Guided Tutorial GRC

From GNU Radio

Contents

Objectives

- Create flowgraphs using the standard block libraries
- Learn how to debug flowgraphs with the instrumentation blocks
- Learn how to use the documentation to figure out block's functionality

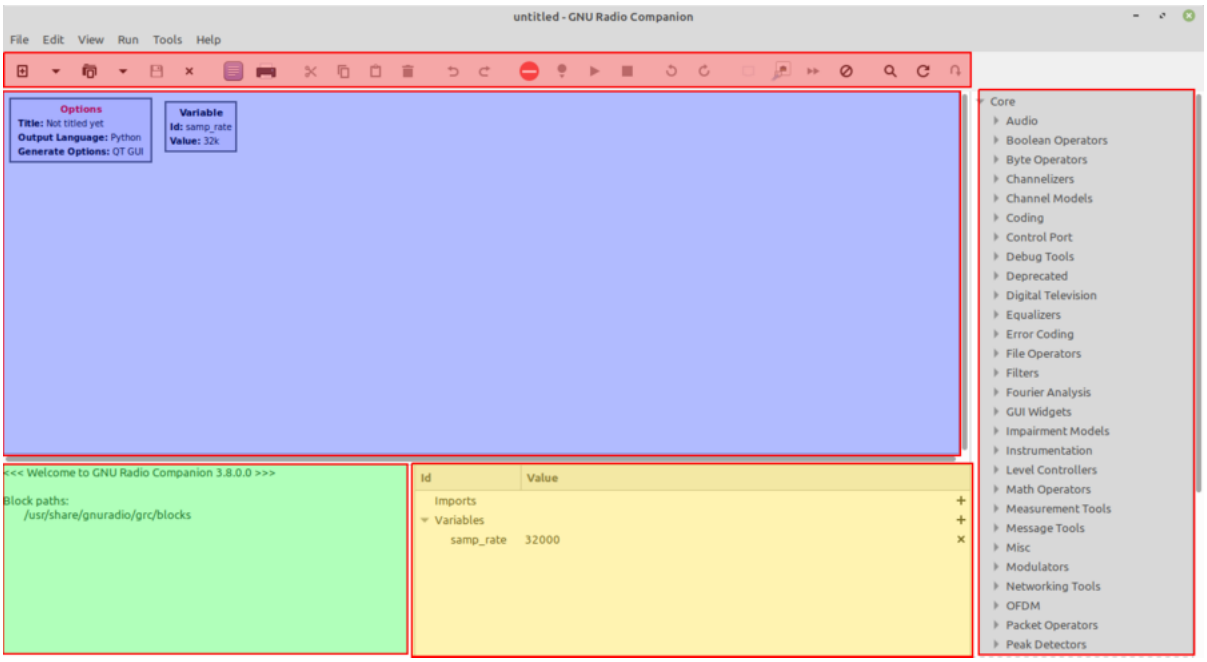
Prerequisites

- GNU Radio 3.8.0.0 or later
- A brief introduction to GNU Radio, SDR, and DSP

Getting to Know the GRC

We have seen in A brief introduction to GNU Radio, SDR, and DSP that GNU Radio is a collection of tools that can be used to develop radio systems in software as opposed to completely in hardware. In this tutorial, we start off simply and explore how to use the GNU Radio Companion (GRC), GNU Radio's graphical tool, to create different tones. We should keep in the back of our mind that GRC was created to simplify the use of GNU Radio by allowing us to create python files graphically as opposed to creating them in code alone (we will discuss this more later).

The first thing to cover is the interface. There are five parts: Library, Toolbar, Terminal, Workspace and variables.



