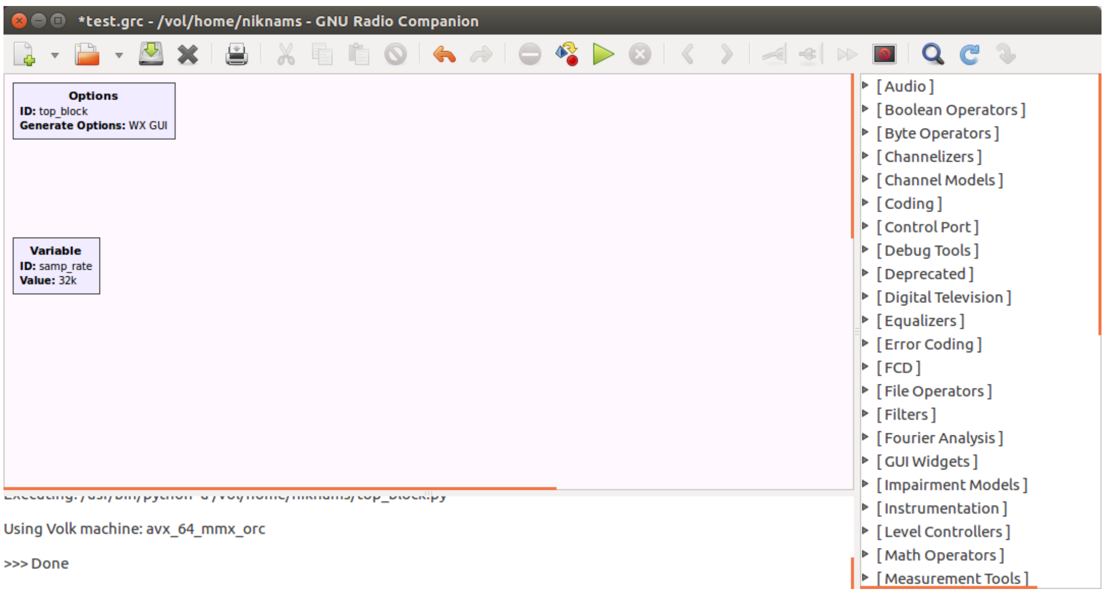
First Assignment

In this assignment, the main objective is to use GNU Radio Companion (GRC) software to create analog/digital signals and apply different modulation schemes on them. The GRC software is a graphical user interface that allows you to build GNU Radio flow graphs. To learn the basics of GNU Radio, a brief tutorial is provided below in which you can learn how to create and execute a GRC flow graph using basic blocks such as signal sources and graphical sinks.

I. Getting Started with the GRC

To launch the GRC software, open a Terminal in Linux and type gnuradio-companion. Then, an untitled GRC window similar to figure 1 should open.



Options
ID: top_block
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 32k

```
Using Volk machine: avx_64_mmx_orc

>>> Done
```

Fig. 1: Blank GRC flow graph

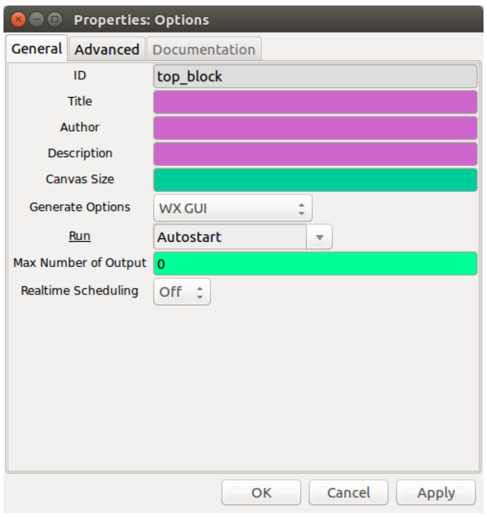
In this window, there are two basic blocks. The Options block sets some general parameters for the flow graph. Double-click on the Options block to see its properties, that is shown in the figure 2. Then, in the opened window

- set Generate Options to WX GUI. This setting controls the way that GUIs are generated for flow graph output. The output plots used in this labs will not be available if this option is set to the default QT GUI,
- set Run to Autostart, and Realtime Scheduling to Off,
- click OK to close the properties dialog.

The other block in the flow graph is a variable block that defines a variable called sample_rate and sets a value for it. Click on this block to see the variable name and its value, that is shown in the figure 3. This variable can be used later to initialize other parameters in flow graph.

