



GNU Radio

THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

RMHAM University
April 9, 2022

Willem Schreüder AC0KQ
willem@prinmath.com

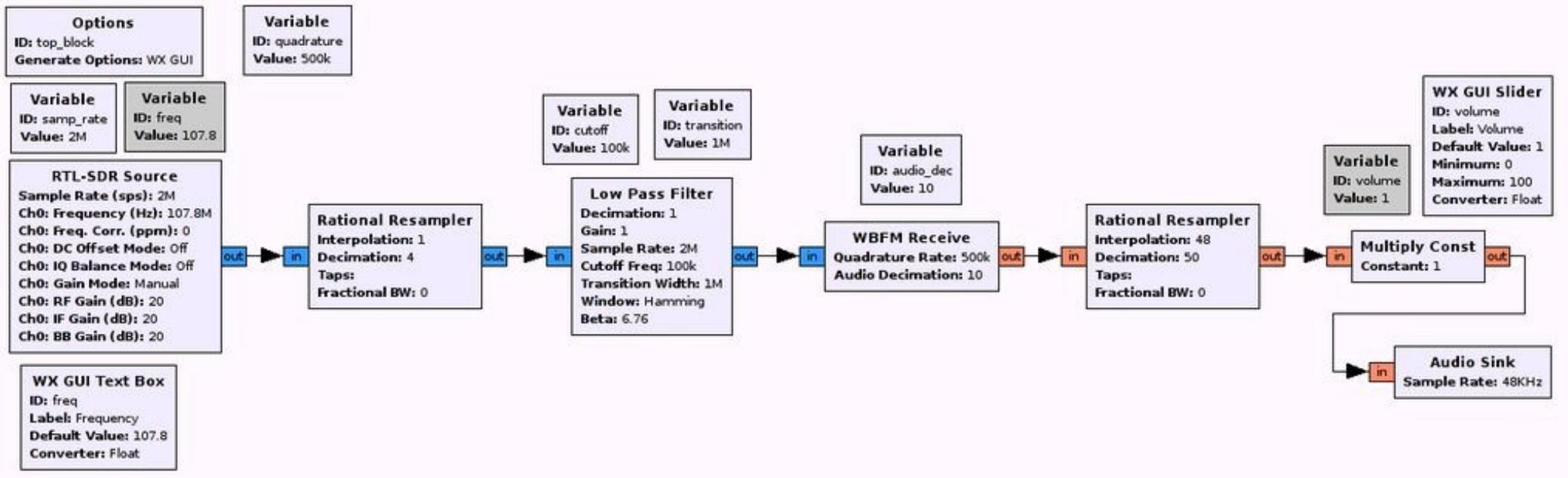
What is GNU Radio

- 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)

The screenshot displays the GNU Radio Companion (GRC) interface for a broadcast FM receiver. The flow graph consists of the following blocks:

- RTL-SDR Source**: Configured with Sync: Unknown PPS, Number Channels: 1, Sample Rate (sps): 2M, Ch0: Frequency (Hz): 98.5M, Ch0: Frequency Correction (ppm): 0, Ch0: DC Offset Mode: 0, Ch0: IQ Balance Mode: 0, Ch0: Gain Mode: True, Ch0: RF Gain (dB): 10, Ch0: IF Gain (dB): 20, and Ch0: BB Gain (dB): 20.
- Low Pass Filter**: Configured with Decimation: 4, Gain: 1, Sample Rate: 2M, Cutoff Freq: 100k, Transition Width: 1M, Window: Hamming, and Beta: 6.76.
- WBFM Receive**: Configured with Quadrature Rate: 500k and Audio Decimation: 20.
- Audio Sink**: Configured with Sample Rate: 25k.

The flow is: RTL-SDR Source (out) → Low Pass Filter (in) → Low Pass Filter (out) → WBFM Receive (in) → WBFM Receive (out) → Audio Sink (in).

Options Panel:

- Output Language: Python
- Generate Options: No GUI
- Run Options: Prompt for Exit

Terminal Output:

```
Found Rafael Micro R820T tuner
Using device #0 Realtek
RTL2838UHIDIR SN: 00000001
Found Rafael Micro R820T tuner
[R82XX] PLL not locked!
Exact sample rate is: 2000000.052982
Hz
```

Block Parameters Table:

Block	Parameter	Value
RTL-SDR Source	Sync	Unknown PPS
	Number Channels	1
	Sample Rate (sps)	2M
	Ch0: Frequency (Hz)	98.5M
	Ch0: Frequency Correction (ppm)	0
	Ch0: DC Offset Mode	0
	Ch0: IQ Balance Mode	0
	Ch0: Gain Mode	True
	Ch0: RF Gain (dB)	10
	Ch0: IF Gain (dB)	20
Ch0: BB Gain (dB)	20	
Low Pass Filter	Decimation	4
	Gain	1
	Sample Rate	2M
	Cutoff Freq	100k
	Transition Width	1M
	Window	Hamming
Beta	6.76	
WBFM Receive	Quadrature Rate	500k
	Audio Decimation	20
Audio Sink	Sample Rate	25k

0: File > New > No GUI

- Delete Variable
- Select RTL-SDR from OsmoSDR

The screenshot displays the GNU Radio Companion (GRC) interface. At the top, the menu bar includes 'File', 'Edit', 'View', 'Run', 'Tools', and 'Help'. Below the menu bar is a toolbar with various icons for file operations, editing, and execution. The main workspace is currently empty. On the left side, there are two panels: 'Options' and 'Variable'. The 'Options' panel shows 'Title: Not titled yet', 'Output Language: Python', and 'Generate Options: QT GUI'. The 'Variable' panel, which is highlighted with a red box, shows 'Id: samp_rate' and 'Value: 32k'. On the right side, there is a block palette with a tree view showing 'Core', 'Custom', and 'OsmoSDR'. Under 'OsmoSDR', there are three items: 'osmocom Sink', 'osmocom Source', and 'RTL-SDR Source', with the latter highlighted by a red box. At the bottom of the window, there is a console window showing the welcome message for GNU Radio Companion 3.8.1.0 and the block paths: '/usr/share/gnuradio/grc-blocks'. To the right of the console is a table with columns 'Id' and 'Value'.

