

GNU Radio Tutorials

Labs 1 – 5

Balint Seeber
Ettus Research

Version 1.0 (18th April 2014)

Comments & suggestions welcome:
balint@ettus.com
@spenchdotnet

Lab 1

- Open GNU Radio Companion:
 - Open a Terminal/Console/Command Prompt
 - Run 'gnuradio-companion'

Lab 1

Lab 1.grc - /home/balint/Desktop/Labs/Live/GRC - GNU Radio Companion

File Edit View Build Help

Options
ID: top_block
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 32k

Variable
ID: my_var
Value: 11

WX GUI Slider
ID: freq
Label: Frequency
Default Value: 1k
Minimum: 0
Maximum: 16k
Converter: Float

Signal Source
Sample Rate: 32k
Waveform: Cosine
Frequency: 1k
Amplitude: 1
Offset: 0

Canvas (the flowgraph construction area)

Throttle
Sample Rate: 32k

WX GUI Scope Sink
Title: Scope Plot
Sample Rate: 32k
Trigger Mode: Auto
Y Axis Label: Counts

Drag blocks from the Block list onto the canvas.

Block list – Press CTRL+F to search for a name

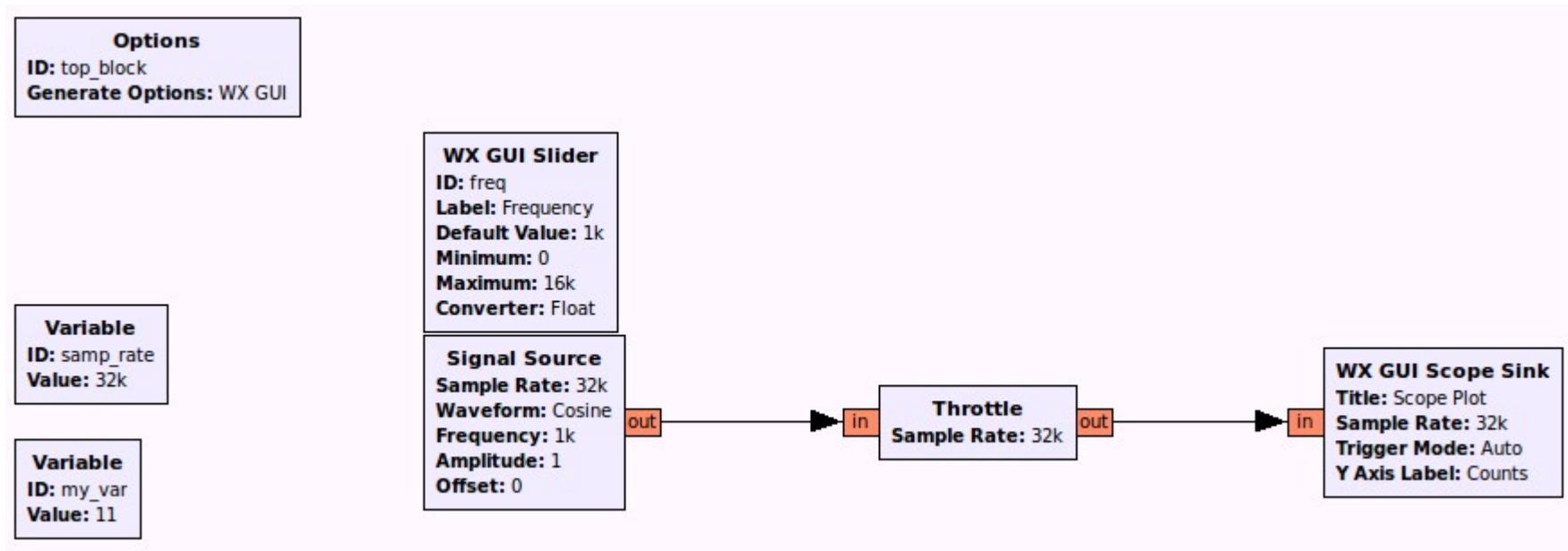
Connect ports by clicking on the chosen port of one block, and then click on the port of the other block. You can delete connections by clicking on the connection's line and pressing the Delete key.

Log window – keep an eye on this, as well as your terminal!

Showing: "/home/balint/Desktop/Labs/Live/GRC/Lab_4.1.grc"
Loading: "/home/balint/Desktop/Labs/Live/GRC/Lab_1.grc"
>>> Done
Showing: "/home/balint/Desktop/Labs/Live/GRC/Lab_1.grc"

- [ACARS]
- [Audio]
- [Boolean Operators]
- [Byte Operators]
- [Channelizers]
- [Channel Models]
- [Coding]
- [Control Port]
- [Debug Tools]
- [Deprecated]
- [digital]
- [DOA]
- [Equalizers]
- [Graphical Sinks]
- [gr-mac]
- [GUI Widgets]
- [IEEE802.11]
- [Impairment Models]
- [Instrumentation]
- [Level Controllers]
- [Math Operators]
- [Measurement Tools]
- [Message Tools]
- [Misc]
- [Modulators]
- [Networking Tools]
- [NOAA]

Lab 1



Create a sine wave & inspect the generated samples with a (time-domain) Scope Sink

Lab 1

(double click)

The diagram shows a block diagram with the following components:

- Options**
ID: top_block
Generate Options: WX GUI
- Variable**
ID: samp_rate
Value: 32k
- Variable**
ID: my_var
Value: 11
- WX GUI Slider**
ID: freq
Label: Frequency
Default Value: 1k
Minimum: 0
Maximum: 16k
Converter: Float
- Signal Source**
Sample Rate: 32k
Waveform: Cosine
Frequency: 1k
Amplitude: 1
Offset: 0

The 'Signal Source' block has an output port labeled 'out'.

The configuration window for the 'Options' block is shown on the right. It has two tabs: 'General' and 'Documentation'. The 'General' tab is active, showing the following settings:

- ID: top_block
- Title: (empty)
- Author: (empty)
- Description: (empty)
- Window Size: 1280, 1024
- Generate Options: WX GUI
- Run: Autostart
- Max Number of Output: 0
- Realtime Scheduling: Off

Buttons for 'Cancel' and 'OK' are at the bottom right.

'Options' block is used to set global parameters

Lab 1

General		Documentation
ID	top_block	Name of generated Python file
Title		Title of main GUI window, or name of Hierarchical block
Author		
Description		
Window Size	1280, 1024	GRC canvas size
Generate Options	WX GUI ▼	Type of code to generate (see next)
<u>Run</u>	Autostart	How to start & stop the flowgraph
Max Number of Output	0	Advanced: limit the number of samples output from each iteration of every block's work function
Realtime Scheduling	Off ▼	If code is run as 'root' (e.g. with 'sudo') ask OS kernel to prioritise this process

Lab 1

The image shows a software configuration window with two tabs: 'General' (selected) and 'Documentation'. The 'General' tab contains several fields and a dropdown menu. The fields are: 'ID' (value: 'top_block'), 'Title' (empty), 'Author' (empty), 'Description' (empty), 'Window Size' (value: '1280, 1024'), 'Generate Options' (dropdown menu), 'Run' (button), 'Max Number of Output' (empty), and 'Realtime Scheduling' (empty). The dropdown menu for 'Generate Options' is open, showing four options: 'WX GUI', 'QT GUI', 'No GUI', and 'Hier Block'. Arrows point from text boxes on the right to these options: 'WX GUI' points to 'GUI app using WX toolkit (use WX GUI blocks)', 'QT GUI' points to 'GUI app using Qt toolkit (use Qt GUI blocks)', 'No GUI' points to 'Command-line app without GUI (text-based, run in a console)', and 'Hier Block' points to a larger box describing hierarchical blocks.

Field	Value
ID	top_block
Title	
Author	
Description	
Window Size	1280, 1024
Generate Options	WX GUI, QT GUI, No GUI, Hier Block
Run	Run
Max Number of Output	
Realtime Scheduling	

GUI app using WX toolkit (use WX GUI blocks)

GUI app using Qt toolkit (use Qt GUI blocks)

Command-line app without GUI (text-based, run in a console)

Create a **Hierarchical** block that will appear in the block list (a reusable component, not an app – use Pad Source/Sink blocks to expose ports, and Parameter blocks to expose configuration variables)

Lab 1

General Documentation

ID top_block

Title

Author

Description

Window Size 1280, 1024

Generate Options WX GUI ▼

Run Autostart ▼

Max Number of Output Autostart

Realtime Scheduling Off

Autostart —▶ Automatically start flowgraph

Off —▶ Do not automatically start flowgraph

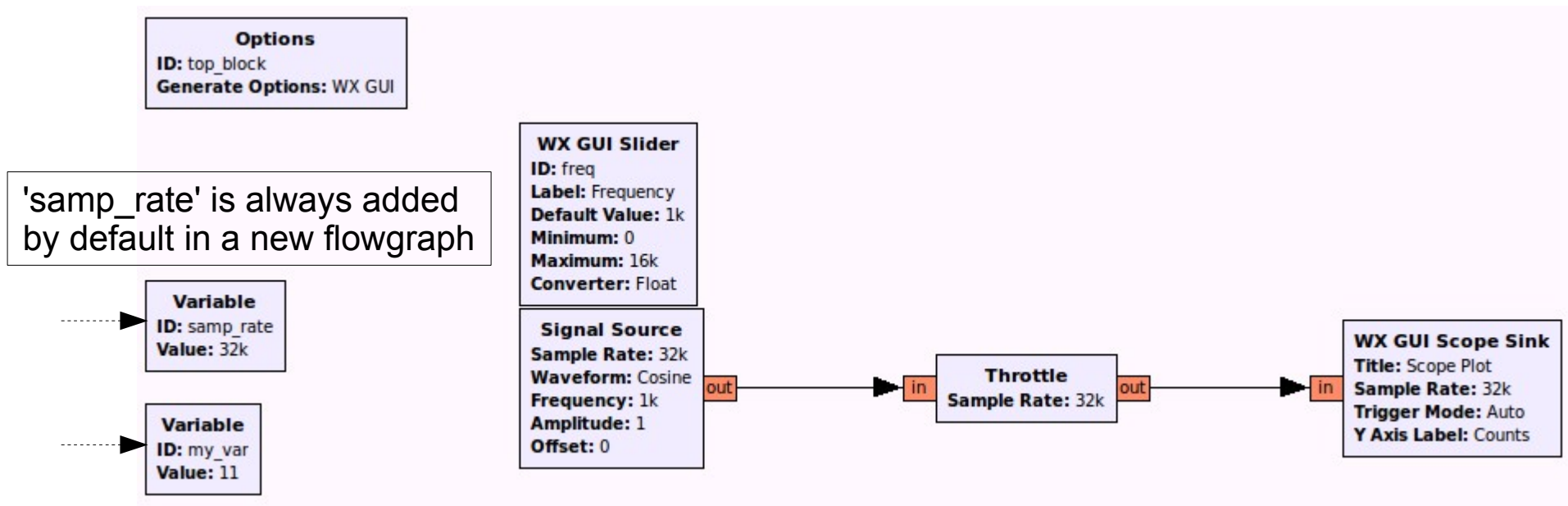
Lab 1

The image shows a configuration window with two tabs: 'General' and 'Documentation'. The 'General' tab is selected. The configuration fields are as follows:

Field	Value
ID	top_block
Title	
Author	
Description	
Window Size	1280, 1024
Generate Options	No GUI
Run Options	Run to Completion
Max Number of Output	Prompt for Exit
Realtime Scheduling	Off

The 'Generate Options' dropdown menu is open, showing three options: 'No GUI', 'Run to Completion', and 'Prompt for Exit'. The 'Run to Completion' option is highlighted, and a tooltip indicates 'Will automatically exit if/when done'. The 'Prompt for Exit' option is also highlighted, and a tooltip indicates 'Pressing ENTER will exit'.

Lab 1



Variable: a block that contains an arbitrary Python expression.

You can refer to it in another block by its **ID**.

Lab 1

Options
ID: top_block
Generate Options: WX GUI

WX GUI SI
ID: freq
Label: Frequ
Default Valu
Minimum: 0
Maximum: 1
Converter:

Variable
ID: samp_rate
Value: 32k

(double click)

Variable
ID: my_var
Value: 11

Signal S
Sample Rat
Waveform:
Frequency:
Amplitude:
Offset: 0

General Documentation

ID	samp_rate
<u>Value</u>	32000

ID: (Python) variable name
Value: arbitrary Python expression, e.g.

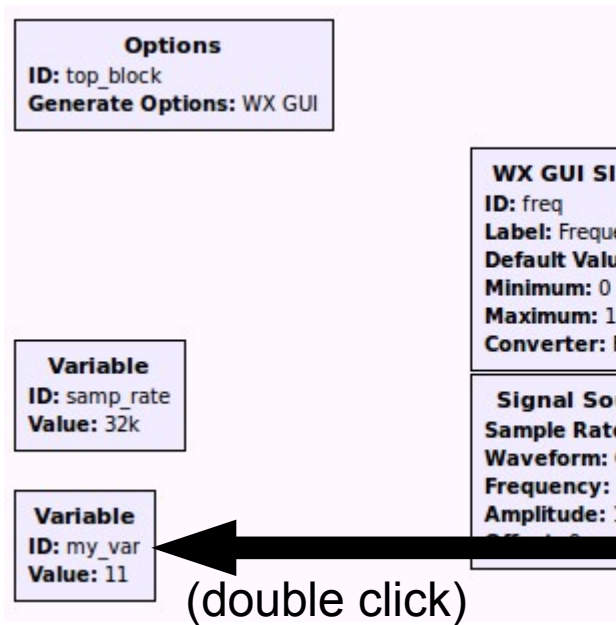
32000 (the default): an integer

32e6: 32000.0 (floating-point number)

int(32e6): 32000 (integer cast of floating-point number)

Cancel OK

Lab 1



'my_var' is just for show here
(it doesn't actually do anything
useful in this flowgraph).

ID	Value
my_var	5 + 6

Key: value
Type: raw
Value: 11

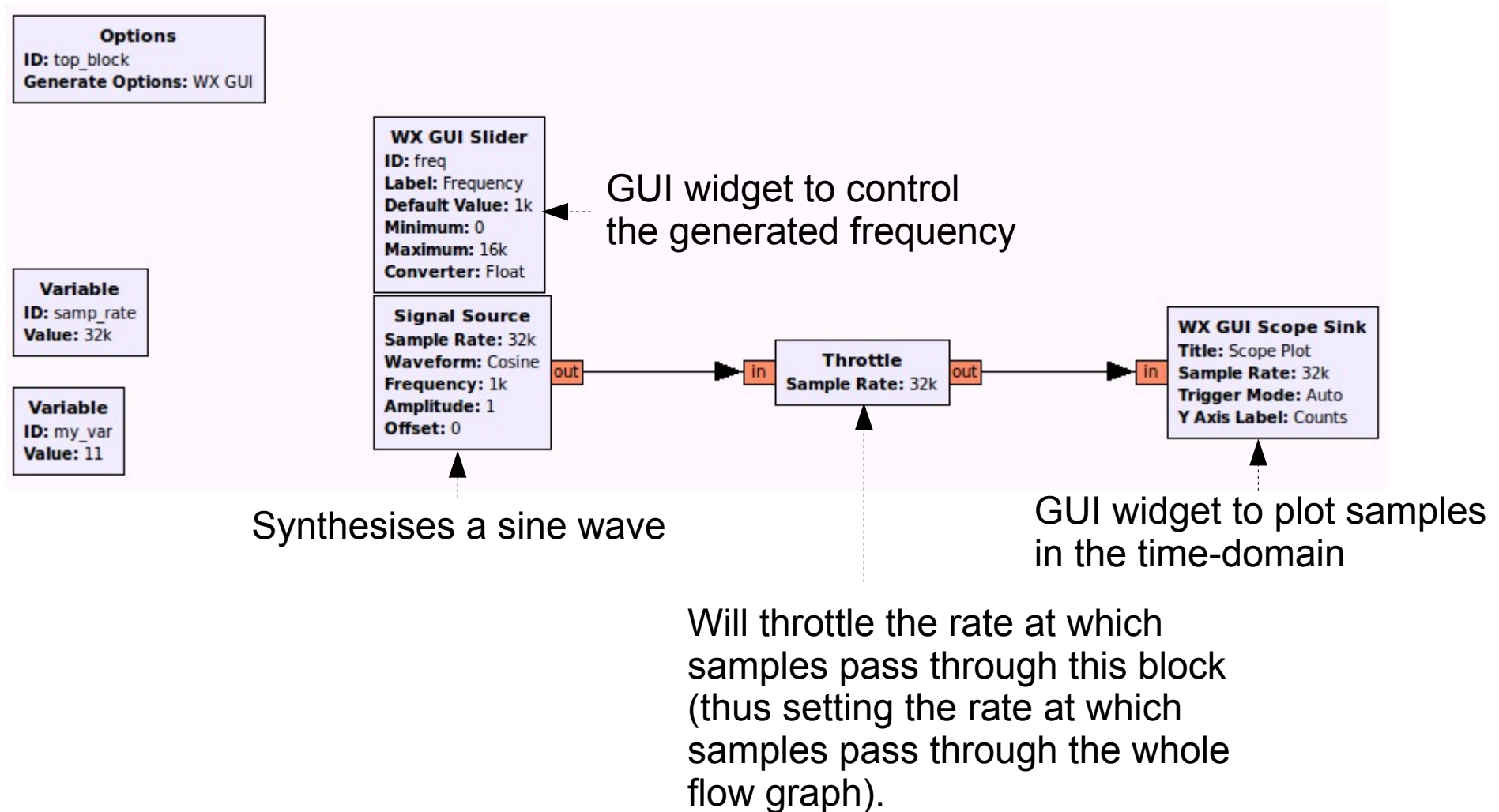
Another example of a simple arbitrary Python expression.

Hover the cursor over any parameter field and the tooltip will show you the expression's *evaluated* result (here $5 + 6 = 11$)

Note: arbitrary expressions can *only* be written into fields that have a white background ('raw' fields).

Cancel OK

Lab 1



Lab 1

Options
ID: top_block
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 32k

WX GUI Slider
ID: freq
Label: Frequency
Default Value: 1k
Minimum: 0
Maximum: 16k
Converter: Float

Signal Source
Sample Rate: 32k
Waveform: Cosine
Frequency: 1k
Amplitude: 1
Offset: 0

Throttle
Sample Rate: 32k

WX GUI Scope Sink
Title: Scope Plot
Sample Rate: 32k
Trigger Mode: Auto
Y Axis Label: Counts

Any *processing* block's 'Sample Rate' parameter is used for DSP calculation, **not** for controlling the rate at which samples are produced. This is distinct from a *hardware* (or Throttle) block where it **is** used to control sample flow.

An underline indicates changing the parameter via any dependent variable will cause the block to trigger an internal callback and update its state (i.e. perform a real-time parameter change)

Properties: Signal Source

General Advanced Documentation

ID	analog_sig_source_x_0	
Output Type	Float	Type of sample (sets port colour)
<u>Sample Rate</u>	samp_rate	
<u>Waveform</u>	Cosine	Type of signal
<u>Frequency</u>	freq	Frequency (here it's linked to the slider)
<u>Amplitude</u>	1	
<u>Offset</u>	0	Phase offset

Sample Rate (DSP)

- If calculating a sine wave where a given frequency *in Hertz* is desired, you actually need to know the sample rate too. This is because the mathematical representation requires both values to calculate the individual sample amplitude at any specific point in time.
- The actual sample rate value used can be anything. It just so happens you'll usually use the same value as in the rest of your flowgraph so that everything will be consistent (operate in the same sample rate domain).

Sample Rate (DSP)

- Think of it as being used to calculate the discrete step size from one sample to the next within a DSP operation (e.g. the time step when calculating the amplitude of the next sample in the sine wave generator)

Sample Rate (Hardware)

- Distinct from mathematical (DSP) calculation, sample rate also refers to the rate at which samples pass through the flowgraph.
- If there is no rate control, hardware clock or throttling mechanism, the samples will be generated, pass through the flowgraph and be consumed as fast as possible (i.e. the flowgraph will be CPU bound).
- This is desirable if you want to perform some fixed DSP on stored data as quickly as possible (e.g. read from a file, resample and write it back).

Sample Rate (Hardware)

- Only a block that represents some underlying hardware with its own clock (e.g. USRP, sound card), or the Throttle Block, will use 'Sample Rate' to set that hardware clock, and therefore have the effect of applying rate control to the samples in the flowgraph.
- A Throttle Block will simply apply host-based timing (against the 'wall clock') to control the rate of the samples it produces (i.e. samples that it makes available on its outputs to downstream blocks).

Sample Rate (Hardware)

- A hardware Sink block will consume samples at a fixed rate (relative to the wall clock)
- The Throttle Block, or a hardware Sink block, will apply 'back pressure' to the upstream blocks (the rate of work of the upstream blocks will be limited by the throttling effect of this rate-controlling block)
- A hardware Source block will produce samples at a fixed rate (relative to the wall clock)

Sample Rate (Hardware)

- In general, there should only ever be one block in a flowgraph that has the ability to throttle sample flow.
- Otherwise you need to be very careful with multiple, unsynchronised clock sources: they will eventually go out of sync and cause overflows/underruns as their production/consumption rates will differ.
 - This is the 'two clock' problem (discussed later)
 - Work arounds: allow non-blocking I/O, and/or tweak resampling rates to account for the clock offsets

Lab 1

A port's colour indicates the type of samples flowing through the port. The colours also apply to block parameter fields.

Options
ID: top_block
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 32k

Variable
ID: my_var
Value: 11

WX GUI Slider
ID: freq
Label: Frequency
Default Value: 1k
Minimum: 0
Maximum: 16k
Converter: Float

Signal Source
Sample Rate: 32k
Waveform: Cosine
Frequency: 1k
Amplitude: 1
Offset: 0

Real single-precision
floating-point values

Throttle
Sample Rate: 32k

Color Mapping

Complex Float 64
Complex Float 32
Complex Integer 64
Complex Integer 32
Complex Integer 16
Complex Integer 8
Float 64
Float 32
Integer 64
Integer 32
Integer 16
Integer 8
Message Queue
Async Message
Bus Connection
Wildcard

Close

Title: Scope Plot
Sample Rate: 32k
Trigger Mode: Auto
Y Axis Label: Counts

Properties: Signal Source

General Advanced Documentation

ID analog_sig_source_x_0

Output Type Float

Sample Rate samp_rate

Waveform Cosine

Frequency freq

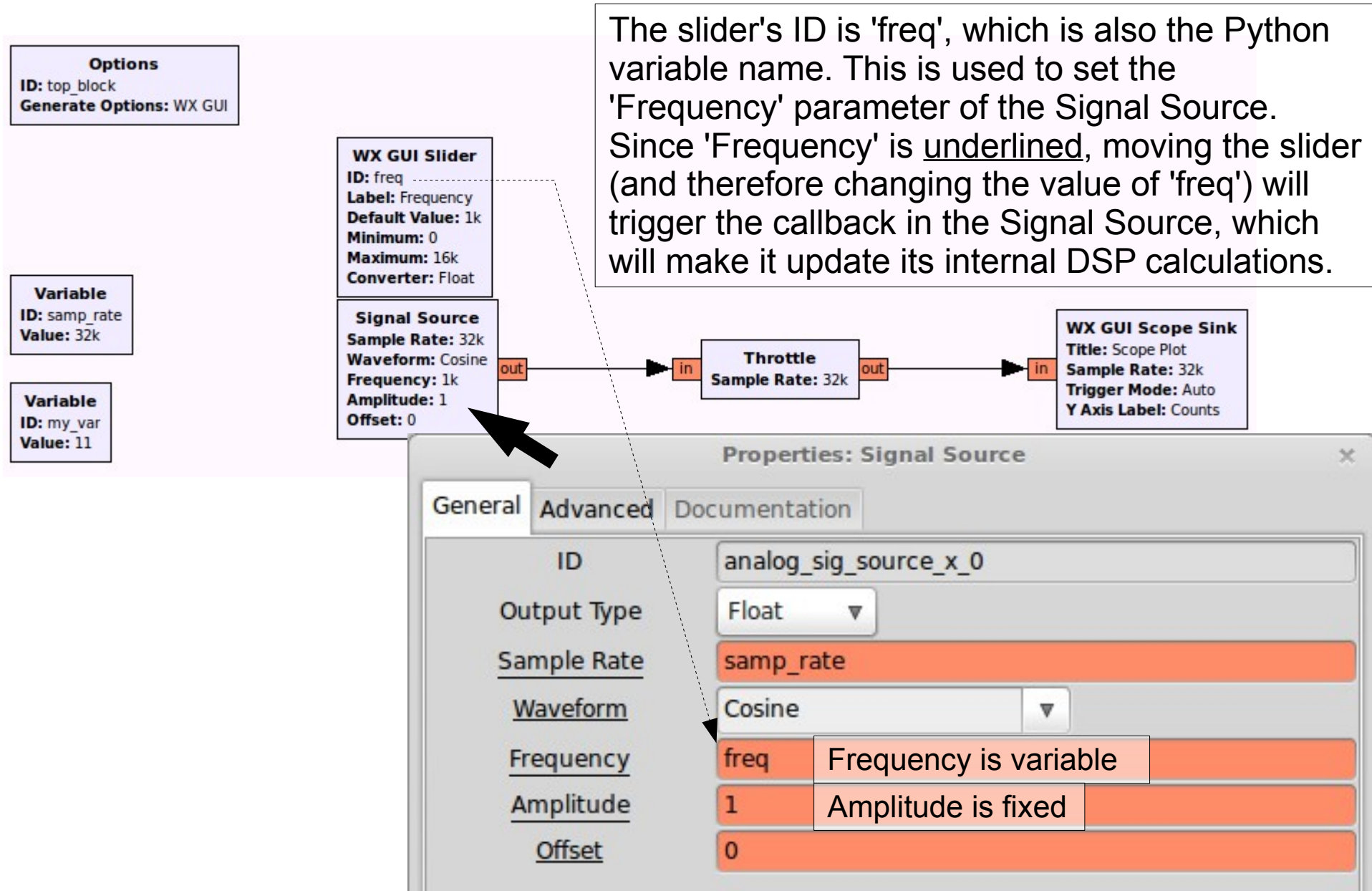
Amplitude 1

Offset 0

Tip:

After single-clicking on block, press the up/down arrow keys to change the type (this actually steps through options in the block's first available parameter).