A. *Adding Blocks to the Flow graph*

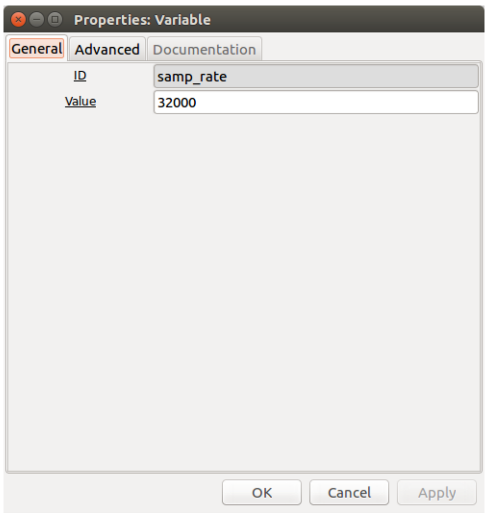
On the right side of the window is a list of the blocks that are available. By expanding any of the categories (click on triangle to the left) you can see the available blocks. You can also click on the magnifying glass in the upper right side of the window and simply type a search term to search all


Properties: Options

General | Advanced | Documentation

ID	top_block
Title	
Author	
Description	
Canvas Size	
Generate Options	WX GUI
Run	Autostart
Max Number of Output	0
Realtime Scheduling	Off

Fig. 2: Properties dialog box for top block


Properties: Variable

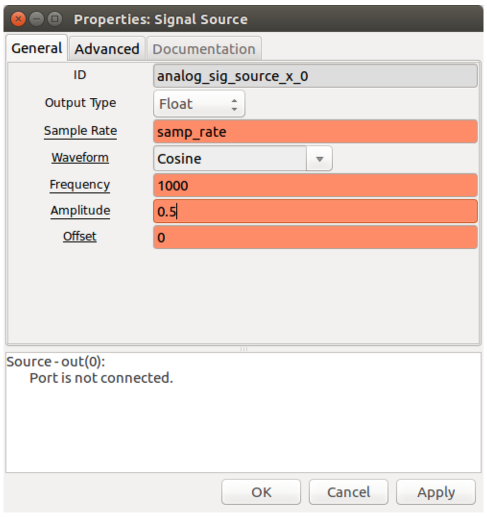
General | Advanced | Documentation

ID	samp_rate
Value	32000

Fig. 3: Properties dialog box for sample_rate

categories. A small text window will appear above the list of blocks in which your search term will be entered.

- Open the **Waveform Generators** category and double-click on the **Signal Source**. Note that a Signal Source block will now appear in the main window.
- Double-click on the Signal Source block and the properties dialog will open. In Waveform option, you can specify the output waveform to be Constant, Sine, Cosine, Square, Triangle, and Sawtooth. Adjust the settings to match those as shown in figure 4 and close the dialog.

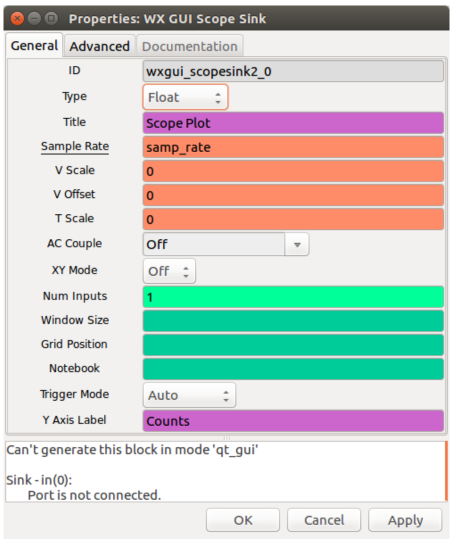

Properties: Signal Source

General | Advanced | Documentation

ID	analog_sig_source_x_0
Output Type	Float
Sample Rate	samp_rate
Waveform	Cosine
Frequency	1000
Amplitude	0.5
Offset	0

Source - out(0):
Port is not connected.

Fig. 4: Properties dialog box for signal source


Properties: WX GUI Scope Sink

General | Advanced | Documentation

ID	wxgui_scopesink2_0
Type	Float
Title	Scope Plot
Sample Rate	samp_rate
V Scale	0
V Offset	0
T Scale	0
AC Couple	Off
XY Mode	Off
Num Inputs	1
Window Size	
Grid Position	
Notebook	
Trigger Mode	Auto
Y Axis Label	Counts

Can't generate this block in mode 'qt_gui'

