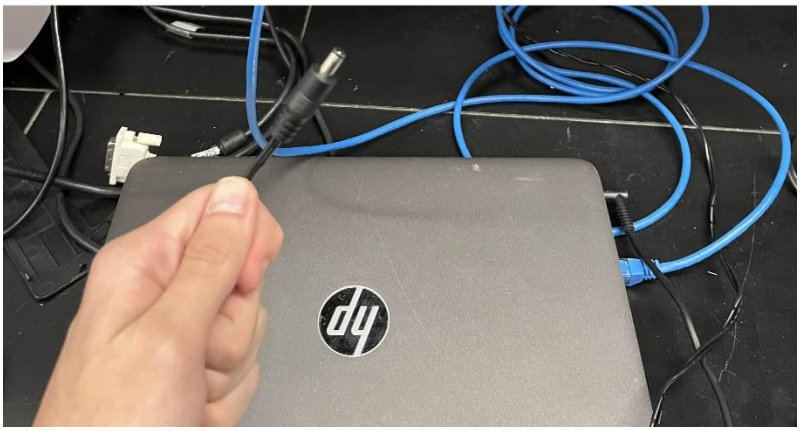
Spring ’25 Introduction to GnuRadio (Last Update: 1/29/25)

Things You Will Need to Get Started

To start working with the SDR available to the club, please contact jay.williams@cooper.edu to obtain access to our lab in room 708, as well as the password to the laptop connected to the SDR. Feel free to ask any associated questions as well.



The above top picture shows the SDR on top, with WiFi antennas on both ports. You will find tape on top of the SDR indicating what kind of port they are, either TX/RX or RX2. The cable shown in the below picture is the power supply for the SDR, to turn on the SDR, plug it in and give it a minute to boot up. When it is ready, lights D and F should be on. The laptop is also shown in the bottom picture. It is connected to the SDR via ethernet. To turn the SDR off, make sure all SDR-associated programs (like GnuRadio, GQRX) are shut down, and unplug the device.

Connecting the Laptop to the SDR

A more detailed description of this step, along with some troubleshooting information, can be found on the ground station github[1] under Documentation -> Spring 24 -> SDR setup docs.

On the laptop, open a terminal in Documents -> SDR utilities, run the command:

```
sudo sh connectToSDR.sh
```

And enter the laptop’s password when prompted.

[1] https://github.com/JayTWilliams/GroundStation

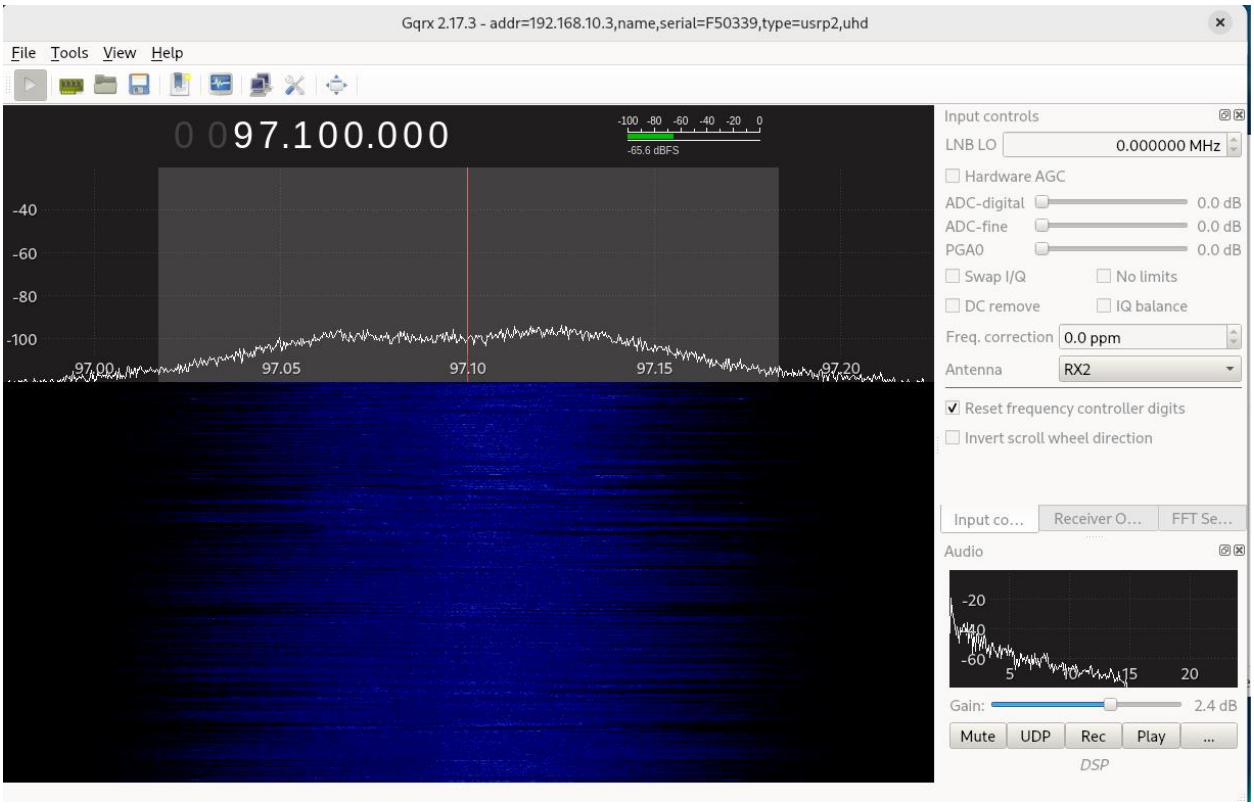
Lab #1: Exploring the Spectrum With GQRX