Lab 1



Lab 1

Options
ID: top_block
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 32k

Variable
ID: my_var
Value: 11

WX GUI Slider
ID: freq
Label: Frequency
Default Value: 1k
Minimum: 0
Maximum: 16k
Converter: Float

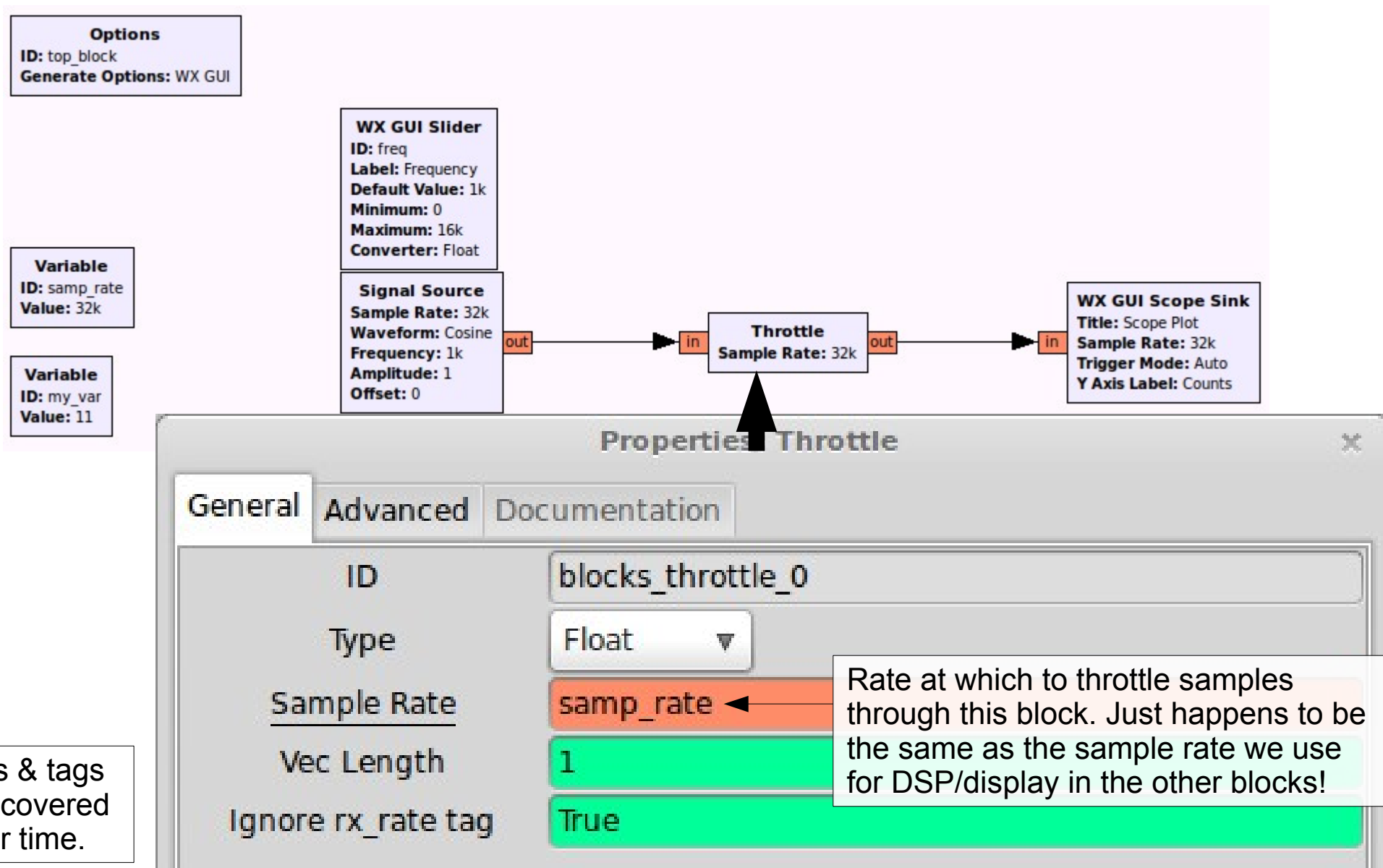
Signal
Sample
Wavefo
Frequen
Amplitu
Offset:

Properties: WX GUI Slider

General Documentation

ID	freq	
Label	Frequency	Label next to widget in the GUI
Default Value	1e3	1000.0 in scientific notation
Minimum	0	
Maximum	16e3	
Num Steps	1000	
Style	Horizontal ▼	
Converter	Float ▼	Whether 'freq' should be a floating-point number, or an integer
Grid Position		
Notebook		

Lab 1



Lab 1

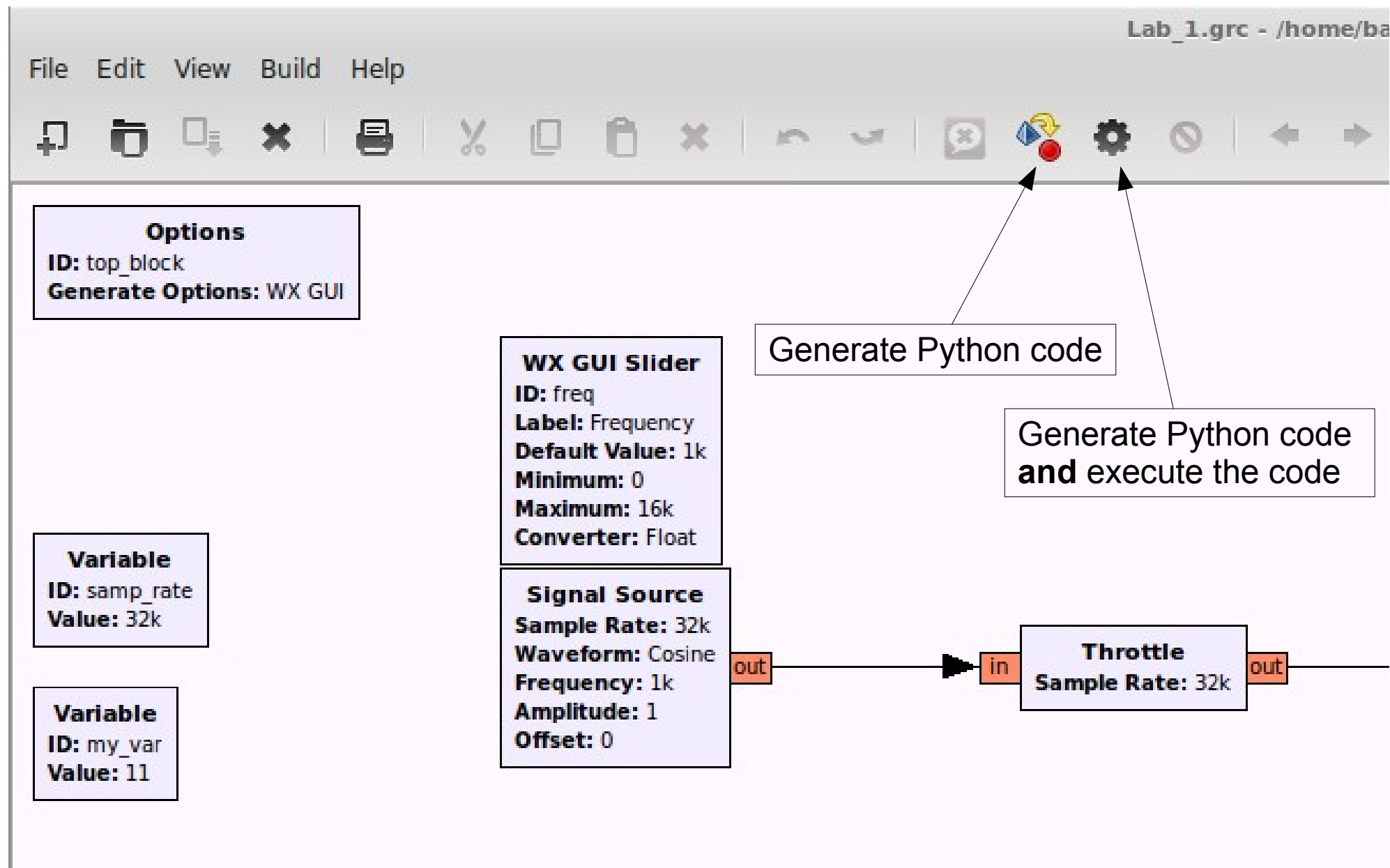
The screenshot shows the 'Properties: WX GUI Scope Sink' dialog box with the 'General' tab selected. The dialog contains various configuration options for a scope sink. Annotations are present to explain certain settings:

- Sample Rate:** The 'samp_rate' field is highlighted in orange. A text box explains: "This is purely for generating the correct step sizes on the drawn X-axis!".
- V Scale, V Offset, T Scale:** These fields are highlighted in orange and set to '0'. A text box explains: "0 will cause the plot to auto-scale to the incoming signal. Entering any other value will set it to a fixed scale/offset in that dimension.".
- Num Inputs:** The '1' field is highlighted in green. A text box explains: "Plot multiple signals (they may not be synchronised when drawn*!)".
- Window Size, Grid Position, Notebook:** These fields are highlighted in green. A text box explains: "These will be covered later, but for now have a look at the Documentation tab where they are discussed."

On the right side, a block diagram shows a signal entering an 'in' port of a 'WX GUI Scope Sink' block. A callout box for this block lists its properties: Title: Scope Plot, Sample Rate: 32k, Trigger Mode: Auto, Y Axis Label: Counts.

* Plot two Float streams in sync by changing Scope's Type to Complex, and use Float to Complex block beforehand.

Lab 1



Python code generated by GRC

Lab 1

```
#!/usr/bin/env python
#####
# Gnuradio Python Flow Graph
# Title: Top Block
# Generated: Wed Apr 16 14:11:52 2014
#####

from gnuradio import analog
from gnuradio import blocks
from gnuradio import eng_notation
from gnuradio import gr
from gnuradio import wxgui
from gnuradio.eng_option import eng_option
from gnuradio.filter import firdes
from gnuradio.wxgui import forms
from gnuradio.wxgui import scopesink2
from grc_gnuradio import wxgui as grc_wxgui
from optparse import OptionParser
import wx

class top_block(grc_wxgui.top_block_gui):

    def __init__(self):
        grc_wxgui.top_block_gui.__init__(self, title="Top Block")

        #####
        # Variables
        #####
        self.samp_rate = samp_rate = 32000
        self.my_var = my_var = 5 + 6
        self.freq = freq = 1e3

        #####
        # Blocks
        #####
        self._freq_size = wx.BoxSizer(wx.VERTICAL)
        self._freq_text_box = forms.text_box(
            parent=self.GetWin(),
            size=_freq_size,
            value=self.freq,
            callback=self.set_freq,
            label="Frequency",
            converter=forms.float_converter(),
            proportion=0,
        )
        self._freq_slider = forms.slider(
            parent=self.GetWin(),
            size=_freq_size,
            value=self.freq,
            callback=self.set_freq,
            minimum=0,
            maximum=16e3,
            num_steps=1000,
            style=wx.SL_HORIZONTAL,
            cast=float,
            trig_mode=wxgui.TRIG_MODE_AUTO,
            y_axis_label="Counts",
        )
        self.Add(self.wxgui_scopesink2_0.win)
        self.blocks_throttle_0 = blocks.throttle(gr.sizeof_float*1, samp_rate, True)
        self.analog_sig_source_x_0 = analog.sig_source_f(samp_rate, analog.GR_COS_WAVE, freq, 1, 0)

        #####
        # Connections
        #####
        self.connect((self.analog_sig_source_x_0, 0), (self.blocks_throttle_0, 0))
        self.connect((self.blocks_throttle_0, 0), (self.wxgui_scopesink2_0, 0))

    # QT sink close method reimplementation

    def get_samp_rate(self):
        return self.samp_rate

    def set_samp_rate(self, samp_rate):
        self.samp_rate = samp_rate
        self.analog_sig_source_x_0.set_sampling_freq(self.samp_rate)
        self.wxgui_scopesink2_0.set_sample_rate(self.samp_rate)
        self.blocks_throttle_0.set_sample_rate(self.samp_rate)

    def get_my_var(self):
        return self.my_var

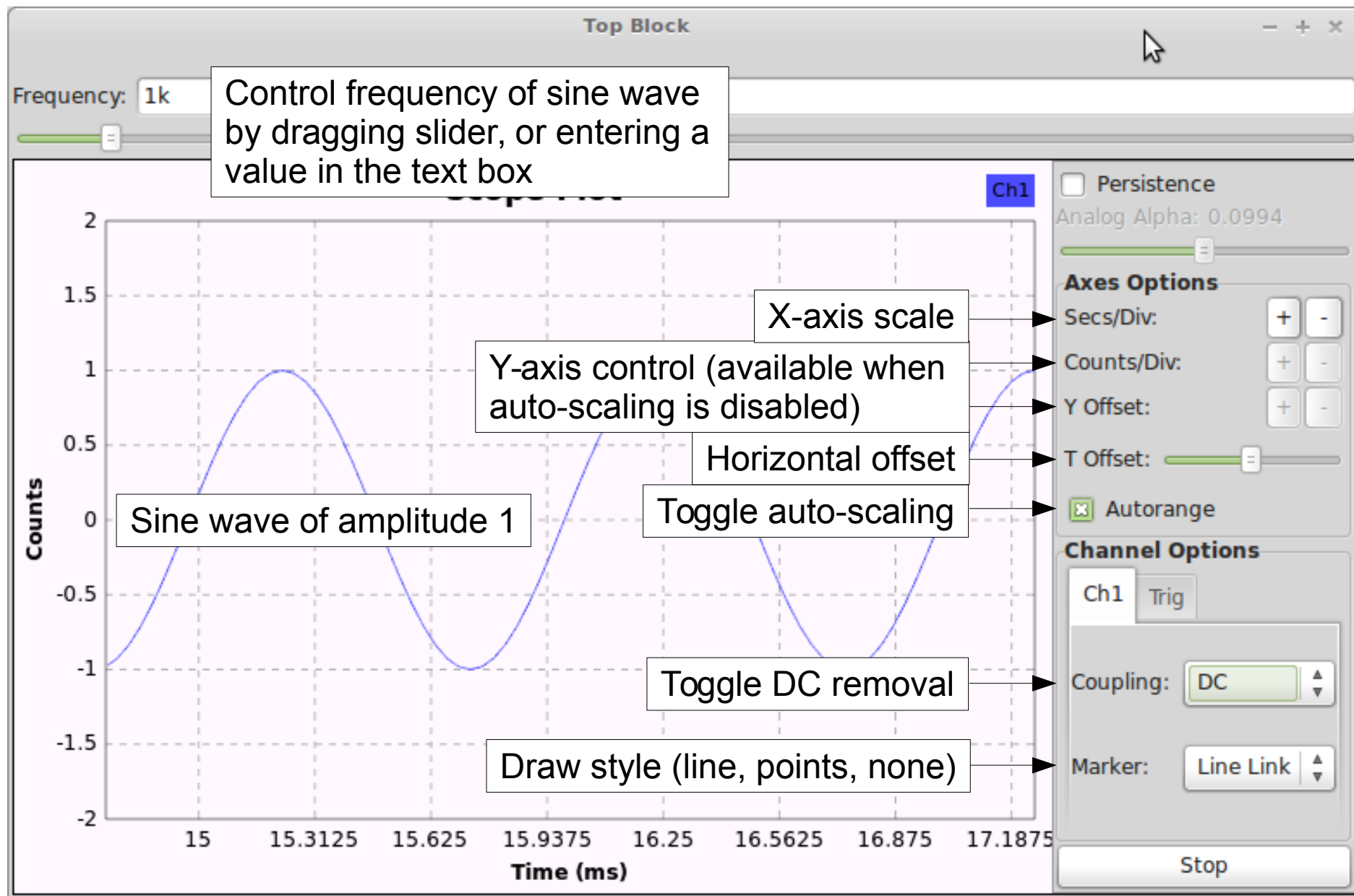
    def set_my_var(self, my_var):
        self.my_var = my_var

    def get_freq(self):
        return self.freq

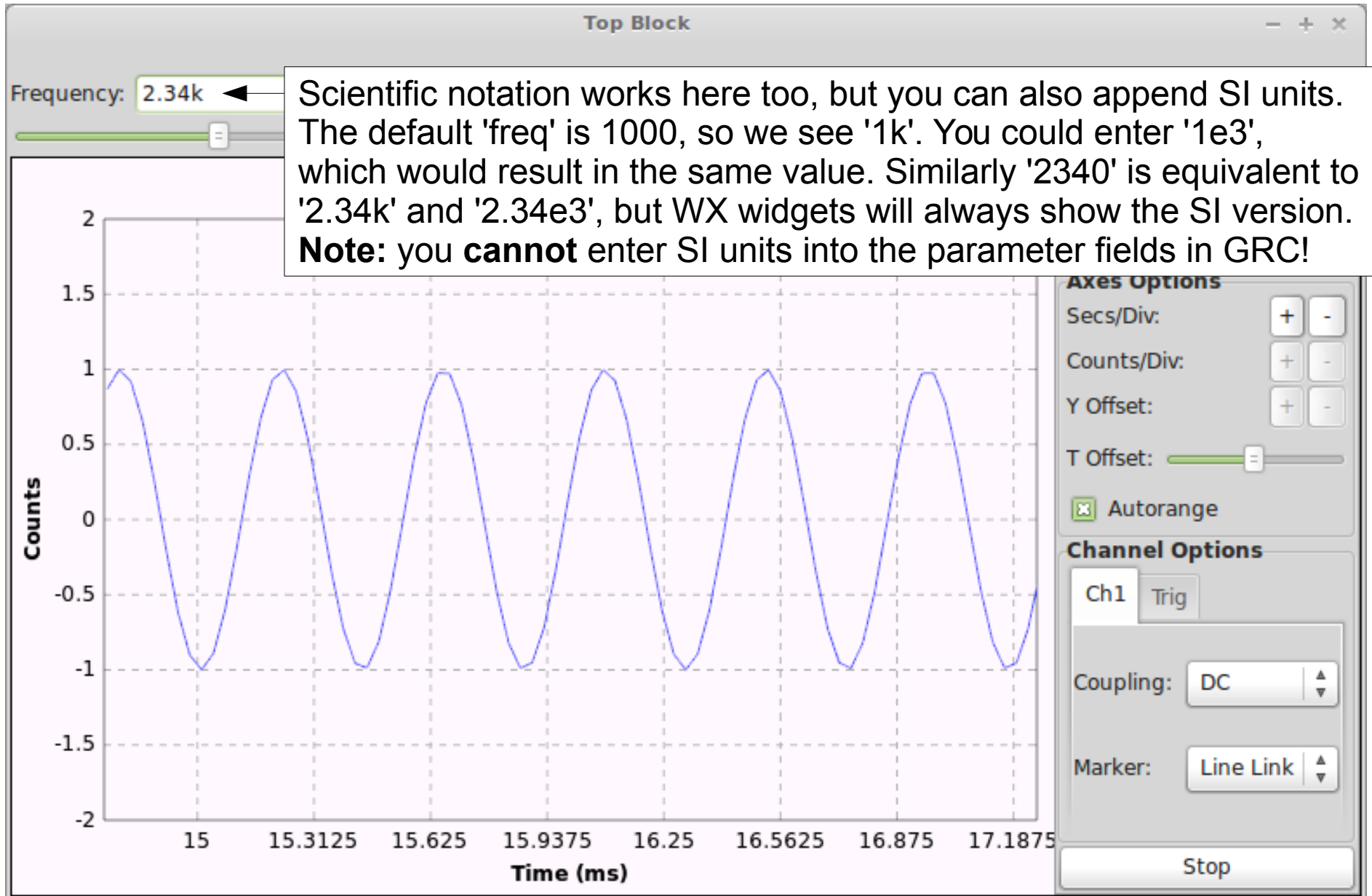
    def set_freq(self, freq):
        self.freq = freq
        self.analog_sig_source_x_0.set_frequency(self.freq)
        self._freq_slider.set_value(self.freq)
        self._freq_text_box.set_value(self.freq)

if __name__ == '__main__':
    import ctypes
    import sys
    if sys.platform.startswith('linux'):
        try:
            x11 = ctypes.cdll.LoadLibrary('libX11.so')
            x11.XInitThreads()
        except:
            print "Warning: failed to XInitThreads()"
    parser = OptionParser(option_class=eng_option, usage="%prog: [options]")
    (options, args) = parser.parse_args()
    tb = top_block()
    tb.Start(True)
    tb.Wait()
```

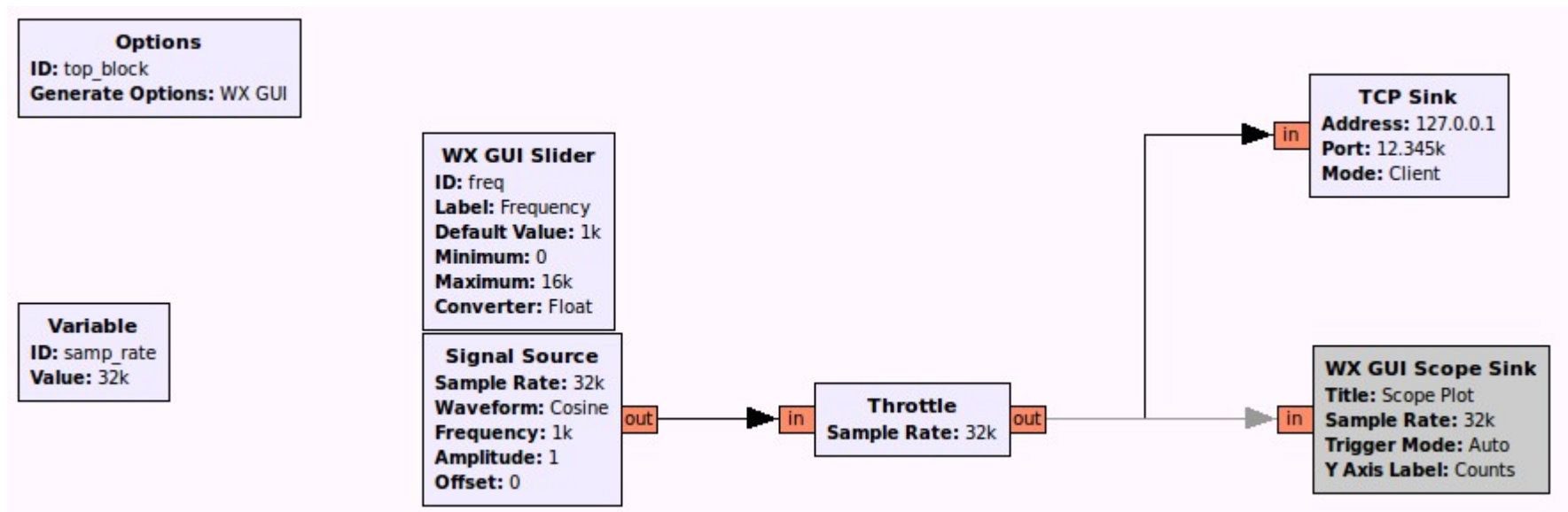
Lab 1



Lab 1

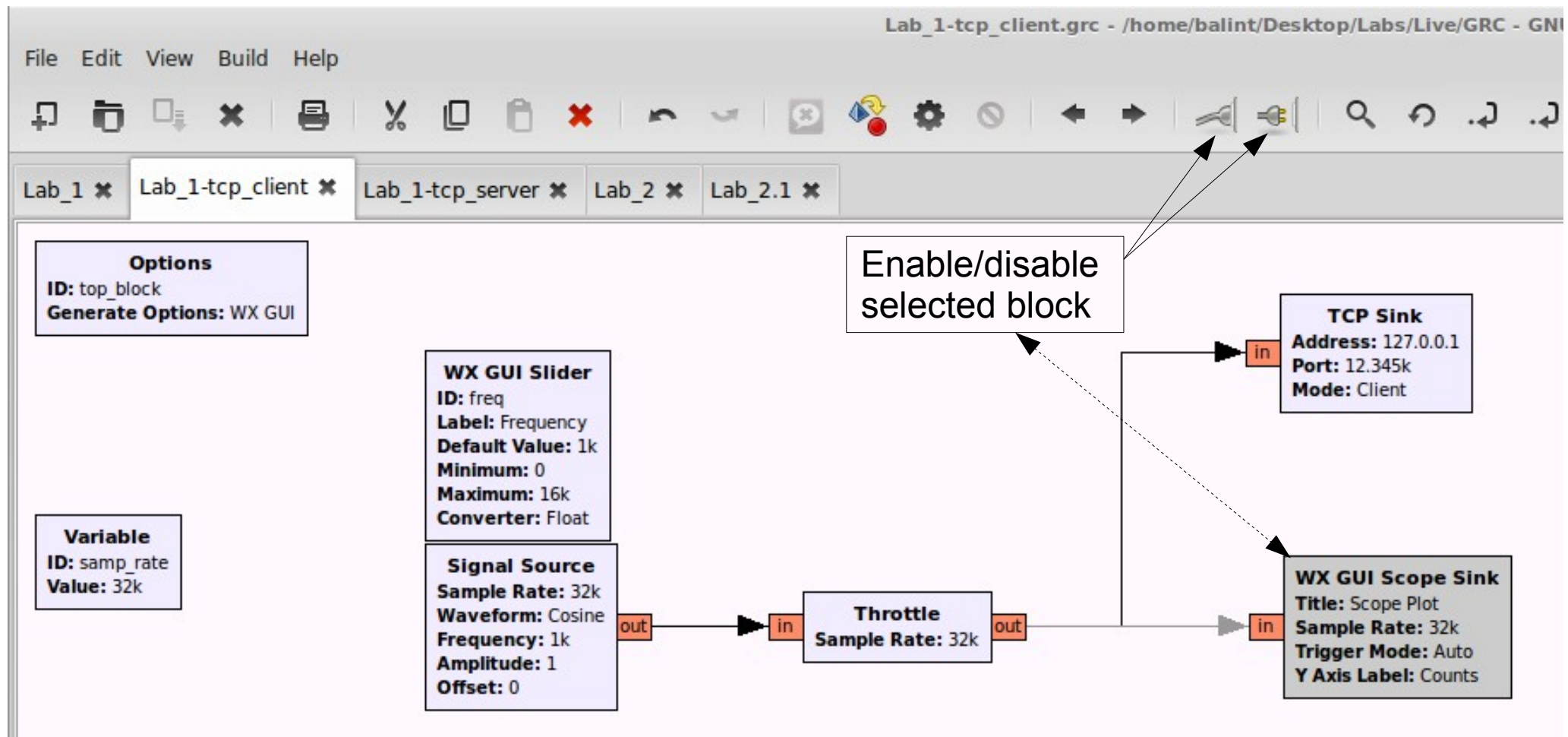


Lab 1: TCP Client (producer)



Create a sine wave & transmit generated samples over a TCP connection

Lab 1: TCP Client (producer)



Lab 1: TCP Client (producer)

WX GUI Scope Sink

Title: Scope Plot

Sample Rate: 32k

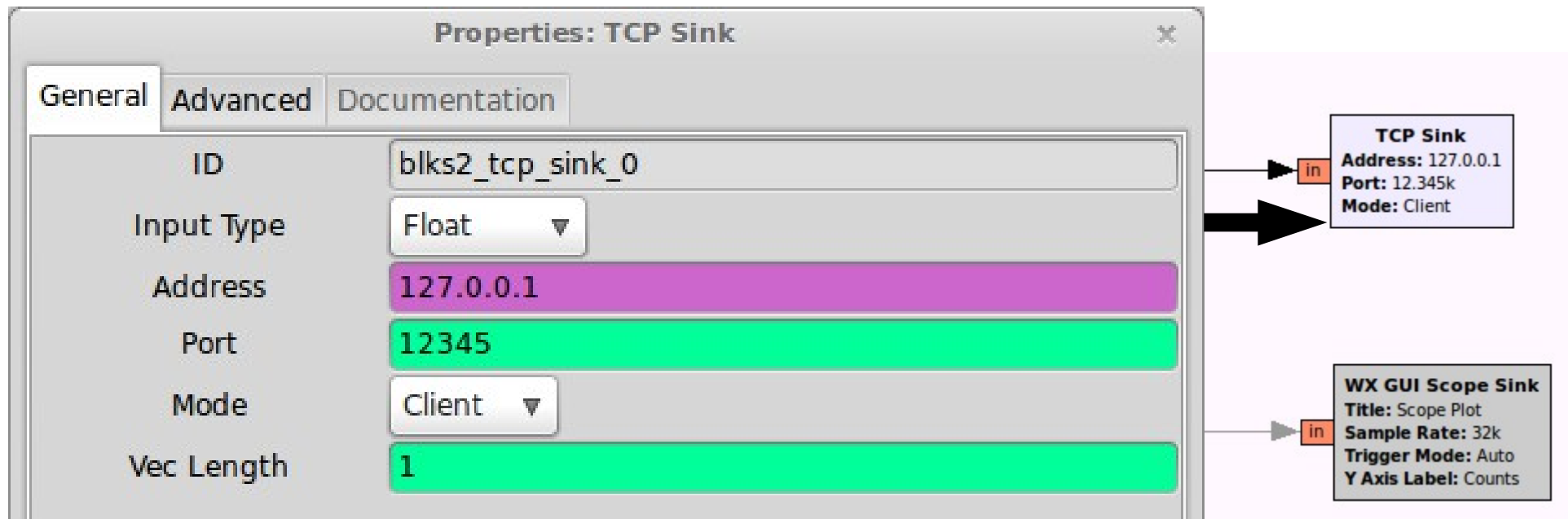
Trigger Mode: Auto

Y Axis Label: Counts

- Cut Ctrl+X
- Copy Ctrl+C
- Paste Ctrl+V
- Delete Delete
- Rotate Counterclockwise Left
- Rotate Clockwise Right
- Enable E
- Disable D
- Create Hier
- Open Hier
- Toggle Source Bus
- Toggle Sink Bus
- Properties Return

Tip:
Make note of the
keyboard shortcuts
in the block
context menu.

