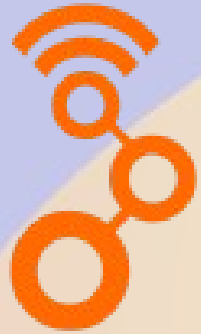


GNURadio

THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

RMHAM University
April 9, 2022

Willem Schreüder AC0KQ
willem@prinmath.com

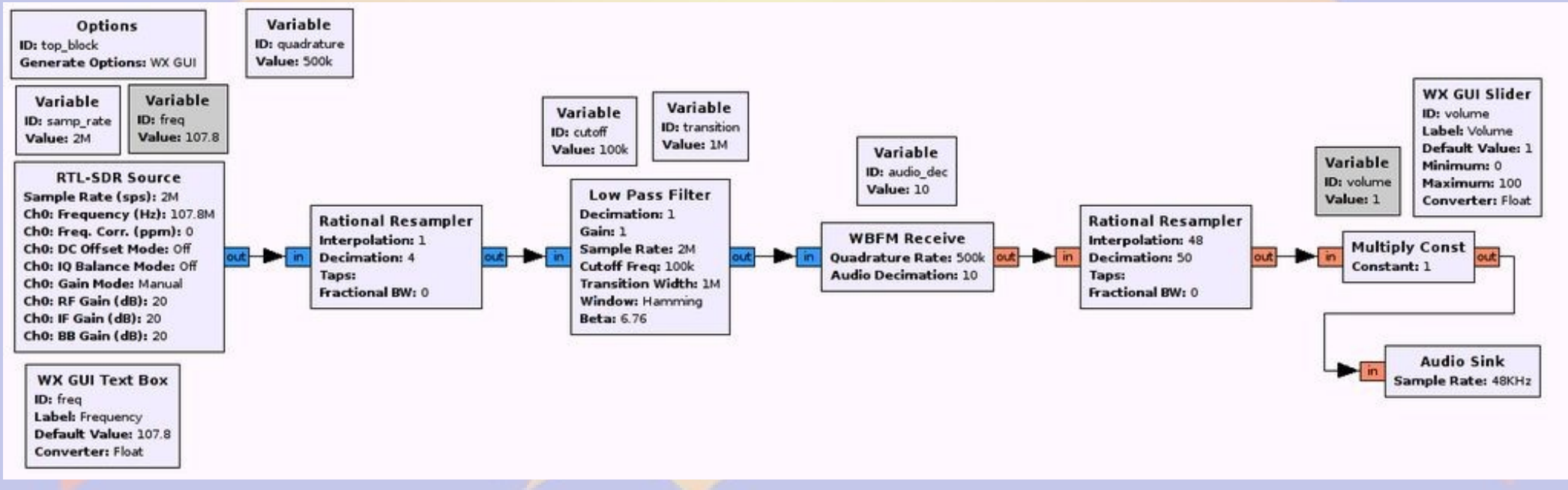
What is GNU Radio

- Software
  - Collection of modules that perform functions required to build a receiver or transmitter
  - Data piped between modules similar to Unix commands
  - End result is a program
- gnu-radio-companion makes it easier to use
  - Python wrapper to connect components
  - Heavy on processor demands
  - Can be run natively on the Raspberry Pi
- Free and extensible

GNU Radio Pros

- Extremely powerful
  - Can build any type of radio
- Supports many hardware types
- Runs on all platforms
- gnu-radio-companion makes it easier to use
  - Python wrapper to connect components
  - Heavy on processor demands
  - Can be run natively on the Raspberry Pi
- Free and extensible

gnu radio companion (grc)



gnu radio hints

- Blue connectors = complex (IQ)
  - dual data stream using complex numbers
- Orange connectors = real
  - single data stream using real numbers
- Gray connectors = data
  - Needs you to set a value or string
- Connector type and rate must match
  - time dilation or stutter if rate mismatch
    - decimation reduces data rate
    - interpolation increase data rate

Installing gnuradio (Ubuntu)

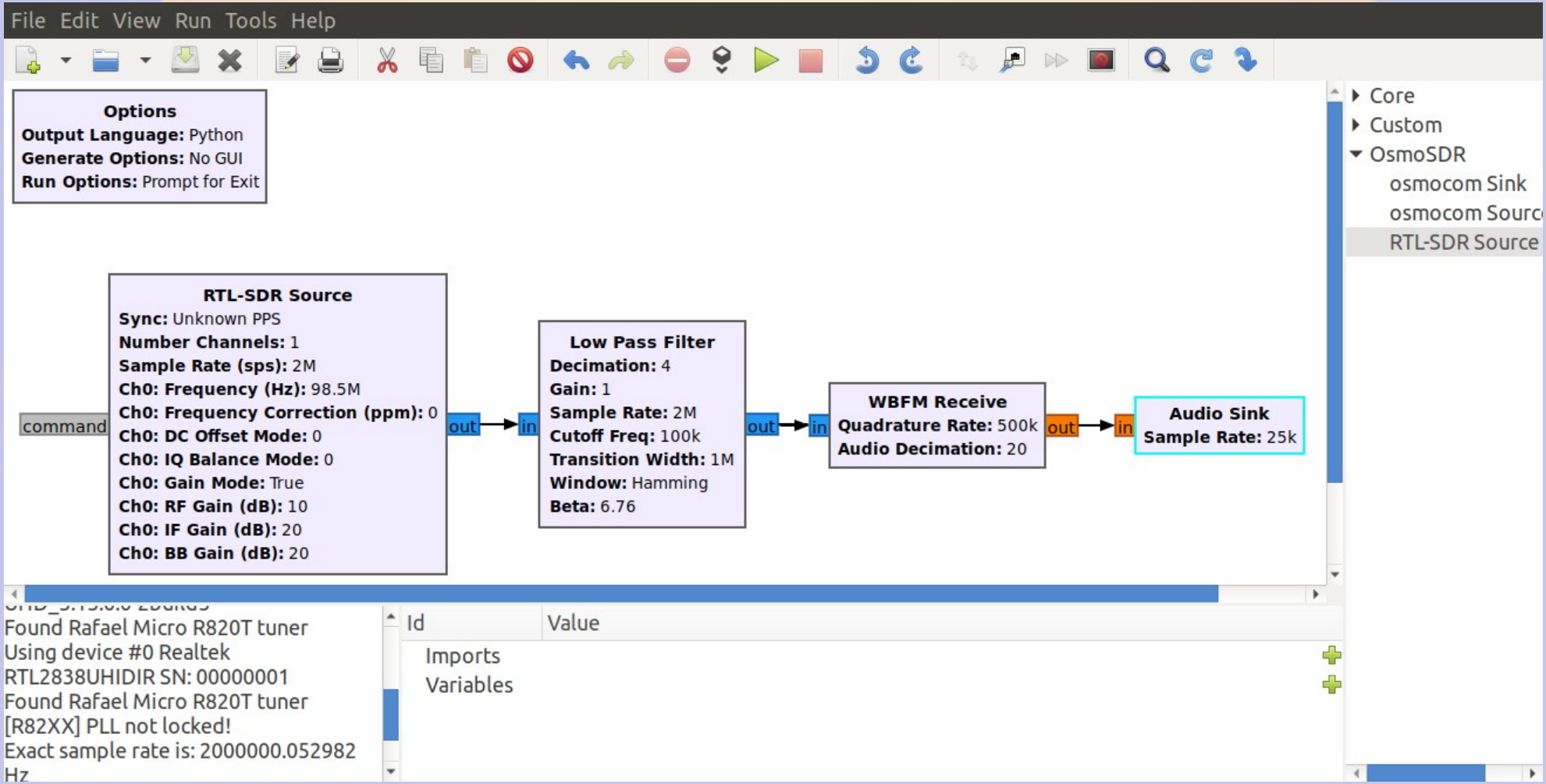
- Install core gnuradio components and grc
  - sudo apt-get install gnuradio
- Install source and sink for for RTL and similar hardware
  - sudo apt-get install gr-osmosdr
- Install RTL-SDR
  - Only needed if directly connecting to RTL-SDR
    - Not needed if connecting via IP
  - Stock RTL-SDR libraries don’t work
  - Needs better UDEV rules

Installing RTL-SDR

- Install prerequisites
  - sudo apt-get install -y cmake pkg-config libusb-1.0
- Download RTL-SDR
  - git clone git://git.osmocom.org/rtl-sdr.git
- Build RTL-SDR
  - cd rtl-sdr
  - mkdir build
  - cd build
  - cmake ../ -DINSTALL_UDEV_RULES=ON
  - make
  - sudo make install
  - sudo ldconfig

