Wireless Course: Things That Aren't Wi-Fi

These slides have been updated and condensed from the original course material

Things that aren’t Wi-Fi

That’s no moon!

Running the gamut

There’s a ton of things out there that aren’t Wi-Fi

Kismet, NetStumbler, etc, will never be able to see them

Doesn’t mean they’re not a weakness!

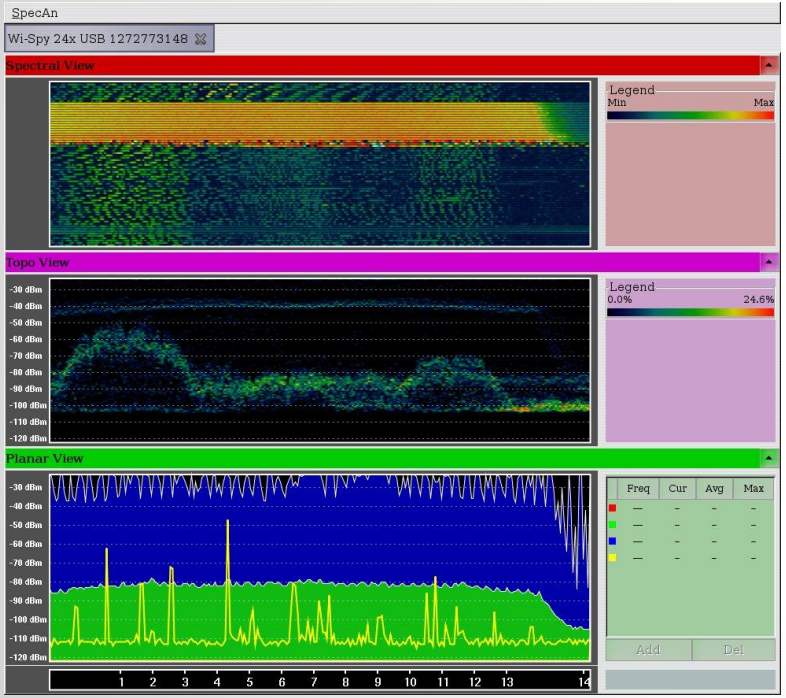
Spectrum Analyzers

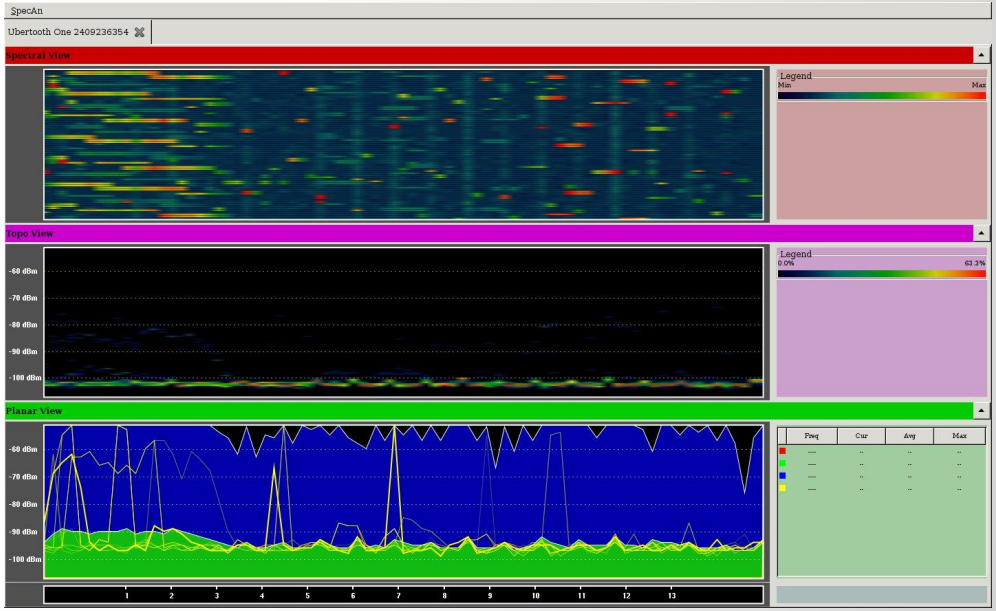
- Basic SA reports signal level at regular intervals
- Cheaper the SA the lower the resolution and the slower the speed
- Can help find non-Wi-Fi interference

Affordable SAs

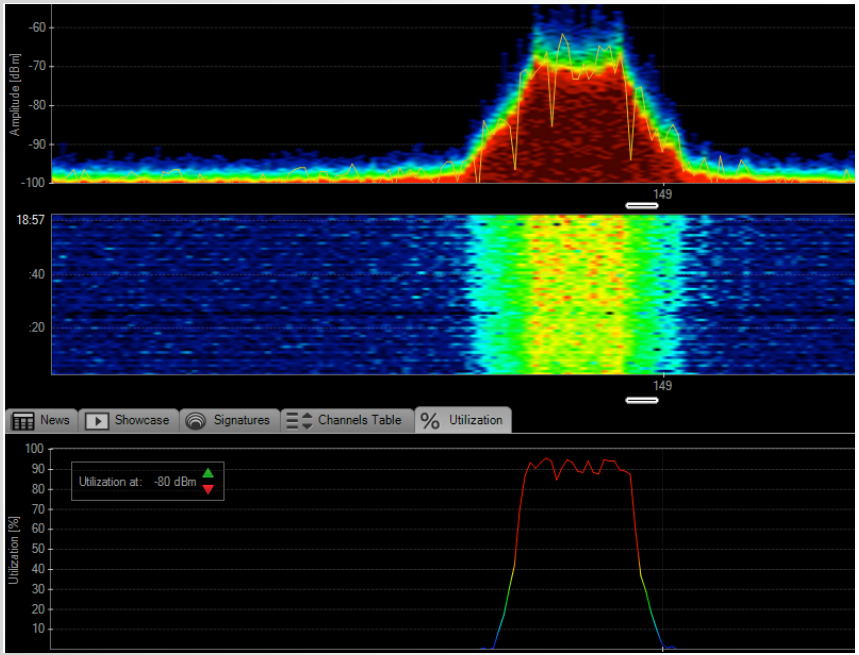
- Metageek Wi-Spy and Chanalyzer software
- Ubertooth One can double as a SA
- Ubiquity

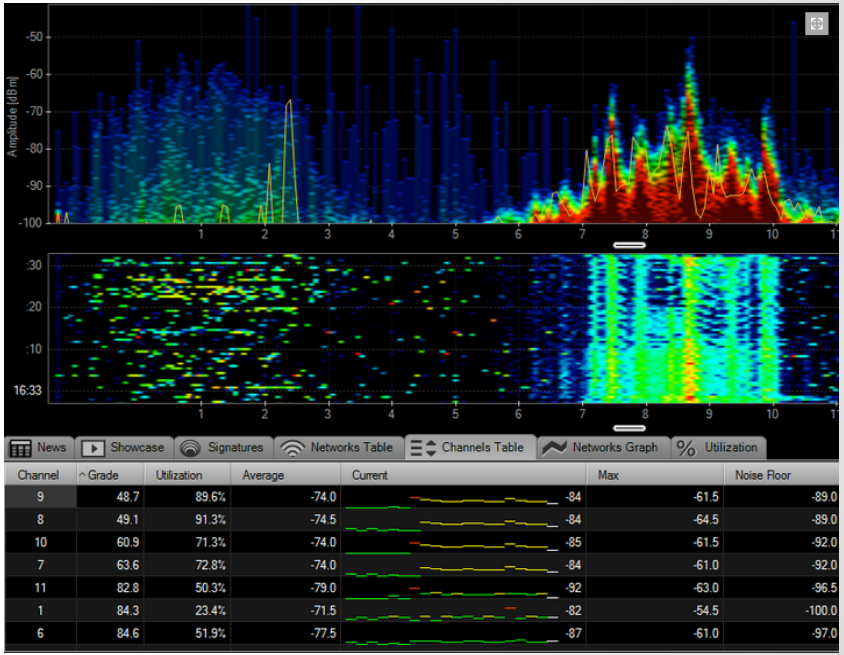
Spectools


SpecAn
Wi-Spy 24x USB 1272773148
Spectral View
Legend
Min
Max
Topo View
Legend
0.0%
24.6%
-30 dBm
-40 dBm
-50 dBm
-60 dBm
-70 dBm
-80 dBm
-90 dBm
-100 dBm
-110 dBm
-120 dBm
Planar View
Freq
Cur
Avg
Max
Add
Del
1 2 3 4 5 6 7 8 9 10 11 12 13 14


SpecAn
Ubertooth One 2409236354
Spectral View
Legend
Min
Max
Topo View
Legend
0.0%
63.3%
-60 dBm
-70 dBm
-80 dBm
-90 dBm
-100 dBm
Planar View
Freq
Cur
Avg
Max
1 2 3 4 5 6 7 8 9 10 11 12 13

Chanalyzer


Amplitude [dBm]
-60
-70
-80
-90
-100
149
18:57
:40
:20
149
News
Showcase
Signatures
Channels Table
Utilization
Utilization at: -80 dBm
Utilization [%]
100
90
80
70
60
50
40
30
20
10


Amplitude [dBm]
-50
-60
-70
-80
-90
-100
1
2
3
4
5
6
7
8
9
10
:30
:20
:10
16:33
News
Showcase
Signatures
Networks Table
Channels Table
Networks Graph
Utilization
Channel
Grade
Utilization
Average
Current
Max
Noise Floor
9 48.7 89.6% -74.0 -84 -61.5 -89.0
8 49.1 91.3% -74.5 -84 -64.5 -89.0
10 60.9 71.3% -74.0 -85 -61.5 -92.0
7 63.6 72.8% -74.0 -84 -61.0 -92.0
11 82.8 50.3% -79.0 -92 -63.0 -96.5
1 84.3 23.4% -71.5 -82 -54.5 -100.0
6 84.6 51.9% -77.5 -87 -61.0 -97.0

Use cases

- SAs excel at finding consistent interference
- Analog video cameras, baby monitors, microwaves are all common
- Generally not (at least, not the affordable ones) so good at finding short-duration interference

Why Can’t My NIC Do This?

- The NIC is meant to decode Wi-Fi
- In theory the radio is capable of doing it, but it would increase the cost
- Some Wi-Fi cards ***can*** do basic spec-an functions
- Generally high-end ones for APs, no good API documentation

Software Defined Radios

- Radio front end
- No dedicated IC backend for decoding radio signal
- Digitize signal and pass it all to the host system
- In theory, if you can tune it, you can be that type of radio

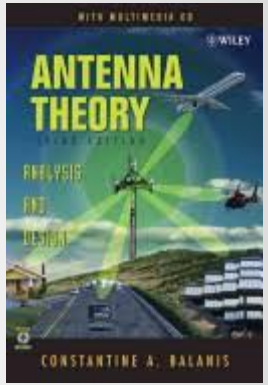
Hardware

- Previously hideously expensive
- Commodity radio hardware like Wi-Fi cards can’t do it because of the cost of digitizers
- Used to be $2000+ to get involved

Enter The Cheapions

- RTL-SDR ; It turns out you can kluge raw signal data out of a DVB tuner for .eu bands
- HackRF ; Mike Ossmann does it again with a killer TX capable SDR board that’s hugely affordable
- BladeRF ; Another very affordable solution

Antennas, aka Why-phy



- Get some books, become a HAM
- Know your connector types

  SMA / R-SMA / N / BNC / etc
- Know your cable types and constraints
- Receive only, very forgiving on antenna types
- Transmit with the wrong antenna/cable…

  Sad Panda

SDR vs Spectrum Analyzers

- SA reports signal level at a given frequency
- SDR reports encoded signal data
- SDR can act like a spectrum analyzer
- Spectrum analyzer can’t act like a SDR

SDR vs Monitor Mode

Monitor mode turns off the filtering at the MAC layer

Sends a stream of packets to the computer

SDR is different - it sends raw signal data

SDR Capture Data

No packets - just raw data

Raw radio samples of some bandwidth per sample

Bandwidth defines amount of spectrum covered by samples

The Problems With SDR

- Until very recently, almost exclusively the domain of academics

- Academics are great, but they don’t usually design for reproducibility / edge cases / etc

- Mike Ossmann’s explicit goal with HackRF is to bring signal analysis to the hackers

- This is very good news

More SDR Considerations

- Almost always the most expensive possible route - but that doesn’t mean don’t do it
- Shifts the cost from designing a chip to supplying the compute power in the backend
- Takes a LOT of bandwidth

On The Brighter Side

- One device can talk nearly infinite protocols
- Able to investigate protocols for which there are no public specs or chips
- Computing power is always growing making the compute cost less of an issue

SDR Specs

- Bit depth of samples (usually 8 or 16 bit) determines fidelity, much like 16 bit color
- Sample width, such as 200KHz or 20MHz, defines how much spectrum can be captured at a time
- Frequency range, such as 30MHz to 4GHz, defines the range the radio can be tuned to

SDR Software

- Multiple tools
- GQRX, SDR# for browsing spectrum
- GNU Radio is the grand-daddy of decoding platforms

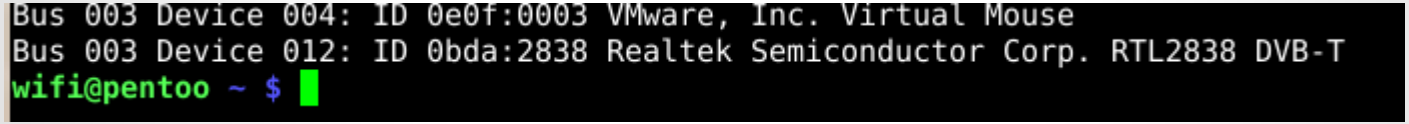
Using a RTLSDR

- Controlled by the rtlsdr software
- Also uses gr-osmosdr (bridge to GNU Radio)
- … and GNU Radio
- Fortunately, Pentoo came with all this installed

Plug it in!