Lab 1: TCP Client (producer)

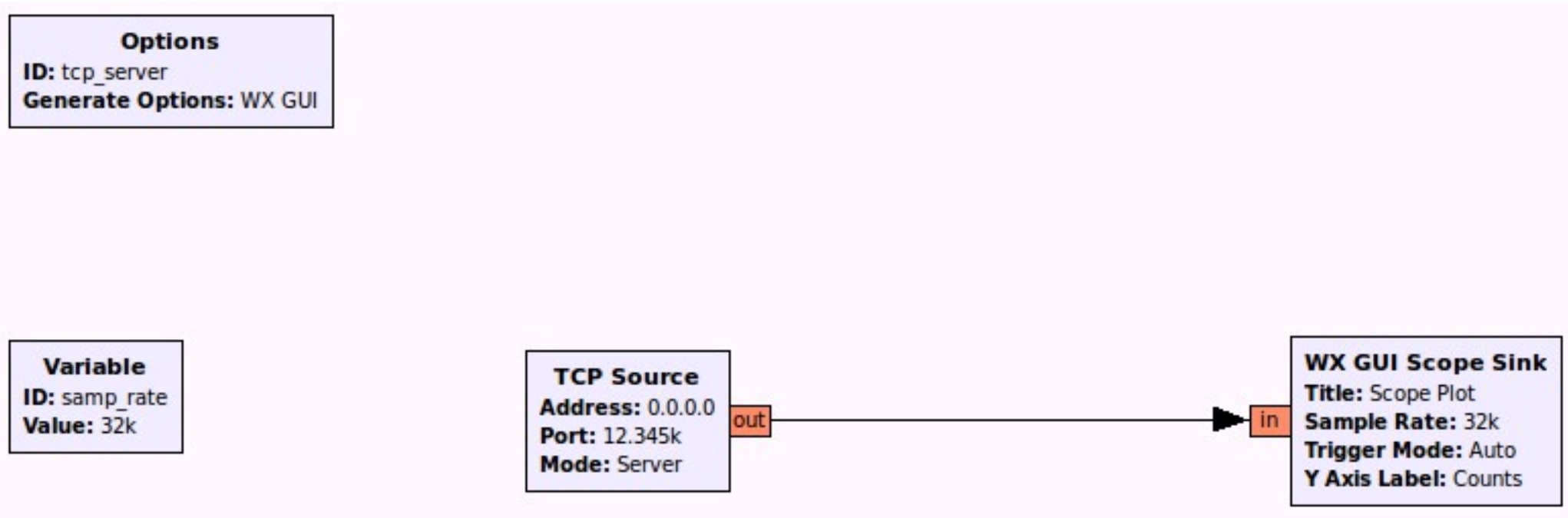


Tip:

The flowgraph will not start unless a TCP connection is established. If the TCP connection fails, a Python exception will be thrown and program will not start.

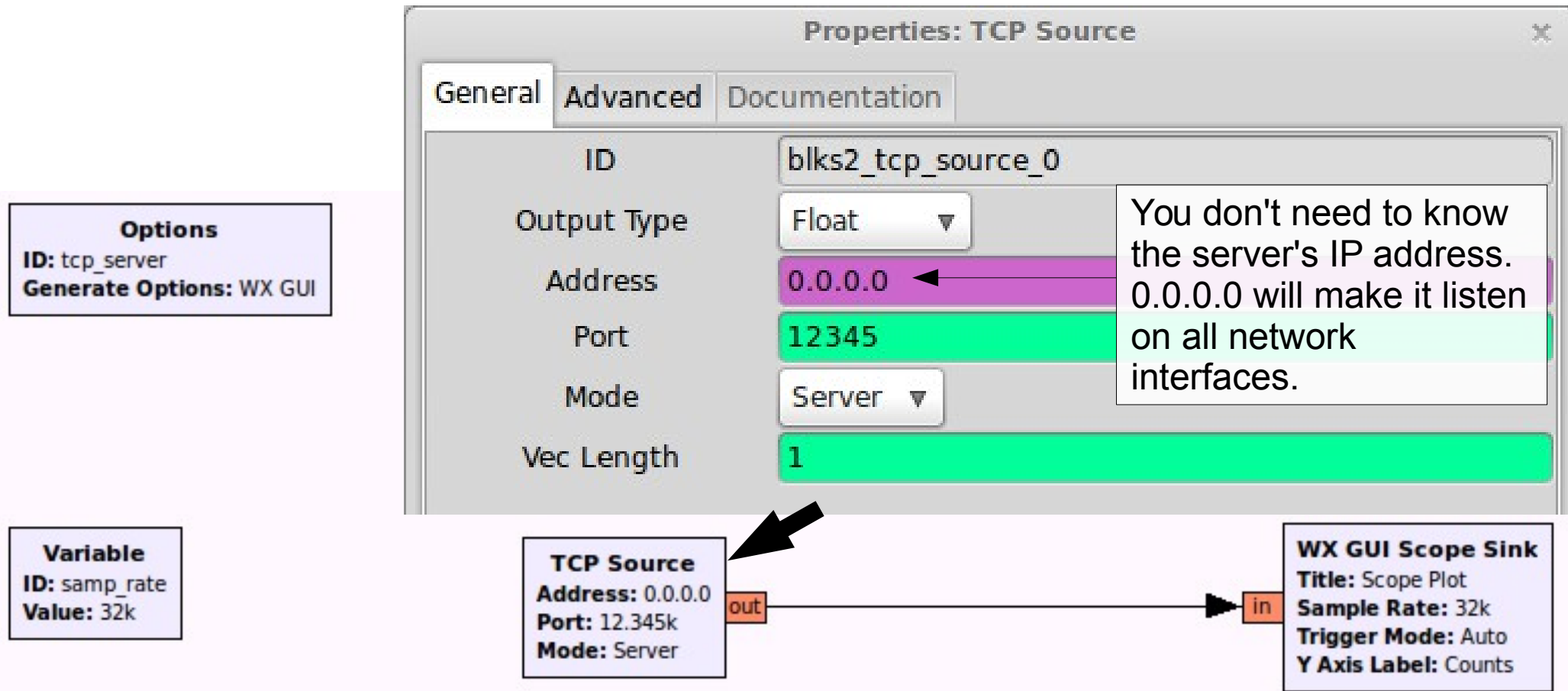
* The current TCP Source/Sink implementation does not work on Windows

Lab 1: TCP Server (consumer)



Receive samples from an incoming TCP connection and plot on a Scope Sink

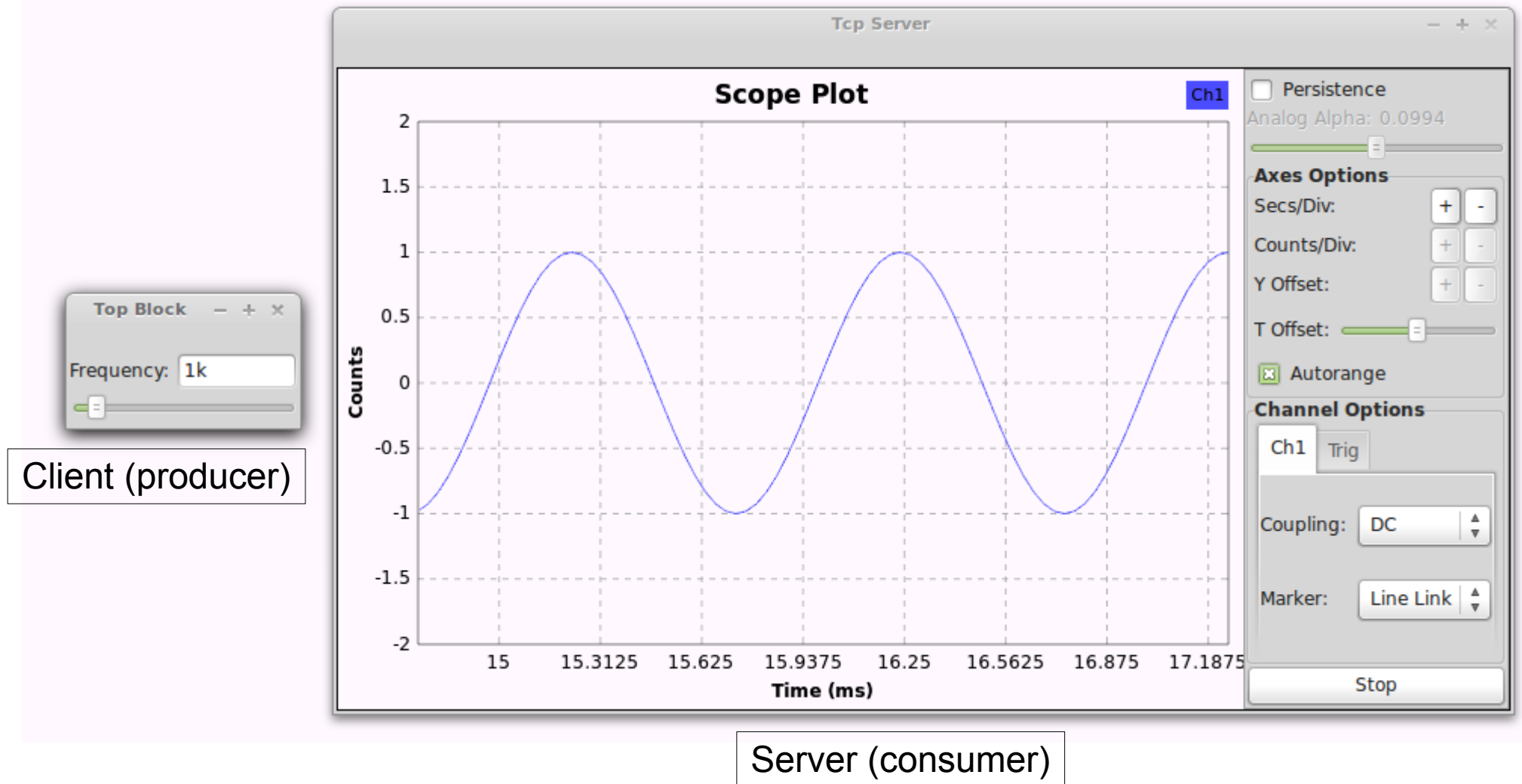
Lab 1: TCP Server (consumer)



Tip:

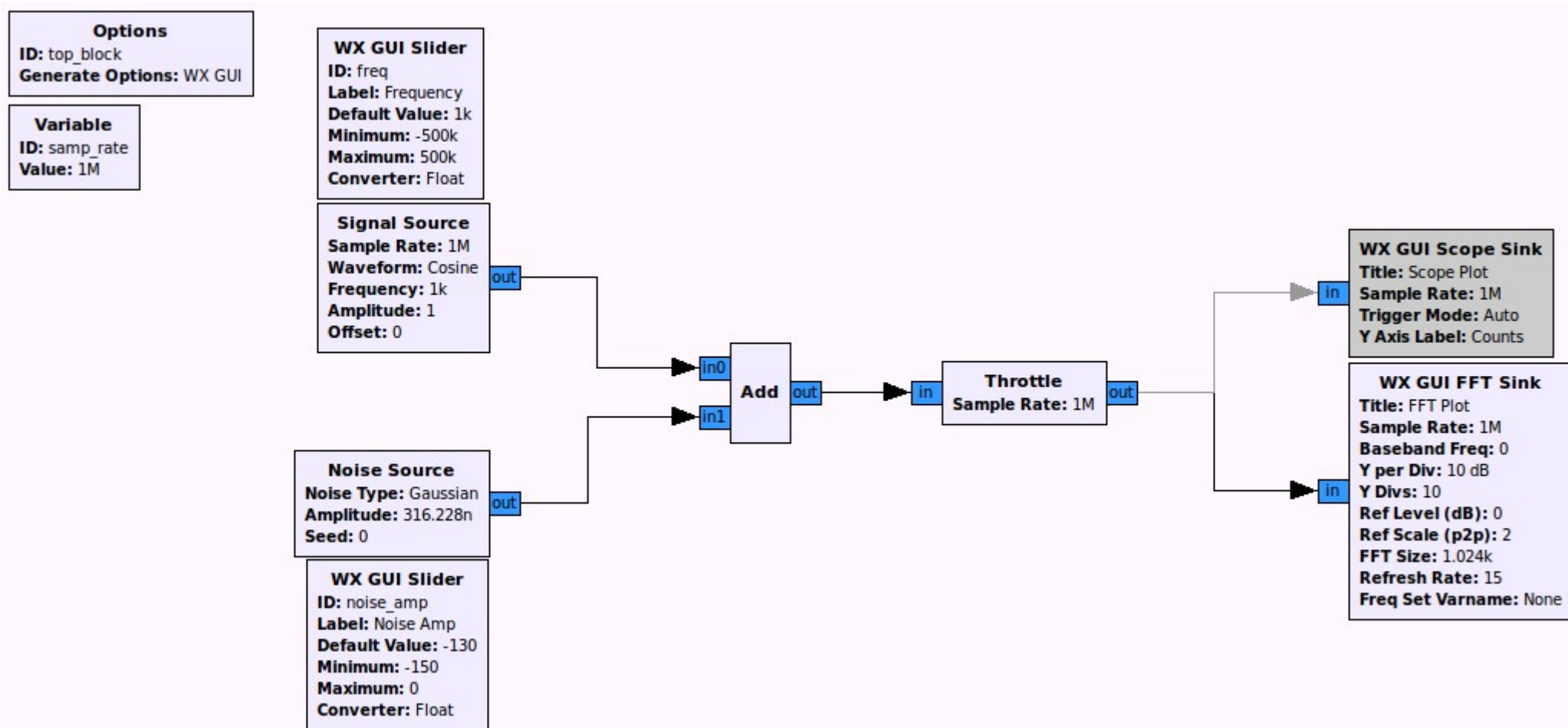
The flowgraph will not start until a TCP connection is accepted. In this case the GUI will not appear until the client has connected.

Lab 1: TCP Server & Client



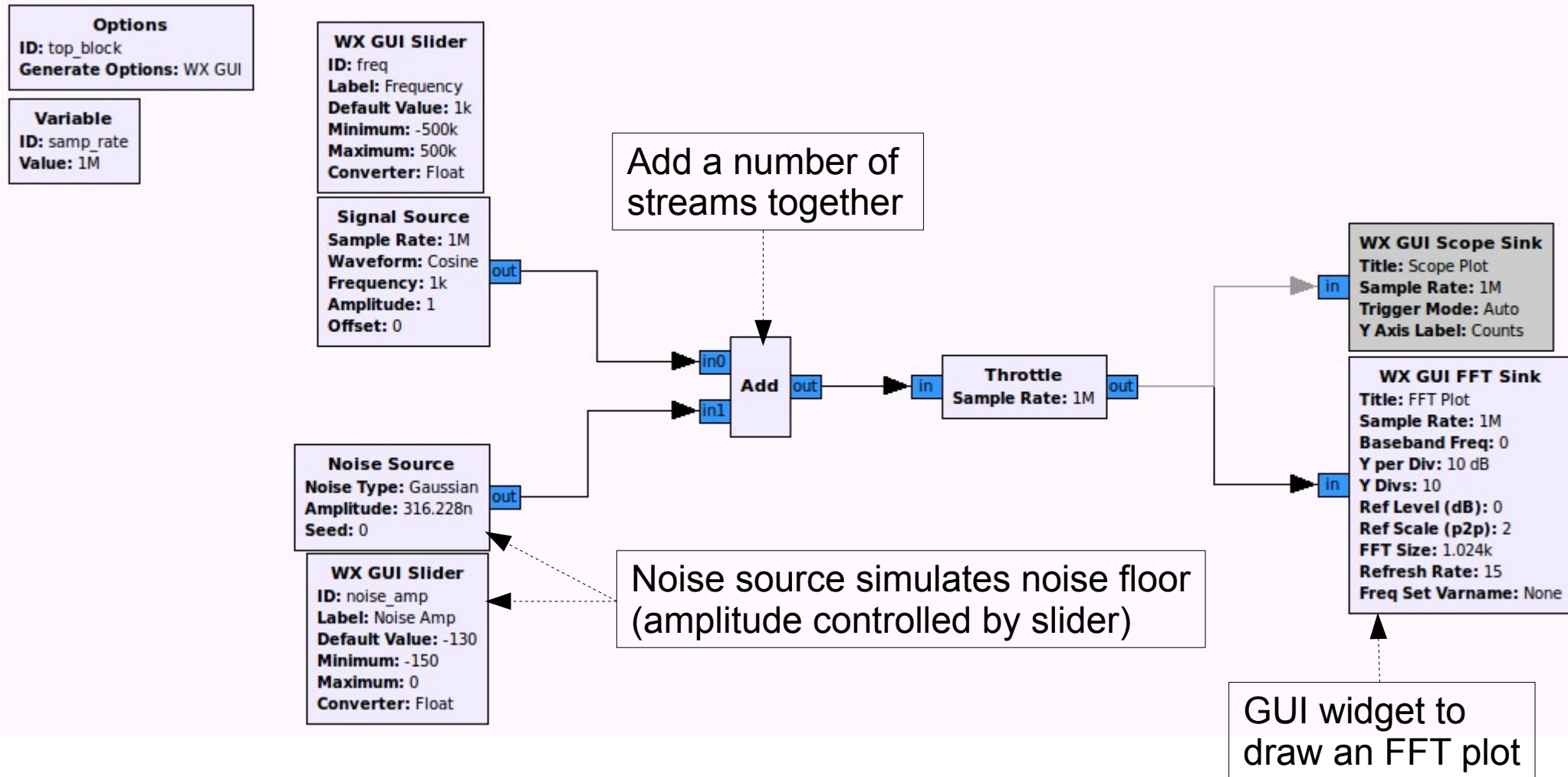
Tip: you can run each application separately on two network-connected machines. Just change the client's destination IP address to the machine on which the server is running.

Lab 2

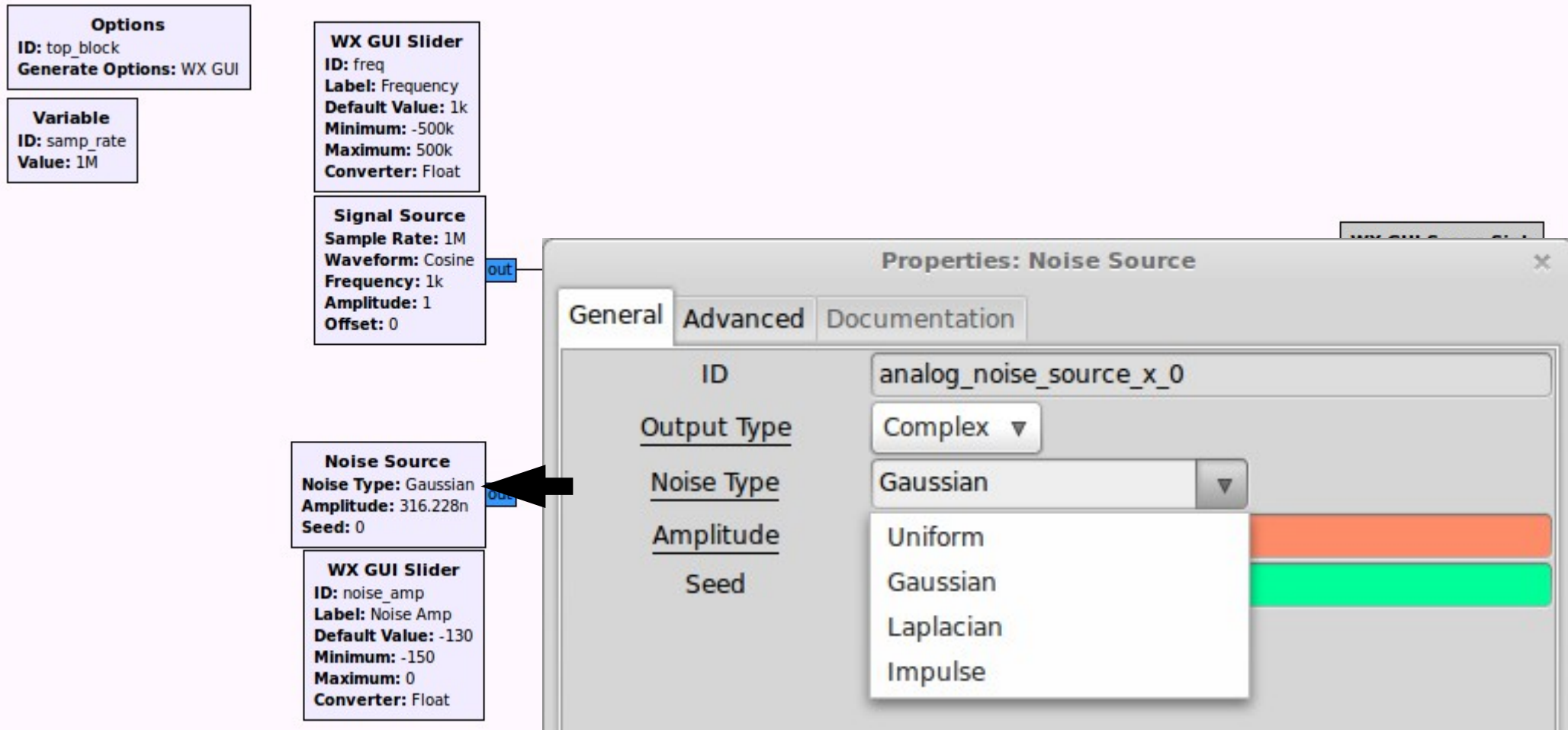


Generate a sine wave & some noise, add both, and plot the resulting signal in the frequency domain.

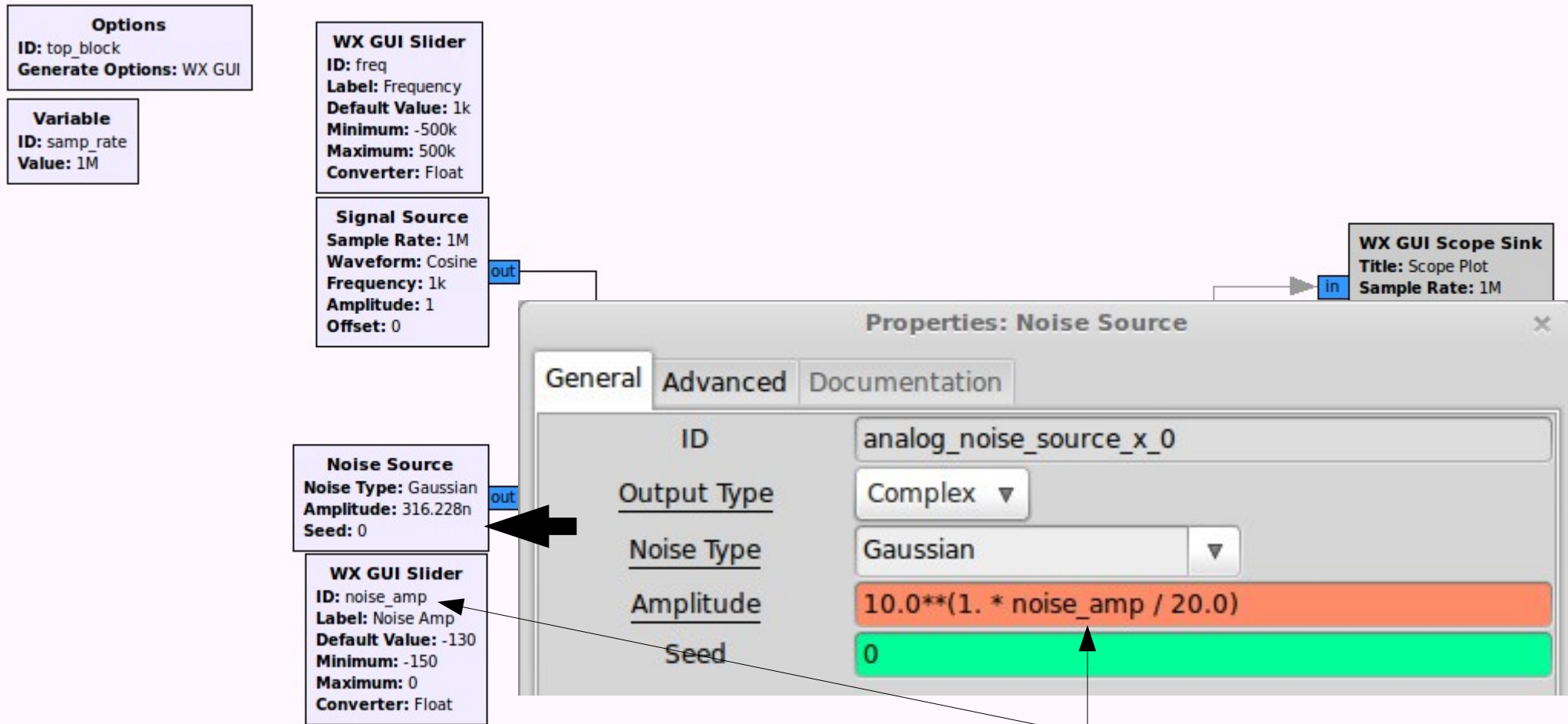
Lab 2



Lab 2



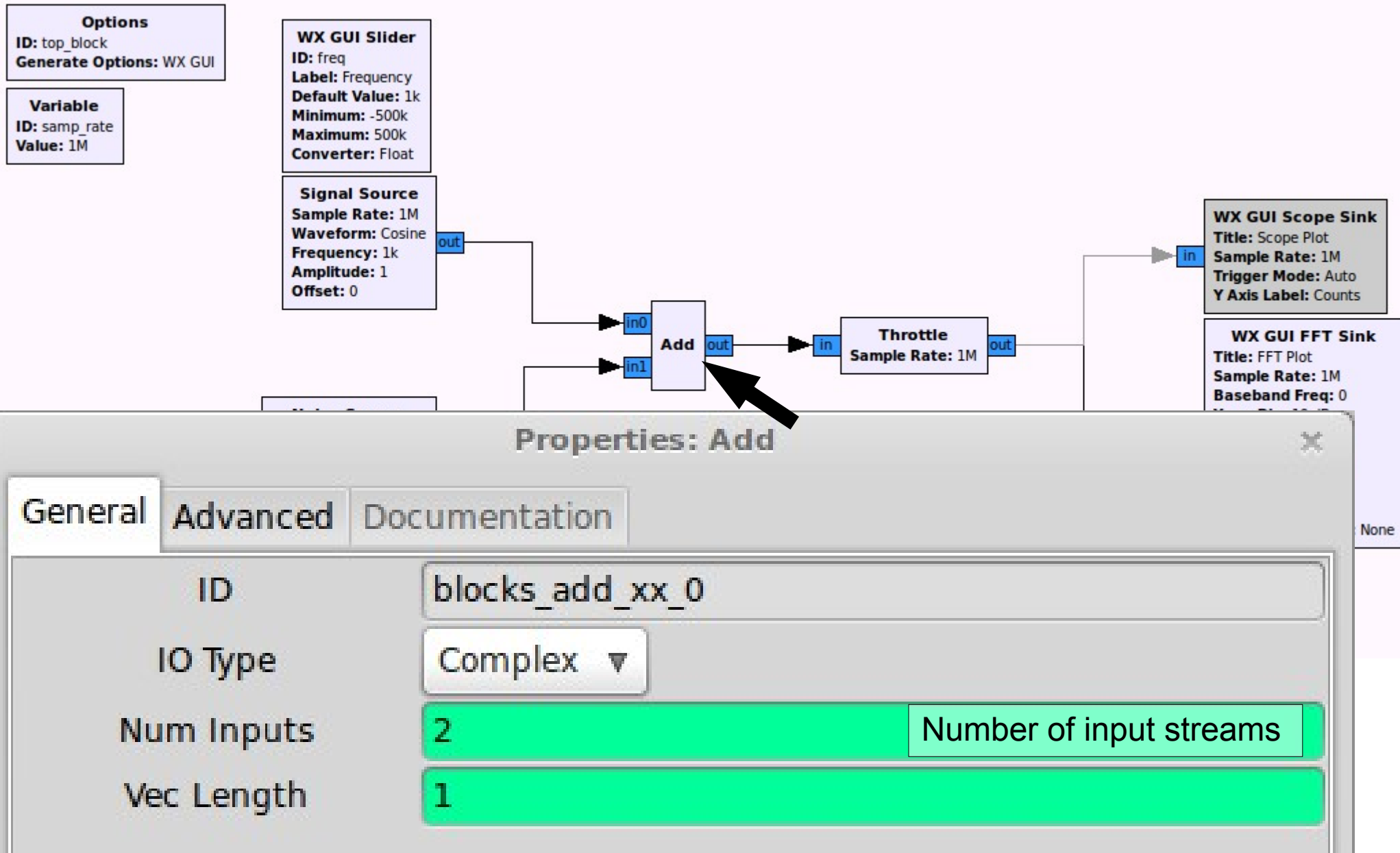
Lab 2



'noise_amp' is the slider value, which (here) we interpret in dB, as opposed to a linear sample amplitude value (e.g. '1.0').

Therefore we need to convert the value in dB to an actual linear amplitude value ('volts') for use by the block (i.e. reverse the 'log10' function). The decimal points are added to force Python to compute with floating-point values (otherwise it would round and produce integers).

Lab 2



Lab 2

Options
ID: top_block
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 1M

WX
ID:
Lab
Def
Min
Max
Con

Si
San
Wa
Fre
Am
Off

No

ID: n
Labe
Defa
Min
Max
Conv

Properties: WX GUI FFT Sink

General Advanced Documentation

ID wxgui_fftsink2_0

Type Complex ▾

Title FFT Plot

Sample Rate samp_rate

Baseband Freq 0

Y per Div 10 dB ▾

Y Divs 10

Ref Level (dB) 0

Ref Scale (p2p) 2.0

FFT Size 1024

Refresh Rate 15

Peak Hold Off ▾

Average Off ▾

Window Automatic ▾

Window Size

Grid Position

Notebook

Freq Set Varname None

Sets the range on the Y-axis

Value added to rendered Y-axis values

Used to control how the computed FFT is scaled and 'fit' to the available plot area.

