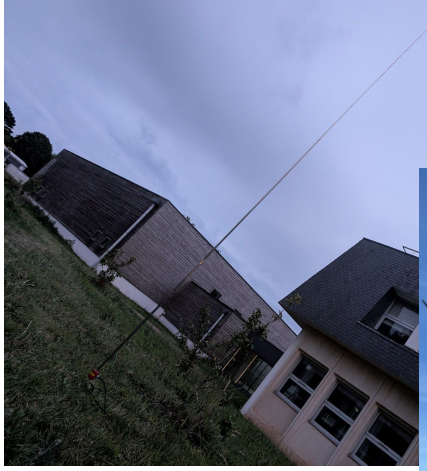
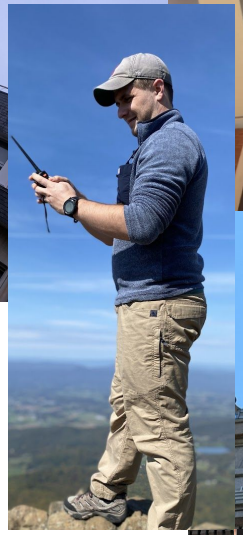
Paging all radio curious hackers!

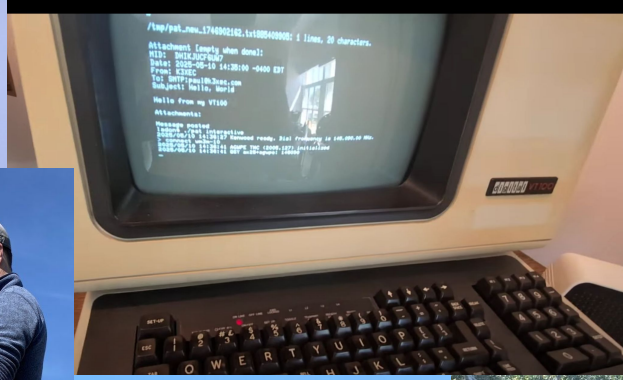
paultag

paultag@gmail.com
@paul@soylent.green
github.com/paultag / pault.ag / k3xec.com

whoami



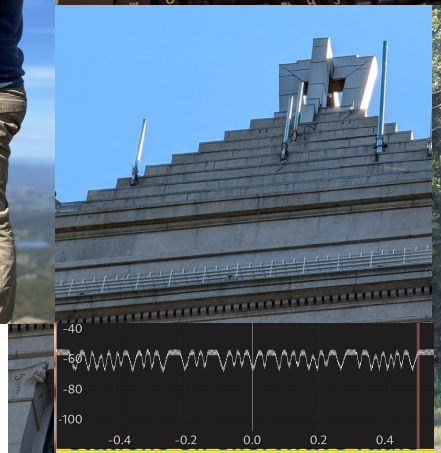


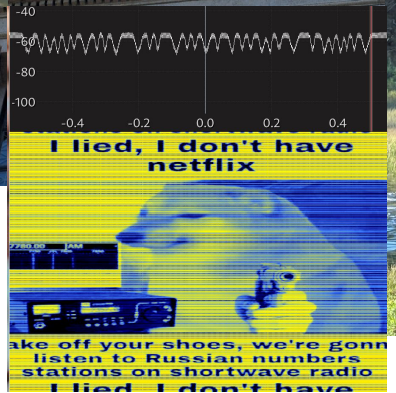


i'm paultag

- debian (ftpteam, tech-ctte)
- ubuntu
- open source initiative (former board member)
- ham radio (K3XEC, trustee for K4US)
- Hy (hylang.org)

I have a job, and I've had some jobs over the years too (not related to this talk so i'll skip that slide)





whyami

- because i'm not qualified!
- i'll keep this to basically "no" radio specific knowledge
  - (but I will assume you know how to program)

whyami

- devices communicate all around us.
- not everything is wifi or bluetooth!
- radios are fun :3

today’s target

“Retekess TD-158 Restaurant Pager System”

Crash course on using the pager system

Pagers are tied to a base station. That base station has a numerical ID (3 digit; 10 bit at least to store 999?)

Call “0” to turn pagers off (sleep mode)
Call a number to ring a pager. Put it back on the base to stop ringing.

Pairing procedure to set the pager number / base station
Similar dance to set beep mode / ring mode
Those are called “F2”-”F5”. Function?.