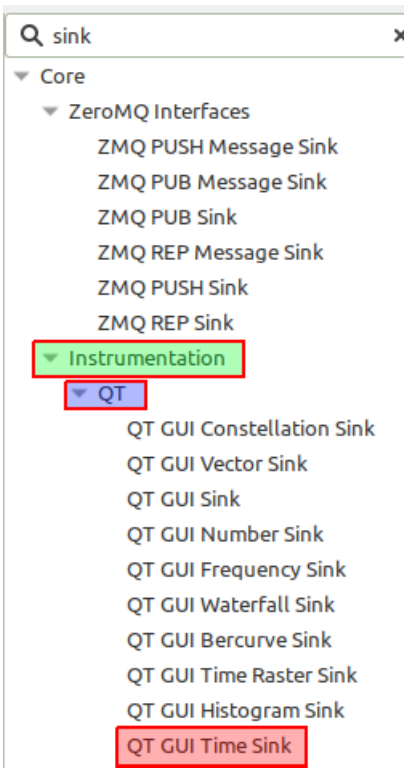
The tutorial is meant to be hands on, so please take a few breaks from reading here and there to experiment. We will reiterate that these tutorials as simply meant as guides and that the best way to learn something is to try it out: come up with a question, think of a possible solution, and try it out. Let us begin by starting up the GRC! This is usually done by opening up a terminal window (ctrl+alt+t in Ubuntu) and typing:

```
$ gnuradio-companion
```

If you are unable to open GRC then you need to go back to InstallingGR and troubleshoot the installation.

Searching for Blocks

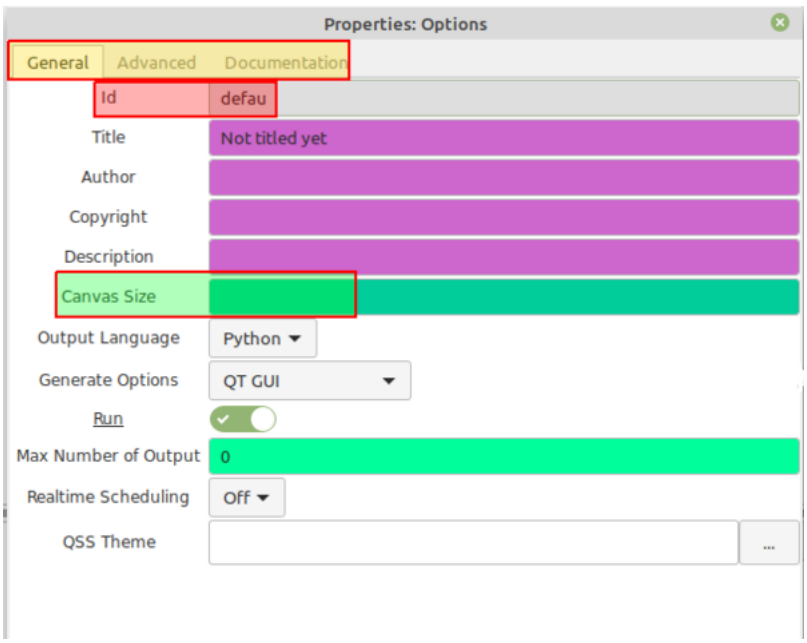
The Library contains the different blocks installed in the GRC block paths. Here we find blocks that are preinstalled in GNU Radio and blocks that are installed on the system. At first it seems daunting to look for blocks. For instance, if we want to generate a waveform, what category should we look in? We see there is a **Waveform Generators** category, okay not bad. But what if we wanted to find some way to display our data but aren't sure of the best way to display it yet? We know from A brief introduction to GNU Radio, SDR, and DSP that this is known as a sink; however, looking through the list there is no Sinks category. The solution is to use the Search feature by either clicking the magnifying glass icon or typing Ctrl+f and then start typing a keyword to identify the block. We see a white box appear at the top of the Library with a cursor. If we type "sink", we can find all the blocks that contain the words "sink" and the categories where we will find each block.



For now, let's add the block called **QT GUI Time Sink** by clicking on its name and dragging it to the Workspace or double-clicking on its name for it to be placed automatically in the Workspace.