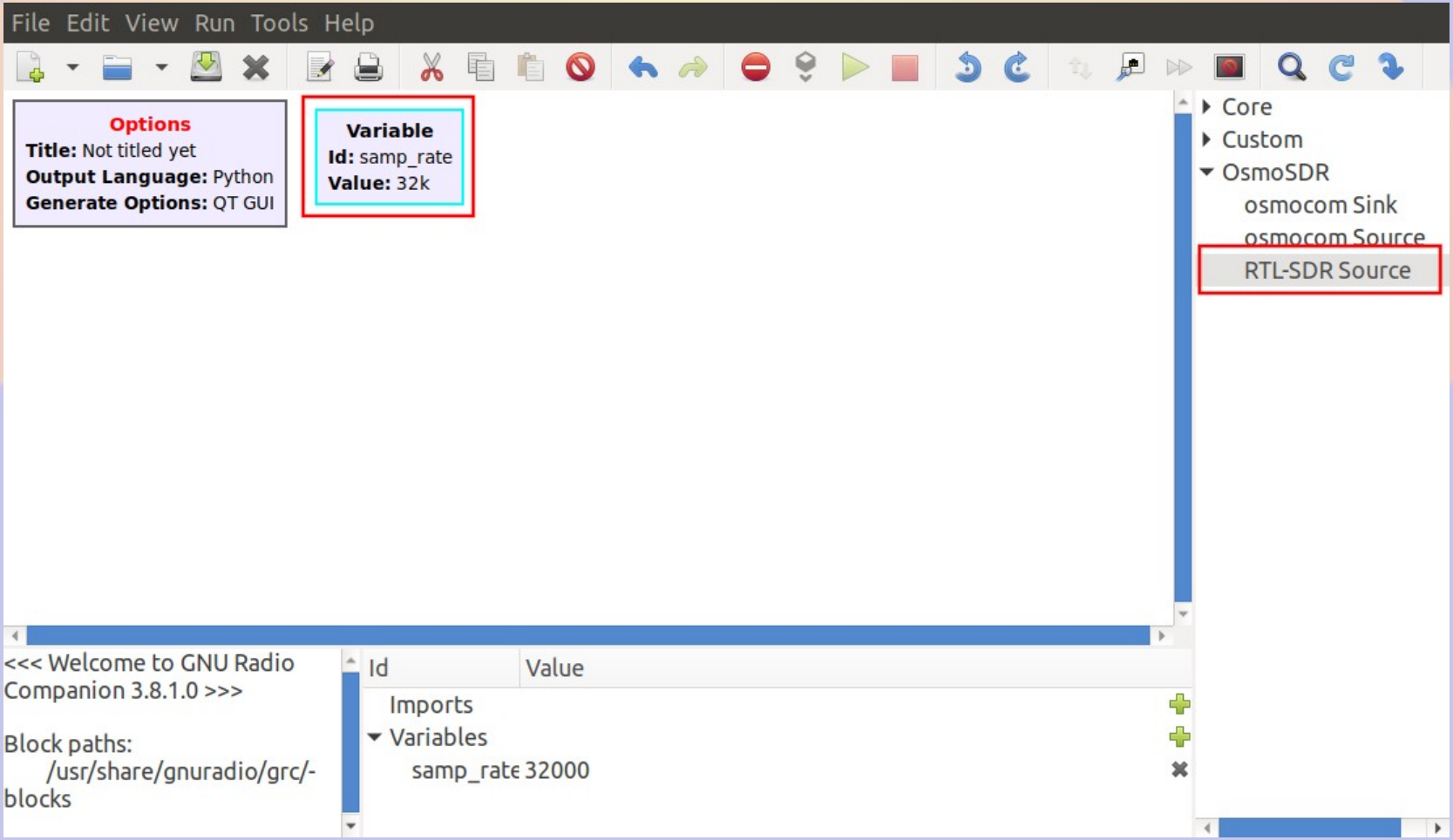
First try: A broadcast FM receiver

- Tuned to 98.5 KYGO (Squaw Mountain)