Time relative to Sample Rate!

WX GUI Scope Sink
Title: Scope Plot
Sample Rate: 1M
Trigger Mode: Auto
Y Axis Label: Counts

WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 1M
Baseband Freq: 0
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): 0
Ref Scale (p2p): 2
FFT Size: 1.024k
Refresh Rate: 15
Freq Set Varname: None

Name of an **existing** GRC Variable that will be set to the frequency you click on if clicking in the FFT plot area.

Lab 2

Options
ID: top_block
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 1M

WX
ID:
Lab
Def
Min
Max
Con

Si
San
Wa
Fre
Am
Off

No
Noise
Amplit
Seed:

WX
ID: n
Labe
Defa
Minir
Maxi
Conv

Properties: WX GUI FFT Sink

General Advanced Documentation

ID	wxgui_fftsink2_0
Type	Complex ▾
Title	FFT Plot
Sample Rate	samp_rate
Baseband Freq	0
Y per Div	10 dB ▾
Y Divs	10
Ref Level (dB)	0
Ref Scale (p2p)	2.0
FFT Size	1024
Refresh Rate	15
Peak Hold	Off ▾
Average	Off ▾
Window	Automatic ▾
Window Size	
Grid Position	
Notebook	
Freq Set Varname	None

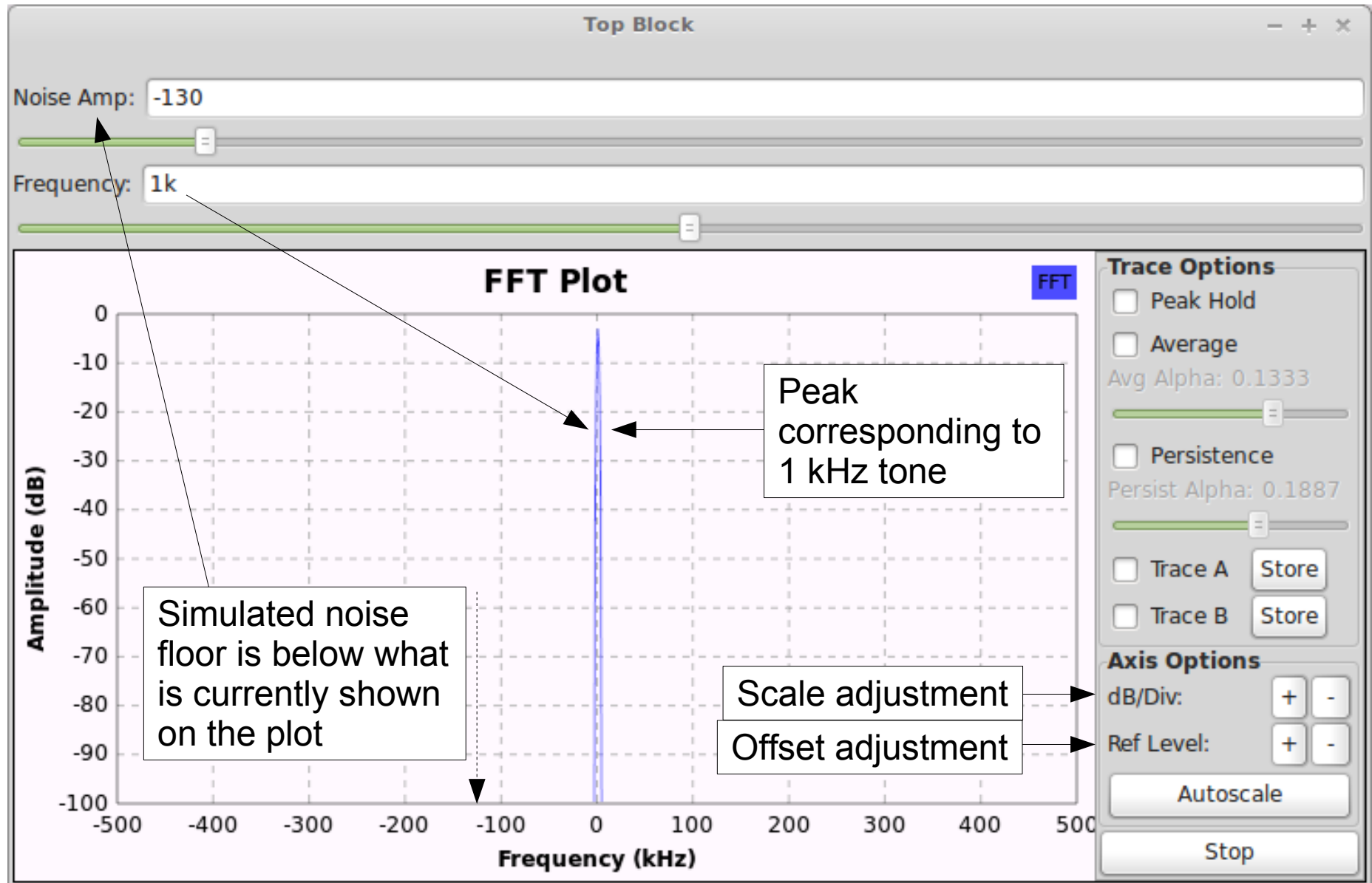
WX GUI Scope Sink
Title: Scope Plot
Sample Rate: 1M
Trigger Mode: Auto
Y Axis Label: Counts

WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 1M
Baseband Freq: 0
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): 0
Ref Scale (p2p): 2
FFT Size: 1.024k
Refresh Rate: 15
Freq Set Varname: None

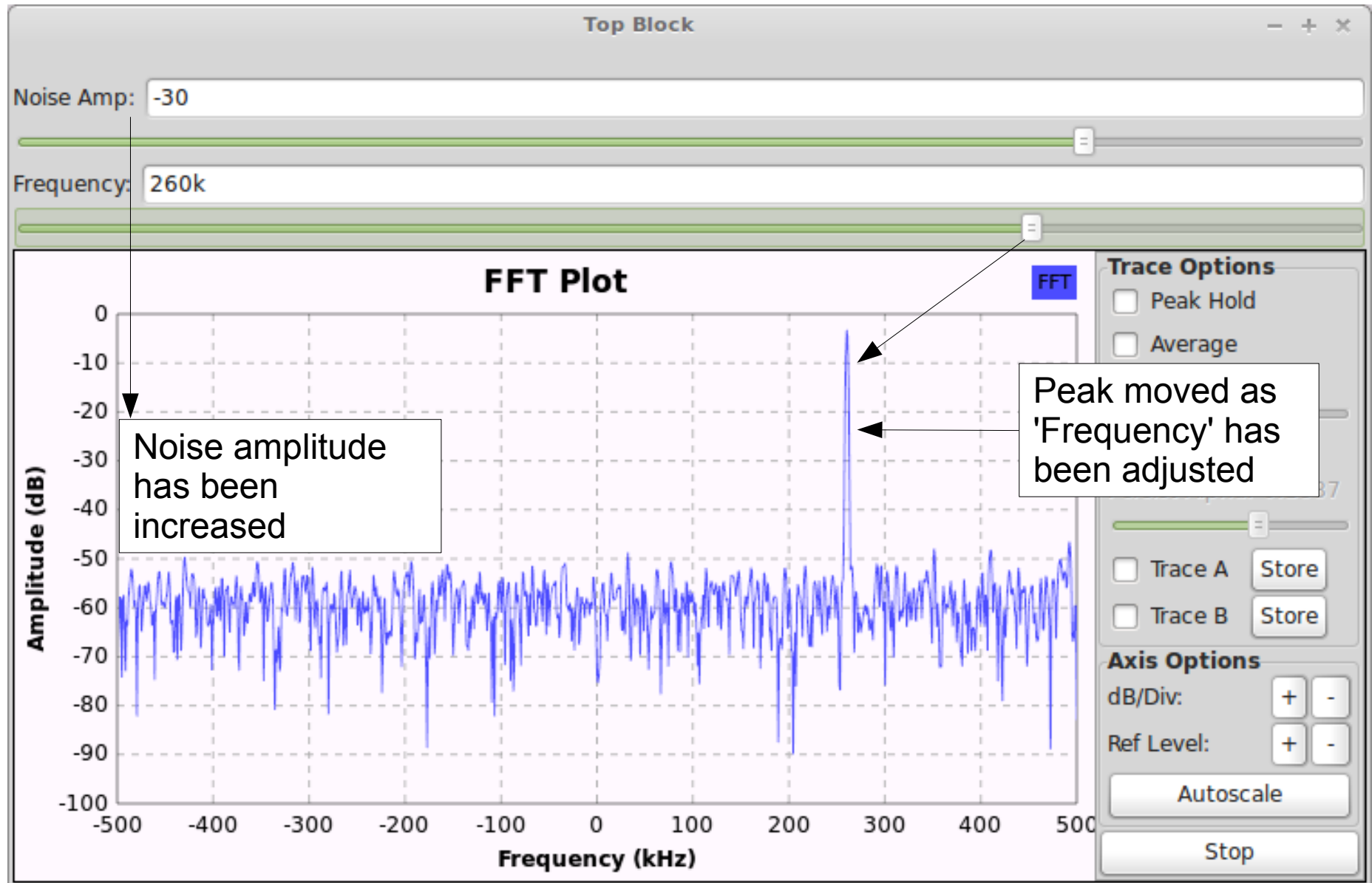
Tip:

If your FFT Sink will show your baseband signal, you can use 'Freq Set Varname' to have your flowgraph process a specific signal-of-interest at the frequency you click on (e.g. with the Freq Xlating FIR filter). More on this later...

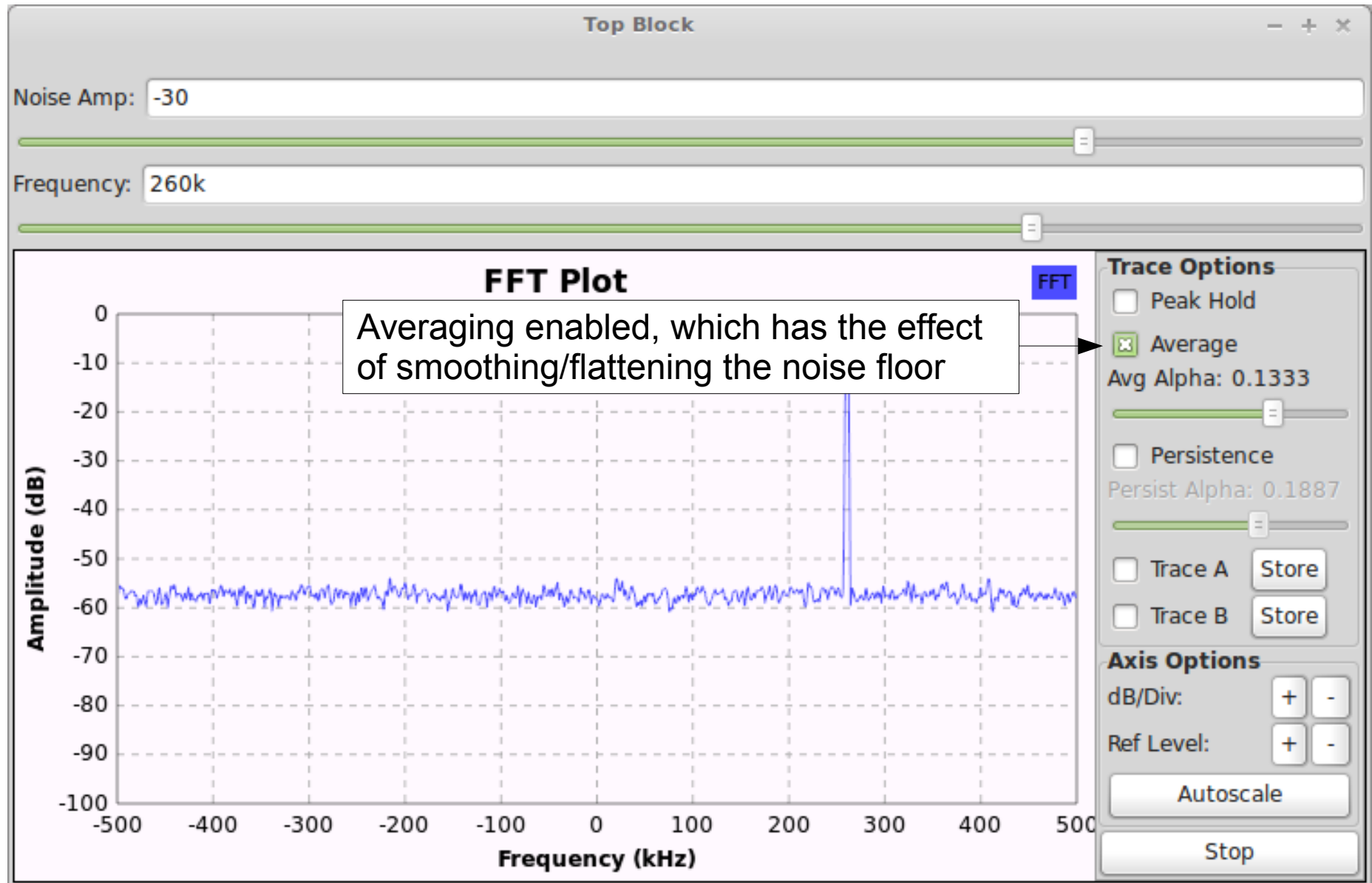
Lab 2



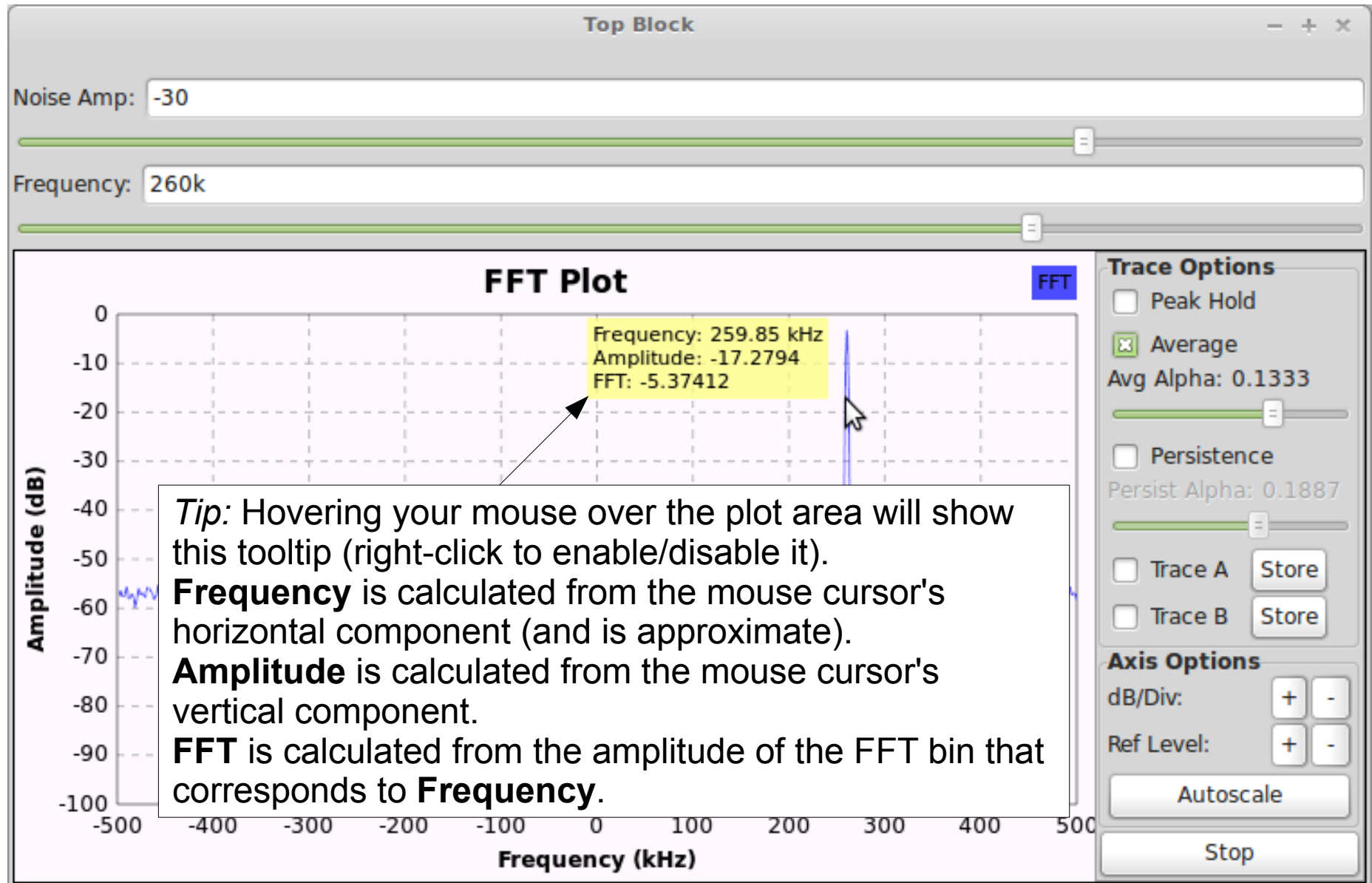
Lab 2



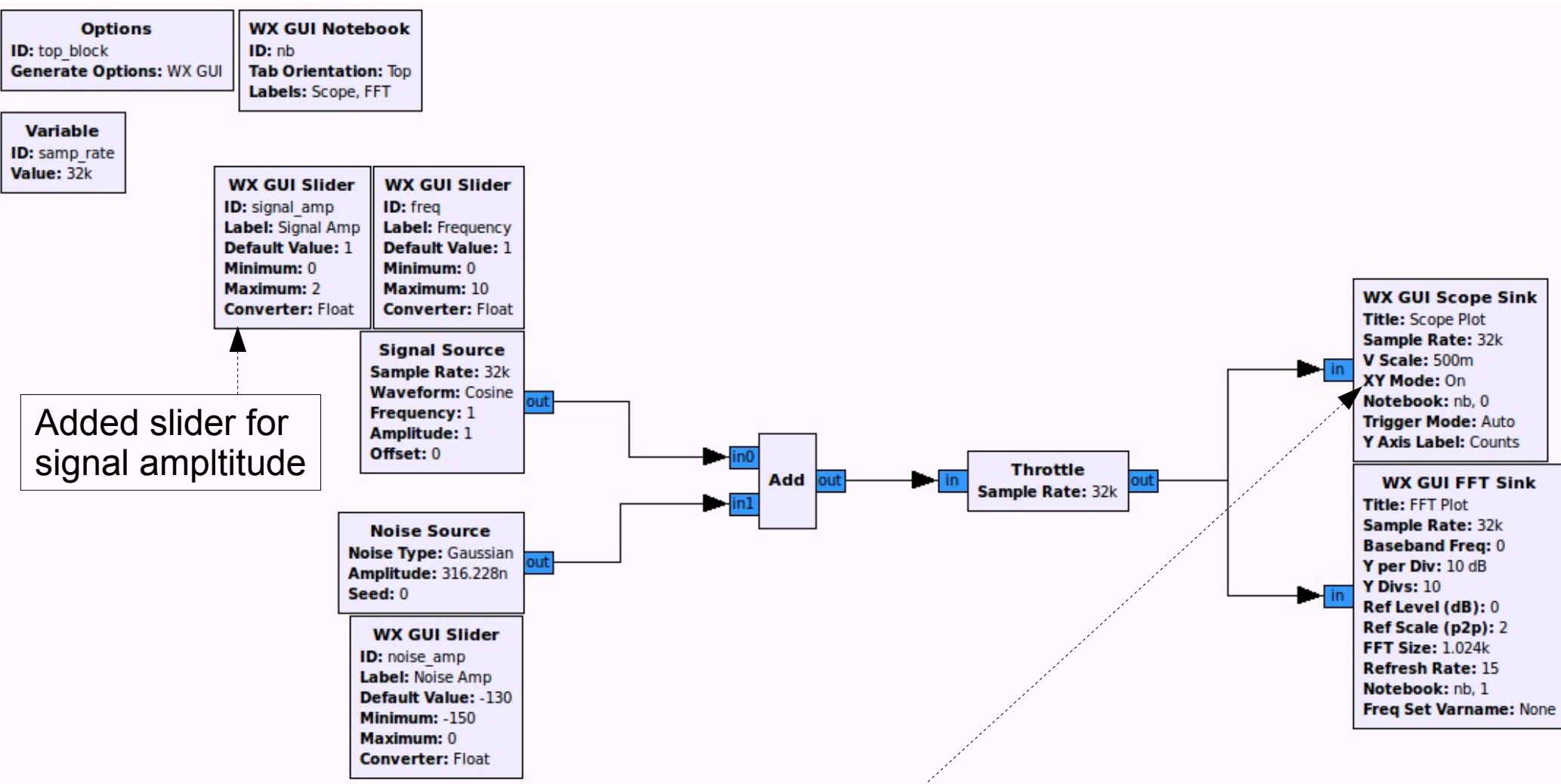
Lab 2



Lab 2

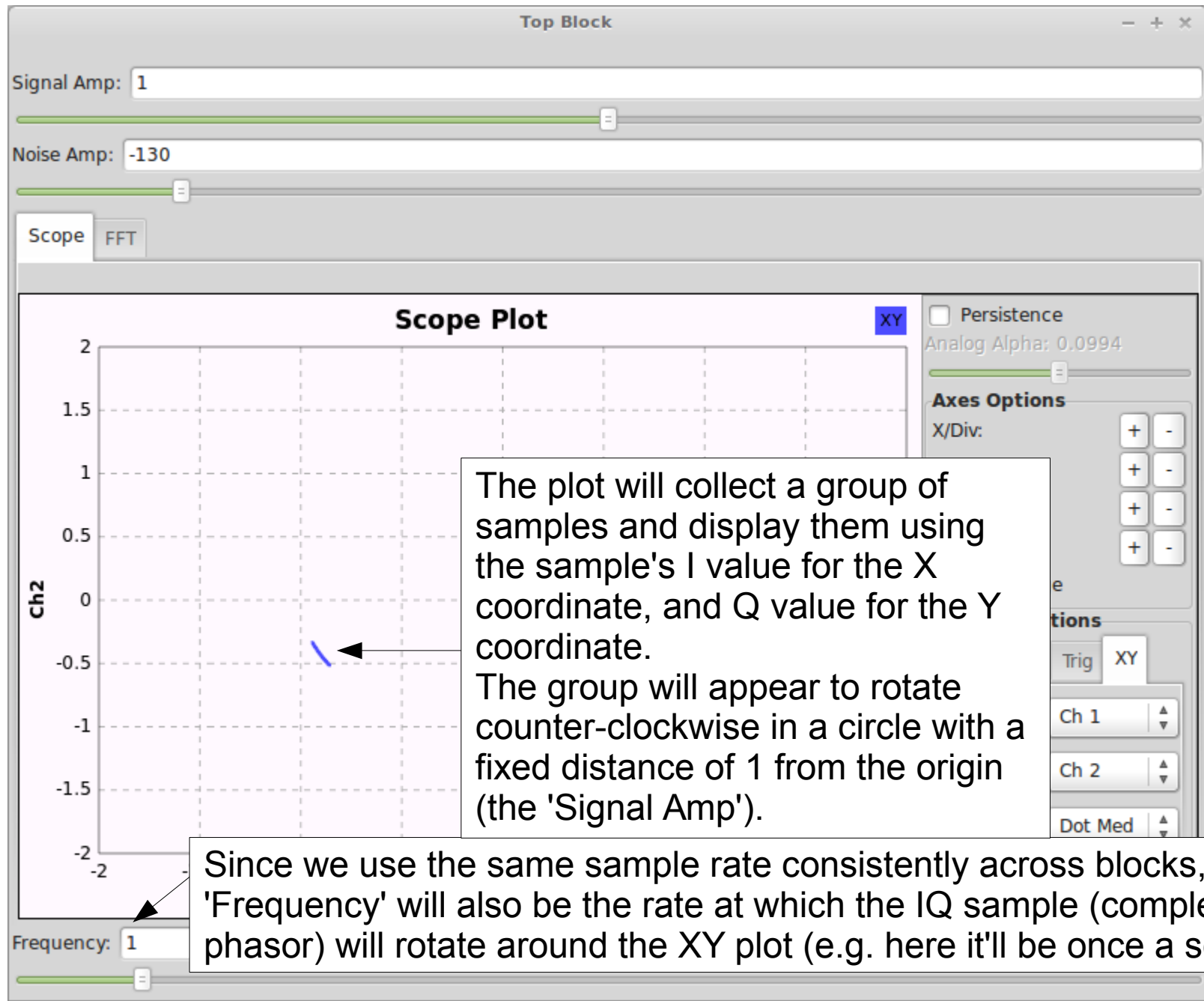


Lab 2: XY Mode

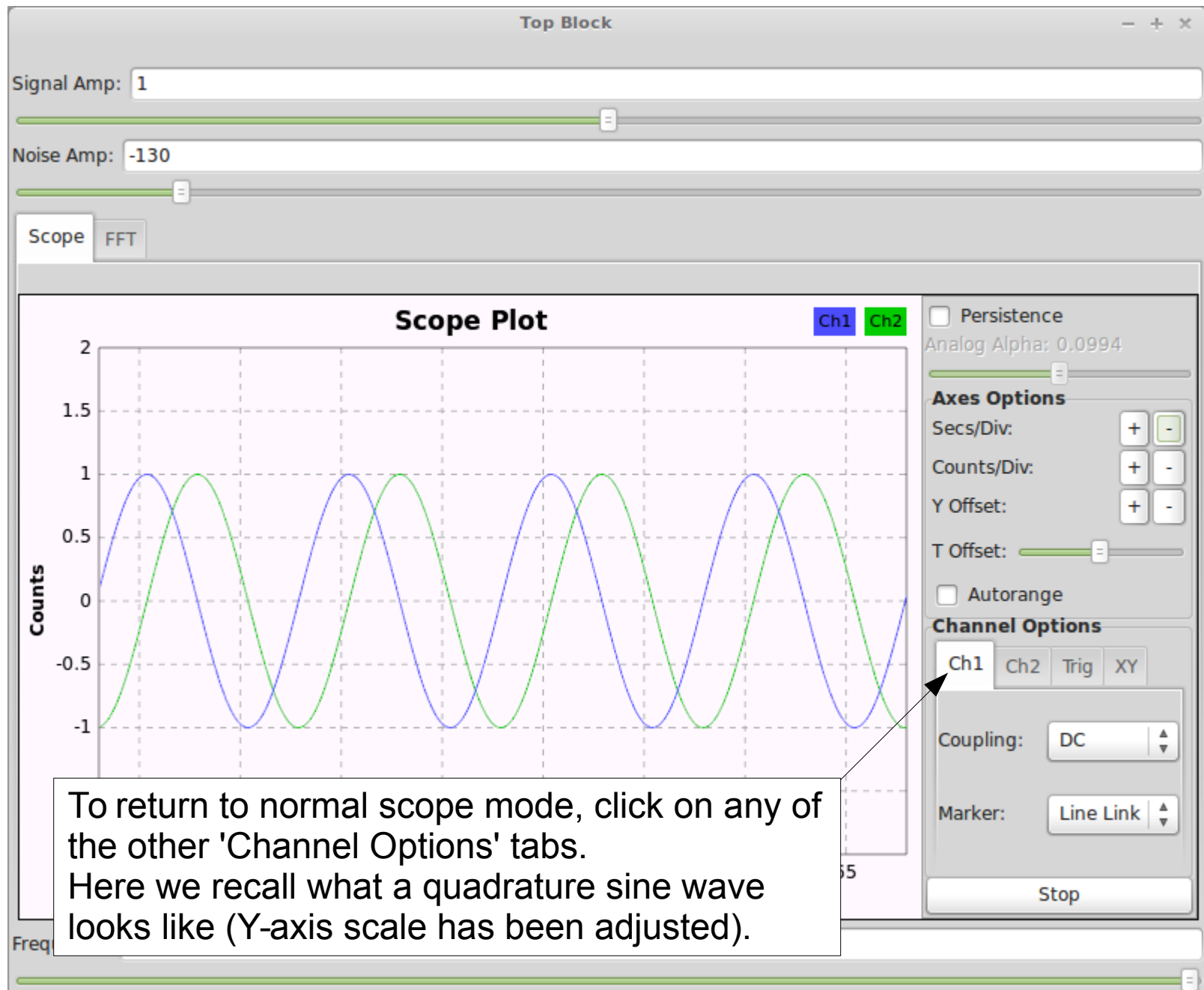


Use a Scope Sink in XY Mode so we can observe the characteristics of an IQ (quadrature) signal