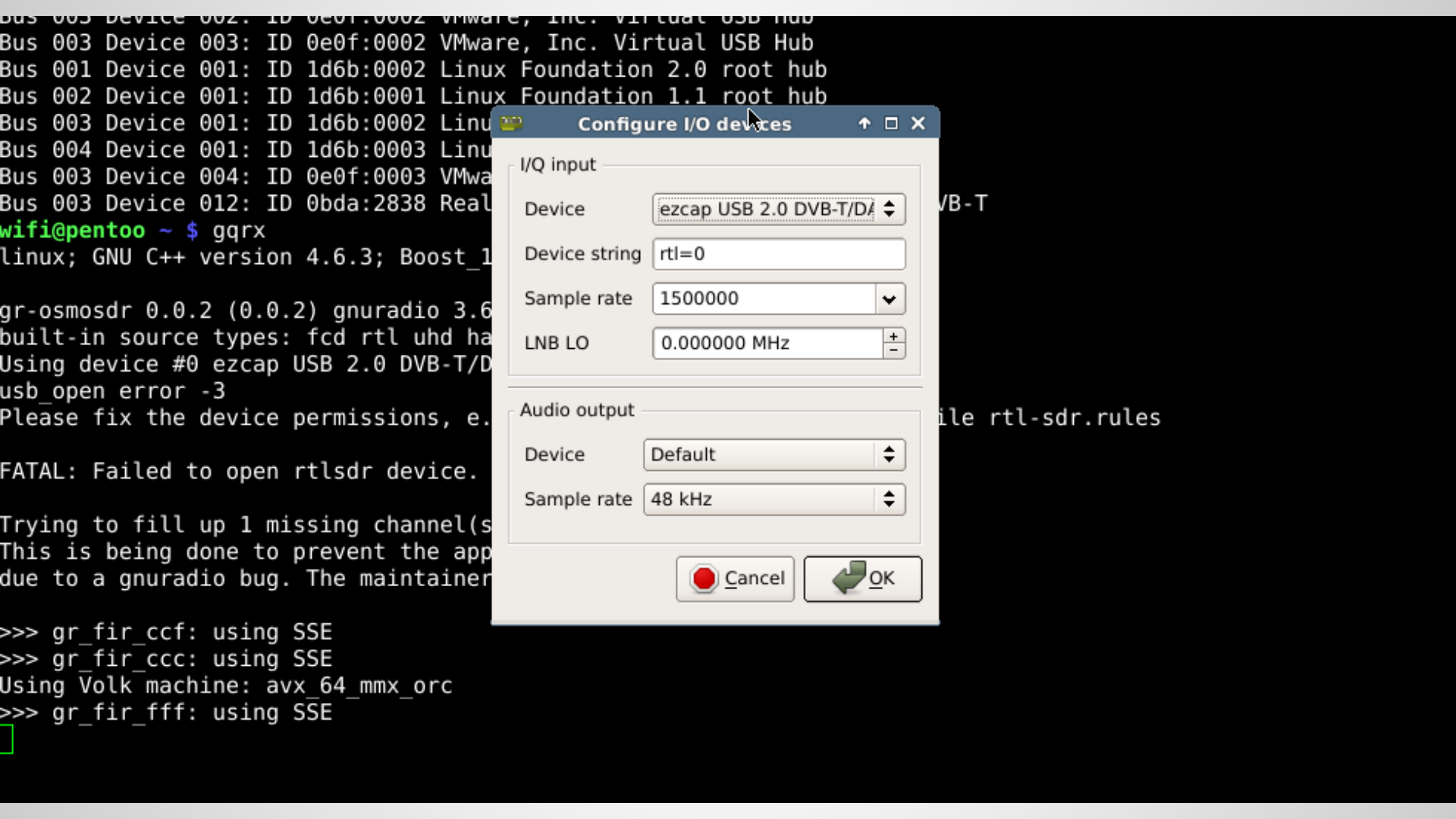
- Make sure it’s being recognized…

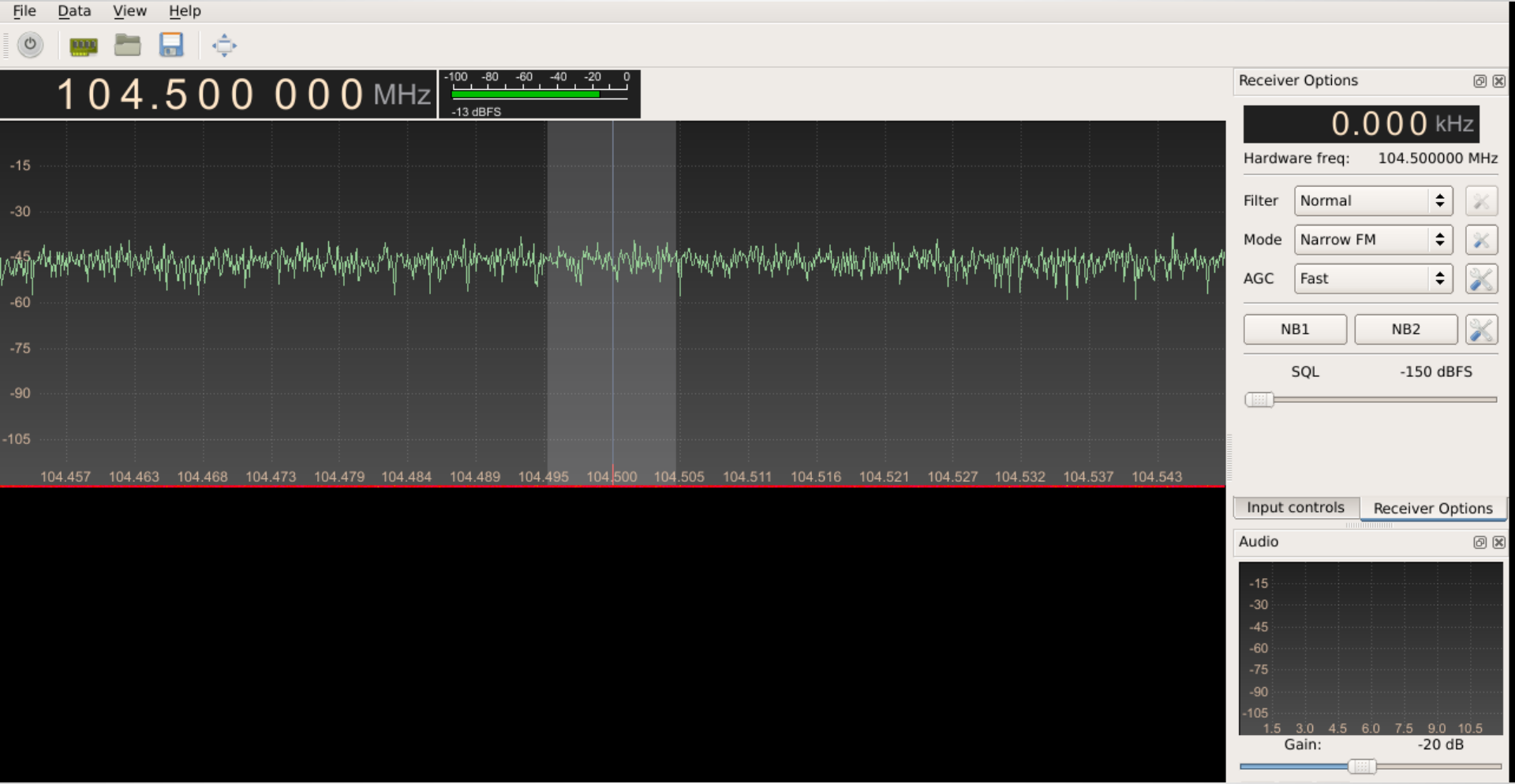
```
$ lsusb
```



Lets Look At Spectrum

- ‘gqrx’ is a great tool for viewing and decoding spectrum
- Lets fire it up, and tune to somewhere that should have lots of obvious signals…
- FM radio! Tune to ~100MHz


Bus 003 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 001: ID 1d6b:0002 Linu
Bus 004 Device 001: ID 1d6b:0003 Linu
Bus 003 Device 004: ID 0e0f:0003 VMwa
Bus 003 Device 012: ID 0bda:2838 Real
VB-T
wifi@pentoo ~ $ gqrx
linux; GNU C++ version 4.6.3; Boost_1
gr-osmosdr 0.0.2 (0.0.2) gnuradio 3.6
built-in source types: fcd rtl uhd ha
Using device #0 ezcap USB 2.0 DVB-T/D
usb_open error -3
Please fix the device permissions, e.
ile rtl-sdr.rules
FATAL: Failed to open rtlsdr device.
Trying to fill up 1 missing channel(s
This is being done to prevent the app
due to a gnuradio bug. The maintainer
>>> gr_fir_ccf: using SSE
>>> gr_fir_ccc: using SSE
Using Volk machine: avx_64_mmx_orc
>>> gr_fir_fff: using SSE
Configure I/O devices
I/Q input
Device
ezcap USB 2.0 DVB-T/D
Device string
rtl=0
Sample rate
1500000
LNB LO
0.000000 MHz
Audio output
Device
Default
Sample rate
48 kHz
Cancel
OK


File
Data
View
Help
104.500 000 MHz
-100 -80 -60 -40 -20 0
-13 dBFS
-15
-30
-45
-60
-75
-90
-105
104.457 104.463 104.468 104.473 104.479 104.484 104.489 104.495 104.500 104.505 104.511 104.516 104.521 104.527 104.532 104.537 104.543
Receiver Options
0.000 kHz
Hardware freq: 104.500000 MHz
Filter Normal
Mode Narrow FM
AGC Fast
NB1
NB2
SQL -150 dBFS
Input controls
Receiver Options
Audio
-15
-30
-45
-60
-75
-90
-105
1.5 3.0 4.5 6.0 7.5 9.0 10.5
Gain: -20 dB

Well That's Weird

- So if something goes wrong, GNU Radio tools fill with random gaussian noise
- Because somehow totally bogus data is… … better?
- Lets see what happened...

Ah.