I set my base station to “55” (unique bit pattern - `0b110111`, helpful for determining endianness, it’s a weird number)

research

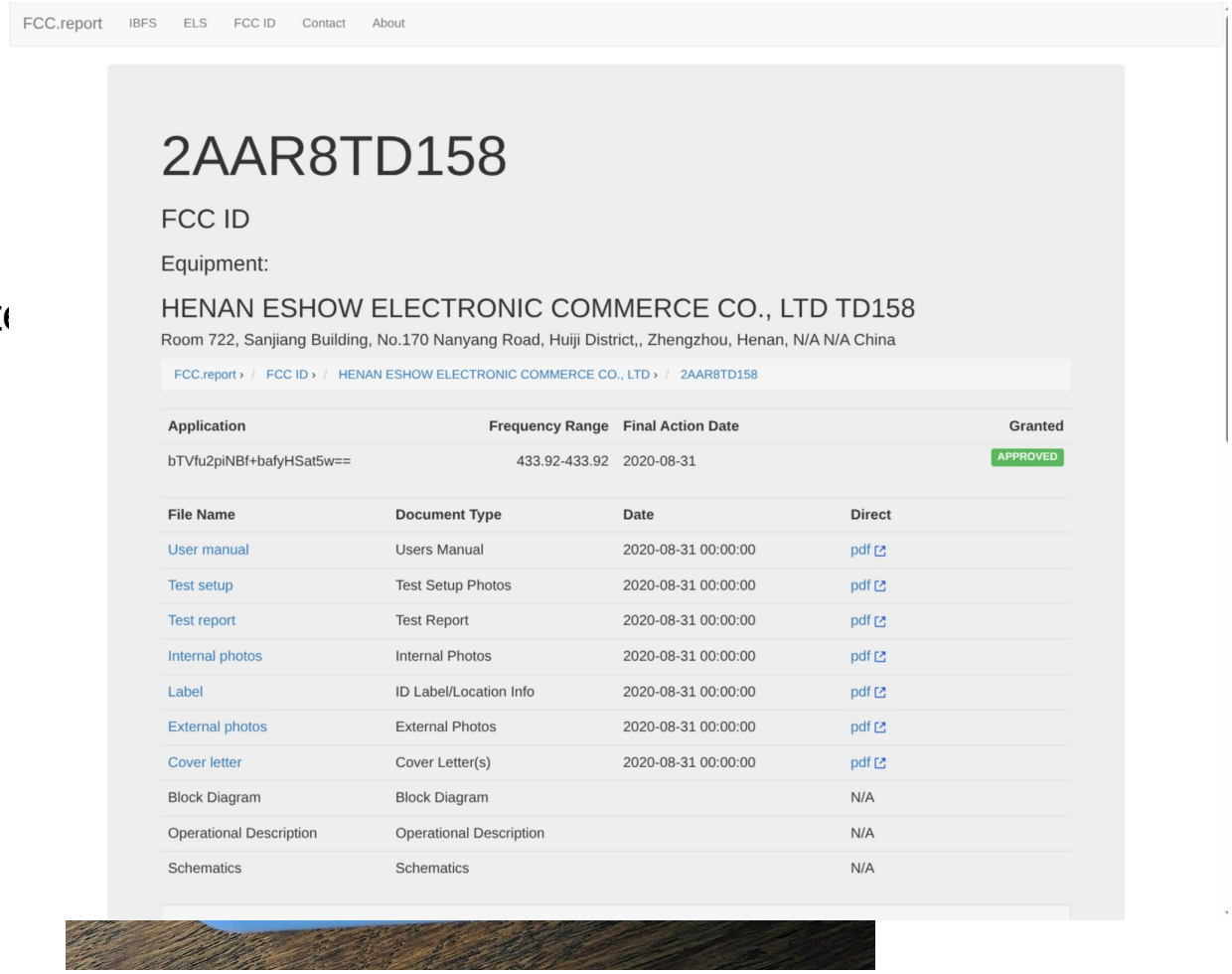

RETEKESS
Queue Wireless Calling System
Model: TD158
FCC ID: 2AAR8TD158
FCC ID: 2AAR8TD158A
IC: 28503-TD158
Henan Eshow Electronic Commerce Co.,Ltd
MADE IN CHINA
S/N:2311T1581100005

research

Manufacturer websites (sometimes) and FCC filings (often) are a goldmine

research

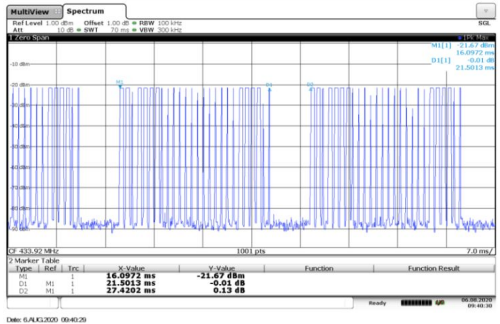
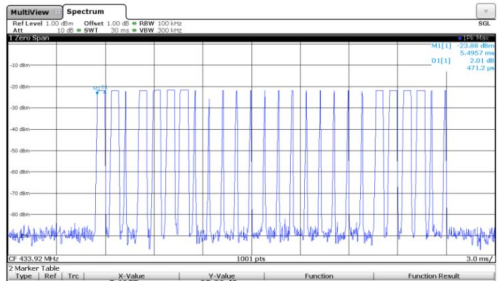
Manufacturer websit



research (fcc)

