

Using radio to transmit text and exfil data

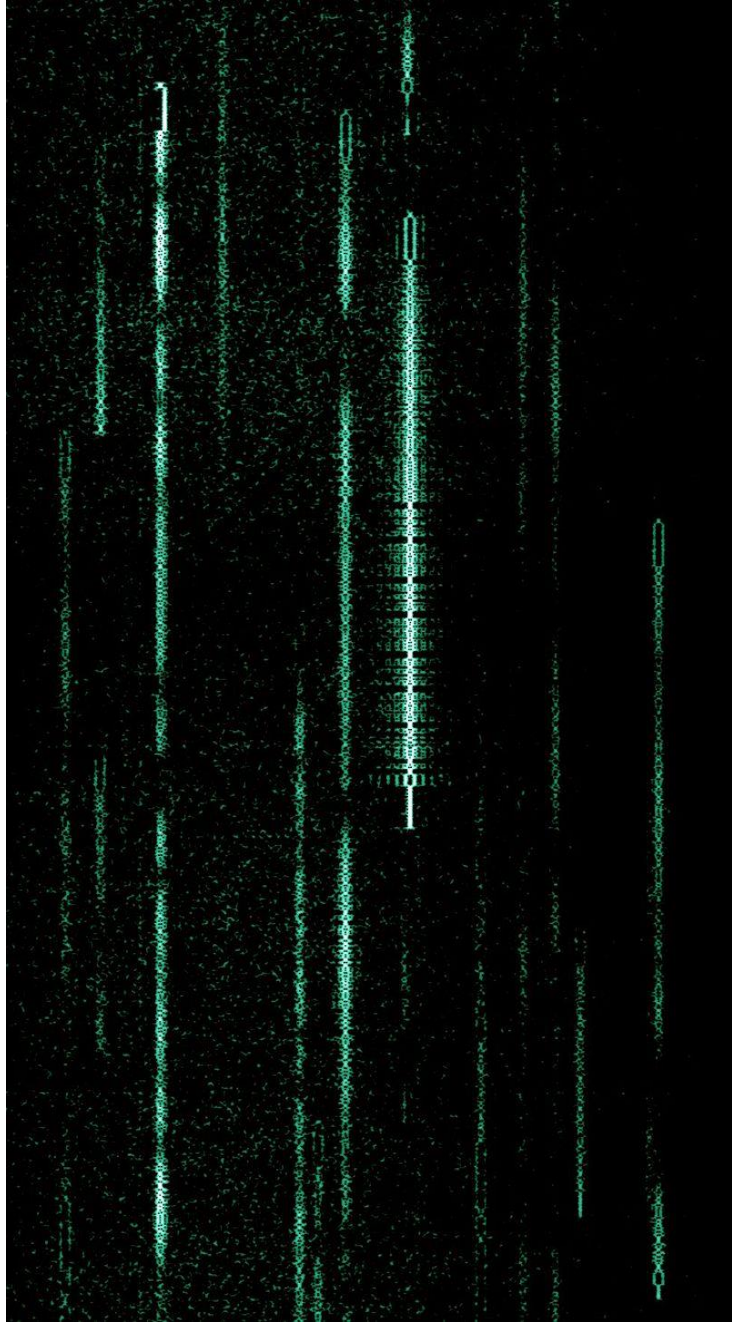
This Workshop will...

- Provide an introduction to GNURadio Companion without the need for special hardware
- Provide an introduction to sending text messages over the air
- Get you started with setting up GNU Radio for raspberry pi projects

Happy St. Patrick's Day!



What does PSK31 look like?



What does PSK31 sound like?

Tell me more about PSK31

What is PSK31?

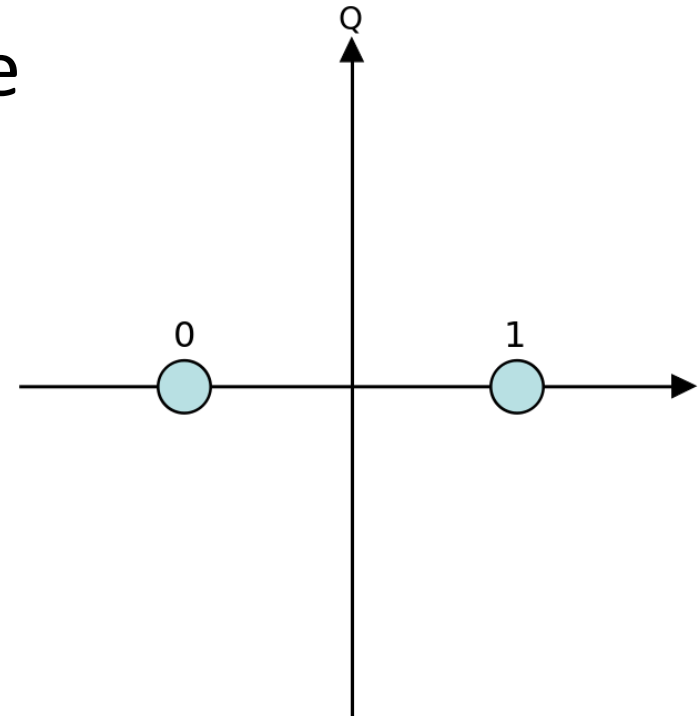
- Created by Peter Martinez (G3PLX)
- Released in 1998
- Used by amateur radio operators for real time chat
- Phase Shift Keying
- 31.25 baud rate & bits per second
- Baud rate matches a typical typing speed of 50 wpm approx.

What is PSK31?

- Phase modulated
- Remember the three main types of modulation are:
 - Amplitude-shift keying (ASK)
 - Frequency-shift keying (FSK)
 - Phase-shift keying (PSK)

What is PSK31?

- Uses binary phase-shift keying (BPSK/2-PSK)
- Two phases
- 180 degrees apart (anywhere on plane)
- BPSK is most tolerant to noise



What is PSK31?

- Not packet based
- Modulates in phase an audio signal
- Audio signal then modulates in amplitude a carrier sent over the air
- Can be used by equipment designed for audio

What is PSK31?