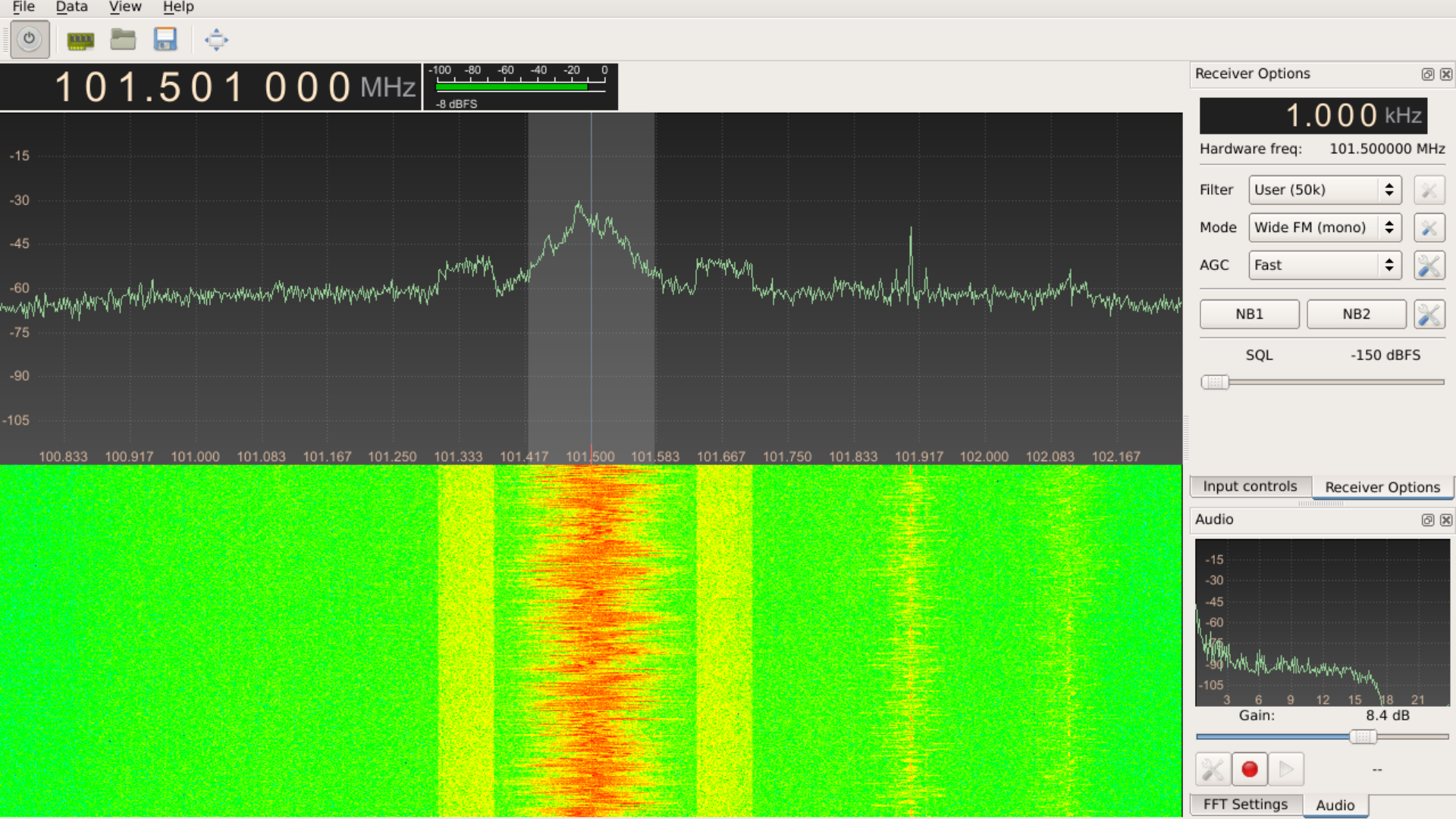
```
>>> gr_fir_ccf: using SSE
>>> gr_fir_ccc: using SSE
Using Volk machine: avx_64_mmx_orc
>>> gr_fir_fff: using SSE
gr-osmosdr 0.0.2 (0.0.2) gnuradio 3.6.5.1
built-in source types: fcd rtl uhd hackrf
Using device #0 ezcap USB 2.0 DVB-T/DAB/FM dongle
usb_open error -3
Please fix the device permissions, e.g. by installing the udev rules file rtl-sdr.rules

FATAL: Failed to open rtlsdr device.

Trying to fill up 1 missing channel(s) with gaussian noise.
This is being done to prevent the application from crashing
due to a gnuradio bug. The maintainers have been informed.
```

So...

- If permission error, check udev rules
- If device error, make sure you picked rtlsdr and not another type of device
- If compiling it yourself, make sure you included gr-osmosdr, etc


File
Data
View
Help
101.501 000 MHz
-100 -80 -60 -40 -20 0
-8 dBFS
-15
-30
-45
-60
-75
-90
-105
100.833 100.917 101.000 101.083 101.167 101.250 101.333 101.417 101.500 101.583 101.667 101.750 101.833 101.917 102.000 102.083 102.167
Receiver Options
1.000 kHz
Hardware freq: 101.500000 MHz
Filter User (50k)
Mode Wide FM (mono)
AGC Fast
NB1
NB2
SQL -150 dBFS
Input controls
Receiver Options
Audio
-15
-30
-45
-60
-90
-105
3 6 9 12 15 18 21
Gain: 8.4 dB
--
FFT Settings
Audio

Cool!

- Now set the mode on the right to “Wide FM” (mono or stereo)
- Move your slider to the middle of a channel
- You should be hearing some radio, I hope!

What Else Can We Look At?

- ISM bands are interesting
- Technically some bands are illegal to look at
- Consult your local laws (fcc.gov)
- In general, ISM bands are legal to look at
- Cell phones and pagers are not
- Lets assume looking at spectrum maps is “ok”
- When it doubt: Don’t look, and do research!

What Am I Hearing/Seeing?

http://www.rtl-sdr.com/signal-identification-guide/

Lets check some of these out…

Of Note:

ACARS, P25 Phase 1, MotoTurbo, POCSAG/FLEX, APCO P-25, NOAA Weather Satellite, Numbers Stations, and more

Do some cruising

- Look around at some frequencies
- 451MHz or so is interesting (often radio repeaters)
- 548MHz should be ATSC TV
- Around 915mhz can be interesting devices

That Weird Spike

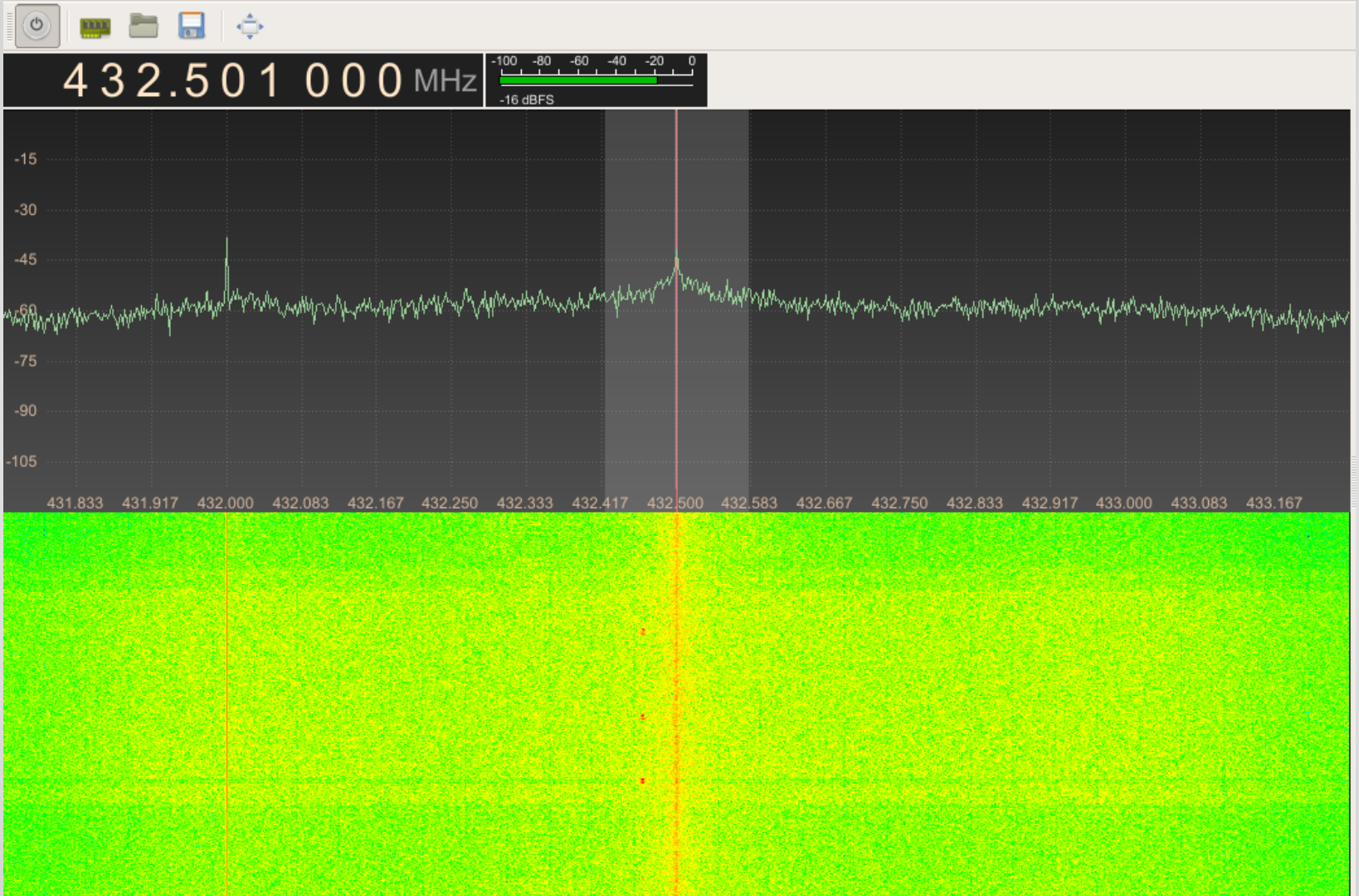
- Notice that weird spike always in the center?
- It’s called DC offset
- The two components of the signal (I and Q) are just slightly out of phase
- This is very common with many radios because it’s very expensive to solve in hw

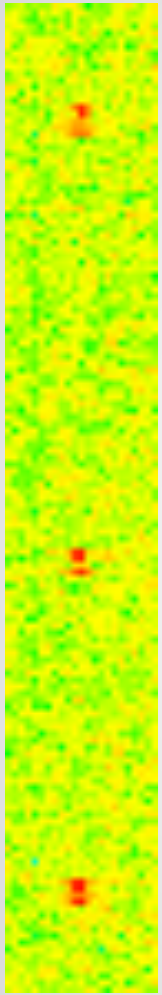
DC Offset

- The solution is basically “don’t tune directly to where you’re interested in”
- Tune slightly to the side and then use software

Lets Look At Something Real

- What do we all have?
- Car keyfobs!
- As numbers they’re not too interesting, as something to explore, they’re cool
- I happen to know Subaru is OOK at 432MHz


432.501 000 MHz
-100 -80 -60 -40 -20 0
-16 dBFS
-15
-30
-45
-60
-75
-90
-105
431.833 431.917 432.000 432.083 432.167 432.250 432.333 432.417 432.500 432.583 432.667 432.750 432.833 432.917 433.000 433.083 433.167



- It was hard to see before
- But see those little bursts?
- That’s our data...

Recording

- RTLSDR comes with recording tools

```
$ rtl_sdr /tmp/capture.bin -s 1.8e6 -f 432.5e6
```

- This sets the bandwidth to 1.8e6, or 1.8MHz
- And the tuning frequency to 432.5e6, or 432.5 MHz
- Yay, scientific!

```
wifi@pentoo ~ $ rtl_sdr /tmp/capture.bin -s 1.8e6 -f 432.5e6
Found 1 device(s):
  0:  Realtek, RTL2838UHIDIR, SN:

Using device 0: ezcap USB 2.0 DVB-T/DAB/FM dongle
Found Elonics E4000 tuner
Tuned to 432500000 Hz.
Reading samples in async mode...
^CSignal caught, exiting!

User cancel, exiting...
```

- Remember we want to tune slightly off center
- Let it run for a little while
- Then cancel with control-c

Looking At Files

- The best right now is Baudline
- Unfortunately it’s not OSS
- And it’s a weird license
- So we have to download it manually..

baudline signal analyzer - FFT s...