Id	Value
Imports	
Variables	
samp_rate	32000

1: Configure RTL-SDR Source

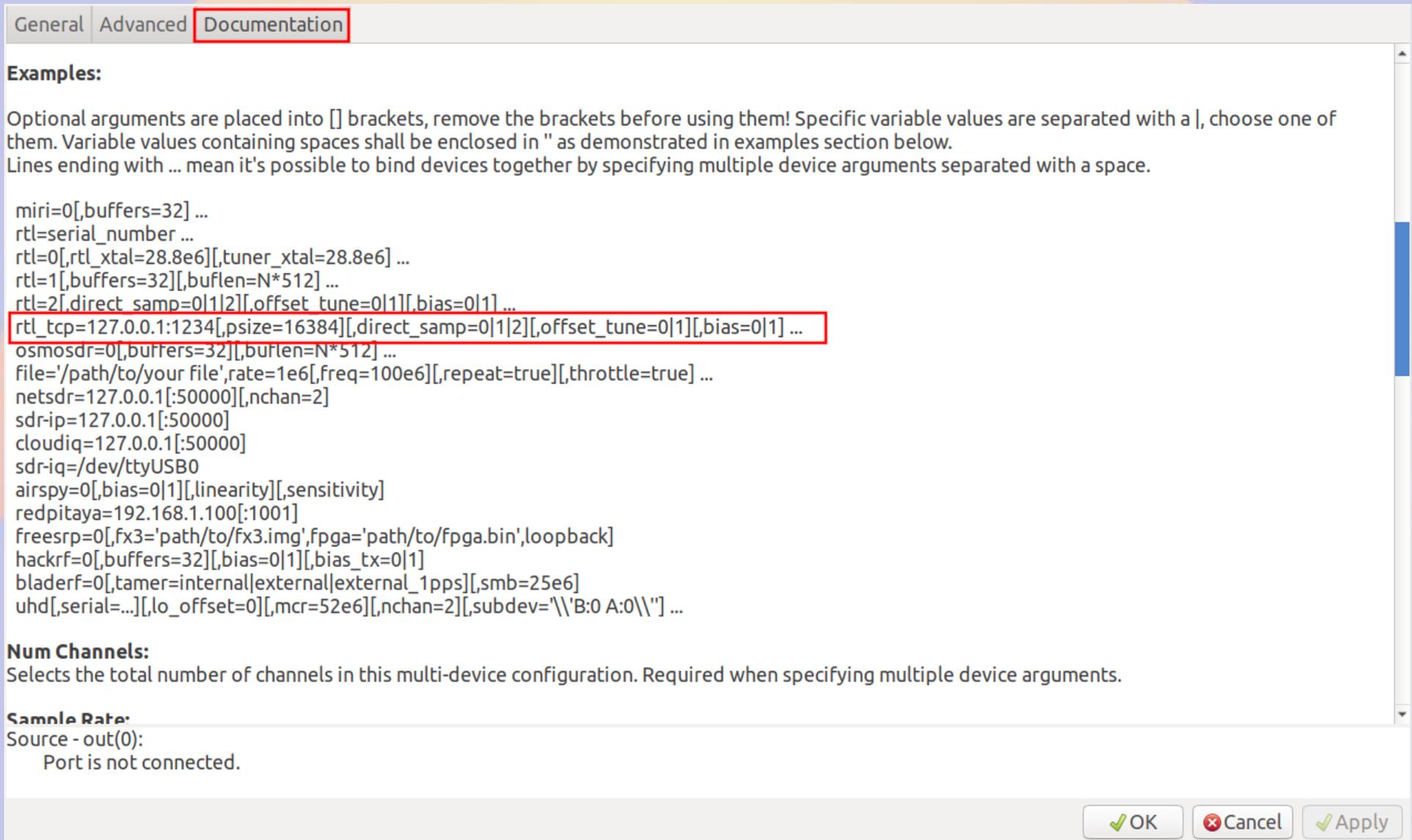
- Blank device arguments selects first RTL-SDR found on USB
- Sample rate 2 MHz
- Ch0 Freq tunes device center frequency to 98.5 Mhz
- *Note output is now 2M samples per second complex (IQ) values*

The screenshot shows the 'General' tab of the RTL-SDR software configuration window. The 'Output Type' is set to 'Complex Float32'. The 'Device Arguments' field is empty. The 'Sync' is set to 'Unknown PPS'. The 'Number MBoards' is 1, 'MB0: Clock Source' is 'Default', 'MB0: Time Source' is 'Default', and 'Number Channels' is 1. The 'Sample Rate (sps)' is 2e6, 'Ch0: Frequency (Hz)' is 98.5e6, and 'Ch0: Frequency Correction (ppm)' is 0. The 'Ch0: DC Offset Mode' is 0, 'Ch0: IQ Balance Mode' is 0, 'Ch0: Gain Mode' is 'I', 'Ch0: RF Gain (dB)' is 10, 'Ch0: IF Gain (dB)' is 20, and 'Ch0: BB Gain (dB)' is 20. The 'Ch0: Antenna' field is empty, and 'Ch0: Bandwidth (Hz)' is 0. At the bottom, there are 'OK', 'Cancel', and 'Apply' buttons.

Field	Value
Output Type	Complex Float32
Device Arguments	
Sync	Unknown PPS
Number MBoards	1
MB0: Clock Source	Default
MB0: Time Source	Default
Number Channels	1
Sample Rate (sps)	2e6
Ch0: Frequency (Hz)	98.5e6
Ch0: Frequency Correction (ppm)	0
Ch0: DC Offset Mode	0
Ch0: IQ Balance Mode	0
Ch0: Gain Mode	I
Ch0: RF Gain (dB)	10
Ch0: IF Gain (dB)	20
Ch0: BB Gain (dB)	20
Ch0: Antenna	
Ch0: Bandwidth (Hz)	0

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



General | Advanced | **Documentation**

Examples:

Optional arguments are placed into [] brackets, remove the brackets before using them! Specific variable values are separated with a |, choose one of them. Variable values containing spaces shall be enclosed in " as demonstrated in examples section below.
Lines ending with ... mean it's possible to bind devices together by specifying multiple device arguments separated with a space.

```
miri=0[,buffers=32] ...
rtl=serial_number ...
rtl=0[,rtl_xtal=28.8e6][,tuner_xtal=28.8e6] ...
rtl=1[,buffers=32][,buflen=N*512] ...
rtl=2[,direct_samp=0|1|2][,offset_tune=0|1][,bias=0|1] ...
rtl_tcp=127.0.0.1:1234[,psize=16384][,direct_samp=0|1|2][,offset_tune=0|1][,bias=0|1] ...
osmosdr=0[,buffers=32][,buflen=N*512] ...
file='/path/to/your file',rate=1e6[,freq=100e6][,repeat=true][,throttle=true] ...
netsdr=127.0.0.1[:50000][,nchan=2]
sdr-ip=127.0.0.1[:50000]
cloudiq=127.0.0.1[:50000]
sdr-iq=/dev/ttyUSB0
airspy=0[,bias=0|1][,linearity][,sensitivity]
redpitaya=192.168.1.100[:1001]
freesrp=0[,fx3='path/to/fx3.img',fpga='path/to/fpga.bin',loopback]
hackrf=0[,buffers=32][,bias=0|1][,bias_tx=0|1]
bladerf=0[,tuner=internal|external|external_1pps][,smb=25e6]
uhd[,serial=...][,lo_offset=0][,mcr=52e6][,nchan=2][,subdev='\\B:0 A:0\\'] ...
```

Num Channels:
Selects the total number of channels in this multi-device configuration. Required when specifying multiple device arguments.

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

OK Cancel Apply

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*

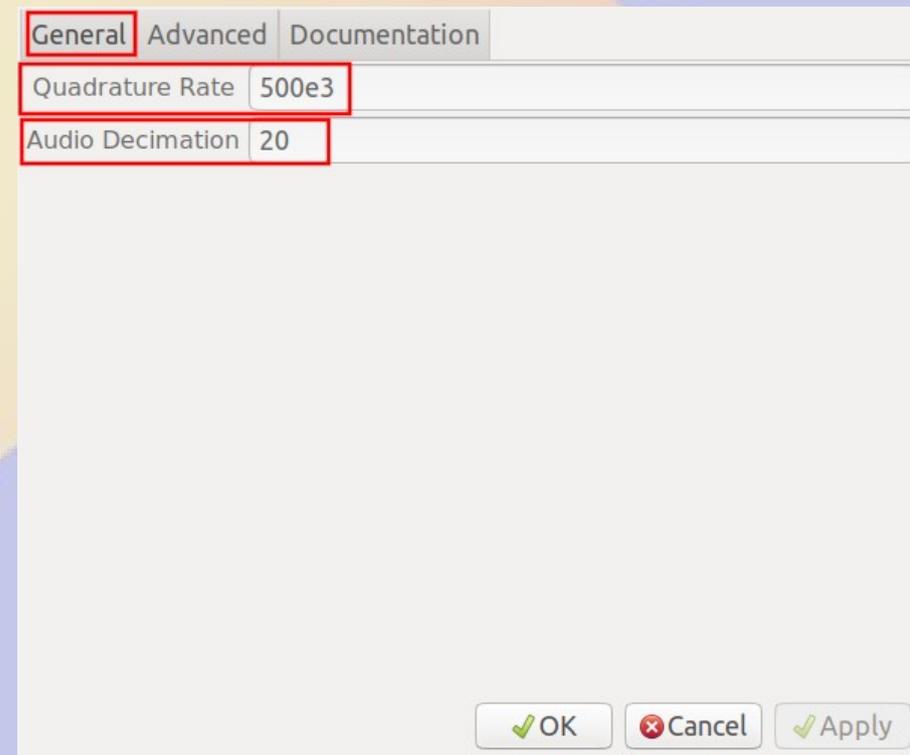
The screenshot shows the configuration for a LowPass Filter. The 'General' tab is active. The 'FIR Type' is set to 'Complex->Complex (Decimating)'. The 'Decimation' is set to 4, 'Gain' is 1, 'Sample Rate' is 2e6, 'Cutoff Freq' is 100e3, 'Transition Width' is 1e6, 'Window' is Hamming, and 'Beta' is 6.76. The 'OK', 'Cancel', and 'Apply' buttons are visible at the bottom right.

Parameter	Value
FIR Type	Complex->Complex (Decimating)
Decimation	4
Gain	1
Sample Rate	2e6
Cutoff Freq	100e3
Transition Width	1e6
Window	Hamming
Beta	6.76

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*



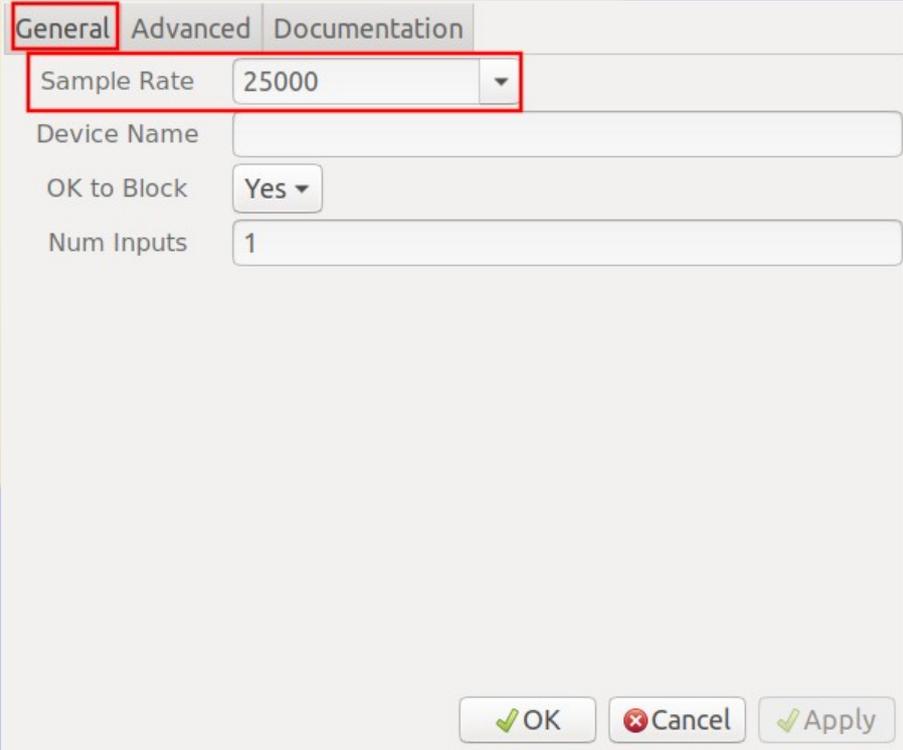
The image shows a configuration dialog box for the 'WBFM Receive' block. It has three tabs: 'General', 'Advanced', and 'Documentation'. The 'General' tab is selected. There are two input fields: 'Quadrature Rate' with the value '500e3' and 'Audio Decimation' with the value '20'. At the bottom right, there are three buttons: 'OK' (with a green checkmark), 'Cancel' (with a red X), and 'Apply' (with a green checkmark).