- 0 = phase shift of π radians
- 1 = no phase shift
- Characters = varicode
- Start of character = 00
- Data rate = 31.25 baud

Varicode

- Includes most of 7-bit ASCII characters
- Start with: 1
- End with: 1
- Never: have 00
- Break between characters: 00
- Prefix with long string of 000s

Varicode

- BSides Canberra:
- 11101011 1101111 1101 101101 11
10111 1 1011011101 1011 1111
1011111 11 10101 10101 1011

```
'a': '1011',  
'b': '1011111',  
'c': '101111',  
'd': '101101',  
'e': '11',  
'f': '111101',  
'g': '1011011',  
'h': '101011',  
'i': '1101',  
'j': '111101011',  
'k': '10111111',  
'l': '11011',  
'm': '111011',  
'n': '1111',  
'o': '111',  
'p': '111111',  
'q': '110111111',  
'r': '10101',  
's': '10111',  
't': '101',  
'u': '110111',  
'v': '1111011',  
'w': '1101011',  
'x': '11011111',  
'y': '1011101',  
'z': '111010101',
```

Generating a message

- Sample rate = 48,000Hz (audio rate)
- 1 channel
- 2 bytes per sample

Materials

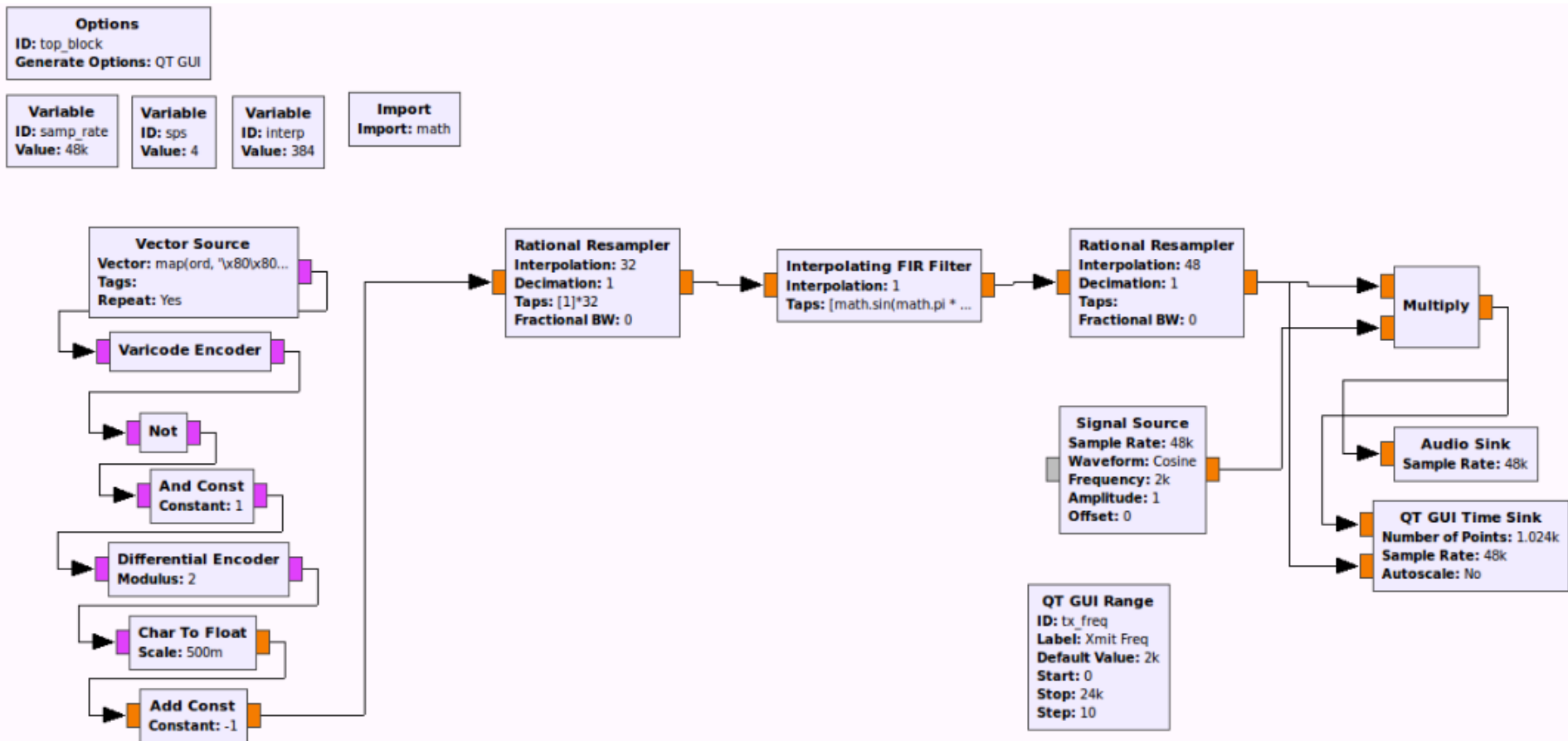
- Using this PSK31 example as a base:
<https://github.com/tkuester/gr-psk31/>
- All files, install instructions and modifications are described here:
<https://github.com/0xsh/Workshops/tree/master/BSides-Canberra-2017>

Some terms

- **Bit rate (bps):** Speed of data in bits per second. With PSK31 where 1 bit per symbol, baud & bit rates are the same.
- **Baud rate (Bd):** number of symbols transmitted over a line per second.
- **Bandwidth (Hz):** difference between upper and lower frequencies of a given spectrum e.g. can have multiple channels within a bandwidth.
- **Baseband:** signal transmitted without modulation i.e. no shift in the range of frequencies of the signal, and is a low frequency - contained within the band of frequencies from close to 0 hertz up to a higher cut-off frequency or maximum bandwidth.

Transmit PSK31 with an audio card

Transmit Graph



Vector Source

Vector source

- A vector is a dynamically sized sequence of objects (compared to a fixed array)
- Interpreted as list of integers
- Truncated to bytes when output
- Vector: “map(ord, ‘BSides Canberra’)”
- Repeat: Yes

Vector Source

Vector Source
Vector: map(ord, '\x80\x80...
Tags:
Repeat: Yes

Varicode Encoder

Not

And Const
Constant: 1

Differential Encoder
Modulus: 2

Char To Float
Scale: 500m

Add Const
Constant: -1

General | Advanced | Documentation

ID	blocks_vector_source_x_0
Output Type	Byte ▾
Vector	map(ord, 'platypus')
Tags	[]
Repeat	Yes ▾
Vec Length	1

