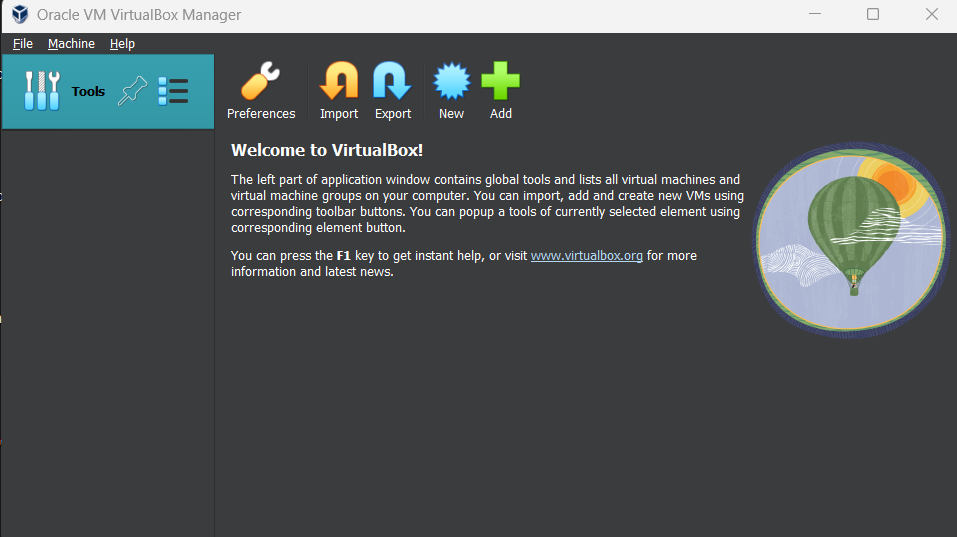
GNU Radio Lab

* **Description:**
  + This lab will explore the software GNU Radio and implement a flow graph that decodes FM signals and plays them. Quoting the GNU Radio projects repository “GNU Radio is a free & open-source software development toolkit that provides signal processing blocks to implement software radios. It can be used with readily-available, low-cost external RF hardware to create software-defined radios, or without hardware in a simulation-like environment. It is widely used in hobbyist, academic, and commercial environments to support both wireless communications research and real-world radio systems.” (<https://github.com/gnuradio/gnuradio>)
* **Objectives:**
  + Download and install Virtualbox so we can run a virtual computer that contains GNU Radio
  + Configure the virtual computer to use our software defined radio
  + Explore GNU Radio
  + Build an FM decoder and listen to a live FM radio station
* **Materials:**
  + The frequency of a Radio station in the area the lab will be held at. (i.e. 92.9)
  + Windows computer with at least 8 CPU cores, 8 GB of ram, and 11 GB of storage
  + Nooelec NESDR mini + antenna
  + Internet connection or USB containing the following files
    - https://download.virtualbox.org/virtualbox/7.0.14/VirtualBox-7.0.14-161095-Win.exe
    - https://download.virtualbox.org/virtualbox/7.0.14/Oracle\_VM\_VirtualBox\_Extension\_Pack-7.0.14.vbox-extpack
    - https://www.gnuradio.org/releases/instant-gnuradio-3.8.2.0-1.0.0.ova

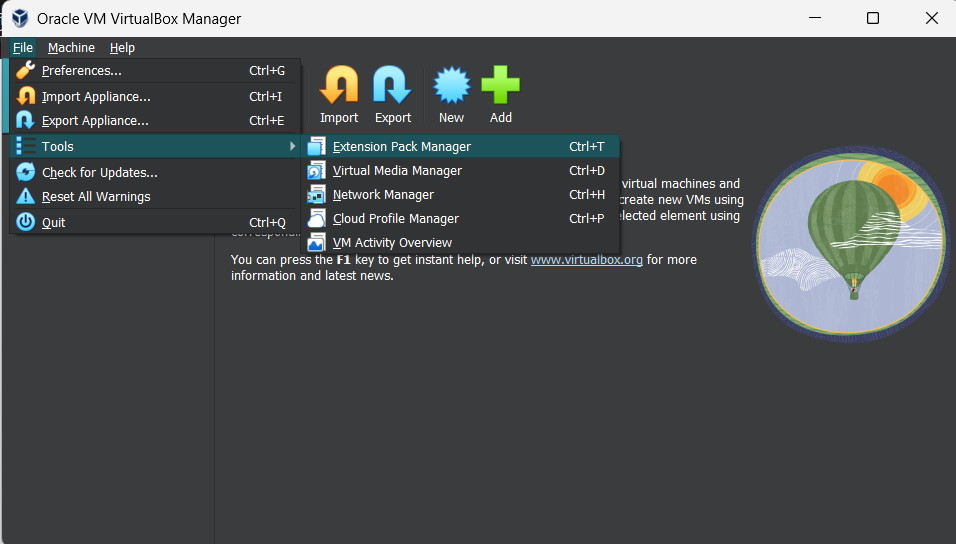
**Setup:**

We first need to configure a virtual machine that contains GNU Radio for use to use. There are several ways of installing GNU Radio but in this lab, we will be using the premade virtual machine approach. This virtual machine has GNU Radio already installed and configured so it can be used right out of the box. It also has contains extra tools that while we won’t be using in this lab, they are very useful in further exploring SDR’s.

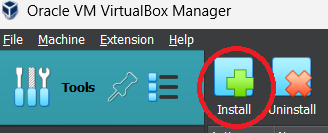
1. Download the 3 files from the links in the material section or copy them off the USB drive the instructor has.
2. Plug in you Nooelec NESDR mini + antenna to the computer.
3. Launch the executable called VirtualBox-7.0.14-161095-Win.exe and follow the install prompts.
4. Once the installation process of virtual box is complete, launch the program. A window that looks like the image below should appear.



1. Next, we need to install the extension package for Virtualbox. To do this go to File → Tools → Extension Pack Manager.

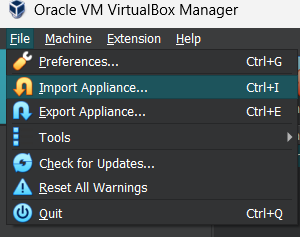


Then click on the install button that has the green + icon that says install.

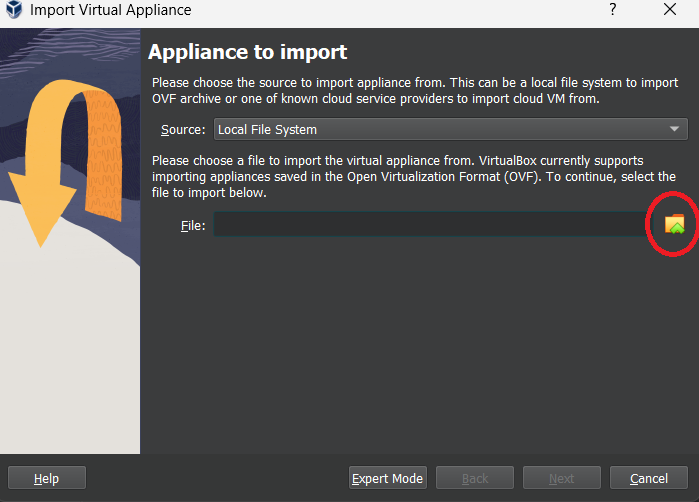


Then select the extension pack that you downloaded or copied and open it to install.

1. Now it's time to create the virtual machine. Go to File → Import Appliance



In the pop-up window that opens, click the folder icon and select the ova file that you downloaded or copied over.

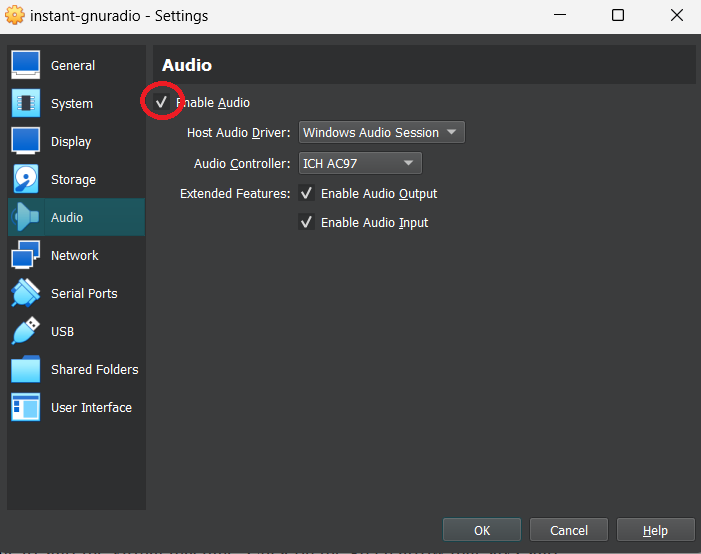


Then click the next button and then the finish button.

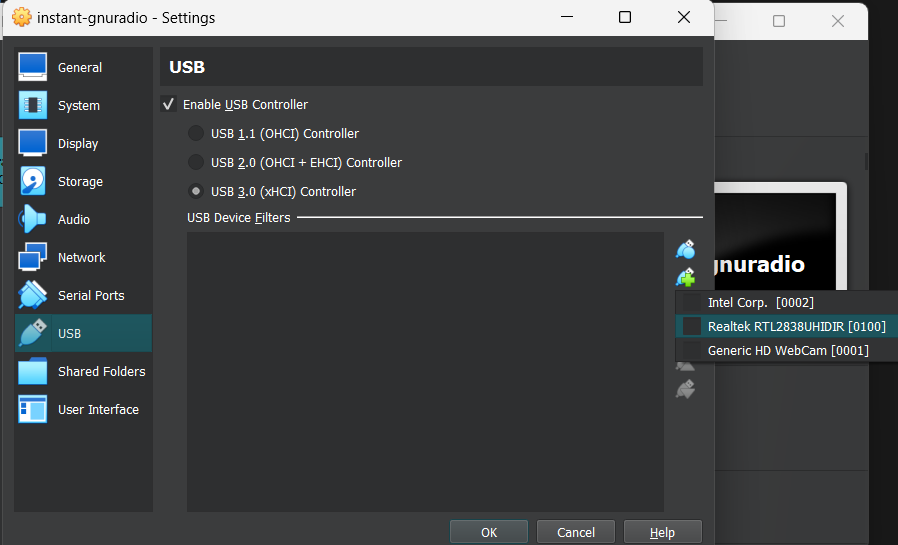
1. Now that the virtual machine has been created, there are some settings that need to be changed. Click on the virtual machine and then click on the settings icon.