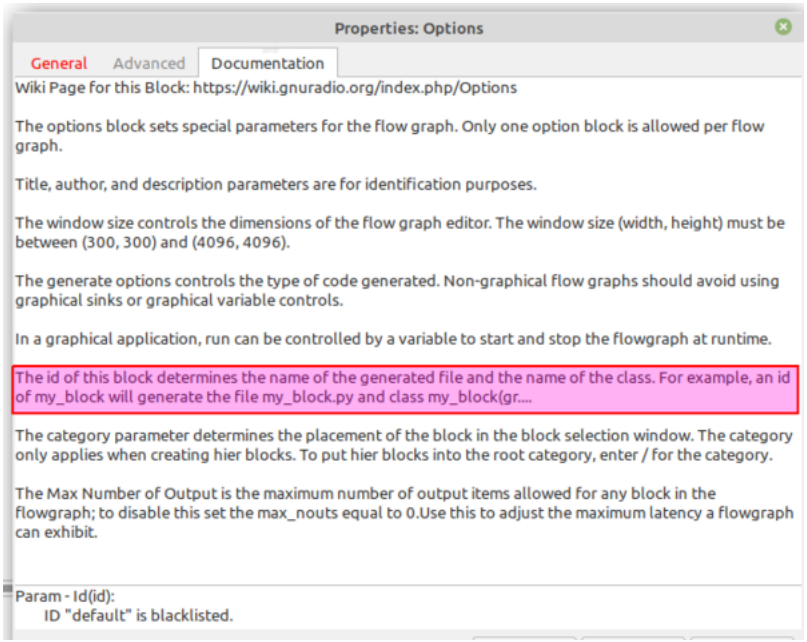
Modifying Block Properties

The workspace (main area of the screen) contains all of our blocks that make up our flowgraph, and inside each block we can see all the different block parameters. There is, however, one special block that each new flowgraph starts with and is required to have, called the **Options Block**. Let us double-click on the Options Block to examine its properties. We see a window as below:

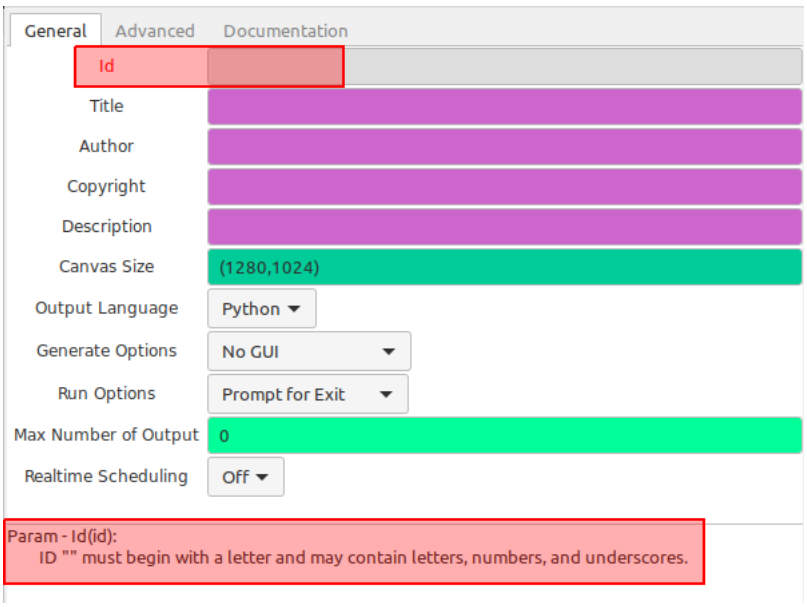


These block properties can be changed from the defaults to accomplish different tasks. Let's remove part of the current name and notice the ID turns blue. This color means that the information has been edited, but has not been saved. If we go back to

the **Options Block** properties, we can see that there are different tabs and one is titled documentation.



If we read a couple lines, we can see that the property **ID** is actually used for the name of the python file the flowgraph generates.



So now let's remove the entire ID string. Notice now that at the bottom appears an error message. Also notice that the parameter **ID** is now red to show us exactly where the error occured.

To keep things organized, let us change the **ID** to "tutorial_two_1". Let us also make sure that the property **Generate Options** is set to "QT GUI" since we are using a graphical sink. The **ID** field allows us to more easily manage our file space. While we save the GRC flowgraph as a <filename>.grc, generating and executing this flowgraph produces another output. GRC is a graphical interface that sits on top of the normal GNU Radio programming environment that is in Python. GRC translates the flowgraph we create in the GUI canvas here into a Python script, so when we execute a flowgraph, we are really running a Python program. The **ID** is used to name that Python file, saved into the same directory as the .grc file. By default, the **ID** is **default** and so it creates a file called **default.py**. Changing the **ID** allows us to