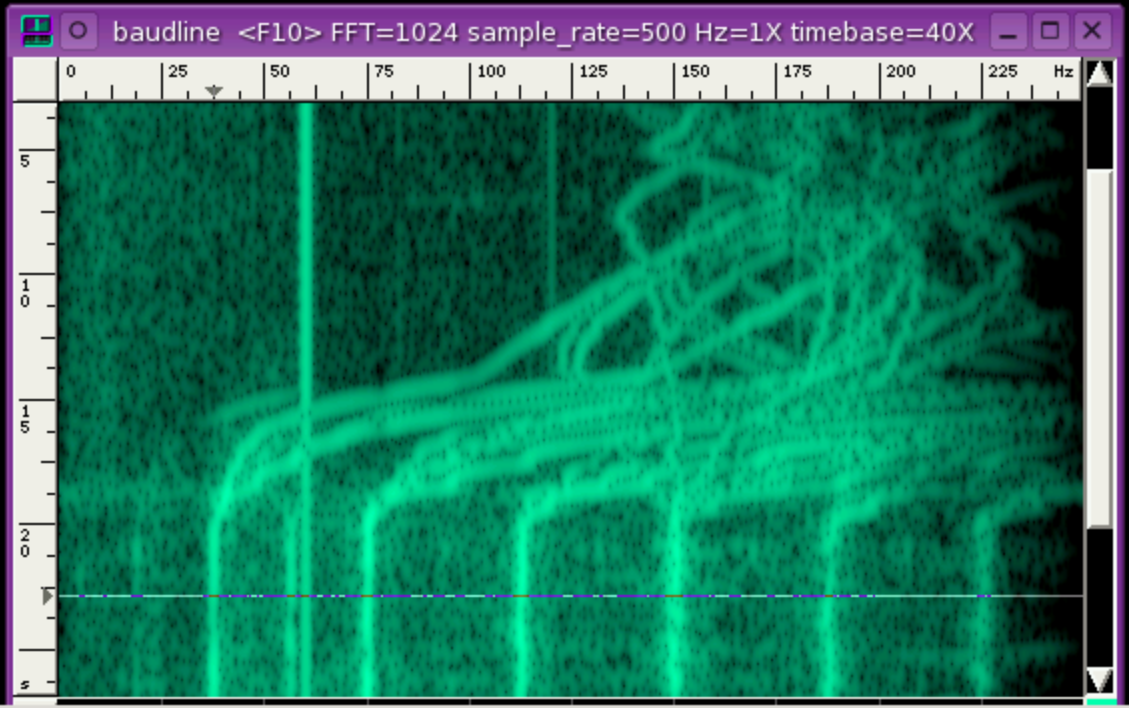
www.baudline.com

Google

baudline

SigBlips DSP engineering
Contract design and development.
Hire us!

- Home
- News
- What is baudline?
- Screenshots
- Download
- FAQ
- Manual
- Search
- Solutions
- Mystery Signal



Quirks

- Baudline doesn’t like large files
- Larger than ~50MB
- Your capture is hopefully less than this
- If not:

```
$ dd if=/tmp/capture.bin of=/tmp/trim.bin
bs=1M count=50
```

- Or use ‘split’

Uncompress & Run Baudline

```
$ cd ~/
$ tar xvf Downloads/baudline…(platform name)
$ cd baudline…
$ ./baudline
```

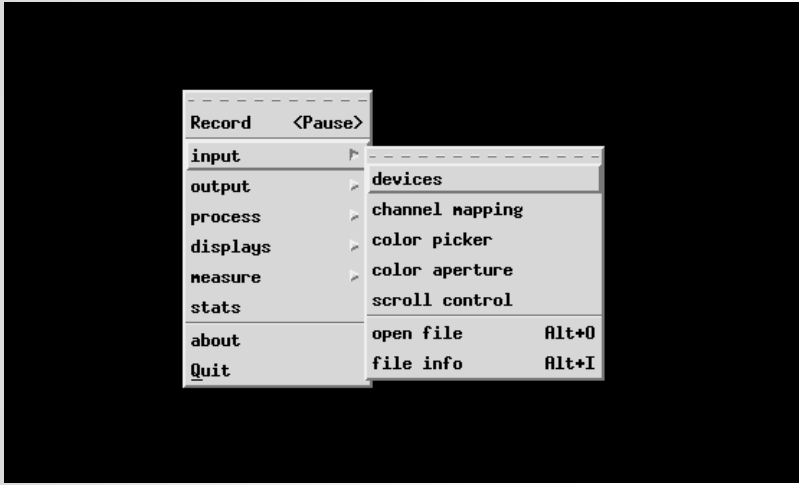

baudline <F1> FFT=2048 sample_rate=44100 Hz=1X timebase=1X
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
13000
14000
15000
16000
17000
18000
19000
20000
21000 Hz
4000
3500
3000
2500
2000
1500
1000
500
ms
all input devices disabled
0
-20
-40
-60
-80
-100
dB
7364.36 Hz
2525 ms

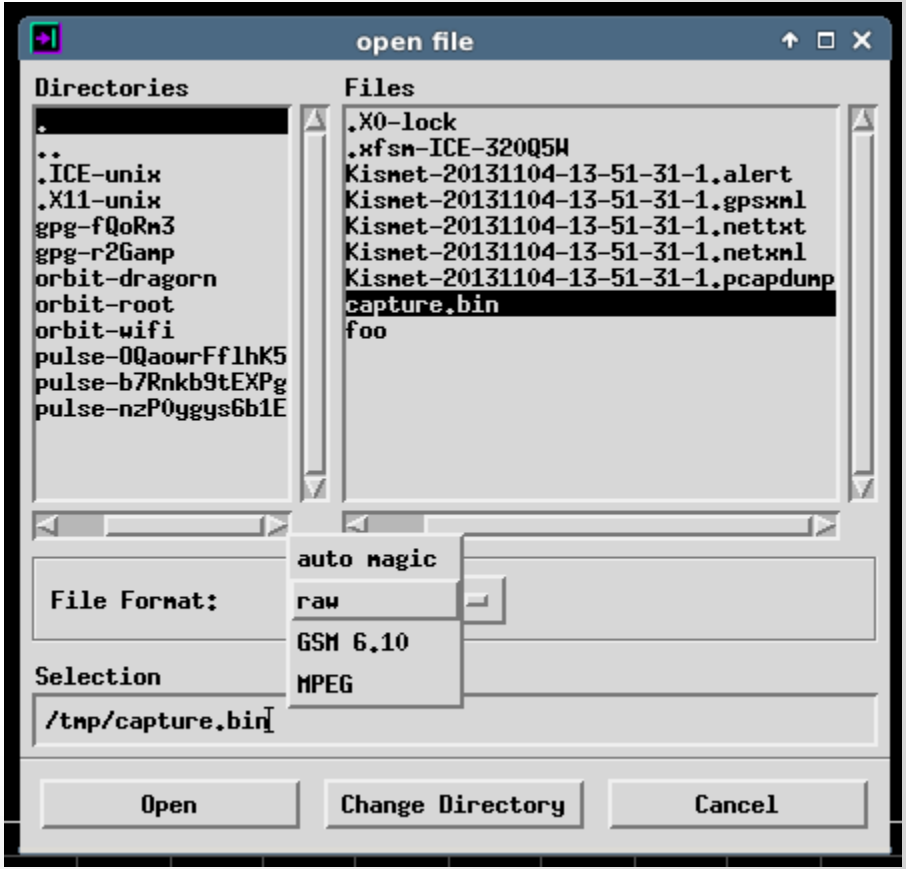
Weird UI

- Yeah, the UI is weird
- It’s the old tkinter UI
- Sorry.