Parameter	Value
Quadrature Rate	500e3
Audio Decimation	20

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*



The screenshot shows a configuration dialog box for an Audio Sink. It has three tabs: 'General' (selected), 'Advanced', and 'Documentation'. The 'General' tab contains the following settings:

- Sample Rate: 25000 (highlighted with a red box)
- Device Name: (empty text field)
- OK to Block: Yes (dropdown menu)
- Num Inputs: 1 (text field)

At the bottom right, there are three buttons: 'OK' (with a green checkmark), 'Cancel' (with a red X), and 'Apply' (with a green checkmark).

5: Connect the modules



Check for incompatible connections



Execute (play)

Options
Output Language: Python
Generate Options: No GUI
Run Options: Prompt for Exit

RTL-SDR Source
Sync: Unknown PPS
Number Channels: 1
Sample Rate (sps): 2M
Ch0: Frequency (Hz): 98.5M
Ch0: Frequency Correction (ppm): 0
Ch0: DC Offset Mode: 0
Ch0: IQ Balance Mode: 0
Ch0: Gain Mode: True
Ch0: RF Gain (dB): 10
Ch0: IF Gain (dB): 20
Ch0: BB Gain (dB): 20

Low Pass Filter
Decimation: 4
Gain: 1
Sample Rate: 2M
Cutoff Freq: 100k
Transition Width: 1M
Window: Hamming
Beta: 6.76

WBFM Receive
Quadrature Rate: 500k
Audio Decimation: 20

Audio Sink
Sample Rate: 25k

command

Found Rafael Micro R820T tuner
Using device #0 Realtek
RTL2838UHIDIR SN: 00000001
Found Rafael Micro R820T tuner
[R82XX] PLL not locked!
Exact sample rate is: 2000000.052982 Hz

Id	Value
Imports	
Variables	

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

The screenshot displays the GNU Radio Companion (GRC) interface for a narrowband FM receiver. The flow graph consists of the following blocks:

- RTL-SDR Source**: Configured with Sync: Unknown PPS, Number Channels: 1, Sample Rate (sps): 1M, Ch0: Frequency (Hz): 146.55M, Ch0: Frequency Correction (ppm): 0, Ch0: DC Offset Mode: 0, Ch0: IQ Balance Mode: 0, Ch0: Gain Mode: True, Ch0: RF Gain (dB): 10, Ch0: IF Gain (dB): 20, and Ch0: BB Gain (dB): 20.
- Low Pass Filter**: Configured with Decimation: 10, Gain: 1, Sample Rate: 1M, Cutoff Freq: 25k, Transition Width: 1k, Window: Hamming, and Beta: 6.76.
- Simple Squelch**: Configured with Threshold (dB): -30 and Alpha: 1.
- NBFM Receive**: Configured with Audio Rate: 25k, Quadrature Rate: 100k, Tau: 75u, and Max Deviation: 3k.
- Audio Sink**: Configured with Sample Rate: 25k.

The interface also shows an Options panel with the following settings:

- Output Language: Python
- Generate Options: No GUI
- Run Options: Run to Completion

On the right side, a menu is open showing various tool categories:

- Impairment Models
- Instrumentation
- IQ Balance
- Level Controllers
 - AGC
 - AGC2
 - AGC3
 - CTCSS Squelch
 - Feed Forward AGC
 - Moving Average
 - Mute
 - Power Squelch
 - Rail
 - Sample and Hold
 - Simple Squelch
 - Standard Squelch
 - Threshold
- Math Operators
- Measurement Tools
- Message Tools
- Misc
- Modulators
- Networking Tools
- OFDM

At the bottom, a terminal window shows the command prompt with the text: >>> Done

Narrowband Demodulator

Core > Modulators > NBFM Receive

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

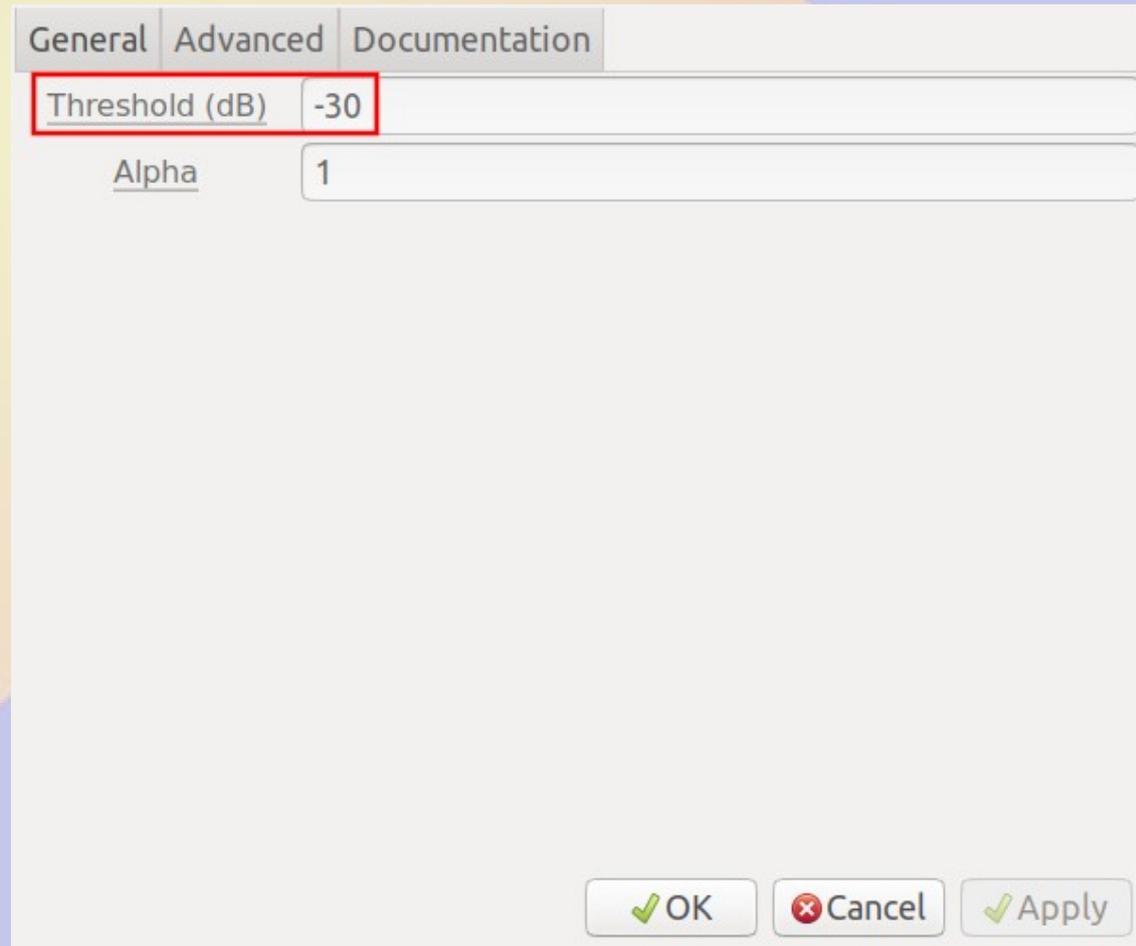
General	Advanced	Documentation
Audio Rate	25000	
Quadrature Rate	100000	
Tau	75e-6	
Max Deviation	3e3	

