Mini-Project 1: OOK Modulation and Demodulation in GNURadio

In this mini-project module, you will learn how to create your own GNURadio flowgraph to transmit and receive a simple modulation scheme: On-Off Keying (OOK). OOK communicates bits by turning a carrier wave on-and-off. If the signal is on, it's considered as bit 1, if it is zero, it's considered as bit 0.

At the end of this module, you will:

1. Understand how sample streaming structure works in GNURadio (i.e. the "flowgraph")
2. How to debug and visualize signals at different locations in the flowgraph
3. Make a simple OOK Modulator and Demodulator flowgraph
   a. Understand the effect of noise on OOK
   b. Understand the effect of frequency offset on OOK

Overall Description:
1. There are a total of six tasks (three for OOK Modulation and three for OOK Demodulation).
2. Some tasks require you to take a screenshot and/or write a very short answer to a question.
3. Submission format: upload the following as a .zip to Gradescope
   a. PDF containing screenshots and short answers
   b. Final GNURadio .grc file that you create.
4. Tips on working as a team:
   a. You can choose to divide the tasks among yourselves. Note that they have to be done in the order they are described.
   b. Include a note on each person's contribution on the pdf.
5. You do not need an SDR for this mini-project.

Walkthrough:

## Introduction to GNURadio Flowgraphs

1. **Goal: Visualize a signal using a time sink (oscilloscope).**
2. Open the GNURadio Companion (GRC) by typing **gnuradio-companion** on the VM terminal
3. GRC is a graphical tool that allows you to interact with GNURadio's python interface. The tool finally produces a python script as we will see later.

   a.
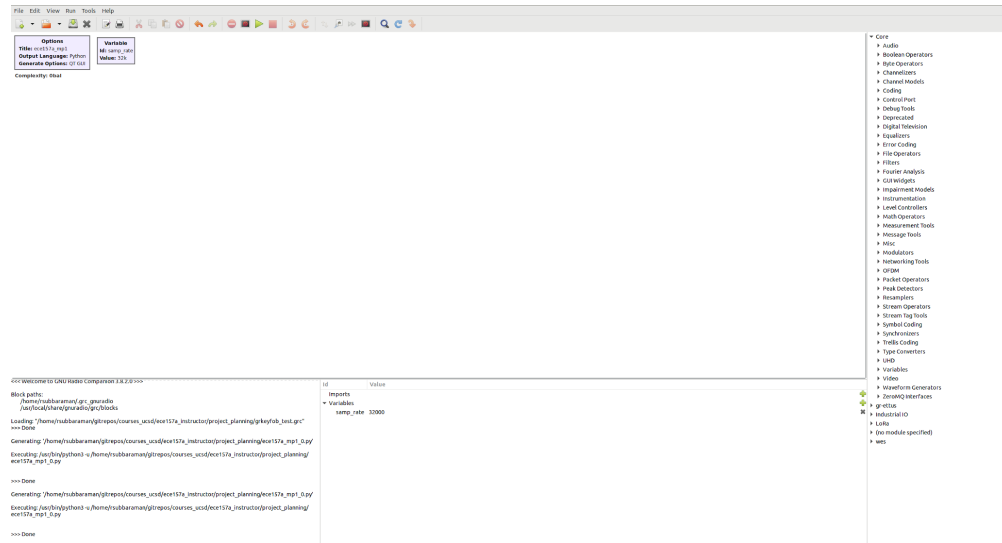
4. Create a new flowgraph, title and save it:
   a. Select File -> New -> QT GUI
   b. Double-click on the **Options** box, title your flowgraph as **ece157a_mp1**
   c. Save the grc at a suitable location.
5. Search for a block using the library pane on the right.