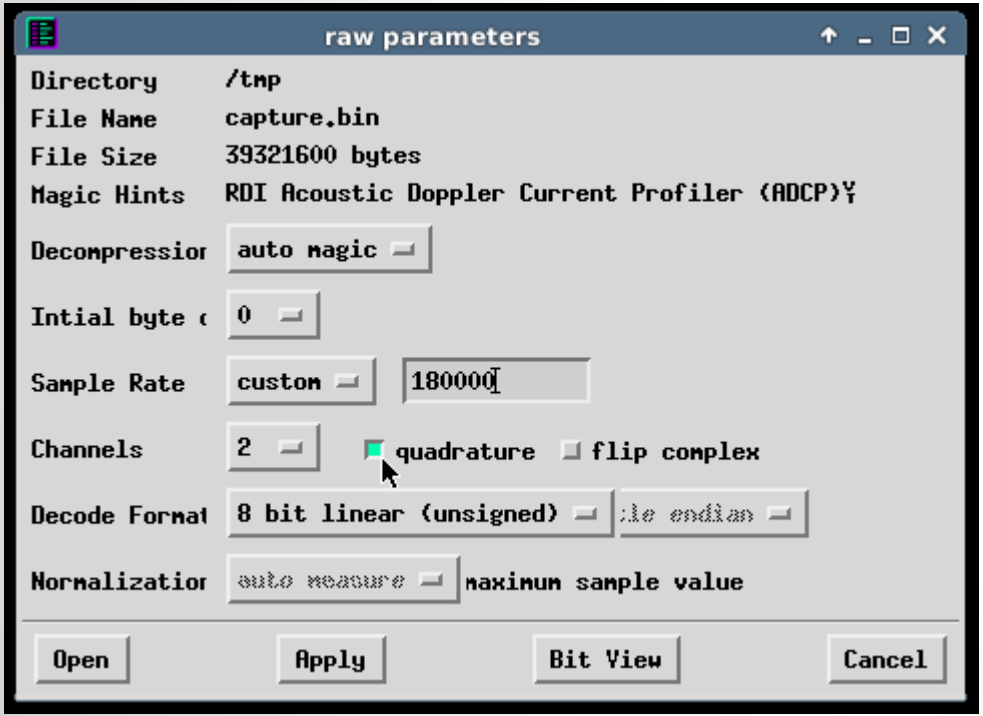
- Right click for menus

Special File

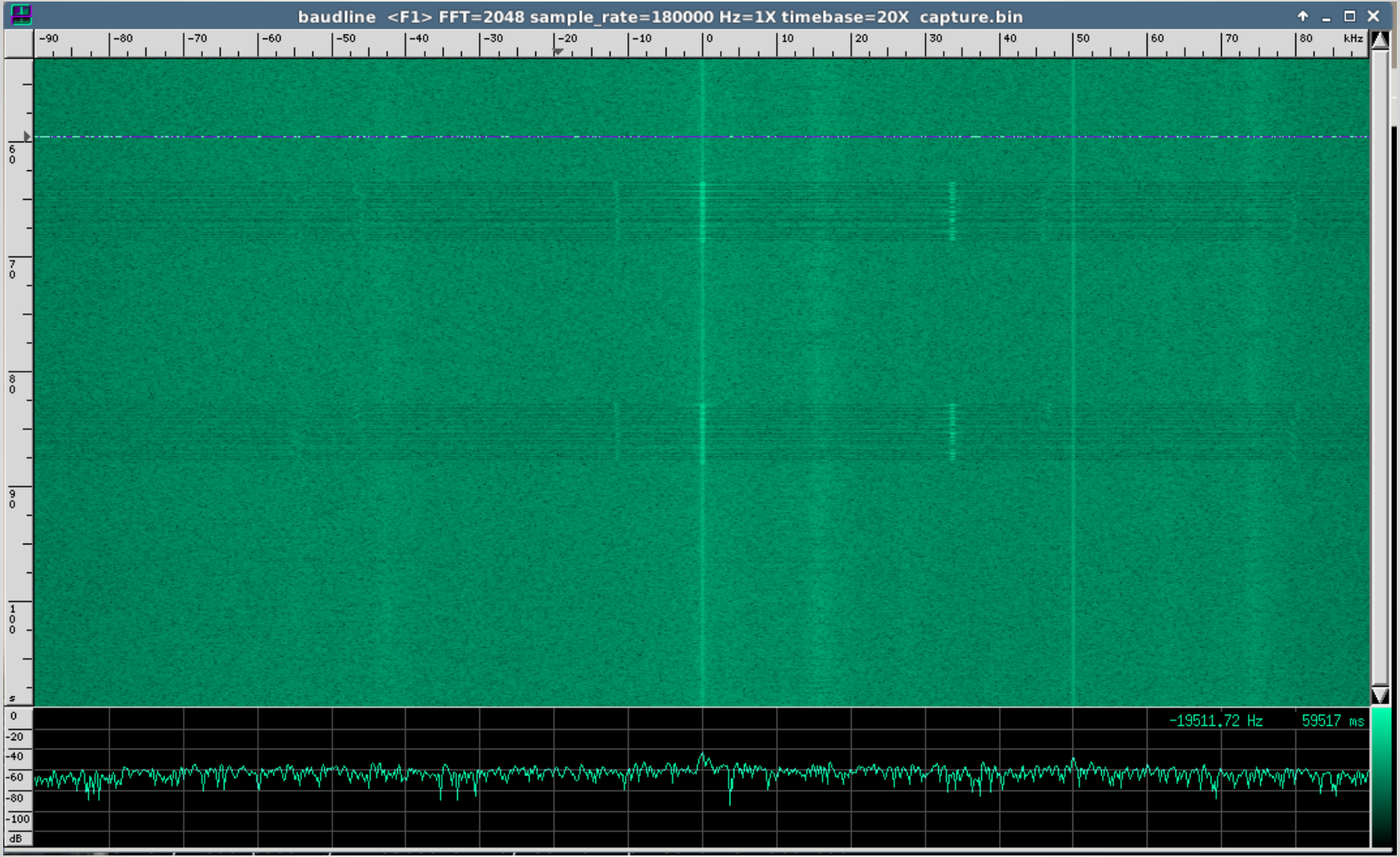


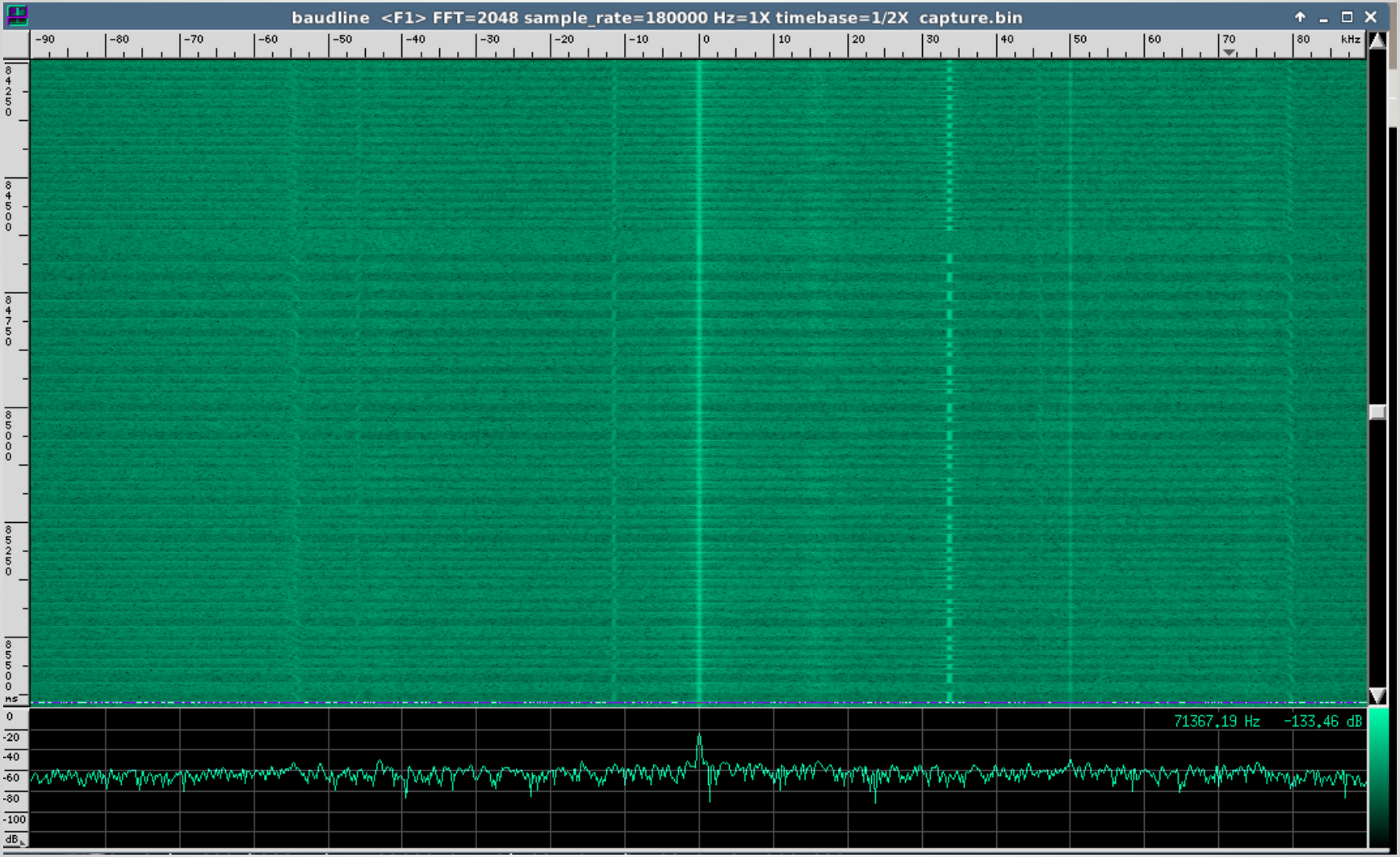


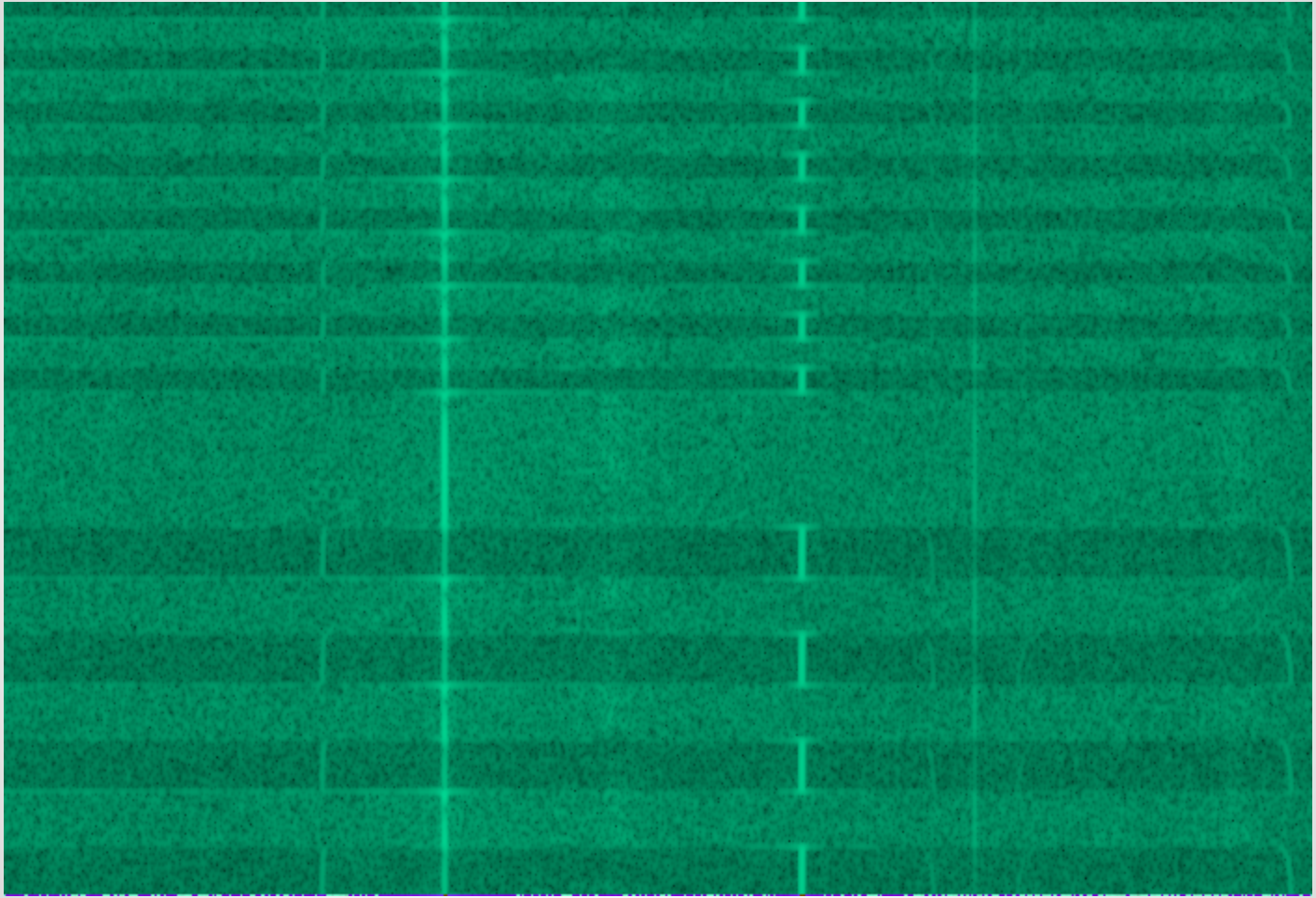
File Options



- Custom sample rate - remember, 1.8MHz
- 2 Channels - I and Q
- Quadrature (Q)
- 8bit unsigned data


baudline <F1> FFT=2048 sample_rate=180000 Hz=1X timebase=20X capture.bin
-90 -80 -70 -60 -50 -40 -30 -20 -10 0 10 20 30 40 50 60 70 80 kHz
60
70
80
90
100
s
0 -20 -40 -60 -80 -100 dB
-19511.72 Hz
59517 ms


baudline <F1> FFT=2048 sample_rate=180000 Hz=1X timebase=1/2X capture.bin
-90 -80 -70 -60 -50 -40 -30 -20 -10 0 10 20 30 40 50 60 70 80 kHz
ms
0 -20 -40 -60 -80 -100 dB
71367.19 Hz -133.46 dB



What Are We Seeing?

- ASK / OOK data - Amplitude Shift Keying, aka On-Off Keying
- Transmitter is on, or transmitter is off
- IE we've increased the amplitude, or power
- Shows up as series of dashes

Looking deeper

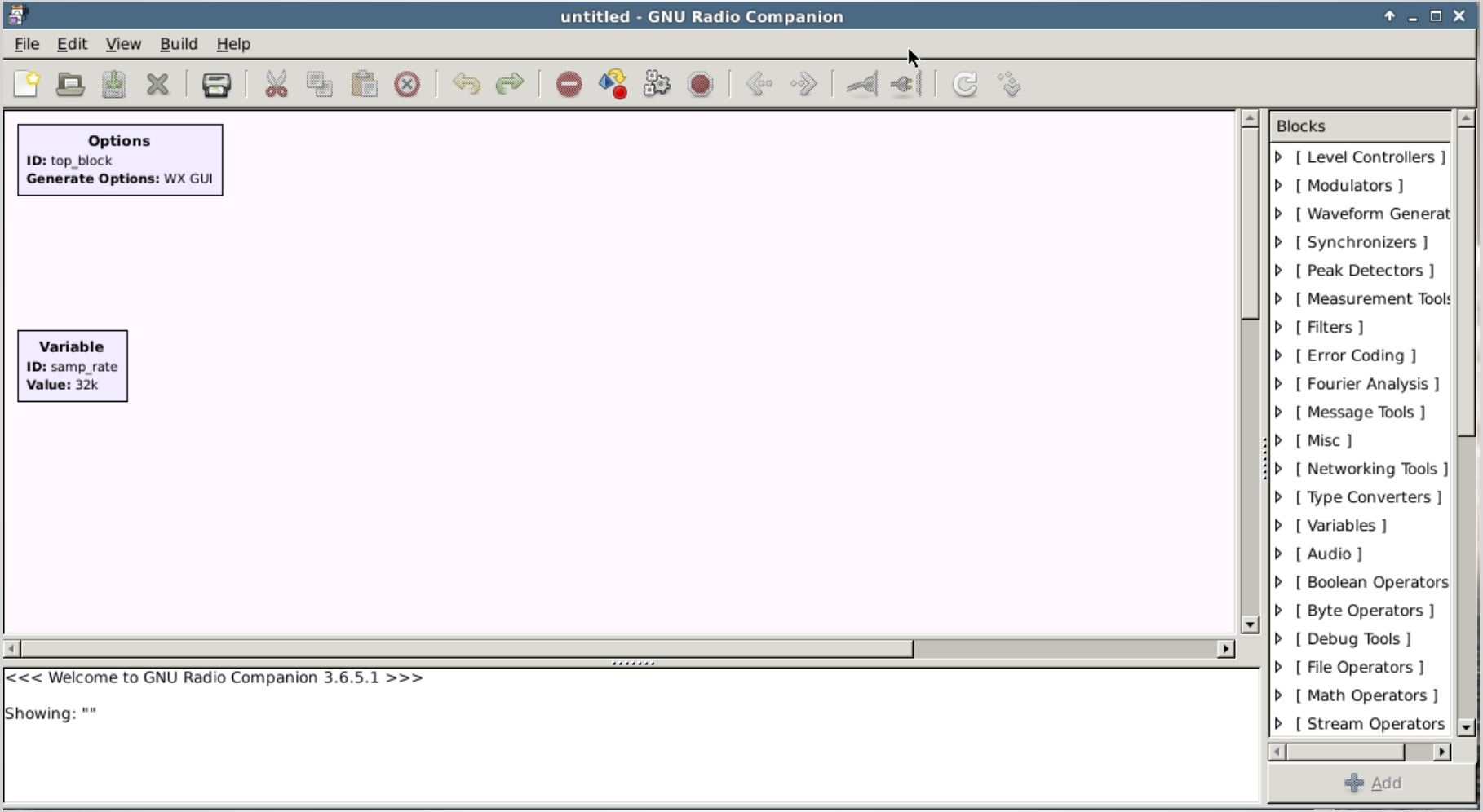
- Notice the pattern at the beginning
- Preamble tells the receiver to pay attention
- Sequence of ‘on’ and ‘off’ patterns
- Notice the equal length on and off segments
- Then notice how it pauses and then goes to variable length

Processing Data

- So we know we've got reasonable data
- How do we process it?
- GNU Radio comes with a ton of modules to do so
- Lets crack into it...

