Lab 2 - Getting Started on PlutoSDR

ECE531 – Software Defined Radio

Spring 2022

Rahel Gerson, rmizrahi@email.arizona.edu

Due Date: February 25, 2021

Table of Contents

List of Figures

List of Tables

1 Radio Setup and Environmental Noise Observations

1.1 Signal Loopback

In the signal loop back experiment, we transmit the same signal that we receive by connecting the TX and RX with a coaxial SMA cable.

1.1.1 MATLAB Loopback

We first perform signal loopback via matlab. A sinusoidal tone is output by the transmitter and is simultaneously received at the receiver. To perform this, the provided loopback.m script was used, but the parameters were changed so that they matched the paramters of the GNU flowgraph below.
This script collects multiple buffers of data, which can be necessary since there is an unknown startup lag for the transmitter and the desired signal can be missed.
The script generated the below signals. Notice the choppiness. This choppiness arises because the parameter *samples per frame* is not a multiple of $\frac{T_x}{T_s}$, where T_x is the period of the signal x(t) and T_s is the sampling interval. When samples per frame is not a multiple of this ratio, the first sample is sampled at a different location than the ending sample, which gives rise to the choppiness observed in the dark graphs below. The light graph shows the output at the receiver when samples per frame is $3 \cdot \frac{T_x}{T_s}$.

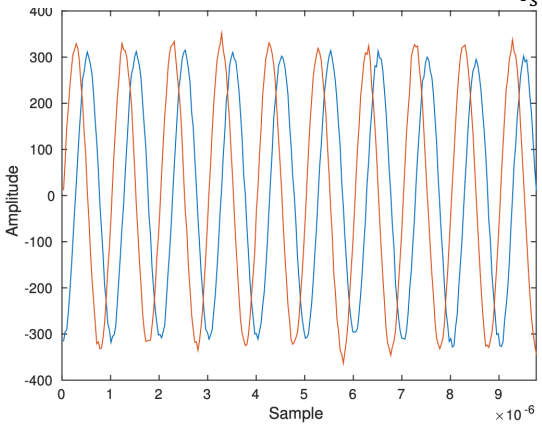

Figure 1. Looped back sinusoid observed from the Pluto receiver when SamplesPerFrame = $3\frac{Tx}{Ts}$

Gain = 30 dB, varied amplitudes.

