SharkFest'22 US
Kansas City, MO
July 9-14

#sf22us

When is a packet not a packet?

Mike Kershaw
Kismet Wireless / Hak5


HELLO
my name is
inigo montoya
you killed my father
prepare to die

Packets!?

I am ***Mike Kershaw***

I do WiFi and radio and packet stuff.

You can find me at @KismetWireless

Scattershot

- Huge topic but we'll try
- When and why SDR
- Some basic SDR concepts
- Digging into some real examples of decoding techniques
- Hopefully enough to start a journey!

Packets and interfaces

- Normally we capture packets from network interfaces
- Ethernet, WiFi, Bluetooth, dedicated NIC
- Even monitor mode WiFi comes in over a network interface
- Pcap + Wireshark
- … but what else is out there?

Aliens. Probably.

But also…

- Power meters
- Airplanes
- Weather stations
- Light switches
- Tire pressure monitors
- Random IOT
- More

Standard hardware

- Fixed-frequency radios
- Demodulation and decode in hardware
- Returns data to the OS as packets
- Usually talks libpcap
- Completely useless at anything else
- Your WiFi card isn't going to see BT

Software Defined Radio

- Multi-frequency receiver
- Analog-to-digital converter
- Sometimes additional accelerators like FPGA or dedicated processors
- Has no idea what a packet is
- Sends constant stream of data 100% of the time

Software Defined Radio

- Swiss army knife
- So why isn't everything done with SDR instead of custom chips?
- Have you ever tried to eat a meal with a Swiss army knife?


VICTORINOX

SDR Plusses

- Discover new protocols & devices
- Capture "infinite" protocols with a single device
- Capture signals no commercial HW exists for
- Can manipulate protocols in ways dedicated HW can't

SDR Negatives

- Can be expensive
- Requires a lot of power (energy)
- Requires a lot of CPU
- Requires lots of bandwidth (usb, ram, etc)
- Rarely a plug-and-play solution
- Some pre-made tools, but a lot of "gradware"

SDR vs Dedicated

- ASIC will *always* win if one is obtainable
- Fixed frequency = less interference
- Dedicated HW uses less power
- Only bothers the OS when there is a packet
- A $750 SDR can *just about* be a $20 WiFi card

SDR Hardware

- Lots of options now
- Cheap
  - RTL-SDR
- Medium-to-Pro
  - HackRF, BladeRF, LimeSDR, Airspy, Lime
- Lab-grade
  - BladeRF, USRP

What makes expensive SDR "better"

- Frequency range
  - What RF frequencies the HW can tune to
- Sample depth
  - Fidelity of captured data
- Transmit capability
  - Many are RX only
- Additional hardware
  - On-board FPGA, etc

Cheap doesn't mean bad (kinda)

- The RTL-SDR is *dirt cheap ($25)* but still very usable
- It's not *good*, really, but it's fine for a lot!
- Looking at many protocols at the same time means you need many SDRs
- Why spend $400+ when you're just starting out?
- Why spend $400+ when $20 is enough sometimes?
- Great intro to the SDR space

Specs

- What specs are you likely to see?
- How much you care depends on what you plan to do
- Remember: "Bigger number means better tool"
- "Better tool means better person"
- Not really

Frequency Ranges

- Everything happens at a frequency
- Measured in Hertz (Hz)
- 1000Hz = 1KHz, 1000KHz = 1MHz, etc
- WiFi is at 2400MHz. GPS is around 1200MHz.
- Non-licensed (consumer) gear tends to cluster in ISM bands
- 433MHz / 900MHz / 2400MHz / 5800MHz

Ranges of SDRs

- Every SDR will list the ranges it can tune to
- Determined by HW
- RTL-SDR ~ 32MHz to 2200MHz but varies
- HackRF 0MHz – 6000MHz
- BladeRF 47MHz – 6000MHz
- USRP – Varies by model and module
- Others – Often 10MHz – 3500MHz

RF Bandwidth

- How much frequency captured at once
- Determines how wide a signal you can see
- More bandwidth = more data = more computer bandwidth, too! (RAM, CPU)
- Most SDR support a range of bandwidths
- You need as much bandwidth as your protocol uses

Bandwidth

- RTL-SDR – 2.4MHz
- HackRF – 20MHz
- BladeRF – 56MHz
- USRP – Varies, ~20MHz to 80MHz+

Antennas

- Antennas are based on frequency
- For WiFi, all antennas are designed to work with WiFi frequencies so they're all interchangeable
- For SDR, this isn't true since you're likely covering wildly different frequencies

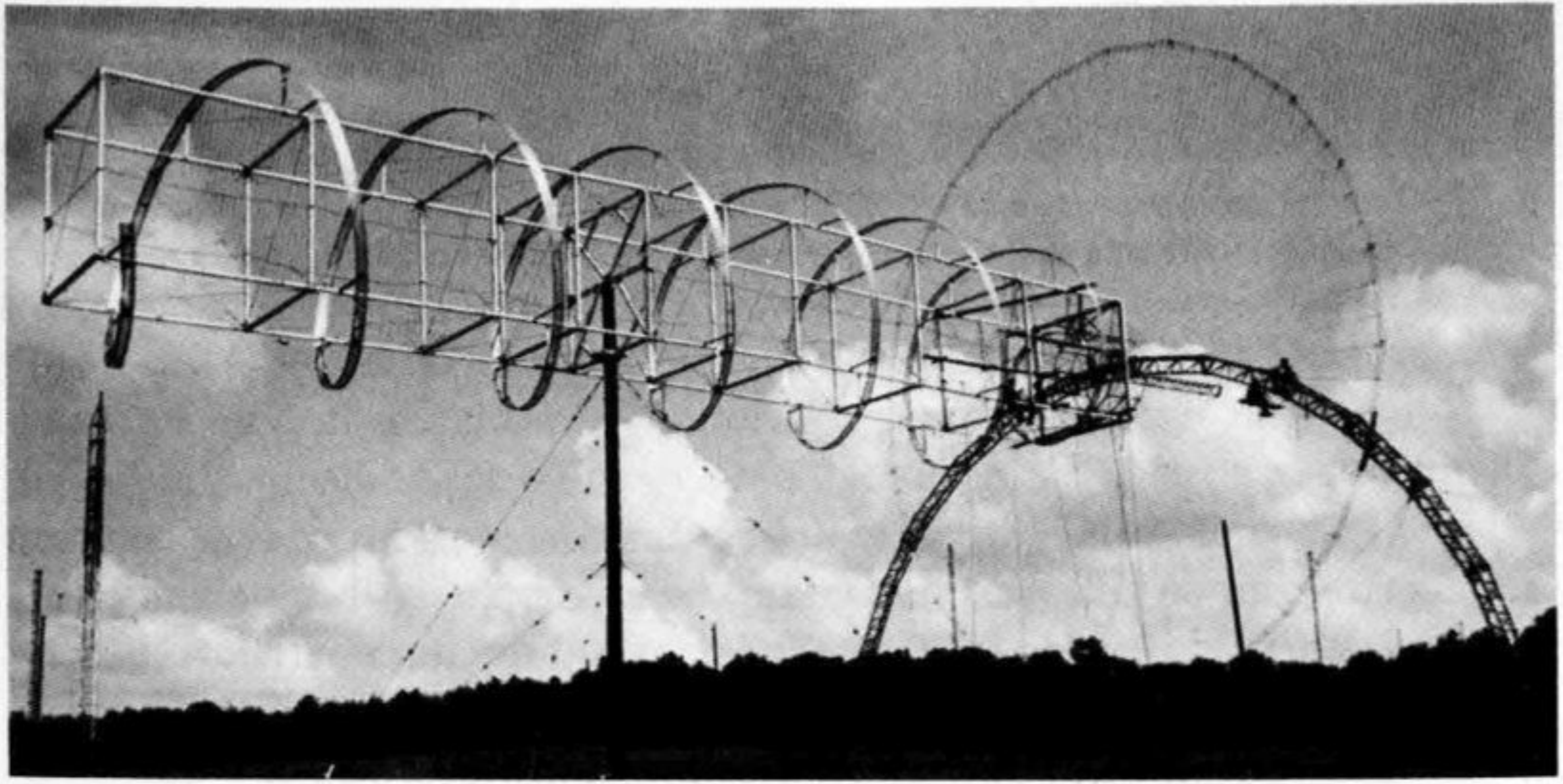


Figure 7-5 Rotatable (in azimuth) 6-turn helical antenna about 45 m long for operation at frequencies around 10 MHz ($\lambda = 30$ m). Note workmen on arch at far end for scale. (*Courtesy Electro-Physics Laboratory.*)

Garbage antennas

- The antennas that come with most cheap SDRs are garbage
- Better than nothing, but maybe only barely
- Consider picking up some proper ones, they're cheap

Variable antennas

- Old-school FM radio telescopic antennas
- The amount you extend the antenna determines the frequency
- Some actually have it marked
- Often ones good at wide-band RX are bad at TX

SharkFest'22 US
Kansas City, MO
July 9-14

#sf22us

Dedicated antennas

- Easy to find dedicated antennas for most frequency bands you'll care about
- 433mhz, 900mhz, etc
- Large HAM community for antenna resources
- Worth considering when you get more serious

Wrong antenna?

- What happens w/ the wrong antenna?
- For RX, just a crappy signal
- For TX, more of a problem
- Don't TX with the wrong antenna
- Worst case, you can damage your equipment!

Antenna gain

- Doesn't make more signal, just directs it
- More gain isn't always better
- Signal comes from somewhere – more gain in horizontal means missing signals above/below
- Don't go crazy buying the biggest antenna first

Filters

- We can filter in SW but that only solves some of the problem
- Commercial radios use tight hardware filters to exclude all other bands
- Signal outside our target band can swamp the RX
- Signal can show up as aliases

Aliasing

- Fundamental to SDR internals
- SDR takes target frequency and resamples it during capture to an intermediary frequency
- Signals on the "opposite side" of the intermediary are indistinguishable from where you tuned
- End result: You'll see FM radio all over!

Rejecting signal

- No way to prevent aliasing in software
- You can buy dedicated hardware filters
- Placed in-line with your antenna before the radio
- Can be powered or passive
- But you need one for each band

Connectors

- SMA is the most common
- Most WiFi uses reverse-polarity connectors
- Most SDR uses traditional connectors
- Don't mix them up!
- Either you'll get no connectivity at all…
- … or you'll mash the two pins together and bend one

Obstructions

- Lower frequencies go through obstacles better
- Most SDR is relatively low frequency…
- But in a basement or bottom of a building isn't going to do you any favors
- An outdoor antenna can be a huge benefit

SharkFest'22 US
Kansas City, MO
July 9-14

#sf22us

Safety

- If you put an antenna outside
- Be aware of power lines!
- Be aware of lightning!
- Unless you're ready to set up proper lightning arrestors...
- Don't make a permanent outdoor antenna!

Talking to a SDR

- Most are USB2 or USB3
- Some PCI
- Some gig-e or 10gbe
- There are no standards for talking to a SDR
- Most just shove data as fast as possible

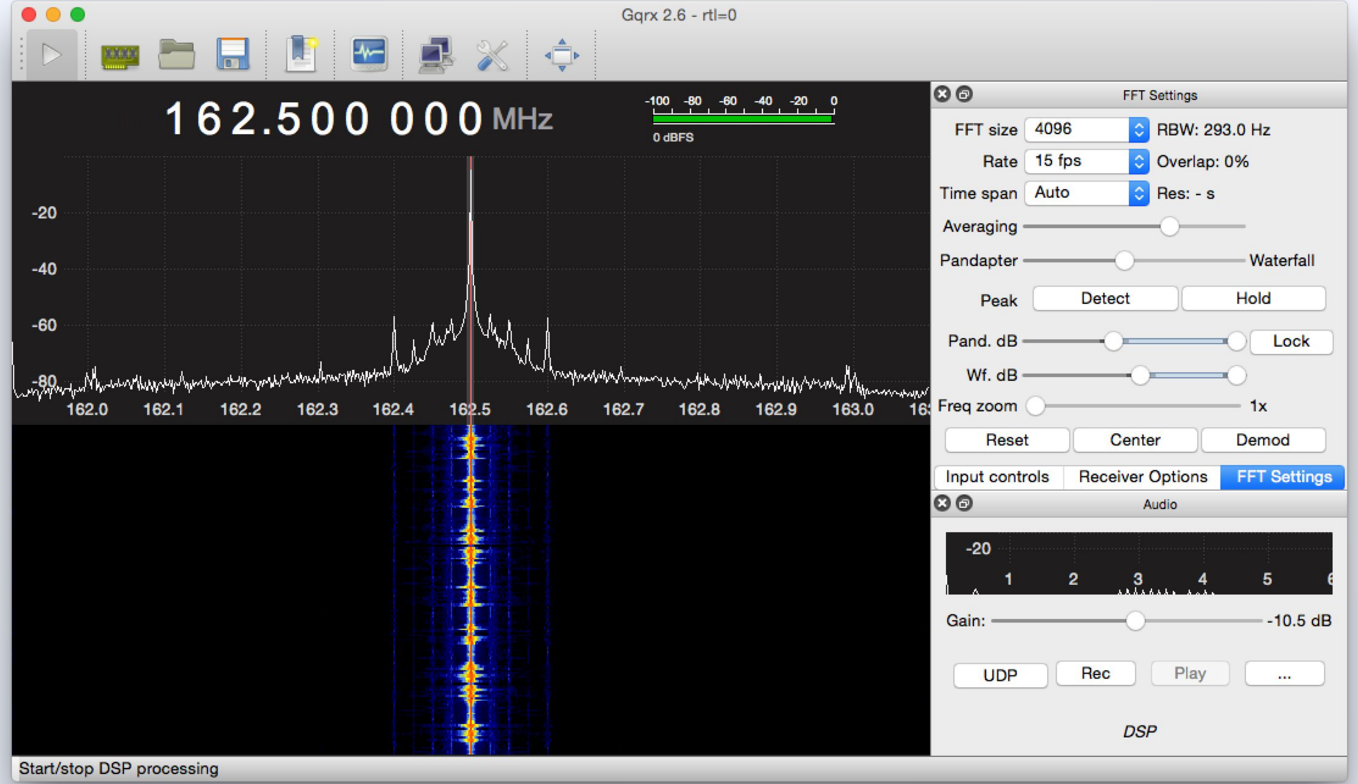
SDR Tools

- GNU Radio
  - Gold standard open source radio framework
- Matlab
  - Commercial math framework
- LiquidSDR
  - Low-level C library for signal processing

