

Project 1

K20 Wearable Police Proximity Awareness Module

The Concept

In response to the heated, racially charged events surrounding Ferguson, Missouri, racial profiling, police brutality and the overall growing strength and prominence of the police force, how could I create something that would bring balance back to our democratic society. Power can be redistributed by finding a way of countering one of the many tools at the police' disposal.

That tool to counter being radio communication.

Project Goal

To make and integrate a modular haptic sensory vest that is tuned to manhattan police radio frequencies. A radio integrated into the vest receives data that is translated by an Arduino into vibration feedback. The vest can easily be attached and detached from a hoodie for recharging of the vest and washing of the host garment. Ultimately it should provide proximity awareness of police for the wearer.

Pieces of the Puzzle

The components for my project

Justin and Michael's help!

The Internet!

A fairly basic understanding of the spectrum is helpful.

A fairly basic understanding on how an antenna works by resonating with different radio frequencies depending on its length to produce sound.

Great for figuring out how to run a Raspberry Pi for the first time.

The radio spectrum! (part of it)

High Frequency (HF) = 3 - 30MHz \rightarrow 100m to 10m

Very High Frequency (VHF) = 30-300MHz \rightarrow 10m to 1m

Ultra High Frequency (UHF) = 300-3kMHz \rightarrow 1m to 100mm

http://invisiblenetworks.co/wp-content/uploads/2015/02/United_States_Frequency_Allocations_Chart_2003_-_The_Radio_Spectrum.jpg

Manhattan police broadcast within 470 - 480MHz
so I am trying to receive UHF frequencies.

Fun fact: wifi routers and microwave ovens operate on the
same frequency, 2.4 GHz!

SDR (Software Defined Radio)

A miniature, usb pluggable radio receiver/tuner.

Currently using R820T



An antenna

Many types of antennas!

Monopole, dipole, stripperpole

I used a monopole and a dipole but no
stripperpole :(

I needed a BNC female to MCX male adaptor to
attach a monopole antenna bought from
RadioShack to the dongle

Raspberry Pi

Running Debian, connected via ethernet (simplest way to connect) to install a few libraries off of git:

libusb-1.0-0 - not totally sure what this is. but it is necessary. assuming it has something to do with usb communication?

steve-m librtlsdr - This is where the magic happens.

plug in the R820T dongle or other SDR dongle.

SDR program

Running within a terminal window:

Example command:

```
rtl_fm -M wbfm -f 89.1M | play -r 32k -t raw -e s -b  
16 -c 1 -V1 -
```

wbfm = wideband fm, which is radio broadcast.

-f = frequency to tune.