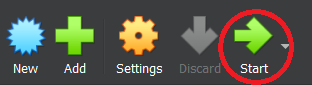
The settings that need changing are first in the Audio section, all that needs changing is check marking the enable box.



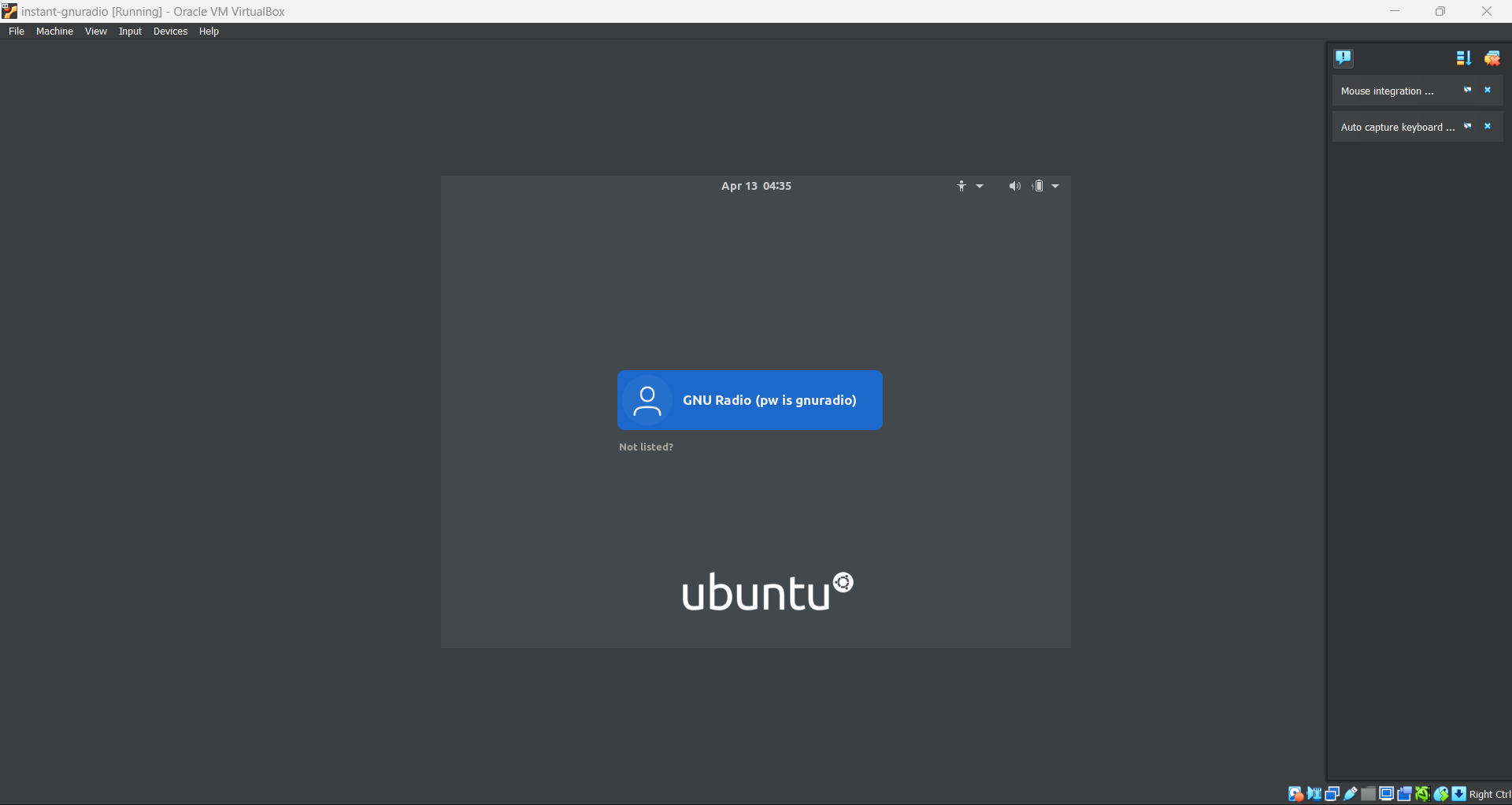
The final change is in the USB section. Click on the green + symbol and add the device that says “Realtek RTL”. Its name is longer than that, but “Realtek RTL” should be enough to identify the correct one. Once done click OK at the bottom to save the changes.



1. Now it time to start the virtual machine. Click on the green arrow that says start.



1. One the virtual machine has booted it should look like this.



Click on the user icon and type in the password that is “gnuradio” (it also lists the password in the username field)

1. You should now be on the desktop of the virtual machine. The screen will probably be too small to comfortably use so we will enlarge it. In the top tool bar of virtual box go to View → Virtual Screen 1 and select 1600x900. Now the screen should be large enough for us to work with.

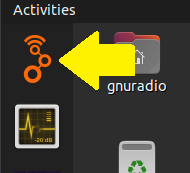
Note:

If once you launch the virtual machine, and only a black screen appears and no log in screen. Close out the virtual machine and restart it.

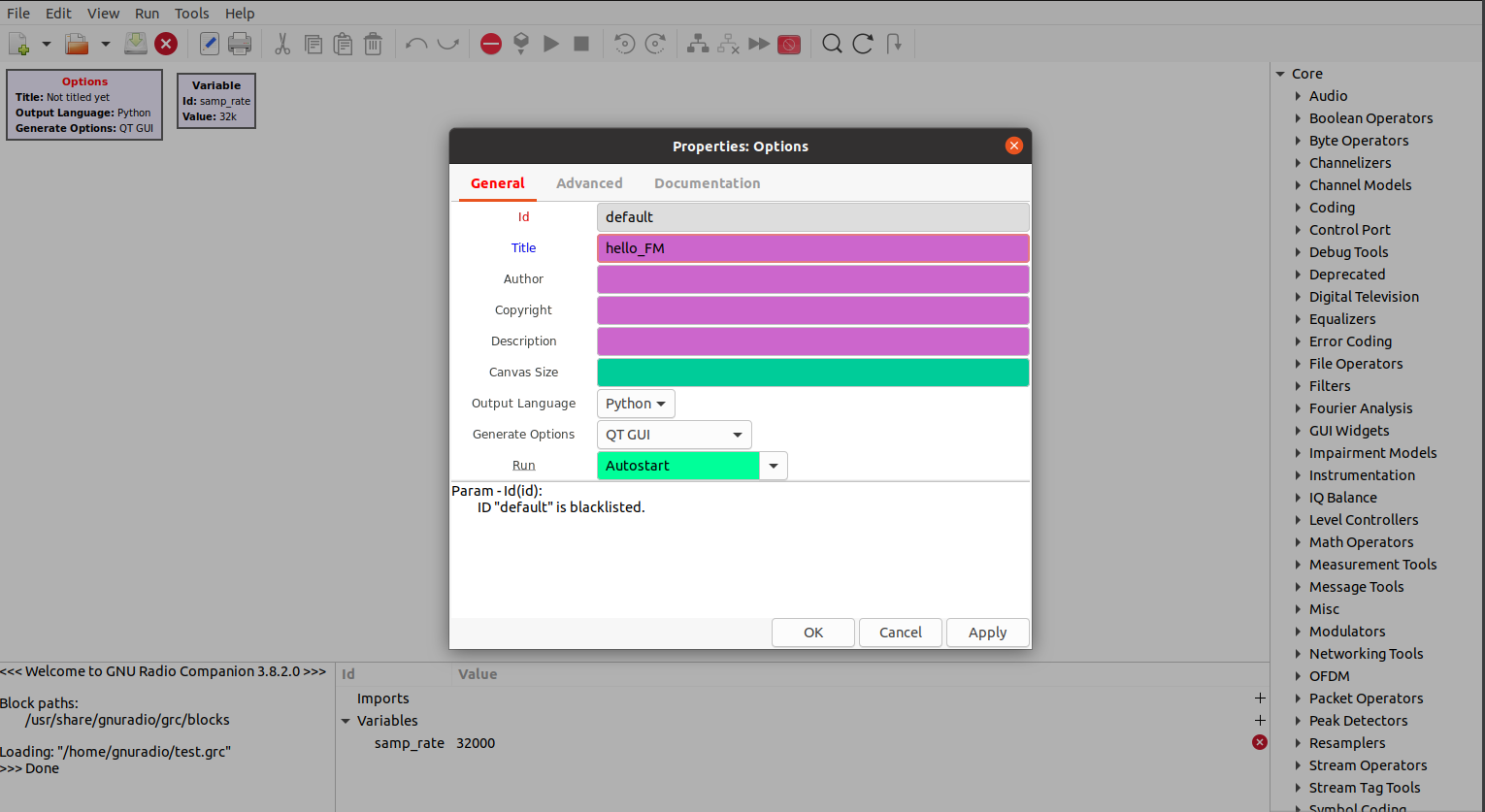
**GNU Radio:**

With the virtual machine now ready we can get to the SDR side of this lab and explore GNU Radio and make an FM decoder.