Download: PDF

EMC TRF Template 21 / 34 100%

Pulse Width :	
Pulse 1:	

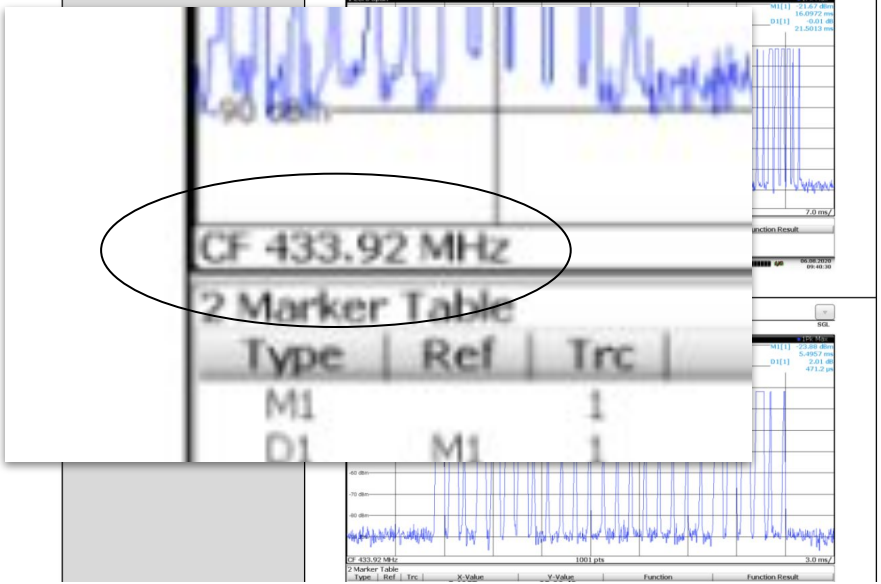
research (fcc)

Download: PDF

EMC TRF Template 21 / 34 100%

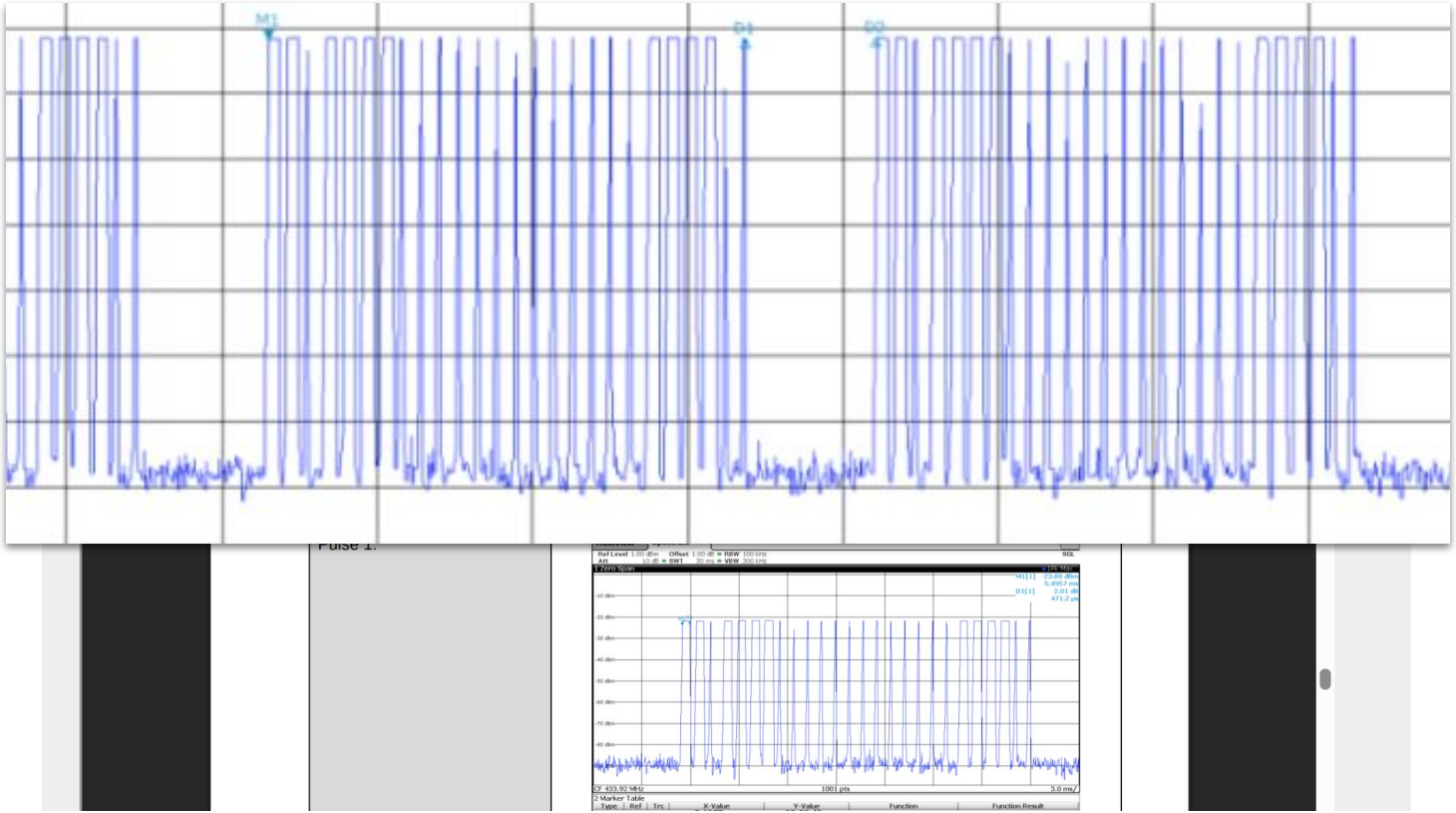
Report No.: CHTEW20080138 Page: 21 of 34 Issued: 2020-08-18

