Lab 2 - Getting Started on PlutoSDR



**ECE531 – Software Defined Radio**

**Spring 2022**

Daniel Gallagher, danielgallagher@arizona.edu

Department of Electrical & Computer Engineering

University of Arizona, Tucson AZ 85721

*Due Date: February 21, 2021*

1. **Radio Setup and Environmental Noise Observations 10**
   1. Signal Loopback . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
   2. Measurements and the Radio . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
2. **Lab Report Preparation & Submission Instructions 15**

# Radio Setup and Environmental Noise Observations

In these laboratory experiments an Analog Devices ADALM-PLUTO SDR with AD9363 transceiver will be used. This SDR in the most basic sense is a System-on-Chip (SoC) with an attached RF module providing complex baseband data. The MATLAB interface utilizes the libiio kernel driver to talk with the SDR through a MATLAB class called iio\_sys\_obj\_matlab. The two system objects provided in the hardware support package for PlutoSDR are:

* comm.SDRRxPluto: PlutoSDR Receiver System object
* comm.SDRTxPluto: PlutoSDR Transmitter System object

These objects are typically constructed through the sdrrx or sdrtx function calls as in:

|  |
| --- |
| rx = sdrrx('Pluto') tx = sdrtx('Pluto') |

1

2

However, these objects can also be directly instantiated directly. The resulting object of sdrrx either way will have the following basic properties, which will be directly printed to the terminal when not using the semicolon as:

|  |
| --- |
| rx = sdrrx('Pluto') rx = comm.SDRRxPluto with properties:  DeviceName: 'Pluto'  RadioID: 'usb:0'  CenterFrequency: 2.4000e+09  GainSource: 'AGC Slow Attack'  ChannelMapping: 1  BasebandSampleRate: 1000000  OutputDataType: 'int16'  SamplesPerFrame: 3660  ShowAdvancedProperties: false |

1

2

3

4

5

6

7

8

9

10

11

12

The transmitter System object comm.SDRTxPluto has near identical properties except for

*GainSource*, *SamplesPerFrame*, and *OutputDataType* which do not make sense in the transmitter context. If you want to examine the available parameters simply type **doc comm.SDRRxPluto** or **doc comm.SDRTxPluto** into the MATLAB command prompt.

Before moving further with the radio it is important to outline how the radio works from the perspective of MATLAB using the sdrrx and sdrtx objects. After an object associated to the SDR has been created, the necessary methods can be run.

When configured in receive mode and the received method is called, with help from the driver, temporary buffers are created of size *SamplesPerFrame*, as set by the class parameter. Then the device will proceed to fill these buffer with contiguous data from the ADCs.

For each receive chain there are technically two ADCs, one for the in-phase portion of the signal and one for the quadrature. This is the reasoning behind the multiple buffers. However, these ADCs are time aligned and sampling of the dual chains happens at the same instances in time. Once the buffers are filled that data is provided to MATLAB and the link between the device and MATLAB is halted. As a result, when the receive method is called again from MATLAB the resulting buffer will be disjoint in time from the preceding buffer. For the overall structure of this communication Analog Devices provides a block diagram presented in Figure 6. The PlutoSDR class for convenience provides the output in a single complex vector, and accepts complex vectors as well.

Within the FMCOMMS/AD9361 direct downconversion is used to translate signals from RF to baseband frequencies. This is different than traditional heterodyne designs which utilize intermediate frequencies. Therefore, all observed signals passed to MATLAB are at baseband which were originally centered around *CenterFrequency* with a bandwidth of *BasebandSampleRate* parameterized by the sdrrx object.

## Signal Loopback

Now that we have a basic understanding of how the radio operates and have it setup we can perform some simple experiments. For the loopback experiment, you should use the coaxial SMA cable provided with the PlutoSDR; not the provided antennas.

### MATLAB Loopback

First we will run a simple loopback test which transmits a sinusoidal tone out the transmitter and is simultaneously received at the receiver. To perform this we will utilize the provided loopback.m script. 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. This script should generate a plot similar to the one shown in Figure 7.

Sample

×

10

-6

8

2

1

0

9

7

6

5

4

3

Amplitude

-400

-300

-200

-100

0

100

200

300

400

Figure 1. : Looped back sinusoid observed from the Pluto receiver.

Gain = -30 dB, varied amplitudes.

|  |  |
| --- | --- |
| Figure 2. gain = 30 dB, amplitude = 1, gain mode: manual | Figure 3. rx gain = 30dB, amp = 0.5 |

ACG Gain Modes

|  |  |
| --- | --- |
| Figure 4. tx gain = -20 dB, amplitude = 1, ACG fast attack | A screenshot of a computer  Description automatically generated with medium confidence  Figure 5. tx gain = -30 dB, amplitude = 1, ACG fast attack |

1. Modify the amplitude of the sinusoid, along with the PLUTO’s *GainSource* and observe their effects.
2. The three GainSource (AGC) settings are: *Manual*, *AGC Slow Attack*, and *AGC Fast*

*Attack*.

1. When *Manual* is selected another option called *Gain* becomes available this parameter can be modified.
2. It maybe useful to use signals with varying amplitudes or zeros.
3. If enough gain is applied the sinusoid should appear as a square wave at the receiver.

Chart

Description automatically generated

Figure 6. gain = -30 dB, amplitude = 1, gain mode = manual, rx gain = 35 dB

What gain value does this occur? this begins to occur at an rx gain of 30 dB.

### GNU Radio Loopback

Now that MATLAB has been used for a loopback test, lets attempt the same test with GNU Radio to verify both tools are functioning properly.

To perform this test, we can utilize 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.