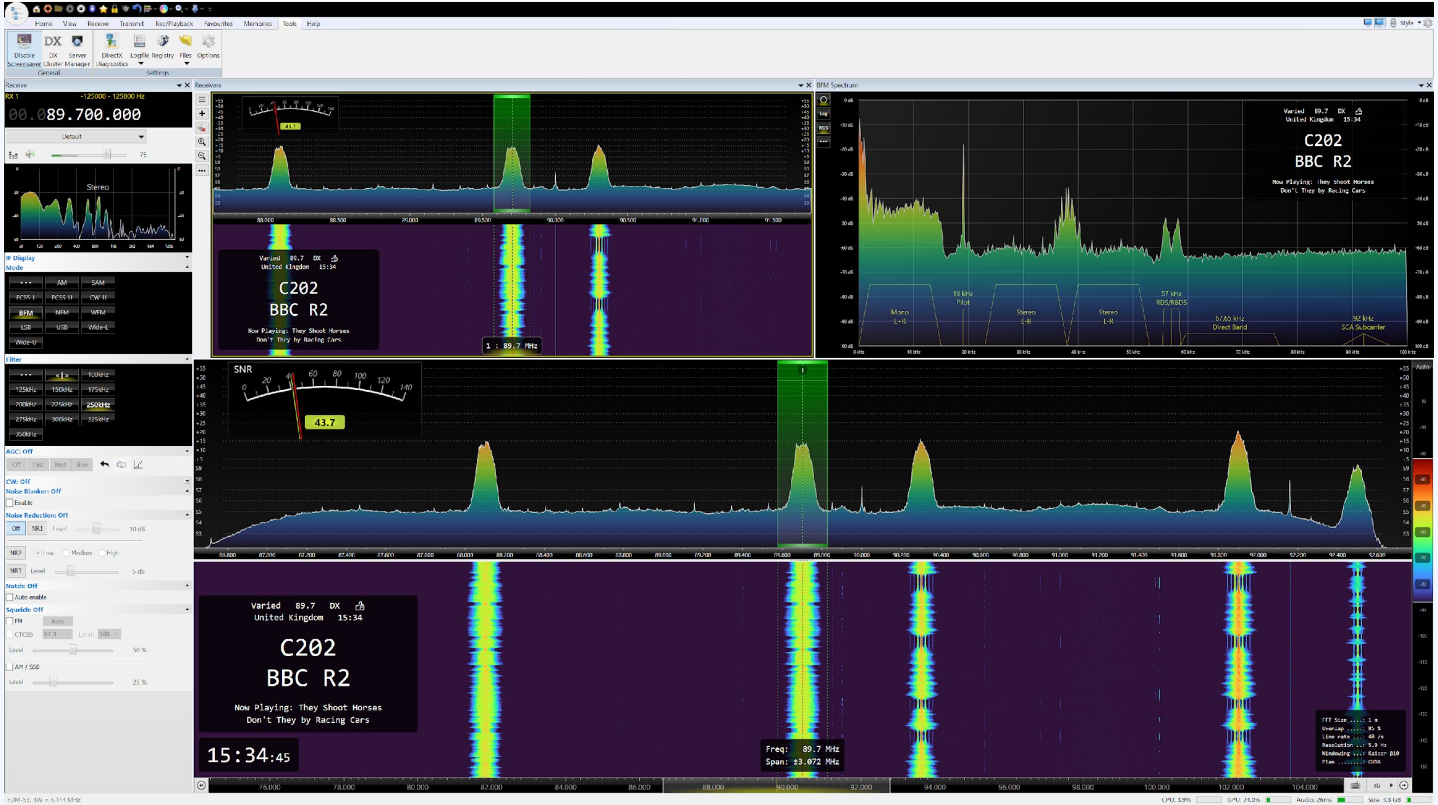
Exploration tools

- GQRX
  - Waterfall, basic demod for some common formats
- SDRSharp
  - Common windows tool
- SDR++
  - New multi-platform exploration tools
- Universal Radio Hacker (URH)
  - Multi-platform protocol decoding & exploration

GQRX


Gqrx 2.6 - rtl=0
162.500 000 MHz
0 dBFS
-20
-40
-60
-80
162.0
162.1
162.2
162.3
162.4
162.5
162.6
162.7
162.8
162.9
163.0
FFT Settings
FFT size 4096
RBW: 293.0 Hz
Rate 15 fps
Overlap: 0%
Time span Auto
Res: - s
Averaging
Pandapter
Waterfall
Peak
Detect
Hold
Pand. dB
Lock
Wf. dB
Freq zoom
1x
Reset
Center
Demod
Input controls
Receiver Options
FFT Settings
Audio
Gain:
-10.5 dB
UDP
Rec
Play
...
DSP
Start/stop DSP processing

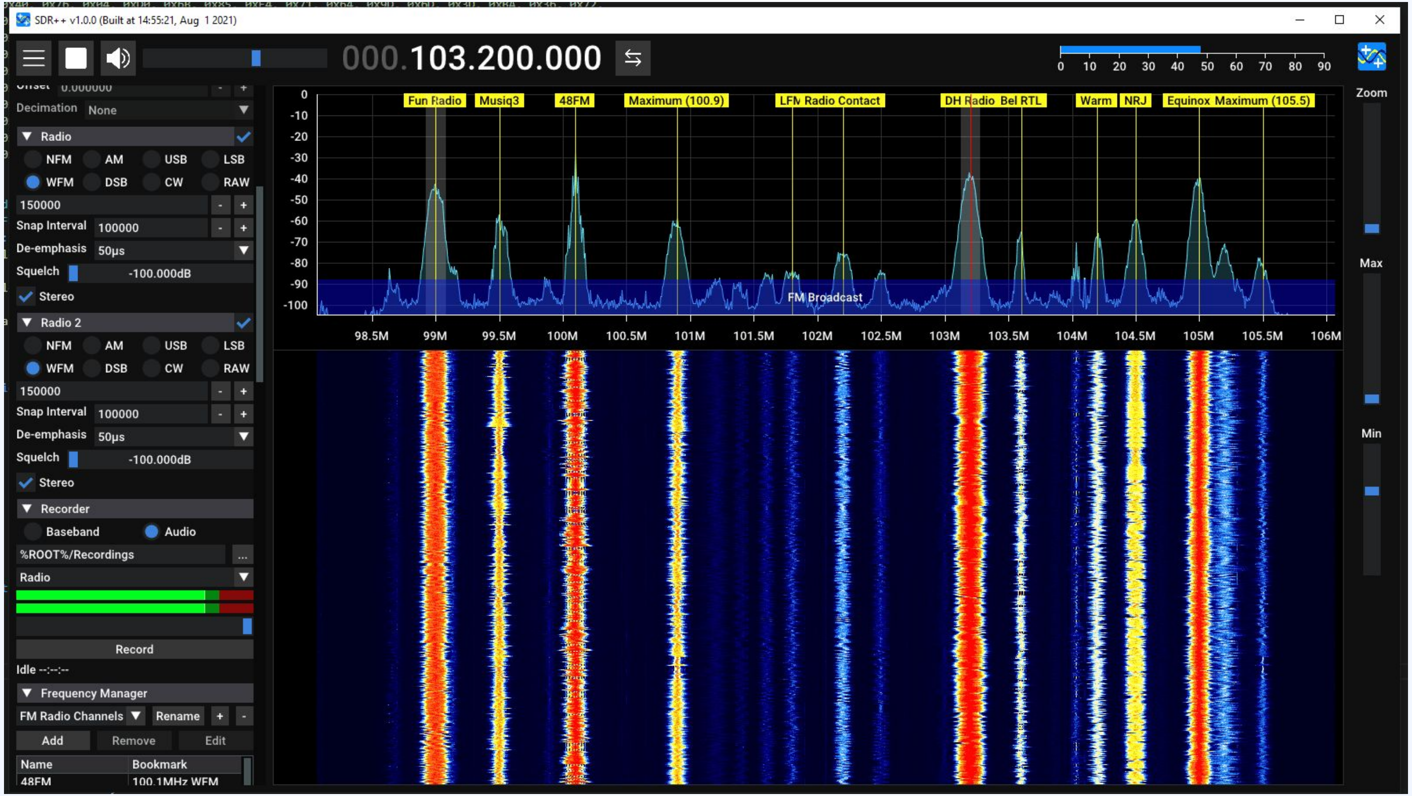
SDR#


Home
View
Receive
Transmit
Rec/Playback
Favourites
Memories
Tools
Help
89.700.000
SNR
43.7
BFM Spectrum
Varied 89.7 DX
United Kingdom 15:34
C202
BBC R2
Now Playing: They Shoot Horses
Don't They by Racing Cars
15:34:45
Freq: 89.7 MHz
Span: ±3.072 MHz

SharkFest'22 US
Kansas City, MO
July 9-14

#sf22us

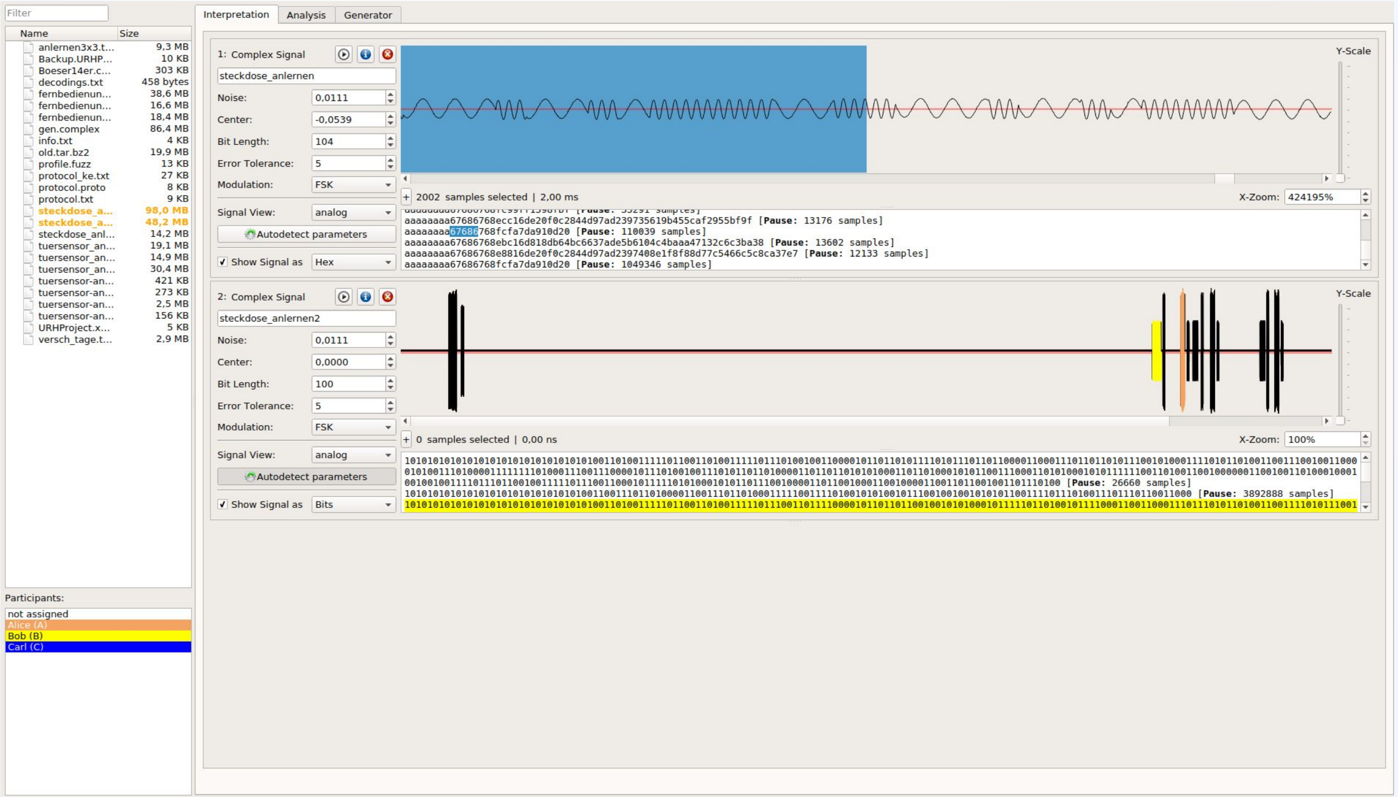
SDR++



SharkFest'22 US
Kansas City, MO
July 9-14

#sf22us

Universal Radio Hacker



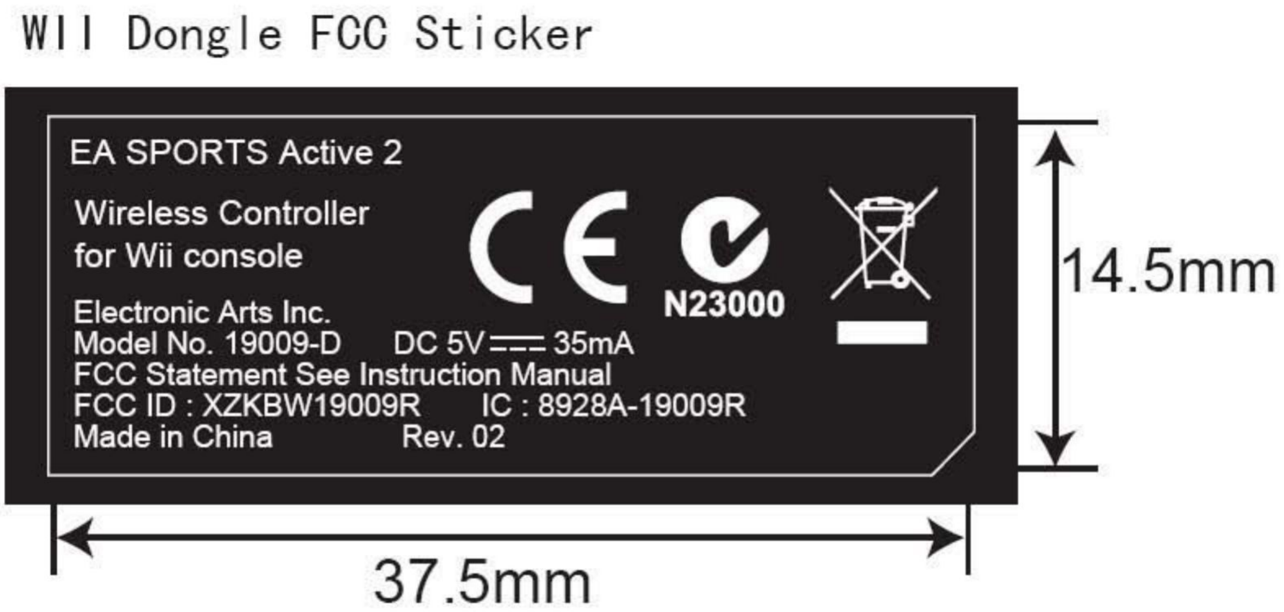
Installing the tools

- Linux: Ubuntu, Pentoo, others have packages
- DragonOS is a custom Linux distro for SDR
- MacOS: Brew can install many of the tools
- Windows: More of a hassle due to USB driver model, but doable

Looking for signals

- Research the protocol/device
- Some devices label the frequencies
- FCC filings
- Public docs (for large public protocols)
- Scan around with survey tools

FCC


WII Dongle FCC Sticker
EA SPORTS Active 2
Wireless Controller
for Wii console
Electronic Arts Inc.
Model No. 19009-D
DC 5V 35mA
FCC Statement See Instruction Manual
FCC ID : XZKBW19009R
IC : 8928A-19009R
Made in China
Rev. 02
N23000
14.5mm
37.5mm

FCCID.io

FCC ID.io Blog Search

1	Original Equipment	2010-09-13	wvZyipQLI6t1Quig4SsQTQ==

Operating Frequencies

Frequency Range	Rule Parts	Line Entry
2.406-2.476 GHz	15C	1

Exhibits

All

Document	Type	Submitted Available
19009-D User Manual	Users Manual Adobe Acrobat PDF (1507 kB)	2010-09-13 2010-09-13
19009-D Test Photo for FCC	Test Setup Photos Adobe Acrobat PDF (217 kB)	2010-09-13 2010-09-13

Know your laws!

Legalisms

- Know the laws for your country / region!
- In the US this is governed by the FCC
- *Transmitting* without a license is almost always illegal
- *Listening* can still be illegal on some frequencies in some jurisdictions!
- In the US, can vary by court district!

Legalisms (2)

- Can someone tell you're listening to a frequency?
- Not unless you talk about it.
- Is it still illegal to do it?
- Sure is! (depending on country)

Legalisms (3): Never, ever

- Cell phone frequencies
- Pager frequencies
- Transmitting on any licensed band
- *Especially* transmitting on cell, pager, and GPS
- Considered *terrorism* in the US for interfering with E-911

Legalisms (4)

- Moral vs Legal
- Is it immoral to listen to a random transmission to learn more?
- *Personally* I'd say usually not…
- Still illegal! So don't do it!
- We're only going to talk about legit things to listen to today!

Existing one-shot tools

- ADSB (airplane): Dump1090, Kismet, Others
- AMR (power meters): rtl-amr, Kismet
- TPMS/Thermometers/Weather Stations/Hundreds of others: rtl_433
- Great for pulling out data from existing devices, but what if we want to go deeper?

