Funny Microphones

RED TEAM

Alex Schoepf & Dane Cucinelli

Table of Contents

- Bios
- Advisory
- Tools
    - Software Defined Radios
    - GNU Radio
- Modulation Types
    - FM
        - Demodulation
        - Modulation
- Squelch
- Roadblocks
    - Nyquist-Shannon
    - Deviation
    - Fine Tuning
- Live Demo
- Other Dangerous Discoveries

Who Are We

Alex Schoepf

- Computer Engineering Major
- Technical Theatre Minor

- Previously K-Fest coordinator and Head of Technology @ WMTU 91.9 FM
- Member of RedTeam, Rozsa Center, SLS, LUG, HARC
- General class amateur radio operator (KE8YAW)

- Likes to figure out how things work

Dane Cucinelli

- Mechanical Engineering Technology

- Member of RedTeam, LUG, HARC, WMTU
- General class amateur radio operator (W8UPR)

- Limited work experience with AC power and RF

- Likes harmless chaos

- Shower connoisseur

Advisory: Do Not Try This at Home

Some things shown in this presentation are illegal. Do not attempt to replicate any demonstrations with the caution sign. We have the qualifications to understand the risks associated with such activities.

FCC fines are ***NOT CHEAP***
(the last one was $34,000)

GNU Radio Demo files will be available after the talk! (I need to fix some things). Please ask if you are unsure what is safe!

The best thing since sliced bread: SDRs

SDR

- Listens to a very wide range of RF at the same time
- Processing done by a general-purpose computer, can listen to only a slice of the received signal, multiple slices at once, etc
- The only limitations for simultaneous listening are sample rate, bandwidth, and compute power
- Much more expensive
- Rarely can transmit, most only receive

Normal Radio

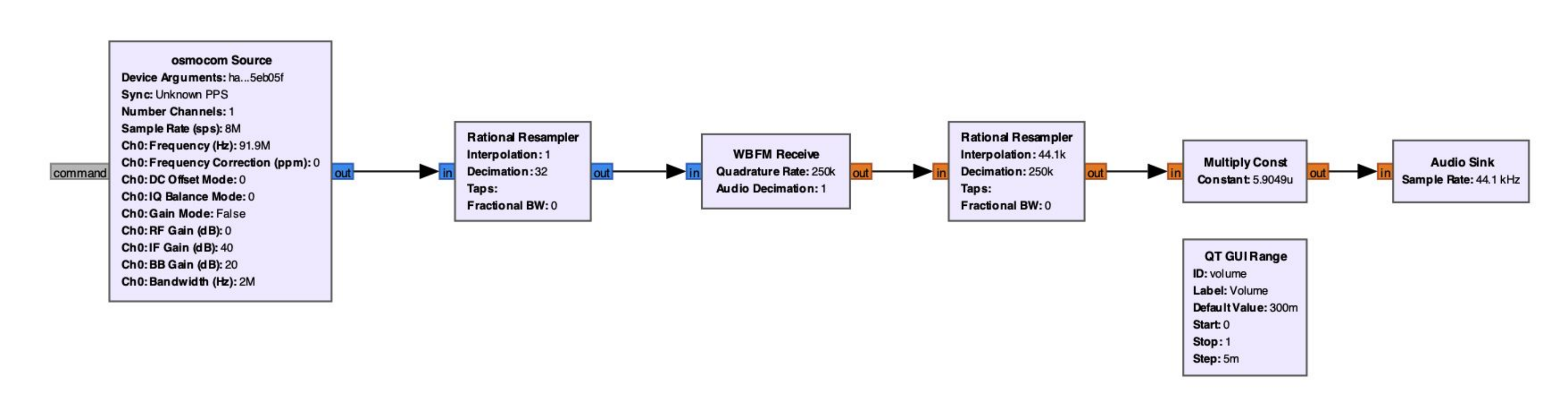
- Tunes to one frequency
- Processing done with analog circuitry
- Much cheaper

GNU Radio

A simple, but powerful tool for manipulating SDR-based radios.