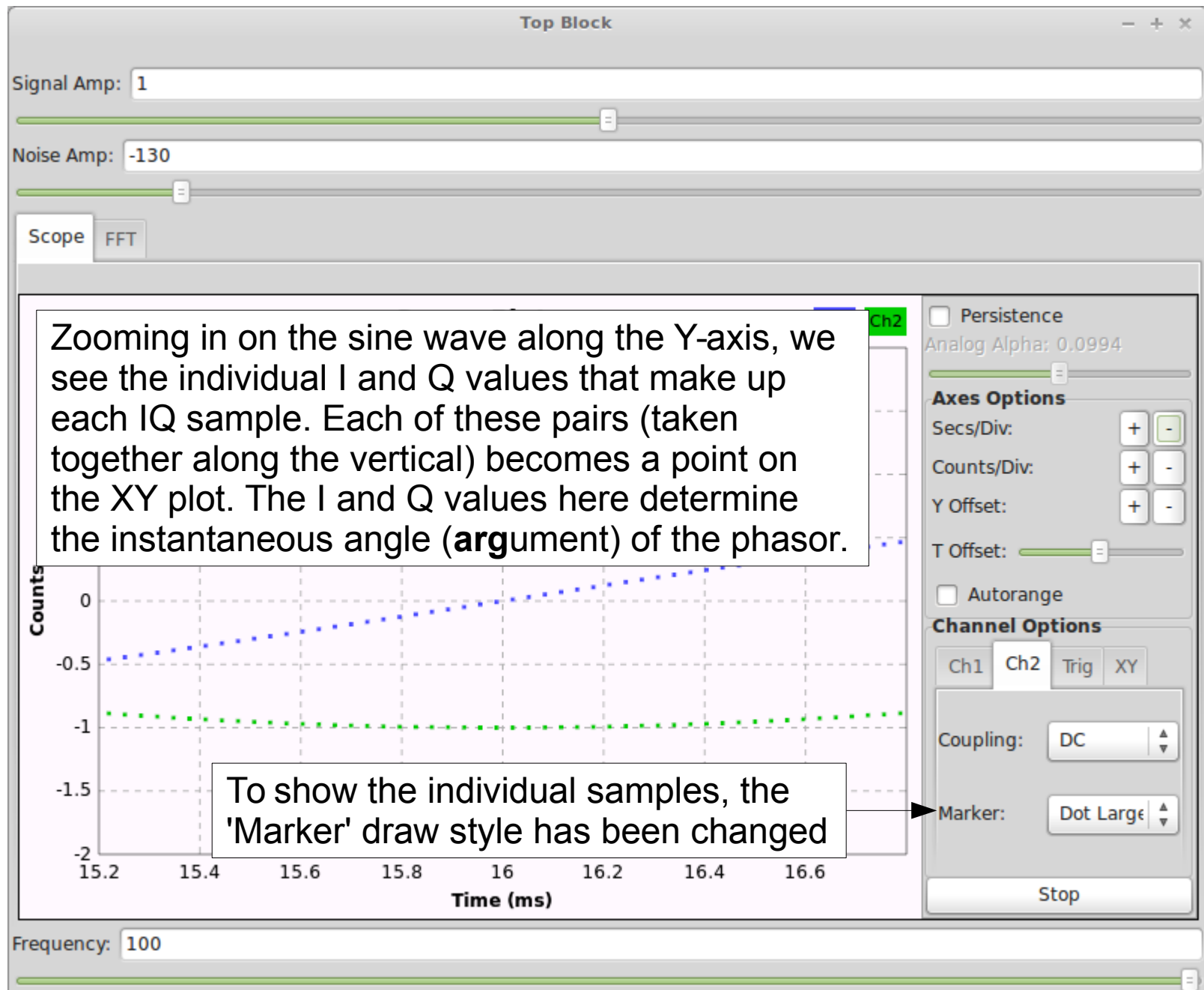
Lab 2: XY Mode



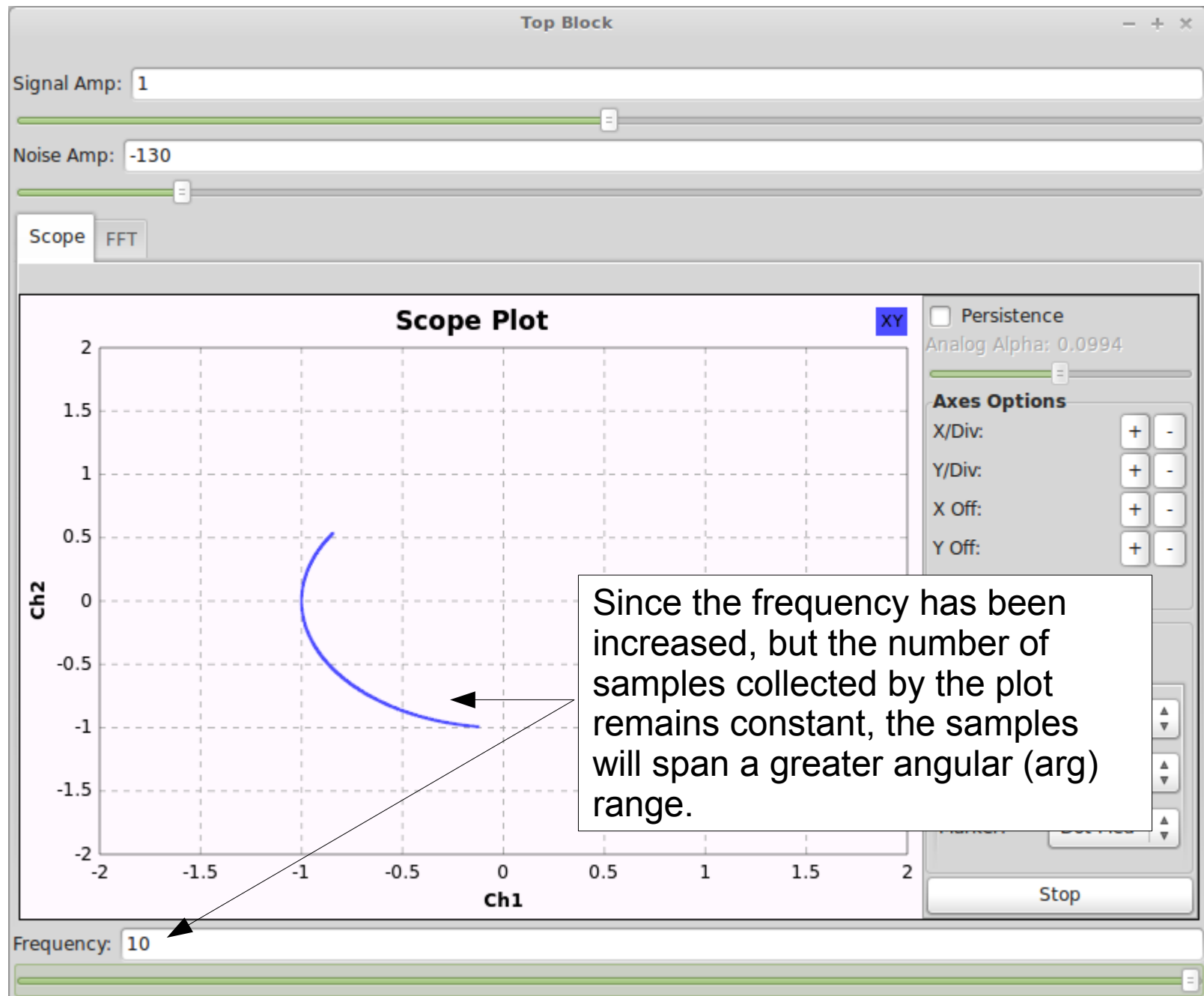
Lab 2: XY Mode



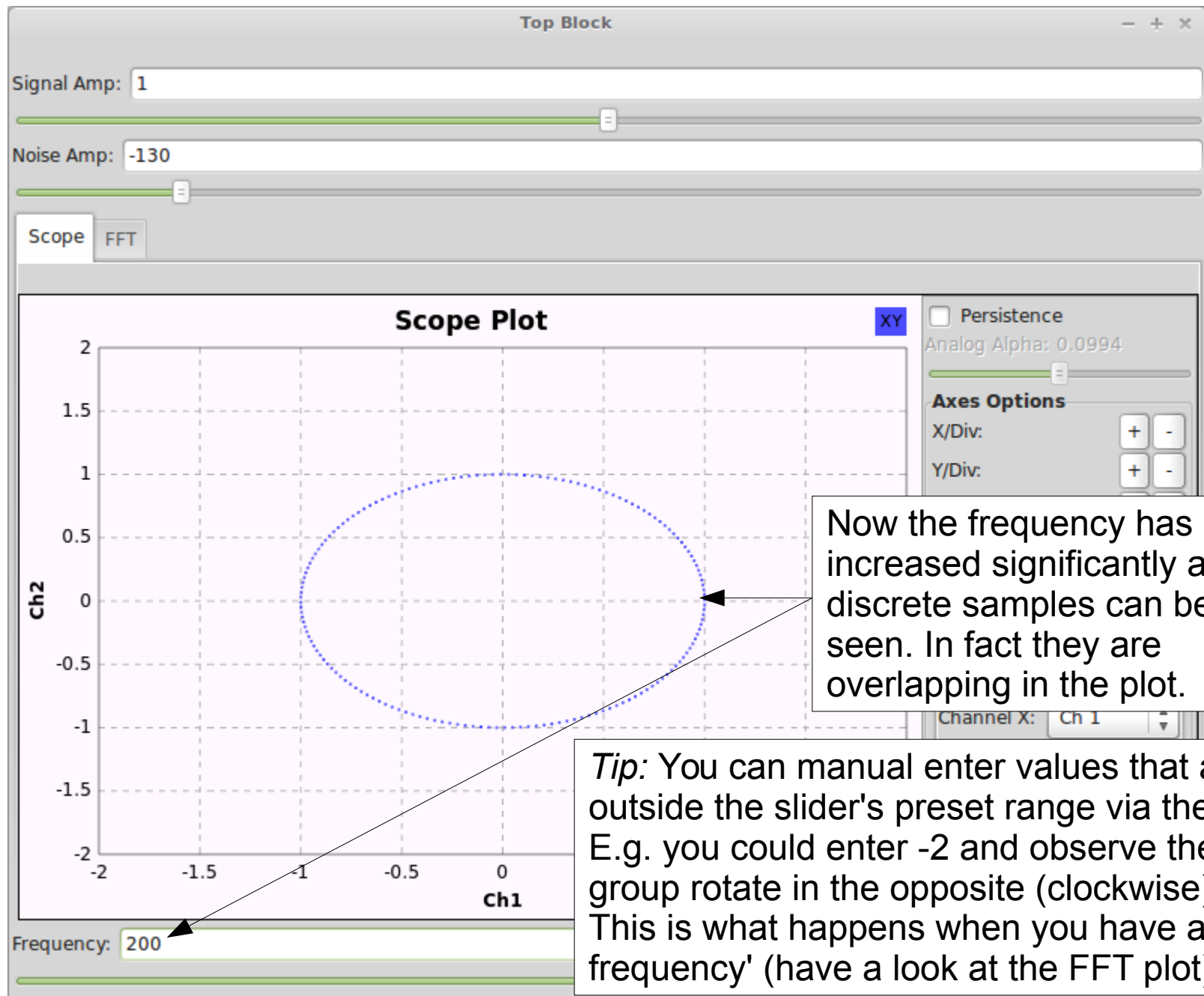
Lab 2: XY Mode



Lab 2: XY Mode



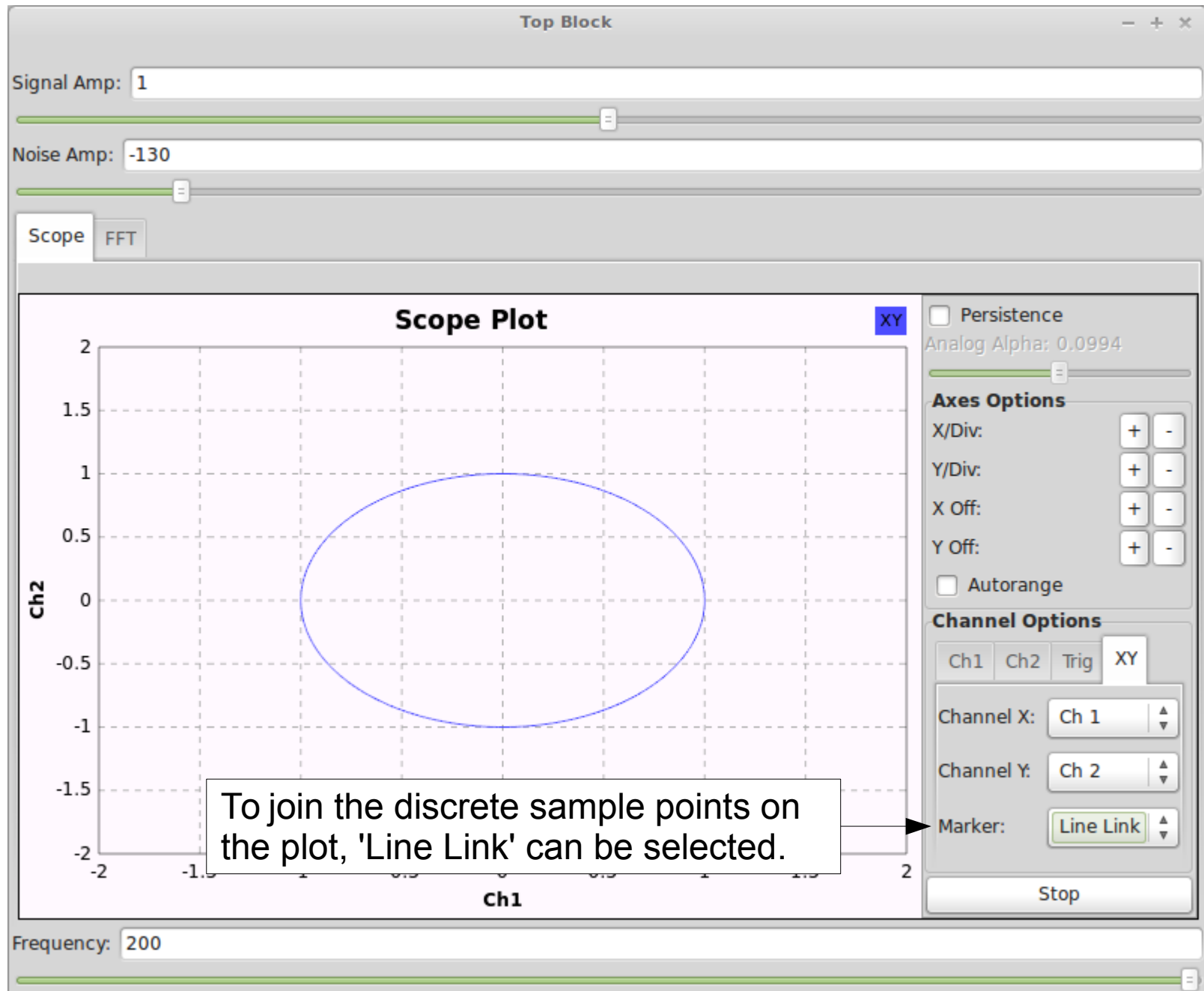
Lab 2: XY Mode



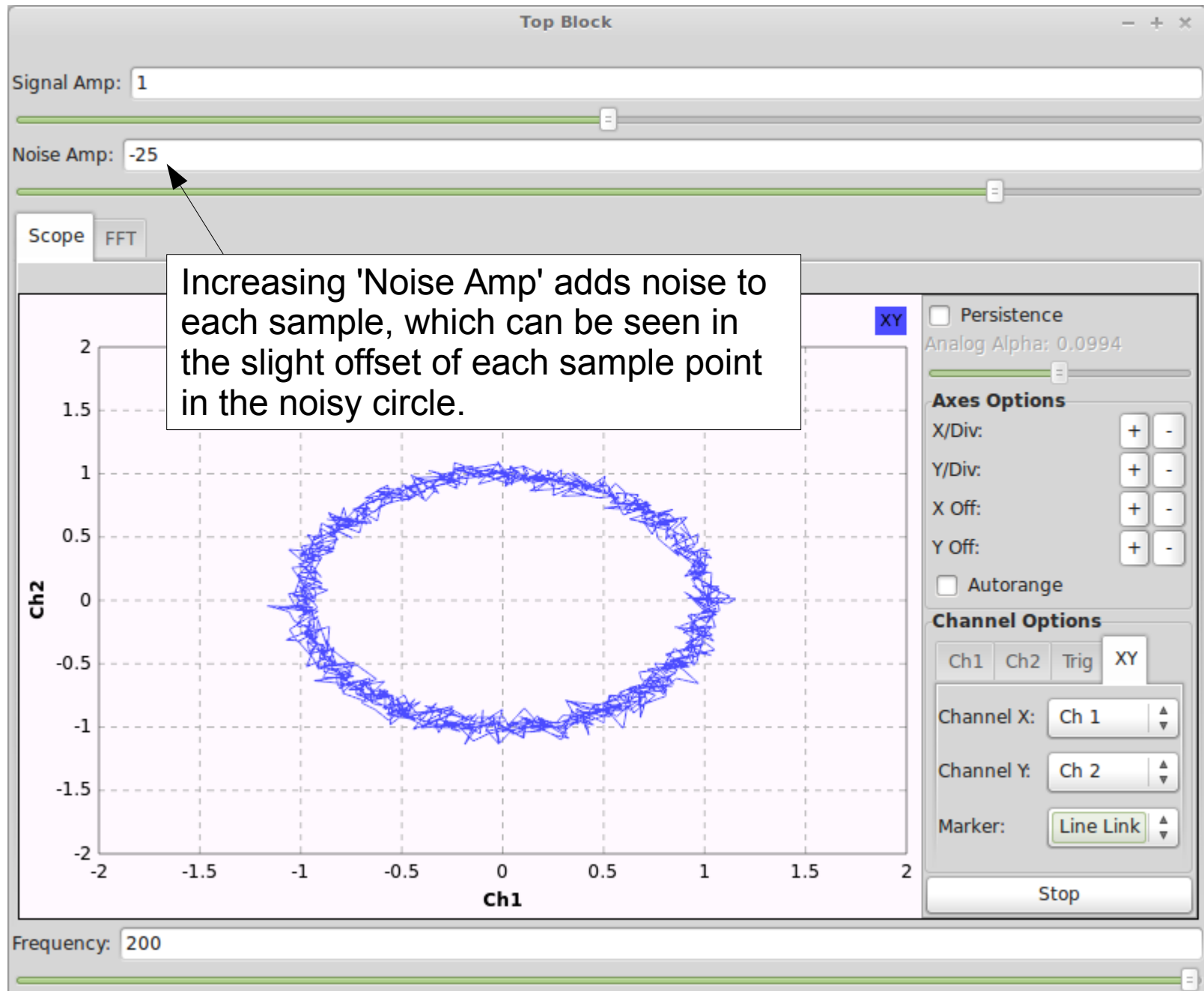
Now the frequency has been increased significantly and the discrete samples can be seen. In fact they are overlapping in the plot.

Tip: You can manual enter values that are outside the slider's preset range via the text box. E.g. you could enter -2 and observe the sample group rotate in the opposite (clockwise) direction. This is what happens when you have a 'negative frequency' (have a look at the FFT plot).

Lab 2: XY Mode

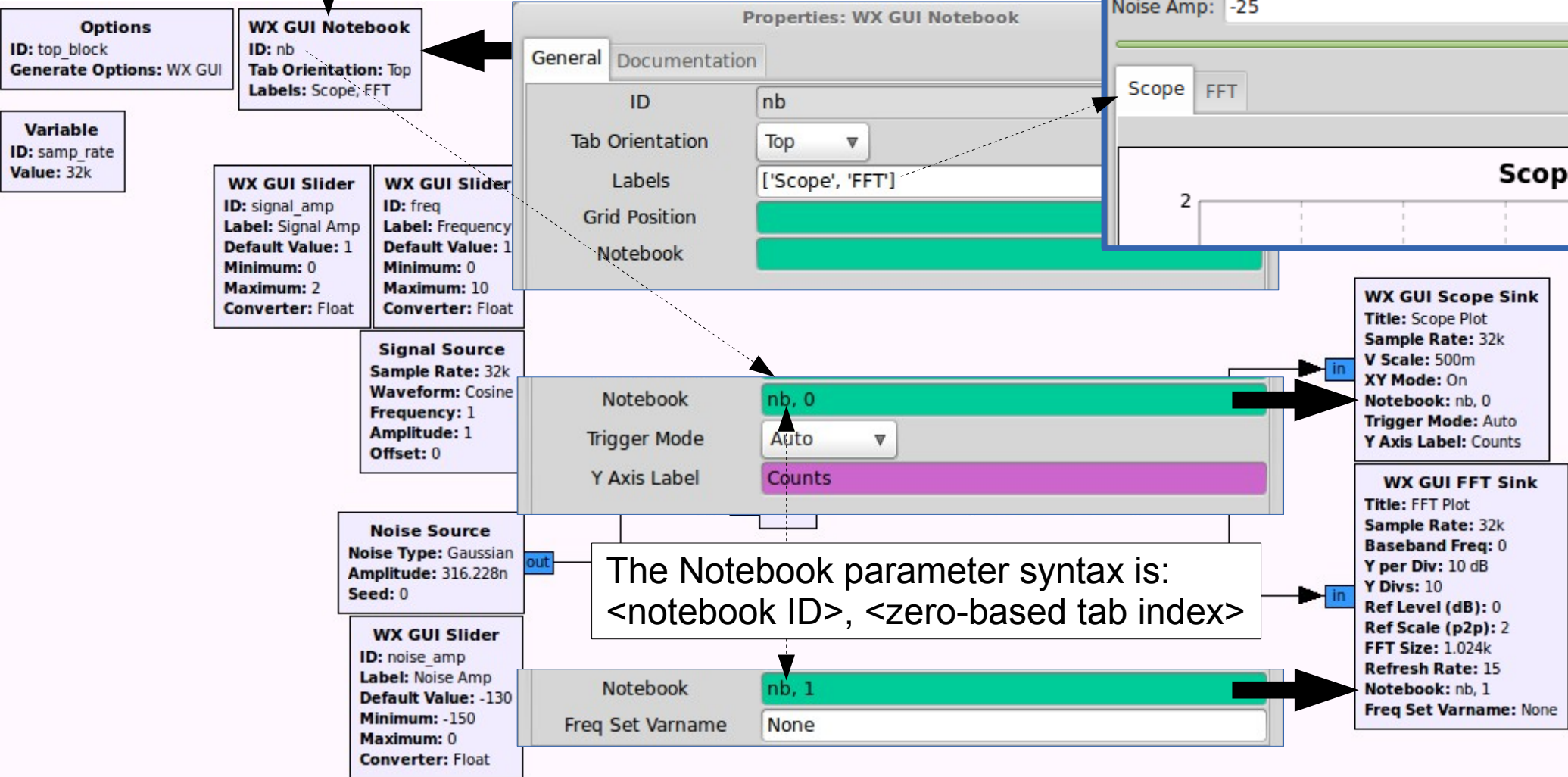
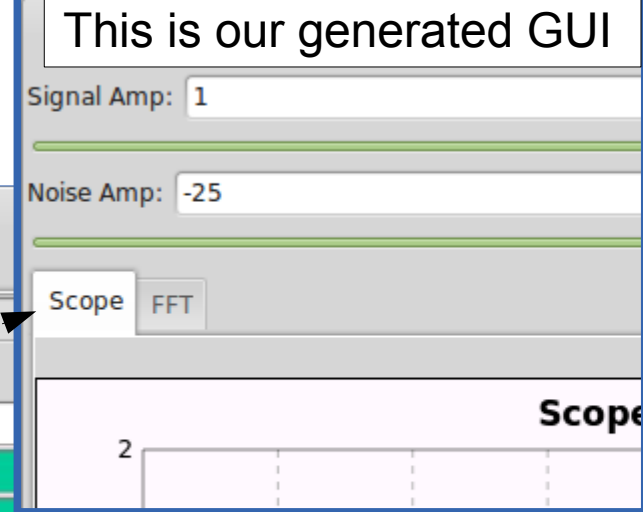


Lab 2: XY Mode

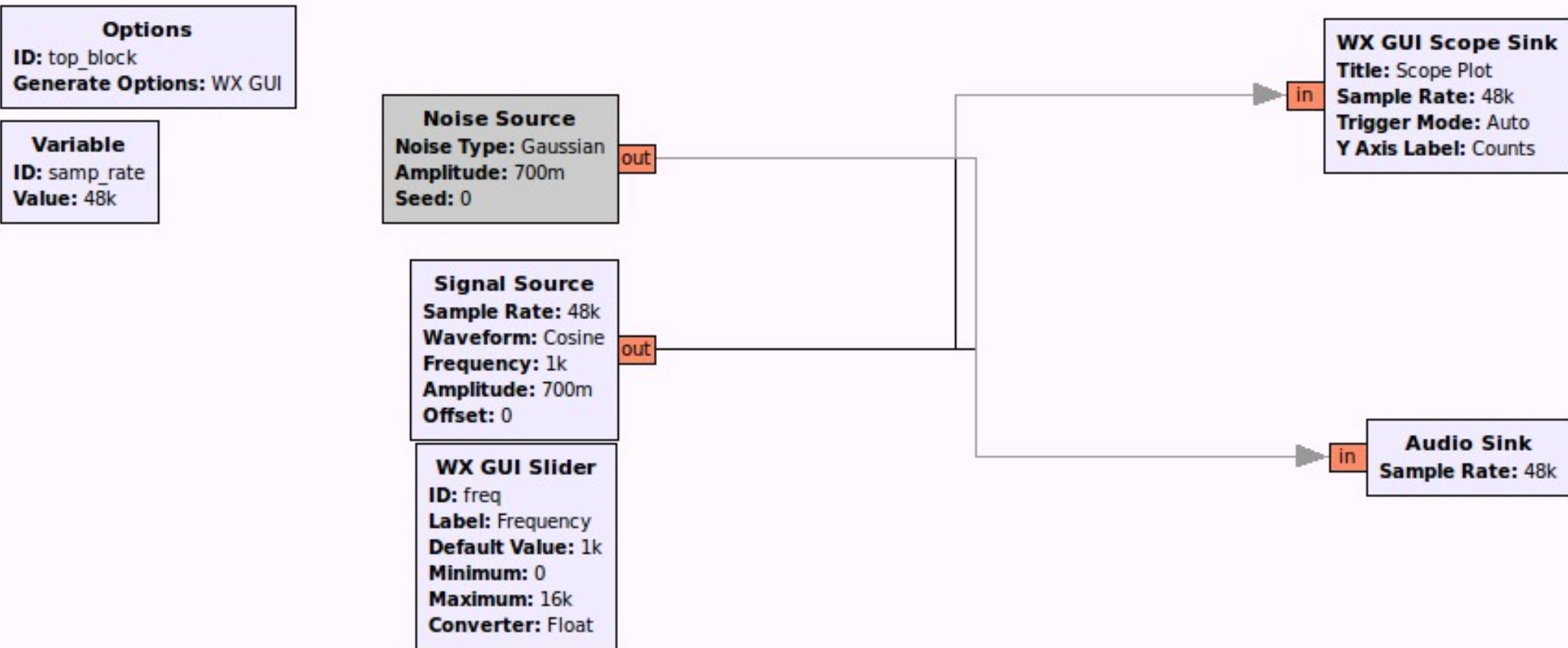


A Notebook can be used to organise GUI widgets in tabs.