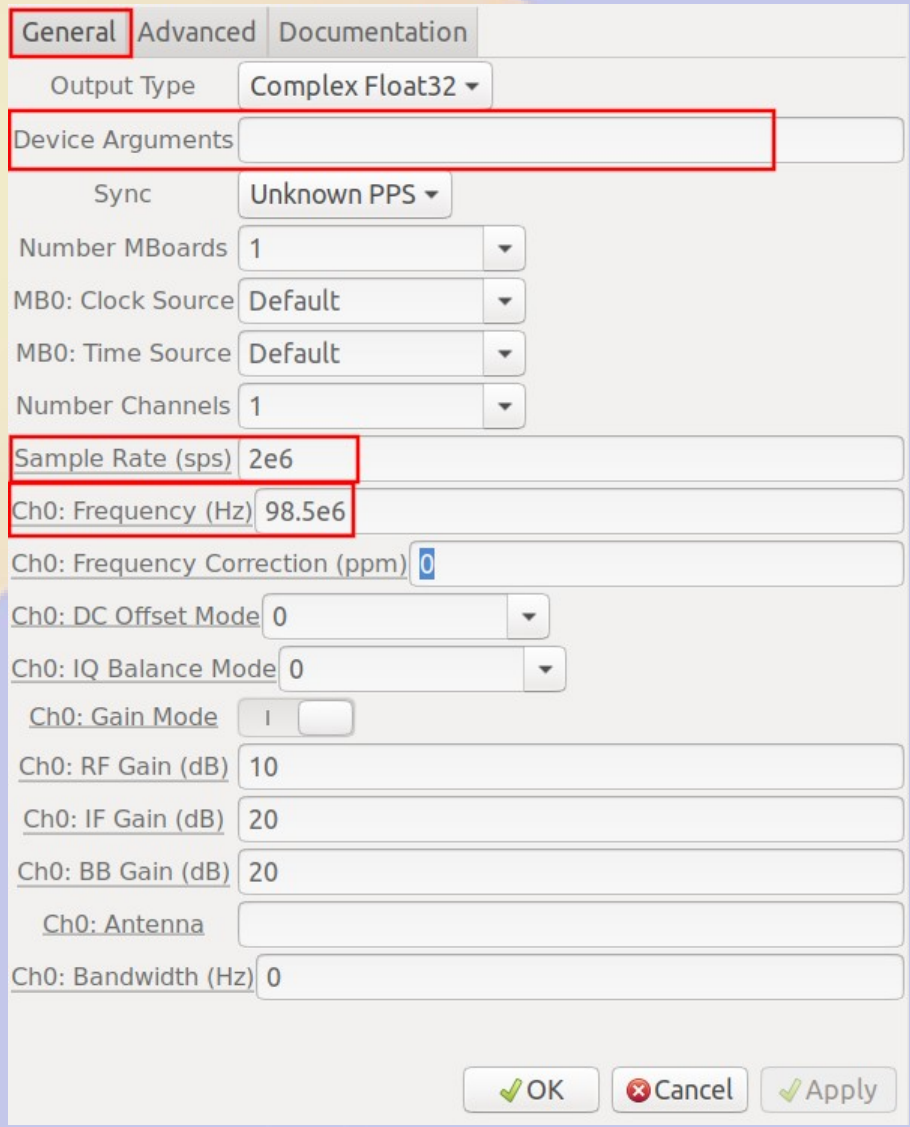
0: File > New > No GUI

- Delete Variable
- Select RTL-SDR from OsmoSDR



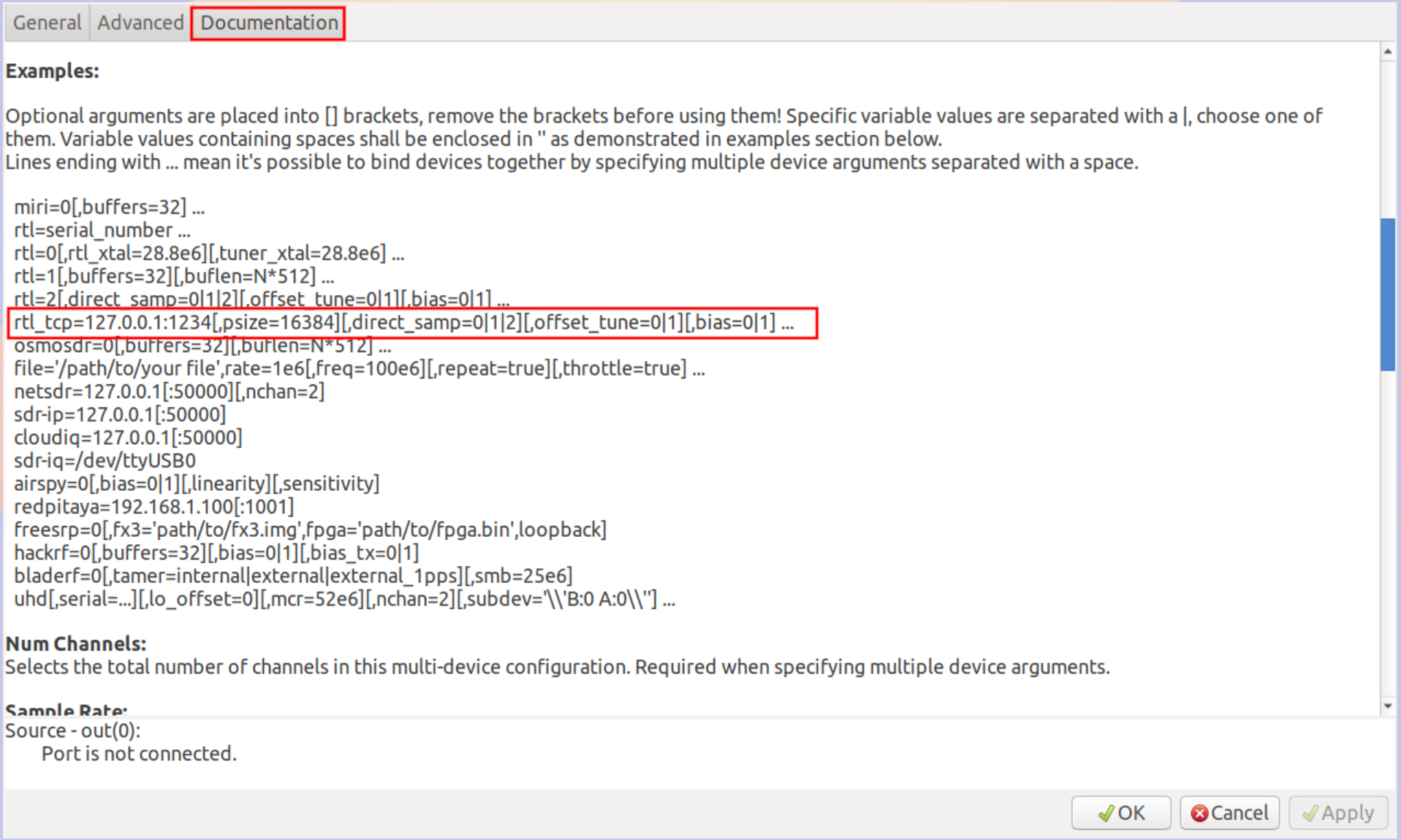
1: Configure RTL-SDR Source

- Blank device arguments selects first RTL-SDR found on USB
- Sample rate 2 MHz
- Ch0 Freq tunes device center frequency to 98.5 Mhz
- *Note output is now 2M samples per second complex (IQ) values*



How to figure stuff out

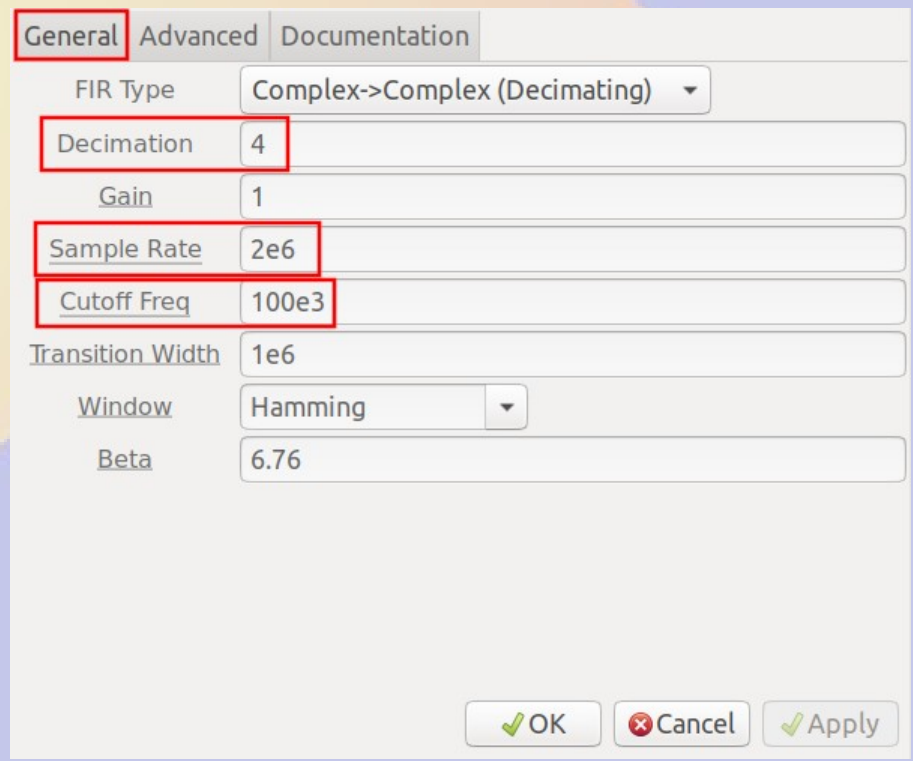
- How to do a remote source over IP with rtl_tcp
  - Device Arguments: rtl_tcp=10.30.60.180:5000



2: Filter data around center frequency

Core > Filters > LowPass Filter

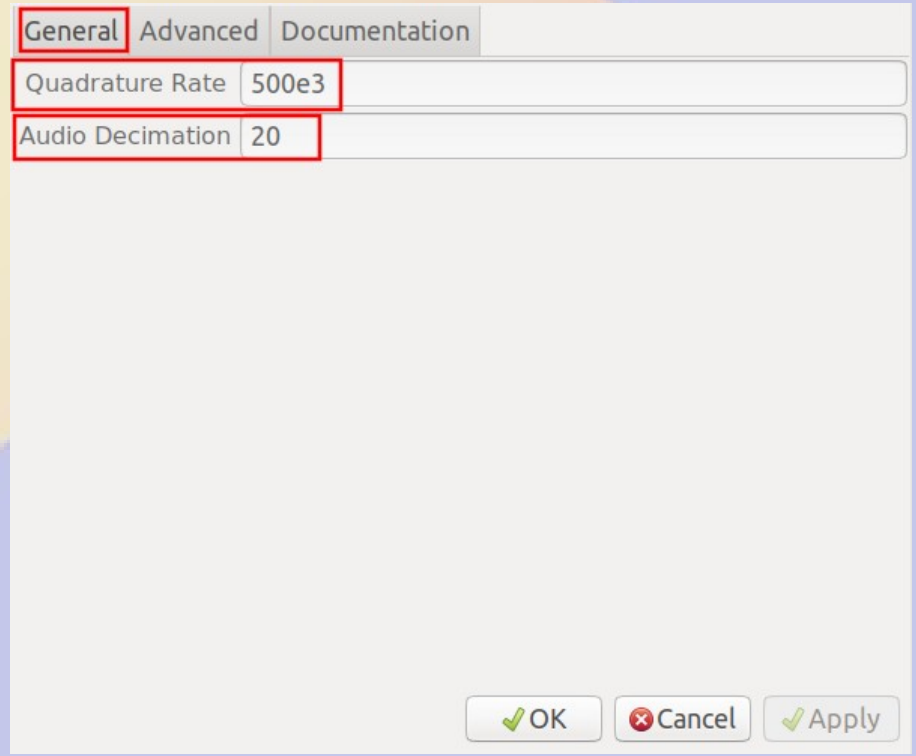
- Sample Rate matches rate from source (2M)
- Decimation 4 reduces output rate 4 fold (500k)
- Cutoff freq sets filter bandwidth (100kHz)
- *Output is 500k complex*



3: Demodulate

Core > Modulators > WBFM Receive

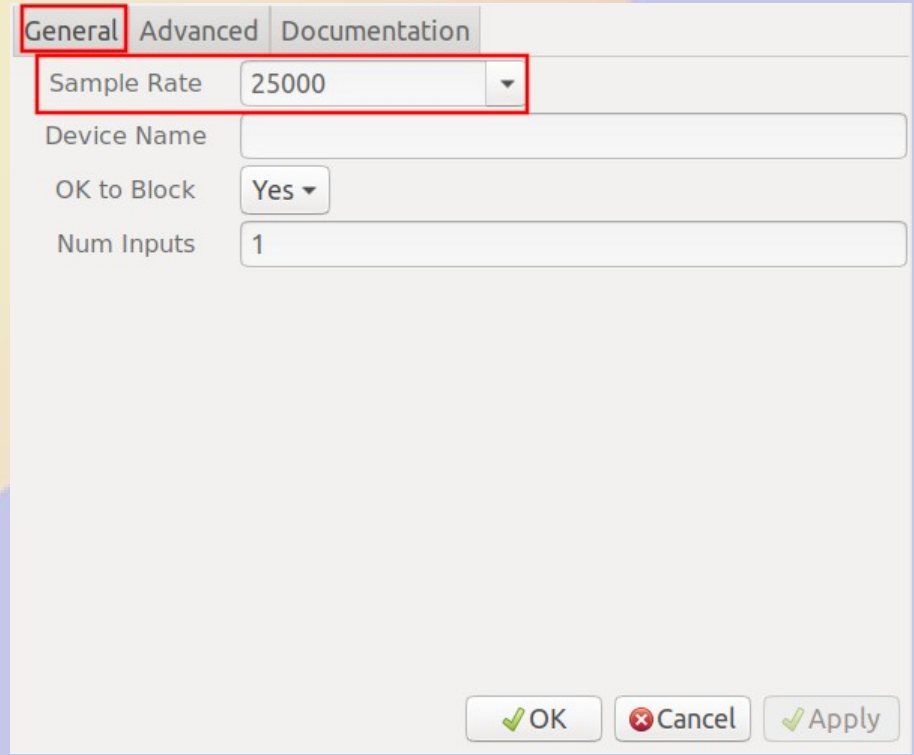
- Quadrature Rate matches rate from filter (500k complex)
- Decimation 20 reduces output rate 20 fold (25k real)
- *Output is 25k real*