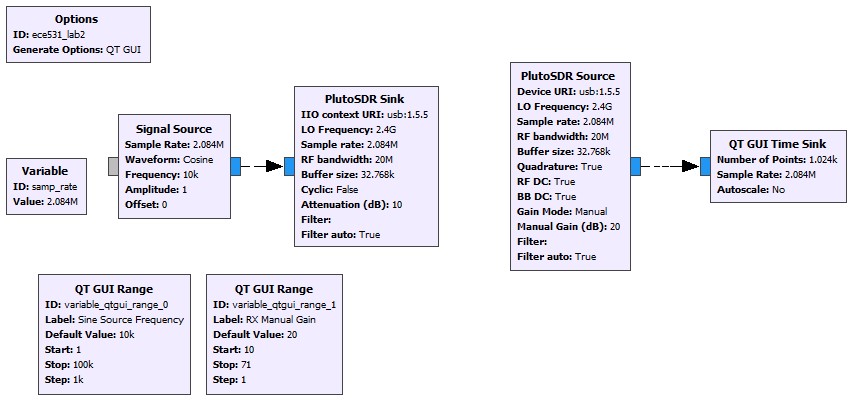


Figure 7: GRC flowgraph for Lab 2, Section 1.1.2

Double-click on each PlutoSDR block (source and sink) to look at the block properties.

* 1. What is the RF Bandwidth? What does this property control in the PlutoSDR? 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.
  2. What is 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.

1. 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]](#footnote-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]](#footnote-2)
2. Run the flowgraph. Note: you may have to enter change the Device URI. On some operating systems libIIO finds the first available device when this field is left blank.

A picture containing graphical user interface

Description automatically generated

Figure 8. Rx gain of 20 dB

1. Adjust the RX gain. At what value does the received signal begin to distort or clip? The signal begins to become more square starting at a gain of 15 dB.
2. Replace the “QT Time Sink” block with a “QT GUI Frequency Sink”. Explore the options provided by the new sink block. What happens if you increase the number of averages? With more averaging, we see more spurs. Why does the frequency response change when you change the window function? The input signal changes because it’s being convolved with different window functions.
3. What is the transmitted RF frequency of the sinusoid? Why? The transmitted RF frequency of the sinusoid is FIXME

### 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.

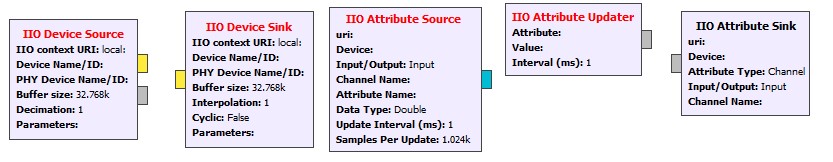


Figure 9: Default values for IIO Device blocks to be used for Section 4.1.3.

Figure 9Repeat the loopback experiment from Section 4.1.2 without using the PlutoSDR source and sink blocks; use only the “IIO Device Source / Sink” and “IIO Attribute Source / Sink / Updater”

1. Change the TX and RX center frequency from 2.4GHz to 915MHz.
2. Increase the sample rate from 2.084 MSPS to a greater value of your choosing.
3. Verify each of the above have been set in the hardware using IIO Scope plugins or iio\_attr.

## Measurements and the Radio

What a SDR device like the PLUTO does is give you digital samples. What these are is nothing more or less than what the ADC makes out of the voltages it observes. Then, those numbers are subject to the receiver processing chain which includes frequency translation (Mixing), decimation and filtering. Analog Devices provides a great overview here [7] for the *ad936x*. Altogether, the complex signal’s envelope coming from the PLUTO should be proportional to the voltages observed by the ADC. Therefore, the magnitude square of these samples should be proportional to the signal power as seen by the ADC. However, it must be stress that these are in relation to the range of the ADC. Therefore, these values are of an arbitrary measure relative to the full scale of the ADC and the remaining receive chain, commonly denoted as dBFS [8]. Scopes provided by MATLAB may denote the signal amplitude in dB but this is just an arbitrary engineering unit.

With this knowledge we cannot directly determine power of an input signal to the SDR, unless we have performed some calibration and can relate individual samples to received energy. However, SNR can still be determined since signal and noise we be subjected to the same ADC effects and receive chain. This can be done with techniques already presented previously. For Figure 10 we again used the transceiver setup, but transmitted signal vectors which contain half zeros and half signal.

Time

0

0.005

0.01

0.015

0.02

0.025

0.03

0.035

Amplitude

-60

-40

-20

0

20

40

60

Signal

Noise

+

Noi

se

Figure 10. Receive loopback signal with signal half nulled.

After extracting the noise and signal plus noise sections the SNR can be manually calculated simply by:

• Repeat the process for different GainSource settings and different source amplitudes by modifying the supplied loopback.m script. Comment on the changes in SNR.

This is a rough method for estimating SNR, but does provides a reasonable metric without additional equipment. To provide better results we would need to remove some of the adaption performed in the receiver conditioning the samples.

# Lab Report Preparation & Submission Instructions

Include all your answers, results, and source code in a laboratory report formatted as follows:

* Cover page: includes course number, laboratory title, name, submission date.
* Suggested: Table of contents, list of tables, list of figures.
* Commentary on designed implementations, responses to laboratory questions, captured outputs, and explanation of observations.
* Meaningful conclusions to the lab.
* Source code (as an appendix). You may also upload source files with report submission.

Remember to write your laboratory report in a descriptive approach, explaining your experience and observations in such a way that it provides the reader with some insight as to what you have accomplished. Furthermore, please include images and outputs wherever possible in your laboratory report document.

# References

1. https://pysdr.org/content/pluto\_advanced.html [↑](#footnote-ref-1)
2. https://pysdr.org/content/pluto\_advanced.html [↑](#footnote-ref-2)