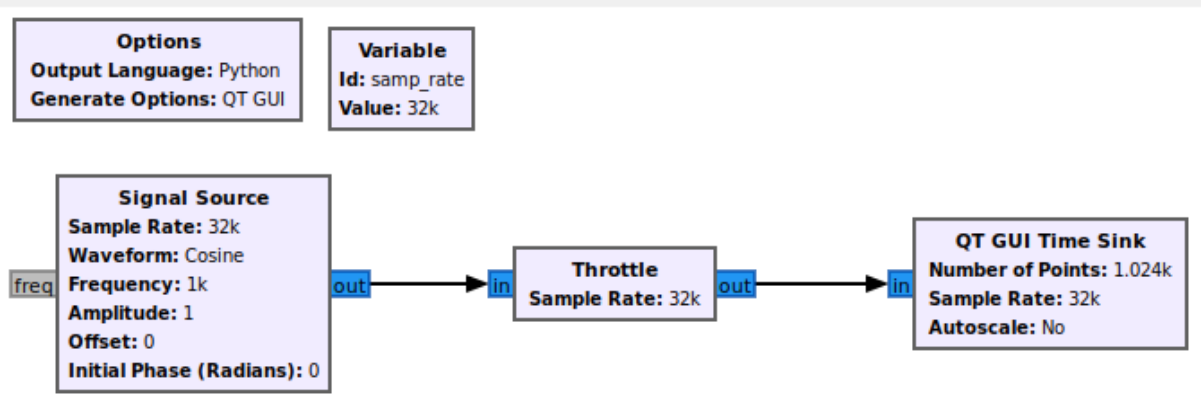
change the saved file name for better file management. In GNUradio 3.8 you will get an error if you don't change the default id, so you need to change this id in order to run the flowgraph.

Another result of this GRC-Python connection is that GRC is actually all Python. In fact, all entry boxes in block properties or variables that we use are interpreted as Python. That means that we can set properties using Python calls, such as calling a numpy or other GNU Radio functions. A common use of this is to call into the **filter.firdes** filter design tool from GNU Radio to build our filter taps.

Another key to notice is the different colors present in the fields we can enter information. These actually correspond to different data types which we will cover later in this tutorial.

My First Flowgraph

Now that we have a better understanding of how to find blocks, how to add them to the workspace, and how to edit the block properties, let's proceed to construct the following flowgraph of a **Signal Source** being sent to a **Throttle Block** and then to our **Time Sink** by clicking on the data type colored ports/tabs one after the other to make connections:



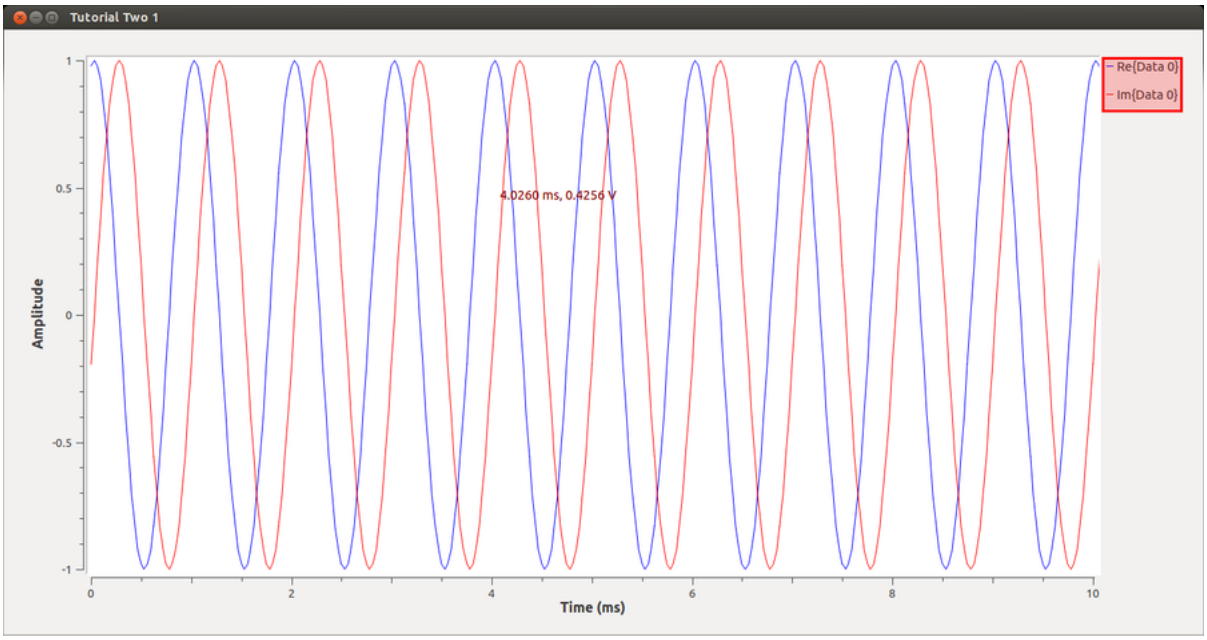
With a usable flowgraph we can now proceed to discuss the Toolbar.