4: Output audio to speaker

Core > Audio > Audio Sink

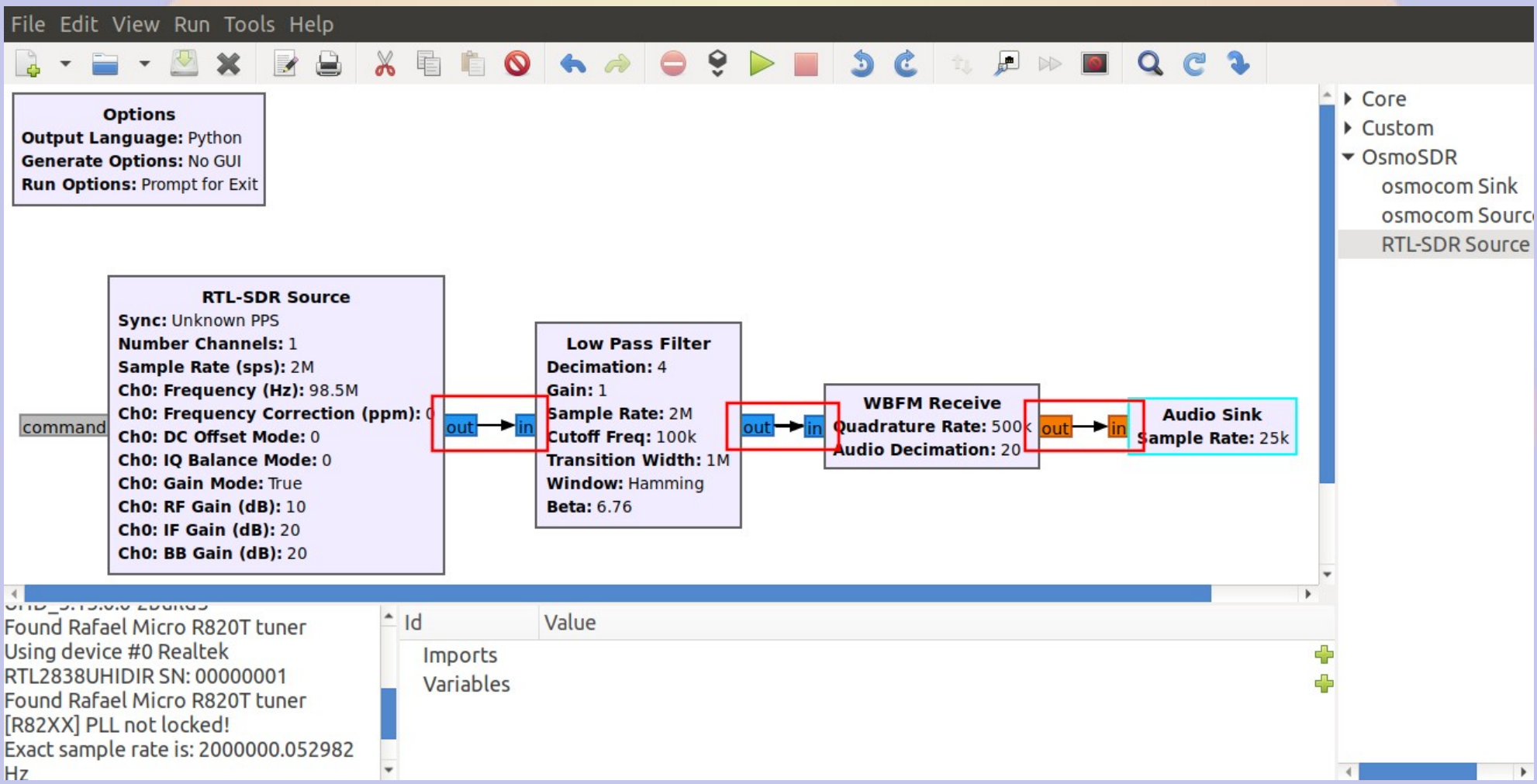
- Sample Rate matches rate from demodulator (25k real)
- Selects default audio output device
- *Output is audio*



5: Connect the modules

Check for incompatible connections

Execute (play)



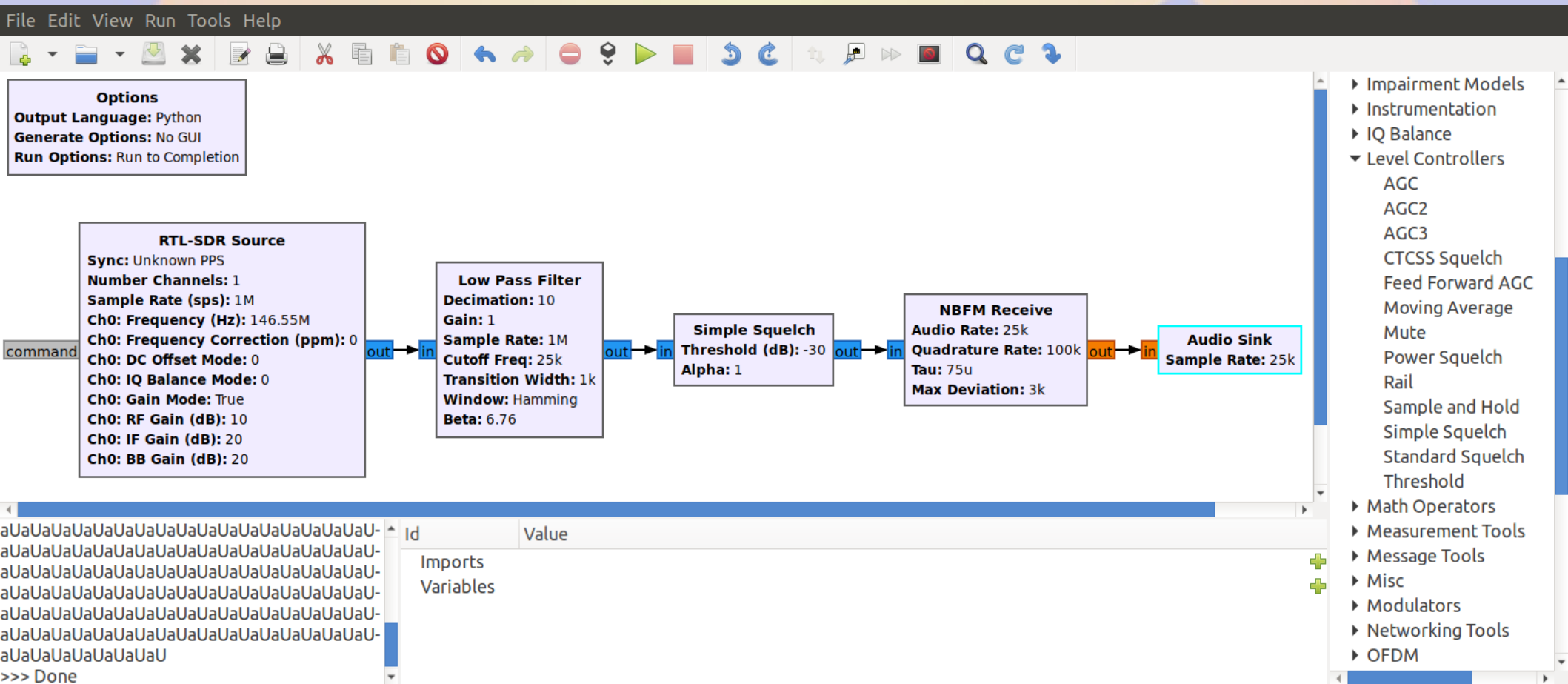
Generate

- Saves the flow graph as a python or C++ program (select in Options Module)
  - Modules are really executable code that can be called from python or C++
- Program can be run from the command line
  - No GUI for command line
  - QT GUI for fancy graphics