Pulse Width :



research (fcc)

Distinctive “On Off Keying”



capture signal using an sdr

What are `Samples`?

You can think of Samples as reading voltage levels over time from an antenna via an analog to digital converter (“ADC”) -- exactly like if you had an Arduino or Raspberry Pi reading analog values from a pin with a very precisely timed loop. The number of samples being read per second is called the sampling rate.

The file we have on disk is all those ADC readings of power over time, at some precise sample rate.

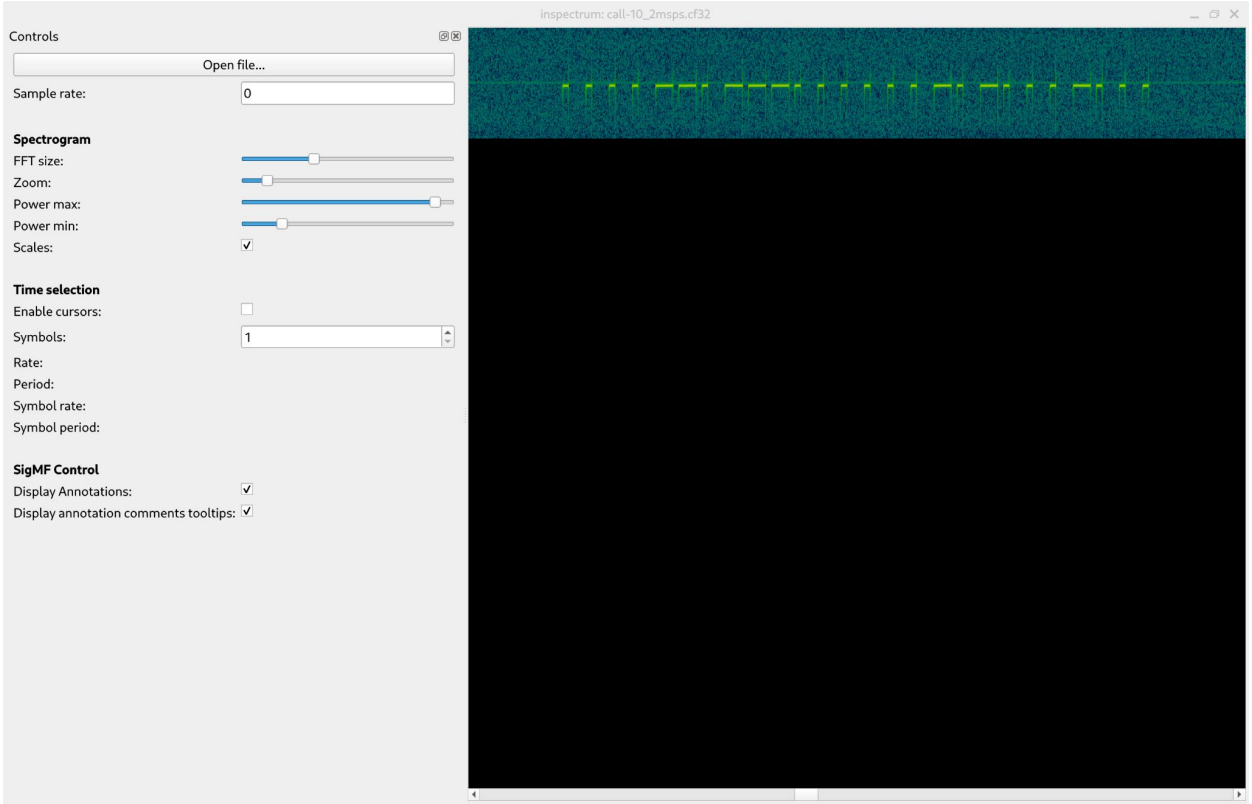
The readings are two numbers together, like a point (x, y) - a complex number.