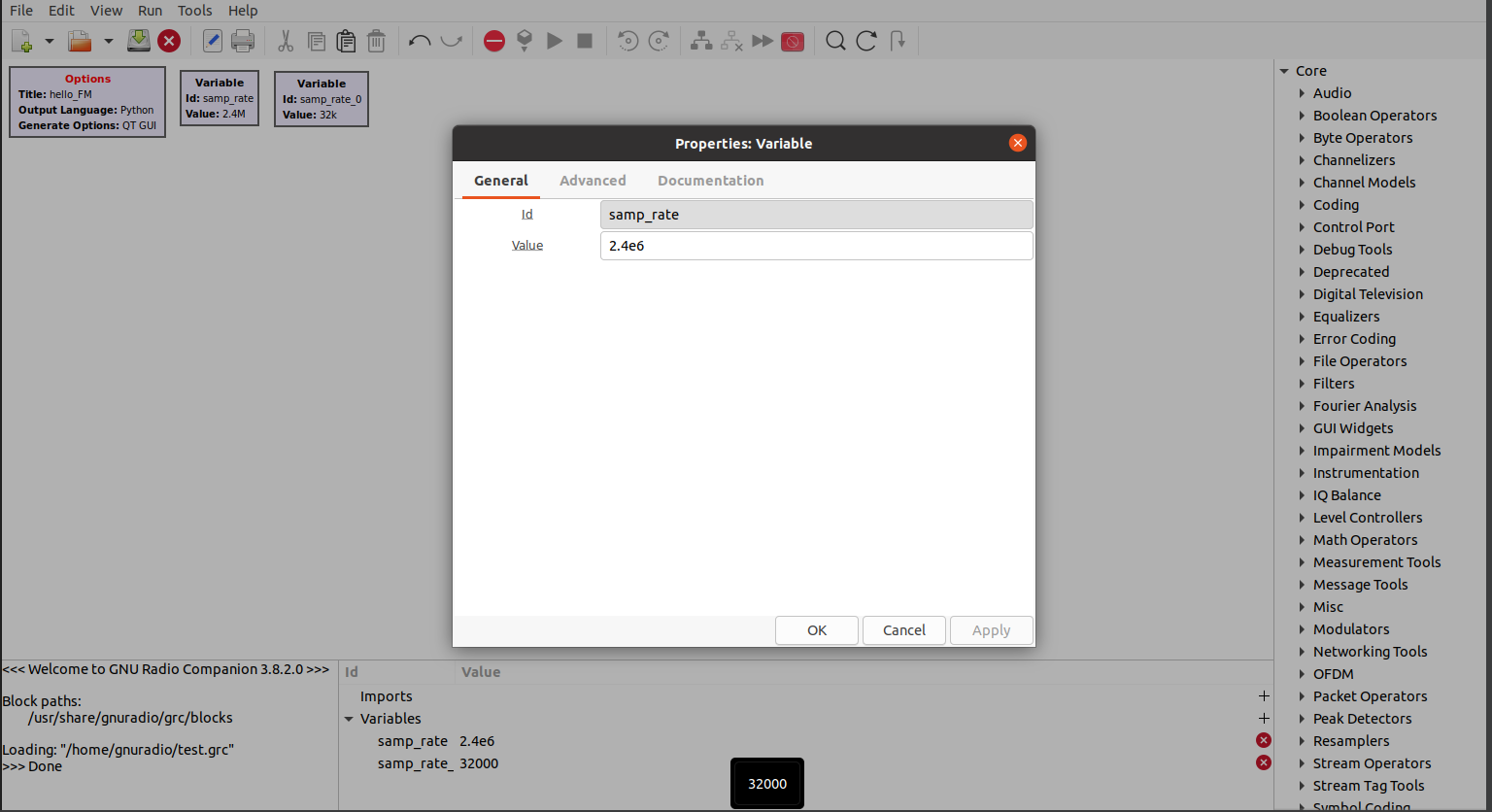
1. Click on the GNU Radio icon in the top left to launch the software.



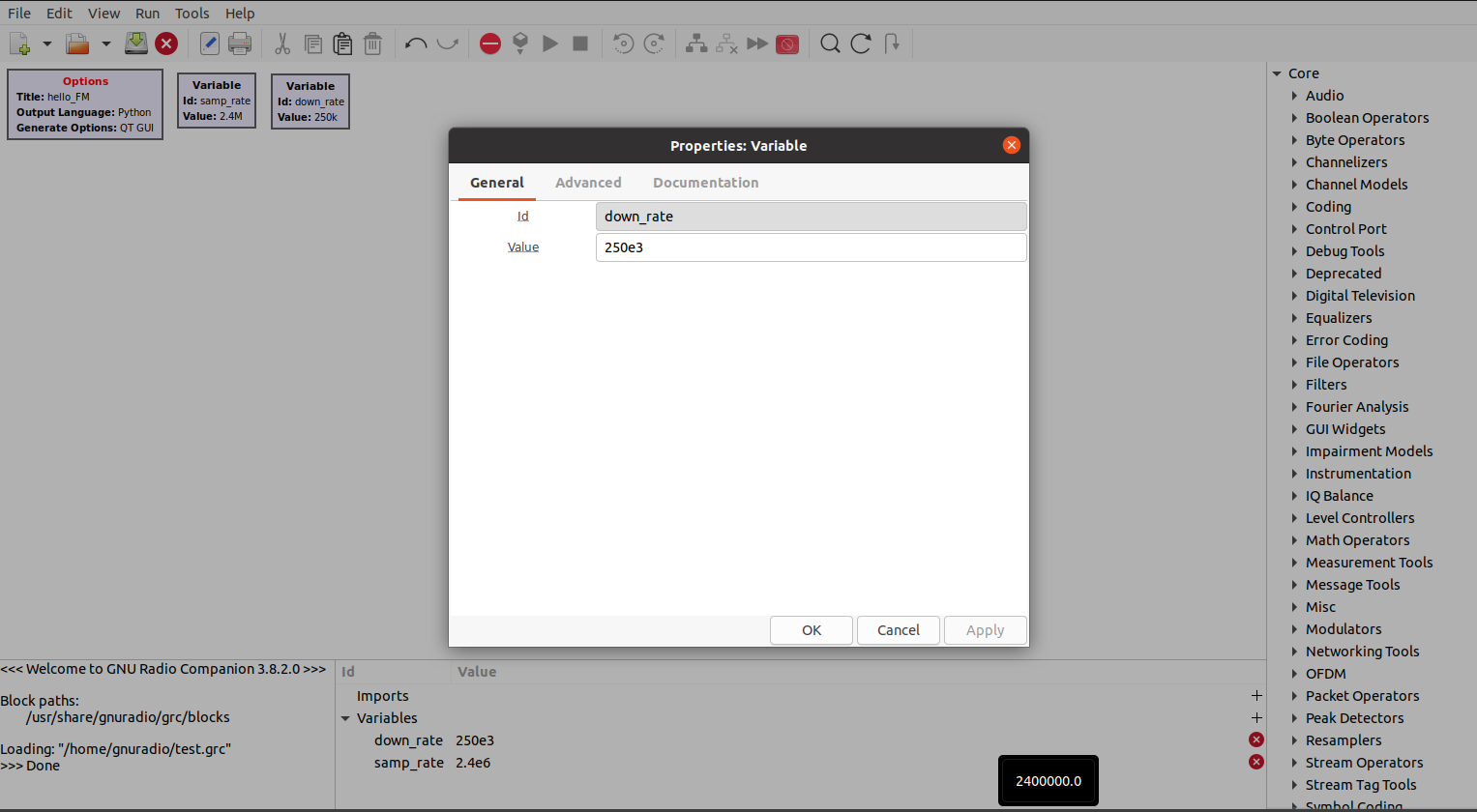
1. The program that opens is the starting screen for your GNU Radio “flow chart”. These “flow chart” are like a canvas for you to drag and drop different building blocks from the tool area on the right. This allows you to build out many different SDR designs using different filters and decoding methods without needing to create code. GNU Radio will do that for you when you build the design. Each block represents a python code block that you can go in and edit or create brand new ones yourself. For this lab we will just use the built-in blocks.
2. There are 2 blocks already on the flow chart by default. They are “Options” and “Variable”. Double click on the “Options” box and in the new pop-up window update the title section with what you want to name your flow chart. For this lab instructions, hello\_FM will be used. Click apply and then ok to save changes.