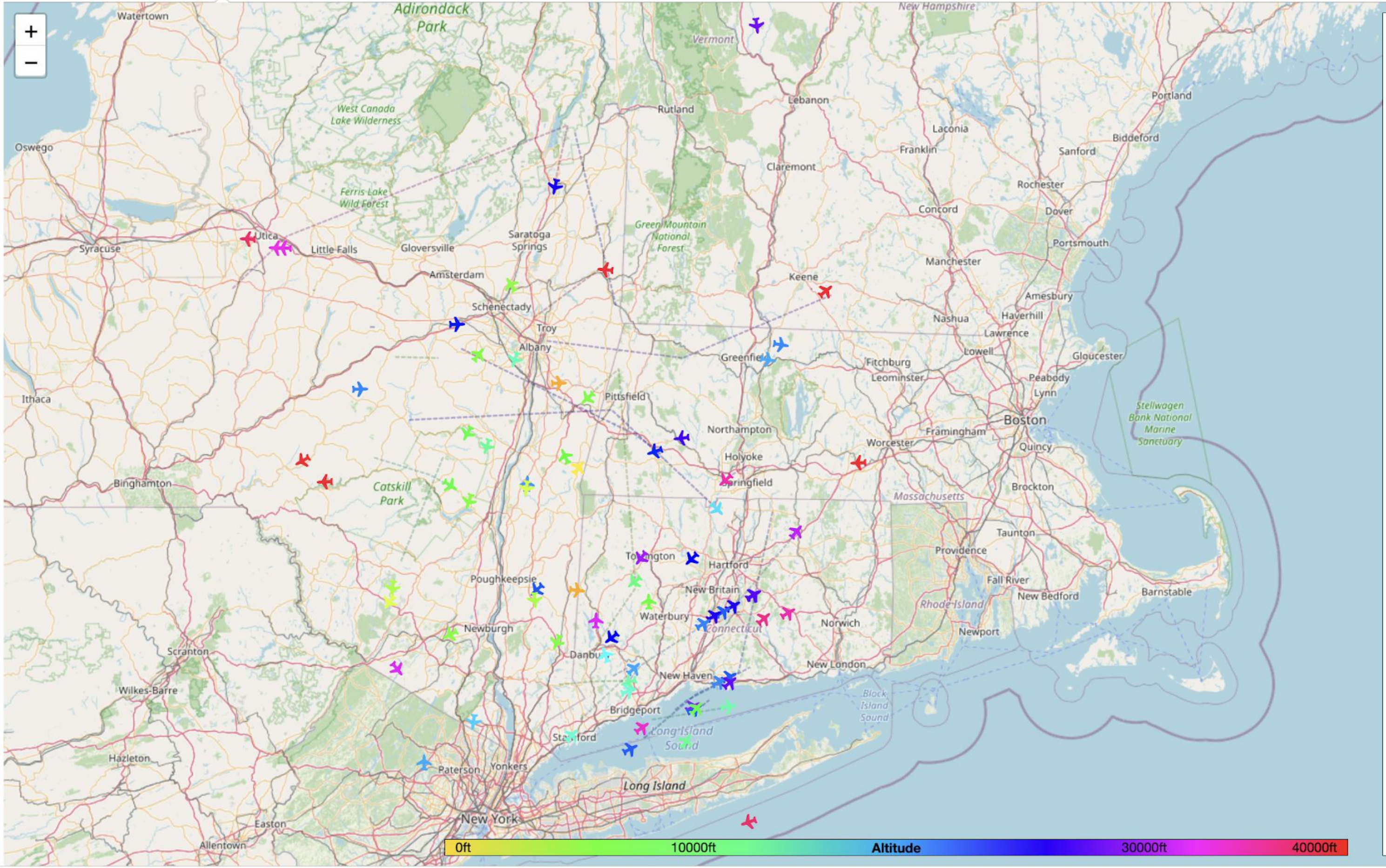
ADSB

```
*8dabb972ea428866d31c083a69b4;
CRC: 000000
RSSI: -13.1 dBFS
Score: 1800
Time: 52733595.42us
DF:17 AA:ABB972 CA:5 ME:EA428866D31C08
 Extended Squitter Target state and status (V2) (29/1)
  ICAO Address:   ABB972 (Mode S / ADS-B)
  Air/Ground:     airborne
  Target State and Status:
    Target altitude:   MCP, 34016 ft
    Altimeter setting: 1013.6 millibars
    Target heading:    253
    ACAS:              operational
    NACp:              8
    NICbaro:           1
    SIL:               3 (per sample)
```

Kismet - xenon

Unknown

Devices | Alerts | SSIDs | **ADSB Live**



102 planes in the past 10 minutes

Flight: MXY210
Model: EMBRAER S A ERJ 190-200 IGW
Operator: BANK OF UTAH TRUSTEE
Altitude: 28000 ft
Speed: 419 MPH

ICAO	ID	Alt	Spd	Hed	Msgs
39b44e	Unknown	28650	491	59	144
3c4b2f	Unknown	35000	496	55	40
3c64ee	Unknown	19450	437	188	134
406814	Unknown	37000	468	49	61
406b20	Unknown	40000	470	232	49
4077d3	Unknown	16775	415	44	10
485b43	Unknown	22950	473	59	71
a008c5	101DQ	21500	432	63	70
a00c80	102DU	31125	449	218	42
a00d5f	102NS	35825	372	72	14
a01f13	107DU	7850	301	202	48
a03210	111Y	24575	394	252	111
a0681c	125UW	7175	241	329	76
a075a3	129FE	0	242	116	1
a11861	17RX	9750	290	359	73
a11fc1	17133	15850	303	252	34
a13687	1777M	27000	489	58	71
a15730	Unknown	8400	380	209	162
a16fca	192BZ	28000	419	263	159
a19372	200QS	3975	268	188	88
a1c229	212NN	34000	413	272	262
a1cc1a	Unknown	5825	179	176	97
a1d096	216JB	26000	412	232	187

Leaflet | © OpenStreetMap

RTL433

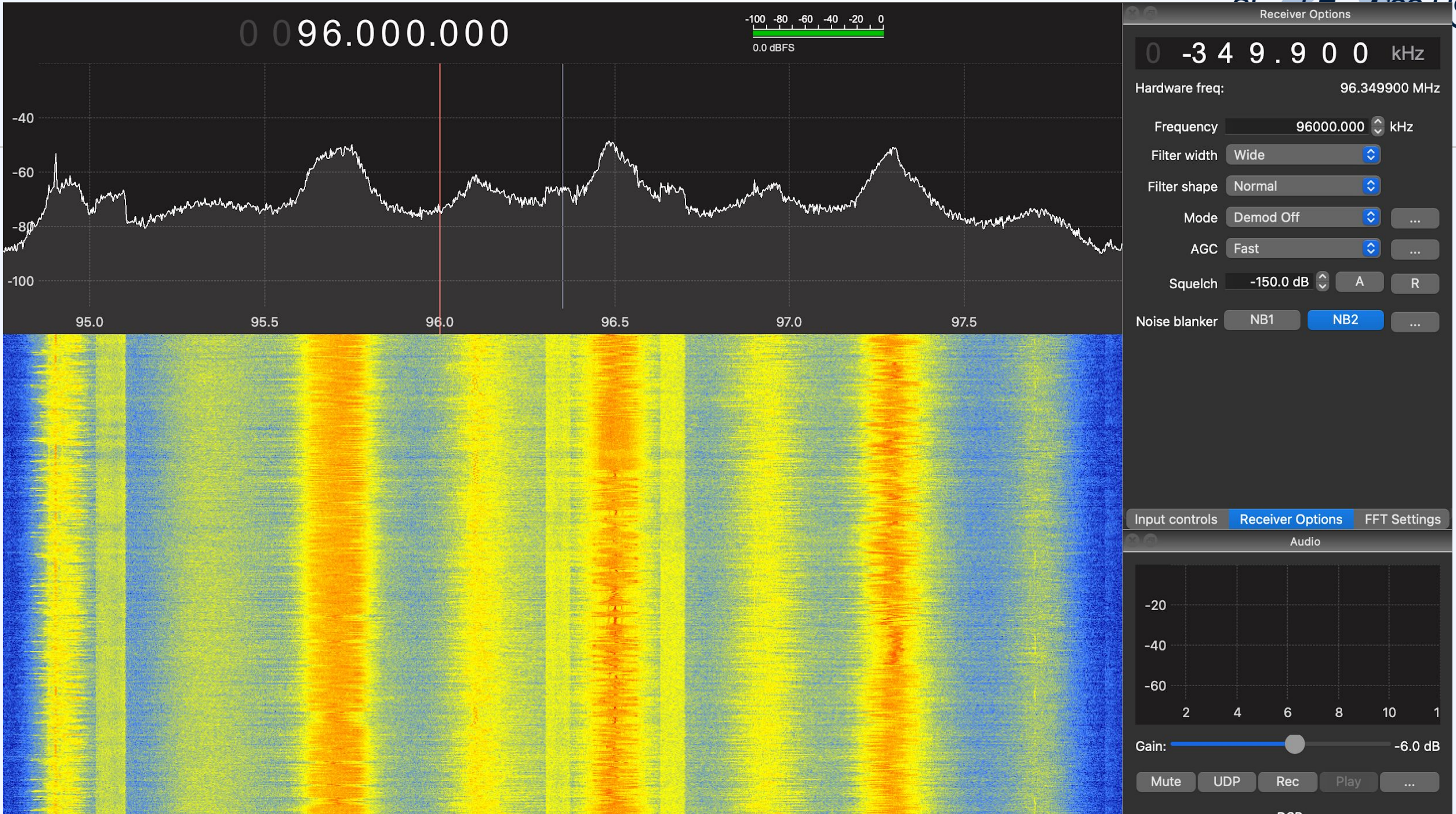
```
Tuned to 433.920MHz.
baseband_demod_FM: low pass filter for 250000 Hz at cutoff 25000 Hz, 40.0 us
-------------------------------------------------------------------------------
time      : 2022-07-11 11:06:05
model     : Schrader      type       : TPMS            flags      : 07
ID        : 0BBB92D
Pressure  : 287.5 kPa     Temperature: 48 C            Integrity  : CRC
-------------------------------------------------------------------------------
time      : 2022-07-11 11:06:07
model     : Schrader      type       : TPMS            flags      : 07
ID        : 0BBB91E
Pressure  : 270.0 kPa     Temperature: 53 C            Integrity  : CRC
```

Identifying signals

- GQRX and other waterfall tools
- Can look for transmission bursts
- Can begin to guess the type of signal

FM Radio

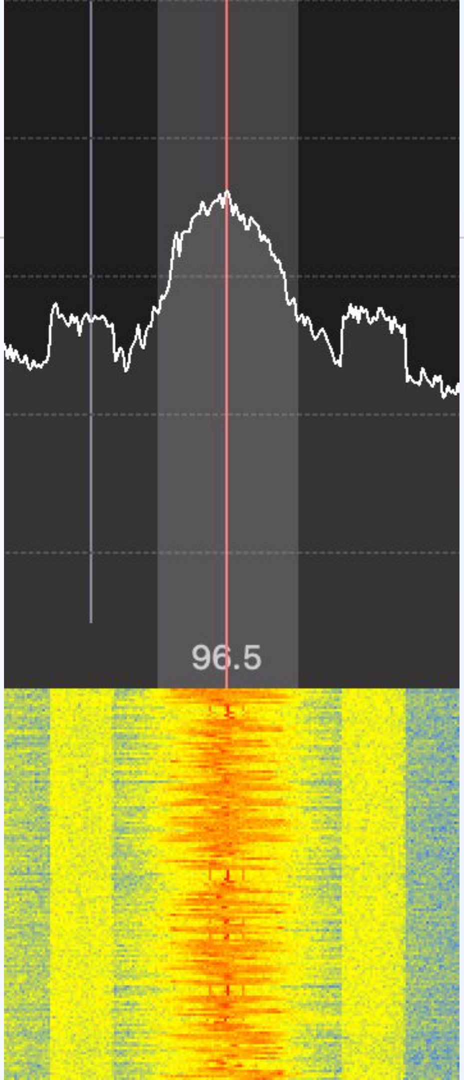
- Everyone's favorite first thing to look at
- Almost any SDR can see it
- It's super loud and obvious



0 096.000.000
-100 -80 -60 -40 -20 0
0.0 dBFS
-40
-60
-80
-100
95.0
95.5
96.0
96.5
97.0
97.5
Receiver Options
0 -3 4 9 . 9 0 0 kHz
Hardware freq:
96.349900 MHz
Frequency
96000.000
kHz
Filter width
Wide
Filter shape
Normal
Mode
Demod Off
...
AGC
Fast
...
Squelch
-150.0 dB
A
R
Noise blanker
NB1
NB2
...
Input controls
Receiver Options
FFT Settings
Audio
-20
-40
-60
2
4
6
8
10
Gain:
-6.0 dB
Mute
UDP
Rec
Play
...

Extra fun in FM

- FM waveform in the center
- Can see the audio wobble
- Notice the weird shoulders?
- Those are the digital sidebands
- HD FM radio + Weather + Traffic + song identifiers



So we found a signal…

- So we scrolled around with a tool and found our target
- Lets say we found something from a wireless thermometer
- Now we get packets, yeah?
- Not so fast…

What we get from the SDR

- Almost all SDR reports signal as “IQ”
- Complex number with real and imaginary components
- Forms sine waves with different amplitude and phase
- Deep dive into IQ would be a week-long event
- TL;DR it lets us model amplitude, frequency, and phase of a signal

Sampling rates

- How many IQ samples per second the radio can deliver to us
- Similar to audio fidelity – more samples is more detail
- How many do you need?
- "It depends"
- At least 2x the transition speed of the signal

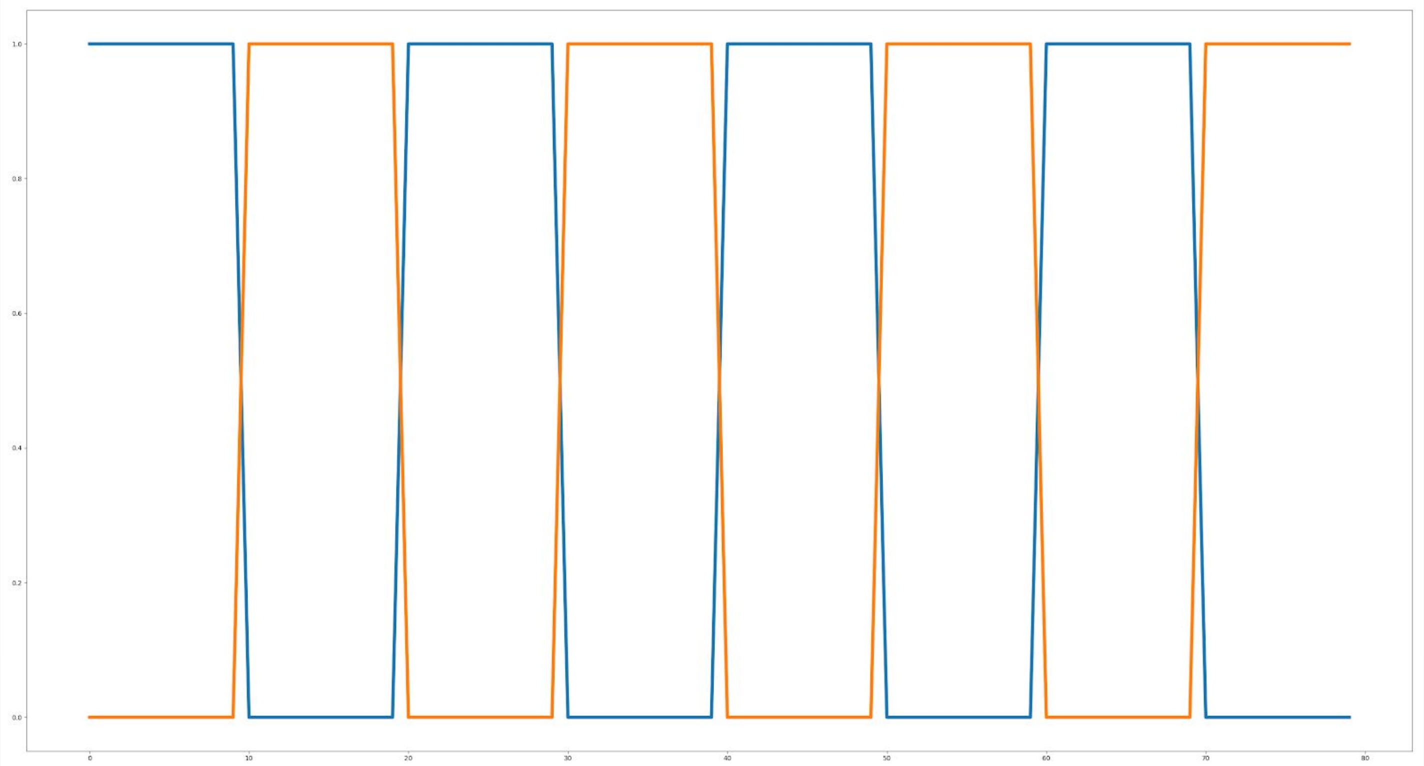
2x = Nyquist

- "Nyquist Rate" is at least 2x the frequency of the signal
- 1mbit rate in the air? You need at least 2mbit sample rate!
- More samples = more fidelity
- … but more samples = more CPU, RAM, network and storage…

