EE2033 Integrated System Lab

MINI-PROJECT GUIDE

Equipment provided

Hardware

- ❑Adalm Pluto
- ❑RTL-SDR
- ❑Analog Discovery2

Software

- ❑Filterpro
- ❑LTspice
- ❑GNUradio

Components

- ❑Breadboard
- ❑Opamps
- ❑Resistors
- ❑Capacitors

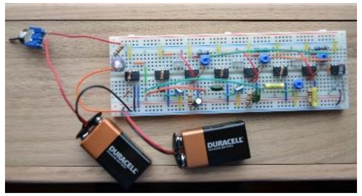
Mini-project files
(Access README_AD2 for more information)

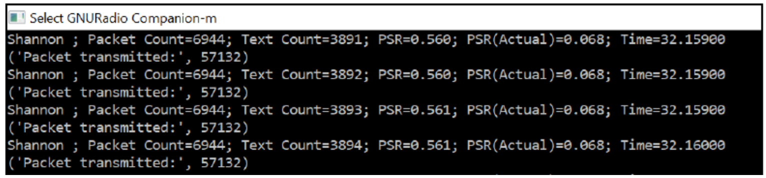
fm-radio-grc.grc:	GNURadio grc file to receive FM radio using RTL-SDR Source.
ook_pluto.grc:	GNURadio grc file to create OOK transceiver using Pluto Adalm Source and Sink.
ook_pluto_rtlsdr.grc:	GNURadio grc file to create OOK transceiver using PlutoAdalm Sink and RTL-SDR Source.
ook_pluto_tx.grc:	GNURadio grc file to create OOK transmitter using PlutoAdalm Sink.
ook_rtlsdr_rx.grc:	GNURadio grc file to create OOK receiver using RTL-SDR Source.
opamp_file.grc:	It will take "test.dat" and perform the OOK demodulation.
csv2grcf.py:	Python file to convert test.csv to test.dat, readable by GNUradio, opamp_file.grc.
test.csv:	File recorded from AD2 stored in csv format.
test.dat:	32-bit Float data binary file converted from test.csv use by opamp_file.grc and LTSPICE.
time.dat:	32-bit Float data binary file converted from test.csv to be used by LTSPICE.

LTspice files
(Access README for more information)

LPF.asc:	LTSPICE lowpass filter file that use sig1.dat to run the simulation.
opamp.lib:	Opamp library model that is used by LPF.asc
opamp_file.grc:	It will take "testspice.dat" and perform the OOK demodulation.
bin2spice.py:	Read in "test.dat" and "time.dat", and convert it into text file "sig1.dat" readable by LTSPICE.
wav2grcf.py:	Python file to convert "out.wav" to "testspice.dat", readable by GNURadio opamp_file.grc.
test.dat:	32-bit Float data (voltage) binary file converted from test.csv to be used by bin2spice.py.
time.dat:	32-bit Float data (time) binary file converted from test.csv to be used by bin2spice.py.
sig1.dat:	data file converted from "test.dat" and "time.dat", readable by LTSPICE as source.

General Flow of the Experiment for Task 1




Select GNURadio Companion-m
```
Shannon ; Packet Count=6944; Text Count=3891; PSR=0.560; PSR(Actual)=0.068; Time=32.15900
('Packet transmitted:', 57132)
Shannon ; Packet Count=6944; Text Count=3892; PSR=0.560; PSR(Actual)=0.068; Time=32.15900
('Packet transmitted:', 57132)
Shannon ; Packet Count=6944; Text Count=3893; PSR=0.561; PSR(Actual)=0.068; Time=32.15900
('Packet transmitted:', 57132)
Shannon ; Packet Count=6944; Text Count=3894; PSR=0.561; PSR(Actual)=0.068; Time=32.16000
('Packet transmitted:', 57132)
```

ook_pluto_tx → **ook_rtlsdr_rx** → **Filter design** → **opamp_file**

- Set the parameters based on your design
- Run the program to send the signal

- Set the parameters based on your design
- Run the program to receive the signal

Filter design
- Build your filter based on the filter conditions that you wish to obtain

opamp_file
- Use the csv2grcf.py file to convert the data collected to test.dat
- Run the opamp file to observe the demodulated signal
- Take note of the following parameters:
  - ➢ Packet Count (Total Packet detected)
  - ➢ Text Count (Total message detected)
  - ➢ PSR (Packet Success Rate)
  - ➢ Time (time taken to achieve the packet count)