capture signal using an sdr

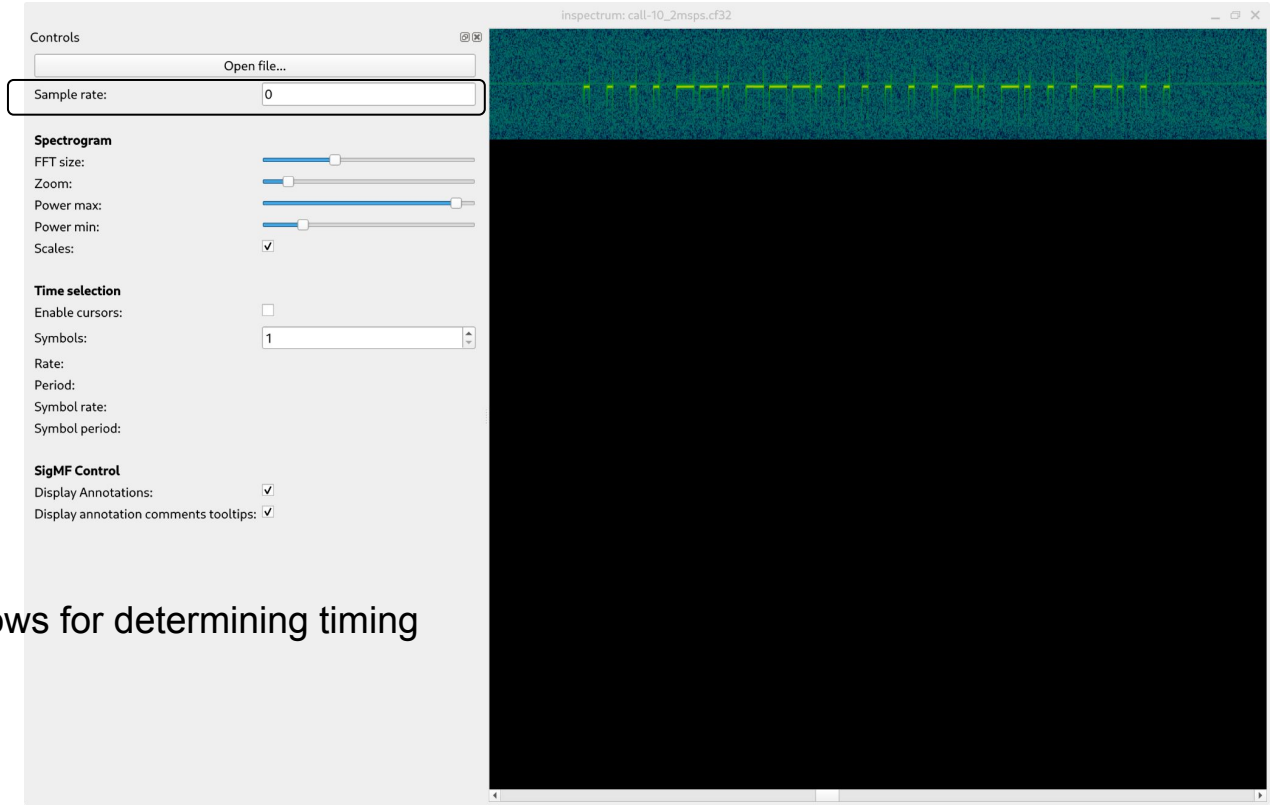
Now that we have our complex valued IQ sample data on disk, we can open it to take a look at it visually.

I'm going to use `inspectrum`. In order for `inspectrum` to read the `data`, be sure your file(s) end in the right extension (for us today: `.sc8)`. If you have a different format, the `--help` flag has the full list.

capture signal using an sdr (this is inspectrum)

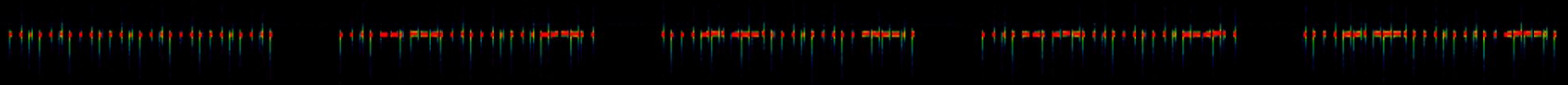


capture signal using an sdr (this is inspectrum)



