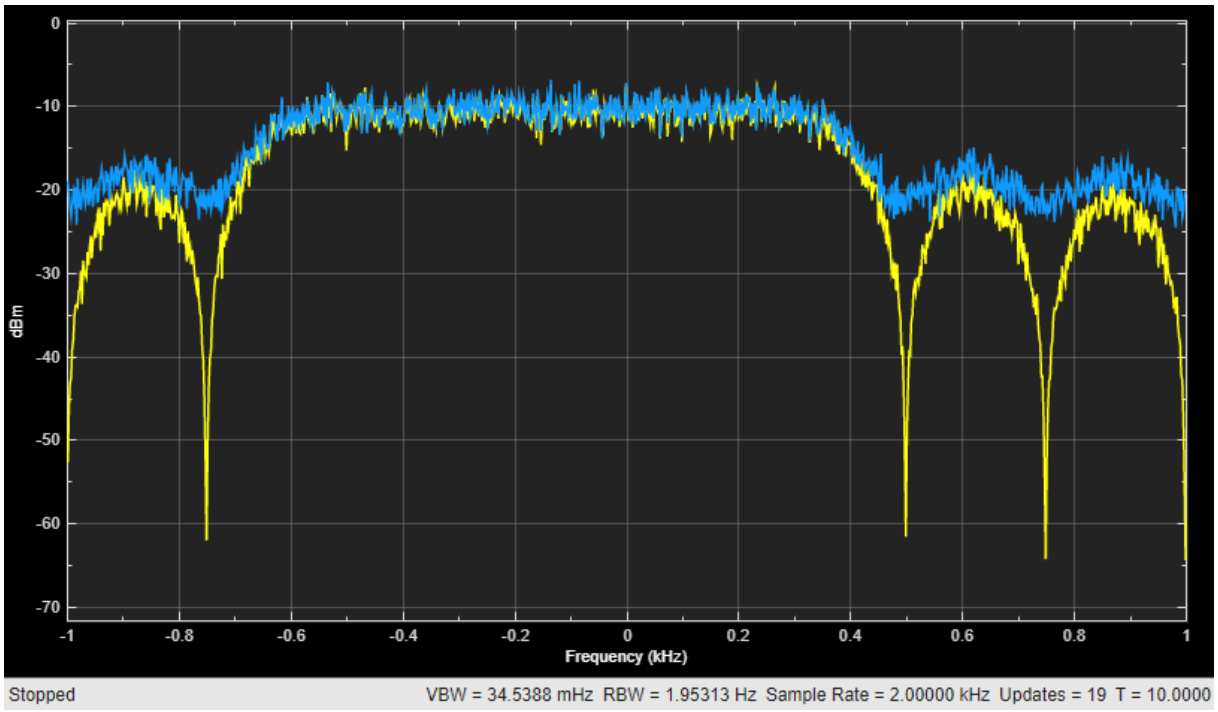


Figure 5-12

Variance = .01, Sample error rate = 0, error rate = 0, number of samples = 10000

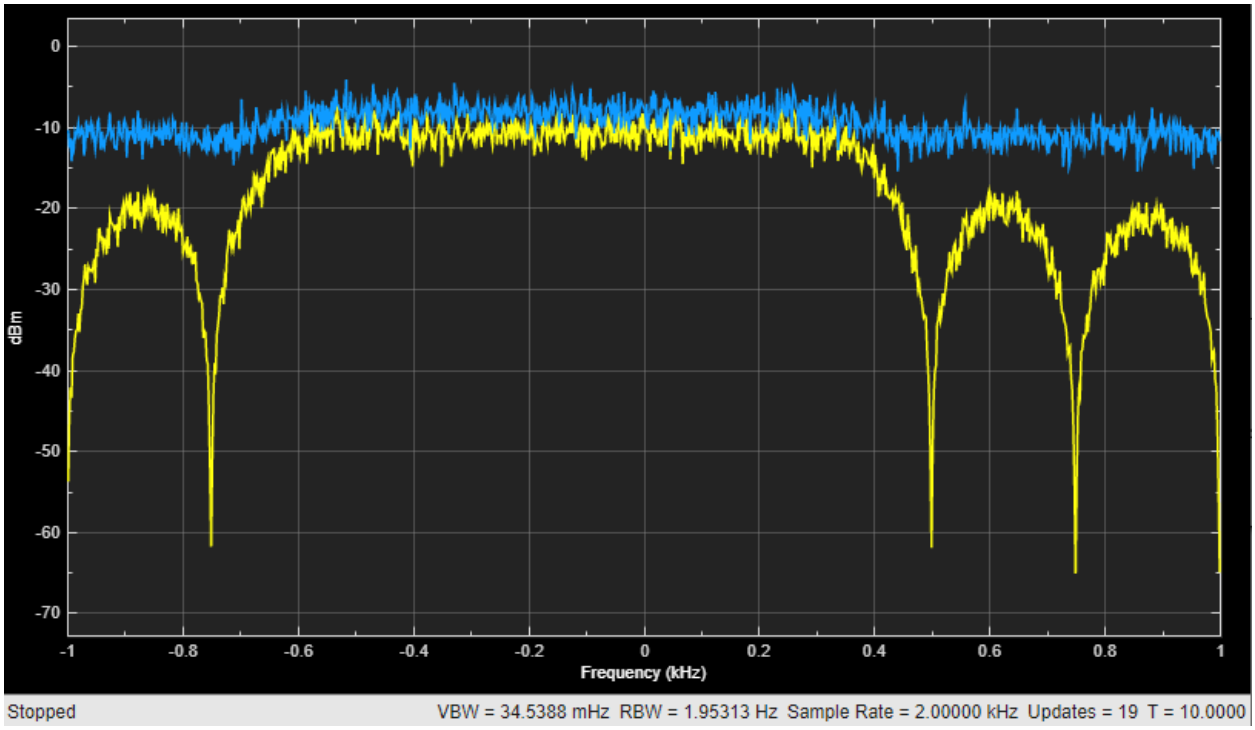


Figure 5-13

Variance = .1, Sample error rate = .05188, error rate = 519, number of samples = 10000

Conclusion:

It is important to understand how multiplexing and OFDM work. The implementation and application of what demodulation and modulation mean. FFT and IFFT are forms of demodulation and modulation. Also, exploring how to differentiate the start and end of the signals are important.

Questions:

Which part of the Simulink model represents the OFDM modulator? The demodulator?

- The OFDM modulator part of the Simulink model would consist of the following blocks: BPSK modulator, Multiport Selector, Vector concatenation, and the IFFT.
- The demodulator part of the SImulink model would consist of the FFT, Multiport selector, and the BPSK demodulator block.

Why did we inject zeros between the data streams?

- The zeros are injected between the data streams to help ensure there is no interference of data. They act like the guard band between symbols.

- Similar to the barker codes, the program knows when the signal starts and ends.

Describe the steps for designing a WLAN transmitter and for designing a WLAN receiver.