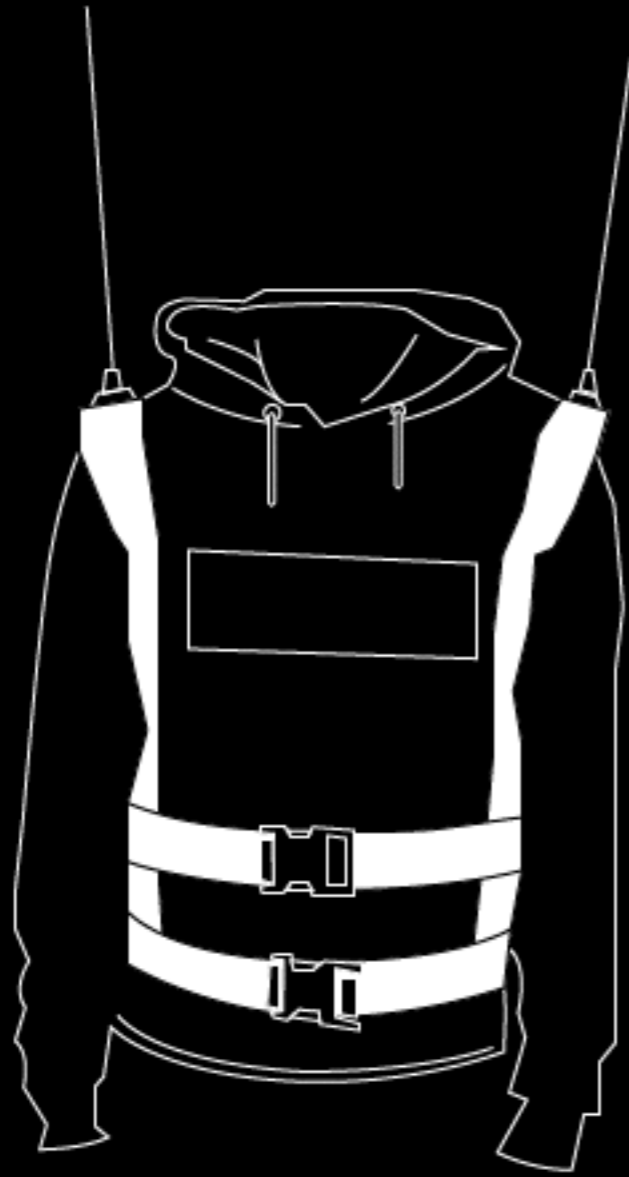
Arduino

Arduino audio input:
parts:

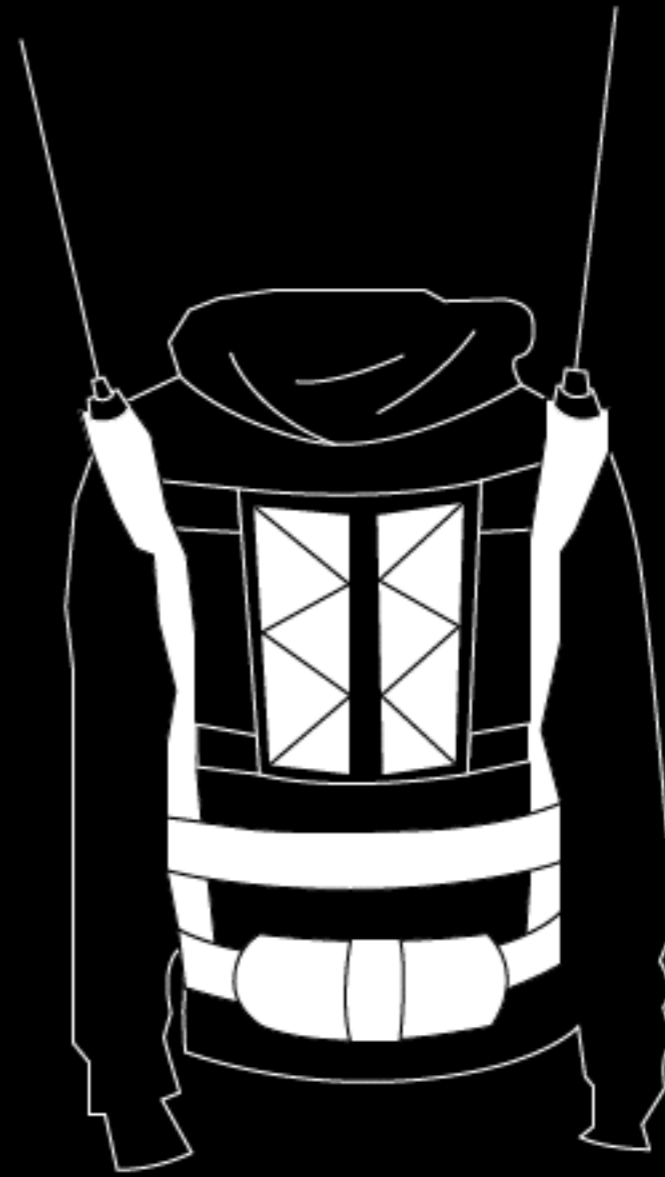
TL072, 9V battery, Snap connector, mono 1/8 audio jack, LED, 10kOhm linear potentiometer, 100kOhm 1/4 watt resistor, 10uF capacitor, 47nF Capacitor, solder, 22 gauge wire, an oscilloscope and of course, the Uno board.