Narrowband FM receiver

Tuned to 146.550MHz

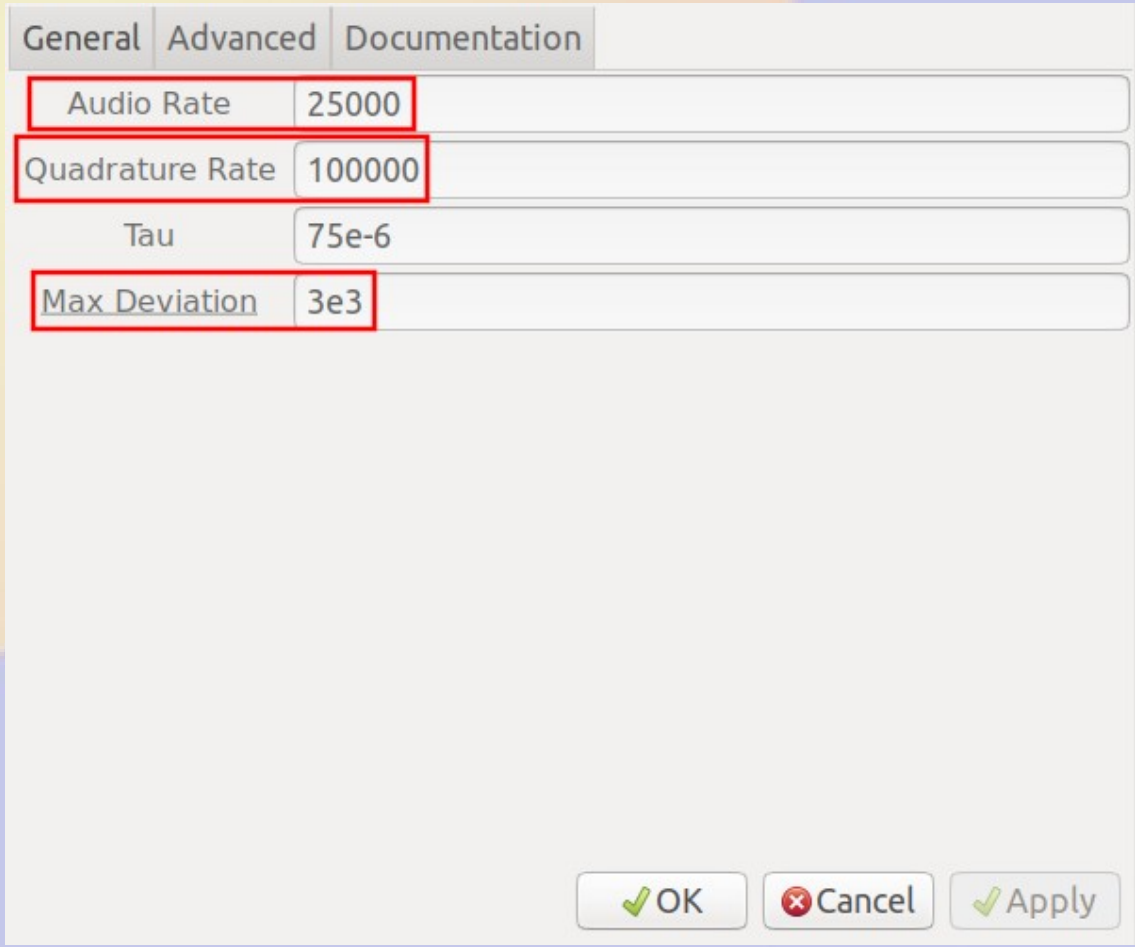
- Reduce RTL-SDR sampling to 1M sps
- Reduce low pass decimation to 10



Narrowband Demodulator

Core > Modulators > NBFM Receive

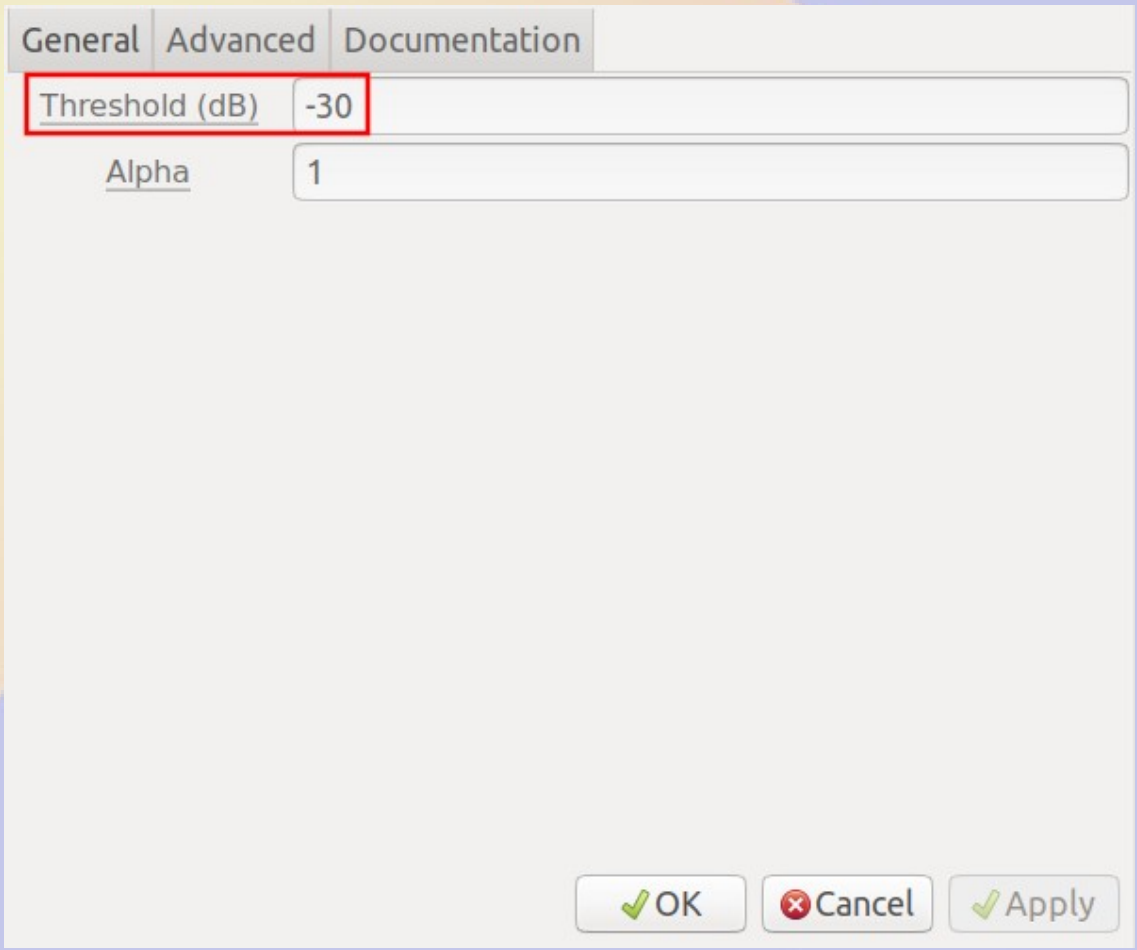
- Quadrature Rate matches rate from filter (100k complex)
- Audio rate 25k real (decimation 4)
- Max deviation 3kHz



Squelch

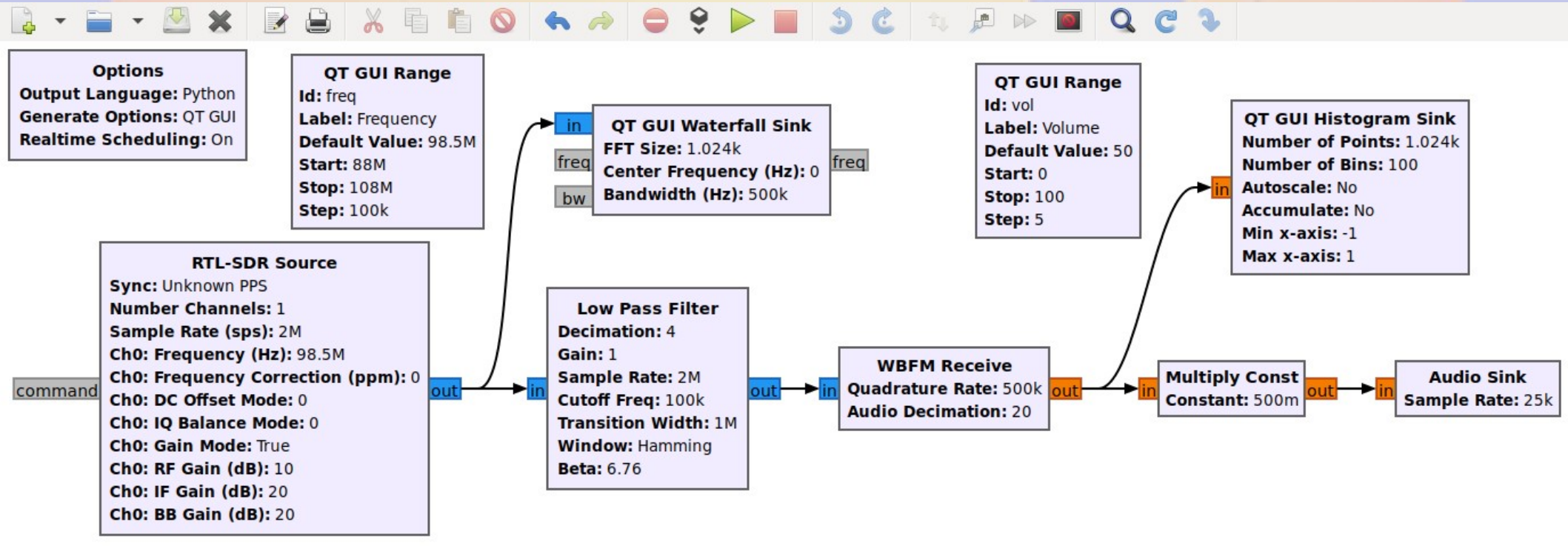
Core > Level Controllers > Simple Squelch