   a. Press the icon and search for the **Signal Source** block. Double-click to add it to the flowgraph
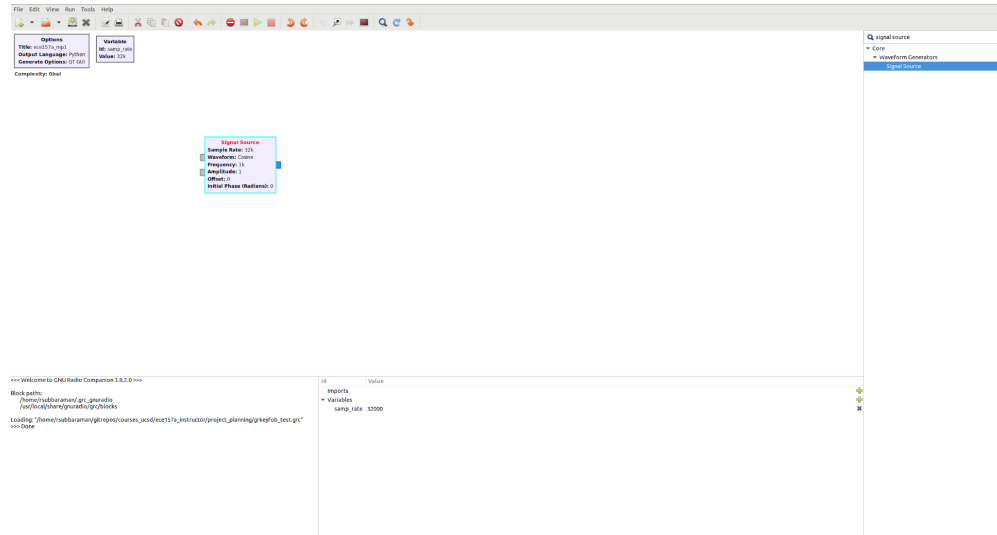
    b.

   c.   This is a source block that outputs items. It will generate complex output items

6. Now we will search for the **QT GUI Time Sink** and add it to the flowgraph

    a.   This is a sink block, it consumes items.

7. Now search for **Throttle** and add it to the flowgraph. The throttle block limits the rate of the stream flowing through it. Without the throttle block, the CPU usage may go very high.

8. Now connect the blocks: **Signal Source → Throttle → QT GUI Time Sink**

    a.   Press the   icon to run the flowgraph.
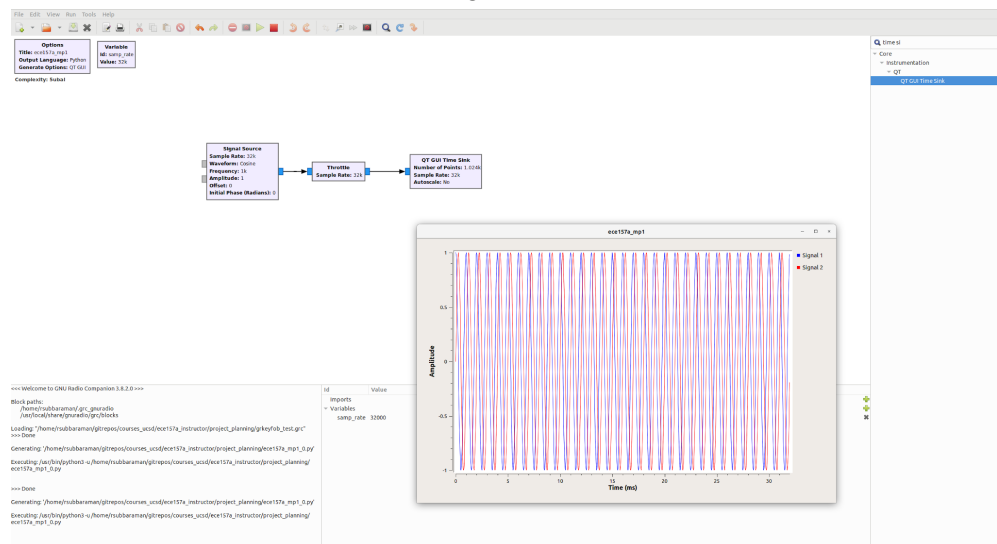

    b.

    c.   A window pops up with a visualization of the signal generated by the **Signal Source**

9. Congratulations! You have made your first flowgraph. Samples "flow" from the **Signal Source** to the **QT GUI Time Sink** through the **Throttle** block. Such "flow" architectures are also referred to as "streaming" architectures or "streams".