- Python based
- Draw wires between nodes to create complex processors
- Emulate hardware radios in software
- Demos will be provided throughout the slides

Simple FM Receiver



osmocom Source
Device Arguments: ha...5eb05f
Sync: Unknown PPS
Number Channels: 1
Sample Rate (sps): 8M
Ch0: Frequency (Hz): 91.9M
Ch0: Frequency Correction (ppm): 0
Ch0: DC Offset Mode: 0
Ch0: IQ Balance Mode: 0
Ch0: Gain Mode: False
Ch0: RF Gain (dB): 0
Ch0: IF Gain (dB): 40
Ch0: BB Gain (dB): 20
Ch0: Bandwidth (Hz): 2M

Rational Resampler
Interpolation: 1
Decimation: 32
Taps:
Fractional BW: 0

WBFM Receive
Quadrature Rate: 250k
Audio Decimation: 1

Rational Resampler
Interpolation: 44.1k
Decimation: 250k
Taps:
Fractional BW: 0

Multiply Const
Constant: 5.9049u

Audio Sink
Sample Rate: 44.1 kHz

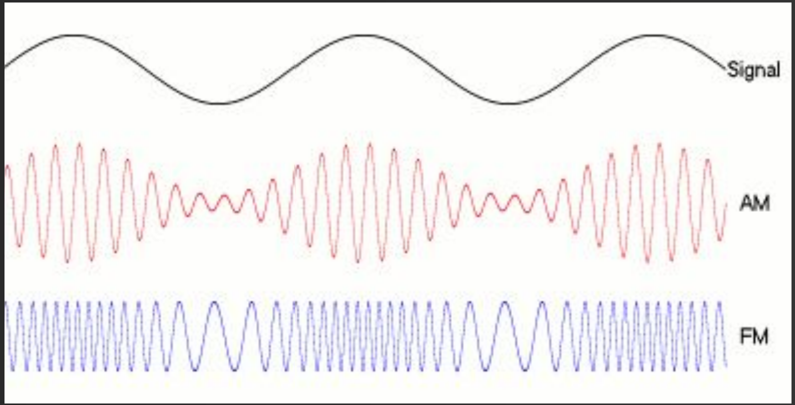
QT GUI Range
ID: volume
Label: Volume
Default Value: 300m
Start: 0
Stop: 1
Step: 5m

SimpleFM.grc

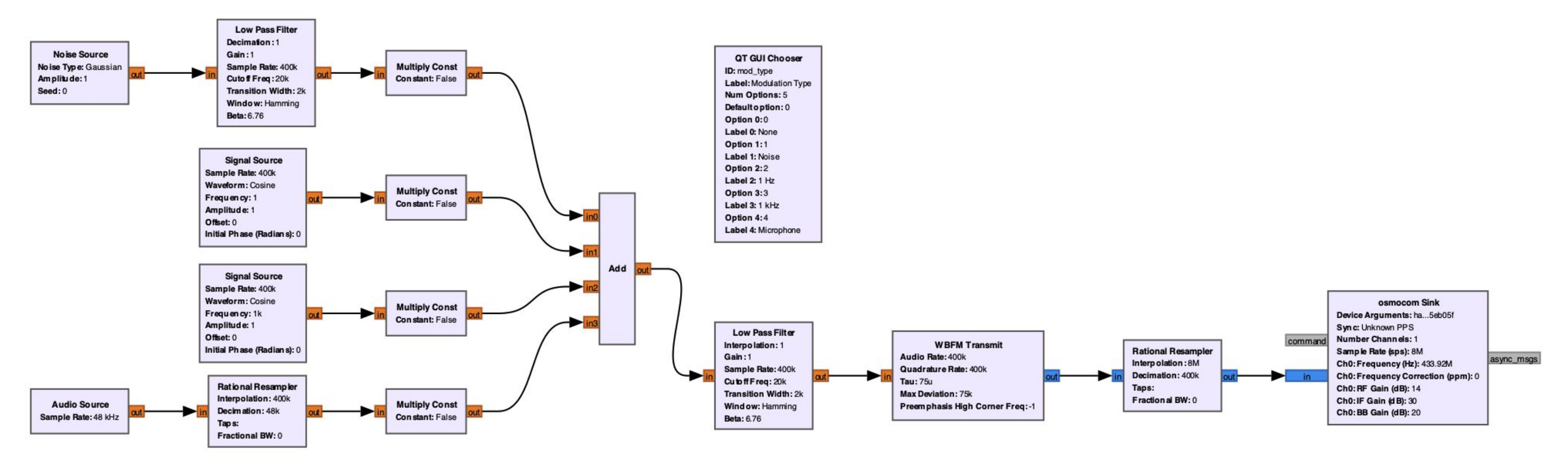
So what exactly is FM?