Lab 2

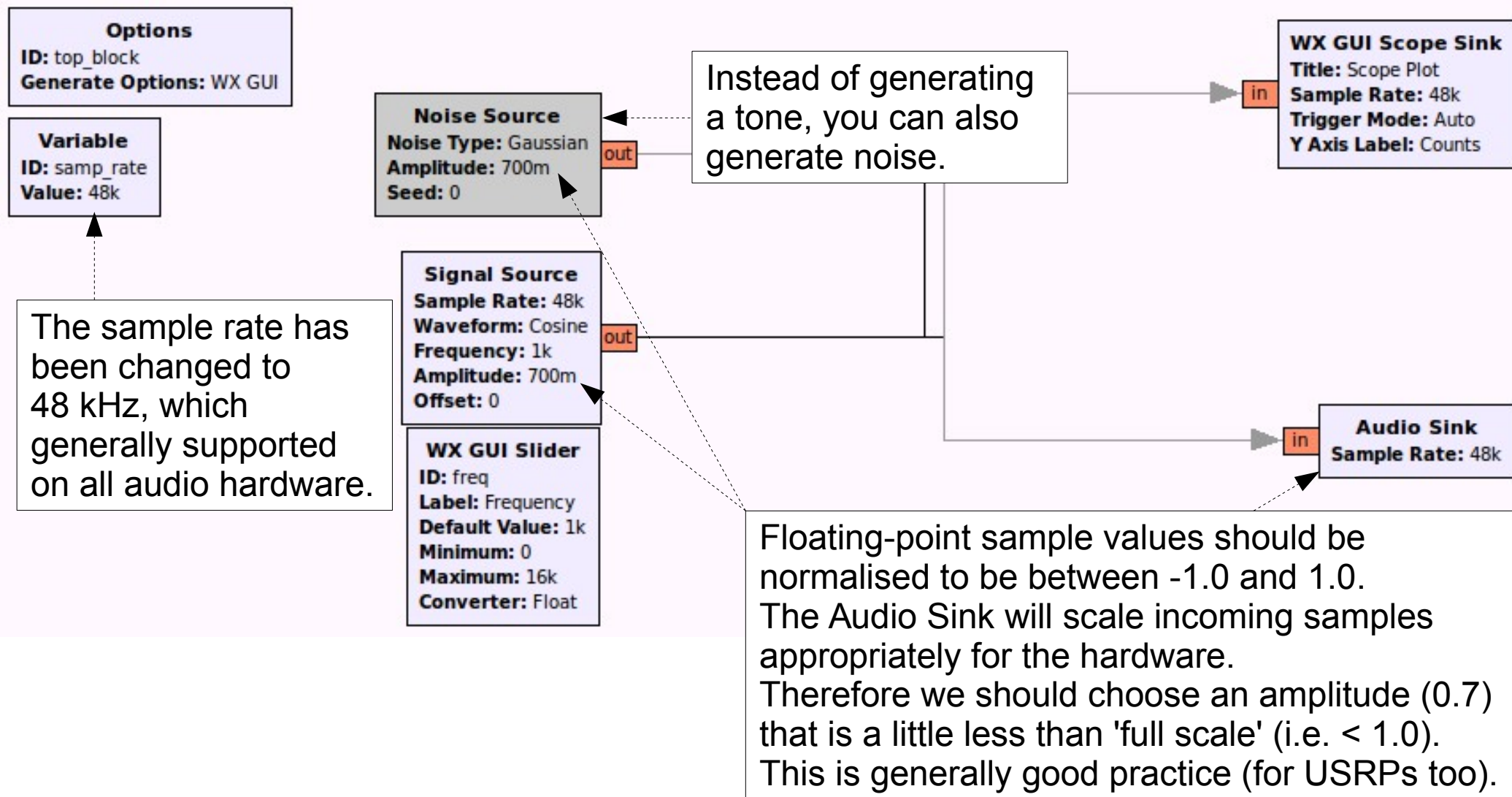


Lab 3: Audio



Output a single tone from the computer's soundcard

Lab 3: Audio



Lab 3: Audio

Options
ID: top_block
Generate Options: WX GUI

Properties: Audio Sink

General Advanced Documentation

ID: audio_sink_0

Sample Rate: samp_rate

Device Name: [text field]

OK to Block: Yes ▾

Num Inputs: 1

Identifier that is platform-specific. Blank implies the default. E.g. on ALSA with pulse audio installed, you could write 'pulse'. Run "aplay -L" in a Linux terminal to see possible ALSA options.

Set to the number of channels you wish to stream to on your audio hardware (e.g. 2 for stereo)

Depending on the underlying implementation, this will instruct the hardware's usermode API to return immediately after being passed a buffer of audio samples (non-blocking mode), or wait until they are consumed (blocking mode). See next page for details.

WX GUI Scope Sink
Title: Scope Plot
Sample Rate: 48k
Trigger Mode: Auto
Y Axis Label: Counts

Audio Sink
Sample Rate: 48k

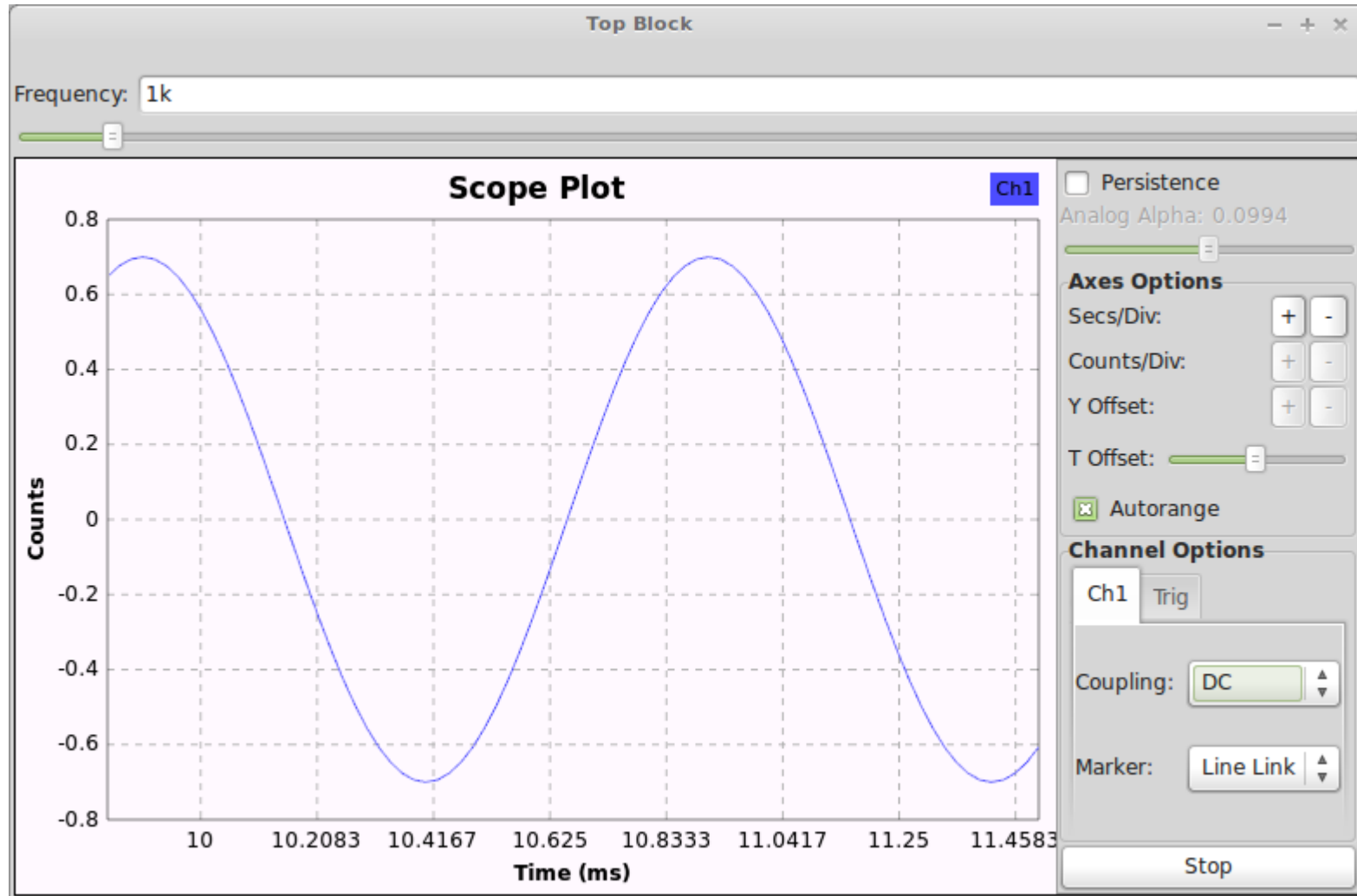
Lab 3: Audio

- Blocking mode ('OK to Block') will apply upstream backpressure, which is good when the Audio Sink is the only hardware device in the flowgraph.
- This can be problematic if the flowgraph source is, for example, a USRP. The source is then also hardware that has its own internal clock and will be throttling the sample production rate while the Audio Sink is throttling consumption with its own unsynchronised clock. This is called the '*two clock*' problem.

Lab 3: Audio

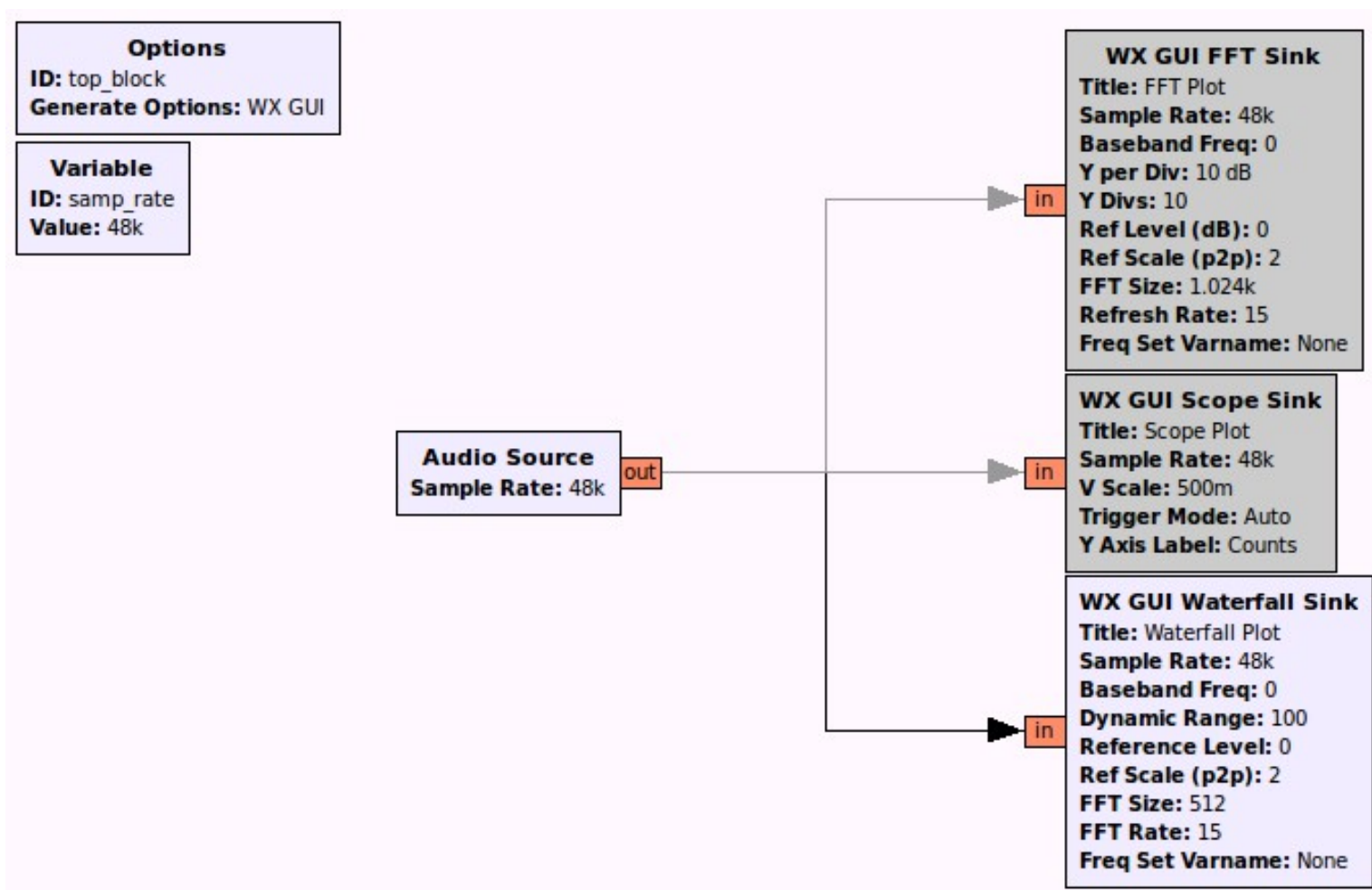
- To workaround this two clock problem, set the Audio Sink to non-blocking mode (*not* 'OK to Block') so that it will never hold up the flowgraph (i.e. not apply backpressure). It will consume samples as normal, but if there is ever an excess (e.g. the USRP is producing samples a little faster than the Audio Sink can consume) it will drop the samples (might cause audio glitches).
- This does not solve the case where samples are being produced *slower* than the Audio Sink's consumption rate (this will produce an underrun: audio will sound choppy and 'aU' will be printed).

Lab 3: Audio



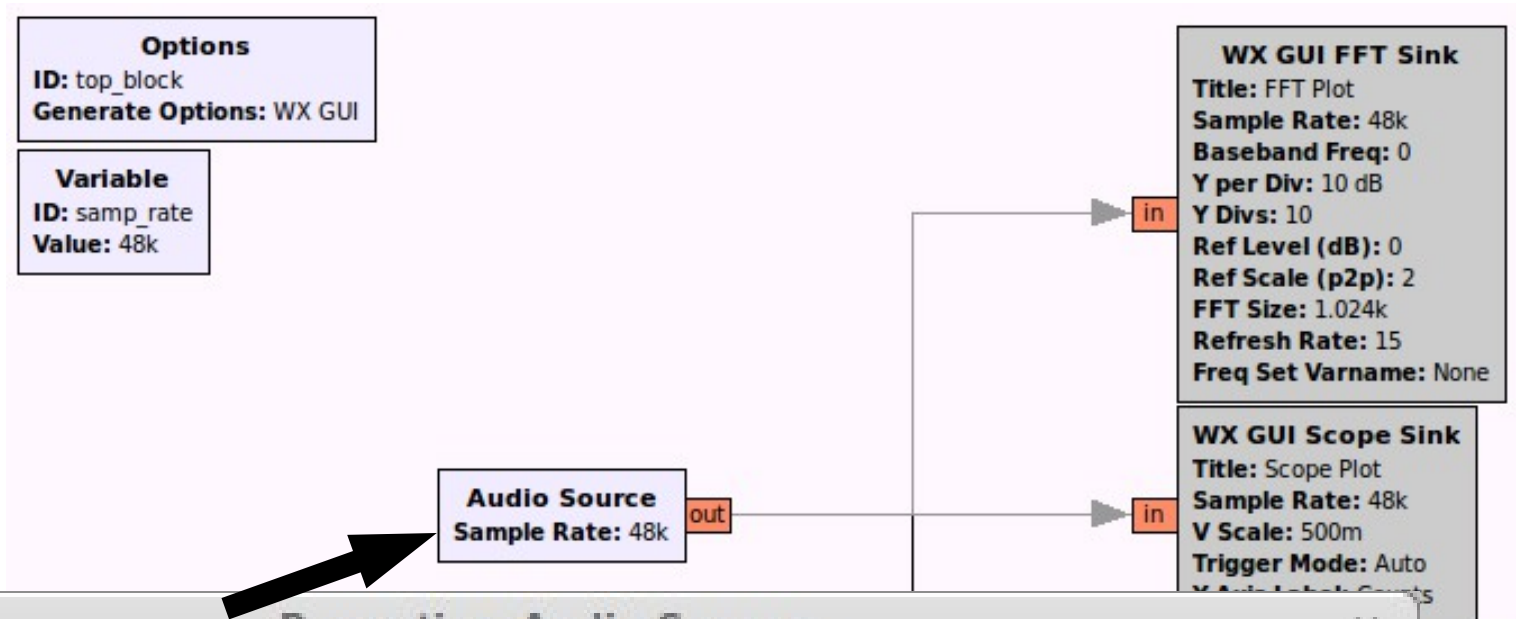
Same sine wave as before, but now we hear it emanating from the computer's speakers.

Lab 3: Audio



Visualise the audio sampled by a soundcard on a time-based scrolling FFT (waterfall/spectrogram).

Lab 3: Audio



Properties: Audio Source

General

Advanced

Documentation

Parameters are identical to the Audio Sink

ID

audio_source_0

Sample Rate

samp_rate

Device Name

OK to Block

Yes

Num Outputs

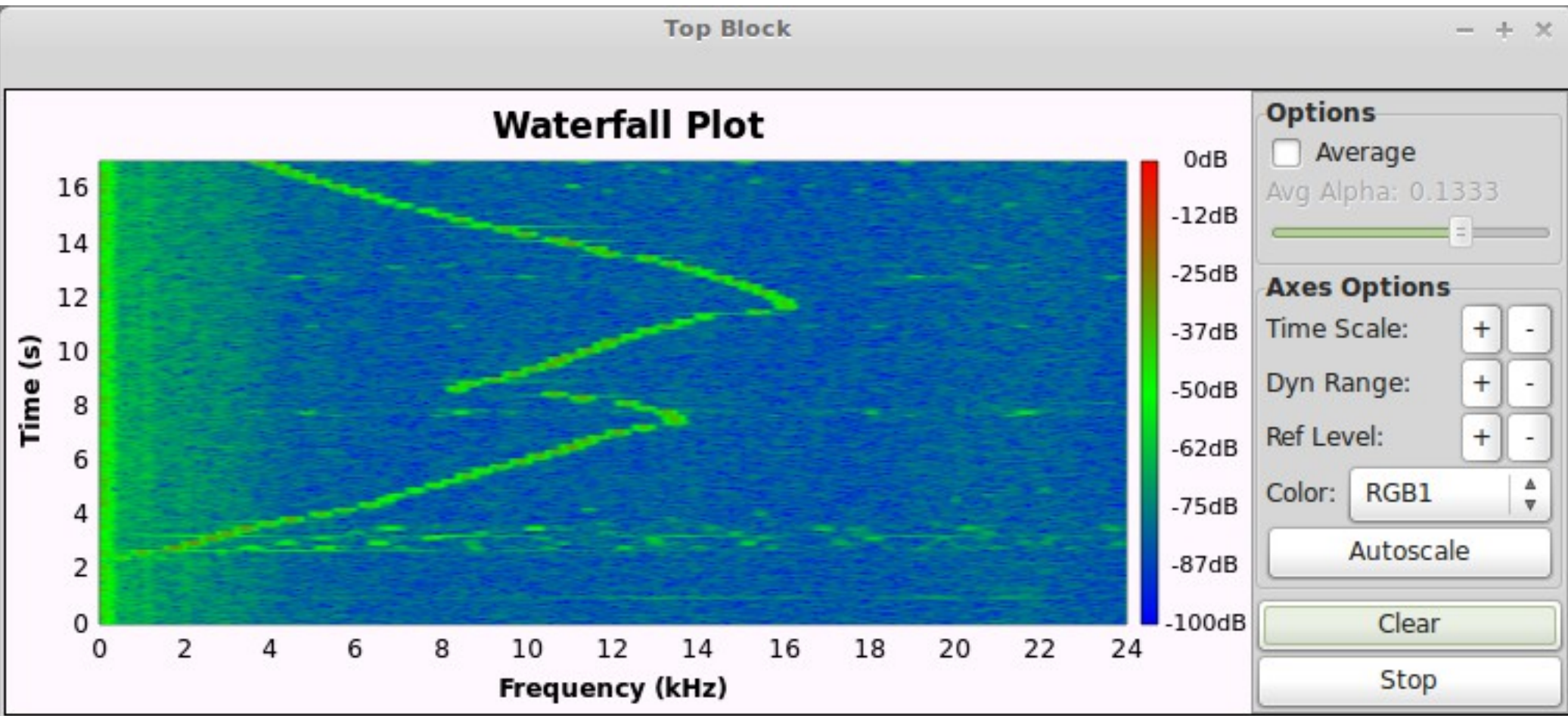
1

Lab 3: Audio



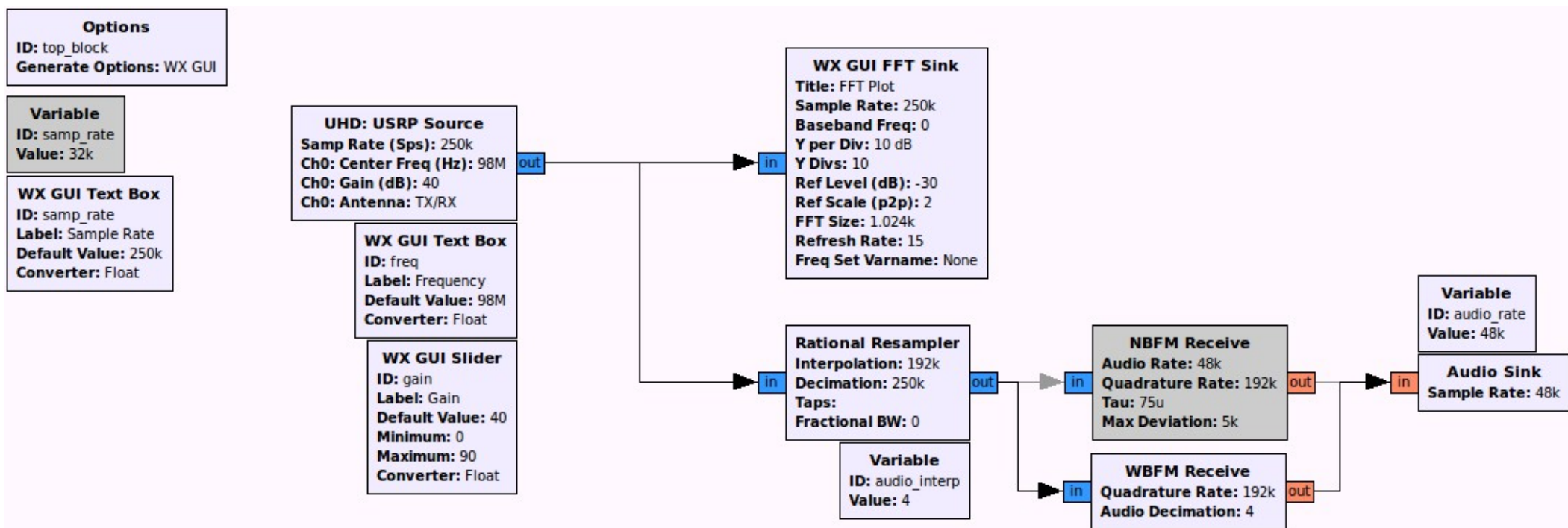
Parameters are identical to the FFT Sink

Lab 3: Audio



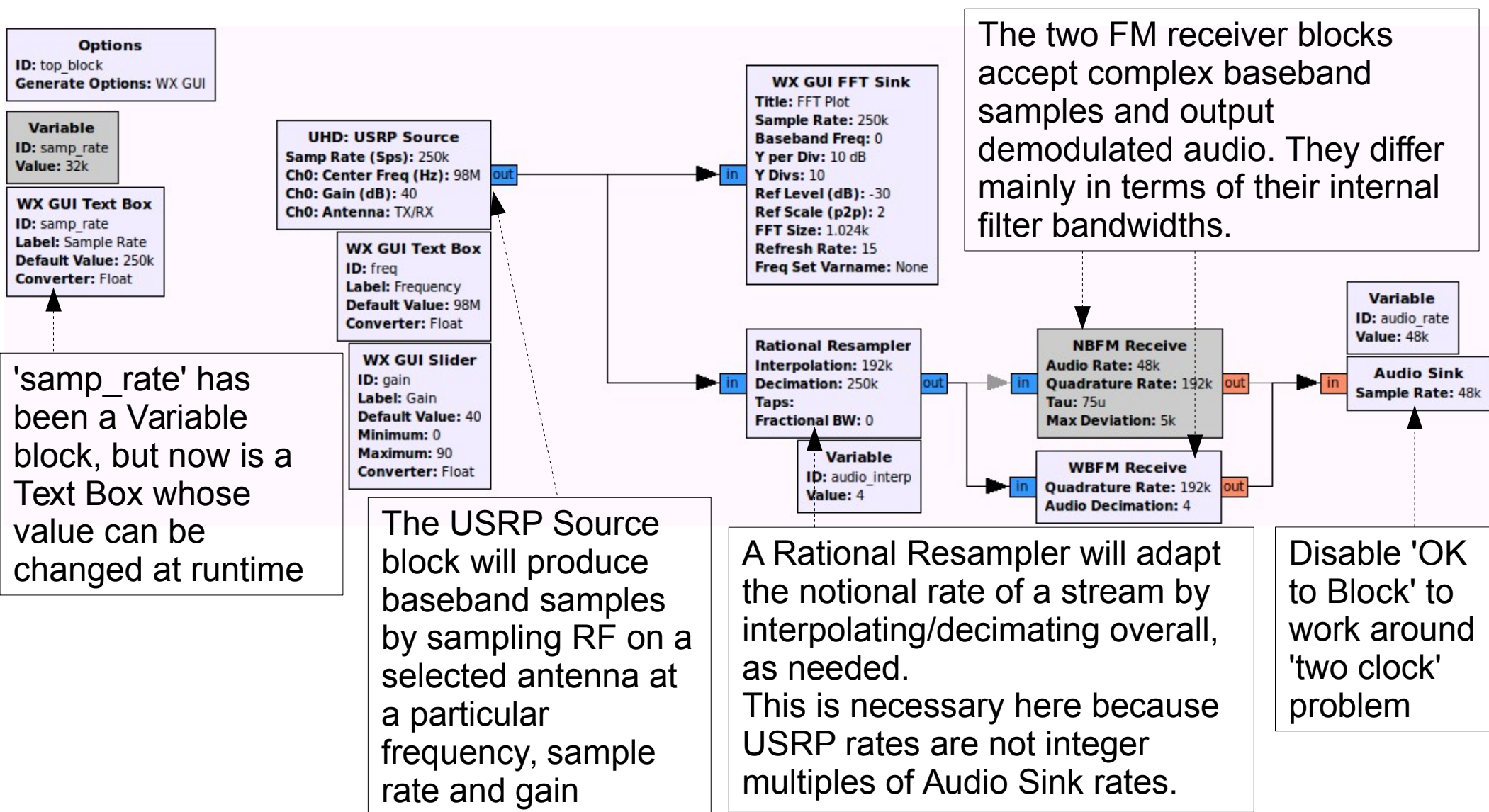
Running the sine wave generator program at the same time, and changing the frequency. This is a rough 'loopback' test where the computer's microphone listens to its speakers.

Lab 4: FM RX



Receive a baseband signal using a USRP and listen to it using a narrow- or wide-band FM demodulator

Lab 4: FM RX



Lab 4: FM RX

Tip: usually all parameters can be left as they are (except for sample rate, frequency, gain and antenna).

Mapping from physical (USRP) channel index to logical (GRC port) channel index (zero-based). Leave as the empty list '[]' for the default linear mapping.

ID: top_b
Generat

Variat
ID: samp
Value: 32k

WX GUI Text Box
ID: samp_rate
Label: Sample Rate
Default Value: 250k
Converter: Float

Ch0: Center Freq (Hz): 98M
Ch0: Gain (dB): 40
Ch0: Antenna: TX/RX

WX GUI Text Box
ID: freq
Label: Frequency
Default Value: 98M
Converter: Float

Sets the number of output ports and duplicates the channel-specific parameters accordingly.

Maximum: 90

Converter: Float

Tip: To be certain about any of the possible parameter values, consult the online documentation for your device and/or daughterboard. You can also run 'uhd_usrp_probe' in a terminal for hardware specs. Watch your console during runtime for any warning messages from UHD regarding invalid settings!