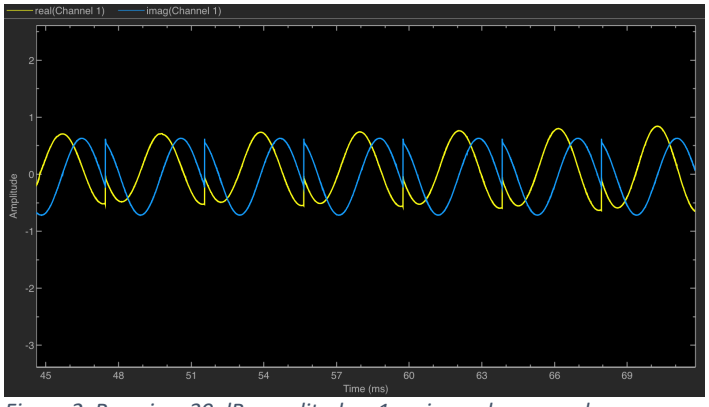

Figure 2. Rx gain = 30 dB, amplitude = 1, gain mode: manual

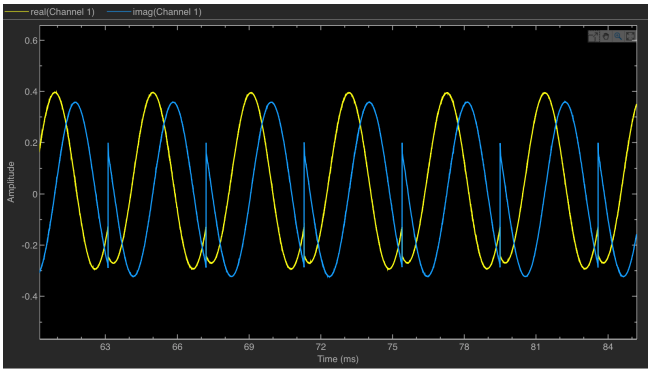

Figure 3. Rx gain = 30dB, amp = 0.5

ACG Gain Modes when amplitude = 1

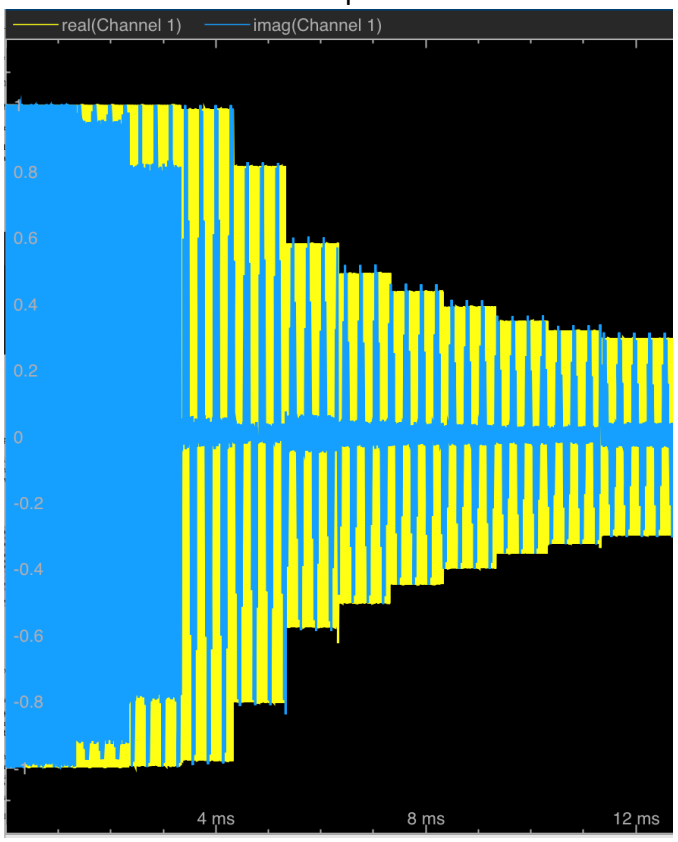

Figure 4, amplitude = 1, ACG fast attack

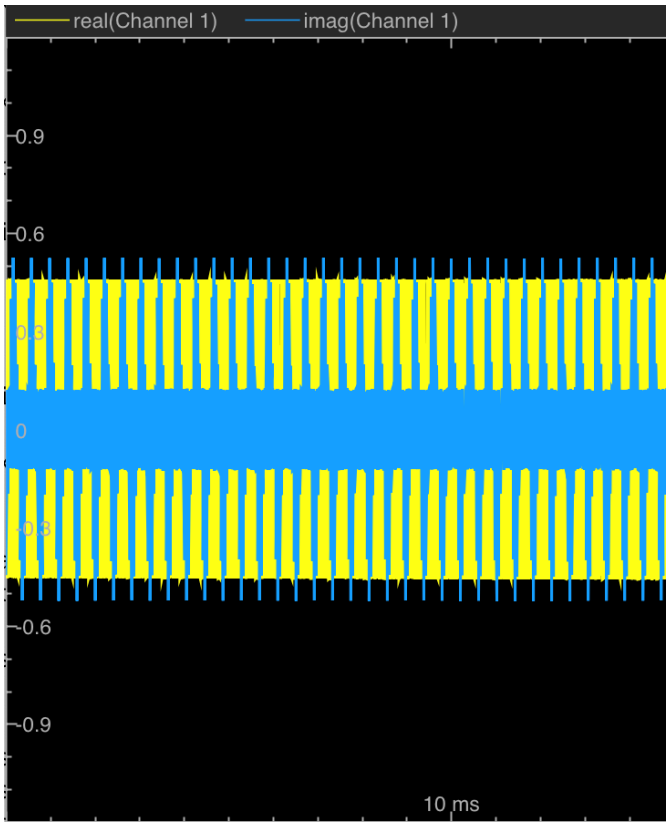

Figure 5. tx gain = -30 dB, amplitude = 1, ACG fast attack

Starting from an Rx gain of 20 dB, the sinusoid appeared as a square wave from the point of view of the receiver, as shown below.