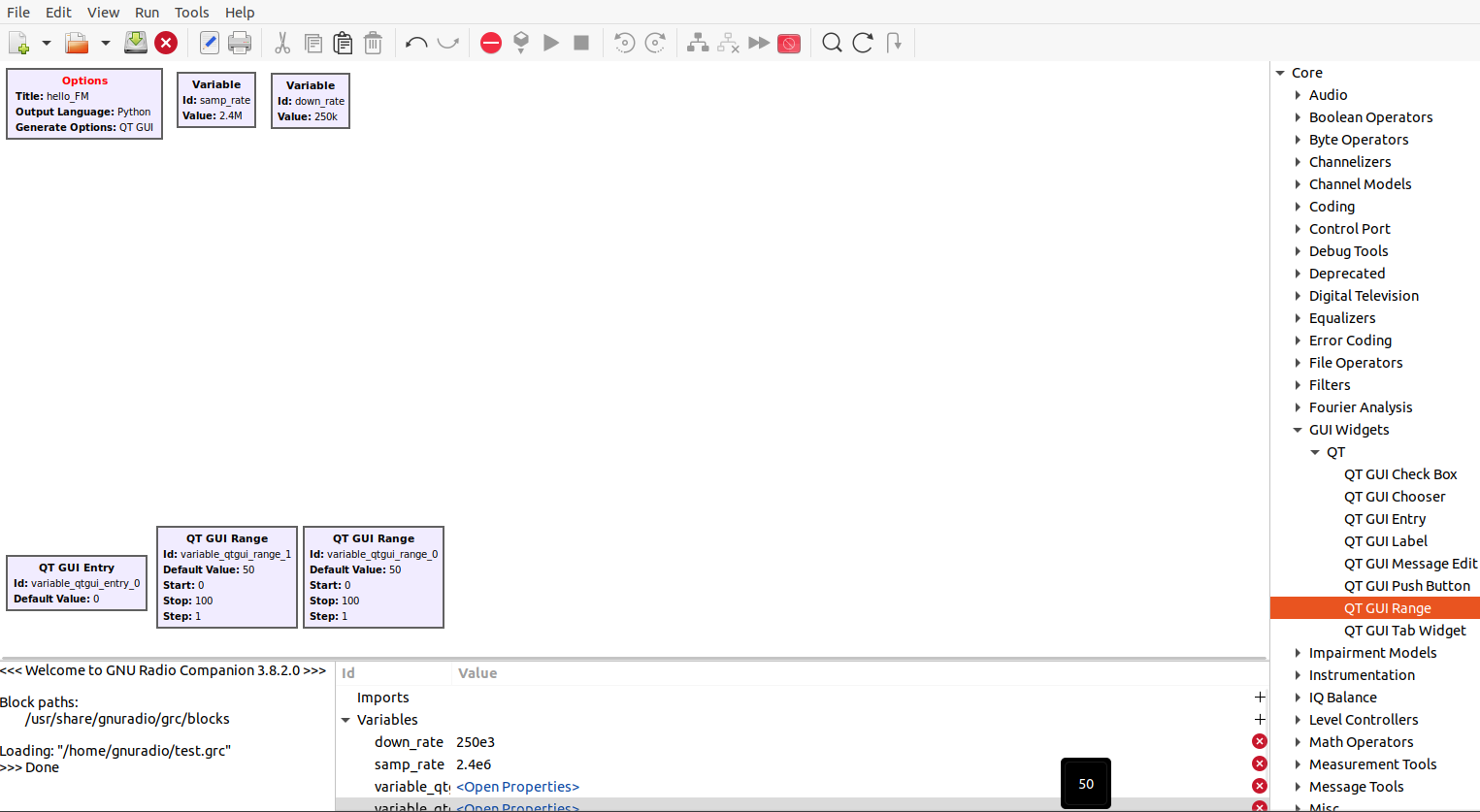
1. Now we will need two variable boxes. Variable boxes allow us to define variables and their values in one spot and just type the variable name in other blocks. This allows us to change variable values in one spot instead of having to change them in every block they are used in. Right click on the variable box and copy it. Then right click on an empty space on the flow chart and select paste. There should now be two variable boxes. Double click the box that says samp\_rate and change the value to 2.4e6. This value will be used to define the sample rate of the SDR which will be 2.4MHz. Hit apply and then ok to save the changes. You can drag the blocks around the flow chart to better organize the block and feel free to do so.



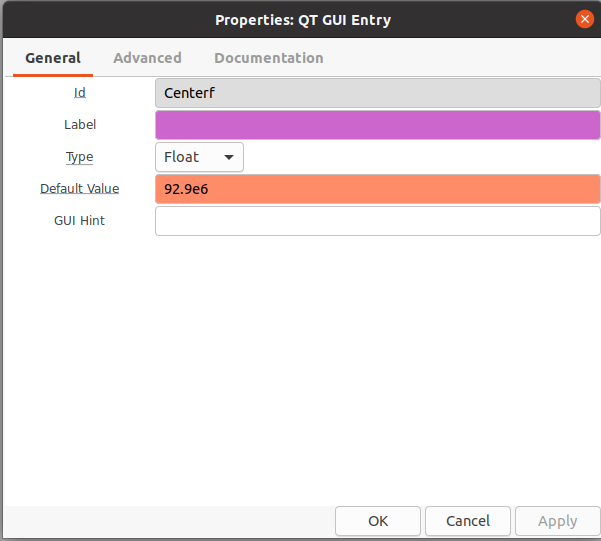
1. Double click on the other variable box. We will need to change both the ID and the value. For the ID change it to down\_rate. And for the value change it to 250e3. We are defining another variable that will be used to down sample to signal to ease processing impairments but without losing the needed information we need. Hit apply and then ok to save the changes.

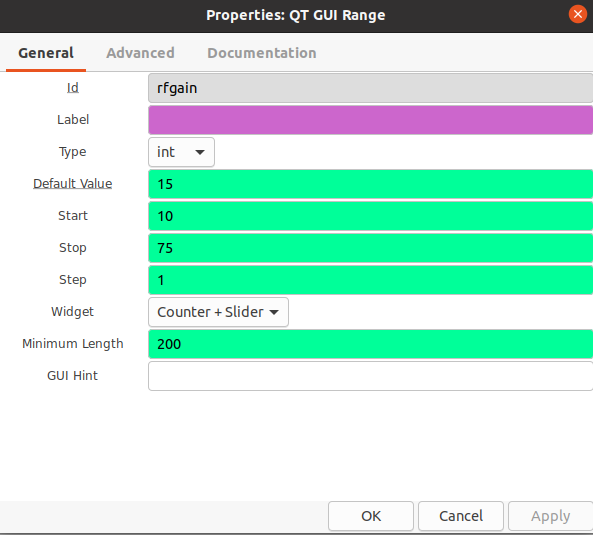


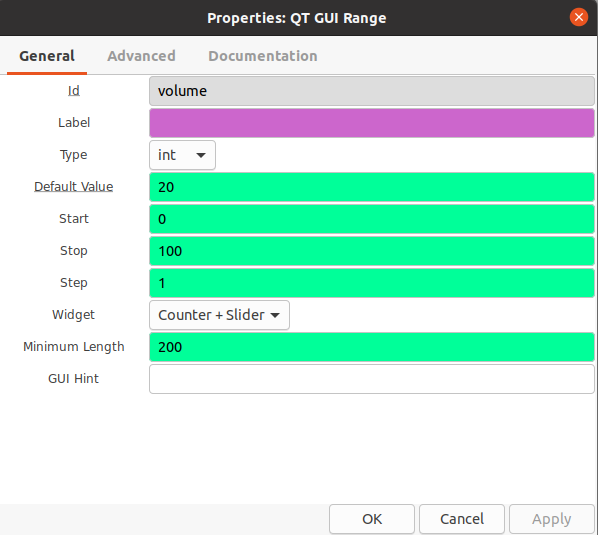
1. It's now time to start pulling new blocks from the toolbox. First go to the Core → GUI Widget→QT section. To add a new block to the flow chart we will double click on an entry to do so. Double click on the “QT GUI Entry” to place it on the flow chart. We will do this again for “QT GUI Range” but this time we need two of them. You should now have 2 “QT GUI Range” blocks and 1 “QT GUI Entry” block. Your flow chart should now look like the following.



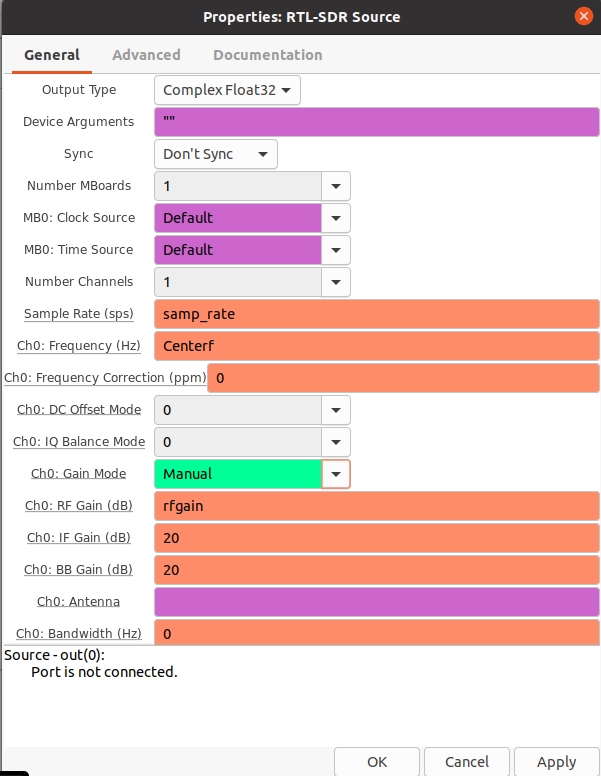
1. We need to now configure these blocks. These blocks give us graphical controls that we can use when the flow chart is running. We are creating the ability for us to change the rf gain, the volume, and the center frequency. When launching this flow chart instead of having to recompile every time we want to change a value. Like before we will double click each one to change settings and then click apply and ok to save. Match the settings in the screenshots below.