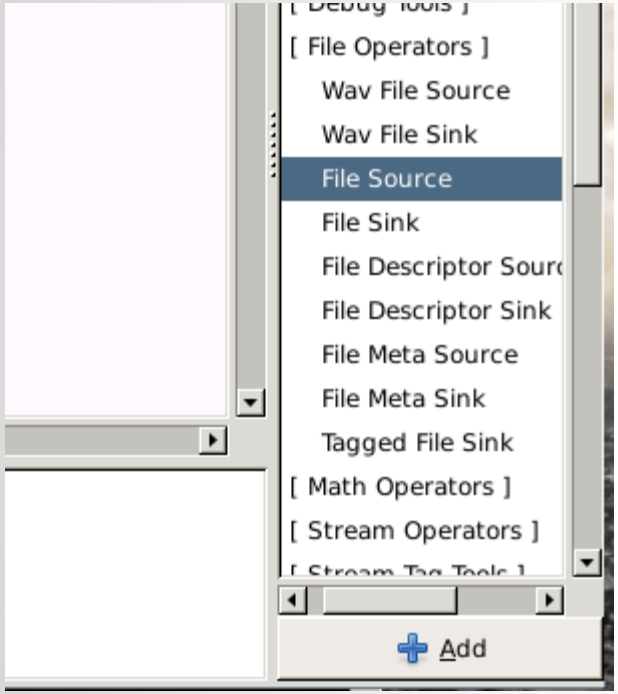
GNU Radio Companion

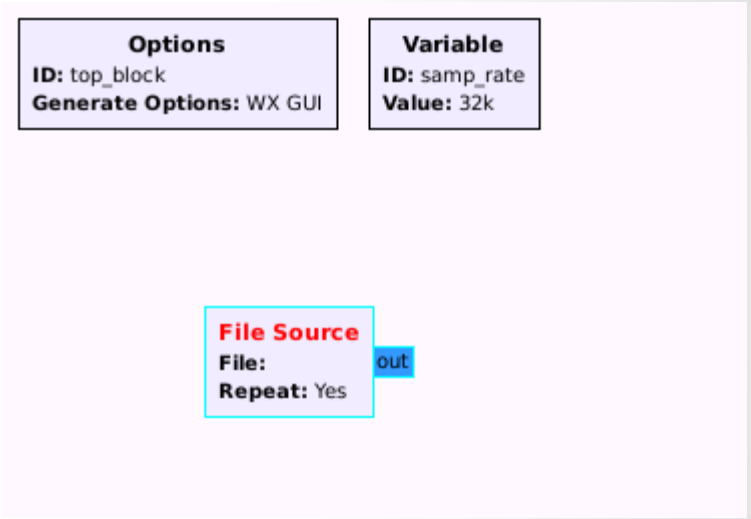
- GRC is a GUI environment for connecting GNU Radio modules
- Works as a flow graph
- Saves as XML
- Compiles to Python
- Amazingly, this actually works. I know.


untitled - GNU Radio Companion
File Edit View Build Help
Options
ID: top_block
Generate Options: WX GUI
Variable
ID: samp_rate
Value: 32k
<<< Welcome to GNU Radio Companion 3.6.5.1 >>>
Showing: ""
Blocks
[Level Controllers]
[Modulators]
[Waveform Generat
[Synchronizers]
[Peak Detectors]
[Measurement Tools
[Filters]
[Error Coding]
[Fourier Analysis]
[Message Tools]
[Misc]
[Networking Tools]
[Type Converters]
[Variables]
[Audio]
[Boolean Operators
[Byte Operators]
[Debug Tools]
[File Operators]
[Math Operators]
[Stream Operators
Add

Basics

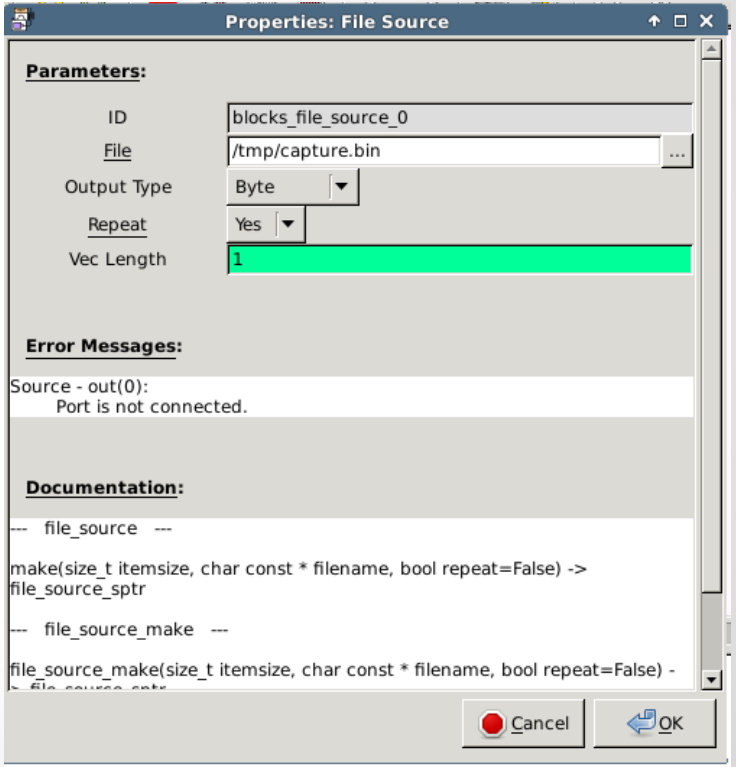
- Modules are on the right
- There are a lot of them
- If you can’t find what you’re looking for, you can try typing part of the name and you’ll get a ‘search results’ category


[File Operators]
Wav File Source
Wav File Sink
File Source
File Sink
File Descriptor Sourc
File Descriptor Sink
File Meta Source
File Meta Sink
Tagged File Sink
[Math Operators]
[Stream Operators]
Add


Options
ID: top_block
Generate Options: WX GUI
Variable
ID: samp_rate
Value: 32k
File Source
File:
Repeat: Yes
out

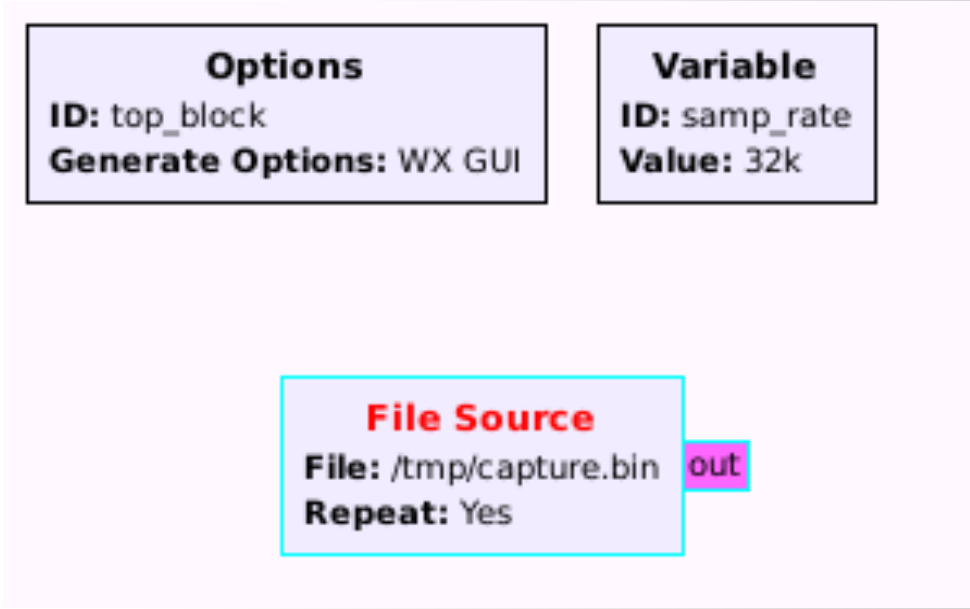
Edit a module

- Doubleclick on the File source
- Set the source
- Output type byte



Things have changed

- Notice the color has changed
- Color indicates output type of each module...



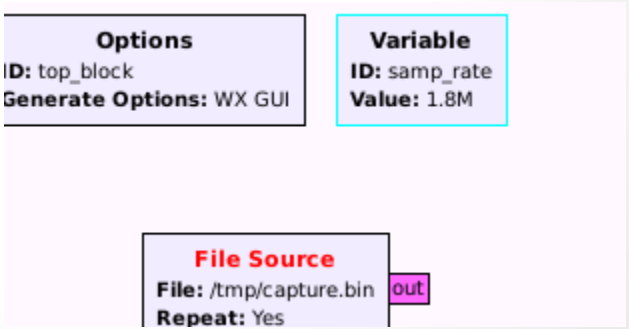
Variables

- Variables work like you’d expect
- Allow you to set data and use it throughout
- Allows math
- Better than hardcoding everything by far
- Doubleclick on the ‘sample rate’ variable

Set The Rate

Parameters:

ID	samp_rate
Value	1800000



Quirks

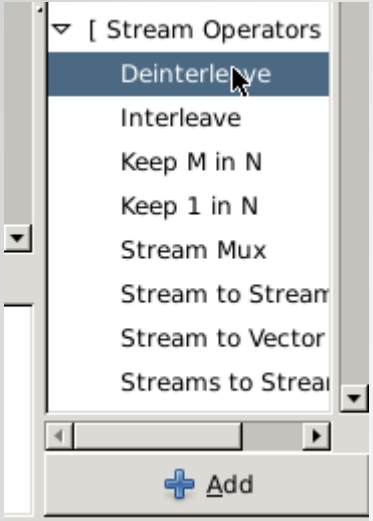
- Notice it changed to “1.8M” in the description
- You may also notice “1.8M” is not a valid input syntax for a number
- Thanks, GNU Radio
- You can use scientific though: 1.8e6
- Yeah. Why not.

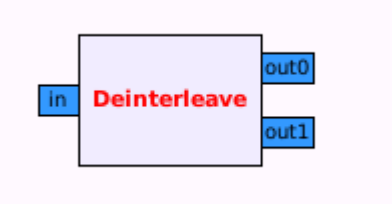
Handling Data

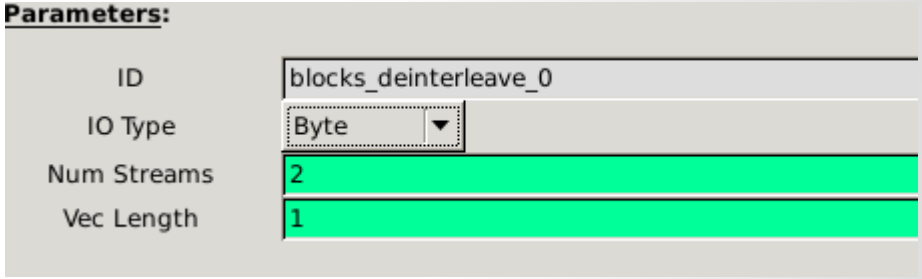
- So we have a weird binary format
- We need to turn that into something useful by GNU Radio
- GR expects ‘Complex’ data - Float IQ in a stream
- We have to get from 8bit interleaved to Complex

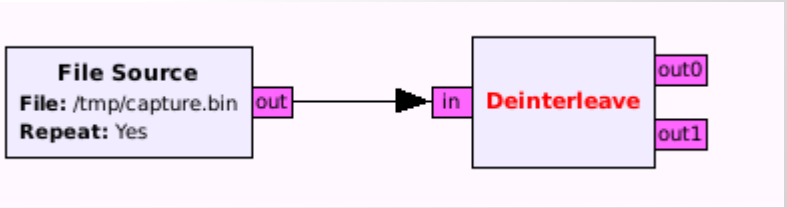
De-Interleave

- De-interleaves a stream
- That is, splits every other byte
- One input, two outputs
- Turns IQIQIQIQ into IIII and QQQQ, basically
- Add module, set type to byte
- Link by clicking ‘out’ of file and ‘in’ of deinterleave