- Short for frequency modulation
- A method of encoding data by varying your carrier frequency

(Useful GIF I found on Wikipedia)



FM Modulation Demo



Noise Source
Noise Type: Gaussian
Amplitude: 1
Seed: 0

Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 400k
Cutoff Freq: 20k
Transition Width: 2k
Window: Hamming
Beta: 6.76

Multiply Const
Constant: False

Signal Source
Sample Rate: 400k
Waveform: Cosine
Frequency: 1
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0

Multiply Const
Constant: False

Signal Source
Sample Rate: 400k
Waveform: Cosine
Frequency: 1k
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0

Multiply Const
Constant: False

Audio Source
Sample Rate: 48 kHz

Rational Resampler
Interpolation: 400k
Decimation: 48k
Taps:
Fractional BW: 0

Multiply Const
Constant: False

Add

QT GUI Chooser
ID: mod_type
Label: Modulation Type
Num Options: 5
Default option: 0
Option 0: 0
Label 0: None
Option 1: 1
Label 1: Noise
Option 2: 2
Label 2: 1 Hz
Option 3: 3
Label 3: 1 kHz
Option 4: 4
Label 4: Microphone

Low Pass Filter
Interpolation: 1
Gain: 1
Sample Rate: 400k
Cutoff Freq: 20k
Transition Width: 2k
Window: Hamming
Beta: 6.76

WBFM Transmit
Audio Rate: 400k
Quadrature Rate: 400k
Tau: 75u
Max Deviation: 75k
Preemphasis High Corner Freq: -1

Rational Resampler
Interpolation: 8M
Decimation: 400k
Taps:
Fractional BW: 0

osmocom Sink
Device Arguments: ha...5eb05f
Sync: Unknown PPS
Number Channels: 1
Sample Rate (sps): 8M
Ch0: Frequency (Hz): 433.92M
Ch0: Frequency Correction (ppm): 0
Ch0: RF Gain (dB): 14
Ch0: IF Gain (dB): 30
Ch0: BB Gain (dB): 20

FM_Mod_Demo.grc

How do radios know when not to turn on

A few common methods:

- Digital Modes (Not vulnerable to static, expensive)
- Squelch (Vulnerable to static, cheapest)
- Tone Squelch (e.g. CTCSS, very cheap) <- Pay attention to this one, it comes back later
- Digital tones (like tone but more complex)

CTCSS

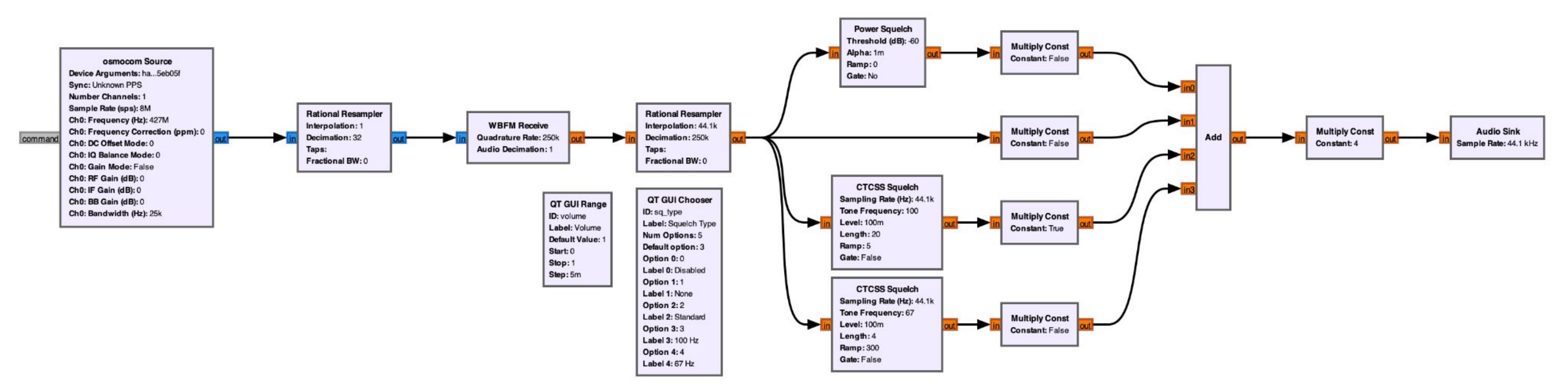
- CTCSS is a common method of squelch
- Not vulnerable to static
- Transmitter plays a tone (simple sine wave) that receiver listens for to enable the speaker output
- Almost always Sub-audible tone, filtered out
- Specific frequencies are used in walkie talkies and handheld radios
- Possible to be done completely analog.