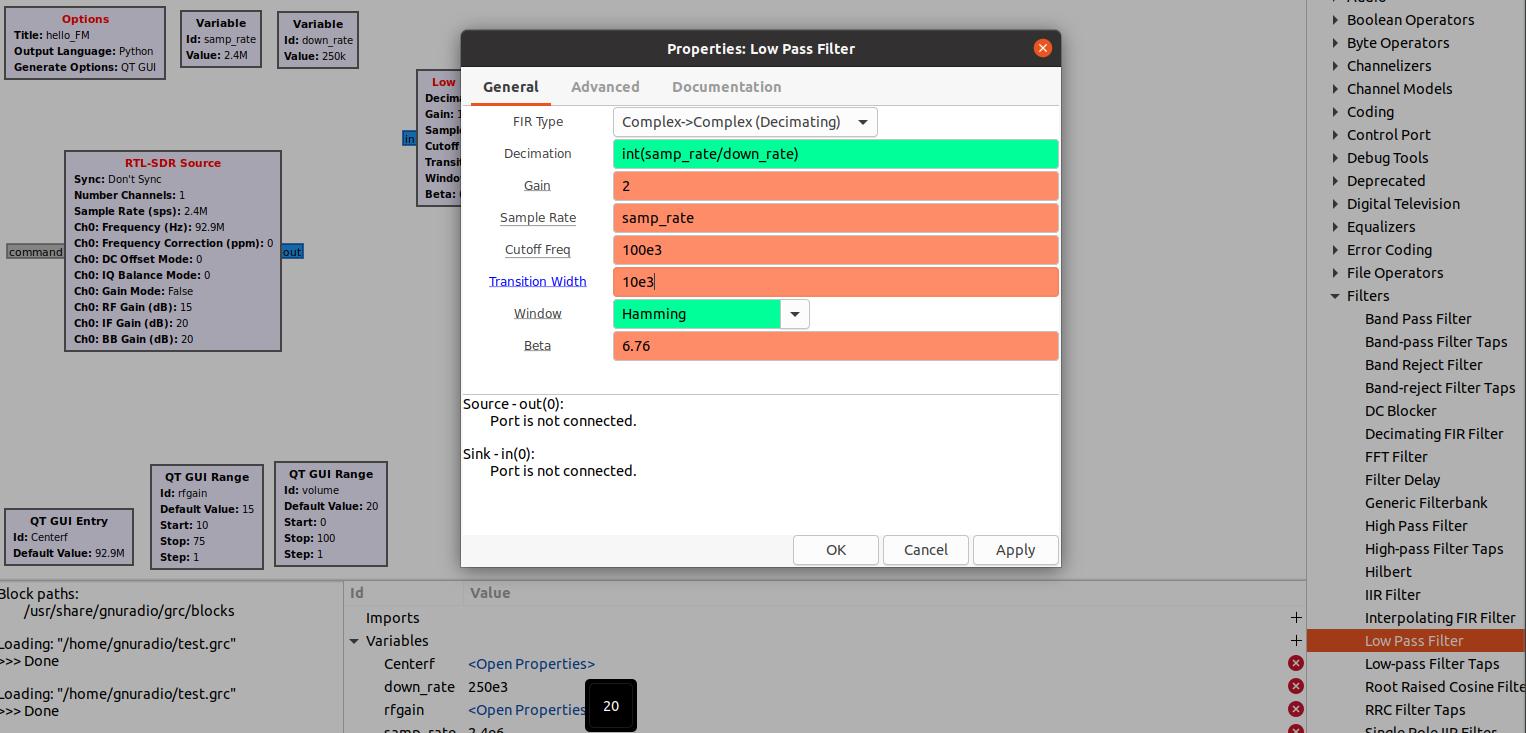


1. Time to place the SDR source block. This block represents the USB SDR we have plugged in and will output the data as complex numbers. In the OsmoSDR section, look for a block called “RTL-SDR Source”. Double click on that entry to add it to the flow chart
2. We are now going to configure it. Double click on it and change the settings to match the image below. Then click apply and ok to save.

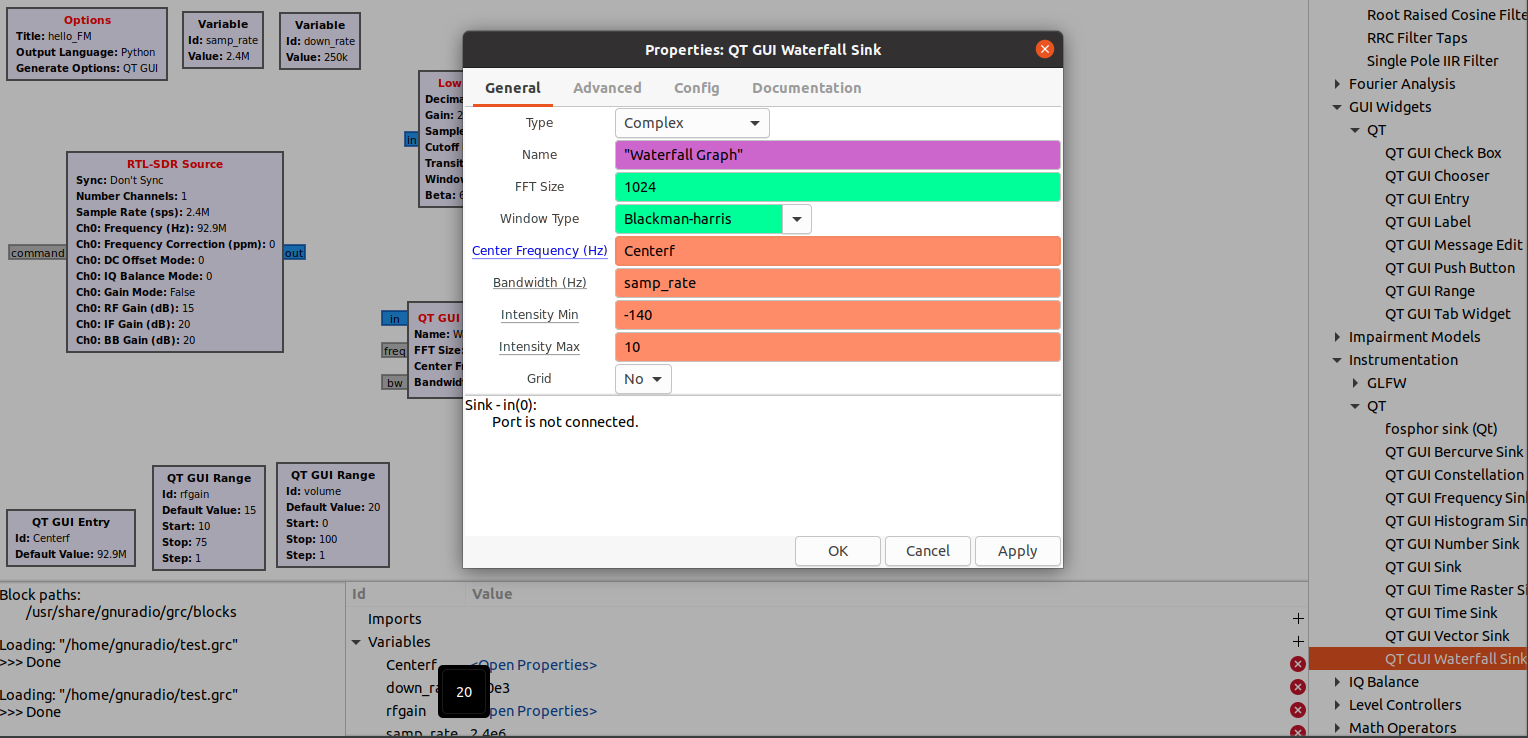


We are using the names of the blocks we created earlier and putting them in the parameter fields here. This way when we use the graphical elements later to change settings, those settings will be placed in these fields.

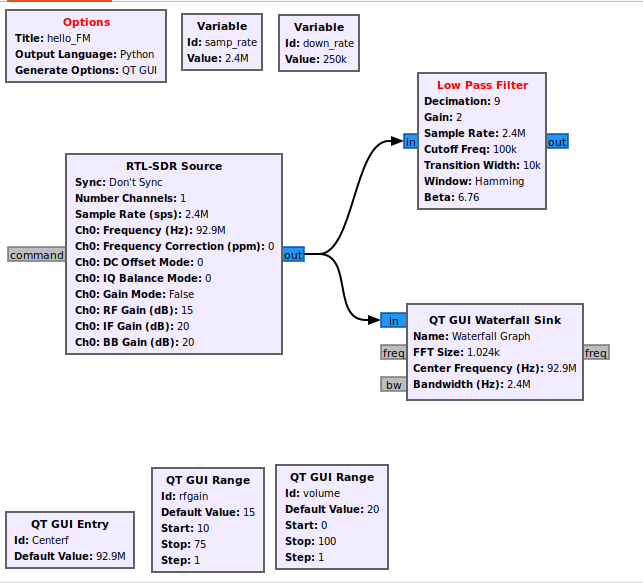
1. With the RTL-SDR Source now configured, time for the next block in the chain to be added. We will be adding two blocks to the output of the RTL-SDR Source. The first of these two is called the “Low Pass Filter”. This block can be found in the Core → Filters section. A low pass filter is a filter, like the name suggests, that allows or passes signals through that the low frequency signals while blocking the higher frequency signals. We need this filter to clear up the signal, so we don’t hear a ton of static later when we play the FM station. Configure the low pass filter like in the screen shot below.