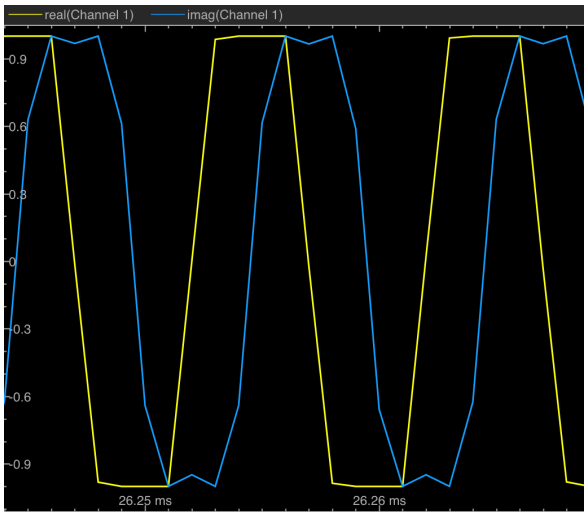

Figure 6. gain = 20 dB, amplitude = 1, gain mode = manual.

1.1.2 GNU Radio Loopback

Now that MATLAB has been used for a loopback test, we attempt the same test with GNU Radio to verify both tools are functioning properly.
To perform this test, we utilized the provided PlutoTestSine.grc flowgraph in GNU Radio Companion. This flowgraph is configured to simultaneously transmit and receive in the 2.4GHz band with a sample rate of 2.084MHz. The “QT GUI Range” variable source frequency and RX gain at run-time.

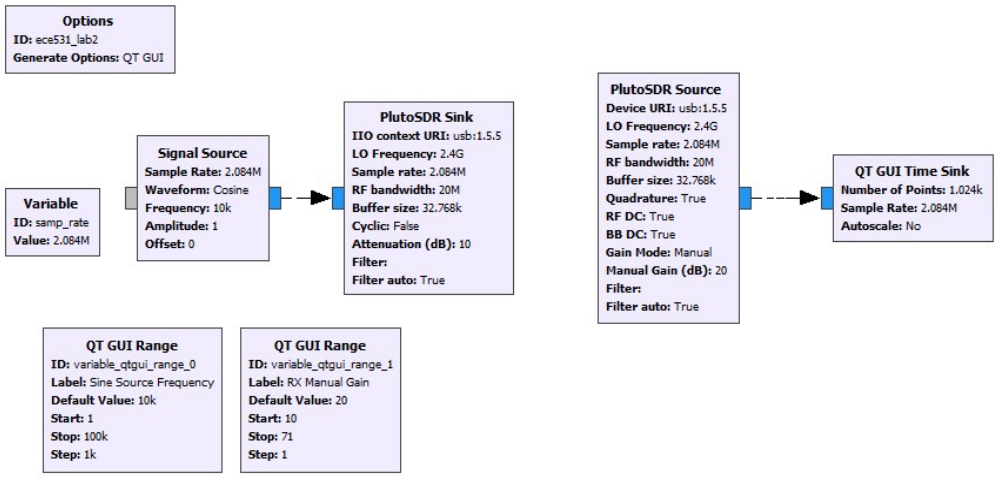

Figure 7: GRC flowgraph for Lab 2, Section 1.1.2

Note: Pluto SDR source is sourcing the data from the Pluto receiver; Pluto SDR sink is feeding the signal in to the Pluto transmitter sink in the GRC = source in the Pluto

In the flowgraph above, the Signal Source is transmitting a cosine wave to the Pluto Tx port, which is represented by the PlutoSDR sink. The Pluto Rx port is providing (i.e sourcing) samples to the QT GUI time sink, represented by the PlutoSDR source feeding data to the QT GUI time sink. Some parameters are described below:

(a) The RF bandwidth refers to the bandwidth of the configurable bandpass filter in Pluto's front end. This is not the same bandwidth as the sink's block. The RF bandwidth was 20 MHz.