- Set threshold for squelch to open (dB)
- Rate unchanged



Adding more usable contols

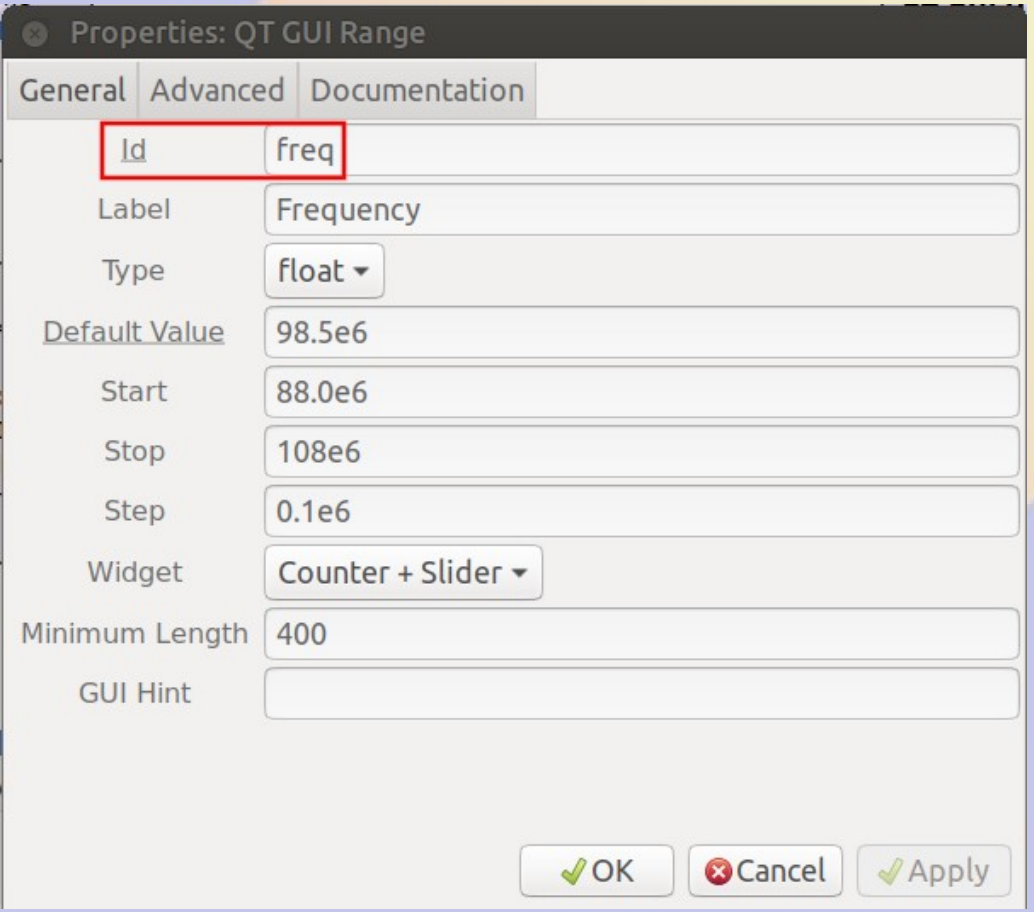
- Set frequency, and volume
  - Replace values with variables
- Display waterfall and spectrum