Will convert the sound current into +5 to 0v wave

End result should look something like this



Front



Back

1st attempt w/ the Pi

- Used the `R820T`
- Installed all necessary dependancies (aside from audio - i just wanted to test if it would at the very least recognize the device)
- did not work.

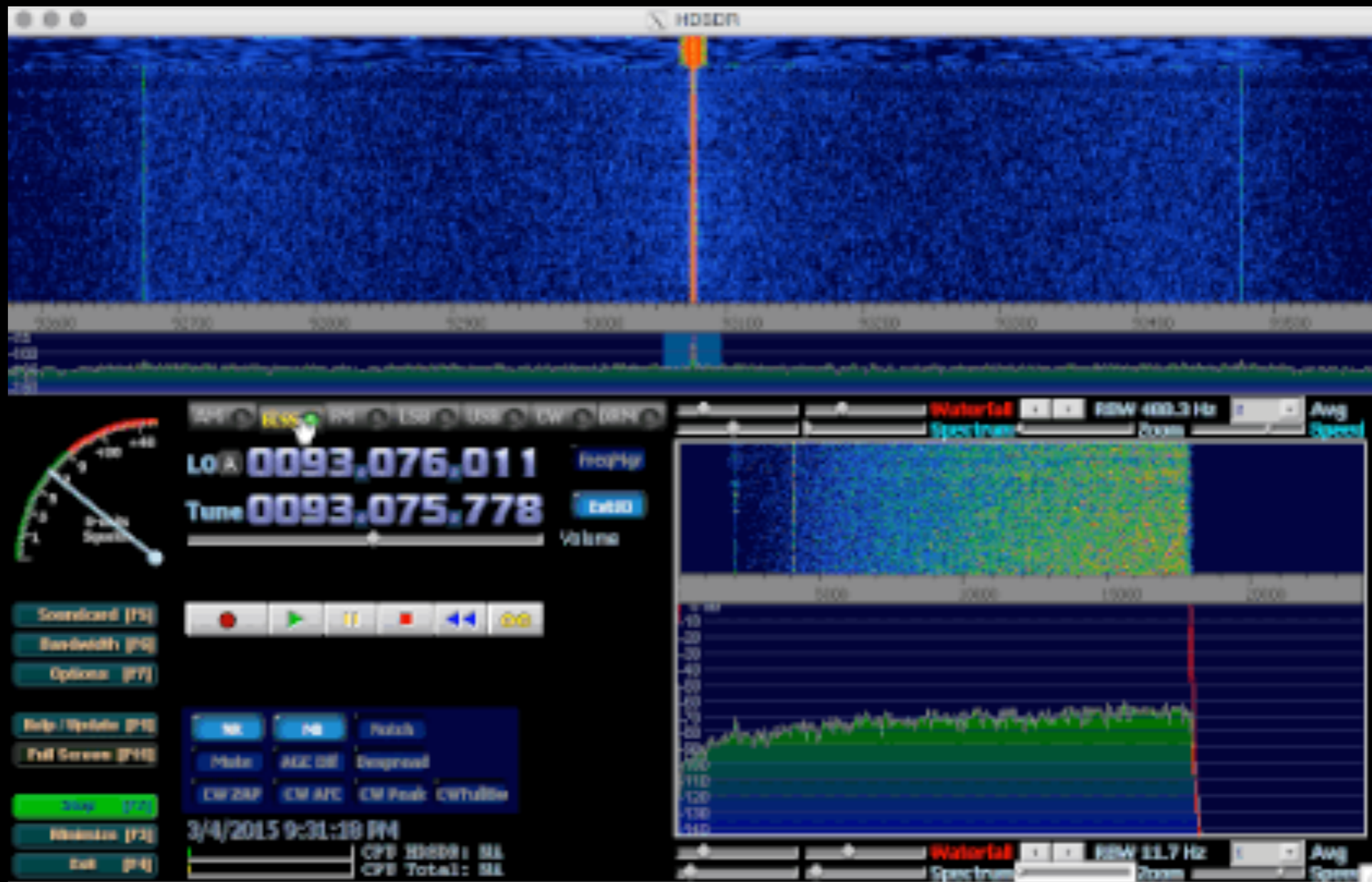


What I learned

- After some searching on forums, I found that the `nooElec R820T` dongle i was using draws too much power from the Pi, and it prevents it from being used.