(b) The "Cyclic" boolean selection for in the PlutoSDR Sink block? If set to true, the first buffer of samples will be repeated until the program is stopped. The PlutoSDR IIO block will report its processing as complete: the blocks connected to the PlutoSDR IIO block won't execute anymore, but the rest of the flow graph will. This is useful for

2. What does the Manual Gain control in the PlutoSDR source? What other strategies are available? the Pluto can be configured to have a fixed receive gain or a dynamic receive gain, called Automatic Gain Control (AGC). AGC is a closed-loop feedback circuit that controls the amplifier's gain in response to the received signal. Its goal is to maintain a constant output power level despite a varying input power level.[1] In contrast, when in manual gain control mode, the ACG circuit is turned off and the user specifies what receive gain to use.[2]

3. The following data was observed when we ran the flowgraph.

[1] https://pysdr.org/content/pluto_advanced.html
[2] https://pysdr.org/content/pluto_advanced.html

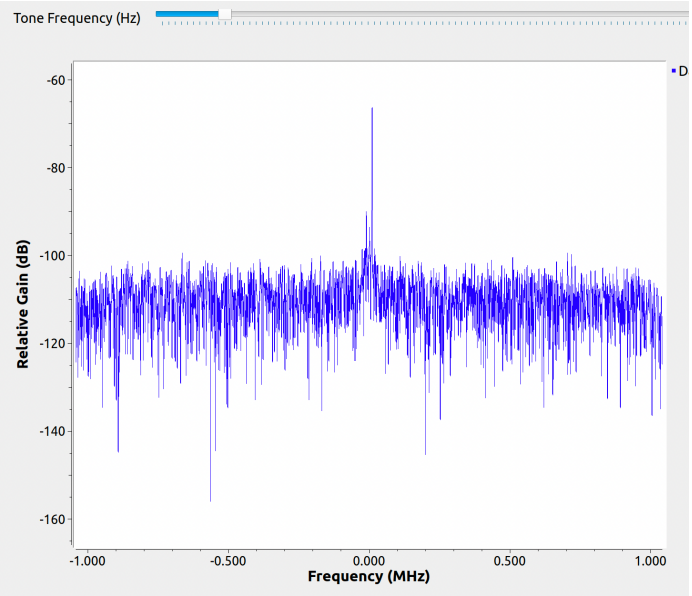

Figure 8. Rx gain of 20 dB

4. The RX gain was then adjusted to determine the db value where the signal begins to distort or clip. This occurs starting at a gain of 15 dB. This is unexpectedly different than the gain value in the previous section, despite identical parameters. Possible causes for this are differences in the way matlab and GNUradio interface and process the samples.
5. Replace the "QT Time Sink" block with a "QT GUI Sink". Explore the options provided by the new sink block. What happens if you increase the number of averages? When the number of averages is zero, some spurs' heights are very large, and on average the heights of the spurs are random. As the number of averages increases, the height of the spurs decreases as frequency increases.
6. Why does the frequency response change when you change the window function? The purpose of window functions is correct the spectral folding that results when you take the FFT of a non periodic signal. There are different windowing functions, such as Hamming, Hanning, Blackman, and Kaiser. What All these functions involve a tradeoff between the narrowness of the main lobe and steepness of the side lobes. For example, Balckman has low side lobes, but its main lobe is wider. When we use different window functions, the input signal changes because it's being convolved with a different window function.
7. What is the transmitted RF frequency of the sinusoid? Why? The transmitted RF frequency of the sinusoid is 2.4 GHz + 10kHz? Mixing the signal with the LO frequency of 2.4 GHz shifts the signal's center from 0 GHz to 2.4GHz, so its new frequency is 2.4GHz + 10kHz.

1.1.3 GNU Radio as a libIIO client

The PlutoSDR source and sink blocks contained in the Industrial IO module provide a convenient way to interface with PlutoSDR hardware. Generic blocks are also available and can be useful when interfacing with generic SDR hardware compatible with IIO. The specific values necessary for the IIO Device Source/Sink and IIO Atribute blocks were found previously using IIO command line tools, such as iio_attr. The default values for these blocks are shown in Figure 9. A quick reference on using these tools will be posted to D2L.