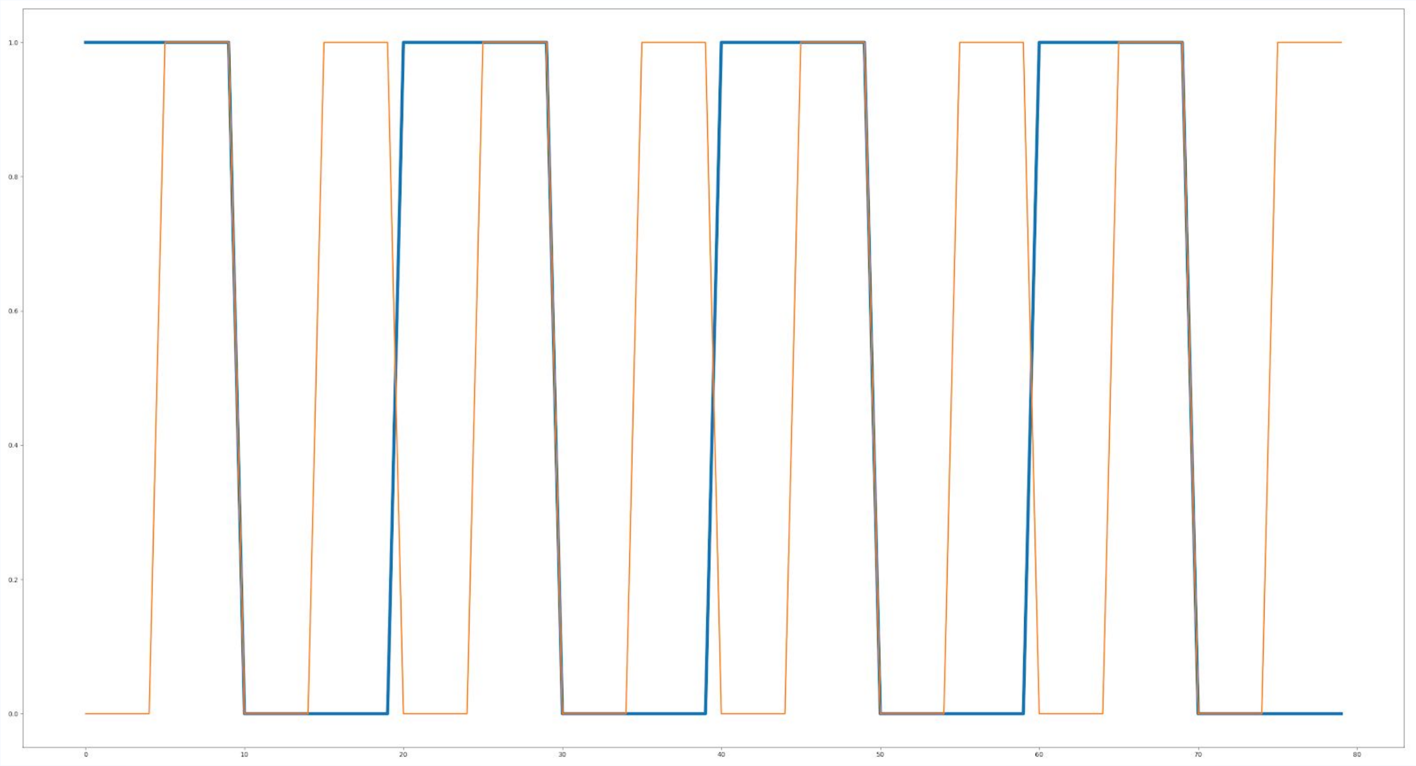
But why?

- Why can't we just sample at the rate we need?
- If the "beat" of the sample matches the signal, you either capture it perfectly, or not at all!
- … And you don't know which!

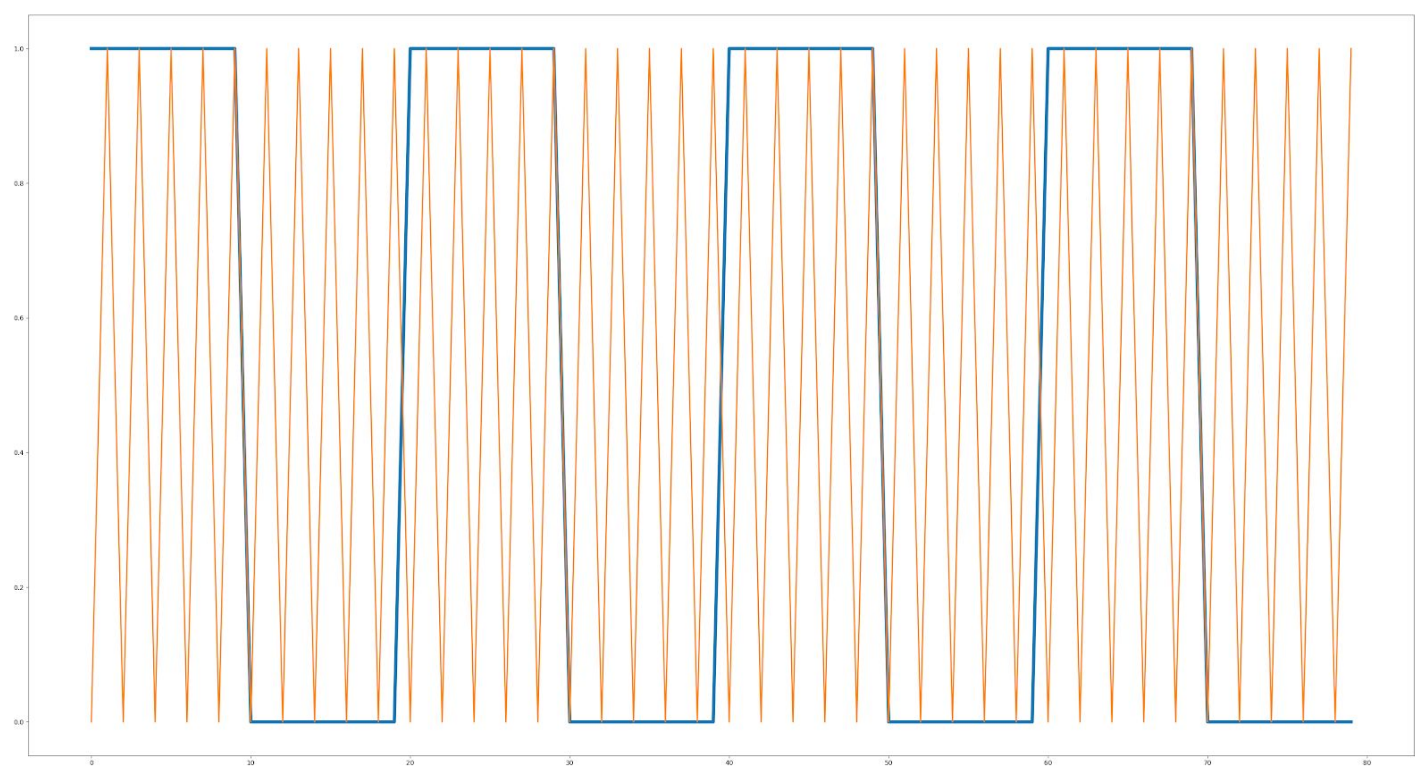
1x sample mismatch



2x sample



10x sample



Picking a sample rate

- "It depends"
- Often a SDR will support specific fixed sample rates, pick the closest
- 2x your signal minimum but also the least you can get away with

We’re all Layer 1 down here, Billy

- The OSI 7 layer model? Forget it.
- We’re going to be dealing with the physical layer almost exclusively
- Many of these protocols don’t even HAVE a MAC layer equivalent!
- Access control what’s this I don’t even
- So, lets define some terms…

Oh no we're starting with *bits*

- A bit is a 0 or a 1. Sure.
- Unfortunately, this is about to get a lot more confusing.
- When talking about signals, we have both the bit in the air, *but also* the bit of data encoded in the protocol!
- They're not the same!

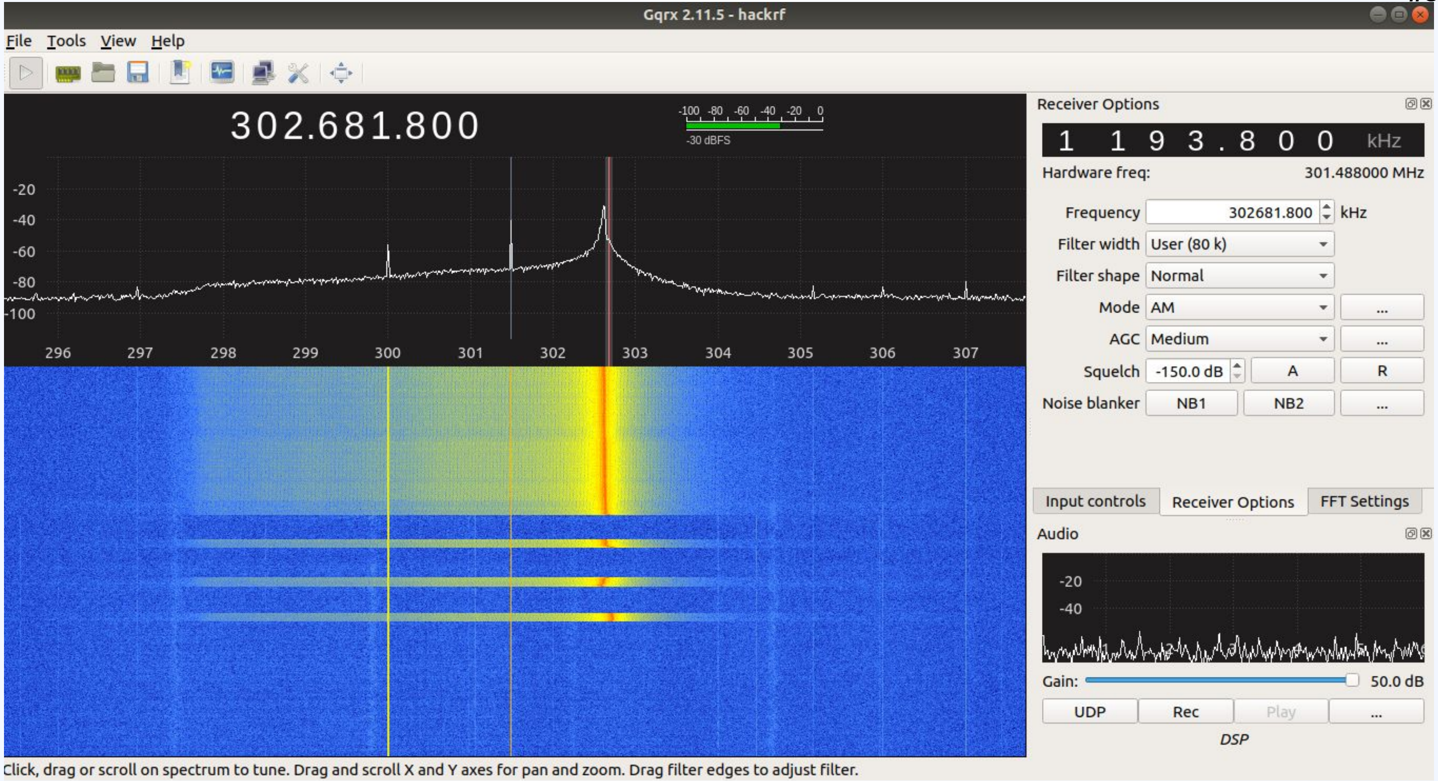
Encoding a bit in the air

- Easy
  - OOK (on/off like morse code)
  - ASK (amplitude – louder is 1, softer is 0)
- Harder
  - FSK (frequency shift to indicate 0 or 1)
  - PSK (phase shift to indicate 0 or 1)
- F-Off Black Magic
  - QAM, OFDM, *n*-PSK

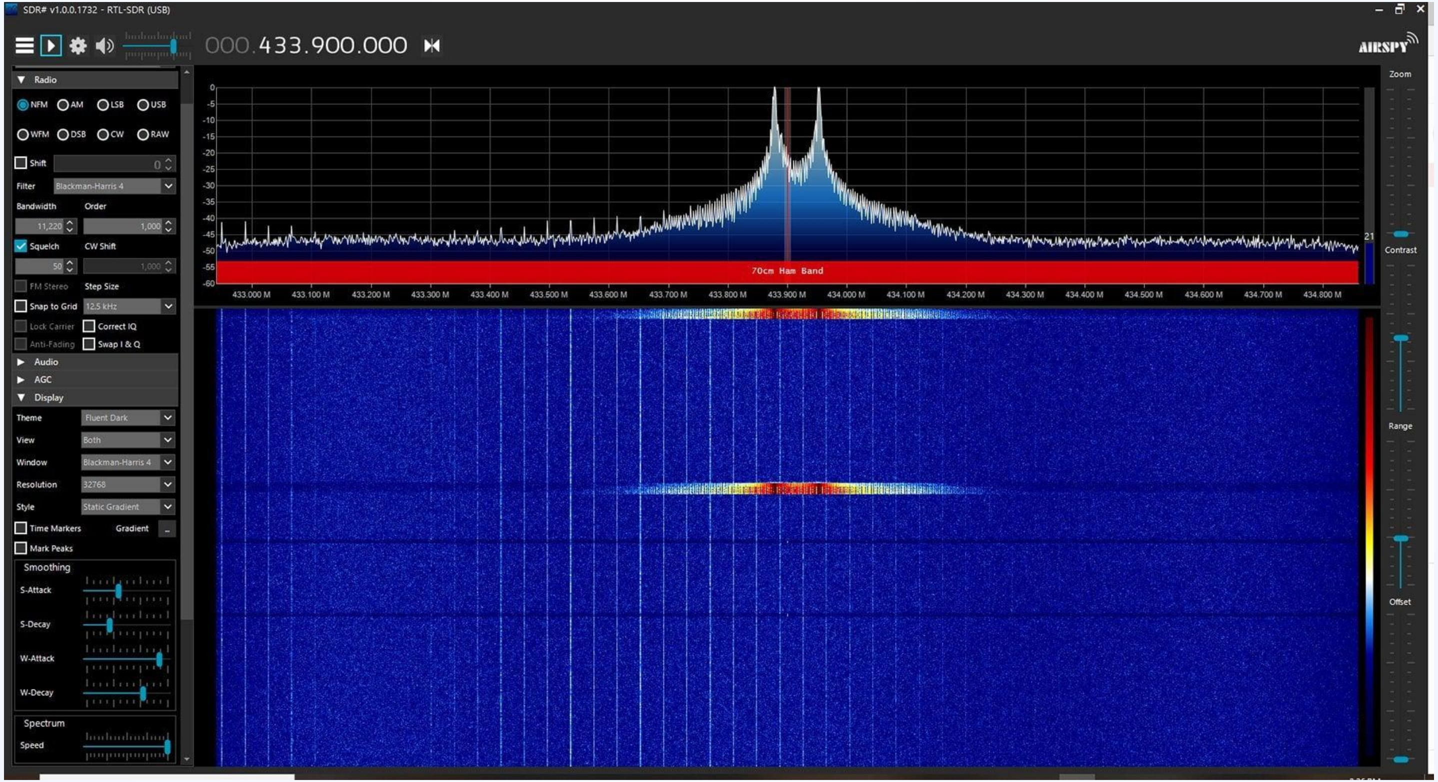
SharkFest'22 US
Kansas City, MO
July 9-14

OOK - Notice the clear on/off

#sf22us

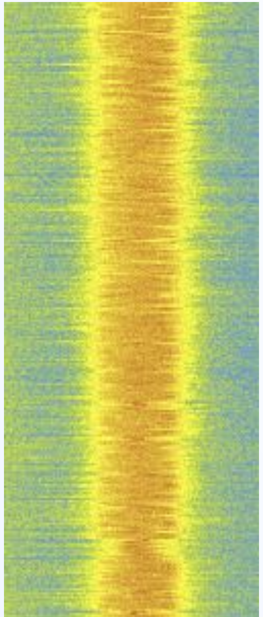


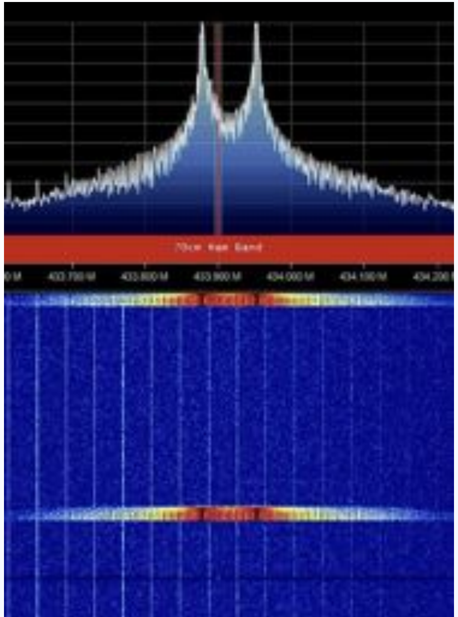
FSK - Notice the two horns



FSK vs FM

Digital vs Analog





Variants

- For every type of encoding there are variants
- Different bandwidth (amount of frequency) used
- Different data rates
- But the same basic schemes

Mixed Variants

- Advanced protocols can combine multiple methods
- Encode on frequency + phase to get 2 states per cycle
- Encode on multiple frequencies to get more states per cycle

Symbols

- When talking about what's being sent at the *radio* layer, we need to think in symbols
- A symbol can encode one – or many – bits
- A symbol can require multiple transitions (ie "bits" in the air) to be encoded
- So to send a '1' bit of logical data, you may need to *transmit* '1101'

Symbol encoding

- Transmission (FSK/ASK/OOK) tells us how to differentiate between states
- But how do we know what a single bit is?
- We don’t know when the other end started transmitting, or if we’ve even seen all of the transmission.