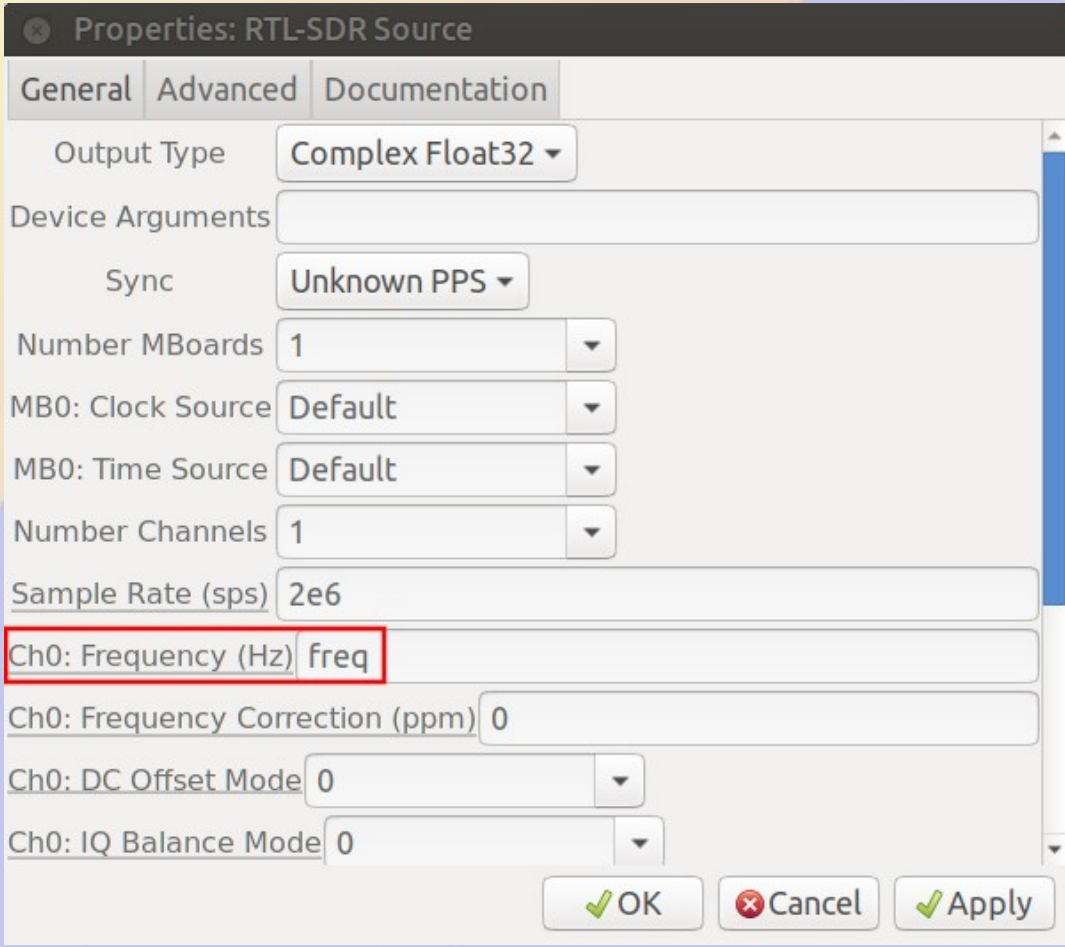


Frequency

Core > GUI Widgets > QT > QT GUI Range

- Replace value with variable **freq**

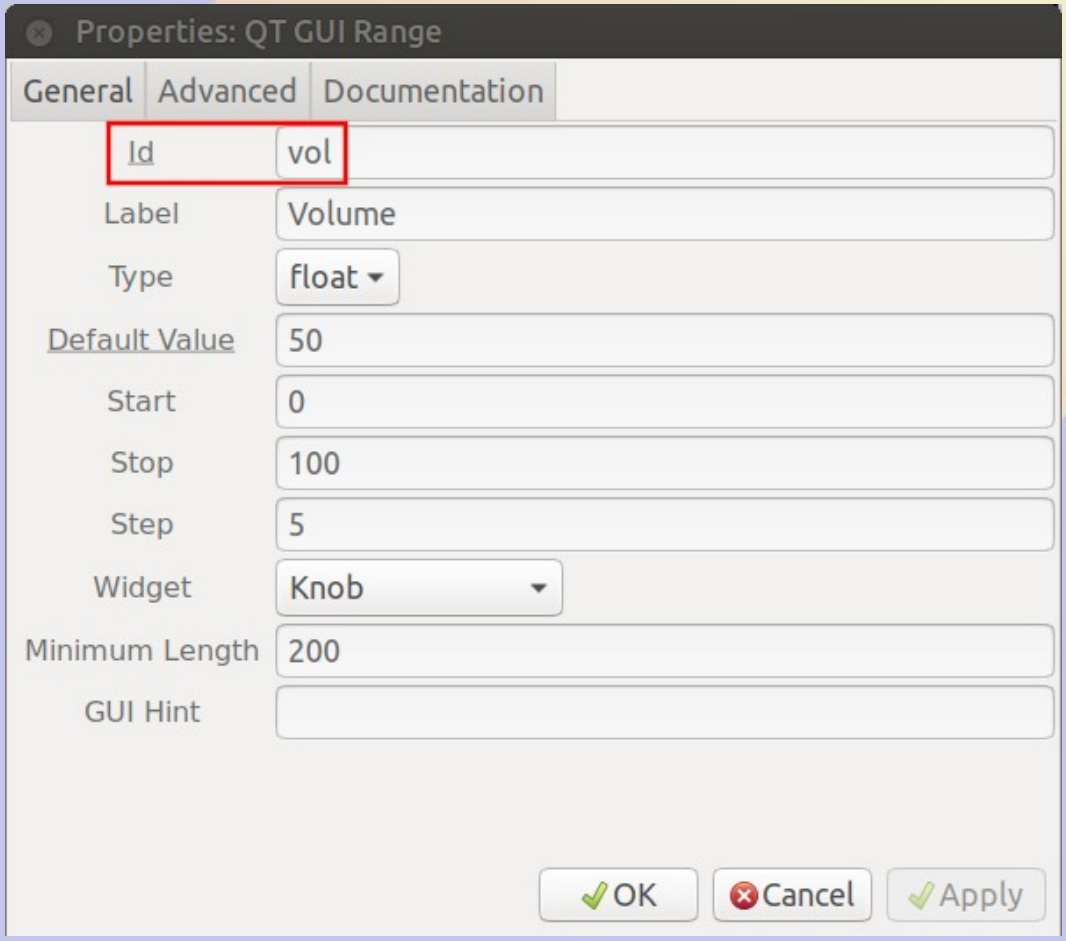


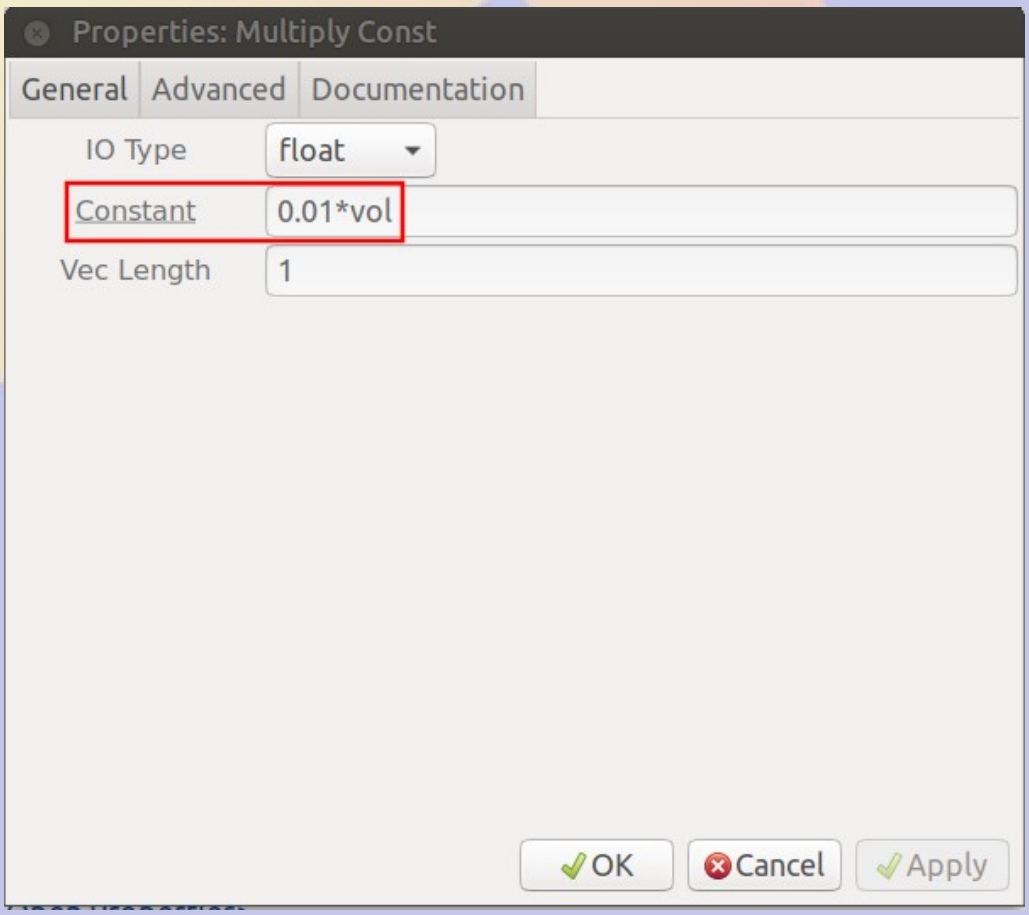


Volume

Core > GUI Widgets > QT > QT GUI Range
Core > Math Operators > Multiply Const

- Add volume control (0-100) named **vol**
- Add multiplier before audio sink (**0.01*vol**)

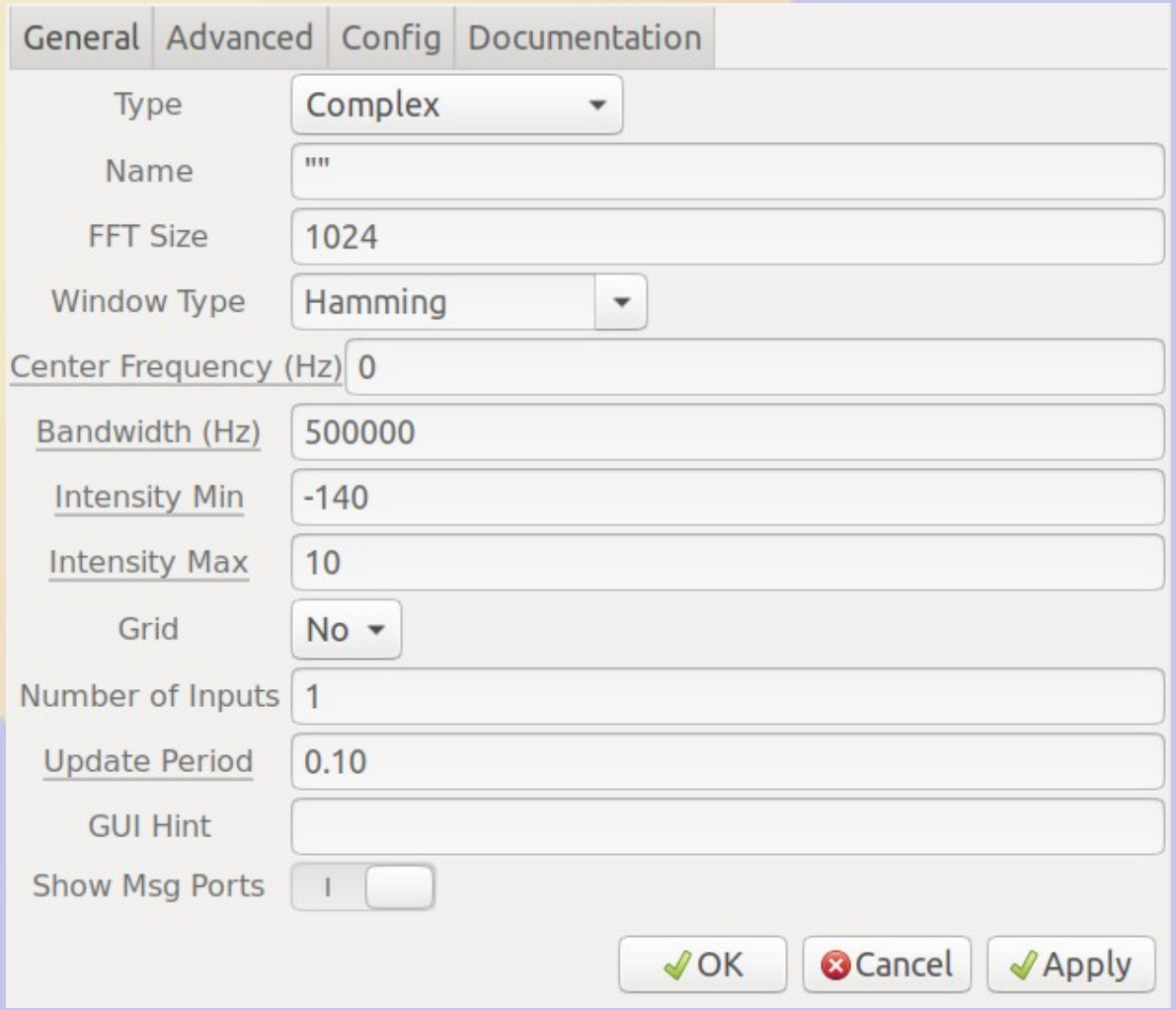




Waterfall

Core > Instrumentation > QT > QT GUI Waterfall Sink

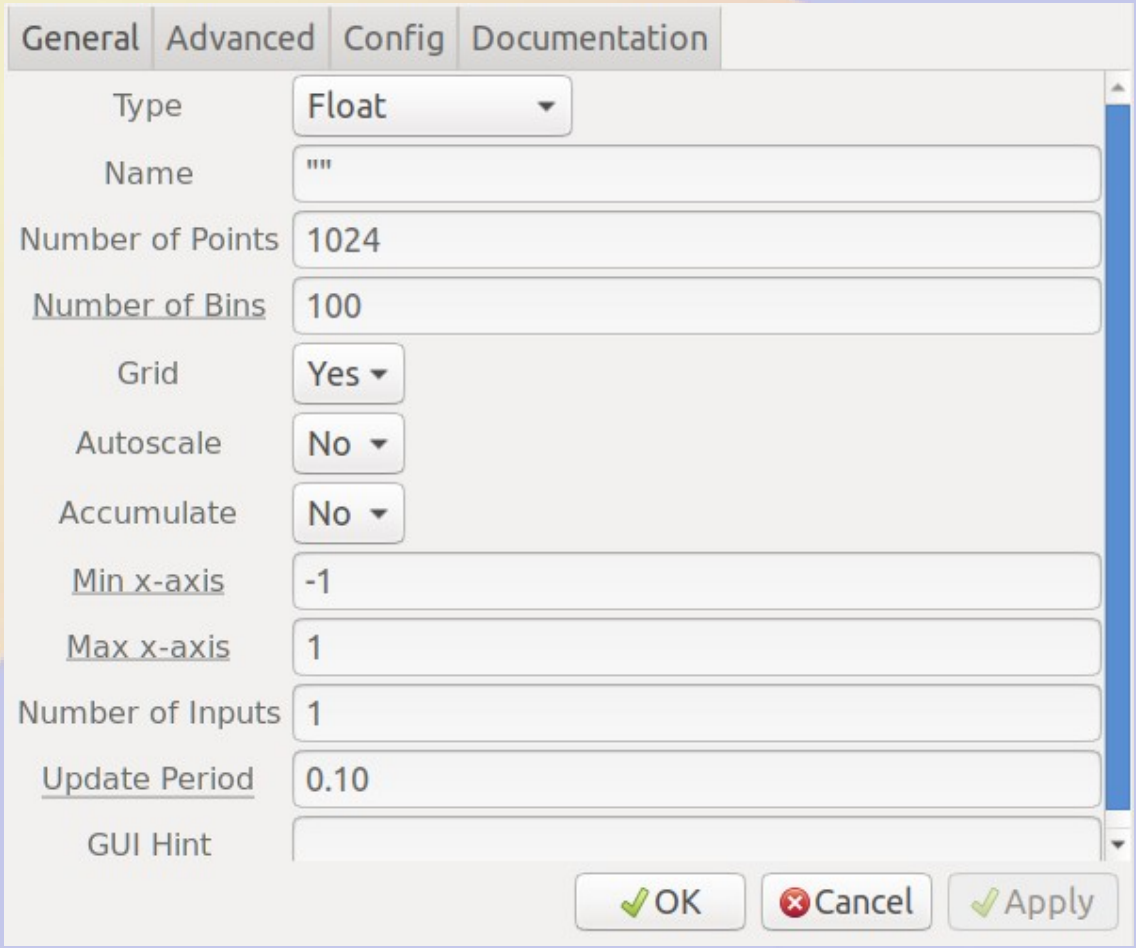
- Connect to RTL-SDR output
  - Parallels output to low pass filter
- Select bandwith to suit