10. Adapted from optional reading: https://wiki.gnuradio.org/index.php/Guided_Tutorial_GRC

Jump Off Point

1. You will start with the following reference flowgraph (incomplete and incorrect):
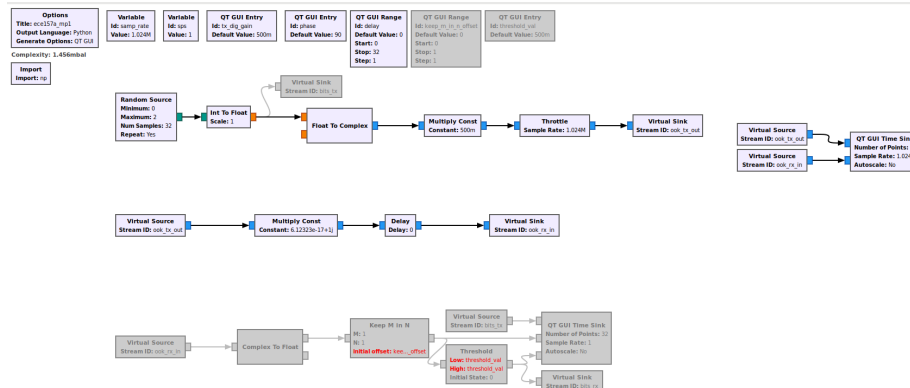
   a.
   b. Make sure you have cloned the latest version of the course git repository:
      https://github.com/ucsdwcsng/ece157A

   c. You can open this flowgraph by:
      cd ece157A/grc/miniproj/
      gnuradio-companion ece157a_mp1_0_reference.grc
   d. Use the "Save As" option to save your modified flowgraph in a different location, name it **ece157a_w21_mp1_<teamidx>.grc**
   e. Some of the modules appear grey because they are disabled. To enable a block, you need to highlight it and press 'E'. To disable a block, press 'D'.
2. The flowgraph has three parts:
   a. OOK Modulator (top)
   b. Channel/Impairments (middle)
   c. OOK Demodulator (bottom)
3. Samples flow from top to bottom, the **Virtual Sink/Source** blocks are "wormholes" for samples. They are used to ease readability.
4. When the reference flowgraph is run, it will open a GUI:
   a. The GUI plots the Tx Signal and RX Signal (Tx Signal + Impairments)
   b. You can scale the Tx Signal using an entry box labeled 'tx_dig_gain'.
   c. You can phase rotate the received signal using the 'phase' entry box.
   d. You can delay the received signal using the delay slider
5. The goal of the rest of this mini-project is to complete the flowgraph and understand it.

OOK Modulation in GNURadio

1. OOK Modulation directly multiplies the carrier with the bit-stream. In this part, we deal only with the **top part of the flowgraph.**
2. **Task 1: Interpolation**

a. Add a **Repeat** block at an appropriate location *before* the **Multiply Const** block to perform upsampling with a rectangular window. Use **sps** as the value in the interpolation field of the **Repeat** block (sps == samples per symbol).

b. We will be using an sps of 128. Change the **Variable** block for sps to 128.

3. **Task 2: Transmitter Frequency Offset**

a. The **Multiply** block takes streams and multiplies them sample-by-sample.
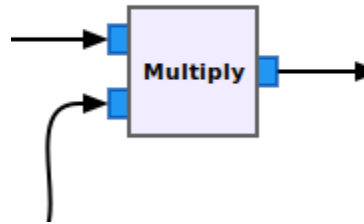
b. Use this block in an appropriate location in the stream to multiply the OOK modulated samples with a Cosine wave from a **Signal Source** block.

c. The frequency of the **Signal Source** can be set manually or using a **Variable** or **QT GUI Entry** block.