OK Cancel Apply

Squelch

Core > Level Controllers > Simple Squelch

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

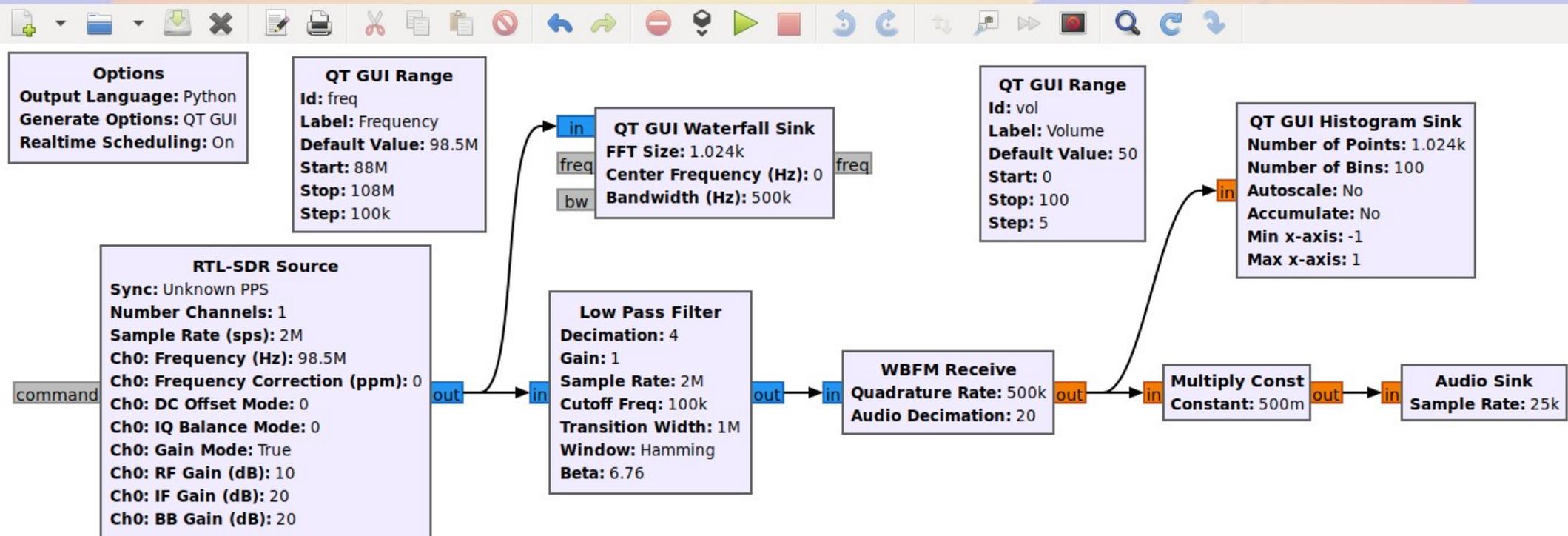


The screenshot shows a configuration dialog box with three tabs: "General", "Advanced", and "Documentation". The "General" tab is selected. It contains two input fields: "Threshold (dB)" with the value "-30" and "Alpha" with the value "1". The "Threshold (dB)" field is highlighted with a red border. At the bottom right, there are three buttons: "OK" (with a green checkmark), "Cancel" (with a red X), and "Apply" (with a green checkmark).

Field	Value
Threshold (dB)	-30
Alpha	1

Adding more usable controls

- 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**

Properties: QT GUI Range

General | Advanced | Documentation

Id	freq
Label	Frequency
Type	float ▾
Default Value	98.5e6
Start	88.0e6
Stop	108e6
Step	0.1e6
Widget	Counter + Slider ▾
Minimum Length	400
GUI Hint	

OK Cancel Apply

Properties: RTL-SDR Source

General | Advanced | Documentation

Output Type	Complex Float32 ▾
Device Arguments	
Sync	Unknown PPS ▾
Number MBoards	1 ▾
MB0: Clock Source	Default ▾
MB0: Time Source	Default ▾
Number Channels	1 ▾
Sample Rate (sps)	2e6
Ch0: Frequency (Hz)	freq
Ch0: Frequency Correction (ppm)	0
Ch0: DC Offset Mode	0 ▾
Ch0: IQ Balance Mode	0 ▾

OK Cancel Apply

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 (**$0.01 * \text{vol}$**)

Properties: QT GUI Range

General | Advanced | Documentation

Id	vol
Label	Volume
Type	float ▾
Default Value	50
Start	0
Stop	100
Step	5
Widget	Knob ▾
Minimum Length	200
GUI Hint	