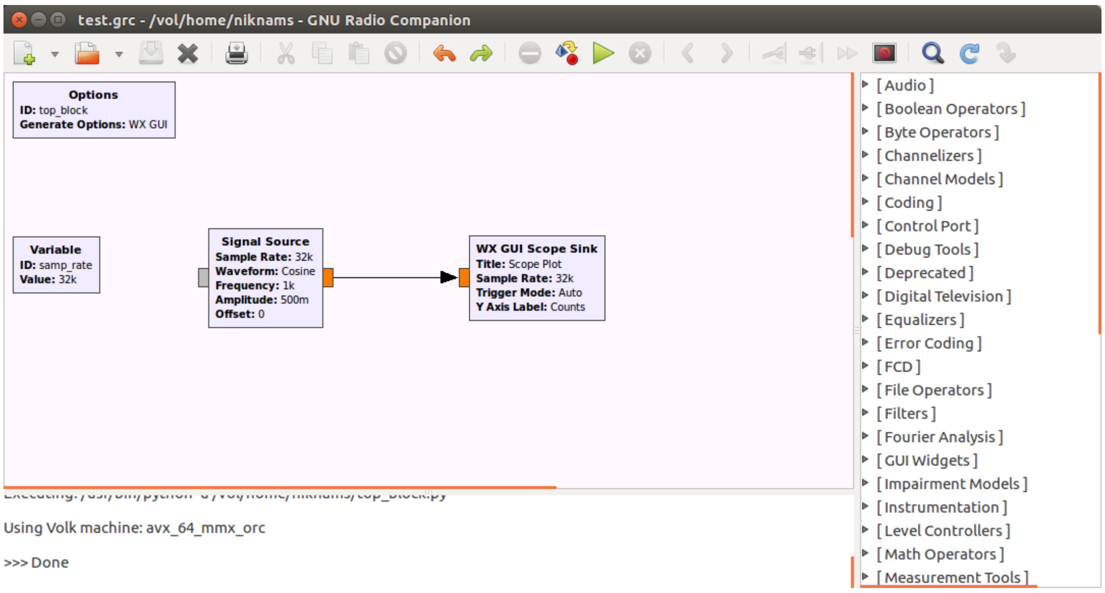
Sink - in(0):
Port is not connected.

Fig. 5: Properties dialog box for scope sink

This Signal Source is now set to output a 1 kHz sinusoid with a peak amplitude of 0.5. In the flow graph, the Signal Source block will have an orange output tab, representing a float data type. If the block output type is chosen as complex instead of float, then the output tab will be blue.

- In order to view this wave, we need one of the graphical sinks. Expand the **Instrumentation** category and then the **WX** subcategory. Double-click on the **WX GUI Scope Sink**. It should appear in the main window. Double-click on the block and change the Type to Float. Leave the other Parameters at their default values as shown in figure 5. Click OK to close the properties dialog.

- In order to connect these two blocks, click once on the out port of the Signal Source, and then once on the in port of the Scope Sink. The resulting flow graph is shown in figure 6.



Options
ID: top_block
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 32k

Signal Source
Sample Rate: 32k
Waveform: Cosine
Frequency: 1k
Amplitude: 500m
Offset: 0

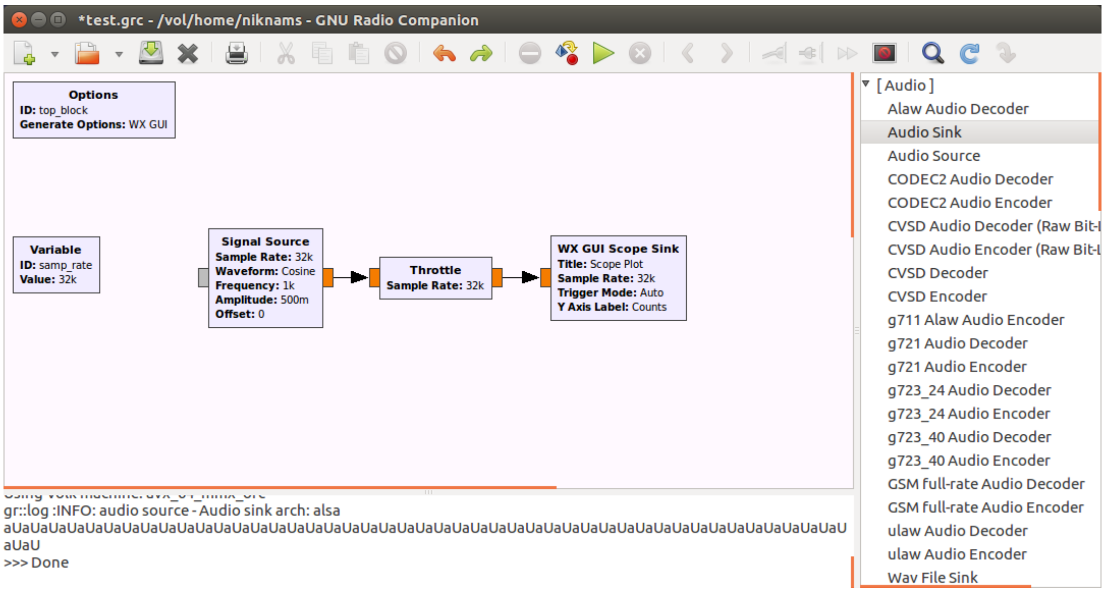
WX GUI Scope Sink
Title: Scope Plot
Sample Rate: 32k
Trigger Mode: Auto
Y Axis Label: Counts

```
Using Volk machine: avx_64_mmx_orc

>>> Done
```