- Transmitter: Modulation, encoding, hardware transmission
- Recliever: hardware reception, there needs to be an analog to digital converter, signal processing for recovery, demodulation(FFT)/decoding

ELECTRICAL ENGINEERING DEPARTMENT
California Polytechnic State University
San Luis Obispo

EE 504 Experiment #6

GNU Radio Blocks and Flowgraphs

GNU Radio Block Basics

Low Pass Filter

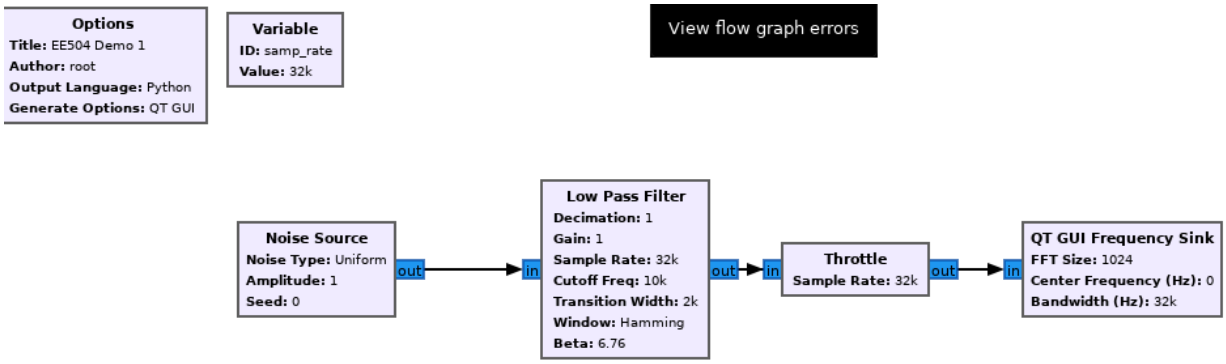


Figure 1: Low pass filter created in the GNU lab

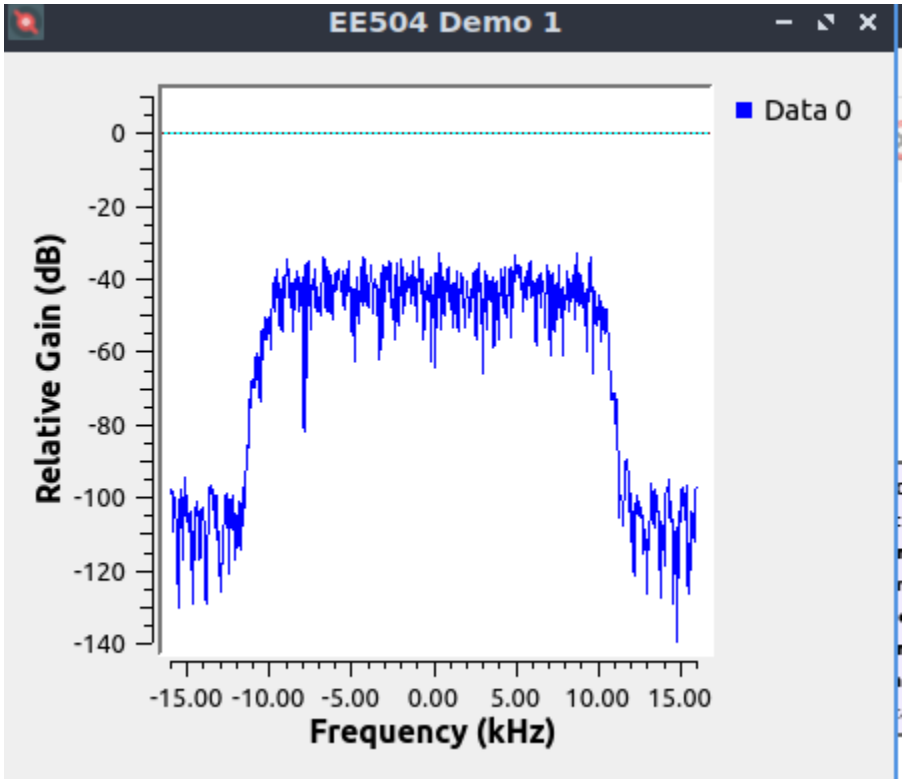


Figure 2: LPF results using VMbox in GNU

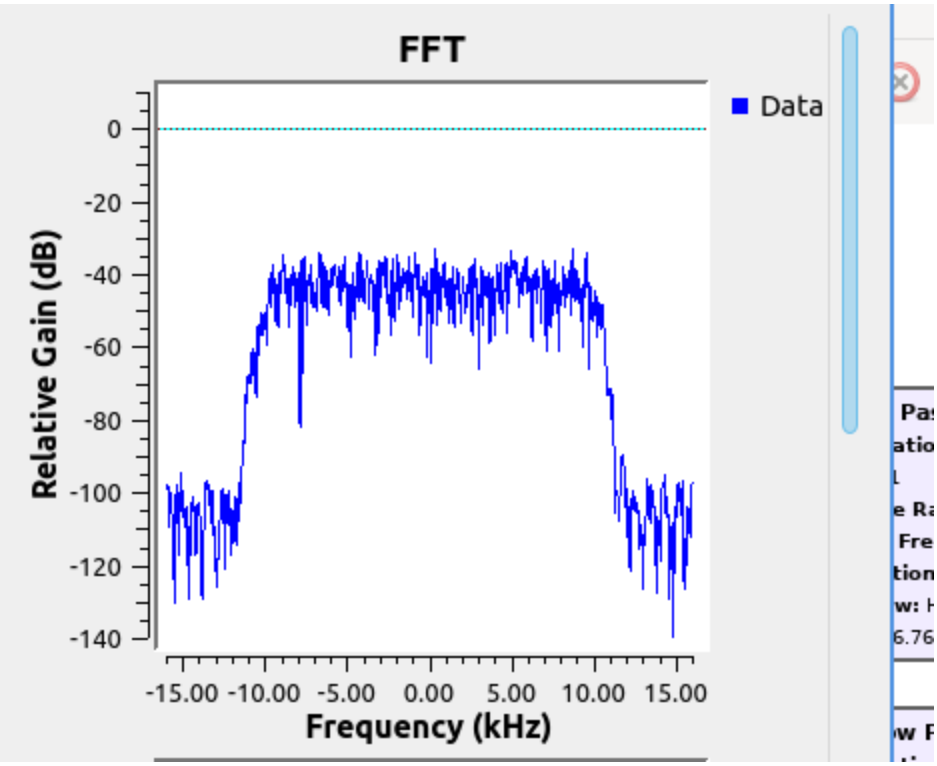