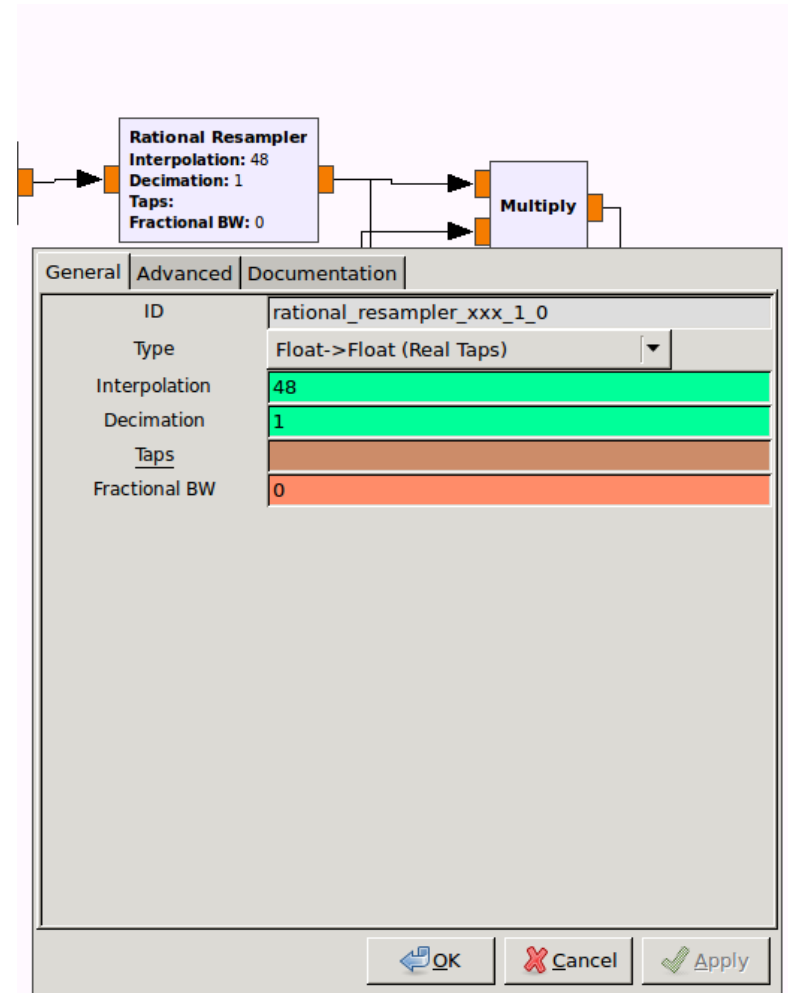
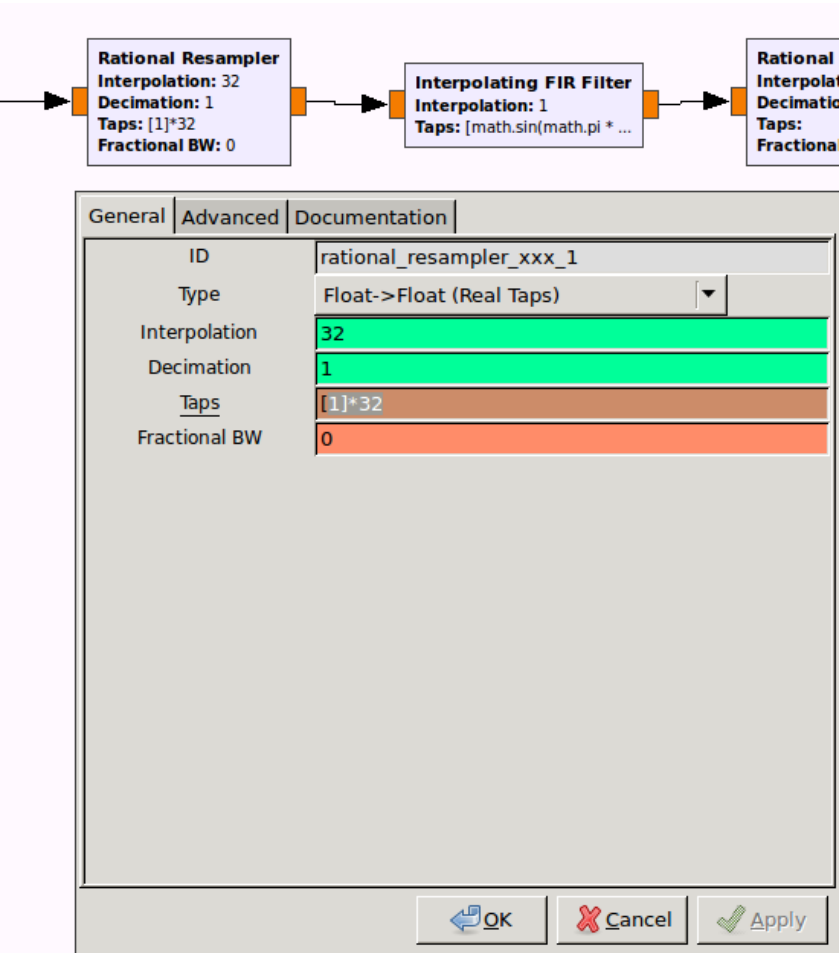
OK Cancel Apply

Rational Resampler

Rational Resampler

- Convert from one sample rate to another
- Combined interpolator (multiply by) & decimator (divide by)
- All following blocks should use newly set sample rate