A note on the **throttle block**: What exactly this does is explained in greater detail later on in this tutorial. For now, it will suffice to know that this block throttles the flow graph to make sure it doesn't consume 100% of your CPU cycles and make your computer unresponsive.

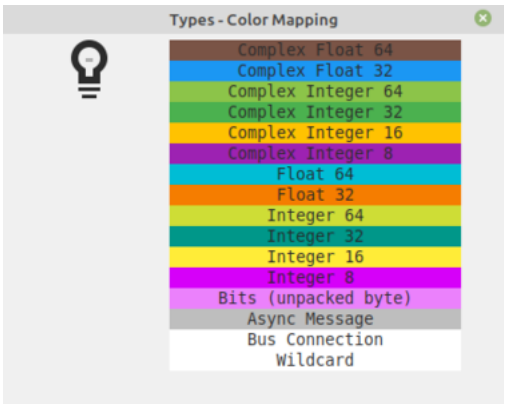
This section of the interface contains commands present in most software such as new, open, save, copy, paste. Let's begin by saving our work so far and titling our flow graph **tutorial_two**. Important tools here are **Generate** flowgraph, **Execute** flowgraph, and **Kill** flowgraph all accessible through F5, F6, and F7 respectively. A good reference is available in **Help->Types** that shows the color mapping of types which we will look into later.

Examining the Output

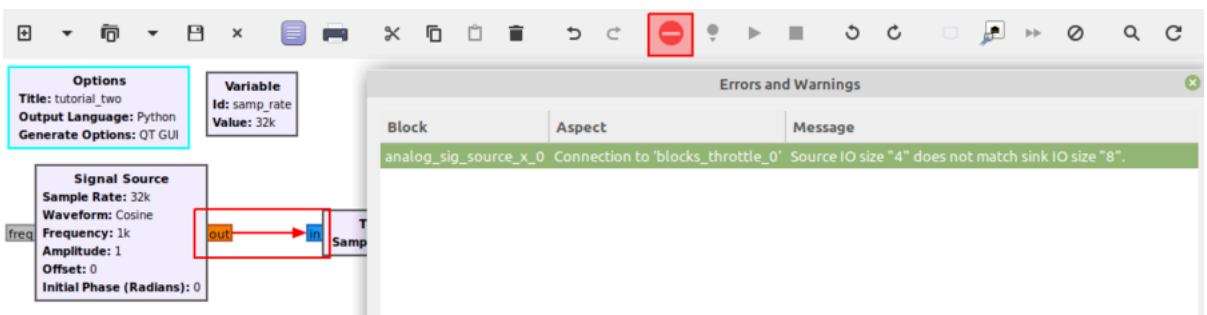
Let's go ahead and **Execute** the flowgraph to see our sine wave.