2nd attempt w/ Mac


- Install HDSDR (a GUI for sdr, very cool)



What I learned

- Despite the awesome interface with the waterfall visual and buttons for tuning, still **no clear signal.** I scanned for a few hours... probably should have moved on at that point, but the interface was too much fun!

3rd attempt w/ Mac

-  Install `homebrew` package compiler
- `homebrew install librtlsdr`
- New dongle! Now `KEEDEEX stl-sdr 2832u R820T`
- `homebrew install sox` (cross-platform audio format converter/player)
- `rtl_fm -M wbfm -f 92.9 | play -r 32k -t raw -e s 16 -`
`c 1 -v | -`

ll tune to a broadcast FM station.

s the frequency to tune in.

fm says to use wideband FM mode, but this is really a shortcut for a tweaked narrowband FM mode. It expands fully into

```
rtl_fm -f 89.1M -M fm -s 170k -A fast -r 32k -l 0 -E deemp | play -r 32k ...
```

'-f ...' indicated the frequency to tune to

-M fm means narrowband FM

-s 170k means to sample the radio at 170k/sec

-A fast uses a fast polynomial approximation of

-r 32k means to lowpass/resample at 32kHz

-l 0 disables squelch

-E deemp applies a deemphasis filter

edless to say, this is a lot just to listen to the local radio

might need to manually add some gain with -g 20 or

<https://www.library.no>

```
deevolution - box - 80x24
box
-: (raw)
  Encoding: Signed PCM
  Channels: 1 @ 16-bit
  Samplerate: 32000Hz
  Replaygain: off
  Duration: unknown

In:0.00% 00:00:00.00 [00:00:00.00] Out:0 | | | |
und 1 device(s):
  0: Realtek, RTL2838UHIDIR, SN: 00000001

Using device 0: Generic RTL2832U DEM
Found Rafael Micro R820T tuner
Tuner gain set to automatic.
Tuned to 93171000 Hz.
Oversampling input by: 8x.
Oversampling output by: 1x.
Buffer size: 8.03ms
Exact sample rate is: 1020000.000000 Hz
Sampling at 1020000 5/5.
```

What I learned

It works!

4th attempt w/ Pi

BOOO-YAH!

What I used

- used the newly ordered `KEEDEX stl-sdr 2832u R820T`
- After running `rtl_test` I got an error.
- `“sudo rmmmod dvb_usb_rtl28xxu”` detached the dvb portion of the dongle
- `rtl_fm -f 96.3e6 -M wbfm -s 200000 -r 48000 – | aplay -r 48k -f S16_LE`



Quick demo

Reflection

If the intended goal is to give citizens more power by providing them with this tool of awareness then:

Does it work? How would it make the wearer feel if he was constantly and relentlessly receiving feedback?

The device may be more paranoia inducing than it would be helpful.

Maybe there are some tweaks to the design that could make it less paranoia inducing, and more helpful? how?

Might there be other tools that could be integrated in the garment to return power to the wearer?

hard to tell until the entire piece is complete and user tested.

Next Steps

- Build my own custom antenna or buy the super wide band Comet BNC-W100RX (25MHz - 1300MHz).
- Buy and test the rtl2832u e4000 usb stick with the Pi.
- build and test the audio to Arduino circuit with the Pi.
- Determine method of power. Solar power? Batteries?
- Integrate into hoodie.

Questions/feedback