[Stream Operators
Deinterleave
Interleave
Keep M in N
Keep 1 in N
Stream Mux
Stream to Stream
Stream to Vector
Streams to Strea
Add

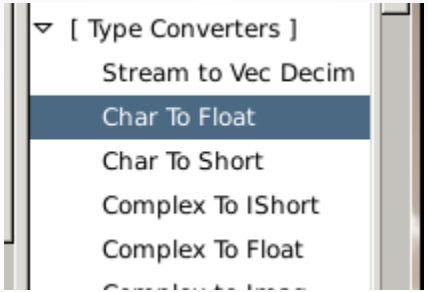

in
Deinterleave
out0
out1

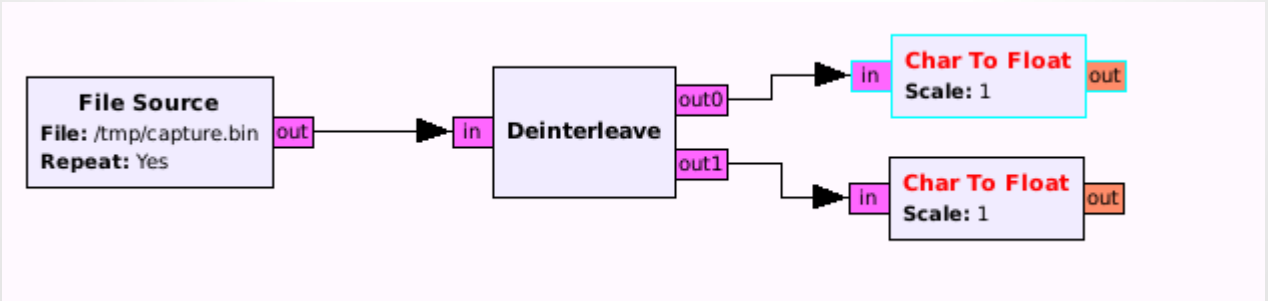

Parameters:
ID
blocks_deinterleave_0
IO Type
Byte
Num Streams
2
Vec Length
1


File Source
File: /tmp/capture.bin
Repeat: Yes
out
in
Deinterleave
out0
out1

Changing Types

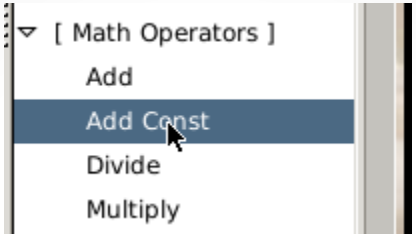
- We need a ‘float’ to turn into a ‘complex’
- So lets change both the I and Q data into ‘float’
- Add module, set type, link - same song, different verse

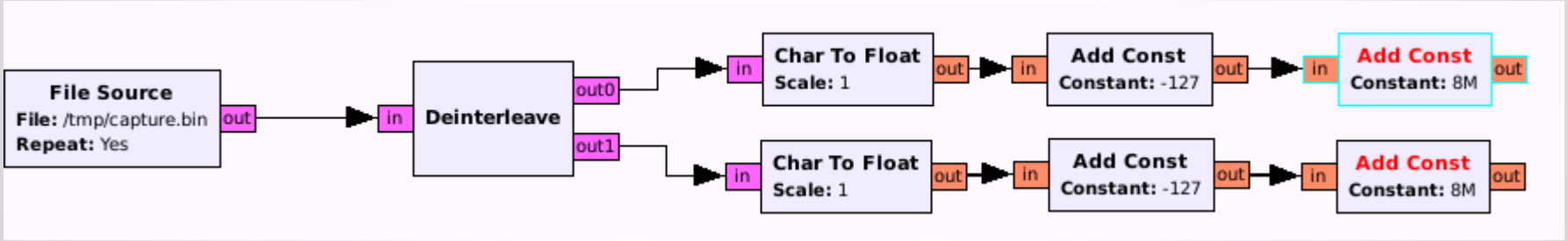

[Type Converters]
Stream to Vec Decim
Char To Float
Char To Short
Complex To IShort
Complex To Float


File Source
File: /tmp/capture.bin
Repeat: Yes
out
in
Deinterleave
out0
out1
in
Char To Float
Scale: 1
out
in
Char To Float
Scale: 1
out

Tweaking Data

- It comes in as an unsigned byte 0..255
- We need to scale it to +/- and re-center it as a float
- Add 2 constants; -127 and 8M
- Add ‘const’ module, set type to float, enter constant

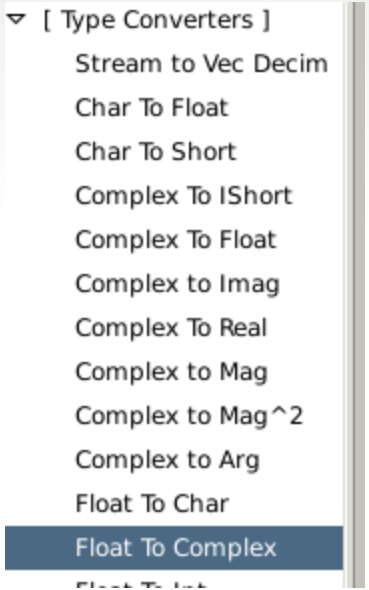

[Math Operators]
Add
Add Const
Divide
Multiply

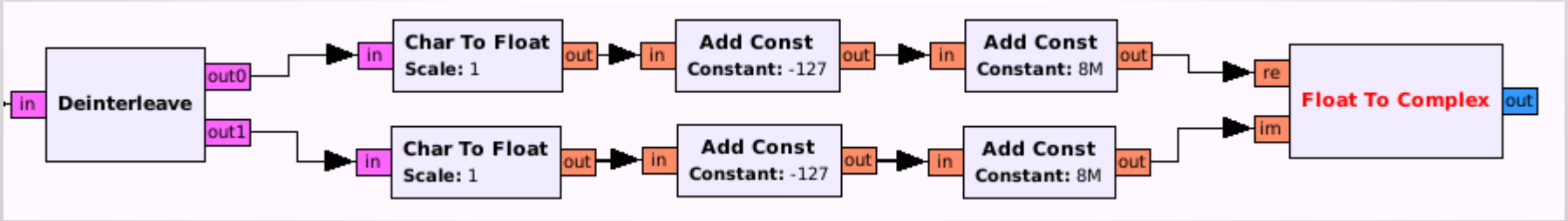

File Source
File: /tmp/capture.bin
Repeat: Yes
out
in
Deinterleave
out0
out1
in
Char To Float
Scale: 1
out
in
Add Const
Constant: -127
out
in
Add Const
Constant: 8M
out
in
Char To Float
Scale: 1
out
in
Add Const
Constant: -127
out
in
Add Const
Constant: 8M
out

Getting Complex

- Convert Float to Complex
- Basically re-interleaving
- Takes I and Q streams and turns them into a Complex stream that GNU Radio expects for just about everything

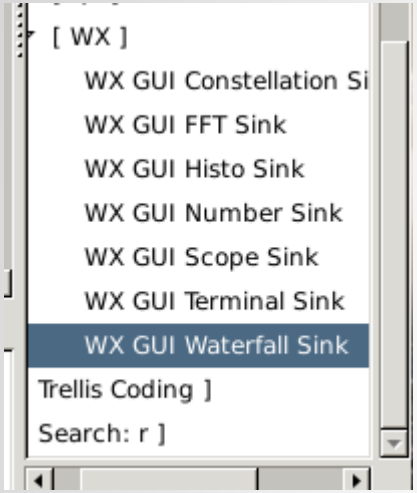
(HINT: MAKE SURE THE COLORS MATCH FOR IN/OUT)

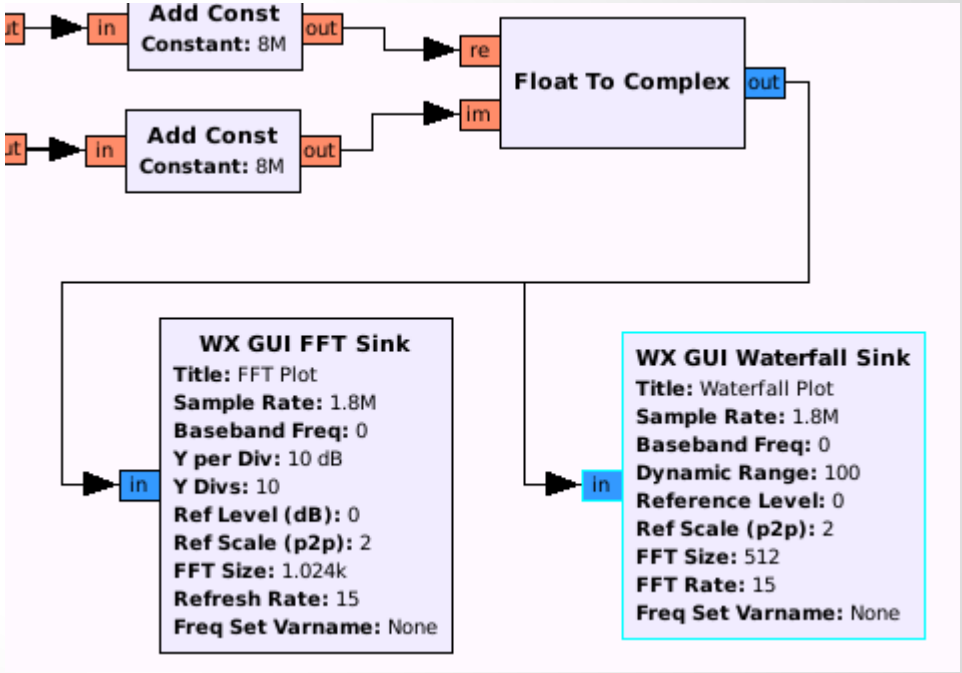

[Type Converters]
Stream to Vec Decim
Char To Float
Char To Short
Complex To IShort
Complex To Float
Complex to Imag
Complex To Real
Complex to Mag
Complex to Mag^2
Complex to Arg
Float To Char
Float To Complex


in
Deinterleave
out0
out1
in
Char To Float
Scale: 1
out
in
Add Const
Constant: -127
out
in
Add Const
Constant: 8M
out
re
im
Float To Complex
out

Lets Look

- Lets look at what we have…
- Add a scope output
- Notice the sample rate
- It has to match the input or things go weird
- Sample rate is conflated with bandwidth for IQ data
- Because reasons.

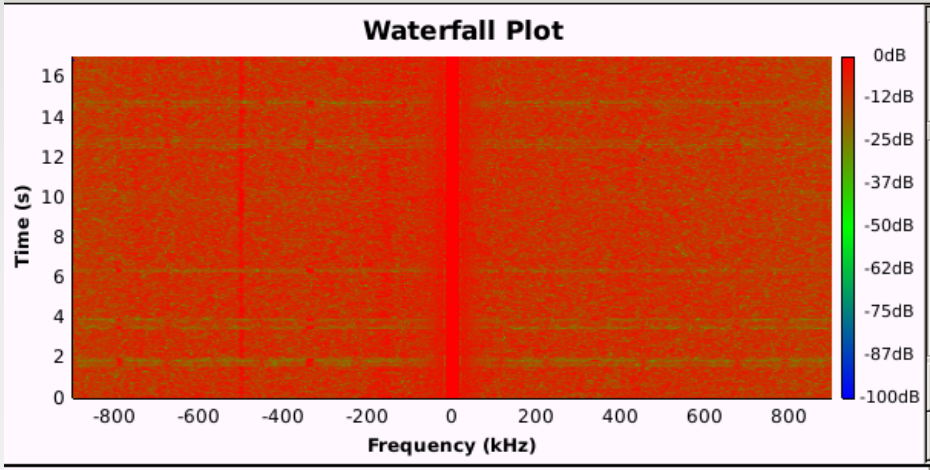

[WX]
WX GUI Constellation Si
WX GUI FFT Sink
WX GUI Histo Sink
WX GUI Number Sink
WX GUI Scope Sink
WX GUI Terminal Sink
WX GUI Waterfall Sink
Trellis Coding]
Search: r]


Add Const
Constant: 8M
in
out
Add Const
Constant: 8M
in
out
re
im
Float To Complex
out
in
WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 1.8M
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
in
WX GUI Waterfall Sink
Title: Waterfall Plot
Sample Rate: 1.8M
Baseband Freq: 0
Dynamic Range: 100
Reference Level: 0
Ref Scale (p2p): 2
FFT Size: 512
FFT Rate: 15
Freq Set Varname: None

Outputs

- One output can be wired to multiple inputs
- FFT is the spectrum-analyzer display we're used to
- Waterfall is the historical level display
- We've basically recreated GQRX!