Building on CTCSS

Digital tones such as Digital Coded Squelch (DCS) also exist, do the same thing but encoding binary data in tones, allows for more possible “tones”

- Not particularly common
- Most systems prefer to go full digital or fully analog
- Not applicable to the mic packs, but was a red herring we did waste some time on

Squelch Demo



osmocom Source
Device Arguments: ha...5eb05f
Sync: Unknown PPS
Number Channels: 1
Sample Rate (sps): 8M
Ch0: Frequency (Hz): 427M
Ch0: Frequency Correction (ppm): 0
Ch0: DC Offset Mode: 0
Ch0: IQ Balance Mode: 0
Ch0: Gain Mode: False
Ch0: RF Gain (dB): 0
Ch0: IF Gain (dB): 0
Ch0: BB Gain (dB): 0
Ch0: Bandwidth (Hz): 25k

Rational Resampler
Interpolation: 1
Decimation: 32
Taps:
Fractional BW: 0

WBFM Receive
Quadrature Rate: 250k
Audio Decimation: 1

Rational Resampler
Interpolation: 44.1k
Decimation: 250k
Taps:
Fractional BW: 0

QT GUI Range
ID: volume
Label: Volume
Default Value: 1
Start: 0
Stop: 1
Step: 5m

QT GUI Chooser
ID: sq_type
Label: Squelch Type
Num Options: 5
Default option: 3
Option 0: 0
Label 0: Disabled
Option 1: 1
Label 1: None
Option 2: 2
Label 2: Standard
Option 3: 3
Label 3: 100 Hz
Option 4: 4
Label 4: 67 Hz

Power Squelch
Threshold (dB): -60
Alpha: 1m
Ramp: 0
Gate: No

CTCSS Squelch
Sampling Rate (Hz): 44.1k
Tone Frequency: 100
Level: 100m
Length: 20
Ramp: 5
Gate: False

CTCSS Squelch
Sampling Rate (Hz): 44.1k
Tone Frequency: 67
Level: 100m
Length: 4
Ramp: 300
Gate: False

Multiply Const
Constant: False

Multiply Const
Constant: False

Multiply Const
Constant: True

Multiply Const
Constant: False

Add

Multiply Const
Constant: 4

Audio Sink
Sample Rate: 44.1 kHz

Squelch_Demo.grc

Listening to a transmitter

- SLX Beltpacks have their assigned frequencies listed
- Tune to the band using GQRX
- Look around a little, try a few different modulation modes
- Ta-Da. You are now able to listen to a beltpack

Okay, we’ve got an idea, now what?

OSINT!

- Start by doing research on the specific mics in question
  - Shure Website
  - FCC License search
- Solidify your understanding of how FM works (this would’ve helped us)
  - Carrier transmits all the time
  - Frequency information is transmitted by moving the carrier at different speeds
  - Volume information is controlled by changing the deviation (distance the carrier moves)
- Be mindful of which frequencies you are licensed to transmit on