Rational Resampler

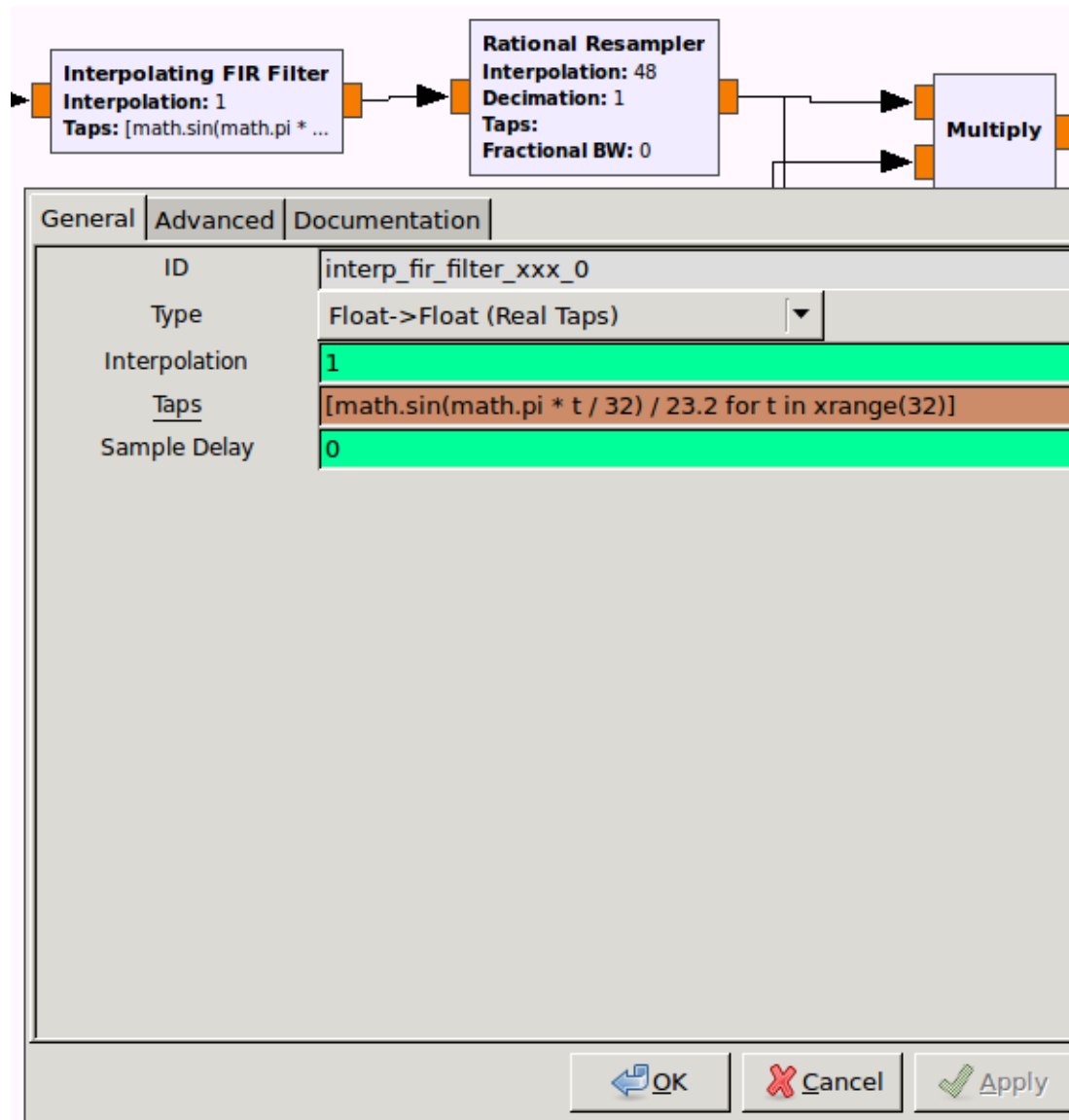


Interpolating FIR Filter

Interpolating FIR Filter

- FIR = Finite Impulse Response filter
- Settles to 0 in finite time
- Tap = a delay
- More taps = more stopband attenuation, less ripple, narrower filters

Interpolating FIR Filter



Audio Sink

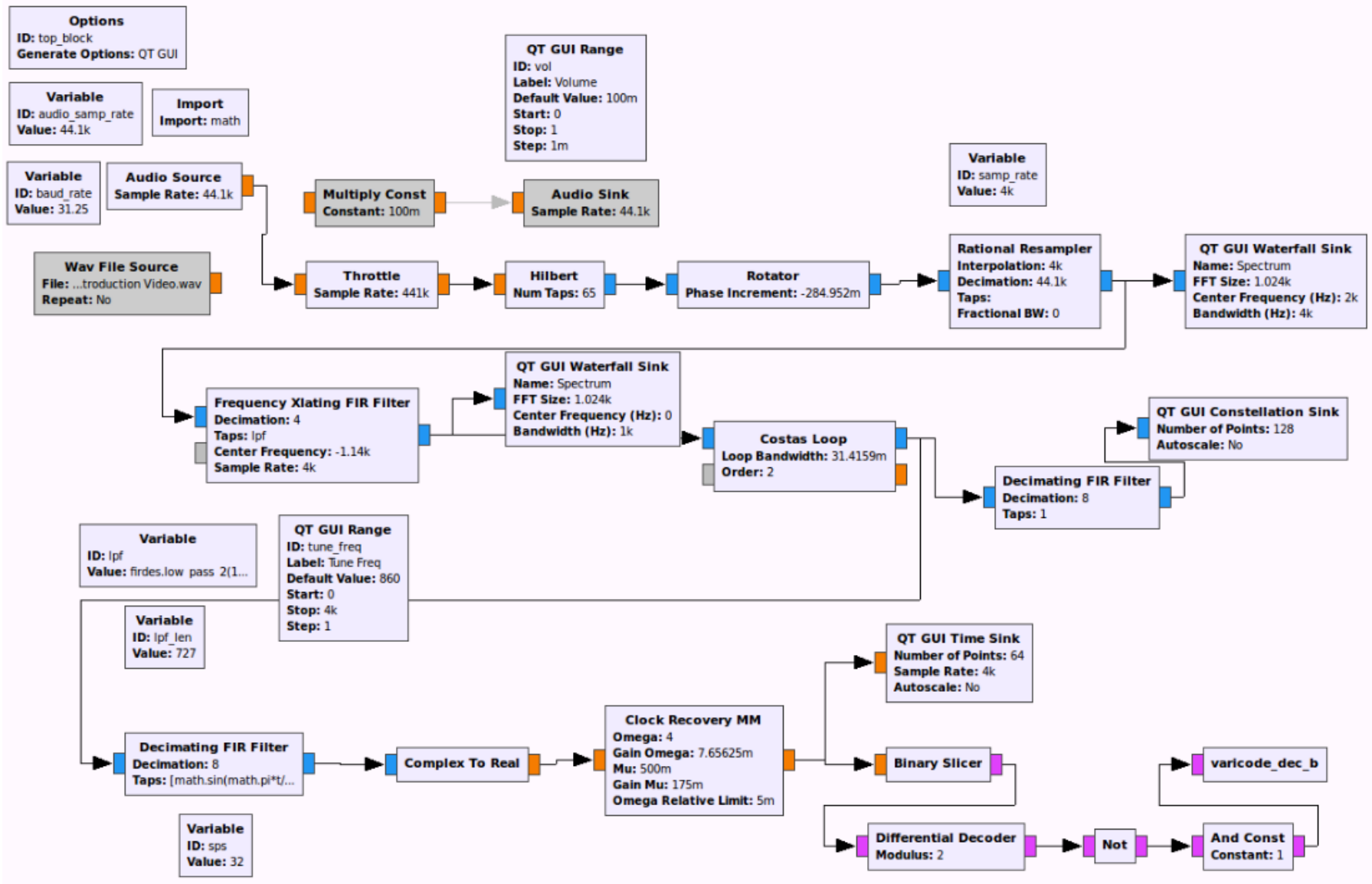
The image shows a software interface for configuring an audio sink. At the top, a block diagram illustrates a 'Signal Source' connected to an 'Audio Sink'. The 'Signal Source' block is labeled with 'Sample Rate: 48k', 'Waveform: Cosine', and 'Frequency: 2k'. The 'Audio Sink' block is labeled with 'Sample Rate: 48k'. Below the diagram is a configuration window with three tabs: 'General', 'Advanced', and 'Documentation'. The 'General' tab is active, displaying a table of configuration parameters. The table has two columns: a label column and a value column. The values for 'Sample Rate', 'Device Name', and 'Num Inputs' are highlighted in green. At the bottom of the window are three buttons: 'OK' (with a blue arrow icon), 'Cancel' (with a red X icon), and 'Apply' (with a green checkmark icon).