Top Block

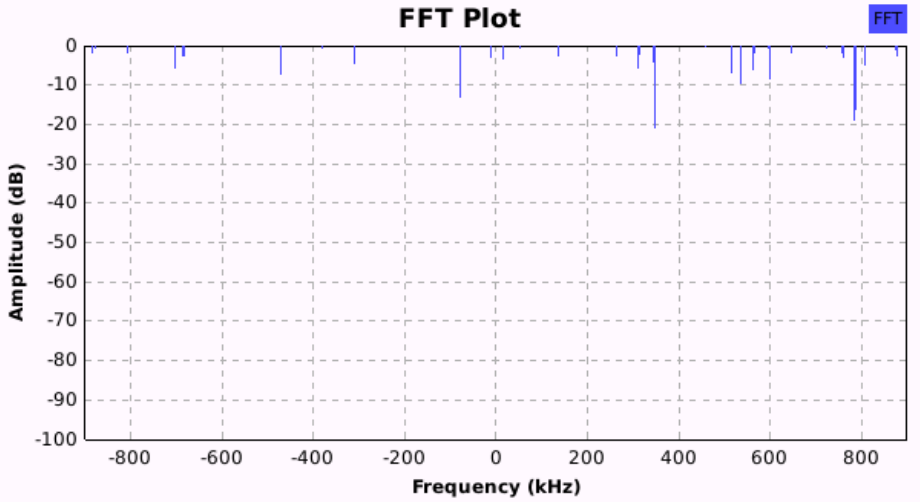


Options
- [ ] Average
Avg Alpha: 0.1333

Axes Options
Time Scale: + -
Dyn Range: + -
Ref Level: + -
Color: RGB1
Autoscale

Clear

Stop



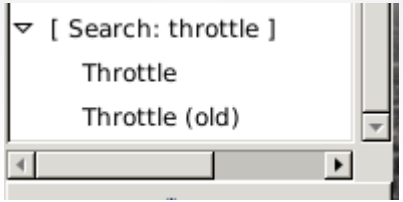
Trace Options
- [ ] Peak Hold
- [ ] Average
Avg Alpha: 0.1333
- [ ] Persistence
Persist Alpha: 0.1887
- [ ] Trace A Store
- [ ] Trace B Store

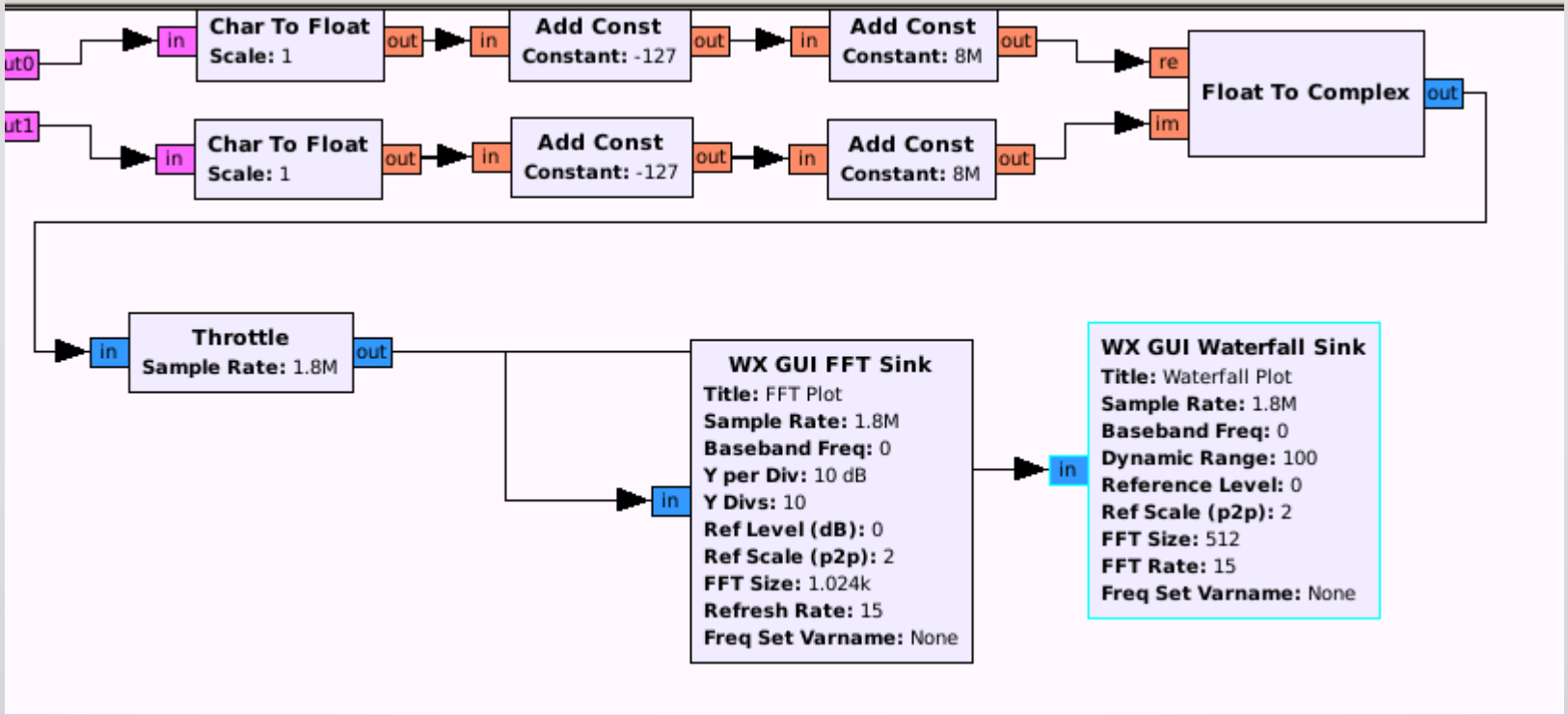
Axis Options
dB/Div: + -
Ref Level: + -
Autoscale

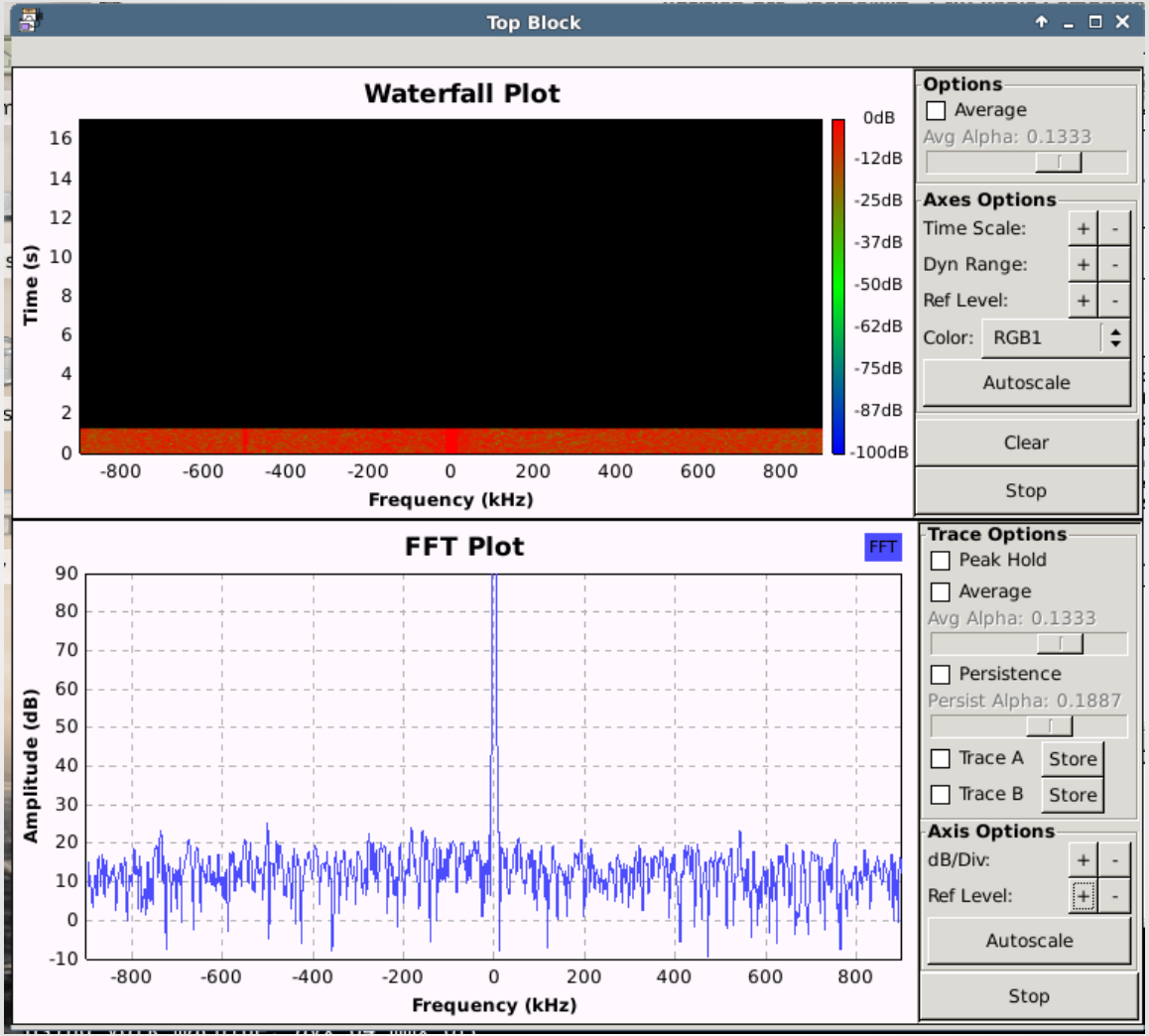
Stop

What Happened?

- Depending how fast your system is, something went wacky
- GNU Radio is shoving data as fast as possible into the graphs
- Lets slow it down...


[Search: throttle]
Throttle
Throttle (old)


Char To Float
Scale: 1
Add Const
Constant: -127
Add Const
Constant: 8M
Char To Float
Scale: 1
Add Const
Constant: -127
Add Const
Constant: 8M
Float To Complex
Throttle
Sample Rate: 1.8M
WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 1.8M
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
WX GUI Waterfall Sink
Title: Waterfall Plot
Sample Rate: 1.8M
Baseband Freq: 0
Dynamic Range: 100
Reference Level: 0
Ref Scale (p2p): 2
FFT Size: 512
FFT Rate: 15
Freq Set Varname: None


Top Block
Waterfall Plot
Time (s)
16
14
12
10
8
6
4
2
0
-800
-600
-400
-200
0
200
400
600
800
Frequency (kHz)
0dB
-12dB
-25dB
-37dB
-50dB
-62dB
-75dB
-87dB
-100dB
Options
Average
Avg Alpha: 0.1333
Axes Options
Time Scale:
Dyn Range:
Ref Level:
Color:
RGB1
Autoscale
Clear
Stop
FFT Plot
FFT
Amplitude (dB)
90
80
70
60
50
40
30
20
10
0
-10
-800
-600
-400
-200
0
200
400
600
800
Frequency (kHz)
Trace Options
Peak Hold
Average
Avg Alpha: 0.1333
Persistence
Persist Alpha: 0.1887
Trace A
Store
Trace B
Store
Axis Options
dB/Div:
Ref Level:
Autoscale
Stop

Scaling Display

- Click the ‘+’ on ‘Ref Level’ to scale the display to center the signal
- Should look like what we saw in GQRX now
- Save this grc! You’ll need to do this conversion on every recorded session.

What other tools exist?

1. dsd (audio input selection problem)
   - Demodulate P25, Mototurbo
2. multimon-ng
   - Demodulates almost ALL THE THINGS
3. smartnet-scanner
   - More P25 goodness (uses radioreference)

Linux Only?

- For most of the tools, yes.
- To look around, no.
- Use the same dongle
- Opposed to GQRX

  SDRSharp - plugins

  HDSDR - one we’ve seen

Where is there more info?

http://www.rtl-sdr.com

http://www.radioreference.com

http://www.dangerousprototypes.com

Oh, and http://www.kismetwireless.net