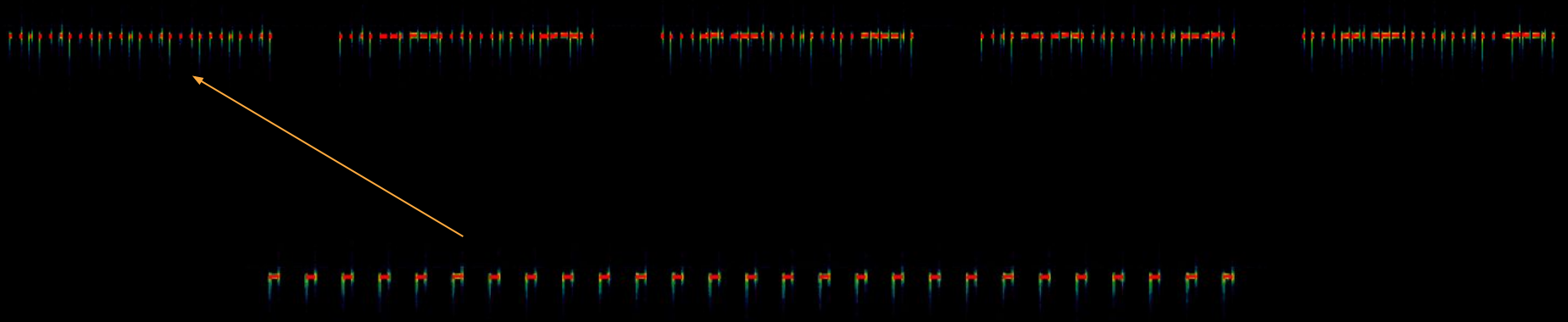
Setting this allows for determining timing

demod



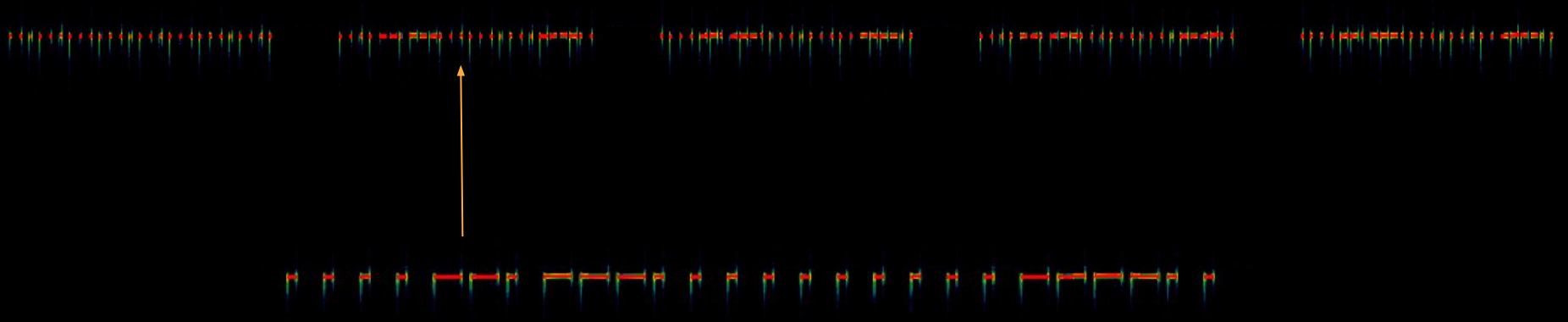
(also inspectrum but cropped for readability)

demod



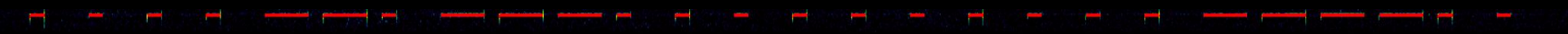
likely “preamble” to wake up listening devices

demod

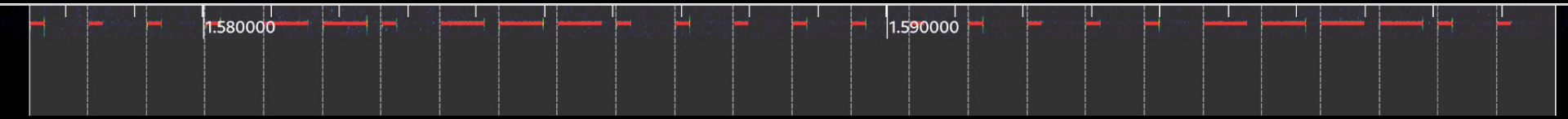


looks like data (subsequent packets are identical)

demod data



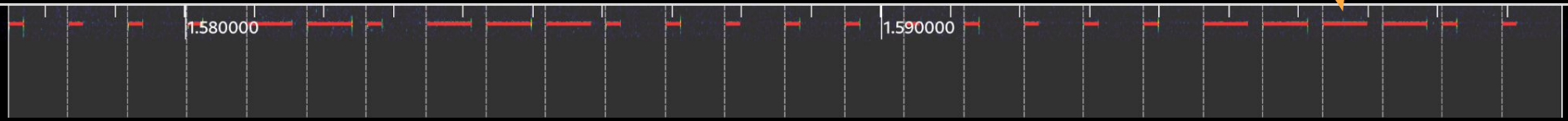
demod data



demod

Time selection

Enable cursors:	☑
Symbols:	26
Rate:	44.8029Hz
Period:	22.32ms
Symbol rate:	1.16487kBd
Symbol period:	858.462µs