Figure 3: FFT Low pass filter results

Low pass filter with FFT

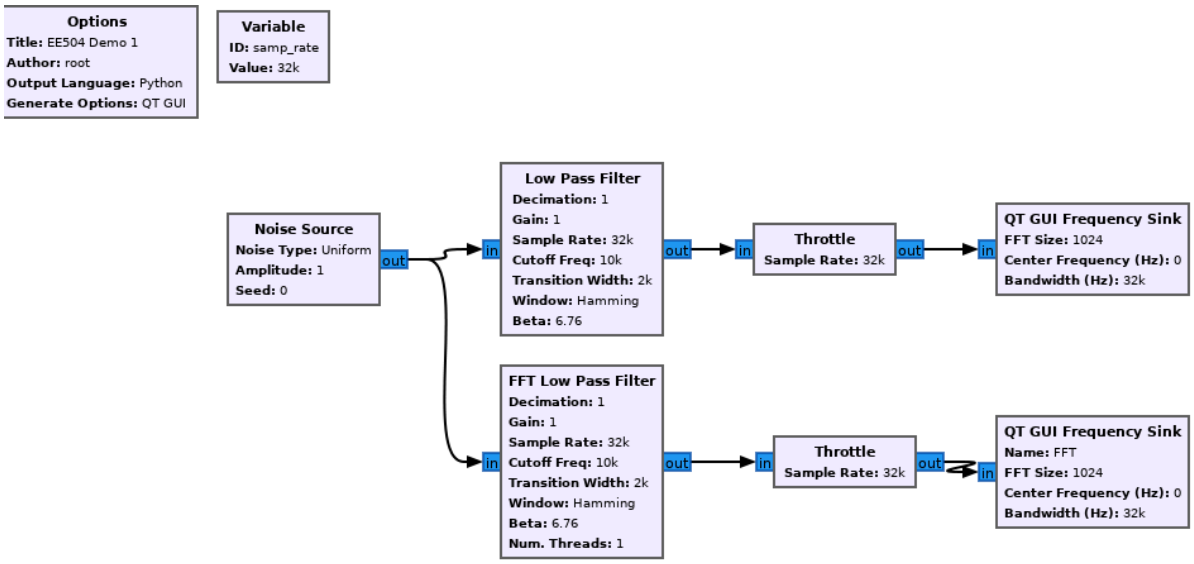


Figure 4: FFT low pass added branch

Low pass filter with added sliders

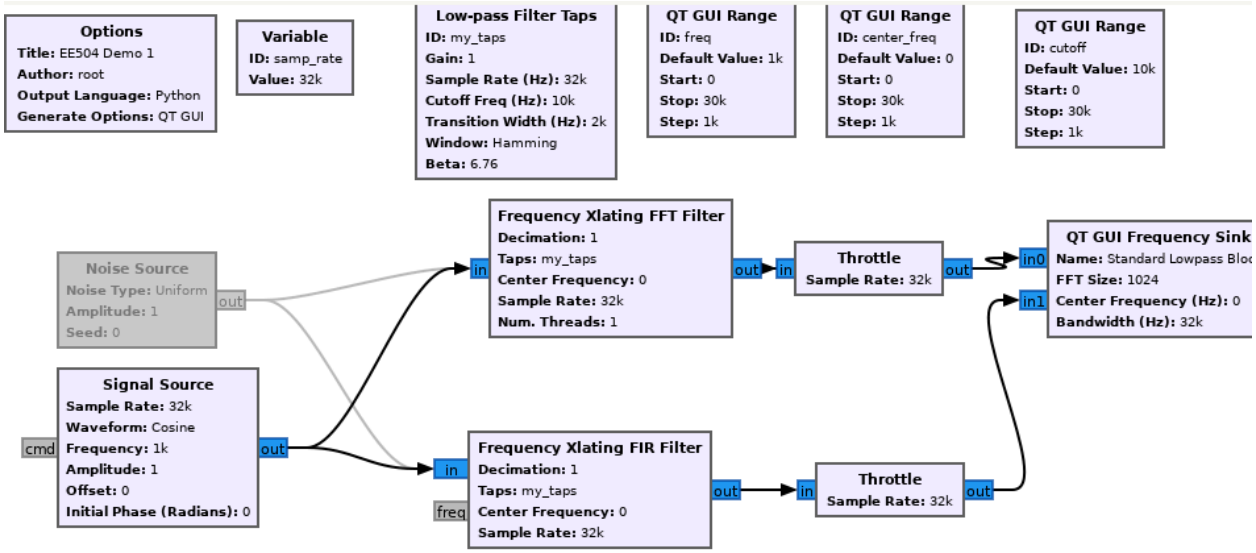


Figure 5: Adding sliders using the FFT and FIR filters

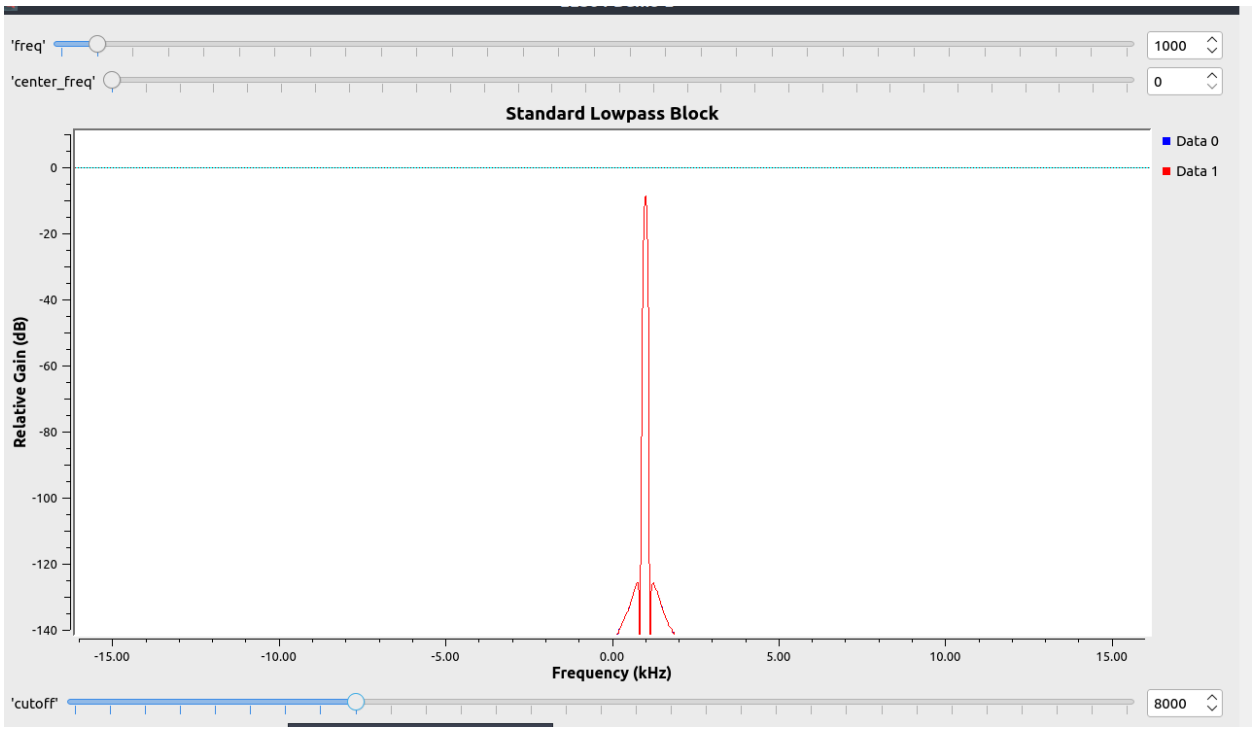


Figure 6: Center frequency is 0, sliders included, data 0 blue represents the FFT and data 1 red represents the FIR filter.

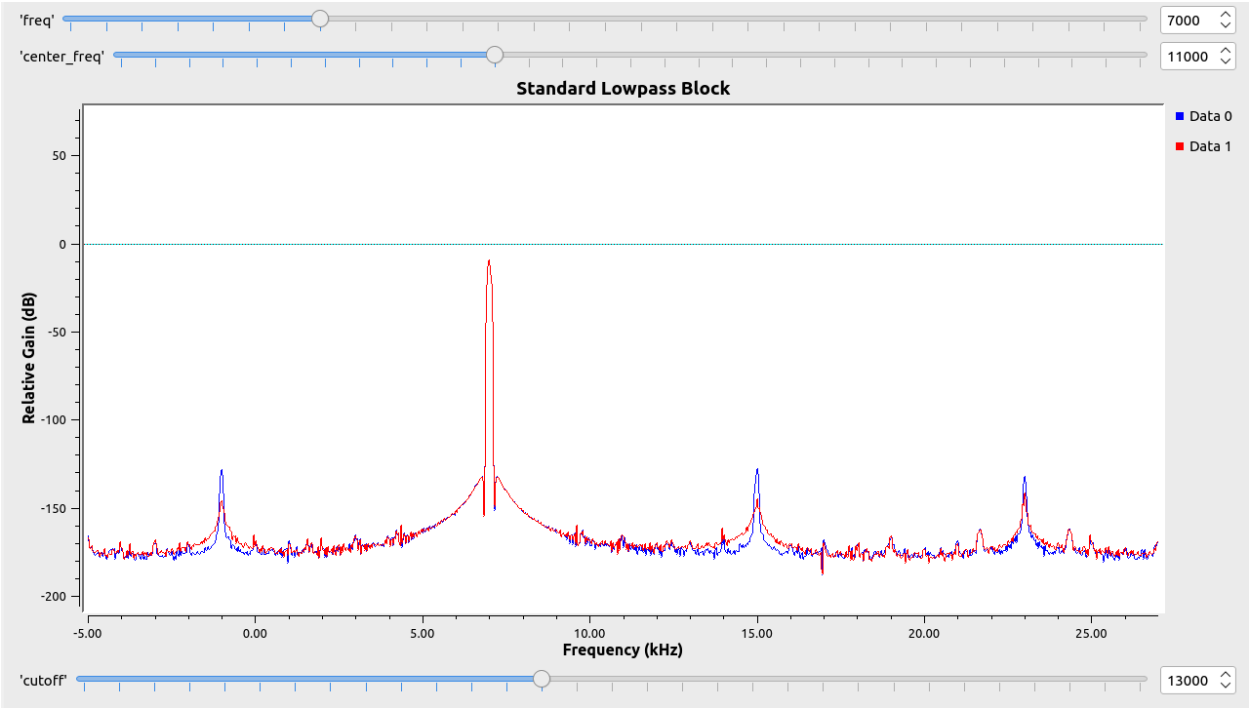


Figure 7: Center frequency is 7000, sliders included, data 0 blue represents the FFT and data 1 red represents the FIR filter.