Fig. 6: Flow graph with signal source and scope sink

- The problem with this flowgraph is that although the sample rate is set to 32000, there is no block which enforces this sample rate. Therefore, the flowgraph will consume as much of the computer's resources as it possibly can which can cause the GRC software to lock up. To fix this problem, disconnect the Signal Source from the WX GUI Scope Sink by clicking on the arrow and pressing the Delete key. Expand the **Misc** category and double-click on the **Throttle**. Connect this block between the Signal Source and the WX GUI Scope Sink as shown in figure 7 (click once on the out port of one block and the in port of the next block).



Options
ID: top_block
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 32k

Signal Source
Sample Rate: 32k
Waveform: Cosine
Frequency: 1k
Amplitude: 500m
Offset: 0

Throttle
Sample Rate: 32k

WX GUI Scope Sink
Title: Scope Plot
Sample Rate: 32k
Trigger Mode: Auto
Y Axis Label: Counts

```
gr::log :INFO: audio source - Audio sink arch: alsa
aUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaU
aUaU
>>> Done
```

Fig. 7: Flow graph with throttle added

B. *Executing the Flow graph*

In order to observe the operation of this simple system, we must generate the flow graph and then execute it. To do so,

- Click first on the icon to generate the flow graph.
- If the flow graph has not yet been saved, a file dialog will appear when you click this button. Name this file as you want and save it to a folder on your home drive. The generation stage converts your flow graph into executable Python code.
- Click the icon to execute the flow graph. The execution stage runs the Python code that was generated in the previous step. A scope plot should open displaying several cycles of the sinusoid. Confirm that the frequency and amplitude match the value that you expect.

C. *Working with the Scope Sink*

- Change the **Channel Options / Marker** setting to **Dot Large**. As shown in figure 8, you can now see the actual sample values. Recall that the Variable block sets the sampling rate to 32000 samples/second or 32 samples/ms. Note that there are in fact 32 samples within one cycle of the wave.