Repeating bits

- '1111'. Is that four '1' bits? Is that two '1' bits but we read them too slow?
- '111'. Is that three '1's? Or two and we were slightly off? Or 4 and we missed one?
- Encoding methods exist to help solve this…
- But encoding requires us to transmit more *in-air* bits to get one *logical* bit!

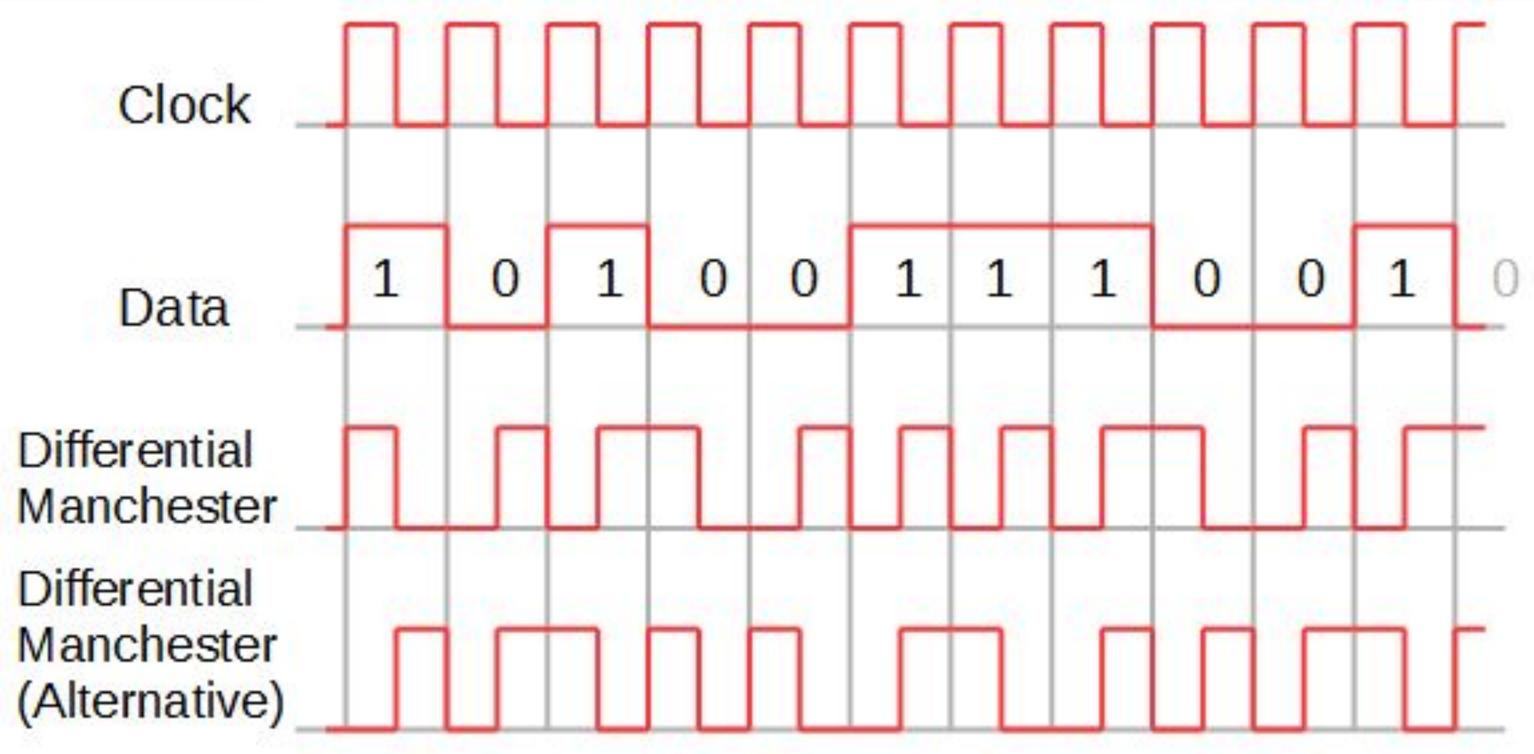
Solving encoding

- Lots of ways to solve encoding
- All based on making very clear what the content is
- Many options
  - NRZ
  - Manchester
  - More

Manchester

- Manchester encoding is a popular method used by many simple devices
- A zero is always "low-high" (or reversed)
- A one is always "high-low" (or reversed)
- So a logical '1111' would be '10101010'
- Now we're less ambiguous, but it takes twice as long!

Manchester


Clock
Data
1 0 1 0 0 1 1 1 0 0 1 0
Differential Manchester
Differential Manchester (Alternative)

Sidebar: Lies!

- Ever wonder why you only get about 1/3 the speed claimed on WiFi?
- This is part of why!
- Marketing brags about the rate in the air – not the rate of symbols!
- Lies, and speed graphs of also lies!

Frequency & Bandwidth

- Transmission frequency is where in the spectrum it lives
- Basic WiFi starts at 2.4GHz or 2400MHz for instance
- Bandwidth is how much of the frequency it uses
- Basic WiFi uses 20MHz, so it could go for instance from 2400MHz to 2420MHz
- Embedded sensors tend to use bandwidth in the KHz

Preamble

- Present in most protocols, including Ethernet and WiFi, you just can't see it
- Indicates a packet is incoming & helps determine speed
- Usually 10101010 or 01010101, may repeat multiple times (0xAAAA, 0x5555)
- Helps us know we've found something; we're going to look for this first!

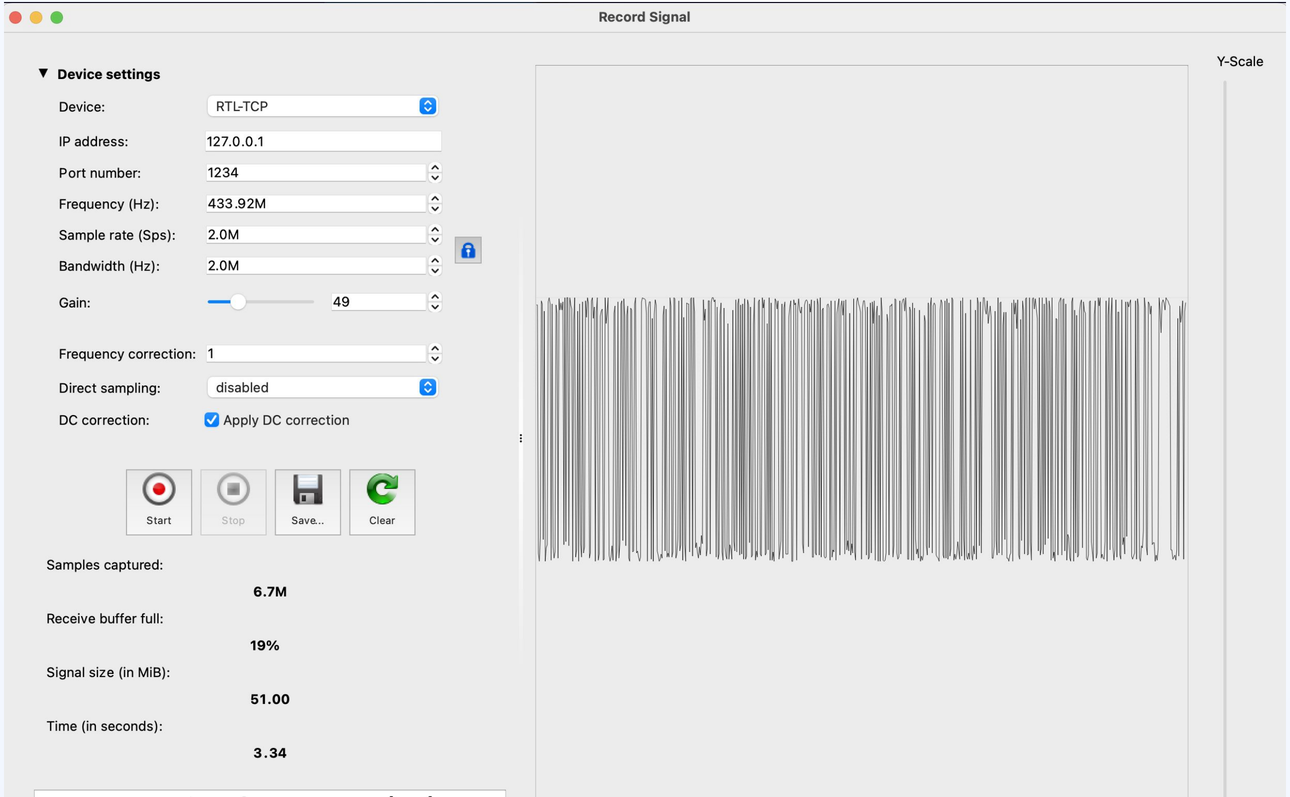
Digital Signal Processing

- Most of the time decoding a protocol will be spent doing filters & demodulation
- We can pick filter performance as we need:
  - Filter high frequency noise
  - Filter low frequency noise
  - Perform basic averaging
  - Filter amplitude noise, frequency noise, etc
- We can explore these w/ cool tools like URH

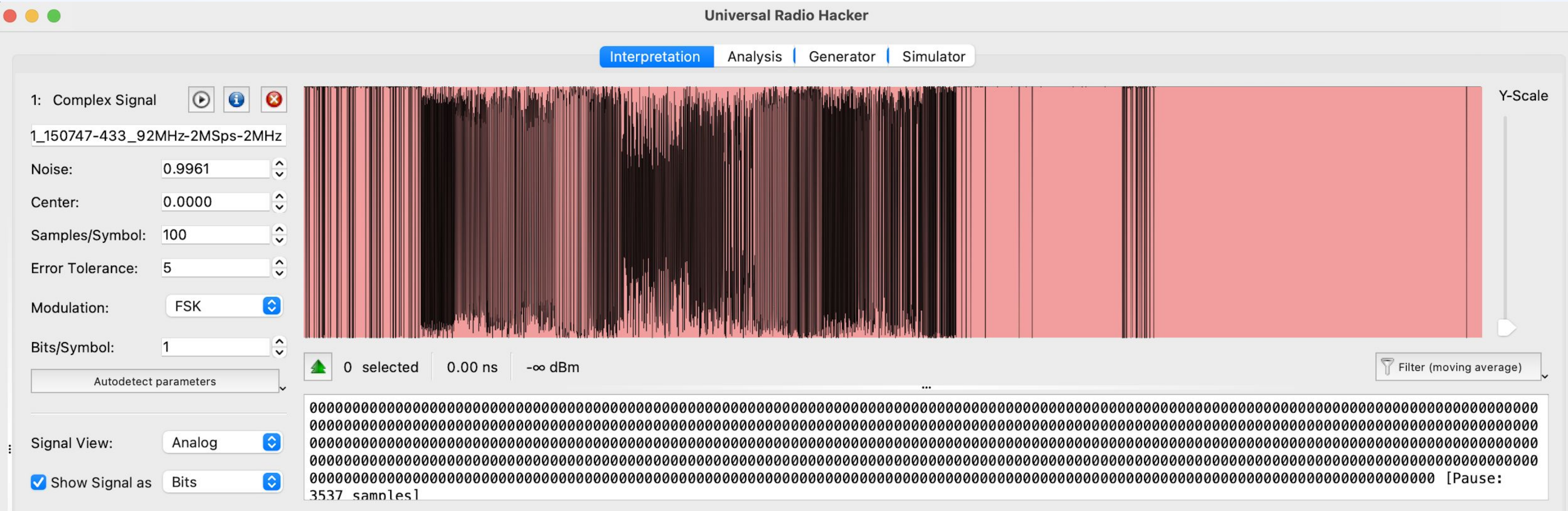
Universal Radio Hacker

- GQRX & friends for finding frequency
- URH for capturing & decode
- Runs on multiple platforms
- Captures directly to a UI for processing signal, applying filters, and comparing decoded data

Record a signal



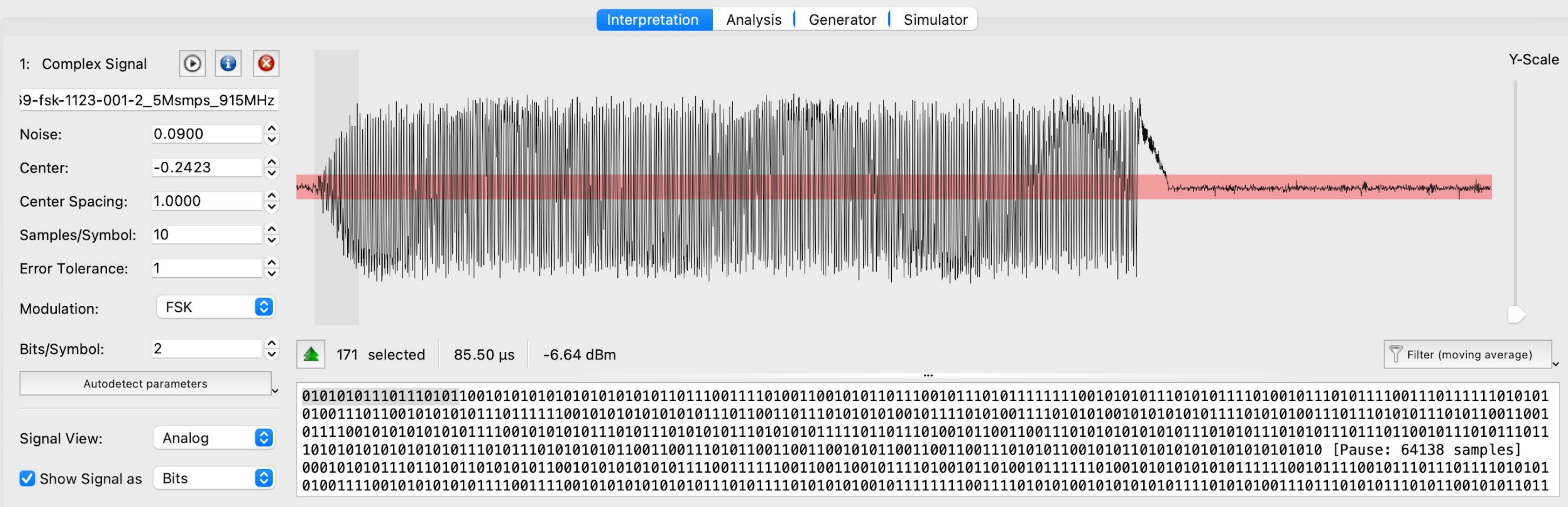
Zooming in - sure looks like *something*