Questions

1. If the FIR and FFT filter function almost identically, what is the point of having both? (Think about hardware/computing cost?)

FIR filters have linear phase and use less computing compared to FFT filters.

2. Is it easier to use the general block and modify the taps parameters or use the wrapper blocks (the first filters we used)?

It is easier to use general blocks and modify the taps to because the variables can be used and the blocks can easily be reintegrated.

Writing your own block

```
"""
Embedded Python Blocks:

Each time this file is saved, GRC will instantiate the first class it finds
to get ports and parameters of your block. The arguments to __init__  will
be the parameters. All of them are required to have default values!
"""

import numpy as np
from gnuradio import gr


class blk(gr.sync_block):  # other base classes are basic_block, decim_block, interp_block
    """Embedded Python Block example - a simple multiply const"""

    def __init__(self, Port_sel=0):  # only default arguments here
        """arguments to this function show up as parameters in GRC"""
        gr.sync_block.__init__(
            self,
            name='Custom Mux',   # will show up in GRC
            in_sig=[np.float32, np.float32, np.float32],
            out_sig=[np.float32]
        )
        # if an attribute with the same name as a parameter is found,
        # a callback is registered (properties work, too).
        self.Port_sel = Port_sel

    def work(self, input_items, output_items):
        """example: multiply with constant"""
        output_items[0][:] = input_items[self.Port_sel]
        return len(output_items[0])
```