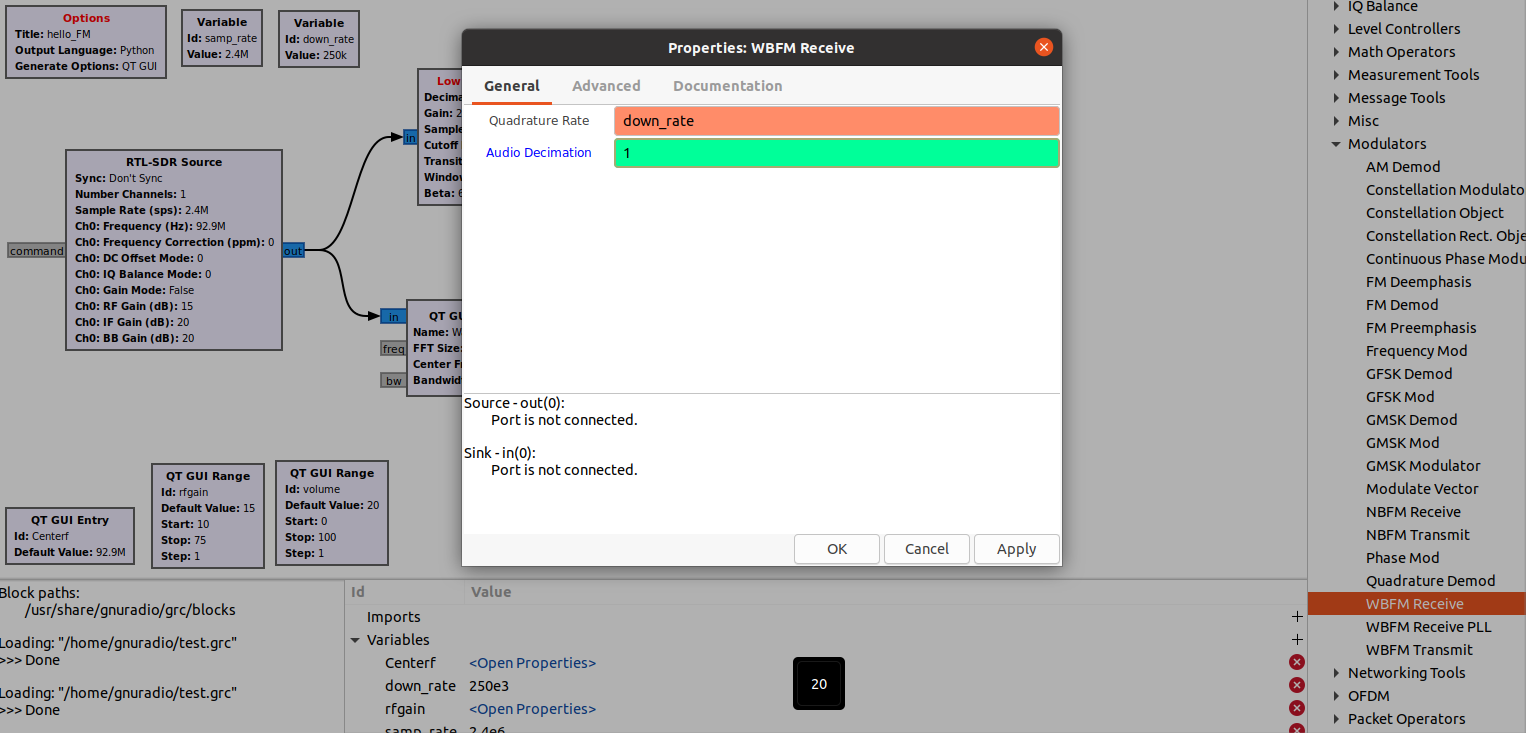
The second block we will be adding to the output of the RTL-SDR Source is called the “QT GUI Waterfall Sink”. This block can be found I the Core → Instrumentation → QT section. This block will just show us the output of the SDR in a way that we can understand it. Configure this block to match the settings in the image below.



1. Now connect the output of the RTL-SDR Source to the inputs of the blocks we just created by clicking and holding the output of the RTL-SDR Source and dragging the curser to the input of each block. You will know it’s working if you see a line with an arrow coming out of the RTL-SDR Source output when you are dragging your mouse to the input of the other two blocks. Your flow chart should look something like this now.

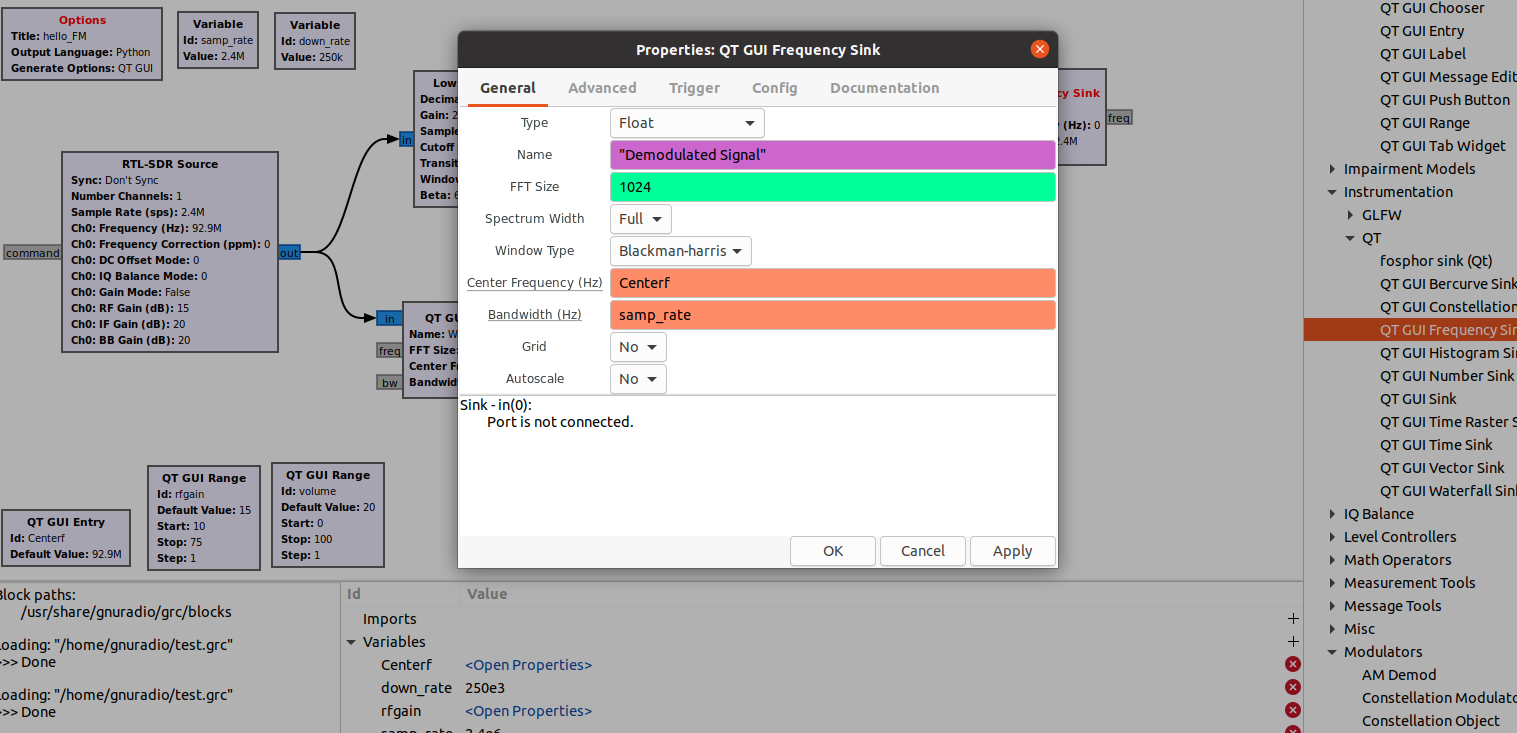


1. Ok, so now we have a cleaner signal coming out of the low pass filter but it's still something we cannot use as it's still encoded. This is where “WBFM Receiver” comes in. WBFM stands for WideBand Frequency Modulation. This is the type of modulation FM radio stations use to broadcast the signals over the air waves. To use this receiver, look in the Core → Modulators section. The set the settings to match the image below. Then connect the output of the low pass filter to the WBFM Receiver’s input.

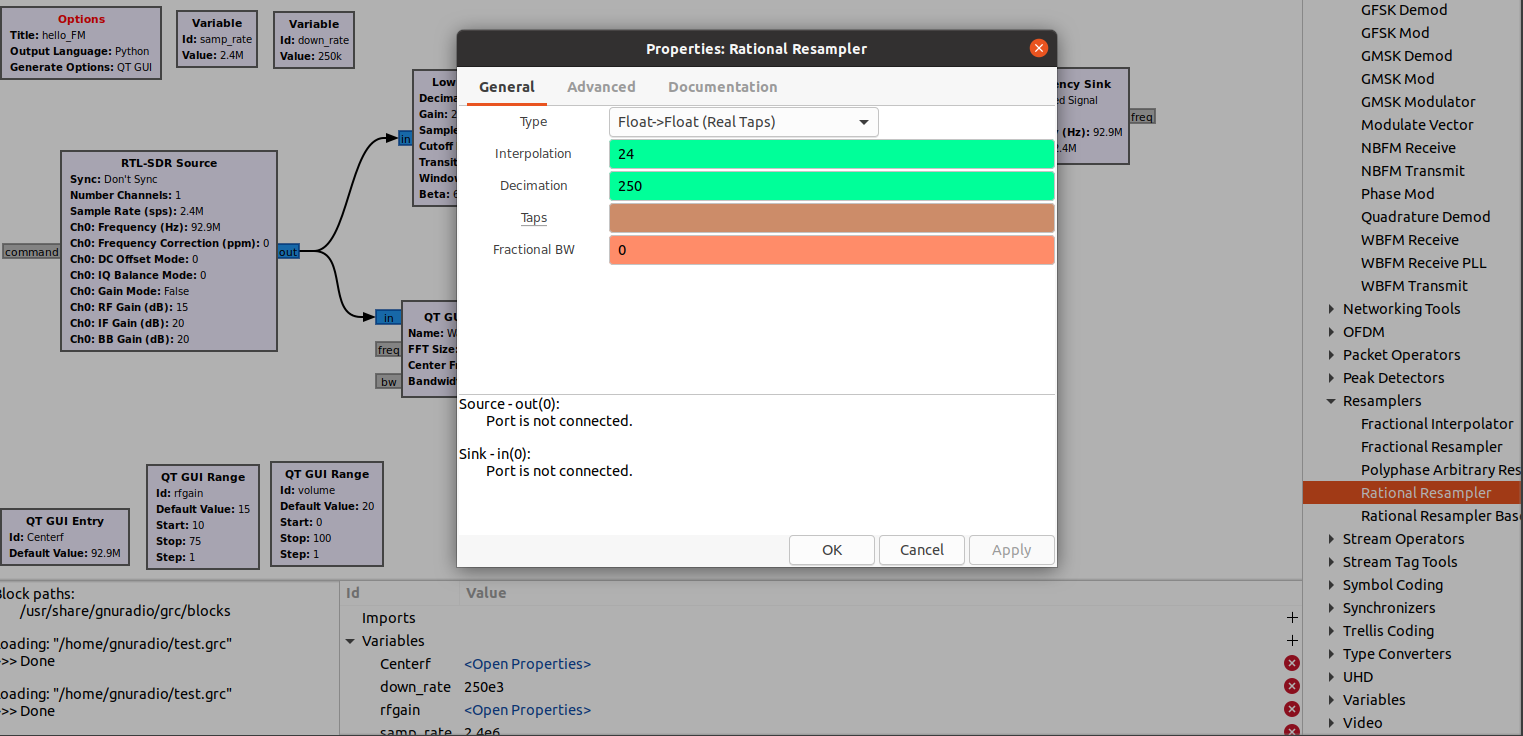


Notice the output of the WBFM Receiver is a different color than the input. That is GNU Radio showing you that the output is a different type of numbers called real numbers while the input was a type of number called complex numbers.