Properties: UHD: U

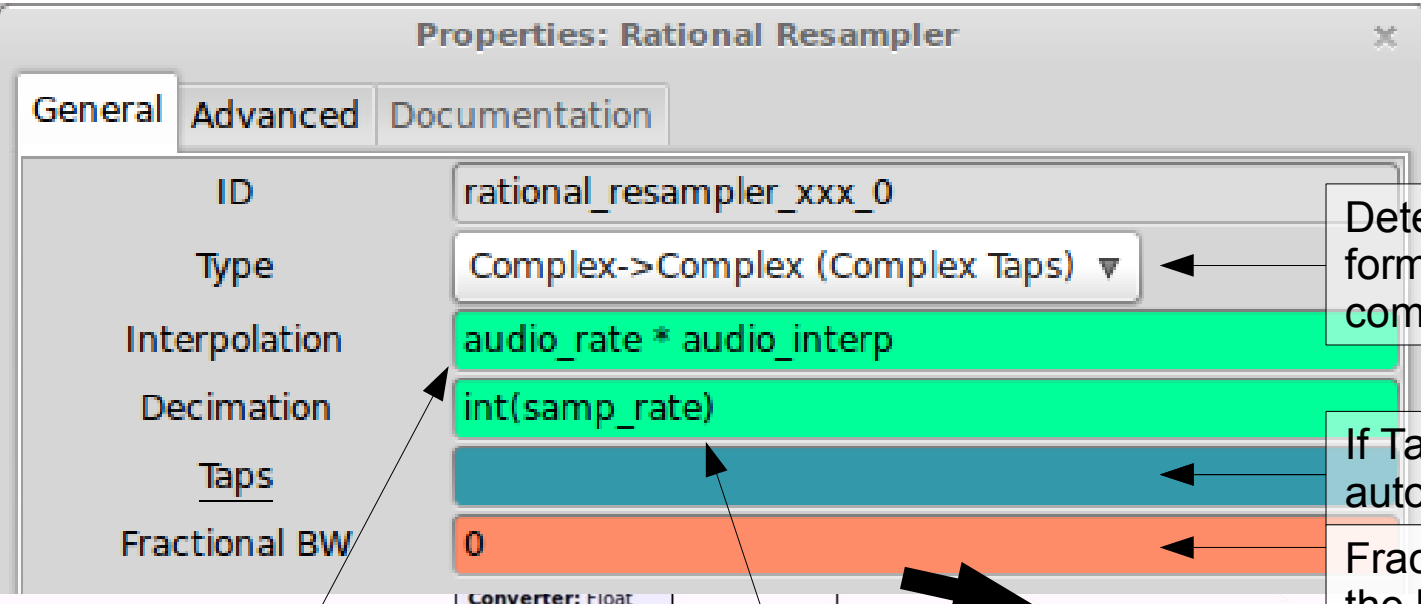
General Advanced Documentation

ID	uhd_usrp_source_0	
Output Type	Complex float32 ▼	Sample type on output port
Wire Format	Automatic	Sample type from USRP
Stream args		RX streamer options
Stream channels	[]	
Device Addr		Same as UHD device args
Sync	don't sync ▼	
Clock Rate (Hz)	Default	
Num Mboards	1	These will be covered later
Mb0: Clock Source	Default	
Mb0: Time Source	Default	
Mb0: Subdev Spec		Selects a 'side', e.g. A:A or A:B
Num Channels	1 ▼	
Samp Rate (Sps)	samp_rate	Valid range depends on hardware
Ch0: Center Freq (Hz)	freq	Valid range depends on hardware
Ch0: Gain (dB)	gain	Valid range depends on hardware
Ch0: Antenna	'TX/RX'	Usually 'TX/RX' or 'RX2'
Ch0: Bandwidth (Hz)	0	Usually 0

Lab 4: FM RX

- This example uses the USRP B200
- Valid ranges:
 - Antenna: TX/RX, RX2
 - Frequency: 70 MHz – 6 GHz
 - RX Gain: 0 – 73 (default of ~25 is a good starting point)
 - Sample Rate: 62.5 ksps – 56 Msps (62.5e3 - 56e6)
 - Default **M**aster **C**lock **R**ate = 32e6 (max: 61.44e6)
 - (MCR / sample rate) **must** be an integer, and **should** be divisible by 4 for the best RF performance (flat spectrum)
 - MCR can be changed with “master_clock_rate=X” in Device Addr, where X is new MCR in Hz (e.g. 40e6)
- A 'O' on the console indicates an overrun, and occurs when the host is not able to consume samples quickly enough.

Lab 4: FM RX



The screenshot shows the 'Properties: Rational Resampler' dialog box with the 'General' tab selected. The dialog has three tabs: 'General', 'Advanced', and 'Documentation'. The 'General' tab contains the following fields:

Property	Value
ID	rational_resampler_xxx_0
Type	Complex->Complex (Complex Taps) ▼
Interpolation	audio_rate * audio_interp
Decimation	int(samp_rate)
Taps	
Fractional BW	0

Annotations with arrows point to the following fields:

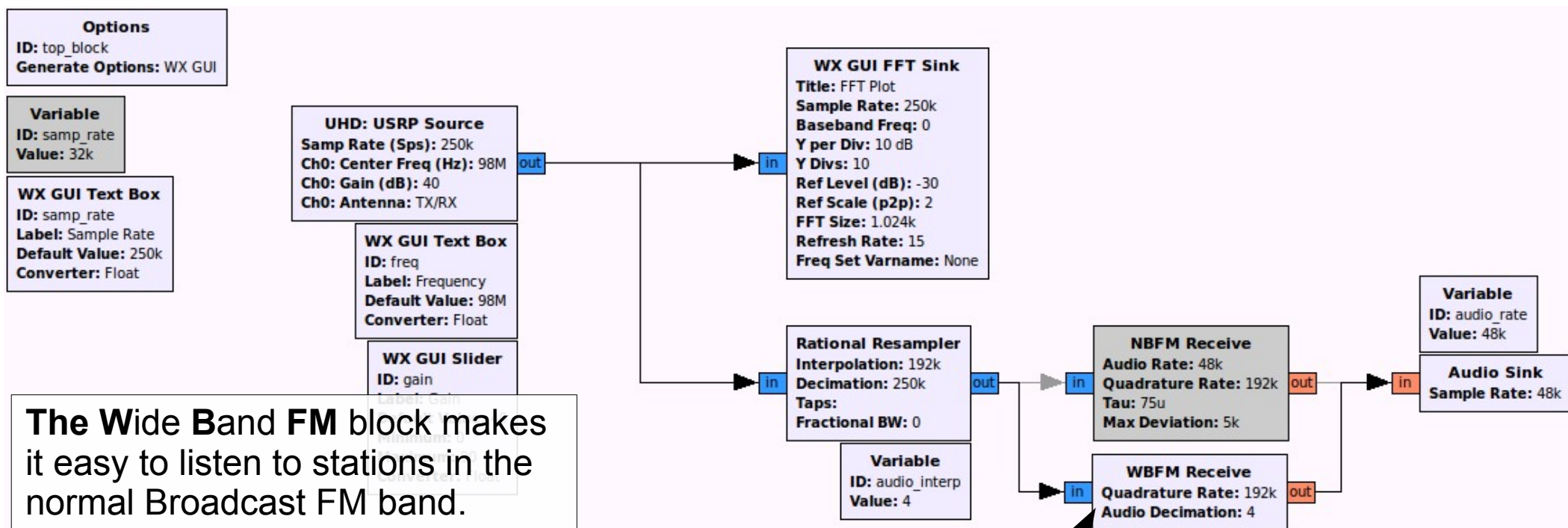
- Type:** Determine the input & output sample format, and what format (real or complex) the filter taps will be
- Decimation:** If Taps is left blank, the taps are automatically computed
- Fractional BW:** Fractional BW affects the shape of the low-pass filter that is generated when no filter taps are supplied. Specifically it determines how steep the low-pass rolloff is. Leaving the default 0 tells the code to select a reasonable default (currently 0.4)

At the bottom of the dialog, there is a 'Converter' dropdown set to 'Float' and a 'Block Size' field set to 192k.

We need to adapt the USRP rate to something suitable for the Audio Sink. The default 'samp_rate' value is 250e3, which sets the rate at which the USRP produces samples. The Audio Sink is configured for a sample (consumption) rate of 48e3, but $(250000 / 48000)$ is not an integer. We can cheat here and set Decimation to be the incoming notional sample rate (250000), and the Interpolation to be a different (non-divisible) outgoing notional sample rate (192000*). The code will calculate the GCD. Since the parameters must be integers, and 'samp_rate' is a floating-point number, we use the Python function 'int' to convert it to an integer.

* 'audio_rate' is multiplied by 4 ('audio_interp') because the demodulator blocks will perform additional decimation (by 4). Specifically the WBFM block should be given a high rate (i.e. high bandwidth signal since FM broadcast channels are 200 kHz wide).

Lab 4: FM RX



Properties: WBFM Receive

General Advanced Documentation

ID

analog_wfm_rcv_0

Quadrature Rate

$\text{audio_rate} * \text{audio_interp}$

Incoming notional sample rate (192e3)

Audio Decimation

audio_interp

Decimation factor: outgoing rate is (incoming / decimation) = 48000

Lab 4: FM RX

To switch between modulators in this example, simply enable one block and disable the other.

* 'audio_interp' is changed to 1 here as NBFM doesn't need such a high bandwidth signal

The Narrow Band FM block makes it easy to listen to narrow analog and digital channels that use FM or FSK.

UHD: USRP Source
Samp Rate (Sps): 250k
Ch0: Center Freq (Hz): 98M
Ch0: Gain (dB): 40
Ch0: Antenna: TX/RX

WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 250k
Baseband Freq: 0
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): -30
Ref Scale (p2p): 2
FFT Size: 1.024k
Refresh Rate: 15
Freq Set Varname: None

Rational Resampler
Interpolation: 48k
Decimation: 250k
Taps:
Fractional BW: 0

NBFM Receive
Audio Rate: 48k
Quadrature Rate: 48k
Tau: 75u
Max Deviation: 5k

WBFM Receive
Quadrature Rate: 48k
Audio Decimation: 1

Variable
ID: audio_rate
Value: 48k

Audio Sink
Sample Rate: 48k

Properties: NBFM Receive

General Advanced Documentation

ID analog_nbfm_rx_0

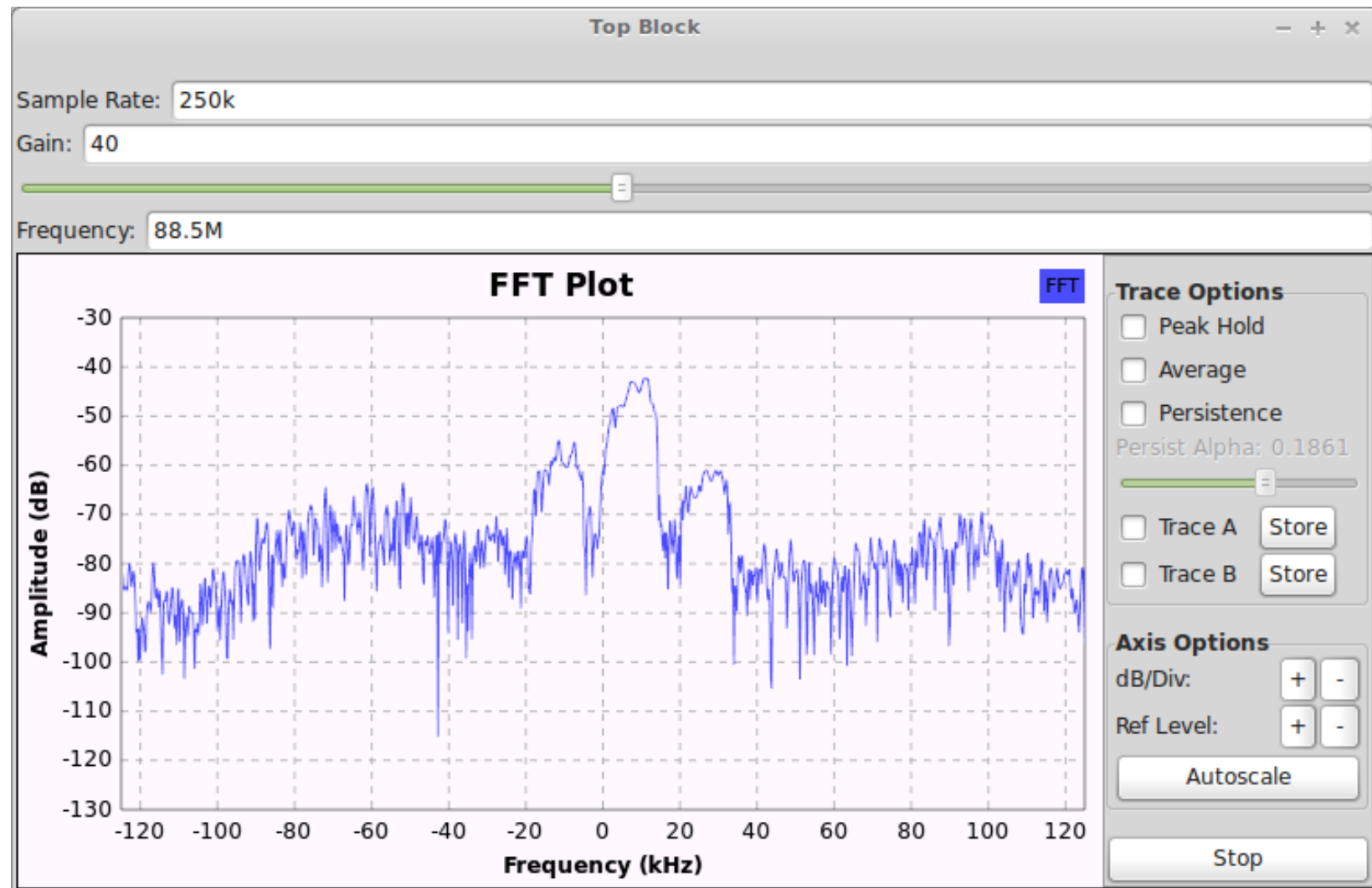
Audio Rate audio_rate Outgoing notional sample rate (48000)

Quadrature Rate audio_rate * audio_interp Incoming notional sample rate (48000*)

Tau 75e-6 FM de-emphasis factor ([more](#) on Wikipedia)

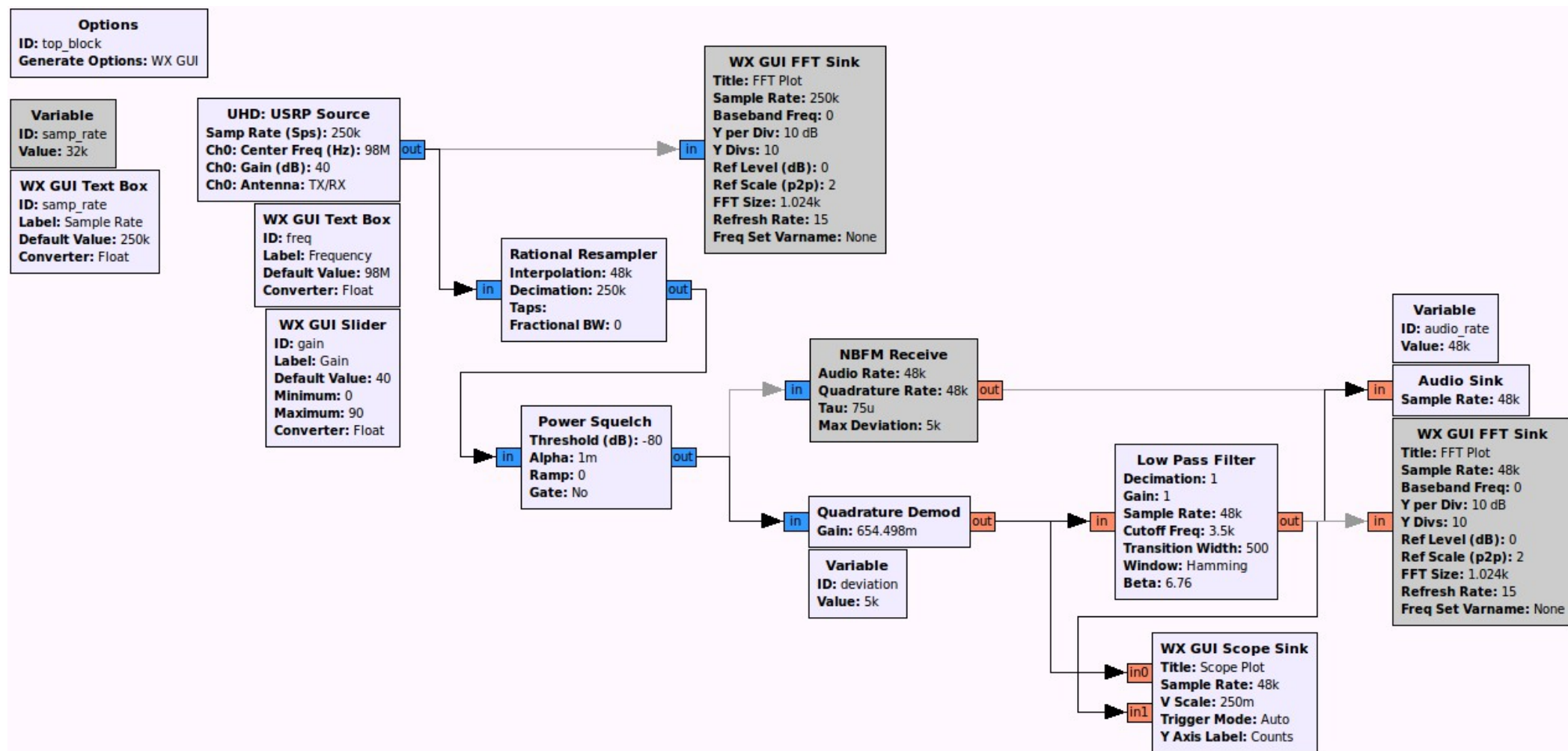
Max Deviation 5e3 Maximum amount signal will deviate from center (0 Hz). This determines output value scaling (e.g. here +5 kHz will be 1.0, -5 kHz will be -1.0).

Lab 4: FM RX



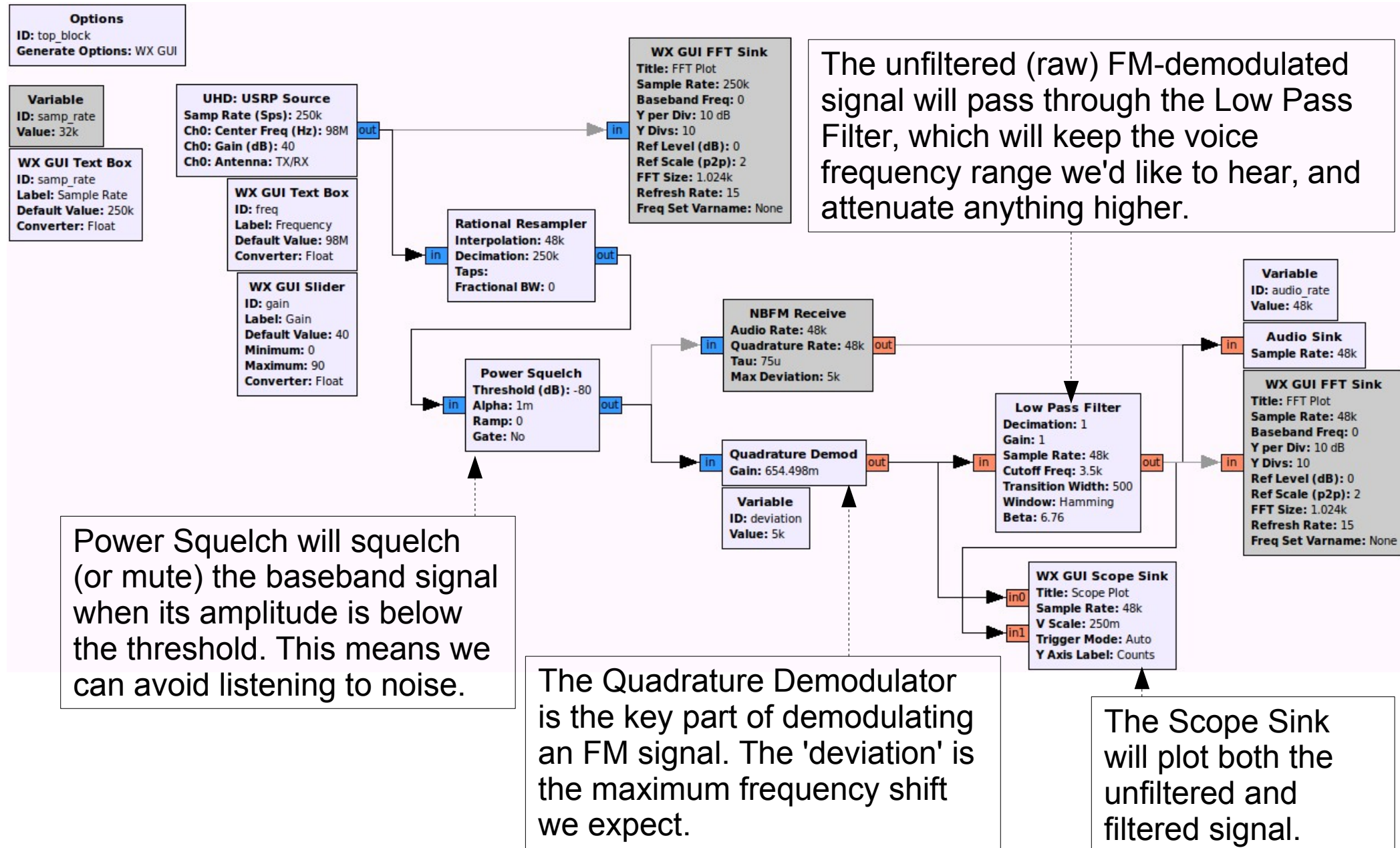
The baseband spectrum (a local radio station) is shown on the FFT plot, and the signal at the center of the spectrum is demodulated producing audio coming out of the host's soundcard.

Lab 4: Manual FM RX

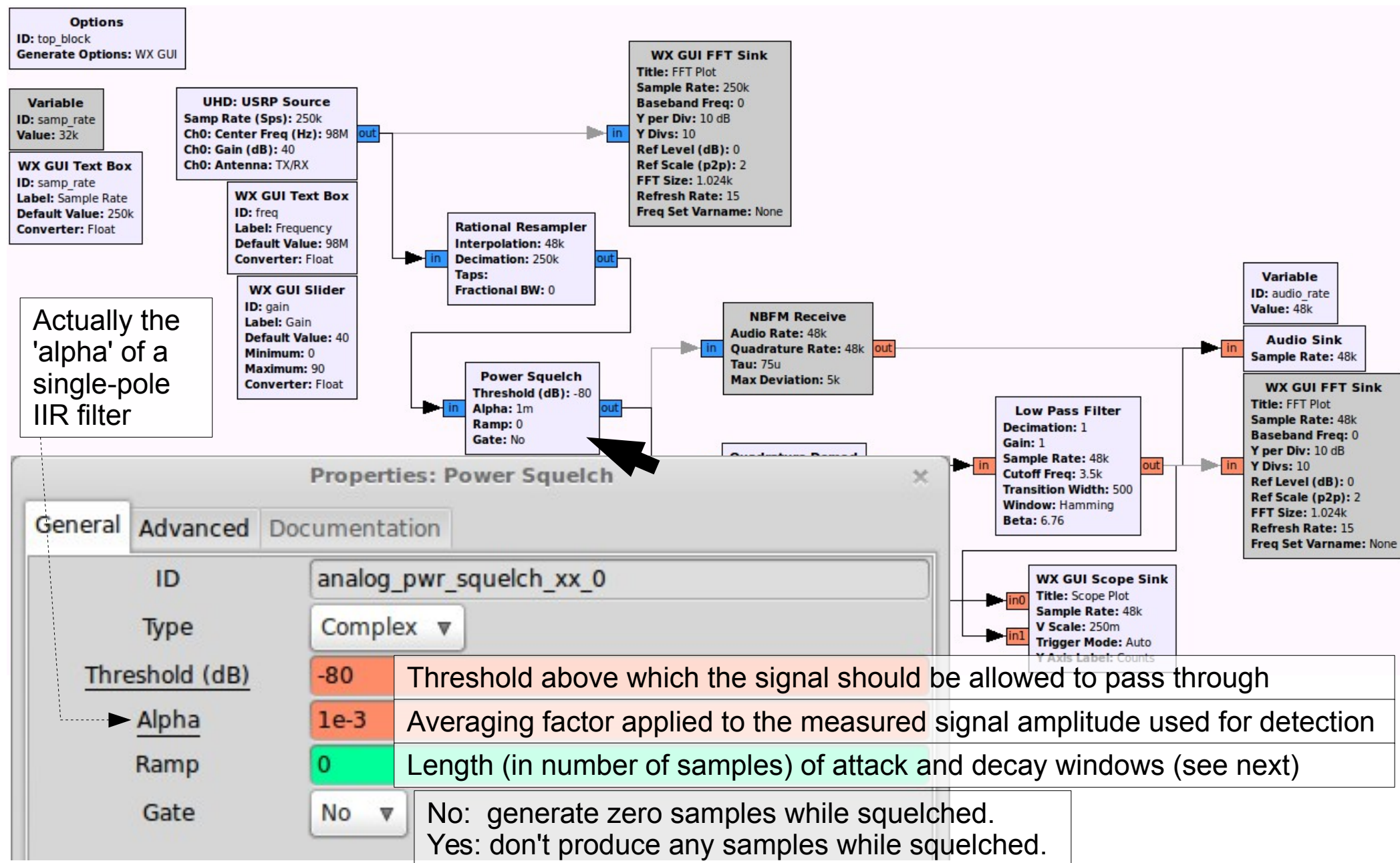


Repeat the Narrow Band FM reception example, but perform the individual demodulation steps.