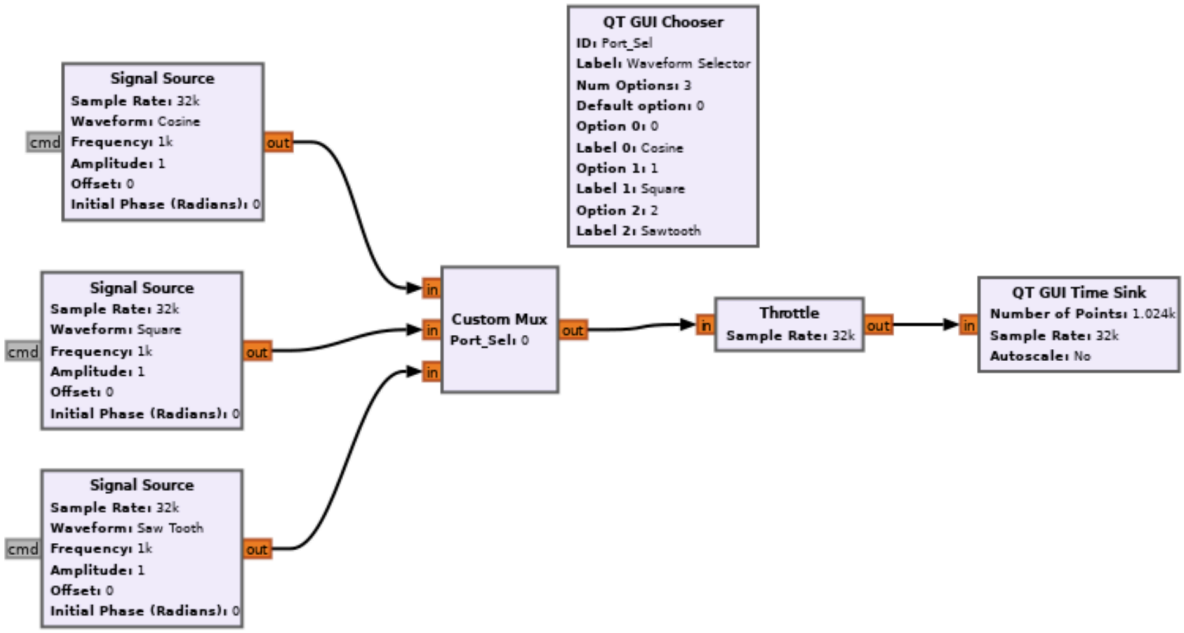


Figure 8: Custom mux flowchart

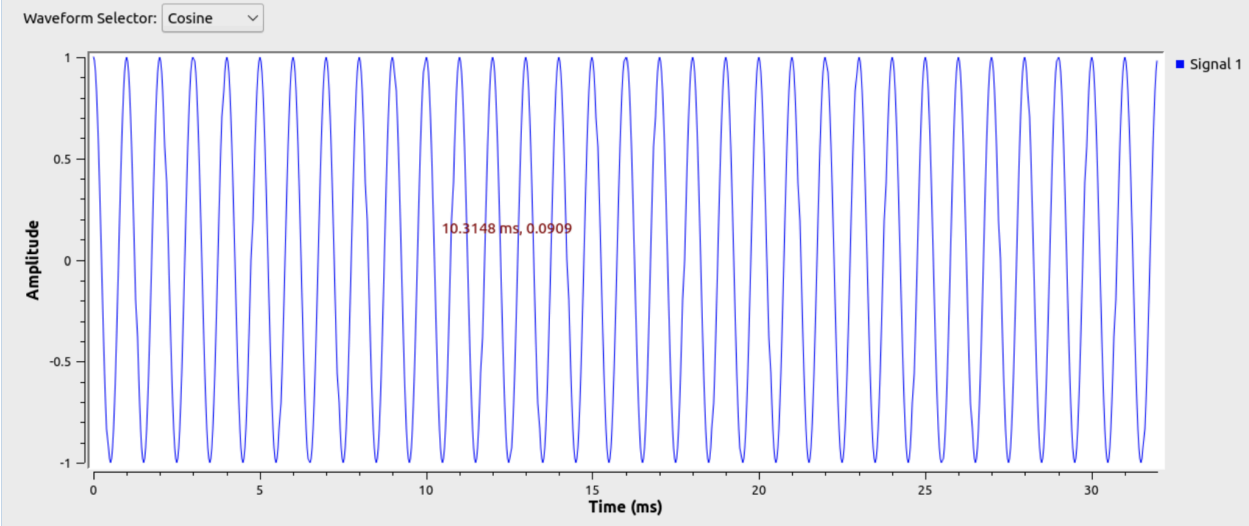


Figure 9: Cosine wave

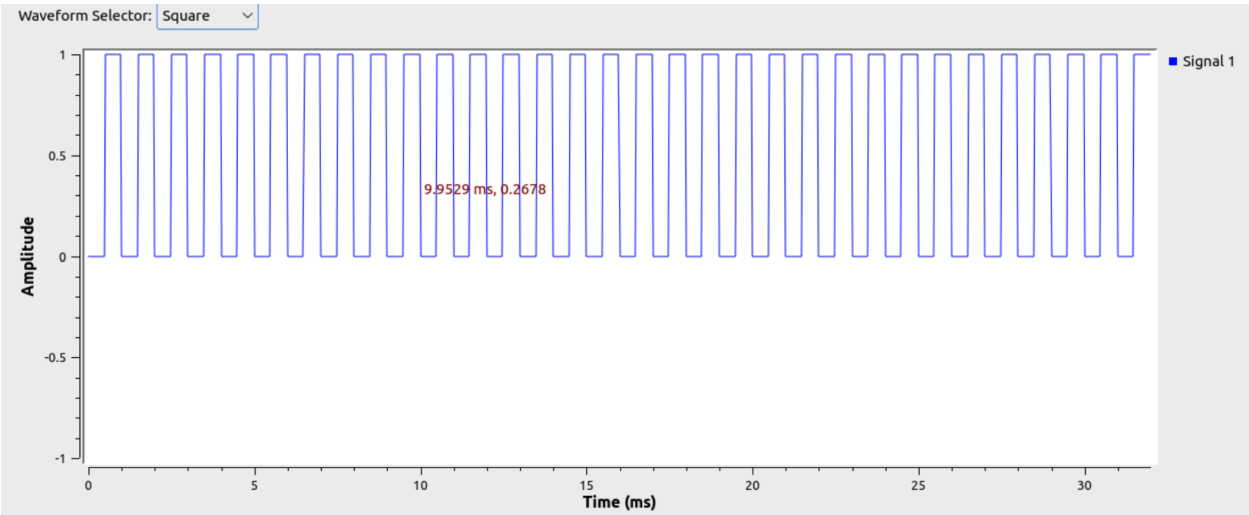


Figure 9: Square wave

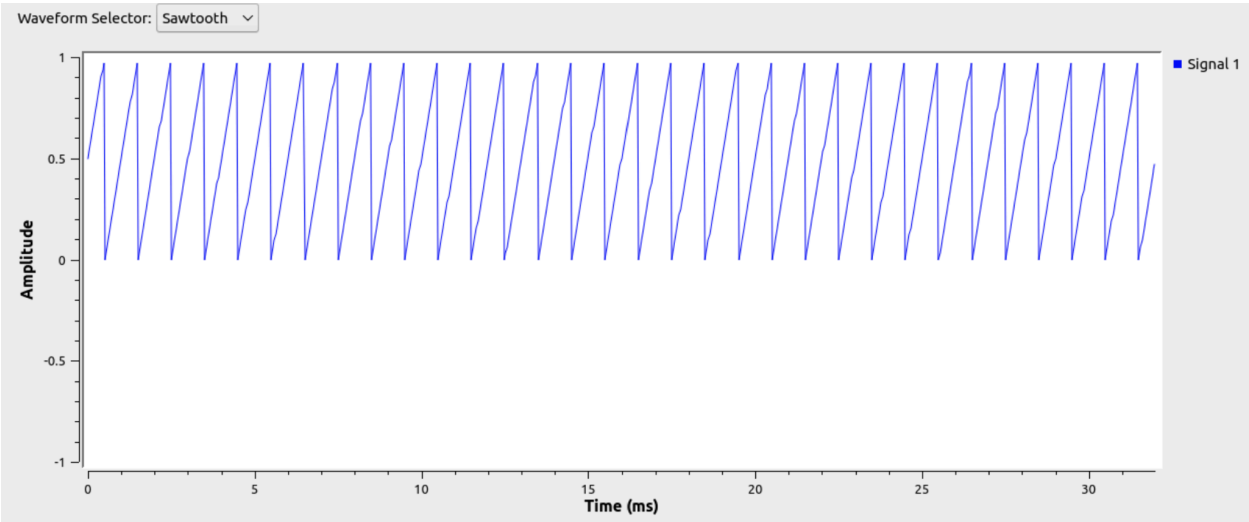
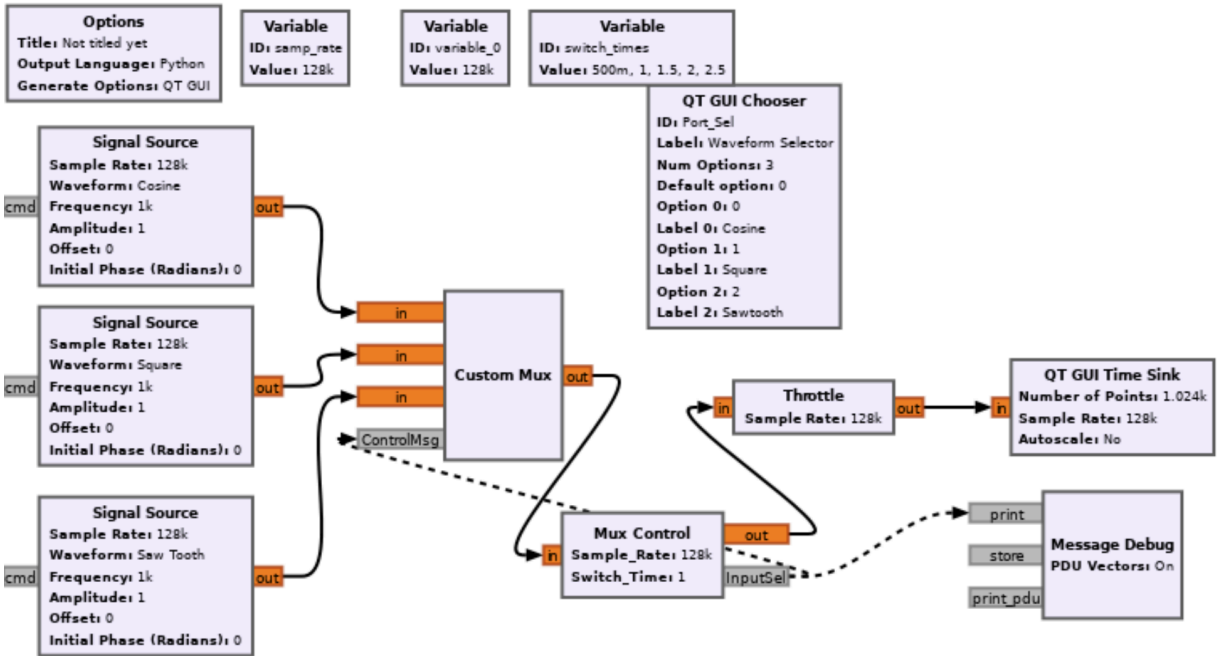


Figure 10: Sawtooth wave

Automating the waveform switching process



MUX Control Code

```
"""
Embedded Python Blocks:
Each time this file is saved, GRC will instantiate the first class it finds
to get ports and parameters of your block. The arguments to __init__  will
be the parameters. All of them are required to have default values!
"""
import numpy as np
from gnuradio import gr
import pmt

class control_block(gr.sync_block):
      def __init__(self, sample_rate=1.0, switch_time=2.0):
      gr.sync_block.__init__(
            self,
            name='Mux Control',
            in_sig=[np.float32],
            out_sig=[np.float32]
      )
      self.sample_rate = sample_rate
      self.switch_time = switch_time
      self.numSamples = int(self.sample_rate * self.switch_time)
      self.counter = 0
```

```
        self.selected_port = 0
        self.portName = "InputSel"
        self.message_port_register_out(pmt.intern(self.portName))

        def work(self, input_items, output_items):
        nitems = len(input_items[0])
        self.counter += nitems
        if self.counter >= self.numSamples:
                self.selected_port = (self.selected_port + 1) % 3
                self.counter = 0
                msg = pmt.from_long(self.selected_port)
                self.message_port_pub(pmt.intern(self.portName), msg)

        output_items[0][:] = input_items[0]
        return len(output_items[0])
```

Custom Mux Code

```
"""
Embedded Python Blocks:

Each time this file is saved, GRC will instantiate the first class it finds
to get ports and parameters of your block. The arguments to __init__  will
be the parameters. All of them are required to have default values!
"""

import numpy as np
from gnuradio import gr
import pmt

class blk(gr.sync_block):  # other base classes are basic_block, decim_block,
interp_block

        def __init__(self):  # only default arguments here
        """arguments to this function show up as parameters in GRC"""
        gr.sync_block.__init__(
                self,
                name='Custom Mux',   # will show up in GRC
                in_sig=[np.float32, np.float32, np.float32],
                out_sig=[np.float32]
        )
```

```
self.Port_sel = 0
self.portName = "ControlMsg"
self.message_port_register_in(pmt.intern(self.portName))
self.set_msg_handler(pmt.intern(self.portName), self.handle_msg)

def handle_msg(self, msg):

self.Port_sel = pmt.to_long(msg)

def work(self, input_items, output_items):

output_items[0][:] = input_items[self.Port_sel]
return len(output_items[0])
```

Questions

1. What are the benefits and disadvantages of asynchronous information in flowgraphs?

The benefits are fewer interruptions and setup. More flexible handling of data and timing. Async requires less CPU and memory usage as blocks can process data at a set pace. The disadvantages are that it adds debugging issues as the system can be more complex. Data may not be processed immediately which increases latency.