General Advanced Documentation		
ID	audio_sink_0_0	
Sample Rate	samp_rate	▼
Device Name		
OK to Block	Yes	▼
Num Inputs	1	

OK Cancel Apply

Receive PSK31 with a microphone

Receive PSK31 with a microphone



Audio Source

Audio Source
Sample Rate: 44.1k

General | Advanced | Documentation

ID	audio_source_0
Sample Rate	audio_samp_rate ▼
Device Name	
OK to Block	Yes ▼
Num Outputs	1

OK Cancel Apply

Throttle

Throttle

- GNU Radio operates at full speed (when no real hardware in way to slow down)
- Need throttle to control rate
- Don't use with real hardware (throttle is a bad clock and will end up with timing issues)

Hilbert Transform

Hilbert Transform

- Truncates the filter to the number of taps
- Introduces a delay into the signal

Hilbert Transform

Block diagram showing the Hilbert Transform process:

```
graph LR; Input --> Hilbert[Hilbert  
Num Taps: 65]; Hilbert --> Rotator[Rotator  
Phase Increment: -284.952m]; Rotator --> Output
```

Configuration window for the Hilbert Transform block:

General | Advanced | Documentation

ID	hilbert_fc_1
Num Taps	65
Window	Hamming
Beta	6.76

Buttons: OK, Cancel, Apply

Additional parameters shown in the top right:

- Rational Resamp
- Interpolation: 4k
- Decimation: 44.1k
- Taps:
- Fractional BW: 0

Rotator

Rotator

- Frequency shifting
- Shift specified as a complex vector (amount of rotation per sample)

Rotator

Rotator
Phase Increment: -284.952m

Rational Resampler
Interpolation: 4k
Decimation: 44.1k
Taps:
Fractional BW: 0

General | Advanced | Documentation

ID	blocks_rotator_cc_0
Phase Increment	$2 \times \text{math.pi} \times -2000 / \text{audio_samp_rate}$

OK Cancel Apply

Rational Resampler

The image shows a block diagram and a configuration dialog for a Rational Resampler. The block diagram at the top shows a 'Rational Resampler' block connected to a 'QT GUI Waterfall Sink' block. The 'Rational Resampler' block has the following parameters: Interpolation: 4k, Decimation: 44.1k, Taps, and Fractional BW: 0. The 'QT GUI Waterfall Sink' block has the following parameters: Name: Spectrum 1, FFT Size: 1.024k, Center Frequency (Hz): 2k, and Bandwidth (Hz): 4k.

The configuration dialog for the 'Rational Resampler' block is shown below. It has three tabs: General, Advanced, and Documentation. The 'General' tab is selected. The dialog contains the following fields:

Parameter	Value
ID	rational_resampler_xxx_2
Type	Complex->Complex (Complex Taps)
Interpolation	4000
Decimation	44100
Taps	
Fractional BW	0

At the bottom of the dialog are three buttons: OK, Cancel, and Apply.

Frequency Xlating FIR Filter

Frequency Xlating FIR Filter

- Frequency-translating FIR filter
- Often used for channel selection block
- Performs frequency translation, channel selection and decimation in one step

Frequency Xlating FIR Filter

Frequency Xlating FIR Filter
Decimation: 4
Taps: lpf
Center Frequency: 3
Sample Rate: 4k

Name: Spectrum 2
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 1k

Co
Loop Ba
Order: 2

General | Advanced | Documentation

ID	freq_xlating_fir_filter_xxx_0
Type	Complex->Complex (Complex Taps)
Decimation	4
Taps	lpf
Center Frequency	tune_freq-2000
Sample Rate	samp_rate