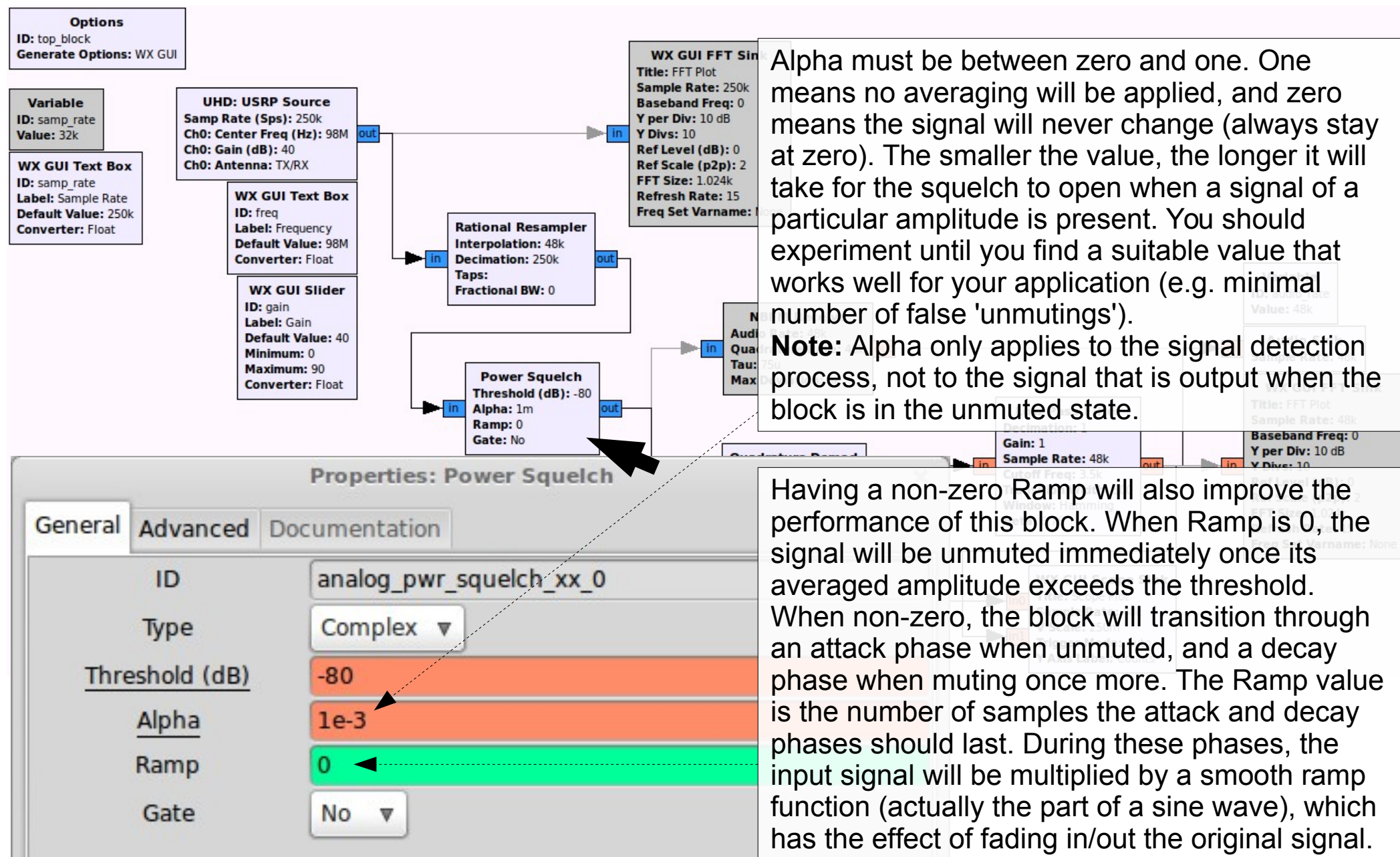
Lab 4: Manual FM RX



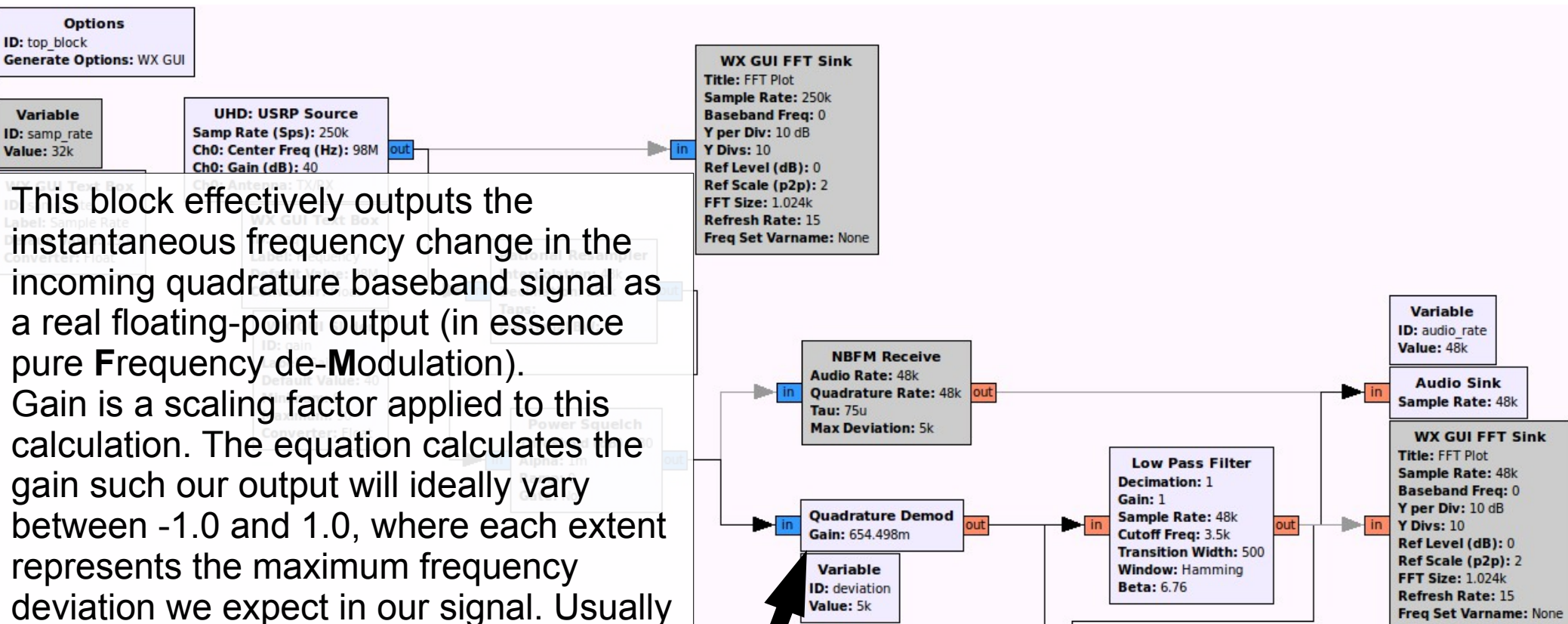
Lab 4: Manual FM RX



Lab 4: Manual FM RX



Lab 4: Manual FM RX

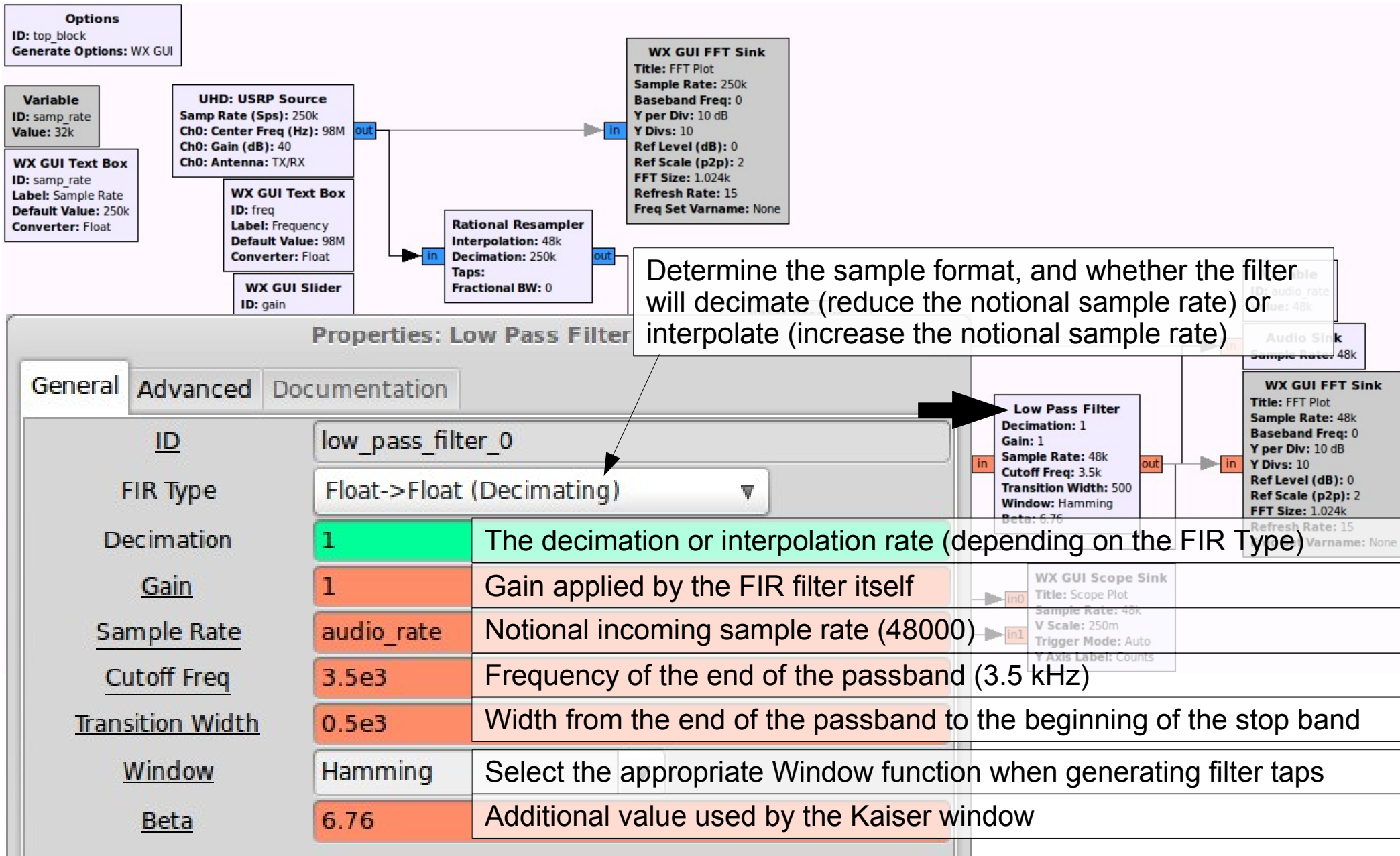


This block effectively outputs the instantaneous frequency change in the incoming quadrature baseband signal as a real floating-point output (in essence pure Frequency de-Modulation). Gain is a scaling factor applied to this calculation. The equation calculates the gain such our output will ideally vary between -1.0 and 1.0, where each extent represents the maximum frequency deviation we expect in our signal. Usually this deviation is set by the transmitter.

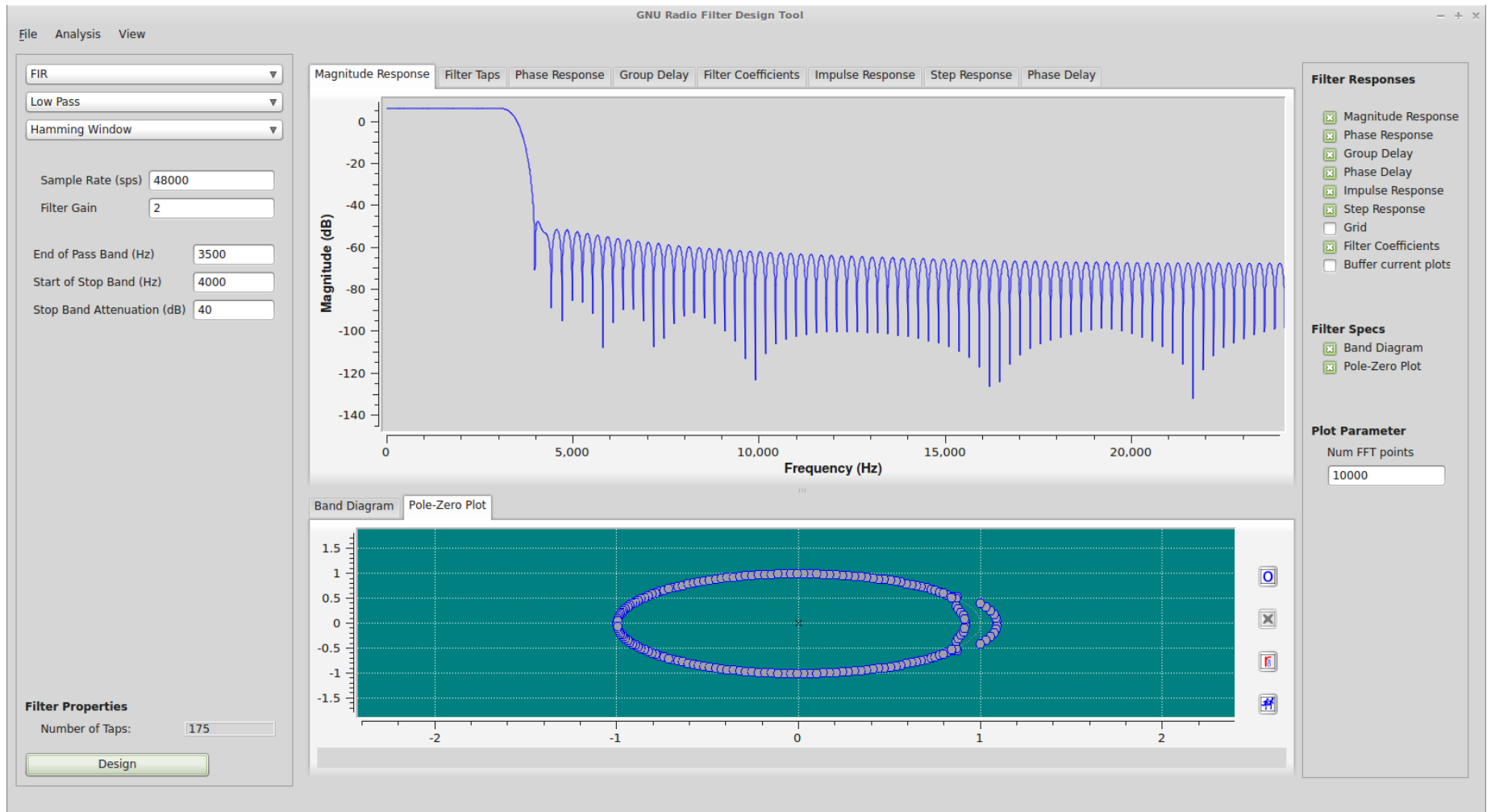
General Advanced Documentation

ID analog_quadrature_demod_cf_0
Gain $(2 * \text{math.pi} * \text{deviation}) / \text{audio_rate}$

Lab 4: Manual FM RX

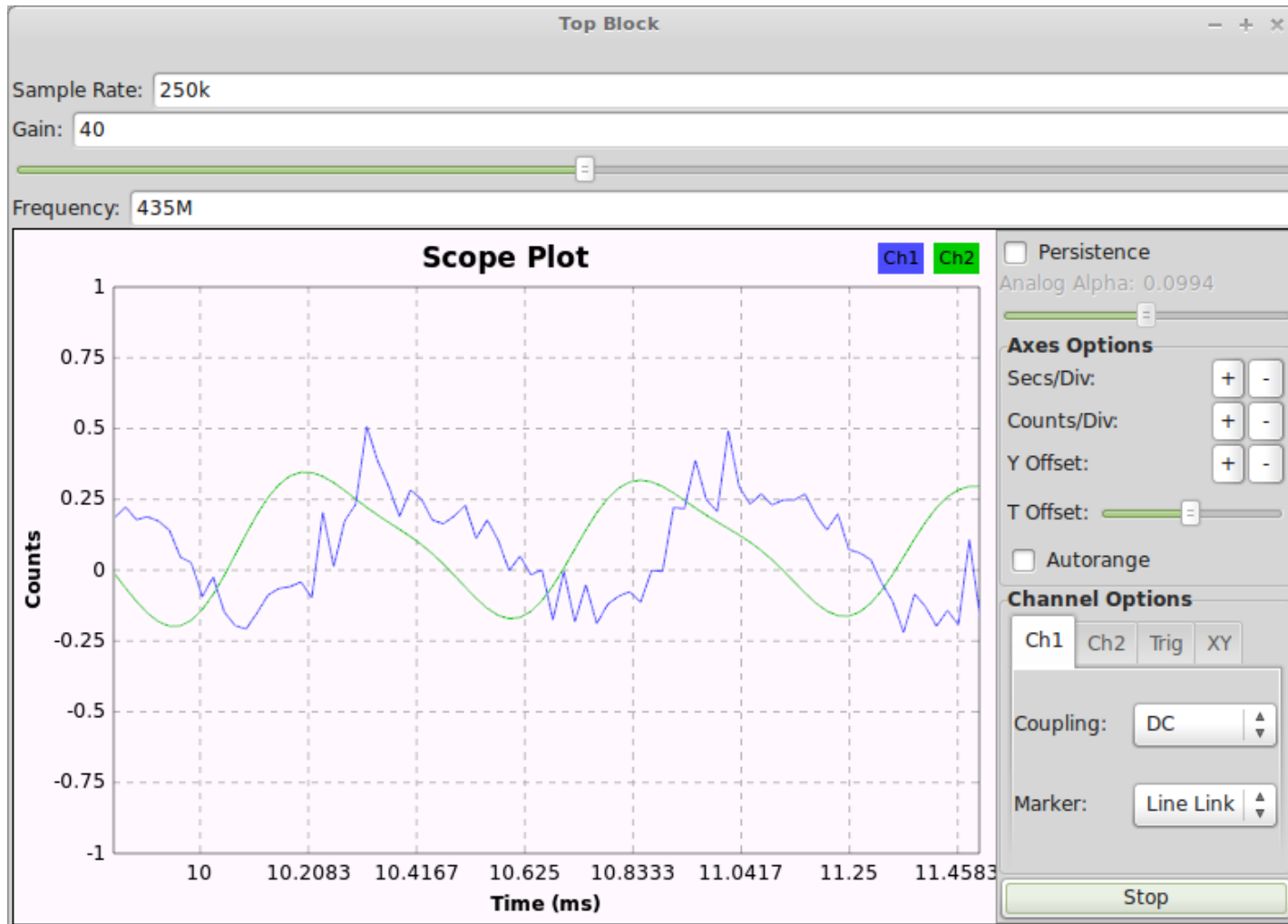


Lab 4: Manual FM RX



The Filter Design Tool (run 'gr_filter_design') is a GUI that allows you to interactively design different types of filters. Once you're happy with your design, you can place an Interpolating/Decimating FIR Filter block into your GRC flowgraph and set its taps using the filter coefficients output by the designer.

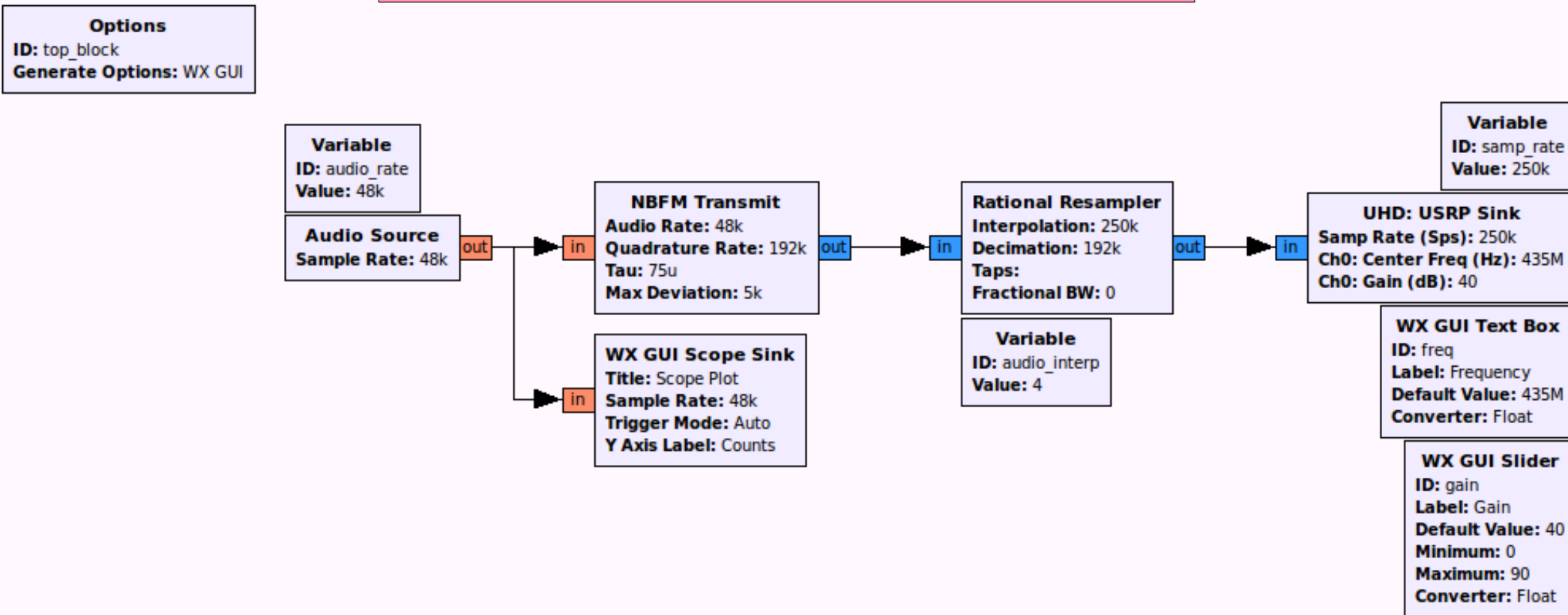
Lab 4: Manual FM RX



The baseband signal will be demodulated as before, however audio will only be heard if there is a strong-enough signal present at the center of the spectrum to open the squelch. The Scope Sink shows the raw demodulated signal (a whistle) in blue, and the low-pass filtered (and therefore slightly delayed) signal in green, which is output to the Audio Sink.

Lab 5: FM TX

You need to have a valid amateur radio (HAM) license to actually transmit on the frequency in this example!



Sample audio from your soundcard and transmit it from a USRP using a **N**arrow **B**and **F**M carrier.

Lab 5: FM TX

You need to have a valid amateur radio (HAM) license to actually transmit on the frequency in this example!

Options
ID: top_block
Generate Options: WX GUI

Variable
ID: audio_rate
Value: 48k

Audio Source
Sample Rate: 48k

NBFM Transmit
Audio Rate: 48k
Quadrature Rate: 192k
Tau: 75u
Max Deviation: 5k

WX GUI Scope Sink
Title: Scope Plot
Sample Rate: 48k
Trigger Mode: Auto
Y Axis Label: Counts

Rational Resampler
Interpolation: 250k
Decimation: 192k
Taps:
Fractional BW: 0

Variable
ID: audio_interp
Value: 4

UHD: USRP Sink
Samp Rate (Sps): 250k
Ch0: Center Freq (Hz): 435M
Ch0: Gain (dB): 40

Variable
ID: samp_rate
Value: 250k

WX GUI Text Box
ID: freq
Label: Frequency
Default Value: 435M
Converter: Float

WX GUI Slider
ID: gain
Label: Gain
Default Value: 40
Minimum: 0
Maximum: 90
Converter: Float

The USRP Sink will transmit the baseband signal.

'OK to Block' has no effect with ALSA. In non-blocking mode (to work around the two clock problem), portaudio (for example) will zero-fill if no audio samples are available.

We apply an additional interpolation factor so the NBFM Transmit block outputs a higher notional sample rate (192000), and then we resample for the USRP (to 250000).

In transmit chain, you will usually be able to control the modulated signal's notional baseband rate (here it is 192000, i.e. prior to resampling for the USRP). This makes for a tradeoff between a higher-rate, potentially higher-quality synthesised baseband signal (at the expense of processing power), or saving CPU cycles for lower-quality. This choice is usually application-/signal-specific.

Lab 5: FM TX

You need to have a valid amateur radio (HAM) license to actually transmit on the frequency in this example!

Options

ID: top_block
Generate Options: WX GUI

Variable

ID: audio_rate
Value: 48k

Audio Source
Sample Rate: 48k

NBFM Transmit
Audio Rate: 48k
Quadrature Rate: 192k
Tau: 75u
Max Deviation: 5k

Rational Resampler
Interpolation: 250k
Decimation: 192k
Taps:
Fractional BW: 0

Variable

ID: samp_rate
Value: 250k

UHD: USRP Sink
Samp Rate (Sps): 250k
Ch0: Center Freq (Hz): 435M
Ch0: Gain (dB): 40

WX GUI Text Box

Properties: NBFM Transmit

General

Advanced

Documentation

ID

analog_nbfn_tx_0

Audio Rate

audio_rate

Incoming notional sample rate (48000)

Quadrature Rate

audio_rate * audio_interp

Outgoing notional sample rate
(192000)

Tau

75e-6

Same as receiver

Max Deviation

5e3

Lab 5: FM TX

You need to have a valid amateur radio (HAM) license to actually transmit on the frequency in this example!

Options

ID: top_block
Generate Options: WX GUI

Variable
ID: audio_rate
Value: 48k

Audio Source
Sample Rate: 48k

NBFM Transmit
Audio Rate: 48k
Quadrature Rate: 192k
Tau: 75u
Max Deviation: 5k

Rational Resampler
Interpolation: 250k
Decimation: 192k
Taps:
Fractional BW: 0

Variable
ID: samp_rate
Value: 250k

UHD: USRP Sink
Samp Rate (Sps): 250k
Ch0: Center Freq (Hz): 435M
Ch0: Gain (dB): 40

Properties: Rational Resampler

General

Advanced

Documentation

ID

rational_resampler_xxx_0

Type

Complex->Complex (Complex Taps) ▼

Interpolation

$\text{int}(\text{samp_rate} * 1.0)$

Outgoing notional sample rate (250000)

Decimation

$\text{audio_rate} * \text{audio_interp}$

Incoming notional sample rate (192000)

Taps

Fractional BW

0

WX GUI Text Box
ID: freq
Label: Frequency
Default Value: 435M
Converter: Float

WX GUI Slider
ID: gain
Label: Gain
Default Value: 40
Minimum: 0
Maximum: 90
Converter: Float

Lab 5: FM TX

You need to have a valid amateur radio (HAM) license to actually transmit on the frequency in this example!

Options

ID: top_block
Generate Options: WX GUI

Variable

ID: audio_rate
Value: 48k

Audio Source
Sample Rate: 48k

Tip: A 'U' on the console indicates the USRP ran out of samples to transmit, so the host isn't producing them quickly enough.

Tip: Certain valid ranges might be different between RX and TX for the same device. E.g. B200 TX gain range is 0 – 89.5.

Properties: UHD: USRP Sink

General Parameters are identical to the USRP Source

ID	uhd_usrp_sink_0
Input Type	Complex float32 ▾
Wire Format	Automatic ▾
Stream args	▾
Stream channels	[]
Device Addr	▾
Sync	don't sync ▾
Clock Rate (Hz)	Default ▾
Num Mboards	1 ▾
Mb0: Clock Source	Default ▾
Mb0: Time Source	Default ▾
Mb0: Subdev Spec	▾
Num Channels	1 ▾
Samp Rate (Sps)	samp_rate
Ch0: Center Freq (Hz)	freq
Ch0: Gain (dB)	gain
Ch0: Antenna	▾
Ch0: Bandwidth (Hz)	0

Variable

ID: samp_rate
Value: 250k

UHD: USRP Sink

Samp Rate (Sps): 250k
Ch0: Center Freq (Hz): 435M
Ch0: Gain (dB): 40

WX GUI Text Box

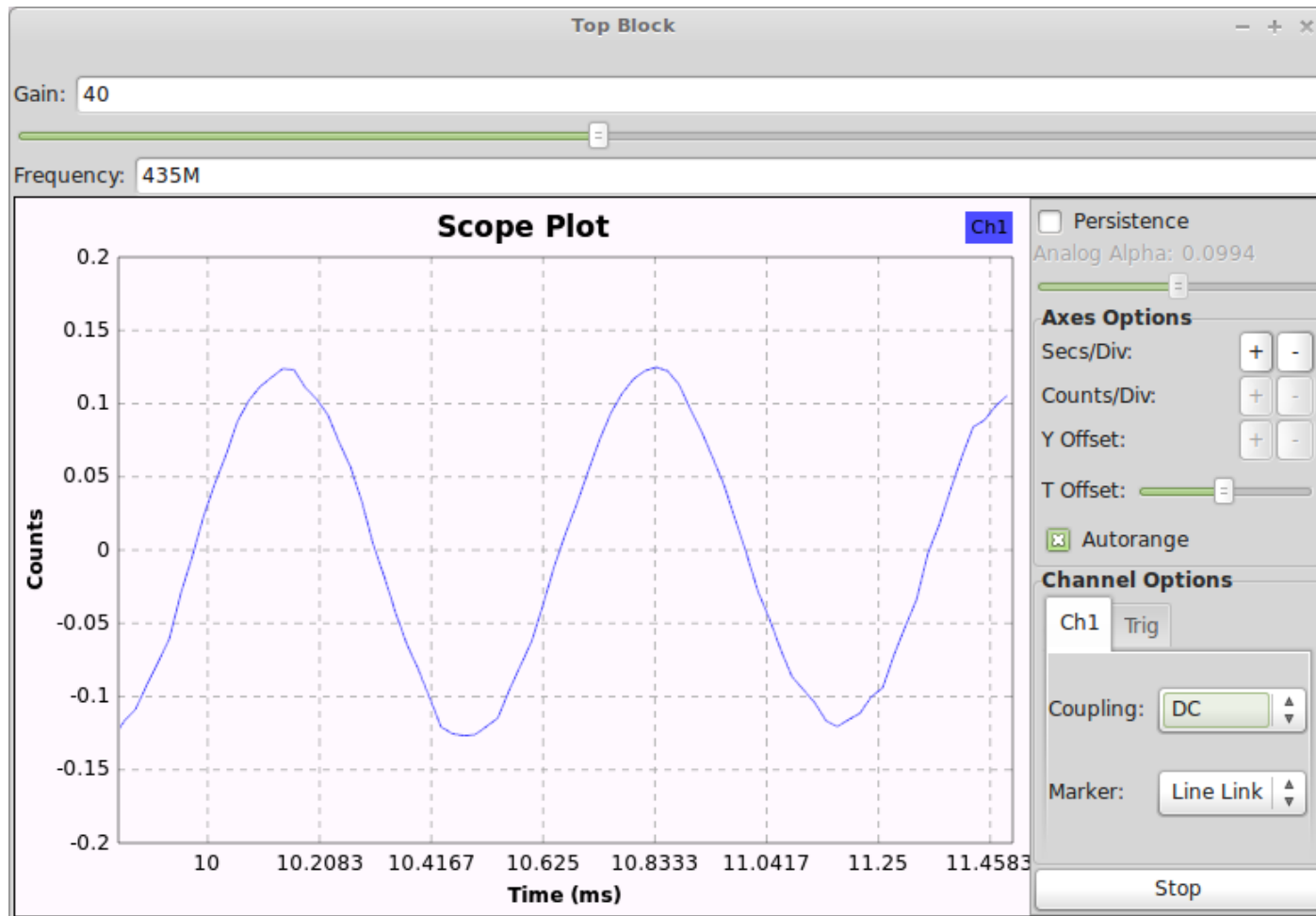
ID: freq
Label: Frequency
Default Value: 435M
Converter: Float

WX GUI Slider

ID: gain
Label: Gain
Default Value: 40
Minimum: 0
Maximum: 90
Converter: Float

Lab 5: FM TX

You need to have a valid amateur radio (HAM) license to actually transmit on the frequency in this example!



The audio (a whistle) picked up by the sound card will be shown in the scope plot, and transmitted by the USRP at the selected frequency.

Lab 5: FM TX

- If you see lots of the letter 'U' in the console, the transmit chain of the USRP is experiencing underruns: samples cannot be produced quickly enough by the host.
- In this example (under Linux/ALSA) it will occur because of the 'two clock' problem, but cannot be fixed by changing 'OK to Block' since the Audio Source is producing samples that are all being consumed without issue, but it happens to be doing this a little too slowly.

Lab 5: FM TX

- It is possible to cheat by adding a 'fudge', or 'twiddle', factor to the Interpolation rate at the Rational Resampler.
- In the example it was:
 - `int(samp_rate * 1.0)`
- We can ask the resampler to produce *more* samples for the same number of input samples so that the USRP will always have enough samples to transmit
- The Interpolation rate would become:
 - `int(samp_rate * 1.01)`
 - The notional output rate was increased by 1% ($1.0 + 0.01$), which equals = 252500.
The USRP UHD Sink will *still* consume at 250000.

GNU Radio:

<http://gnuradio.org/>

CGRAN:

<http://cgran.org/>

Ettus Research:

<http://ettus.com/>

UHD Docs:

http://files.ettus.com/uhd_docs/doxymanual/html/