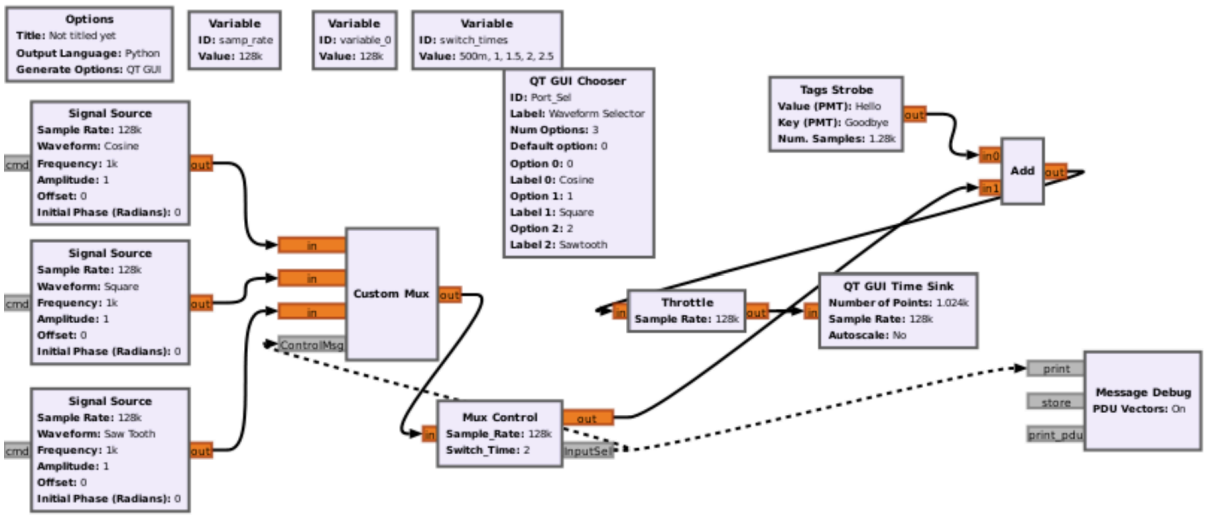
2. What other advantages might the message debugger have if you spend the time to write debug messages?

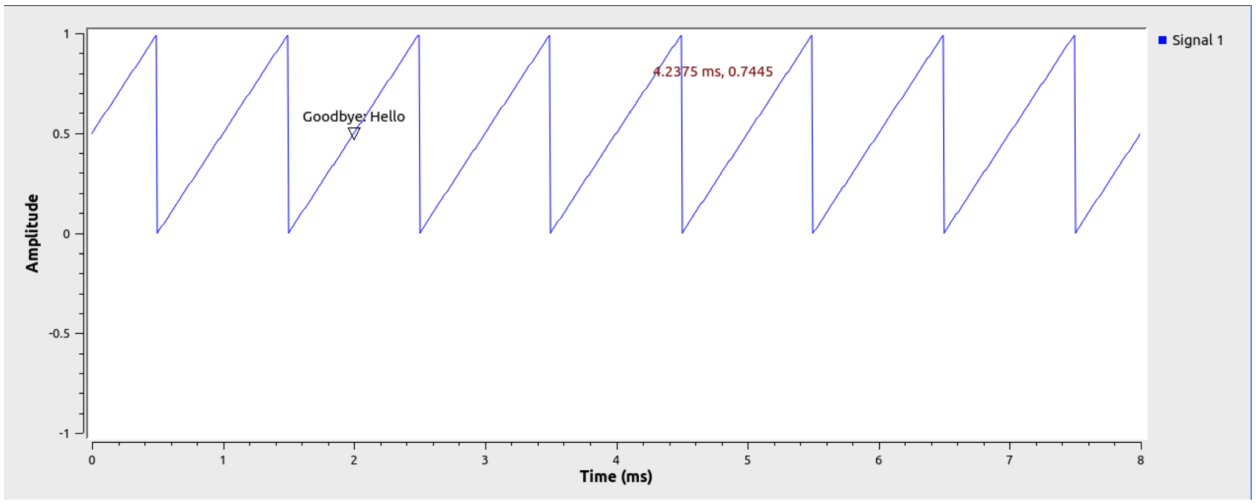
The advantages may be to identify where the problem occurs by programming the blocks to write debug messages.