Auto-parameters

- URH is amazing
- Select the signal in the graph
- Crop to selection
- Try an encoding (like FSK)
- Click “auto detect”

Pulling some data

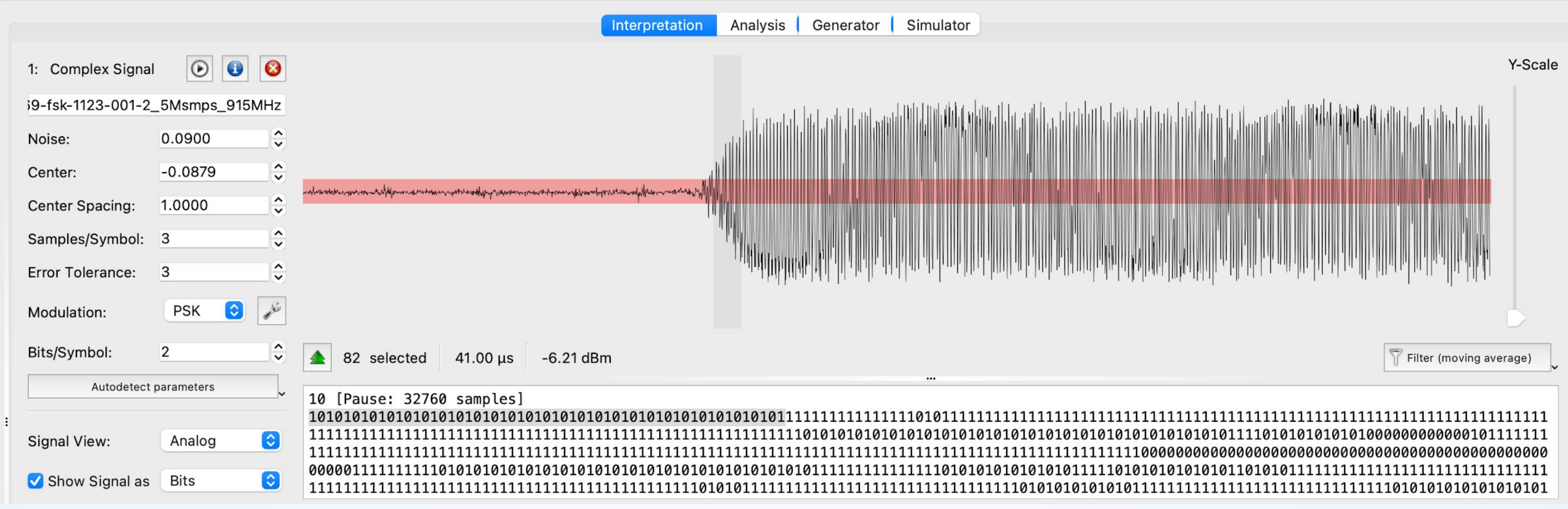


SharkFest'22 US
Kansas City, MO
July 9-14

#sf22us

Looking at the signal itself

- Sure is a still pile of garbage



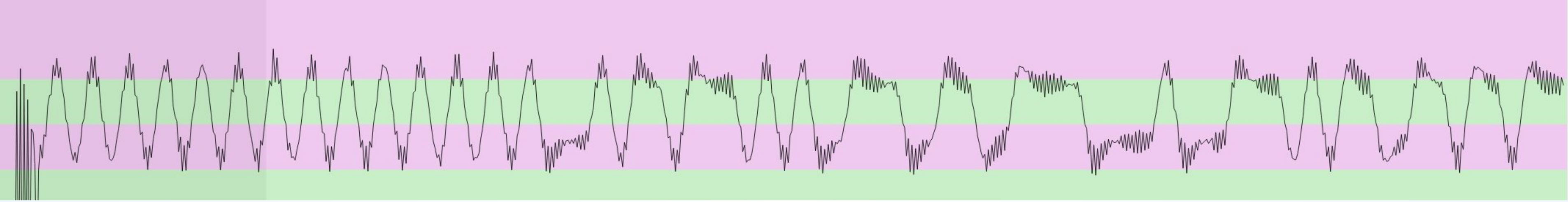
Demod view

- URH can demod multiple basic encodings
- ASK, OOK, FSK
- Flip through them and see if any make the signal make sense
- Most embedded devices use one of these

Switch to 'demod' view

- Apply the FSK demod and suddenly

Why is this interesting?



- Remember manchester encoding?
- Down-Up for 1, Up-Down for 0 (or reversed)
- Seeing groupings of single and double transitions is very promising

SharkFest'22 US
Kansas City, MO
July 9-14

#sf22us

Increased filtering

- Apply a filter right in the URH UI
- Suddenly it's much more clean looking!

Now we have bits

- Using URH we know the encoding type and data rates, because we got a preamble out
- So we can just read data, right?
- Welllll.....

Of course it's not that simple

- Additional randomization for TX
- Also no reason text is ASCII
- No reason data is even 8 bits!
- Many protocols are transmitted from tiny embedded devices
- Optimized for other reasons

Real-world encoding in ADSB

- 13 bit altitude. In feet. Made of non-contiguous bits.
- Also a 12 bit altitude. Non-contiguous bits. Multiplied by 25. Subtract 1000.
- Non-ASCII non-contiguous alphabet
- Helps to have external knowledge of the protocols

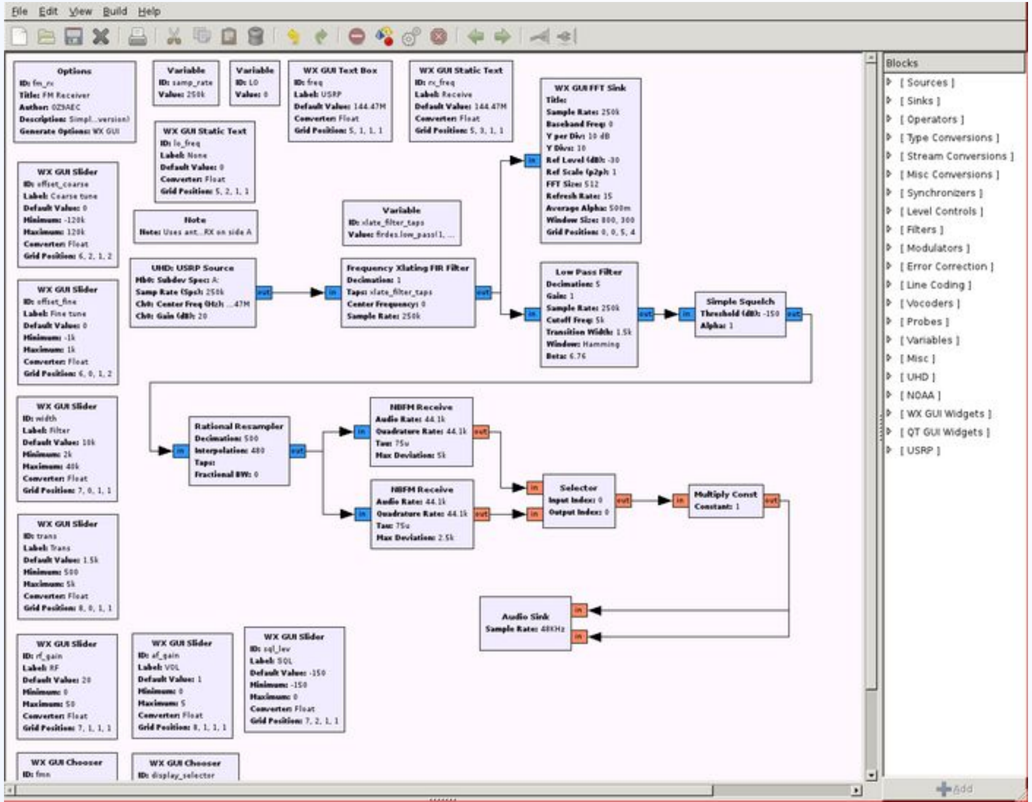
Practical decoding

- So we've done some exploring
- We have a protocol we know we want
- URH can export it, but that's not really great for long-term processing
- How can we turn this into a tool we can integrate?

GRC

- Gnu Radio Companion
- Lego modules that let us put together a decode
- Good for deeper exploration, not great for automatic tools

GRC


File Edit View Build Help
Options
Variable
WX GUI Text Box
WX GUI Static Text
WX GUI FFT Sink
WX GUI Slider
Note
UHD: USRP Source
Frequency Xlating FIR Filter
Low Pass Filter
Simple Squelch
Rational Resampler
NBFM Receive
Selector
Multiply Const
Audio Sink
WX GUI Chooser
Blocks
[Sources]
[Sinks]
[Operators]
[Type Conversions]
[Stream Conversions]
[Misc Conversions]
[Synchronizers]
[Level Controls]
[Filters]
[Modulators]
[Error Correction]
[Line Coding]
[Vocoders]
[Probes]
[Variables]
[Misc]
[UHD]
[NOAA]
[WX GUI Widgets]
[QT GUI Widgets]
[USRP]
Add

Python

- Able to talk to radio drivers
- Python + NumPy + SciPy gives us programmatic manipulations
- Trickier to get the speed we need

C/C++

- Harder to manipulate arbitrary byte streams
- Some good libraries for DSP
- Can be faster

Basic process

- We'll go with Python because it's the most readable
- NumPy is a high-speed (native processor speed) processing library we'll use heavily
- Lots of numerical & scientific algos available
- 1000 foot view only, still boring, sorry!

Pythonisms

- Python slices and iterates are hugely helpful here
- Array[start:end:step]
- Array[::*n*] skips by N entries; you'll see us use this a lot in the examples
- Use NumPy whenever possible for speed!

Basic steps

- Acquire data
- Convert IQ
- Apply filters
- Turn values into bits
- Find preamble
- Turn bits into our data

Acquiring data

- Librtlsdr, soapy, etc have APIs
- Can also read saved files from other tools
- Cython / CFFI lets us talk right to the USB drivers w/ minimal pain
- No matter how we import data, we end up with an array of interleaved IQ data

Converting IQ

- Usually we need to convert the IQ imaginary into real; depends on encoding
- Adding the square of I and Q gets us amplitude
- Tricks for speed like precomputed squares

```
 buf = np.add(self.square_lut[buf[::2]],
self.square_lut[buf[1::2]])
```

Adding squares

```
buf =
np.add(self.square_lut[buf[::2]],
self.square_lut[buf[1::2]])
```

Add the square of each I and Q together.

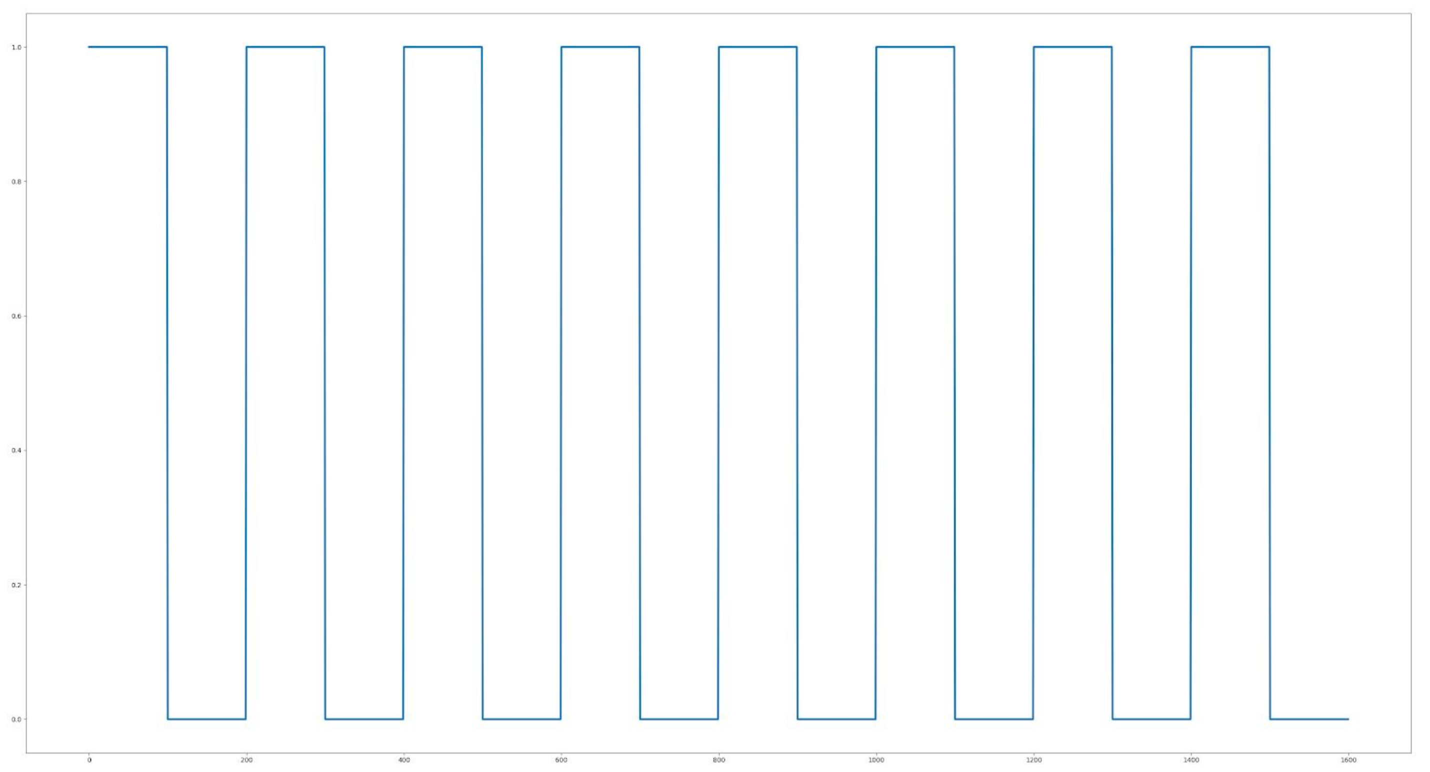
We pre-computed every 8bit square!

Remember Python slices and steps?