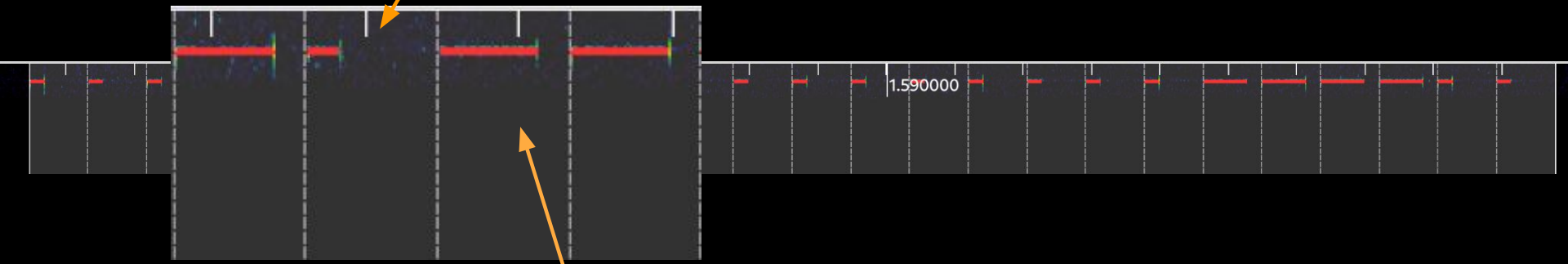
demod

(¼ of the symbol “on”)

short

858us per symbol
214.5us for ¼ of a symbol

214.5us for short
643.5us for long



(¾ of the symbol “on”)

long

demod

Time selection

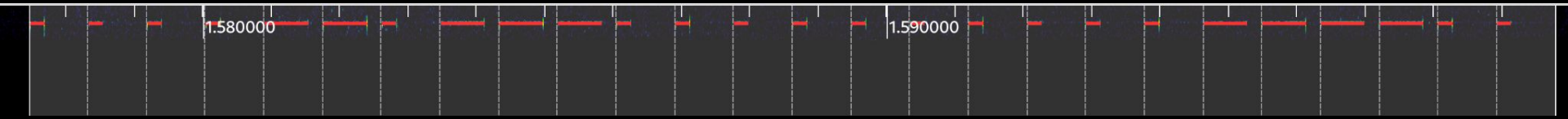
Enable cursors:	☑
Symbols:	26
Rate:	44.8029Hz
Period:	22.32ms
Symbol rate:	1.16487kBd
Symbol period:	858.462µs

858us per symbol

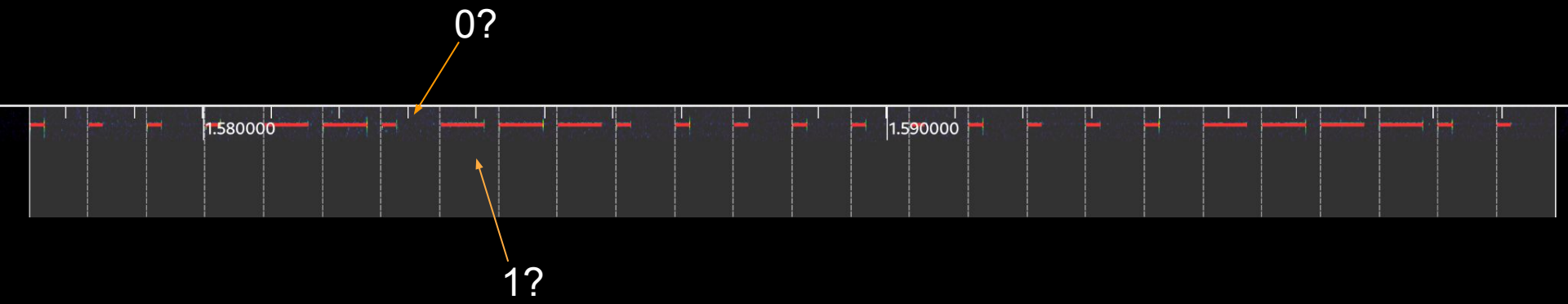
6 “dead air” symbols between packets - roughly 5.15ms

this may not matter, but hey, why not be pendants and get it right.