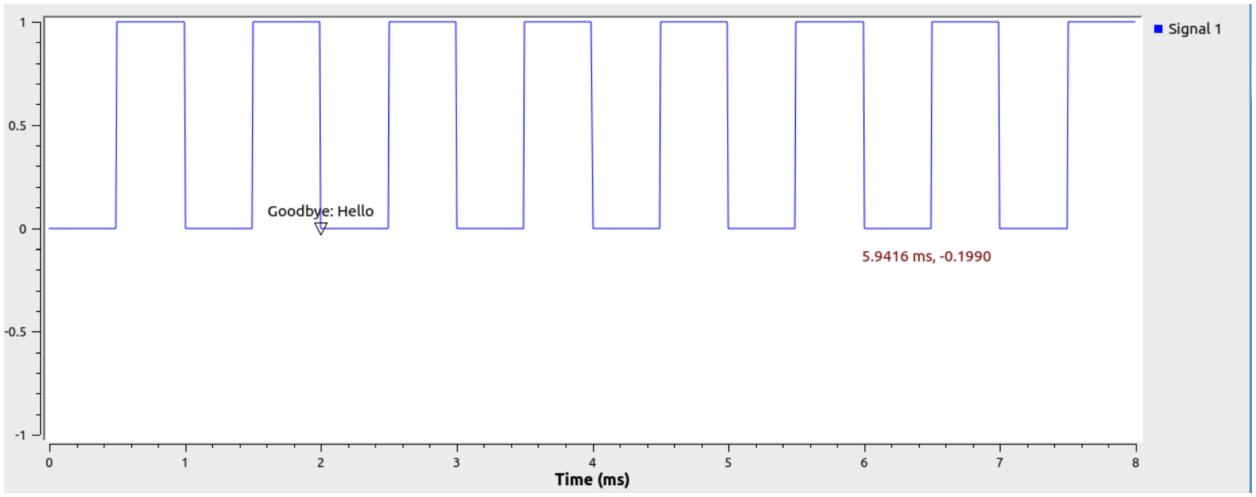
Demo Video

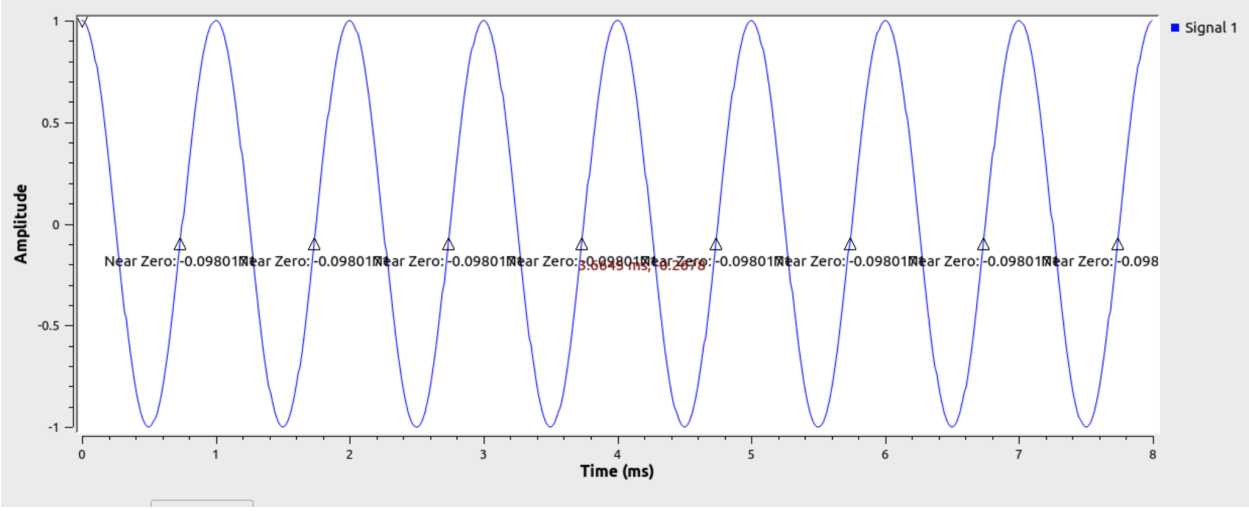
https://youtu.be/gH0glrHc5vo

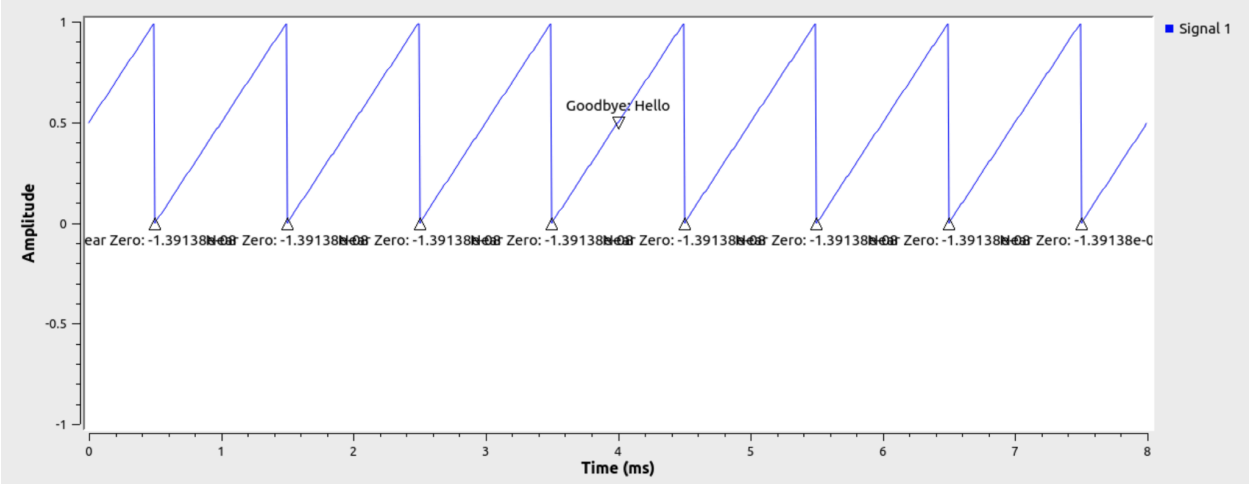
Streams and Tagging Streams


Options
Title: Not titled yet
Output Language: Python
Generate Options: QT GUI
Variable
ID: samp_rate
Value: 128k
Variable
ID: variable_0
Value: 128k
Variable
ID: switch_times
Value: 500m, 1, 1.5, 2, 2.5
QT GUI Chooser
ID: Port_Sel
Label: Waveform Selector
Num Options: 3
Default option: 0
Option 0: 0
Label 0: Cosine
Option 1: 1
Label 1: Square
Option 2: 2
Label 2: Sawtooth
Signal Source
Sample Rate: 128k
Waveform: Cosine
Frequency: 1k
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0
Signal Source
Sample Rate: 128k
Waveform: Square
Frequency: 1k
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0
Signal Source
Sample Rate: 128k
Waveform: Saw Tooth
Frequency: 1k
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0
Custom Mux
ControlMsg
Mux Control
Sample_Rate: 128k
Switch_Time: 2
inputSel
Tags Strobe
Value (PMT): Hello
Key (PMT): Goodbye
Num. Samples: 1.28k
Add
Throttle
Sample Rate: 128k
QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 128k
Autoscale: No
Message Debug
PDU Vectors: On
print
store
print_pdu


Signal 1
4.2375 ms, 0.7445
Goodbye: Hello
Amplitude
Time (ms)


Signal 1
Goodbye: Hello
5.9416 ms, -0.1990
Time (ms)