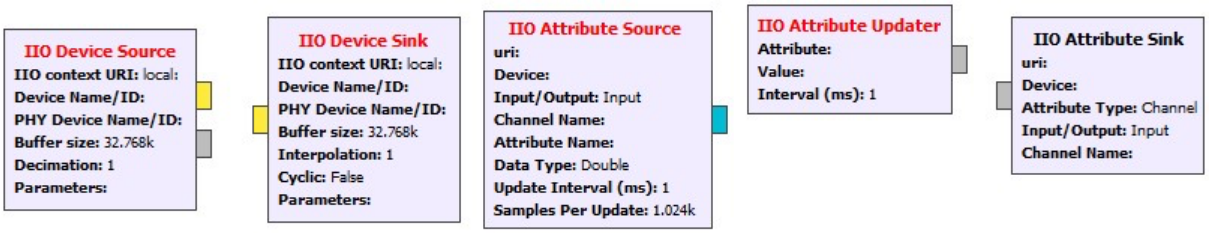

Figure 9: Default values for IIO Device blocks to be used for Section 4.1.3.

In this section, the loopback experiment from Section 1.1.2 was repeated in GNU radio without using the PlutoSDR source and sink blocks. Only the "IIO Device Source / Sink" and "IIO Attribute Source / Sink / Updater" were used. For reasons still unbeknownst to us, our flow diagram is not functional even after following along the the professor's tutorial in lecture8. Thus, we tried to update the attributes through terminal. When we use iio_attr, we notice that the value is unchanged. But we "cat" out this value, the value appears to be set correctly. Below are the different device names and their descriptions.

	Name	Description
device0	adm1177	power monitor
device1	ad9361-phy	controls transceiver
device2	xadc	Zynq IP core
device3	cf-ad9361-dds-core-lp	DAC / TX output driver
device4	cf-ad9361-lpc	ADC / RX capture driver

1. We changed the TX and RX center frequency from 2.4GHz to 915MHz.

Before changing the frequency

```
# cat out_altvoltage0_RX_LO_frequency
2400000000
# cat out_altvoltage1_TX_LO_frequency
2449999998
```

using iio_attr, we notice that the value is still 2.5GHz.

```
# iio_attr -a -c ad9361-phy altvoltage1
Using auto-detected IIO context at URI "local:"
dev 'ad9361-phy', channel 'altvoltage1' (output), id 'TX_LO', attr 'external', value '0'
dev 'ad9361-phy', channel 'altvoltage1' (output), id 'TX_LO', attr 'fastlock_load', value '0'
dev 'ad9361-phy', channel 'altvoltage1' (output), id 'TX_LO', attr 'fastlock_recall', ERROR: Invalid argument (-22)
dev 'ad9361-phy', channel 'altvoltage1' (output), id 'TX_LO', attr 'fastlock_save', value '0 60,32,62,188,62,60,185
,124,124,62,60,188,60,60,60,62'
dev 'ad9361-phy', channel 'altvoltage1' (output), id 'TX_LO', attr 'fastlock_store', value '0'
dev 'ad9361-phy', channel 'altvoltage1' (output), id 'TX_LO', attr 'frequency', value '2449999998'
```

Using cat, the value is 9.15 MHz.

```
# cat out_altvoltage1_frequency
915000
# echo 915000 > out_altvoltage0_frequency
# cat out_altvoltage0_frequency
915000
```

2. We increased the sample rate from 2.084 MSPS to a greater value of 21 MSPS.

Sampling frequency is an attribute corresponding to the transceiver, so we look in ad9361-phy. We look for attribute name corresponding to sampling frequency.

```
# iio_attr -a -c ad9361-phy
Using auto-detected IIO context at URI "local:"
dev 'ad9361-phy', channel 'altvoltage1', id 'TX_LO' (output), found 8 channel-specific attributes
dev 'ad9361-phy', channel 'voltage0' (input), found 15 channel-specific attributes
dev 'ad9361-phy', channel 'voltage3' (output), found 8 channel-specific attributes
dev 'ad9361-phy', channel 'altvoltage0', id 'RX_LO' (output), found 8 channel-specific attributes
dev 'ad9361-phy', channel 'voltage2' (output), found 8 channel-specific attributes
dev 'ad9361-phy', channel 'temp0' (input), found 1 channel-specific attributes
dev 'ad9361-phy', channel 'voltage0' (output), found 10 channel-specific attributes
dev 'ad9361-phy', channel 'voltage2' (input), found 13 channel-specific attributes
dev 'ad9361-phy', channel 'out' (input), found 1 channel-specific attributes
```