Throughout your learning process, you will be creating flowgraph files and collecting data, so please create a working directory for yourself in the “Individual Workspaces” folder under Documents.

Before looking at GnuRadio, a fun way to explore the radio spectrum is through another software, GQRX. To open it, press the windows key and search for GQRX. It should automatically detect the SDR. To start receiving, press the gray play button on the top left of the window. You can tune to a specific frequency by hovering the cursor over the large number on the top left (in units of megahertz MHz) and typing in a new number.

The frequency 101.1 is usually active, but when tuning to it, it is unlikely that you will hear anything using the little WiFi antennas already connected. They aren’t designed for this frequency but work better at 2400 (2.4 GHz). But an interesting experiment is to grab, with your hand, the receiving antenna[2], and magically, the signal comes through[3].

An interesting band to look at is the 2400 to 2500 band, which is an unrestricted ISM (Industrial, scientific, and medical) band, that has a lot of activity. Bluetooth devices can be seen transmitting around 2400-2480 but can be hard to find as they hop around the spectrum very often.



[2] To set which antenna is receiving,
[3] Can you explain why?

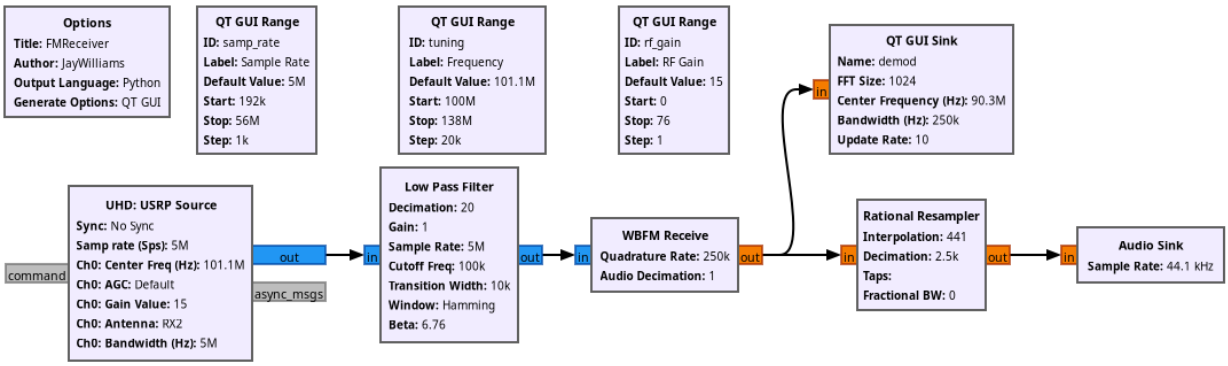
A view of GQRX is visible above. It is not tuned to 101.1 but instead tuned to another broadband FM radio frequency. The selected antenna is RX2.

It is also possible to record audio data using GQRX. This is useful for storing transmitted data, as some transmission formats pack their data into the 20kHz audible spectrum (such as NOAA weather satellite data[4]). At the bottom of the GQRX window, there is a recording button. This will save wav files to the laptop, which can be played back as regular audio. Before recording, set the location to save the files to your working directory. Click the (…) in the bottom right, go to the recording tab, and change the saving directory to your own. These files are saved with a date and time in the name.

Lab #2: Doing the Same Thing, But This Time with GnuRadio

GnuRadio is open-source software that helps interface with software-defined radios. While one can write python code using the libraries, the software was ultimately designed to work by connecting blocks to create flowgraphs. Custom blocks can be written in python or C++, but that is a much more advanced task.

A rather convenient video[5] exists to help build an FM receiver in GnuRadio, specifically to USRP devices, of which we own the USRP N200 SDR. This video uses an outdated version of the software, but in general still applies. Several blocks used in this video no longer exist, but alternatives do.



To ease the pain of finding the alternatives to the blocks, provided above is a complete flowgraph for listening to FM broadband radio, which is largely faithful to the Ettus tutorial. These blocks can be found by clicking the magnifying glass on the top of the window and searching by name. Some important parameters are missing from the above image, but they will be described.

The USRP Source block at the start of the chain is what collects signal data from the SDR. You may tune to any broadband FM station of your choice, but 101.1 is one that is consistently active.