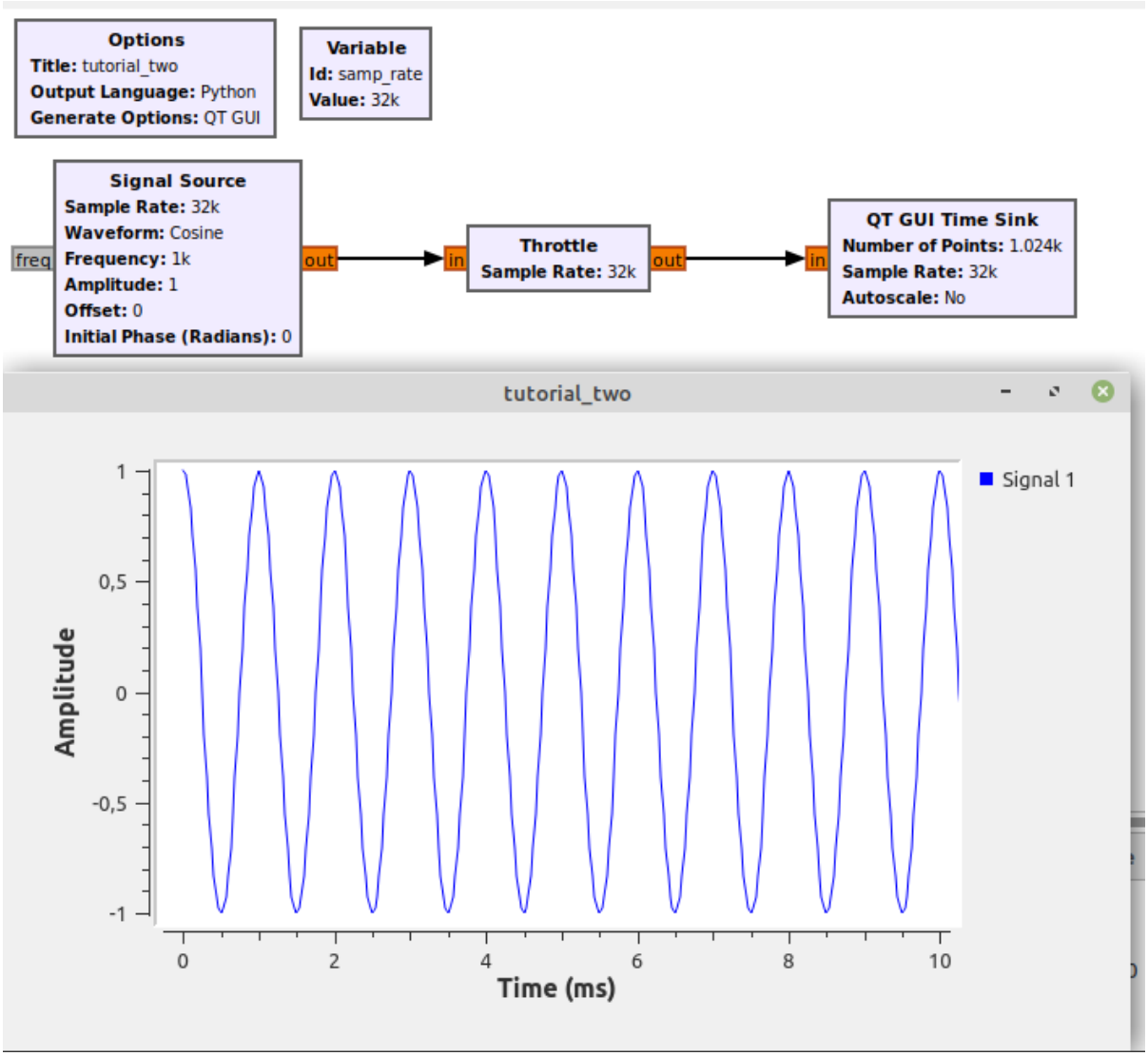
We should get the above which is a complex sinusoid of the form e^{jwt}. Let us keep things simple by avoiding complex signals for now. First we want to kill the flowgraph with the **Kill** flowgraph button or by simply closing the **Time Sink** GUI. Now is a good time to go over data types in GNU Radio by opening up the **Help->Types** window.



We can see common data types seen in many programming languages. We can see our blocks (blue ports) are currently **Complex Float 32** type which means they contain both a real and imaginary part each being a **Float 32** type. We can reason that when the **Time Sink** takes in a complex data type, it outputs both the real and imaginary part on separate channels. So let's now change the **Signal Source** to a float by going into its block properties and changing the **Output Type** parameter. We see that its port turns orange, there is now a red arrow pointing to the **Throttle Block**, and in the Toolbar there is a red "-" button (red) that reads "View flow graph errors". Let's go ahead and click that.



We see that in the specified connection, there is size mismatch. This is due to our data type size mismatch. GNU Radio will not allow us to chain together blocks of different data sizes, so let's change the data type of all of our subsequent blocks. We can now generate and execute as before. We now see our sine wave on one channel.



Retrieved from "https://wiki.gnuradio.org/index.php?title=Guided_Tutorial_GRC&oldid=7786"

Category: Guided Tutorials

Tutorials > Guided Tutorials