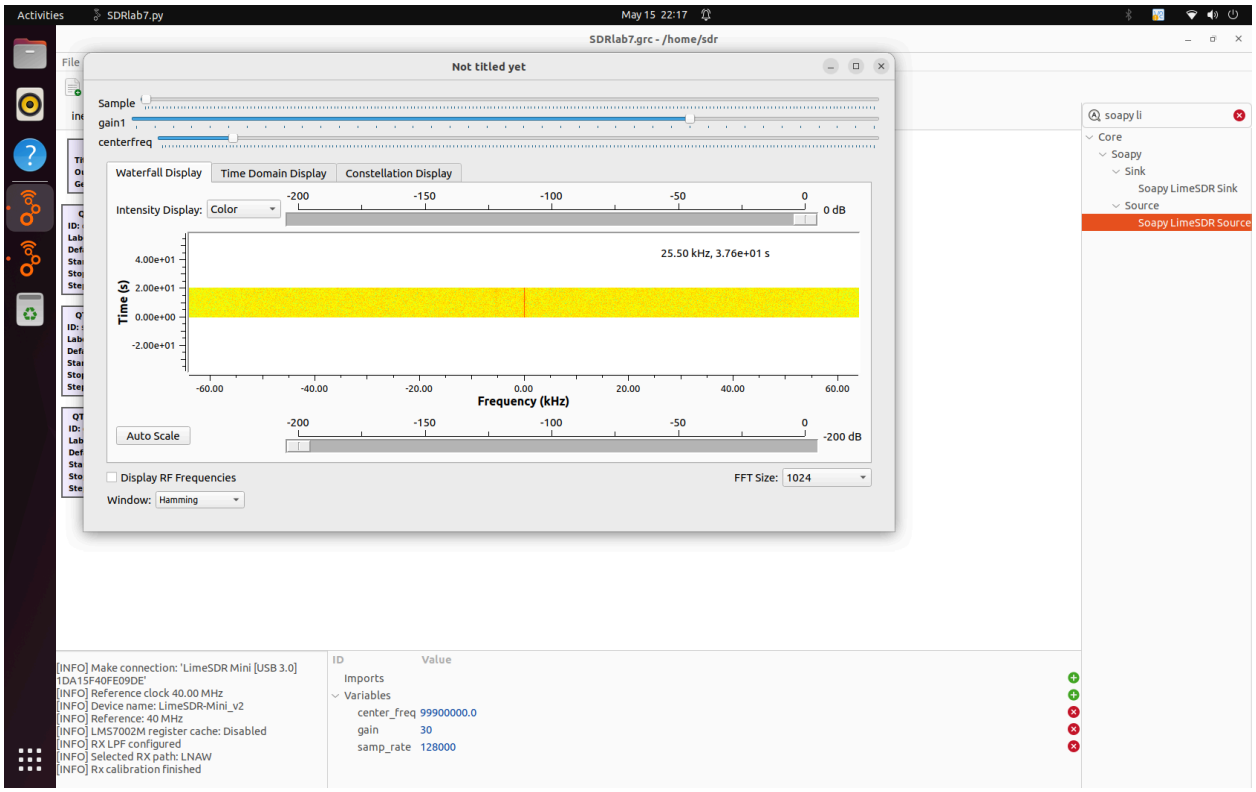
Questions

1. What is the main difference between tags and messages? What are the use cases for both?

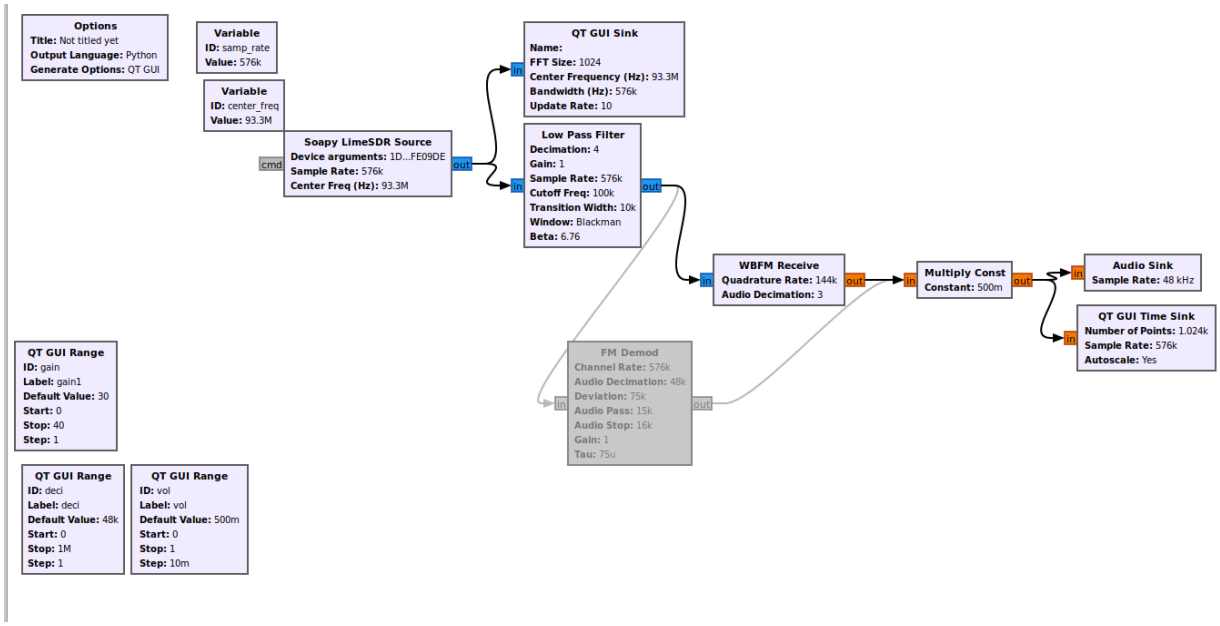
Tags annotate spaces in data by providing metadata at a range or sample. Messages pass information that does not fit into data streams. They are asynchronous unlike tags. Control commands, notifications of events are the best use for messages.

2. Why are tags particularly useful for formatting PDUs/packets in flowgraphs?
Tags are useful because they have metadata, synchronize data streams marking the beginning and end of PDUs, and error handling to find bad or corrupted parts of the data.

Hardware and Real Systems

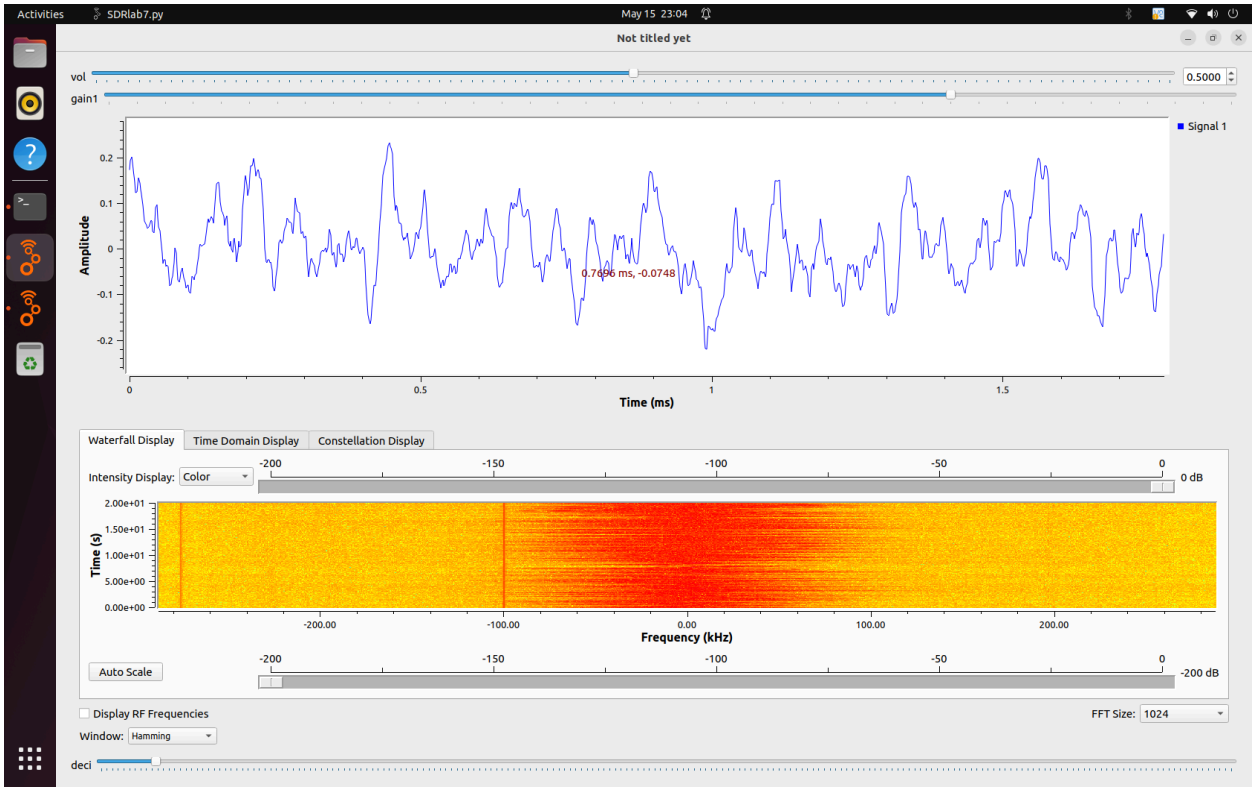


Simulation showing noise reception



Flowchart for FM receiver/Demodulator

Question: What does the center frequency of the LineSDR block do: The center frequency changes the center of the frequency you are visually analyzing on the graph. The complex waveform output by the SDR is centered at the frequency of the baseband signal.



FM frequency 99.3MHz

Demo Video

https://youtu.be/byIsJyfengk

Conclusion

It is important to understand how to interface the LimeSDR with the Raspberry PI using GNU radio. Analyzing the noise of the system is important to see if the antennas are interfaced properly. Building out the software to receive and transmit signals at a certain frequency, and decimating are important factors to understand. Tags annotate spaces in data by providing metadata at a range or sample. Messages pass information that does not fit into data streams. They are asynchronous unlike tags. Control commands, notifications of events are the best use for messages. The FM receiver and demodulator function properly after finding the proper sample rate and center frequency.