Week 1 Lab: Introduction to GNU Radio for Radio Astronomy and Signal Processing

1 Introduction to GNU Radio

GNU Radio is an open-source software toolkit that provides a framework for building software-defined radio (SDR) systems. It allows you to design and implement radio communication systems using software and general-purpose hardware, offering flexibility, customization, and real-time processing capabilities. In this lab, you'll explore GNU Radio in the context of radio astronomy and signal processing, focusing on fundamental principles of electromagnetic waves.

GNU Radio is designed to be highly extensible, allowing users to create custom signal processing blocks and build complex radio systems. Its graphical interface, GNU Radio Companion (GRC), provides a visual environment for creating flowgraphs, which are graphical representations of signal processing workflows. By using GNU Radio, you will gain hands-on experience with radio system design, waveform generation, modulation/demodulation techniques, and signal analysis.

Throughout this lab, you will explore various aspects of GNU Radio, from basic signal generation to signal processing techniques. By the end of the lab, you should have a solid understanding of how GNU Radio works, its applications in radio astronomy and signal processing, and the fundamental principles of electromagnetic waves.

2 Waveform Generation and Visualization Flowgraph

1. Open GNU Radio Companion (GRC) and ensure that you have a new workspace open, you should see an **Options** and a **Variable** block in the upper right corner of the workspace.

2. Using the search feature (magnifying glass icon in the top menu), find and add a **Signal Source** block to your workspace by double clicking or drag and dropping. Double click on the block to open the properties window and change the output type

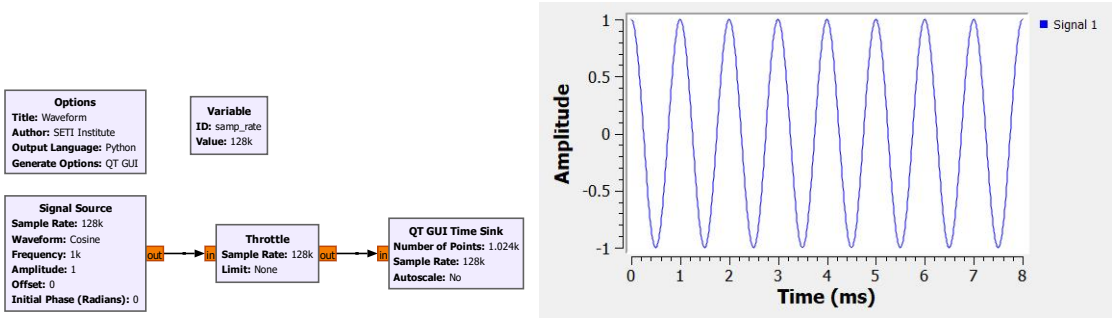
to float. You can also change the data type of a block by left-clicking on it and using the up/down arrows to scroll through the available options. The color of the input/output port indicates the data type, a list of data types and their corresponding colors can be found in the top menu under *help* → *types*. This block can generate a variety of waveforms (constant, sine, cosine, square, triangle, saw tooth) with the usual properties of an electromagnetic wave: frequency, amplitude, and phase.

3. Search for and add a **Throttle** block to your workspace; change the data type to float. Connect the *out* port of the Signal Source block to the *in* port of the Throttle block. This can be done either by dragging between the ports, or by left clicking on each port in succession. The connecting arrow should be black; if the arrow is red then it is indicating an error, usually that the blocks you're trying to connect have different data types. This block will sample the incoming signal at a specific sample rate. Without this block your computer will attempt to sample the signal as fast as it possibly can, which can overload your CPU, all for the simple task of generating a sine wave.

4. Search for and add a **QT GUI Time Sink** block to your workspace. Open the properties window and change the type to float; connect it to the Throttle block. This block will allow you to visualize the waveform in an amplitude vs. time plot.

5. In the **Variable** block, change the sample rate to 128000.

6. Run the flowgraph using the play button on the top menu. Your flowgraph and plot should look like the figures below.



3 Data Types and Variables

GRC uses Python data types to represent variables. Python supports several built-in data types, including integers (int), floating-point numbers (float), strings (str), booleans (bool), lists, tuples, sets, and dictionaries. Integers are whole numbers without decimal points,

while floating-point numbers include decimal points. Strings are sequences of characters enclosed in single (' ') or double (" ") quotes and are used to represent text data. Booleans have two possible values, True or False, and are often used in conditional statements and logical operations.

Lists are ordered collections of items that can be of different data types, and they are mutable, meaning their elements can be changed. Tuples are similar to lists but are immutable, meaning their elements cannot be changed after creation. Sets are unordered collections of unique items, useful for mathematical operations like union, intersection, and difference. Dictionaries are collections of key-value pairs, where each key is associated with a value, enabling efficient data retrieval based on keys.

In Python, variables are used to store and manipulate data. They act as placeholders that represent values or objects in memory. When you assign a value to a variable, Python allocates memory space to store that value, making it accessible throughout your program. Variables can hold various data types, such as numbers (integers or floating-point), strings (text), booleans (True or False), lists, tuples, dictionaries, and more complex objects.

4 Runtime Updating Variables

With the current flowgraph, the variables such as frequency, amplitude, and phase are fixed once the flowgraph is running. In order to change these variables you would have to end the flowgraph, edit the properties in the Signal Source block, and run the flowgraph again. However, there is an easy way to change variables in the flowgraph while it's running using QT GUI Widgets.

1. Search for the **QT GUI Range** block and add it to your workspace. This block will allow you to create a variable used in the flowgraph which you can change while the flowgraph is running. Open the properties and make the following changes:

   (a) ID: freq

   (b) Label: Signal Frequency

   (c) Default Value: 1000

   (d) Start: 0

   (e) Stop: 10000

   (f) Step: 100

2. In the Signal Source block, enter the variable ID [freq] for the frequency.

3. Run the flowgraph again using the play button on the top menu. You should see the same display as before, but with a slider at the top of the graph that allows you to adjust the frequency while the flowgraph is running. From what you know about the relationship between frequency and wavelength, what do you expect will happen to the wavelength as you increase the frequency?