1. For the output of the WBFM Receiver we will be putting two blocks. One will be in the Core → GUI Widgets → QT and is called “QT GUI Frequency Sink”. This is another graphical element that we will use to see the signal that is coming out of the WBFM Receiver. Match the settings to the ones in the image below.

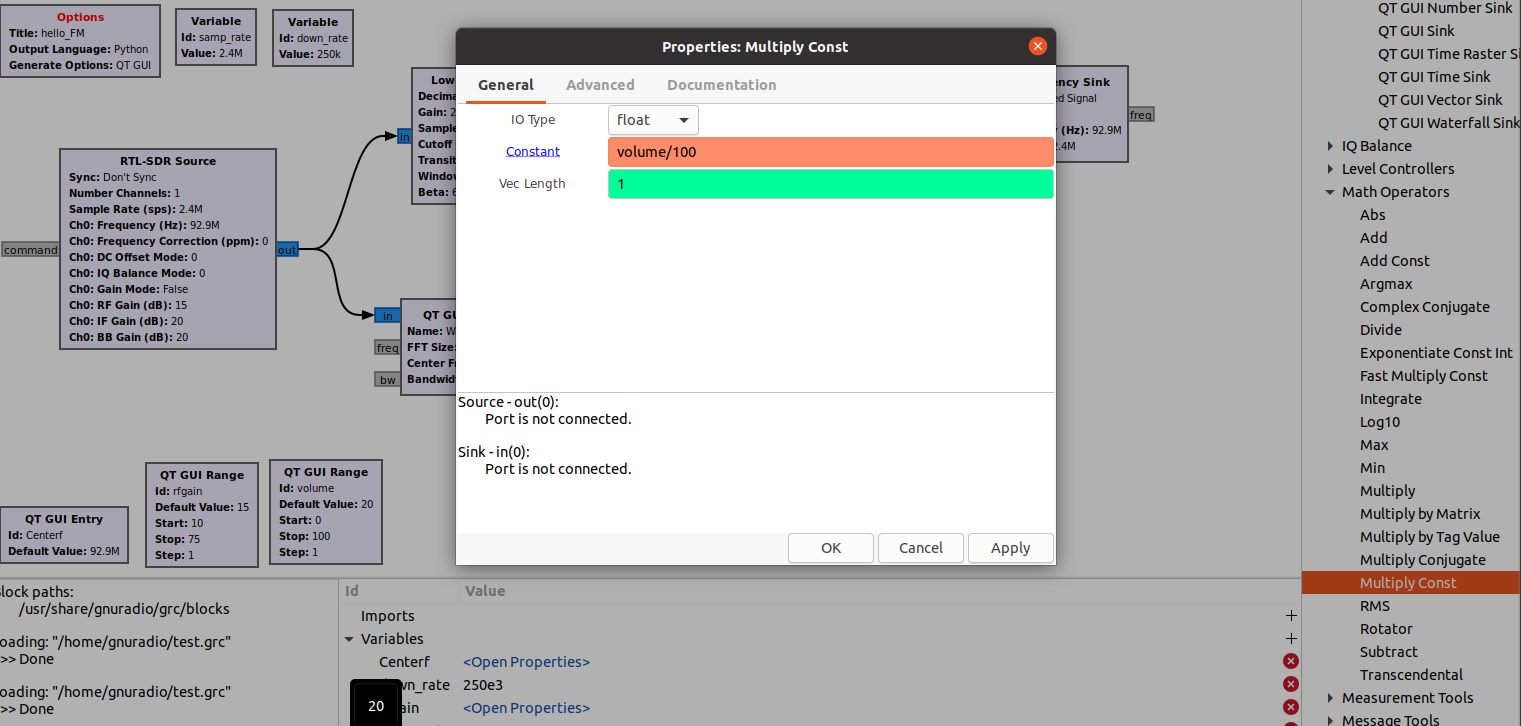


The other block we will need is the “Rational Resample”. It can be found in the Core → Resamplers. This block will take the input signal and in this case down samples it to a lower sample rate. We are doing this to lessen the computation burden as we just want the audio in the signal and we are going for perfect, we just want to be able understand what is being said clearly in the FM transmission. Match the settings to the ones in the image below.



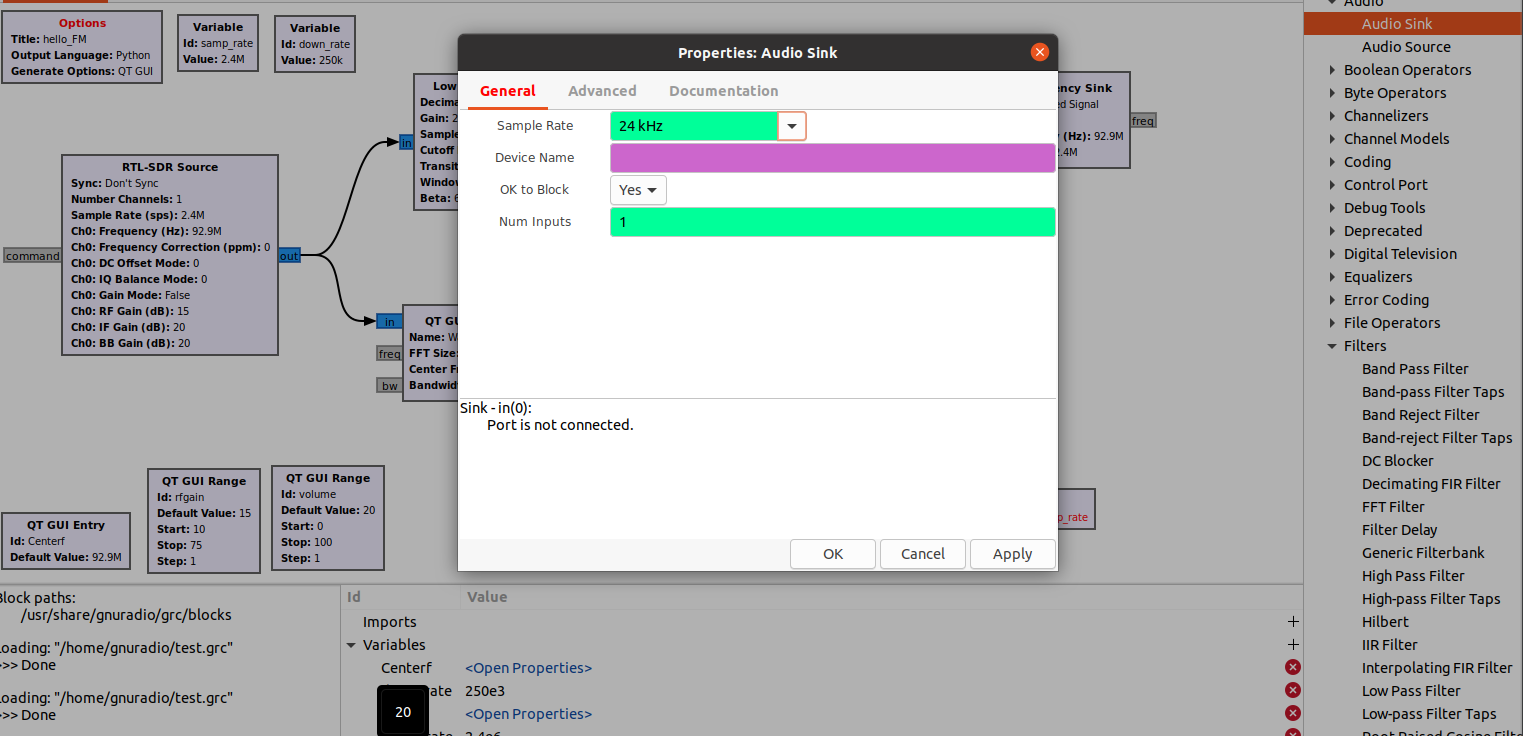
Then connect the output of the WBFM Receiver with the two blocks you just configured.

1. We could now just output this decoded signal out to the speakers, but we will not have control on the volume of the sound. To give us this ability we will be using the “Multiply Const” block from the Core → Math Operators. This block is very simple in that it just takes the input and multiplies it by a constant number. Combine this with the graphical block we created earlier we now can control how loud the audio will be when we listen to the FM signal. Match the settings to the ones in the image below.