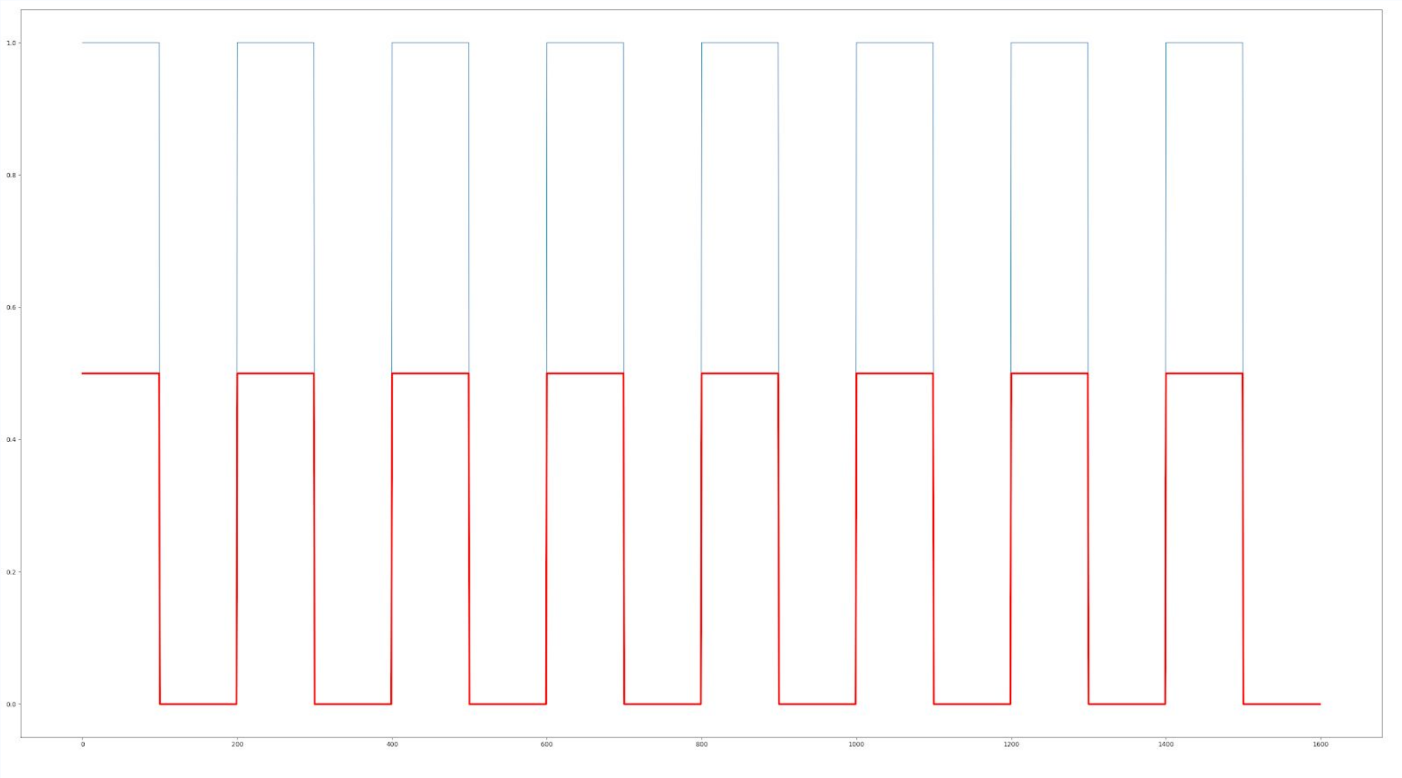
Turning it into binary

- So what does it take to turn an analog signal into binary?
- Lets take a basic on/off (OOK) waveform
- Analog data so we’re -128 to +128 (8 bit)
- Let’s assume anything > 50% is “on” and < 50% is “off”

Signal, in theory



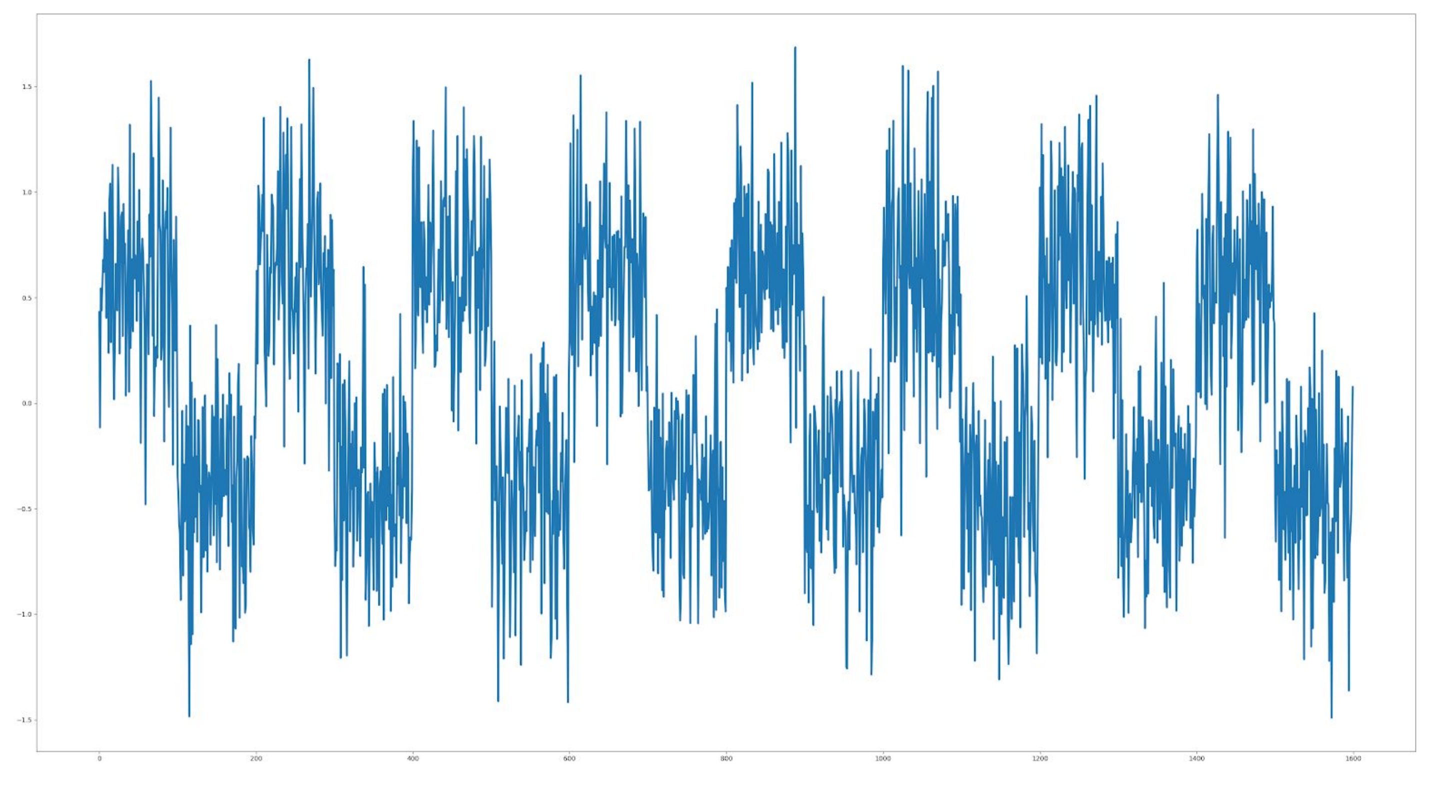
Extracting bits by signal level



Looks pretty good

- That looks pretty good; we get a nice representation

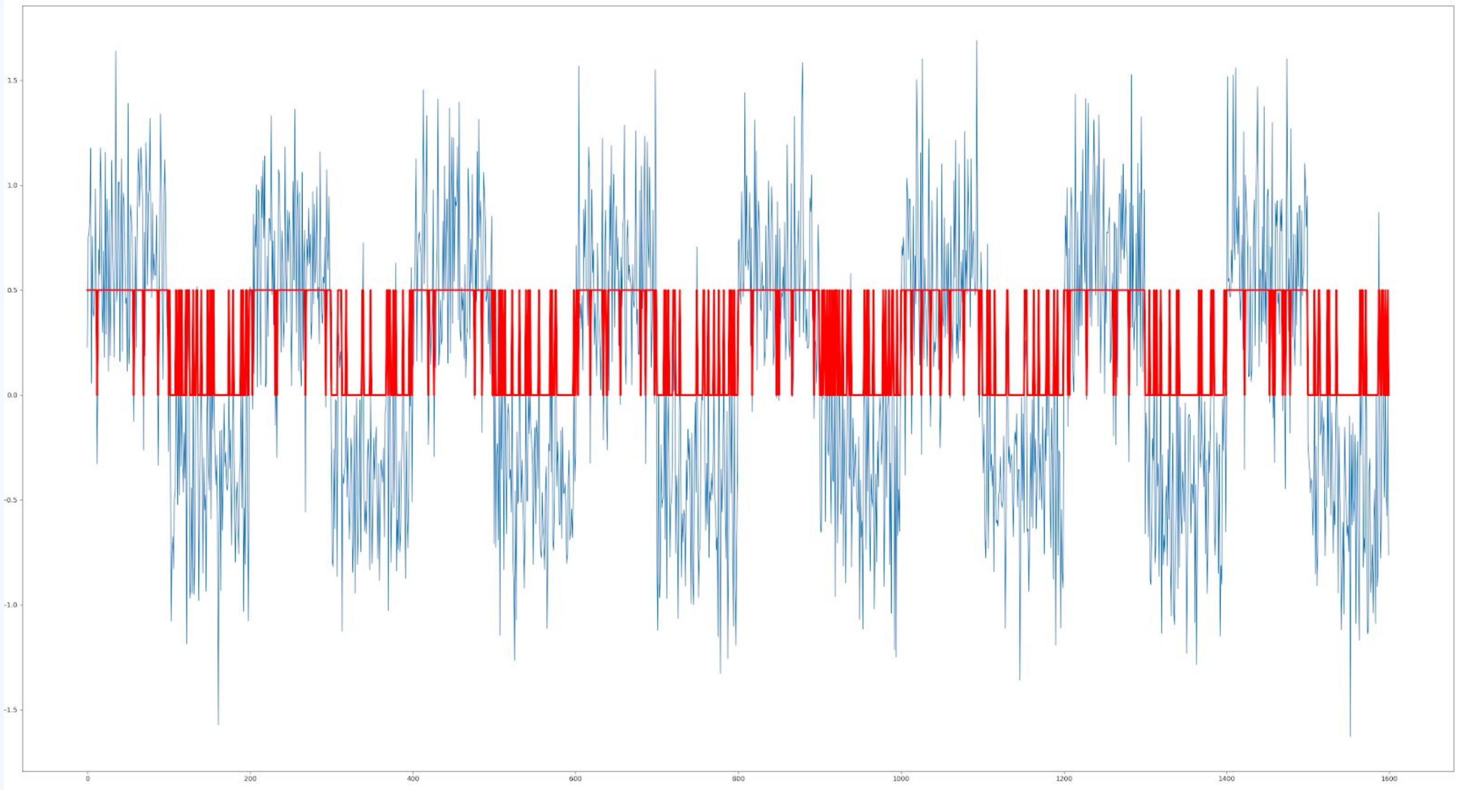
Signal, in reality



SharkFest'22 US
Kansas City, MO
July 9-14

Now we get garbage

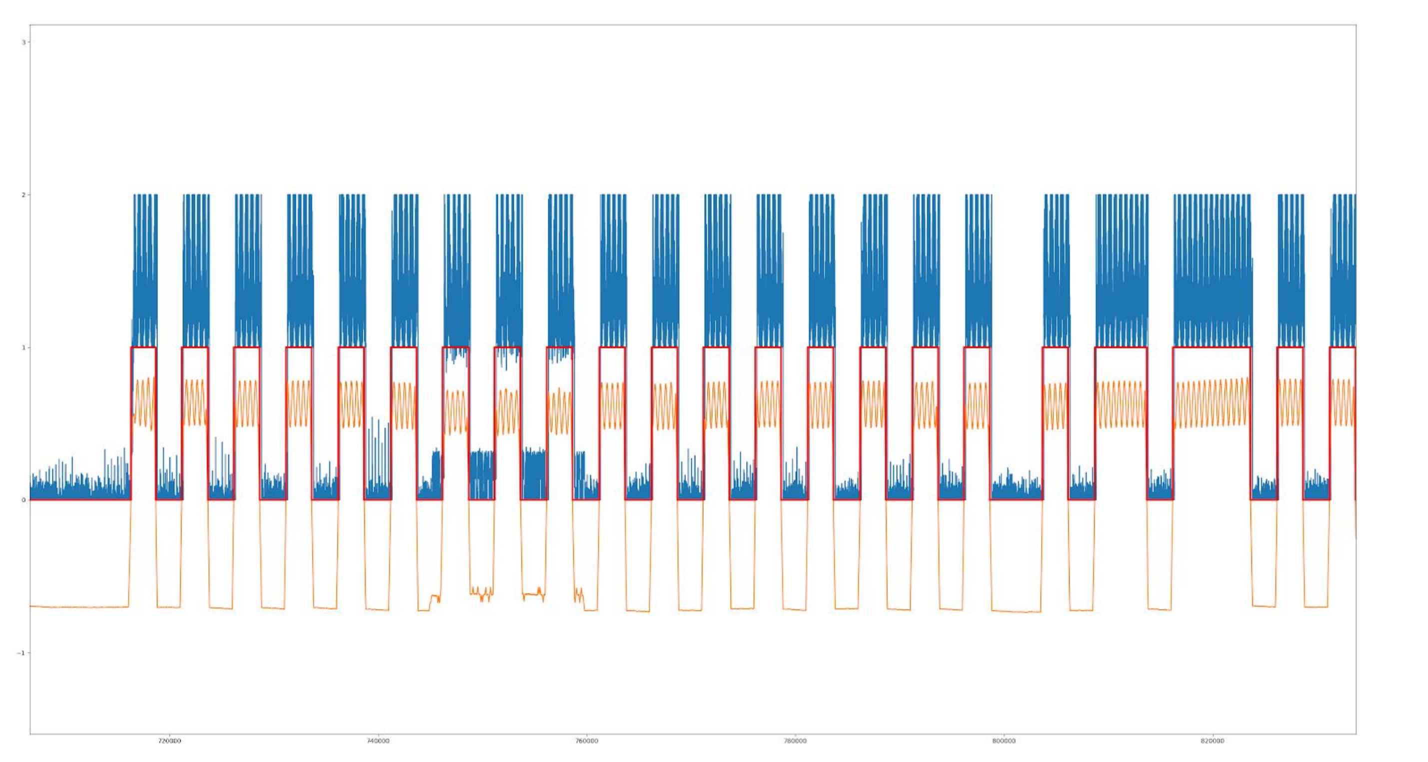
#sf22us



Bleh

- Reality is noisy
- Why?
- Wrong antenna freq can round edges
- Cheap TX HW is often crummy
- Other signals can overlap
- SDR aliasing

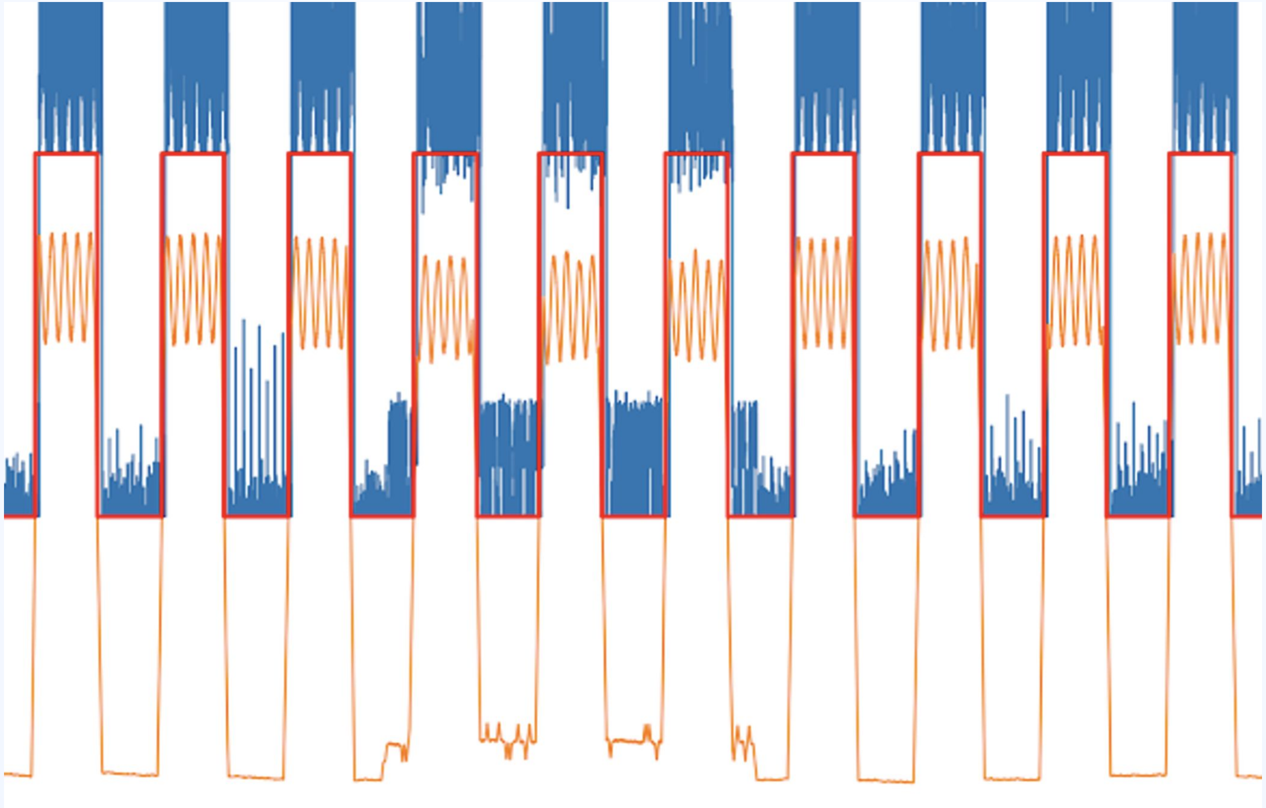
Other signal in the middle of our packet