OK Cancel Apply

Properties: Multiply Const

General | Advanced | Documentation

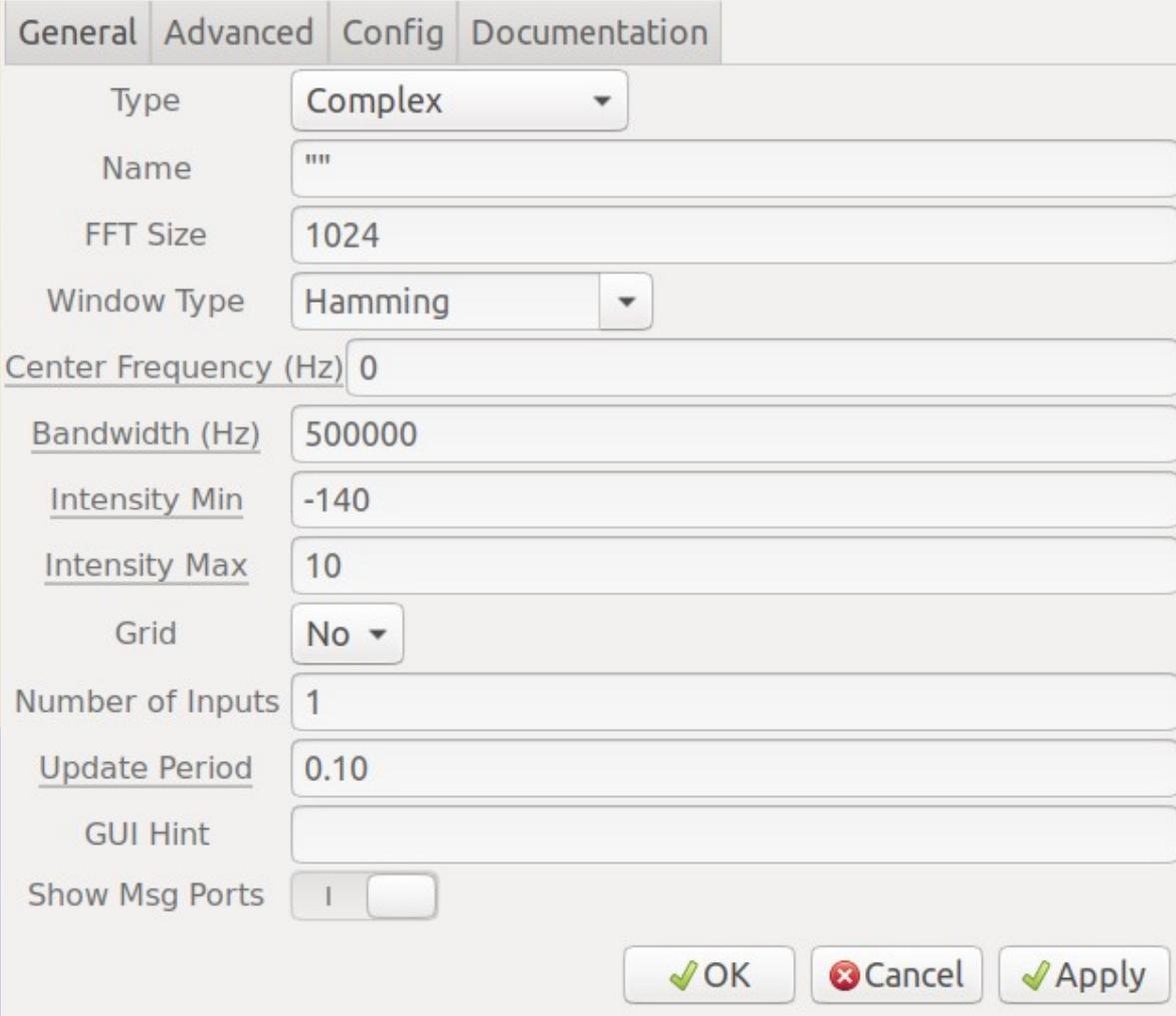
IO Type	float ▾
Constant	$0.01 * \text{vol}$
Vec Length	1

OK Cancel Apply

Waterfall

Core > Instrumentation > QT > QT GUI Waterfall Sink

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



The image shows a configuration dialog box for the QT GUI Waterfall Sink. The dialog has four tabs: General, Advanced, Config, and Documentation. The General tab is selected. The configuration parameters are as follows:

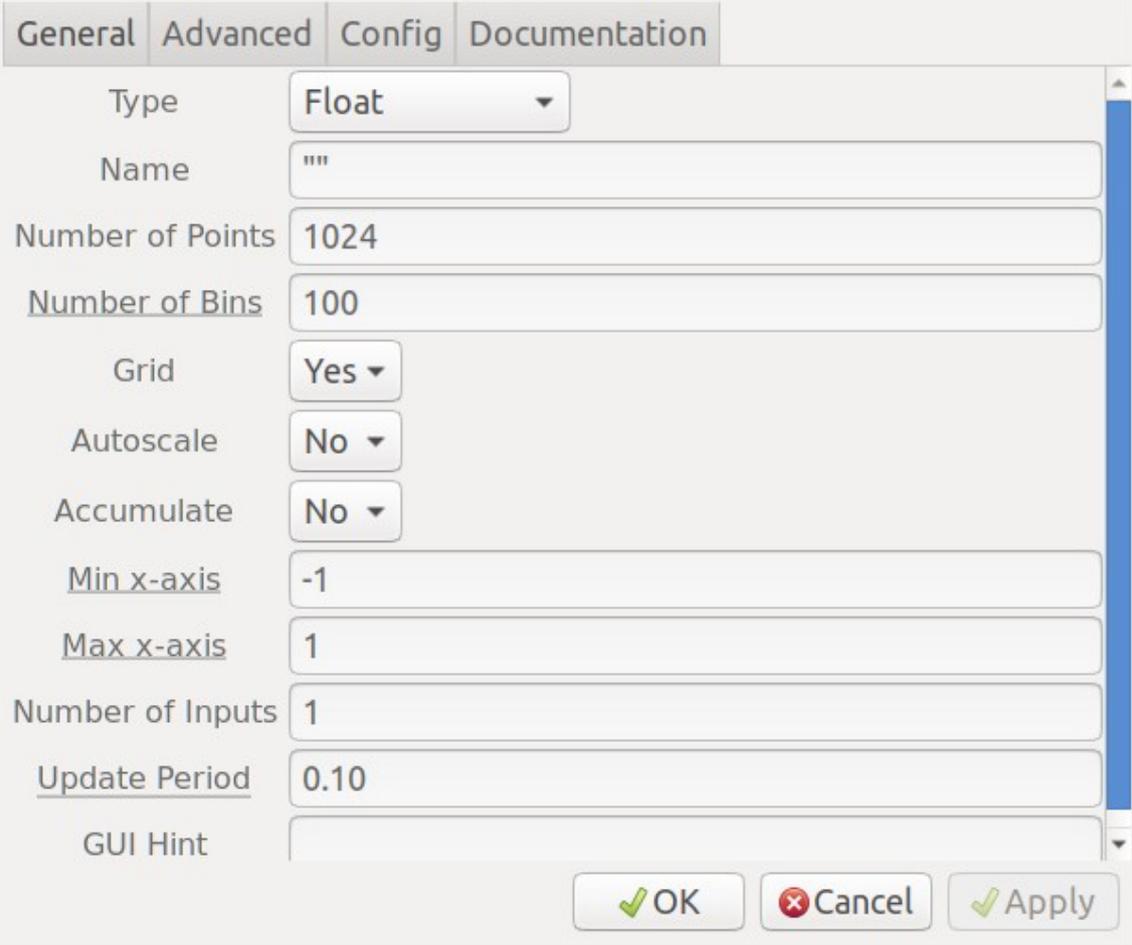
Parameter	Value
Type	Complex
Name	""
FFT Size	1024
Window Type	Hamming
Center Frequency (Hz)	0
Bandwidth (Hz)	500000
Intensity Min	-140
Intensity Max	10
Grid	No
Number of Inputs	1
Update Period	0.10
GUI Hint	
Show Msg Ports	<input type="checkbox"/>