Look in channel voltage0

```
# iio_attr -a -c ad9361-phy voltage0
Using auto-detected IIO context at URI "local:"
```

Use cat to read the sampling frequency

```
dev 'ad9361-phy', channel 'voltage0' (output), attr 'sampling_frequency', value '30719999'
dev 'ad9361-phy', channel 'voltage0' (output), attr 'sampling_frequency_available', value '[2083333 1 61440000]'
# echo 2100000 > out_voltage0_sampling_frequency
# cat out_voltage0_sampling_frequency
2100000
```

1.2 Measurements and the Radio

In this experiment, we show that we can get more accurate measurements if our signal amplitudes are as close as possible to full scale. This is because the receiver has inherent noise, so with a larger signal amplitude there will be more distance between the noise floor and our signal. For the 12-bit Pluto, this number is $2^{12} - 1 = 4095$.

To measure our signal quality, we used the signal to noise ratio (SNR).

$$SNR\ (dB) = 10\log_{10}\left(\frac{P_{SN} - P_N}{P_N}\right)$$

The power of a discrete periodic signal is the magnitude squared of the signal divided the period of the signal

$$P_g = \frac{1}{T}\sum_T |g(t)|^2, where\ |g(t)|^2 = g(t)g^*(t)$$

Since our signal consisted of 50% zeros and 50% signal, we calculated P_g by dividing over the number of samples. Because we have equal number of samples for both signal and noise, both methods of power calculations are equivalent.

To illustrate the principle that we can get more accurate measurements as our signal's amplitude approaches full scale, we transmitted a vector of zeros concatenated with a sine wave. When observing the signal from the perspective of the receiver, the zero samples were not zero. Rather, they had some amplitude to them (noise) which came from the receiver hardware itself.

To minimize the impact of hardware imperfections, we increased the amplitude of the signal. The results are shown below. We see that the SNR improves as we increase our signal amplitude:

Table 1. SNR for different signal amplitudes

signal amplitude	signal power (dBm)	noise power (dBm)	SNR
0.10	0.03	5.364670E-04	17.53
0.40	0.13	9.700000E-04	21.15
0.70	0.23	8.760000E-04	24.23
1.00	0.31	4.048829E-04	28.79

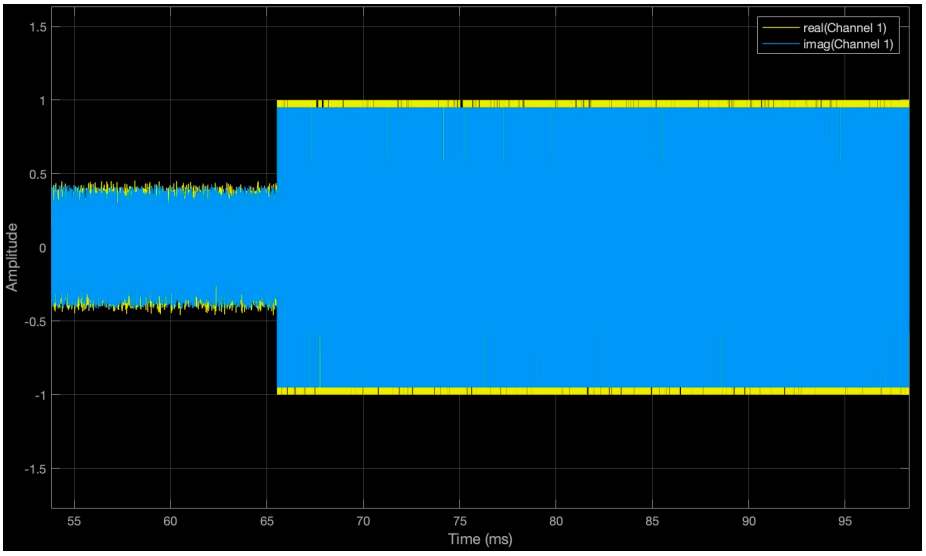

Figure 10. Received loopback signal with signal half nulled, amplitude = 1.