SharkFest'22 US
Kansas City, MO
July 9-14

Hey there buddy

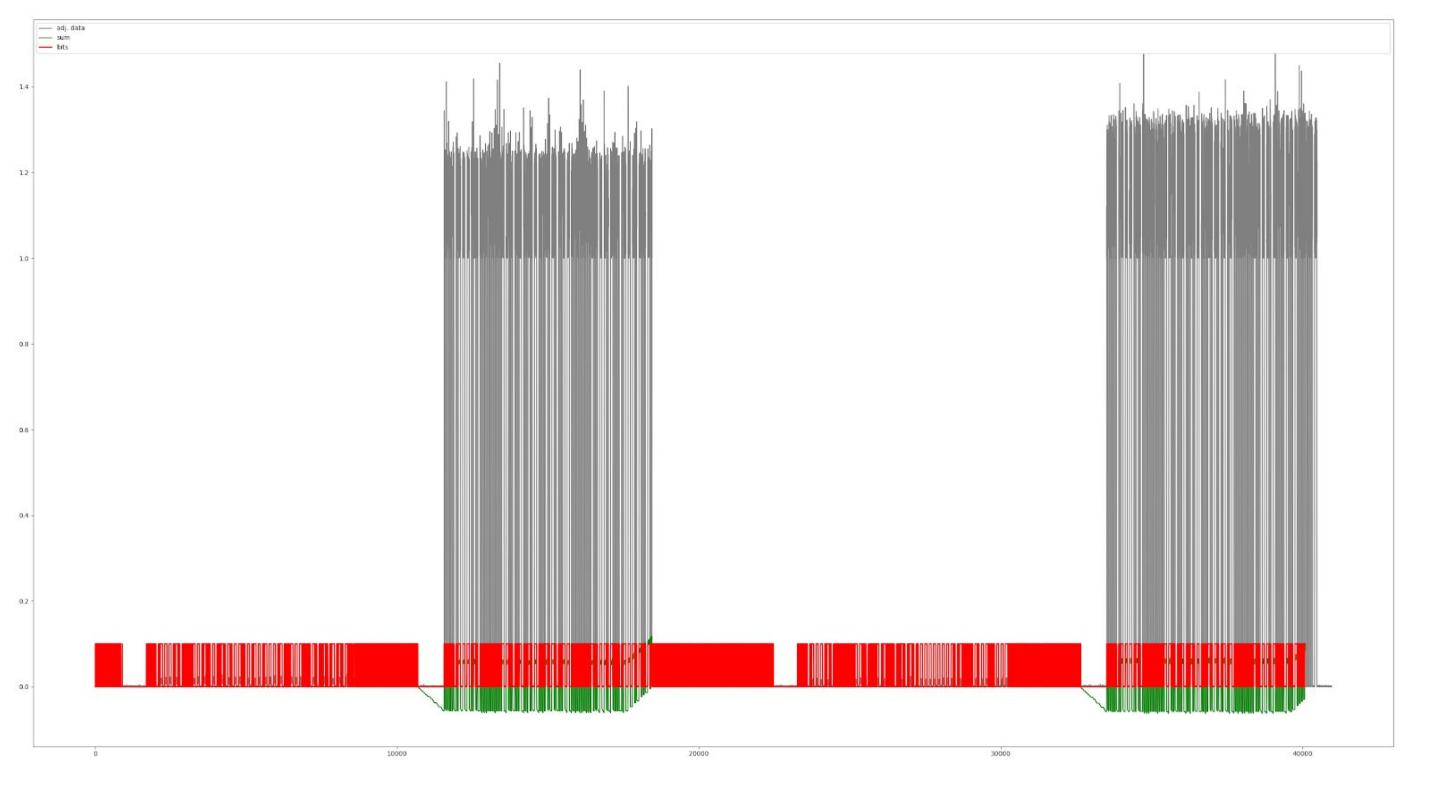
#sf22us



SharkFest'22 US
Kansas City, MO
July 9-14

Wildly different signal levels

#sf22us



So what can we do?

- We want to turn wobbly fuzzy analog into more readable trends
- We want to ignore noise as best we can
- We do this by applying filters
- As many filter variants as you can imagine
- Welcome to DSP – digital signal processing

Filtering data

- Many filters to pick from
- We'll look at moving average
- Other more advanced filter techniques available
- Links at end for more…

Simple as simple

- Simplest filter may be running average
- Sliding window average using NumPy cumulative sums:

```
def cumsum(data, wndo):
    ret = np.cumsum(data)
    ret[wndo:] = ret[wndo:] - ret[:-wndo]
    return ret[wndo - 1:] / wndo
```

Moving Average

- ret = np.cumsum(data)
  Adds each element to the previous

```
>>> numpy.cumsum([0, 1, 2, 3, 4])
array([ 0,  1,  3,  6, 10])
```

Moving Average

- **ret[wndo:] = ret[wndo:] - ret[:-wndo]**

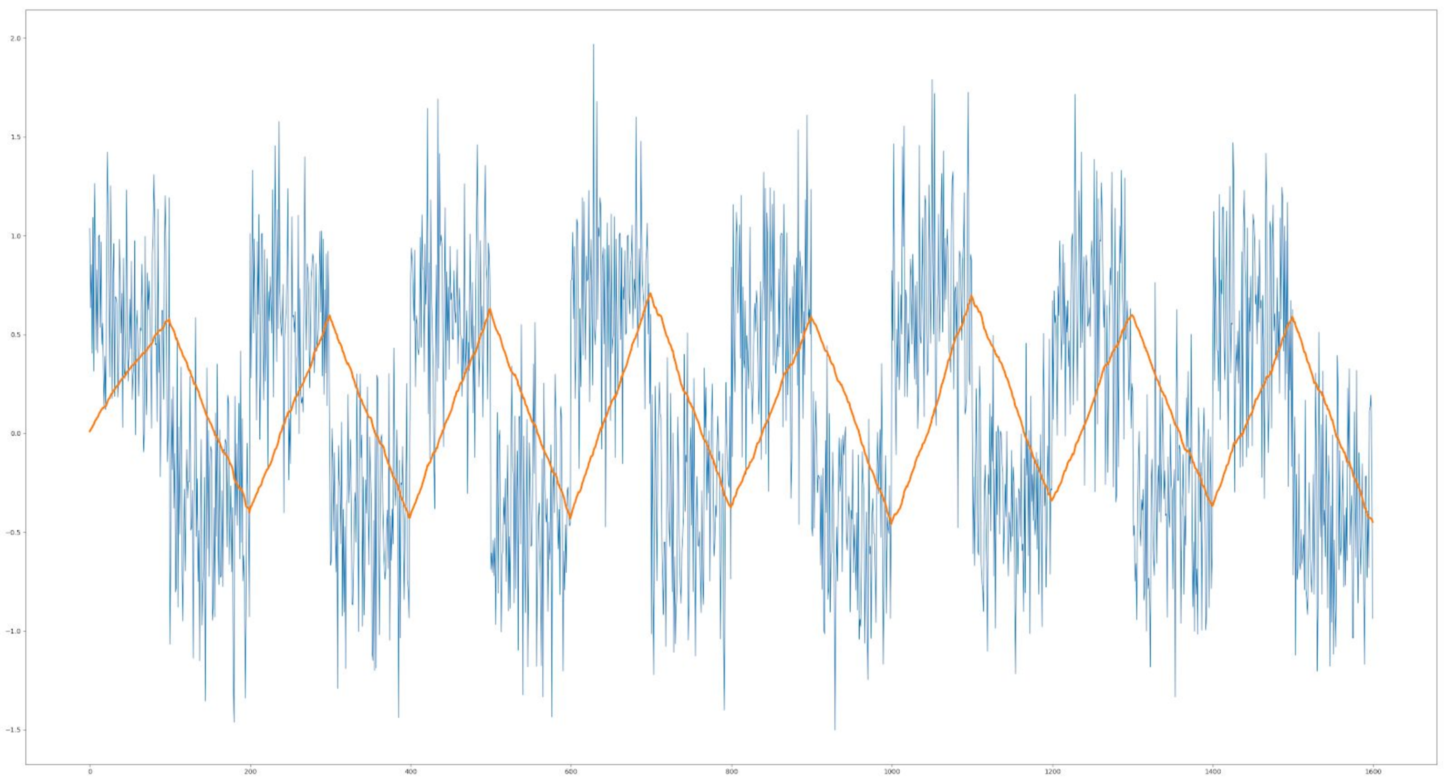
  Slice the first and last "window" off the data (more on windows soon)

Moving Average

- **return ret[wndo - 1:] / wndo**

Resample the data by average – dividing by the number of samples in each window

End result of the moving average

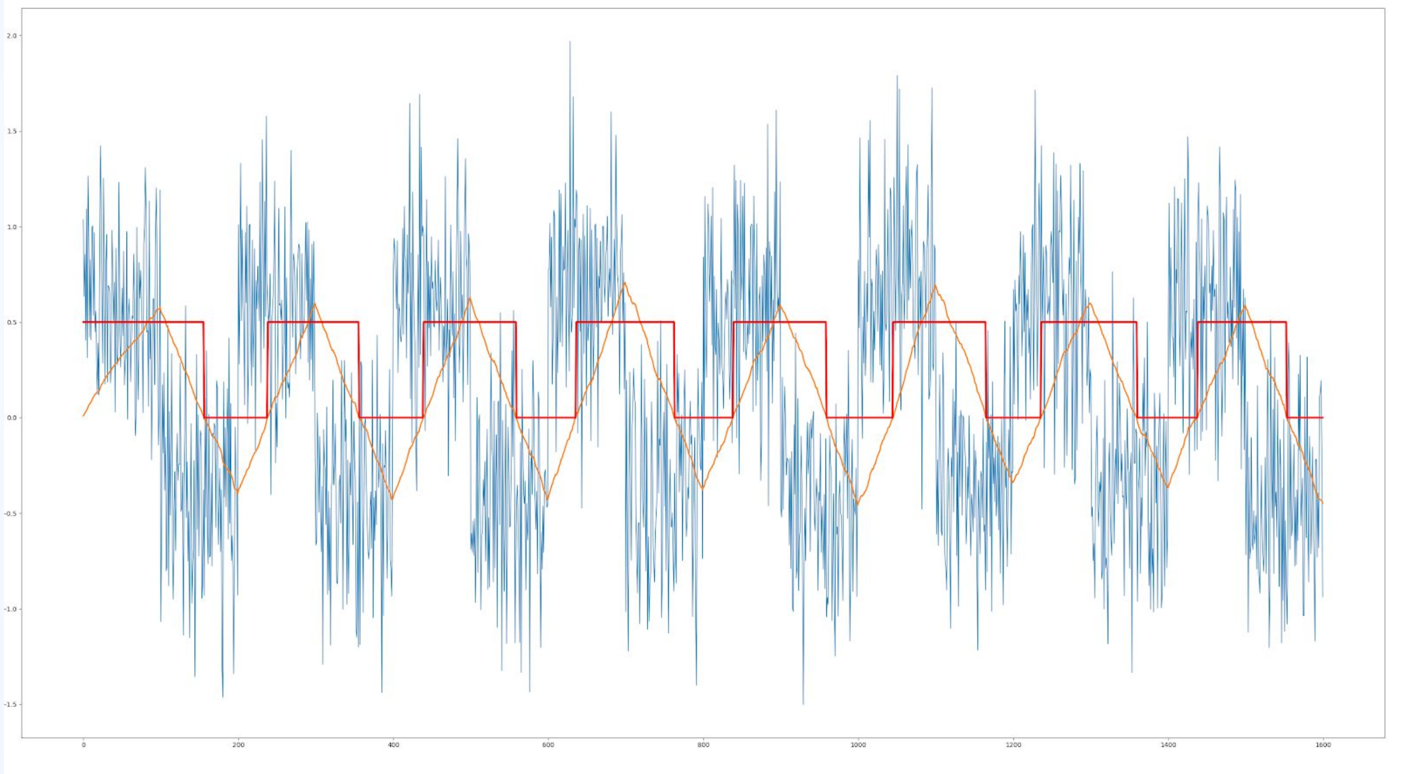


Turning analog into binary

- After filtering, converting is simple

```
bits = np.where(buffer > 0, 1, 0)
```

Converting to bits looks a lot better!



Decimating our signal

- Decimation: Reducing the # of samples
- Yes, I know it's not 10%
- Used to reduce our over-sampling due to Nyquist + radio sample rates
- Right now our signal is stretched by whatever the multiple of sample rate to symbol rate is
- In this code, it's a multiple of 24

```
bits = bits[::self.decimation]
```

Finding the preamble

- We need to find our preamble
- Either we can do a naïve search for the preamble as a sliding window…
- Or we use some NumPy functions to do correlation to find likely matches, then do an exact match
- If we find something, we look deeper

Correlation

```
self.scm_preamble = np.array([1, 0, 1, 0, 1, 0, 1, 0,
1,])
…
corr = np.correlate(buf, self.search_preamble)
return np.argmax(corr)
```

Search the buffer for something that looks like 10101010

Return the most confident match (highest value)

Likely the start of our packet!

Start decoding the packet

- We have a stream of bits from the radio
- We think we have the start of a packet
- Apply the symbol decode (ie, Manchester, etc)

Manchester in Python

```
def _single_manchester(self, a, b, c, d):
       bit_p = a > b
       bit = c > d
       if bit and ((bit_p and c > b) or (not bit_p and d < b)):
           return 1
       if not bit and ((bit_p and d > b) or (bit_p and c < b)):
           return 0
```

Checksums

- Almost every packet has a checksum
- Radio transmission is a garbage fire
- Hopefully, we know how to validate the checksum!
- Brute force checksum tools online
- If you're feeling spicy, you can try to auto-correct missing bits and try again

Processing the packet content

- We finally have a stream of bytes we think is our packet
- Now we just need to know *what goes into that packet*
- So that's fun.
- Just remember words may not be 8 bits
- Text may not be ASCII

Example: Meters

```
[  0 : 21 ] 21 Sync / RF Preamble 1F2A60
[ 21 : 23 ] 2  ID MSB
[ 23      ] 1  Reserved
[ 24 : 26 ] 2  Physical tamper
[ 26 : 30 ] 4  Endpoint type
[ 30 : 32 ] 2  Endpoint tamper
[ 32 : 56 ] 24 Consumption value
[ 56 : 80 ] 24 ID LSB
[ 80 : 96 ] 16 Checksum
```

Now what?

- Celebrate our victory
- Dump the data to somewhere we can use it
- If it has a DLT, we can write it to PCAP
- Otherwise may need custom decoders
- Tools like Kismet can talk arbitrary content
- JSON + MQTT?

Super-advanced SDR-foo

- FPGA enabled SDR
- BladeRF, USRP, some others
- If you can implement the full decode in FPGA, you offload all the work
- FPGA still power hungry
- Still expensive
- Rare skillset

FPGA Examples

- BladeRF WiPhy
  - Full 802.11n implementation in FPGA
  - Public license!
  - Inject *anything* with no firmware interference
- BladeRF ADSB
  - Parallel packet recovery tries thousands of CRC permutations

Transmission

- We haven't covered transmission at all
- "The same, but, backwards."
- You have to synthesize a binary stream that represents the signal
- URH can automate it for things
- Beyond today – rtl-sdr can't TX, and TX may require licensing, etc

S.L.O. - SDR-Like-Objects

- Yardstick One
- Ubertooth
- RFCat
- Others

Flexible ASICs

- Usually built on a TI-CC radio chip
- Microcontroller + Radio with multiple encodings
- Usually FSK and PSK and some others
- Often found in cheap consumer devices

RFCat

- RFCat is a python framework for talking to TI-CC
- If your protocol happens to be compatible with one in the TI-CC suite, you can use the ASIC
- Let it handle the filtering, decode, etc
- You just set the attributes and go

Why all this SDR stuff then?

- Sometimes you'll get lucky
- Often you won't
- Even if you do, you need to
  - Find the signal
  - Find the encoding
  - Find the data rates
- SDR and URH still have a huge role

Resources!

Universal Radio Hacker (github)

Kismet (github)

SDR with HackRF Series (youtube)

Rtl-sdr.com

Dspguide.com