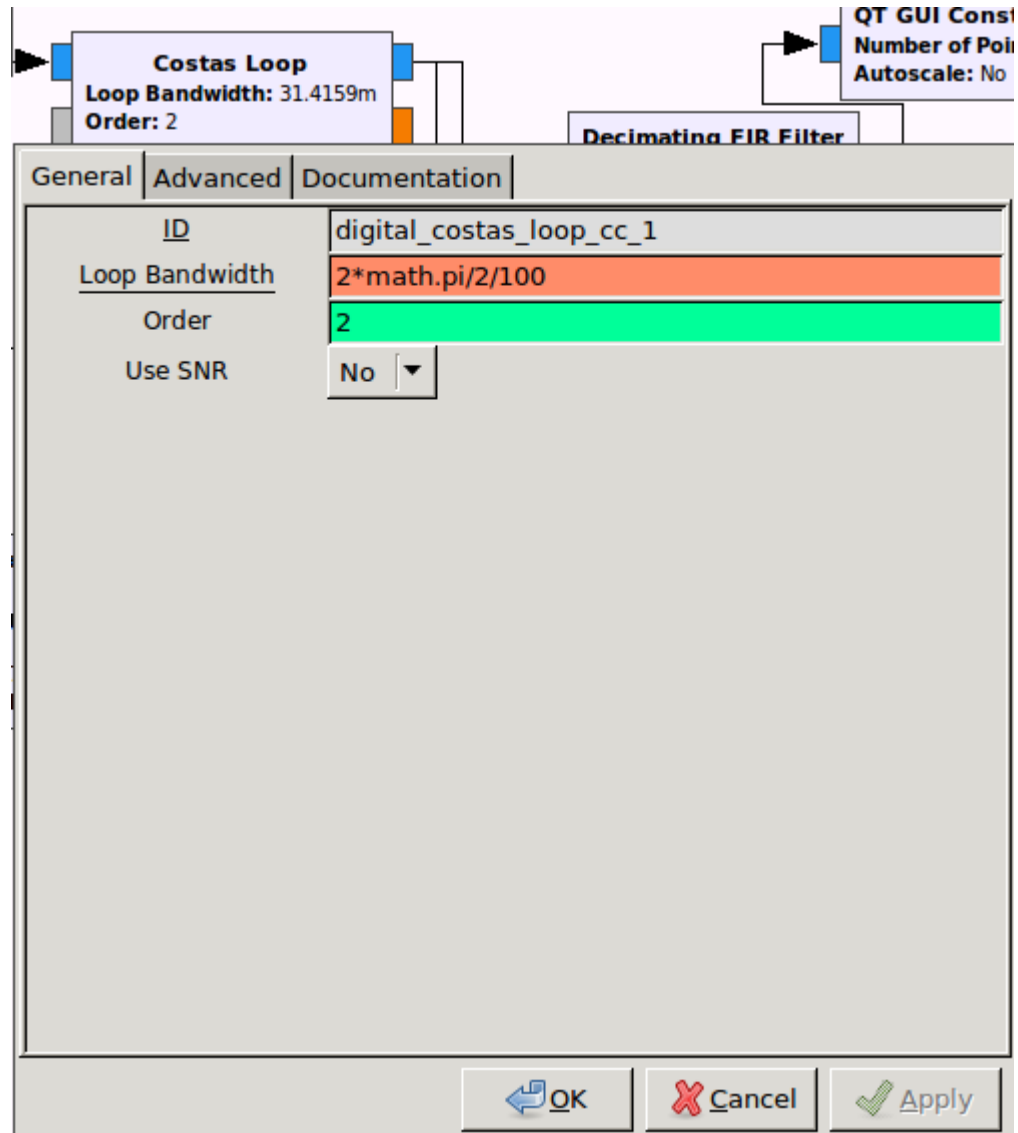
OK Cancel Apply

Costas Loop

Costas Loop

- Locks to the centre frequency of a signal and downconverts it to baseband

Costas Loop

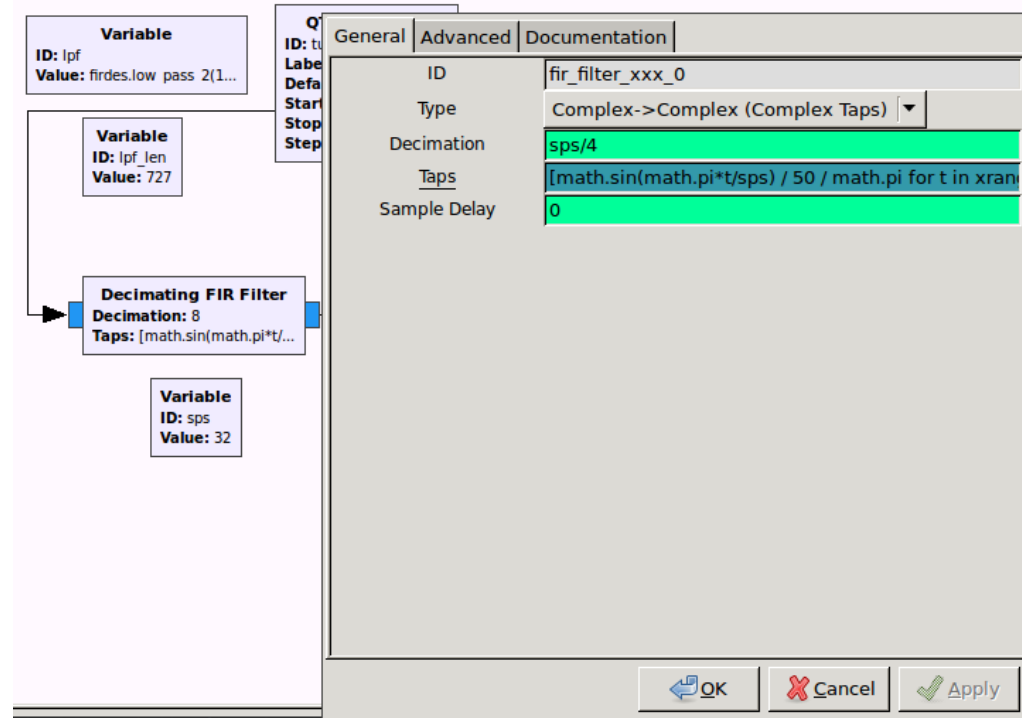
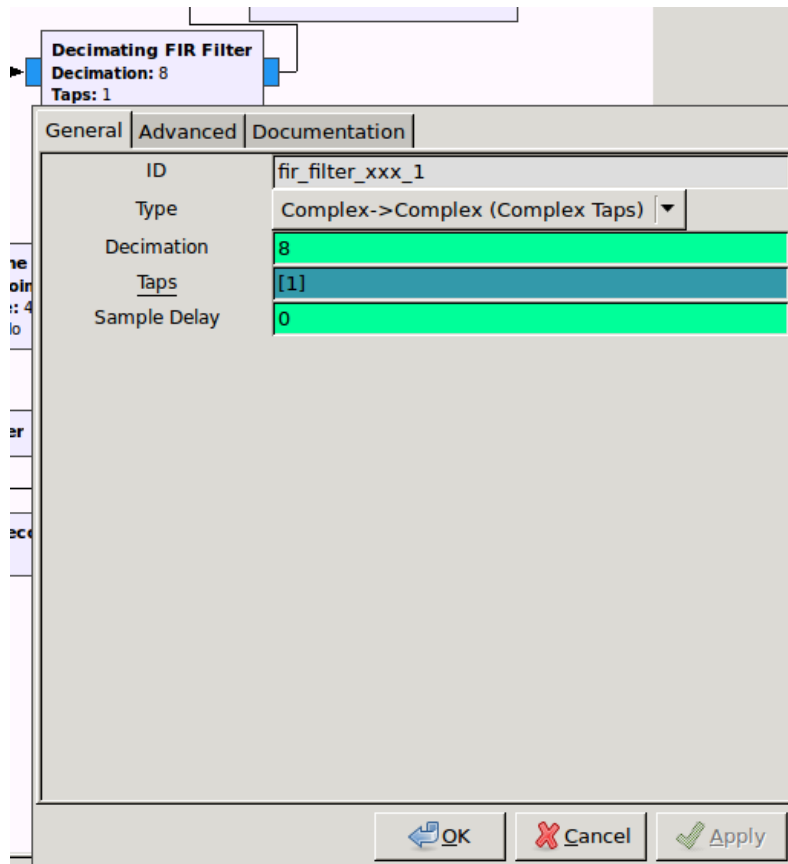


Decimating FIR Filters

Decimating FIR Filters

- Decimation is reducing the output sampling rate by ignoring all but every N th sample

Decimating FIR Filters



Clock Recovery

The screenshot displays a configuration window for a 'Clock Recovery MM' block. The 'Advanced' tab is selected, showing the following parameters:

Parameter	Value
ID	digital_clock_recovery_mm_xx_0
Type	Float
<u>Omega</u>	$4 \cdot (1 + 0.0)$
<u>Gain Omega</u>	$0.25 \cdot 0.175 \cdot 0.175$
<u>Mu</u>	0.5
<u>Gain Mu</u>	0.175
Omega Relative Limit	0.005

At the bottom of the window are buttons for OK, Cancel, and Apply.