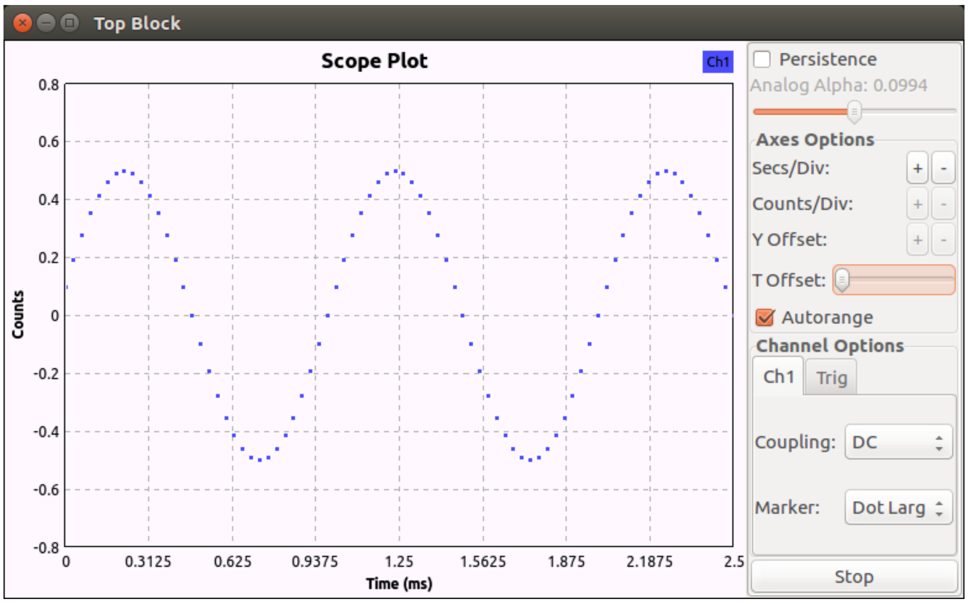


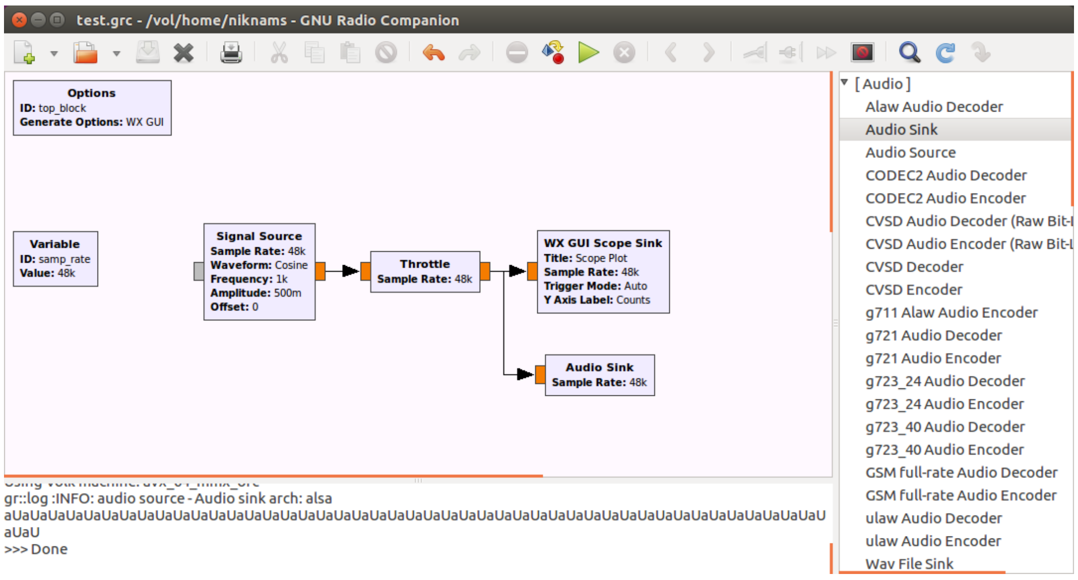
Fig. 8: Scope sink with large dot markers

- Close the scope and reduce the sample rate to 10000 by double-clicking on the Variable block and entering 10e3 in the Value box. Note that you can use this exponential notation anywhere when GRC requires a number.
- Re-generate and execute the flow graph. Note that there are now fewer points per cycle. (T1) **How low can you drop the sample rate?** Recall that the Nyquist sampling theorem requires that we sample at most two times the highest frequency. (T2) **Experiment with this and see how the output changes as you drop below the Nyquist rate.**

D. *Working with Audio I/O*

Create the flowgraph shown in figure 9.

- The **Audio Sink** is found in the **Audio** category. The Audio Sink block directs the signal to the audio card of your computer. Note that the sample rate is set to 48000, a sample rate that is usually, but not always supported by computer audio hardware.



```
gr::log :INFO: audio source - Audio sink arch: alsa
aUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaUaU
aUaU
>>> Done
```

Fig. 9: Flow graph for audio I/O demonstration

- Generate and execute this flow graph. The graphical display of the scope opens as before. However, now you should also hear the 1 kHz tone. **If you do not hear the tone, ensure that the output from the computer is connected to the speakers (or headset) and that the volume is turned up**. Experiment with changing both the overall sample rate in the flow graph as well as the sample rate in the audio sink to see how the tone is affected.

E. *Multiple plots in one graph*

For this assignment, you need to create graphs for your final report, and use them to reason why your implementations are correct.
In order to do this, it is useful to have the input and output signals in one plot. Double click on a placed WX GUI Scope Sink, go to field '*inputs*', and set it to 2. Your scope sink can now handle 2 inputs, as seen in the following Figure. The output of above sample looks like:

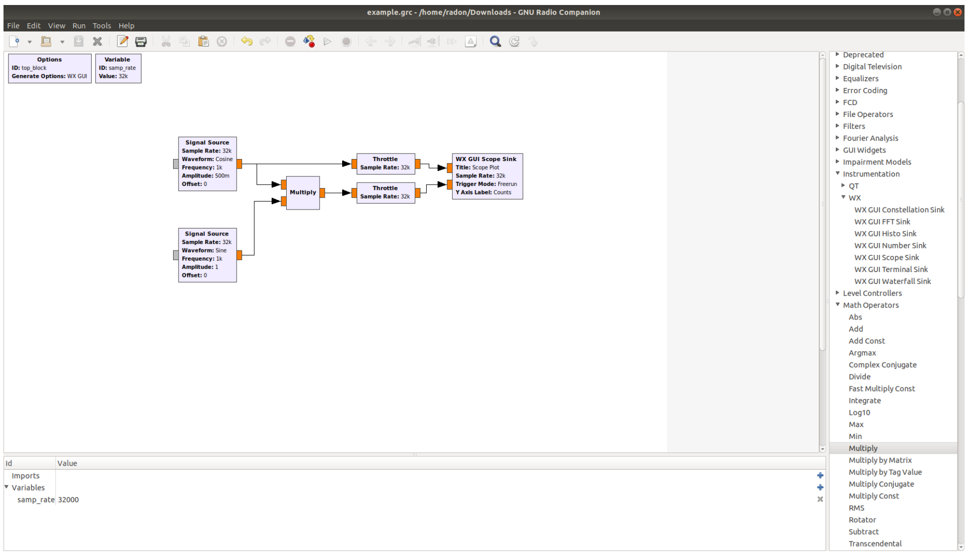


Fig. 10: Example graph leading input and output signals to scope sink

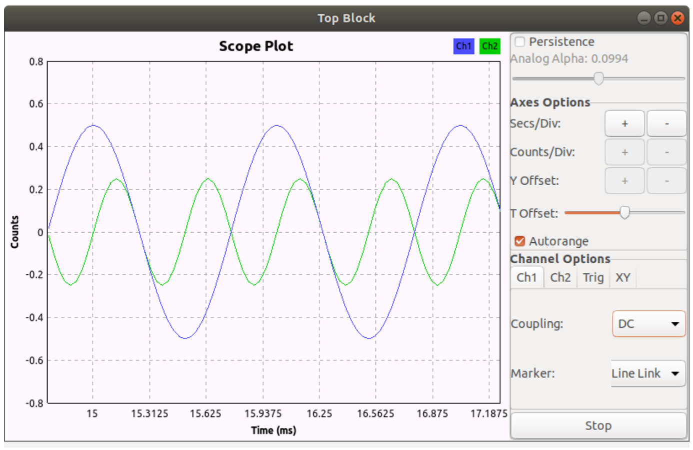
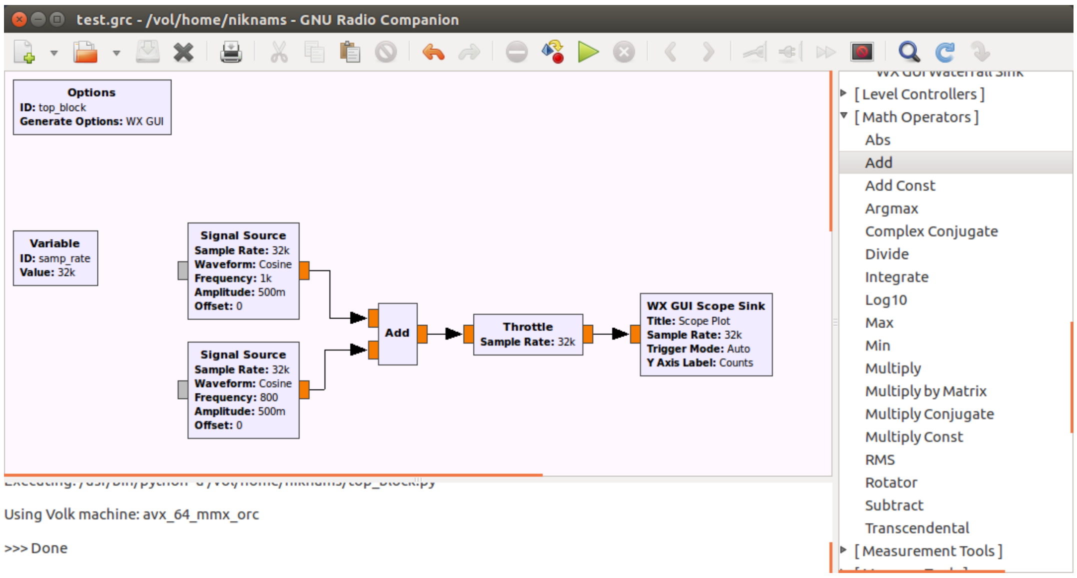


Fig. 11: Output plot with multiple input signals

F. Math Operations

- Construct the flow graph in figure 12. Set the sample rate to 32000. The two Signal Sources should have frequencies of 1000 and 800, respectively. The Add block is found in the Math Operators category.



Options
ID: top_block
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 32k

Signal Source
Sample Rate: 32k
Waveform: Cosine
Frequency: 1k
Amplitude: 500m
Offset: 0

Signal Source
Sample Rate: 32k
Waveform: Cosine
Frequency: 800
Amplitude: 500m
Offset: 0

Throttle
Sample Rate: 32k

WX GUI Scope Sink
Title: Scope Plot
Sample Rate: 32k
Trigger Mode: Auto
Y Axis Label: Counts

```
Using Volk machine: avx_64_mmx_orc

>>> Done
```

Fig. 12: Flow graph with Add block

- Generate and execute the flow graph. On the Scope plot you should observe a waveform corresponding to the sum of two sinusoids.
- (T3) **Replace the Add block with a Multiply block. What output do you expect from the product of two sinusoids? Confirm your result on the Scope display.**
- Note that the other math operations exist under the Math Operators category and experiment with a few to see if the result is as expected.