Next, we examine the effects of different Gain Source settings on the SNR, with the signal's amplitude fixed at 1. As we increase the manual gain from 10, the SNR improves up to a gain value of 30 dB. It then decreases drastically such that the signal power and noise power are nearly identical. When we use Automatic gain Control Fast attack, the signal looks almost identical as before, except for a large discontinuity. This discontinuity brought up the noise power so that it was almost as high the signal power. Next, we used Automatic Gain Control Slow attack. In this mode, the signal looked erratic and the SNR blew up to -30 dB. This makes sense because AGC slow attack doesn't do well with signals whose amplitude changes quickly, as ours does.

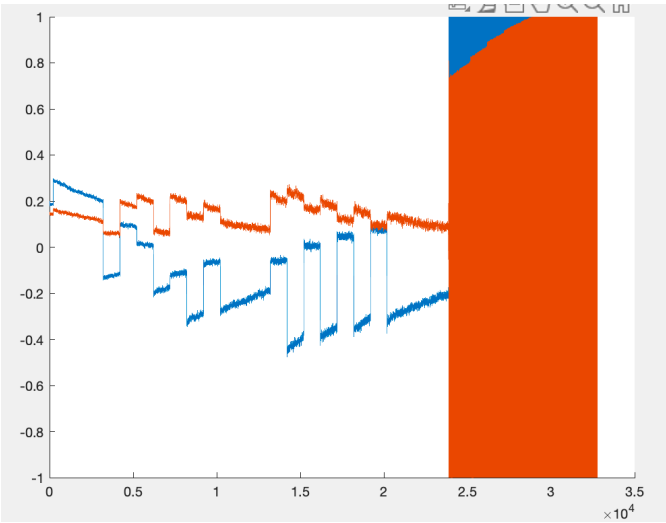

Figure 11. SignalSink buffer when AGC = Slow Attack

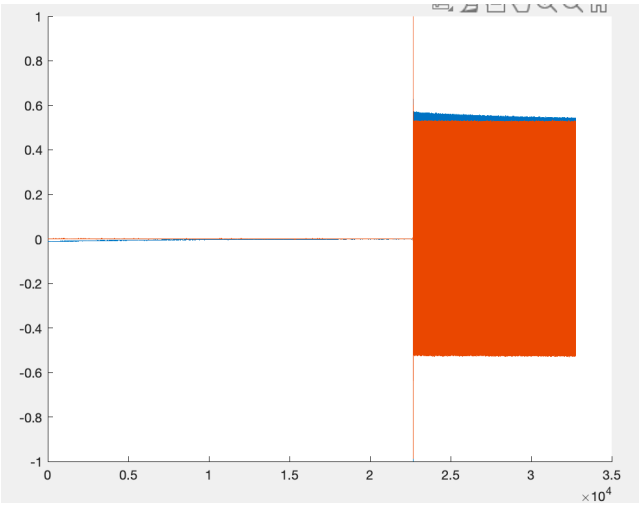

Figure 12. SignalSink buffer when ACG mode = Fast Attack

Table 2. Gain source settings versus SNR

Gain Source	signal power (dBm)	noise power (dBm)	SNR
10.00	0.04	2.940000E-04	21.41
20.00	0.12	4.070000E-04	24.67
30.00	0.31	5.880000E-04	27.15
40	0.001511	1.953000E-03	NaN
AGC Fast Attack	0.000465	4.800000E-04	Nan
AGC Slow Attack	0.186523	1.863480E-01	-30.27

3 Conclusions

We can adjust parameters in the *hardware* of the Pluto through iio and access them through the Linux file system with sysfs or MATLAB. We can use these different interfaces to verify that our settings are correct. The iio tree structure is helpful to keep in mind when updating these parametrs. In addition, we can interface with the Pluto by visualize what signals are present at its Tx and Rx terminals.