Below the configuration window, a block diagram shows the 'Clock Recovery MM' block in a system. The block's current settings are displayed as follows:

- Clock Recovery MM**
- Omega: 4
- Gain Omega: 7.65625m
- Mu: 500m
- Gain Mu: 175m
- Omega Relative Limit: 5m

The block is connected between a 'Complex To Real' block and a 'Binary Slicer' block.

Workshop time!

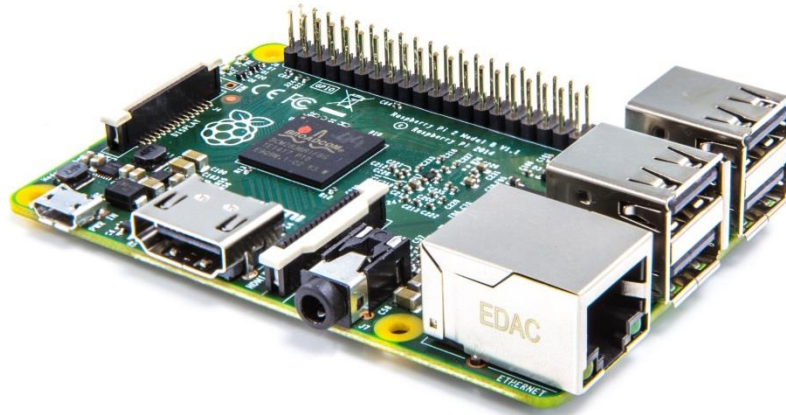
- Now it's your turn to follow steps 1, 2 & 3 here:
<https://github.com/0xsh/Workshops/tree/master/BSides-Canberra-2017>
- Please collect a live USB Stick
 - Boot “Persistence Kali” menu option, this has GNURadio installed for you



Porting to a Raspberry Pi

Raspberry Pi – Data Exfil POC

- Raspberry Pi + USB Microphone + 4G Dongle



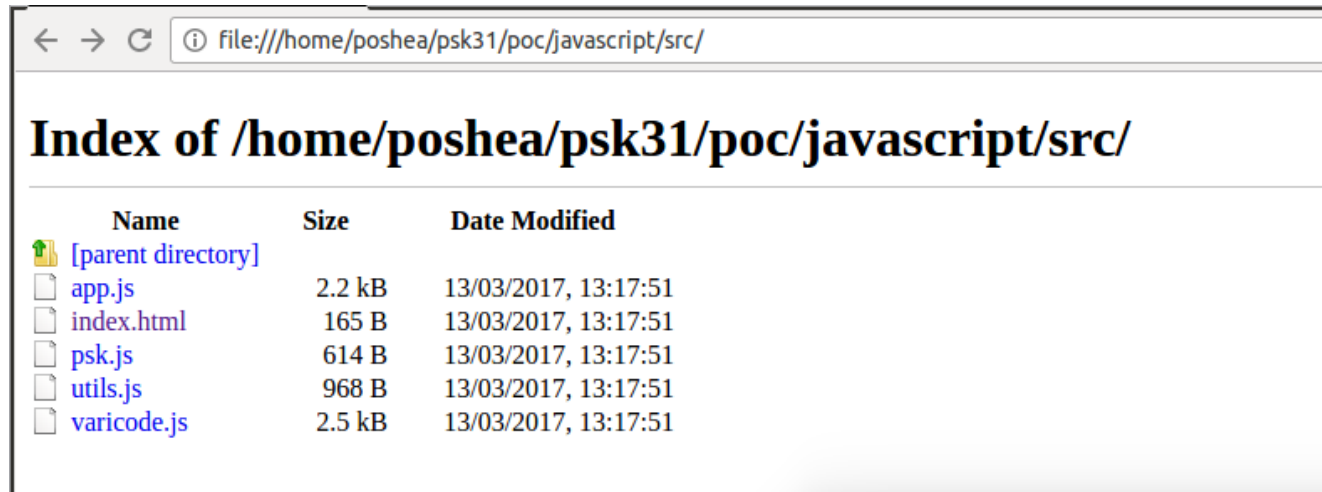
Raspberry Pi – Data Exfil POC

- Tutorial with GNU Radio:
<http://randomkeystrokes.com/2017/03/13/data-exfiltration-via-psk31-with-gnu-radio/>
- Tutorial without GNU Radio:
<http://randomkeystrokes.com/2017/03/13/data-exfiltration-via-psk31-without-gnu-radio/>

Raspberry Pi – Data Exfil POC







- Try out the JavaScript for transmitting PSK31 here:
<https://github.com/0xsh/Workshops/tree/master/BSides-Canberra-2017/JavaScript>
- You can keep the same GNU Radio receive graph to test it out on your laptop – but this also works on a Raspberry Pi with a USB microphone
- Can also use tools like Fldigi instead of GNU Radio for receiving or even more JavaScript!