II. Tasks

Using the above tutorial, you should implement the following signal modulation schemes:

- Analogue modulation schemes
  - Frequency Modulation (FM)

  - (T4) **Amplitude Modulation (AM)**
  - (T5) **Phase Modulation (PM)**
- Digital modulation schemes
  - (T6) **Frequency-shift keying (FSK)**
  - (T7) **Amplitude-shift keying (ASK)**
  - (T8) **Phase-shift keying (PSK)**

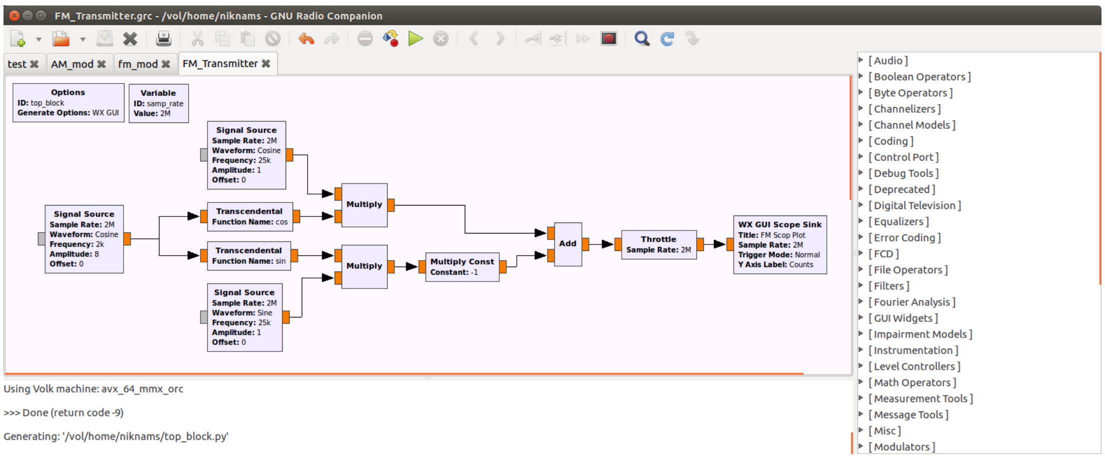
  Please note that for generating digital signals you should use either **Vector Source** block under **Misc** category or **Square** wave in **Signal Source** block. Then, your above implementations should confirm the resemblance of
  - (T9) **FM and PM modulations.**
  - (T10) **FSK and PSK modulations on square and triangle waveforms, respectively.**

A. *Reference implementation of FM*

To get a hint how to implement the modulation schemes, in this section, a step-by-step implementation of Frequency Modulation (FM) is provided. Recall from the Encodings lecture: $s(t) = A\cos(2\pi f_c t + \phi(t))$. For FM, $\phi(t) = n_f \times m(t)$. For a modulating wave $x_m(t) = A_m \cos(2\pi f_m t)$ and a carrier wave $m(t) = A_c \cos(2\pi f_c t)$, the FM wave is $s(t) = A_c \cos(2\pi f_c t + A_m \cos(2\pi f_m t))$.
Sadly, there is no convenient way to perform $\cos(x+y)$ in GNU Radio Companion. However, we know $\cos(x+y) = \cos(x)\cos(y) - \sin(x)\sin(y)$. With this information, we can expand our formula to

$$s(t) = A_c \cos(2\pi f_c t)\cos(A_m \cos(2\pi f_m t)) - A_c \sin(2\pi f_c t)\sin(A_m \cos(2\pi f_m t)). \qquad (1)$$

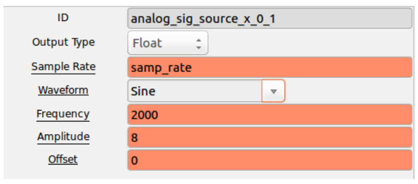
Figure 13 shows the flow graph of FM based on Equation 1 with f_m = 2 KHz , A_m = 8, f_c = 25 KHz, A_c = 1, and using a sample rate f_s = 2 MHz.



```
Using Volk machine: avx_64_mmx_orc
>>> Done (return code -9)
Generating: '/vol/home/niknams/top_block.py'
```

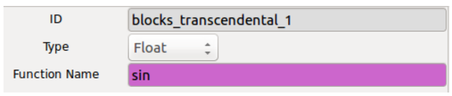
Fig. 13: Flow graph of Frequency Modulation