---

[4] https://noaa-apt.mbernardi.com.ar/index.html

[5] https://www.youtube.com/watch?v=KWeY2yqwVA0

This is set using the tuning variable, which is adjustable with a slider. The samp_rate variable declares the sampling rate of the analog-to-digital converter within the SDR. This value should be left unchanged from 5MHz for this lab. The gain is in units of dB, set by the rf_gain variable, and should be left unchanged for this lab. The selected antenna is RX2, just like the labels on the radio itself. Bandwidth is set to 5MHz as well, which is arguably overkill for this experiment, but that will be discussed later.

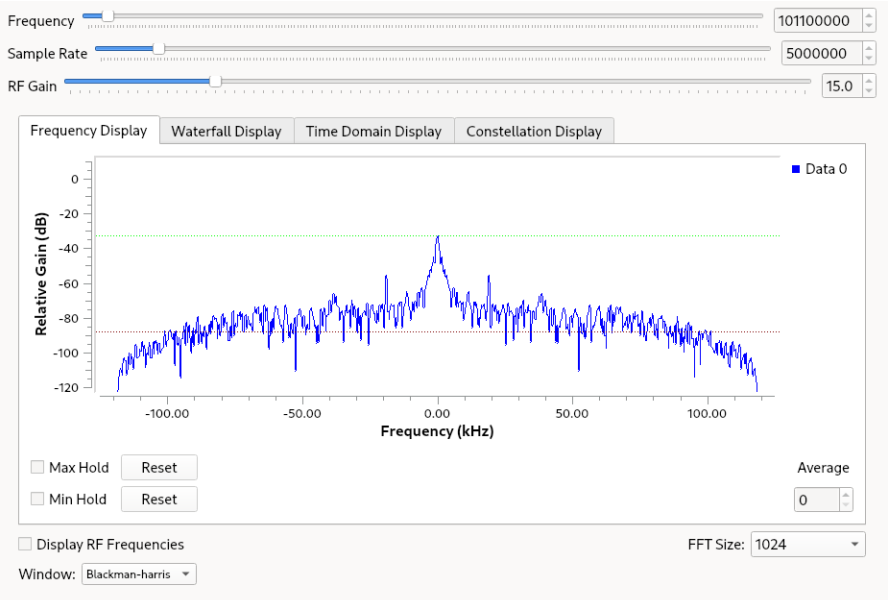
The low pass filter (LPF) serves to both filter out the inaudible high-frequency signals, and reduce the number of samples in the signal, reducing data load. Its FIR type should be set to complex->complex (Decimating).

WBFM Receive is quite a large mathematical abstraction, but what it essentially does is demodulate the FM (frequency modulated) signal into something listenable with a little bit more processing. At this point it should be noted that this block expects an input of type 'complex' (complex number) but produces an output of type 'float' (real number).

The rational resampler is a block that helps us change the sampling rate of our signal, without being too destructive to the information it carries. Interpolating means we are adding extra data points in between the already existing ones, and decimating means we are removing data points. It is important that this block is set to Float->Float (Real Taps), because a float input will be given, and the next block expects a float input.

The audio sink simply tells the computer's sound card to play whatever signal is being passed to it, at a sampling rate of 44.1 kHz.

Lastly, for a visual aid, the QT GUI Sink block is used. This will also need to have its input type changed to float. The center frequency value is meaningless, for now. This block displays a Fourier-Transform of the incoming audio signal.



Exercise for the Reader:

2.1) Consider all the values for USRP's sampling rate, the LPF's sampling rate and decimation, WBFM's quadrature rate, the rational resampler's interpolation and decimation, and the audio sink's sampling rate. How do they relate as the signal flows down the graph? Try changing the LPF's decimation and the rational resampler's decimation, as well as WBFM's quadrature rate, and observe what sounds different.

Lab #3: Reading and Writing Files

This lab will help aid you in understanding the file source and sink blocks. In the lessons folder you will find a file named "FMIQdata.dat" which contains raw sample data from an FM radio station sampled at 250kHz (How will this change the decimation parameter in the LPF block?). Using this file, as well as the flowgraph created in lab #2, convert the raw data into a listenable wav file.

The blocks you will need are File Source, Wav File Sink, and Throttle. Take note of the type of input that the file source and sink blocks are expecting. This is a commonly overlooked parameter!

Hint: You no longer need a connection to a radio to complete this!

Exercises for the reader:

3.1) Look at the file sizes for your output wav file and compare it to the file size of the raw data you were given. What do you notice?

3.2) Return to your flowgraph for lab #2. You can record your own IQ data by connecting a File Sink directly at the output of the USRP source (think about your answer to 3.1 before recording). Try recording your own data but change the sampling rate of the radio (you will need to understand the answer to 2.1 so that the audio comes through clearly). How do the file sizes of both the raw data file and the wav file change in proportion to the length of the recording and the sampling rate of the radio?