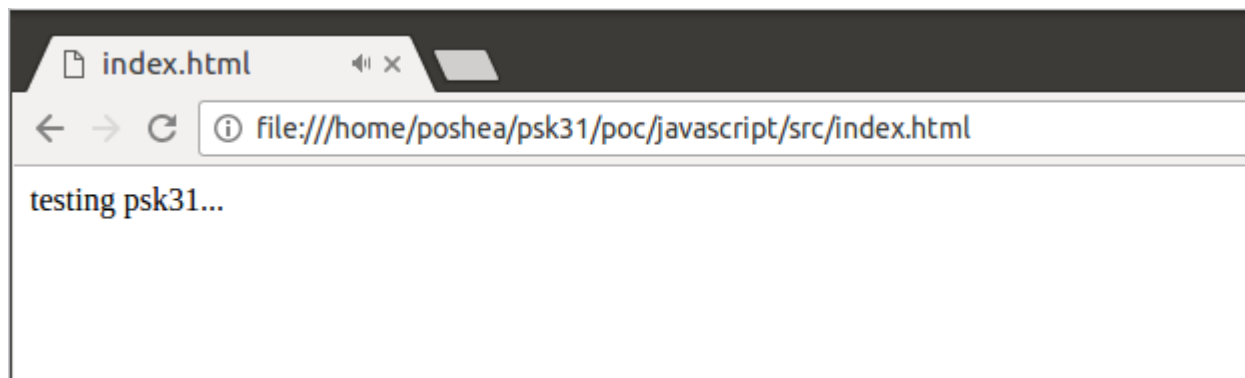
JavaScript for transmitting



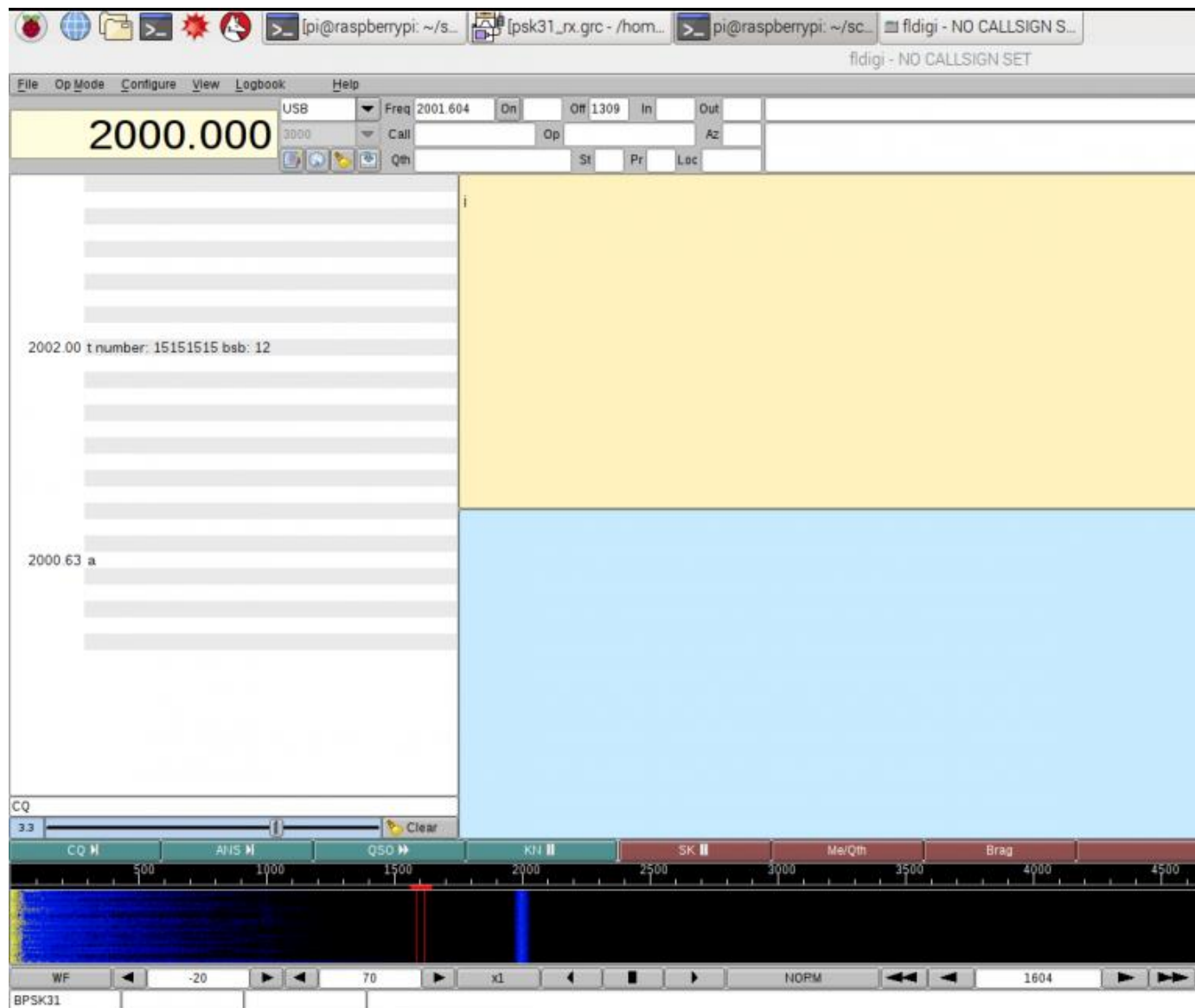
file:///home/poshea/psk31/poc/javascript/src/

Index of /home/poshea/psk31/poc/javascript/src/

Name	Size	Date Modified
 [parent directory]		
 app.js	2.2 kB	13/03/2017, 13:17:51
 index.html	165 B	13/03/2017, 13:17:51
 psk.js	614 B	13/03/2017, 13:17:51
 utils.js	968 B	13/03/2017, 13:17:51
 varicode.js	2.5 kB	13/03/2017, 13:17:51



Raspberry Pi – Receiving with Fldigi



Other work you can play with...

- **GoodPSK** “is a tool for generating PSK31 audio recordings, sometimes with strange or clever attributes” & check out the lectures!: <https://github.com/travisgoodspeed/goodpsk>
- <https://sdradventure.wordpress.com/2011/10/15/gnuradio-psk31-decoder-part-1>
- <https://sdradventure.wordpress.com/2011/10/15/gnuradio-psk31-decoder-part-2/>
- <https://github.com/JasonBens/PSK31-transceiver>
- <https://github.com/tkuester/gr-psk31>
- <https://github.com/christophL/gr-digimodes>
- <https://github.com/argilo/sdr-examples>

References

- <http://aintel.bi.ehu.es/psk31theory.html>
- <http://www.arrl.org/digital-data-modes>
- [https://en.wikipedia.org/wiki/Phase-shift keying](https://en.wikipedia.org/wiki/Phase-shift_keying)
- <https://en.wikipedia.org/wiki/PSK31>
- <https://en.wikipedia.org/wiki/Baud>
- <http://people.scs.carleton.ca/~barbeau/SDRCRBook/Content/chapter09.pdf>
- <https://github.com/travisgoodspeed/goodpsk>
- <https://github.com/tkuester/gr-psk31>
- <http://www.w1hkj.com/FldigiHelp/index.html>

Join us at Cyberspectrum Melbourne

Twitter

@sdr_melbourne

Email

sdr.melbourne@gmail.com

Slack

sdr-australia.slack.com (email for an invite)

Meetup

<https://www.meetup.com/Cyberspectrum-Melbourne>

Blog

<http://randomkeystrokes.com/category/sdr/>

Github

<https://github.com/sdr-melbourne>

YouTube

<https://www.youtube.com/channel/UCLBloqxOXEj4fH7s>
ULA (SDR Melbourne Channel)



Please return the USB sticks...
or kitteh haz cry...