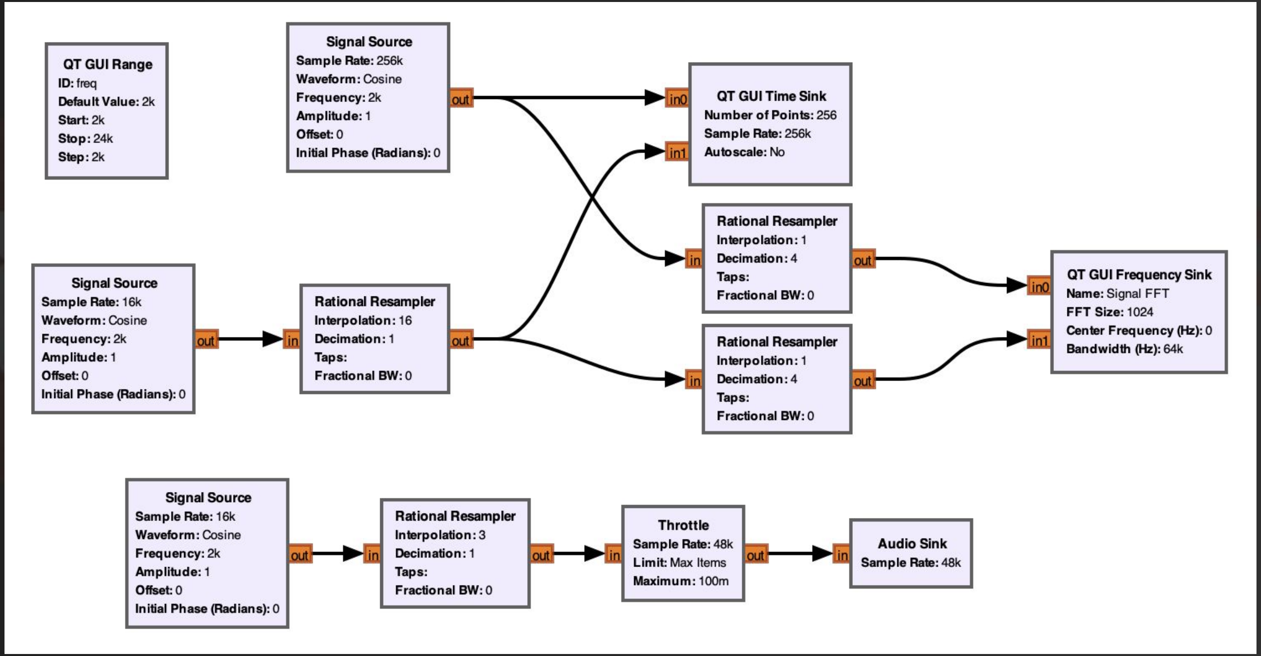
Roadblock 1: Nyquist-Shannon Sampling Theorem

- Sounds scary, but is actually quite simple
- Basis for all digital audio
- Maximum producible frequency = ½ Sampling rate

Roadblock 1 (Cont.)

- Why is this an issue?
- SLX Microphones use a 32 kHz tone
  - Super-audible
- Common sampling rates include 44.1 kHz and 48 kHz
- We would need to sample at least 64 kHz to reproduce the tone
- So I decided 400 kHz should be more than enough

Nyquist Demo



QT GUI Range
ID: freq
Default Value: 2k
Start: 2k
Stop: 24k
Step: 2k

Signal Source
Sample Rate: 256k
Waveform: Cosine
Frequency: 2k
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0

QT GUI Time Sink
Number of Points: 256
Sample Rate: 256k
Autoscale: No

Rational Resampler
Interpolation: 1
Decimation: 4
Taps:
Fractional BW: 0

QT GUI Frequency Sink
Name: Signal FFT
FFT Size: 1024
Center Frequency (Hz): 0
Bandwidth (Hz): 64k

Signal Source
Sample Rate: 16k
Waveform: Cosine
Frequency: 2k
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0

Rational Resampler
Interpolation: 16
Decimation: 1
Taps:
Fractional BW: 0

Rational Resampler
Interpolation: 1
Decimation: 4
Taps:
Fractional BW: 0

Signal Source
Sample Rate: 16k
Waveform: Cosine
Frequency: 2k
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0

Rational Resampler
Interpolation: 3
Decimation: 1
Taps:
Fractional BW: 0

Throttle
Sample Rate: 48k
Limit: Max Items
Maximum: 100m

Audio Sink
Sample Rate: 48k

Nyquist_Demo.grc

Roadblock 2: FM Deviation

Roadblock 2 (Cont.)

- Made more progress after upping sampling rate
- Audio now sounds “crunchy”
- FM volume is measured in deviation
  - Aka: how much it moves from the carrier frequency
- Our deviation was too low and would cause clipping

Fine Tuning

- Changed deviation
- Fixed ratio of tone to audio
  - Ratio and deviation was a bit of back and forth
- Double checked the power output from HackRF

LIVE DEMO

Yeah, we weren’t kidding.
Illegal things are about
to happen lol

Other Dangerous Discoveries

The Beltpacks don’t mute when you press the mute button, they only decrease the volume by 40db.

Don’t say anything you wouldn’t want heard if the mic is on, even if it’s muted

Someone with an SDR can just increase the volume back up and hear everything you are saying.