At the bottom right of the dialog, there are three buttons: OK (with a green checkmark), Cancel (with a red X), and Apply (with a green checkmark).

Spectrum (Histogram)

Core > Instrumentation > QT > QT GUI Histogram Sink

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



The image shows the configuration dialog for the QT GUI Histogram Sink. The dialog has four tabs: General, Advanced, Config, and Documentation. The General tab is selected. The configuration parameters are as follows:

Parameter	Value
Type	Float
Name	""
Number of Points	1024
Number of Bins	100
Grid	Yes
Autoscale	No
Accumulate	No
Min x-axis	-1
Max x-axis	1
Number of Inputs	1
Update Period	0.10
GUI Hint	

At the bottom of the dialog, there are three buttons: OK (with a green checkmark), Cancel (with a red X), and Apply (with a green checkmark).

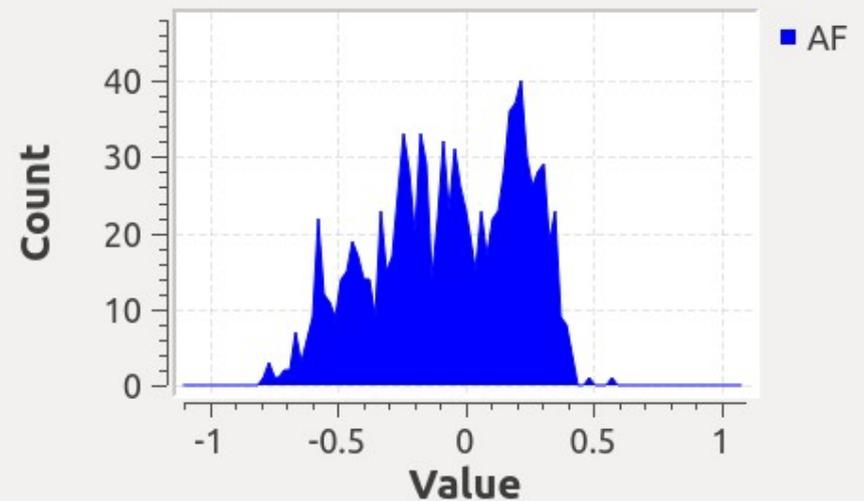
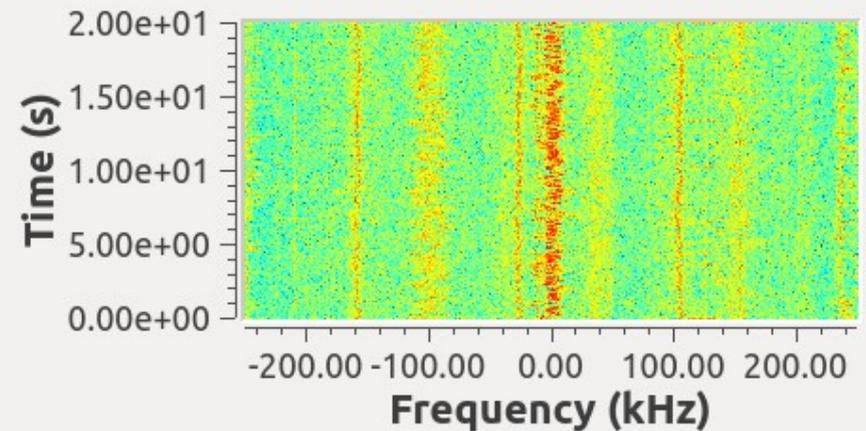
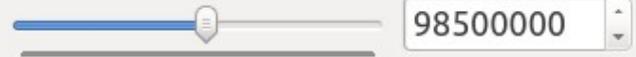
Run It