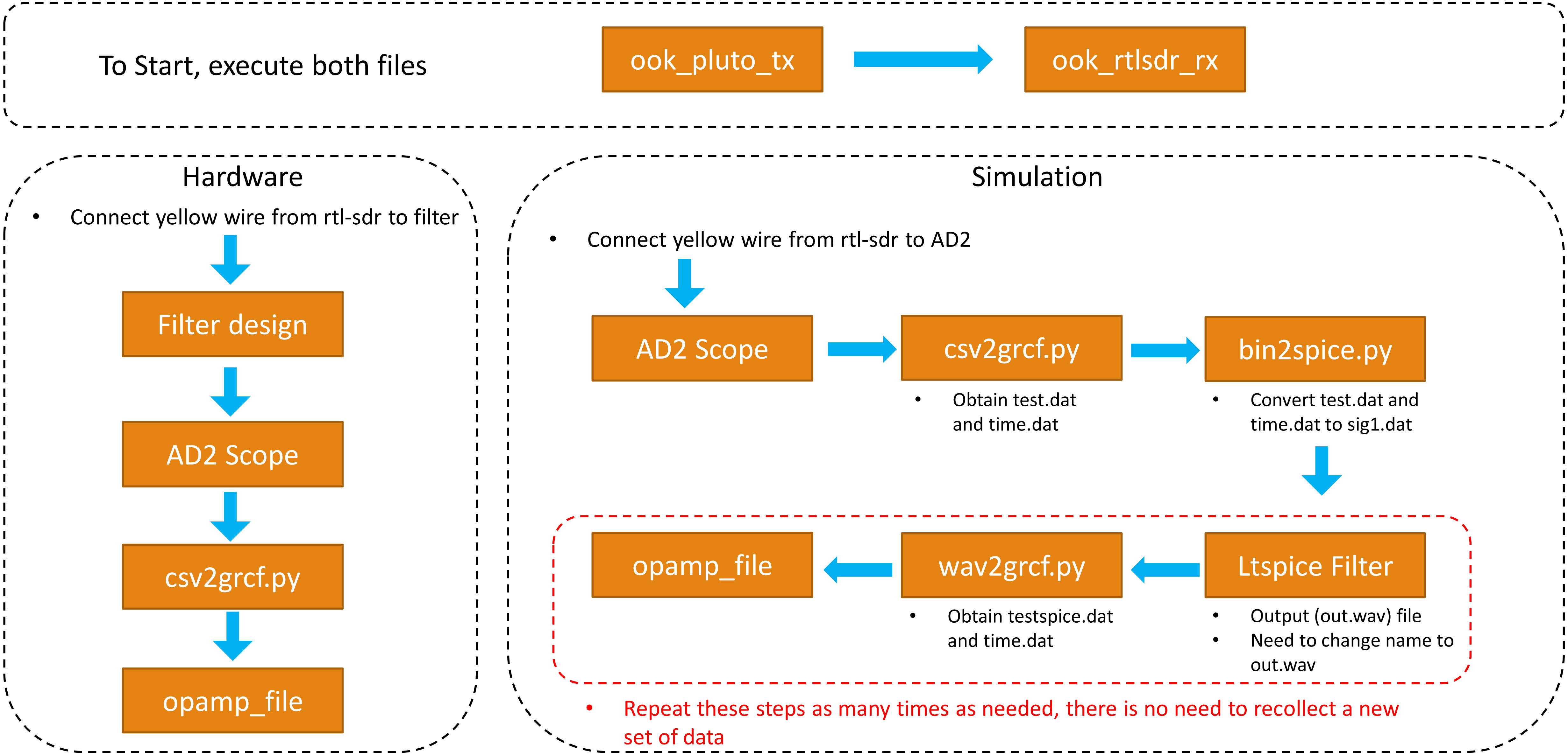
- Please ground your antenna for the RTL-SDR by adding a wire at (2) indicated in the image and ground it to your breadboard ground.
- To extract the data received from your RTL-SDR, use one of the yellow wires that are present indicated by (1). This wire will be used as the input to your filter circuit. You can also read the output directly by connecting to AD2
- Please go through the slides on RTL-SDR spectrum conversion for more information

AD2 usage
- Power supply to power your opamp for filter circuit
- Network function to characterize your filter
- Scope function to record the data that is being received
- Spectrum function to observe the spectrum plot

For AD2 data recording:

i. Select “Settings -> Device Manager -> Option 2 (2x16k)” to maximize the buffer size
ii. Change the “mode” to “Record”
iii. Click “Config”, choose “Samples” of “32768”, “Rate” of “4 MHz”, “Base” of “7.5ms”, and click “Start”
iv. You can repeatedly click “Record” button to capture the data.
v. At the side panel, click “Options”, uncheck “noise”, click “Zero offset”
vi. Once done, select ”File -> Export”. Under “Data” tab, select “Save” and store the data into “test.csv”.

Process Flow for experiment



Mission breakdown for Task 1

Mission 1: Understanding the overall project requirements

Mission 2: Understanding the overall project execution

Mission 3: Filter Design and verification

Mission 1: Understanding the overall project requirement

1. Setup the ADALM Pluto and RTL-SDR as follows

ook_pluto_tx

ook_rtlsdr_rx

- Open “ook_pluto_tx” in GNUradio
- Set sample rate, carrier frequency, symbol rate, signal amplitude, interference frequency
- Set interference amplitude to 0
- Run “ook_pluto_tx”

- Open “ook_rtlsdr_rx” in GNUradio
- Keep pluto and rtl-sdr 0.5m apart
- Run “ook_rtlsdr_rx”

2. Observe the results from terminal. You will see the Packet count, Text Count, PSR, PSR (Actual) and Time.

3. Observe PSR and time changes with increasing interference amplitude

Mission 2: understanding the overall project execution

1. Add on to the previous setup as shown in the whole signal chain below

ook_pluto_tx

Open and run ook_pluto_tx in GNUradio

Set interference amplitude to 0

ook_rtlsdr_rx

Open and run ook_rtlsdr_rx in GNUradio

AD2

Connect the yellow wire to the channel 1+ (Orange wire) of AD2.

Ground the antenna and channel 1- (Orange/white wire) to the ground of AD2

Recording

Record and save the data in CSV format that is being transmitted

csvtogrcf.py

Convert the CSV data to test.dat

opamp_file

Run the opamp_file using the new test.dat to observe your PSR and time

Mission 3: Filter Design and verification

1. Define your filter specifications based on the spectrum you have observed

2. Use Filterpro for your initial design based on the parameters that you have defined

3. Simulate the filter design in LTSpice using your choice of opamps, resistors and capacitors you have selected

4. Optimize the filter performance in LTSpice

5. Implement your filter design on a breadboard

6. Characterize the filter circuit on the breadboard to ensure that it matches the specifications that you require

7. Modify and optimize your circuit to meet the objective that you need

8. Confirm your design for Demo