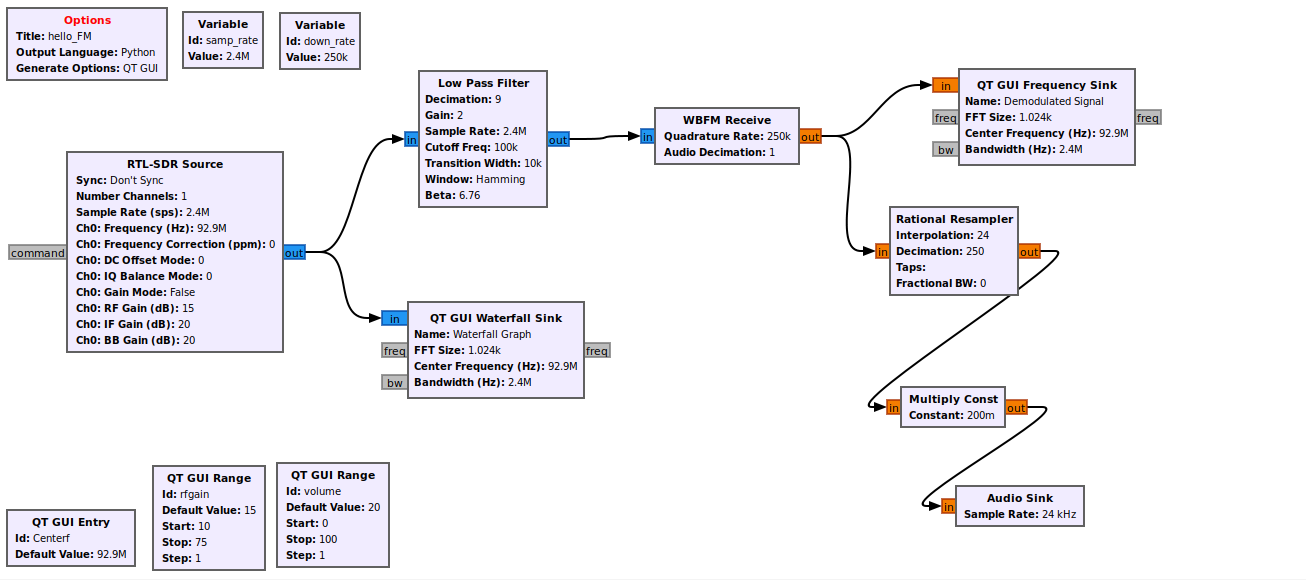
Then connect the output of the Rational Resample block with the input of the Multiply Const.

1. Now the signal is ready to be listen too but currently the flow chart has no way of getting that audio out. To do that we need to add an “Audio Sink”. It can be found in the Core → Audio section. This block will sink the signal we are pulling from the air wave, decoding it, and then finally playing it out of our computer's speakers. Match the settings to the ones in the image below.



Then connect the output of the Multiply Const block with the input of the Audio Sink block.

1. Your flow chart should look like this.



If not, please double check your flow chart with the lab procedures.

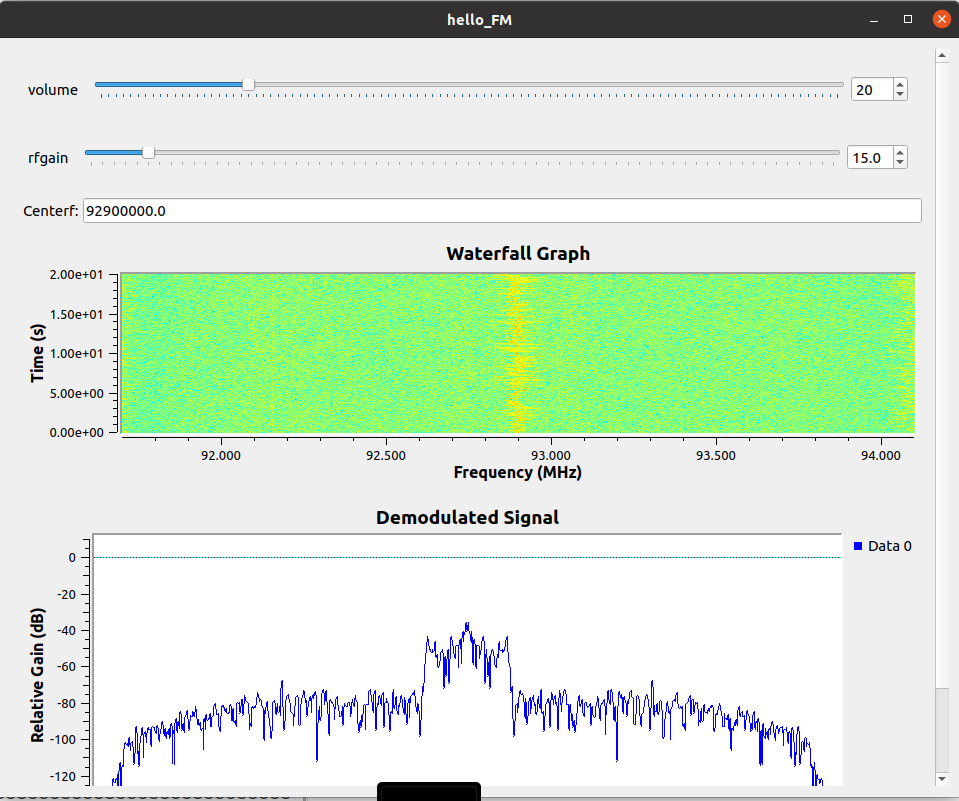
1. With the flow chart built, we need to do a quick save. To do this click on the button circled in the image below.



1. Now it's time for the magic…. actually, it's science but it is magical. Click on the button circled in the image below.

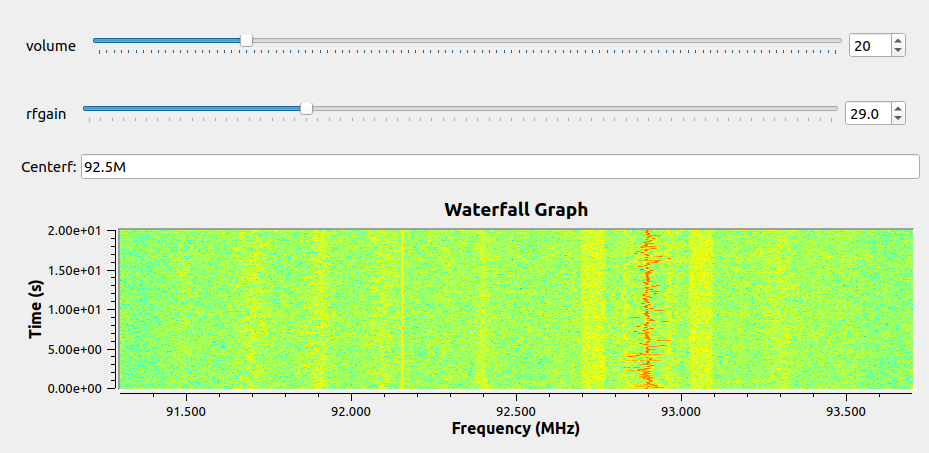


This button will tell GNU Radio to take our flow chart and generate the code for it. It will then run, and a pop-up window should appear like the one below. An error window may pop up. You can just click ok as that error does not concern us. The pop-up window may also open up small. You can resize it to the sides you desire.



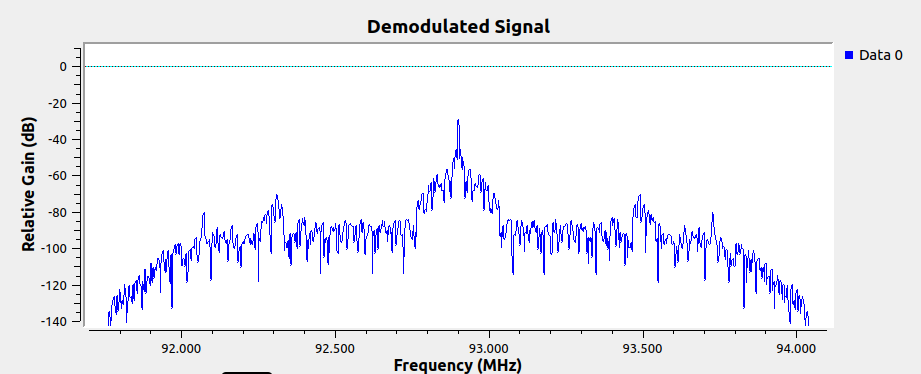
You will probably hear just static at this point as it is currently tuned to 92.9 MHz. Explaining the different things in the pop-up window

* **Volume** is self-explainable. It controls how loud the playback of the FM station is.
* **Rfgain** controls the how much gain the SDR should use. The more gain there, the clearer faint signals become but too much gain and even static will start looking like signals. So, a balance needs to be made. Try finding that balance.
* **Centerf** is a number field that we can tell the SDR to change frequencies too. This is where you put the frequency of the radio station you want to tune to. Say you want to tune to 99.7. You will type 99.7M into this box and hit enter. It will then cause the SDR to tune to that frequency as its center.
* The **Waterfall Graph** is used extensively in SDR work to express signals. You can use this graph to see what is in the spectrum around the frequency you tuned too. Take for example if I tuned to 92.5M. As shown in the image below, there is no signal here, but there are two signals nearby.



The signal at 92.15MHz is a data stream signal in the FM range. While we can see it thanks to the waterfall graph, our GNU Radio flow chart is not built to decode it and even if we tune to it, we will get no audio. Now on the other hand, at 92.9MHz, that is an FM radio station. We can tell by the signal being wider and varying in frequency (that’s why it called FM radio, Frequency Modulated). If I turned by typing in 92.9M in the Centerf field, I would then begin to hear the radio station.

* The line graph at the bottom shows what the signal looks like after we demodulate it. As can be seen in the image below, if you tune to a radio station you will see a spike right at the frequency you tuned to with the rest of the line down at -70 and below. This means the signal is coming in clear and we can separate the signal from the noise that exists in the atmosphere.



note: if the audio suddenly stops, close the pop-up window and relaunch your flow chart. The flow chart had an internal error and that can be fixed with a simple restart of the flow chart.

**Conclusion:**

You should now be able to tune to different radio stations and listen to them. In this lab you were exposed to GNU Radio, a tool used extensively in the SDR world, and built a functional flow chart that used an SDR, decoded an FM signal, and listened to it. This lab was a very high-level overview of what GNU Radio can do with several parts just glossed over. GNU Radio and SDR’s are capable of significantly more stuff, and you are encouraged to investigate the different applications of both GNU Radio and SDR’s.