4. **Task 3: Observations and screenshots**

a. Take a screenshot of your flowgraph at this stage

b. Run your flowgraph with a frequency offset of 0.5 kHz, phase offset of 45 degrees and take a screenshot.
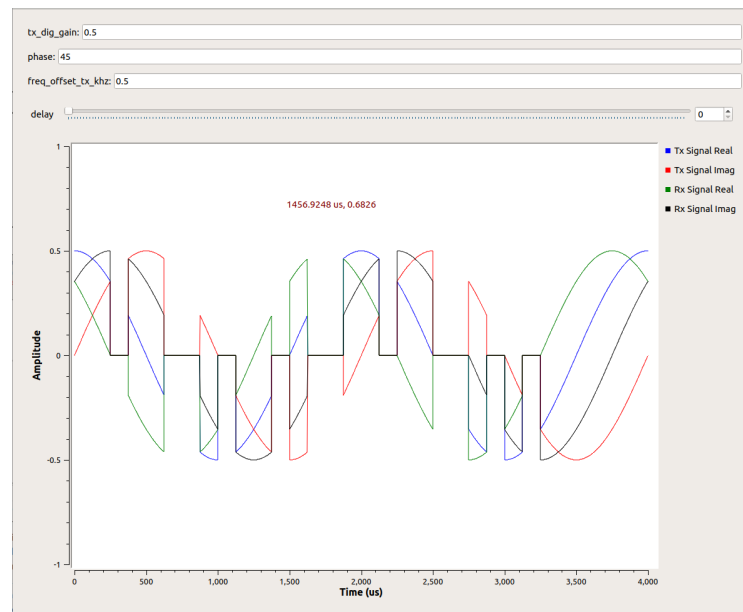
i.

ii. You should see a static image as above. (the actual bits may be different since they are being randomly generated, just make sure the image is static).

OOK Demodulation in GNURadio

1. **Task 1: Rx Gain AWGN Addition**
   a. To digitally scale the Rx Signal, add another **Multiply Const** block and parameterize it similar to the Tx chain (top). Place this block right after the **Multiply Const** Block that is performing phase rotation
   b. To simulate the effect of added noise, we will use a **Noise Source** and an **Add** block.
   c. The **Add** block is similar to **Multiply**: it adds all input streams in a sample-by-sample manner and provides one output
   d. Place the both the blocks between the **Delay** block and the **Virtual Sink** block.
   e. Create a new **QT GUI Entry Block** to supply the Noise Amplitude of the **Noise Source** block
   f. Run your flowgraph with Noise Amplitude 0.1 and take a screenshot.
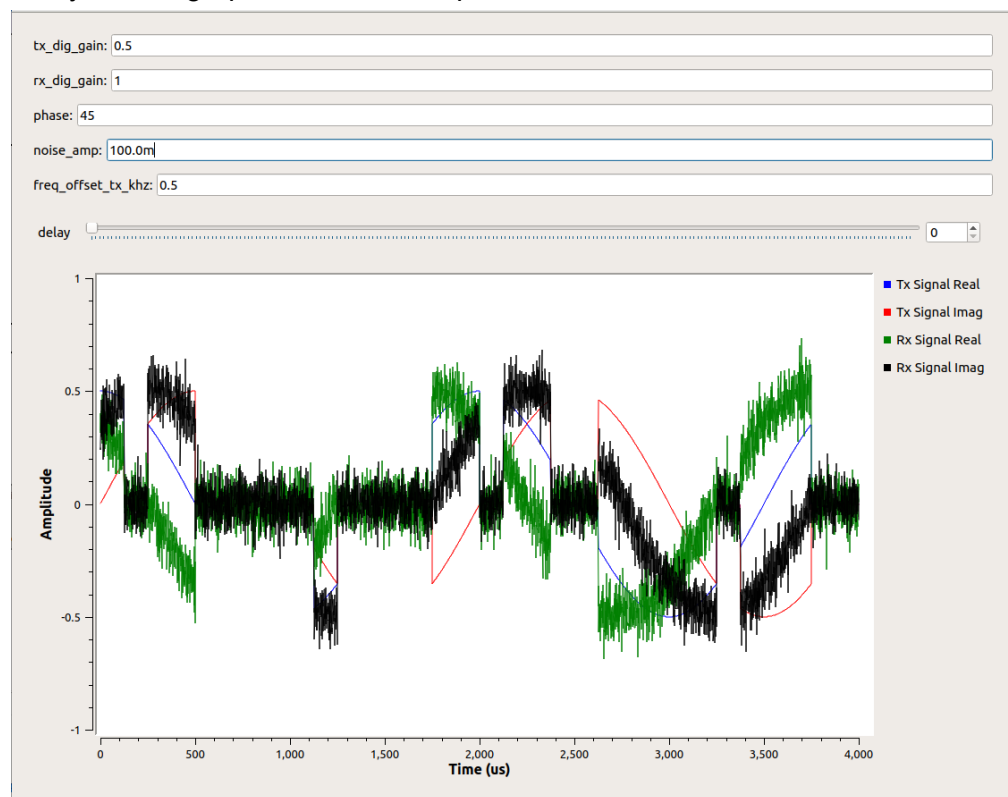
   g.
2. **Task 2: OOK Demodulation**
   a. For this task, set the variables to the following:
      i. Tx_dig_gain = 1
      ii. Rx_dig_gain = 1
      iii. Frequency offset = 0
      iv. Phase offset = 0
      v. Noise amplitude = 0
      vi. Threshold_val = 0.5
   b. Since the pulse-shape is rectangular, we can use a **Moving Average** block as our match filter