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

Volume

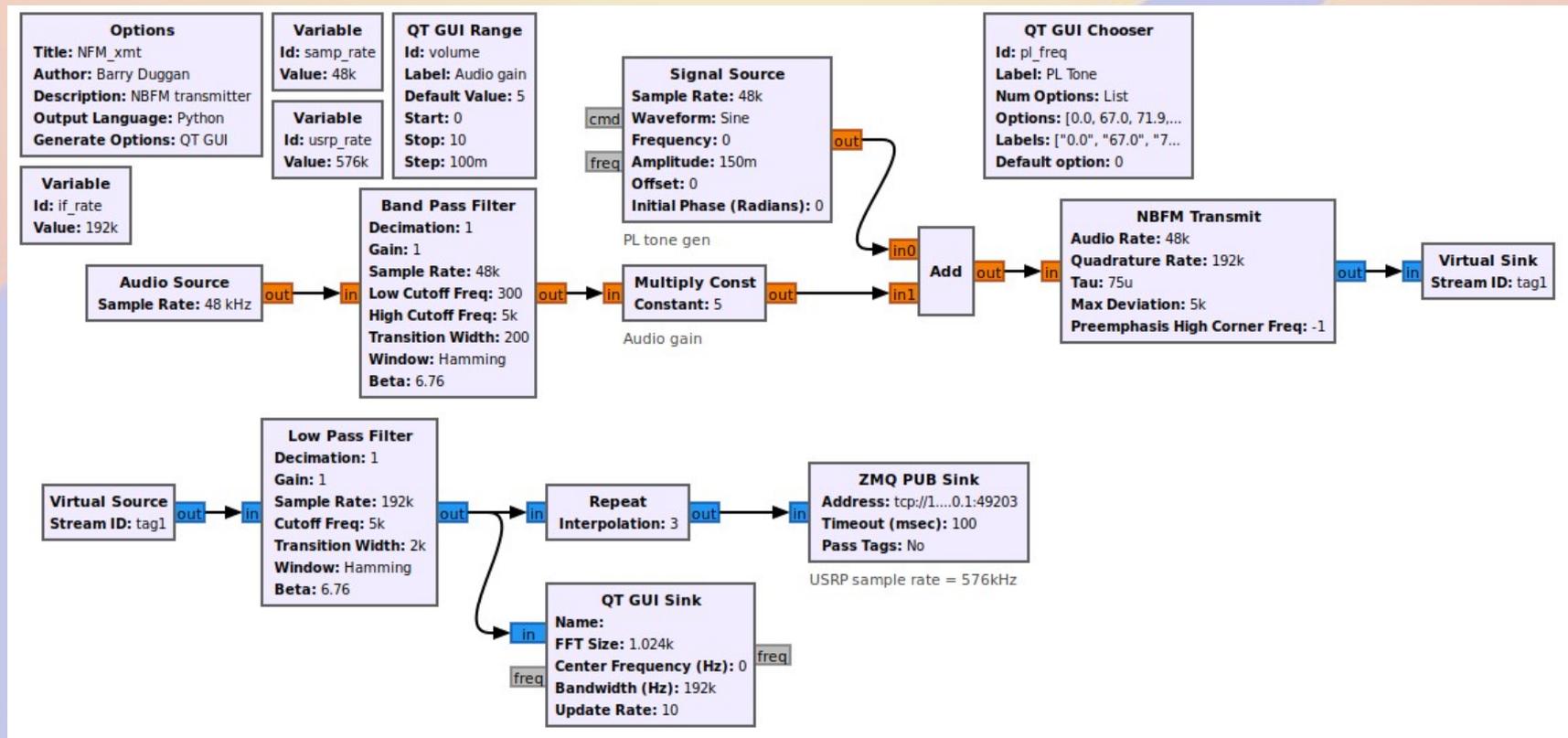


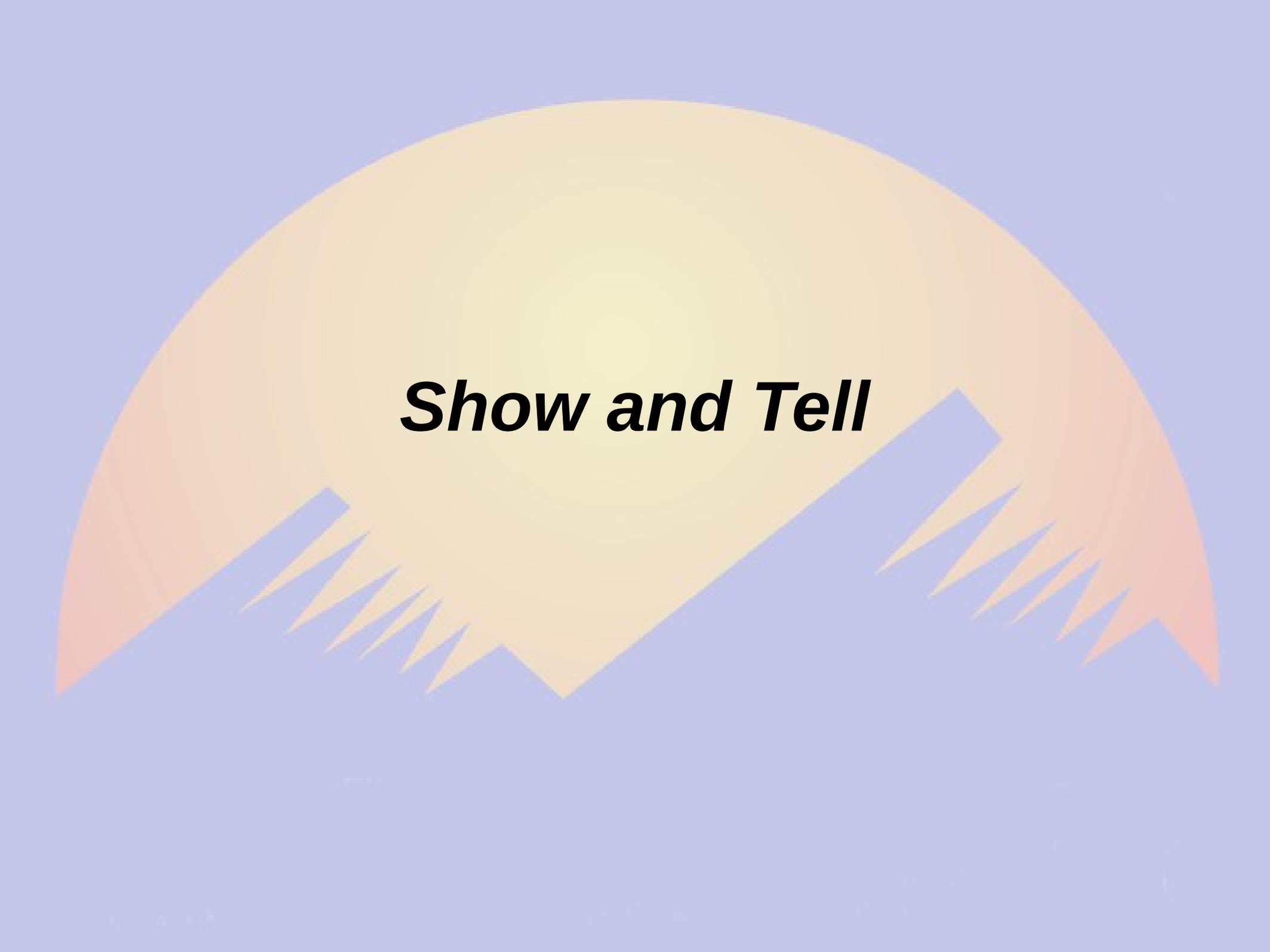
Frequency



Lots more possibilities....

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





Show and Tell