For the blocks used in Figure 13, a brief explanation is provided in the following:
- In Figure 13, the first block is used for creating the signal $x_m(t) = A_m \cos(2\pi f_m t)$ which is parameterized as below:

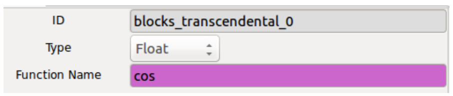


ID	analog_sig_source_x_0_1
Output Type	Float
Sample Rate	samp_rate
Waveform	Sine
Frequency	2000
Amplitude	8
Offset	0

- Transcendental Blocks are used to obtain the sine and cosine of $x_m(t)$ and create $I(t) = \sin(A_m \cos(2\pi f_m t))$ and $Q(t) = \cos(A_m \cos(2\pi f_m t))$ in equation (1). Double click on the Transcendental Blocks and set their parameters as below:

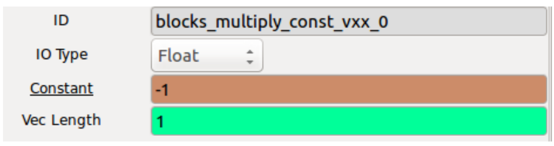


ID	blocks_transcendental_1
Type	Float
Function Name	sin



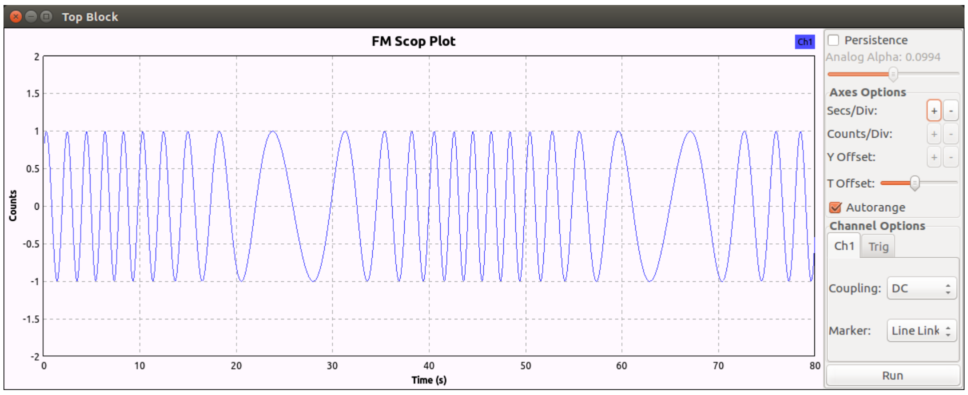
ID	blocks_transcendental_0
Type	Float
Function Name	cos

- Then, I(t) and Q(t) are multiplied by $A_c \sin(2\pi f_c t)$ and $A_c \cos(2\pi f_c t)$ which are created using two signal source blocks.
- To deduct the $A_c \cos(2\pi f_c t) * Q(t)$ from $A_c \sin(2\pi f_c t) * I(t)$, $A_c \sin(2\pi f_c t) * I(t)$ is multiplied to -1 using Multiply Const block that is parameters as below:



ID	blocks_multiply_const_vxx_0
IO Type	Float
Constant	-1
Vec Length	1

- The results of $A_c \cos(2\pi f_c t) * Q(t)$ and $-A_c \sin(2\pi f_c t) * I(t)$ are then added using the Add block with two inputs.
- Finally, the Throttle sample rate is set to samp_rate and WX GUI Scope Sink is used to see properties of FM signal with Sinusoidal message. You can change the carrier frequency and message frequency and amplitude and see the changes. Following figure shows the result for the default parameters.



B. *Teams, Submission and Grading*

You can work on this assignment in a team of **at most two persons**. Teams need to submit the following items (only 1 member per team has to do this):

- .grc files answering questions T4, T5, $\cdots$, T10.
- A well-organized report in PDF-format, answering the three questions of tutorial T1, T2, T3. For T4, T5..., T10, describe how you created your implementations, include a picture of the output graph, and show that your implementation of the algorithms work by referring to the graphs and respective algorithm formulas.

Please create a "gzipped tar" archive of your submitting items and name it as **assignment1-sXXXXXXX-sYYYYYYY.tar.gz** in which XXXXXXX and YYYYYYY are your student IDs. Submit the tar-archive by e-mail to "s.f.alvarez.rodriguez@umail.leidenuniv.nl". Please be aware that only two lab sessions are assigned for this assignment: This might not be sufficient to finish the above tasks. So, take some additional time during week to be able to finish the tasks and hand everything in on time.
Deadline: **Monday 2 March, 2020**, at **23:59**.
For late submissions, we subtract 1 point per day past the deadline.