c. Highlight all the blocks in the bottom part of the flowgraph and press "E" to enable them. You should also enable all the variable blocks.

d. Place the **Moving Average** block before the complex to float block.

e. Determine the length (number of samples to add up) and scaling (number to divide). Supply the value in block properties. Leave other properties unchanged

f. Now we need to decimate the output of the match filter by picking out one sample every *sps* samples. The **Keep M in N** block does this for us. It picks out M samples every N samples with some initial offset. In this case M=1, N=(length of integrator) and **you have to find the correct value for the initial offset**.

3. **Task 3: OOK Demodulation with Offsets**

a. **Phase Offset**

Keeping all other variables the same, change phase offset to 90 degrees. You would see that the received symbols drastically change because of phase offsets. This is due to the receive chain incorrectly using the **Complex to Float** block and passing on only the real part. Ideally, it should have passed on the magnitude. Replace the block with the correct operation.

b. At this stage, your Tx Bits, Rx Bits and Rx Symbols should all line up:
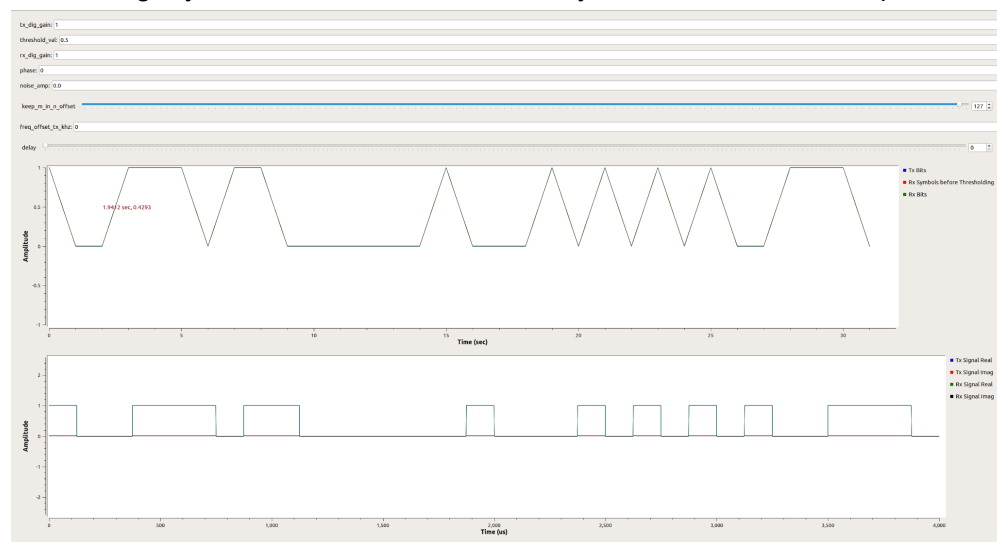
c. **Noise**

Keeping all other variables the same, change noise amplitude to 1.0. Write a one-two line explanation of what you see happening to the Rx Symbols.

d. **Frequency Offset**

Change noise amplitude back to zero. Add a frequency offset of 10 kHz. The Rx Symbols would have all gone close to zero. Why do you think this happened? Does reducing the frequency offset to 2 kHz help? Why/Why Not (one line answers for everything).