Spectrum (Histogram)

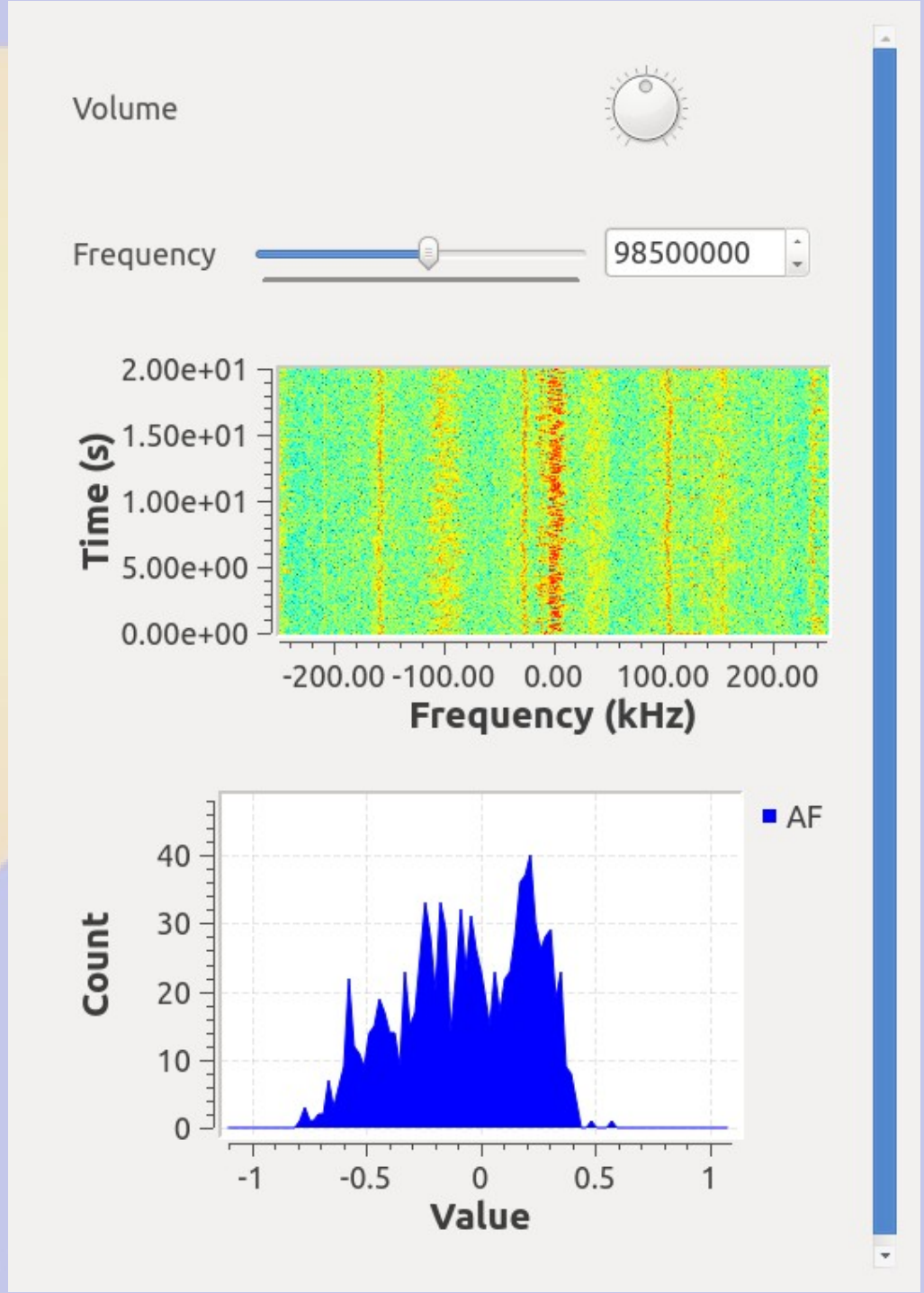
Core > Instrumentation > QT > QT GUI Histogram Sink

- Connect to demodulator (real) output
  - Parallels audio output
- Select range



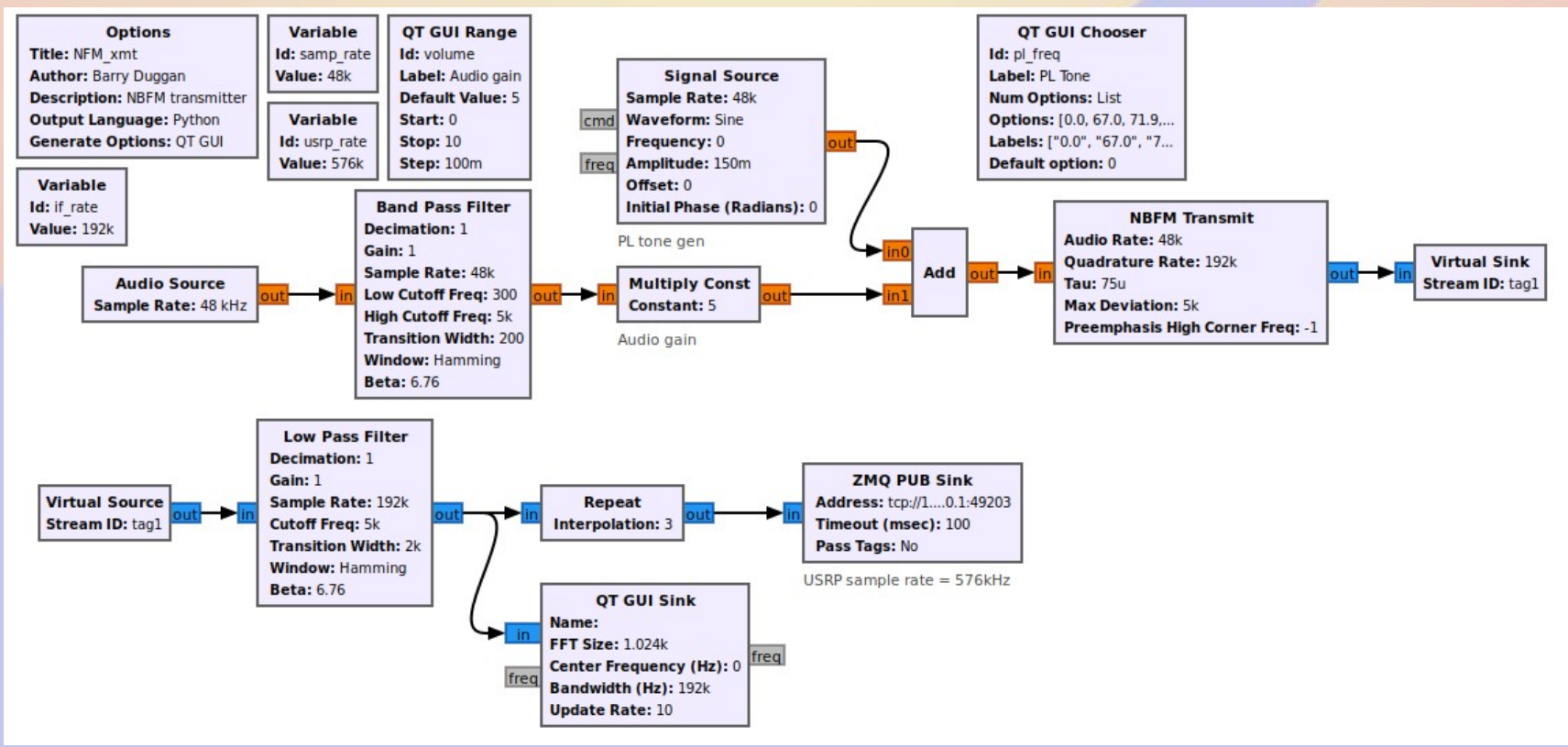
Run It

- Volume know
- Frequency slider
- Waterfall
- Spectrum
- *Works from both the command line and grc*



Lots more possibilities….

- https://www.gnuradio.org/
- https://wiki.gnuradio.org/
  - Lots of examples, tutorials and howto’s



Show and Tell