demod data

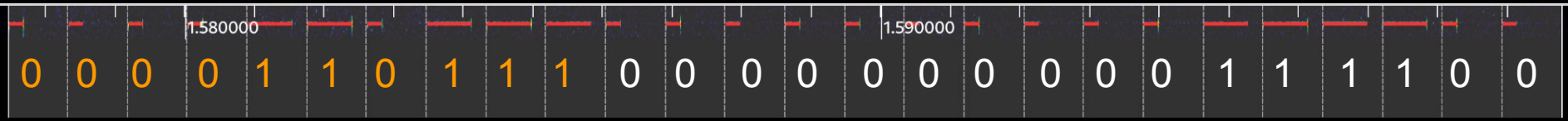


demod data



demod data

demod data



base station ID (55; 0b110111)

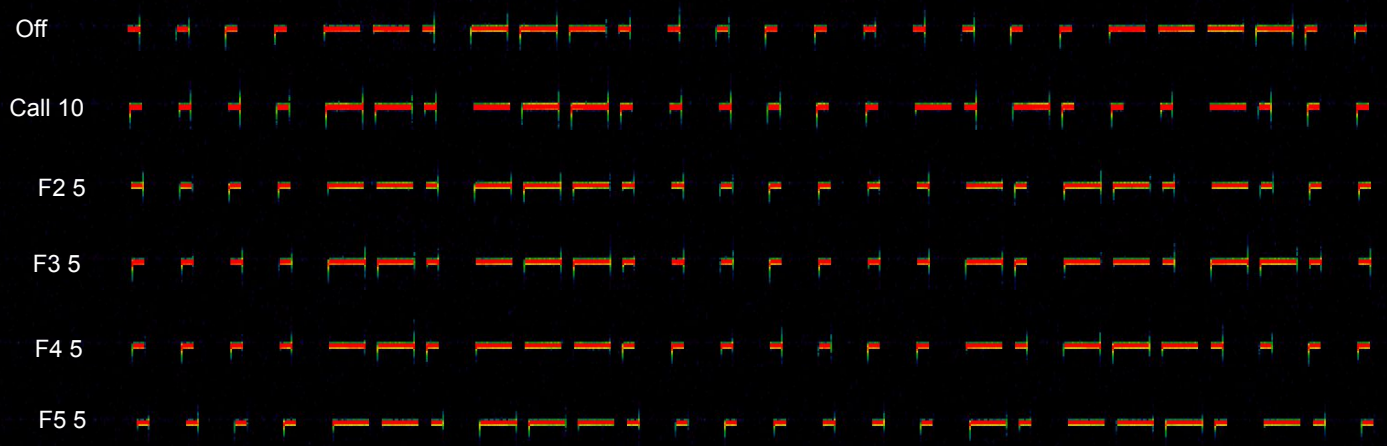
demod data

base station ID (55; 0b110111)
lol uhhh, bits?

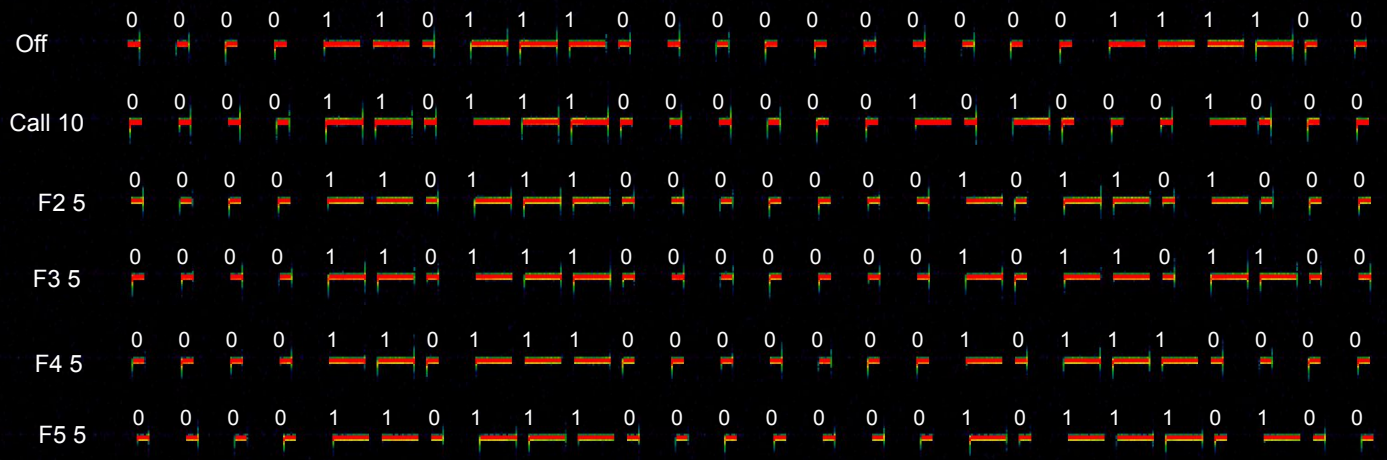
We need more data, time to capture all signals

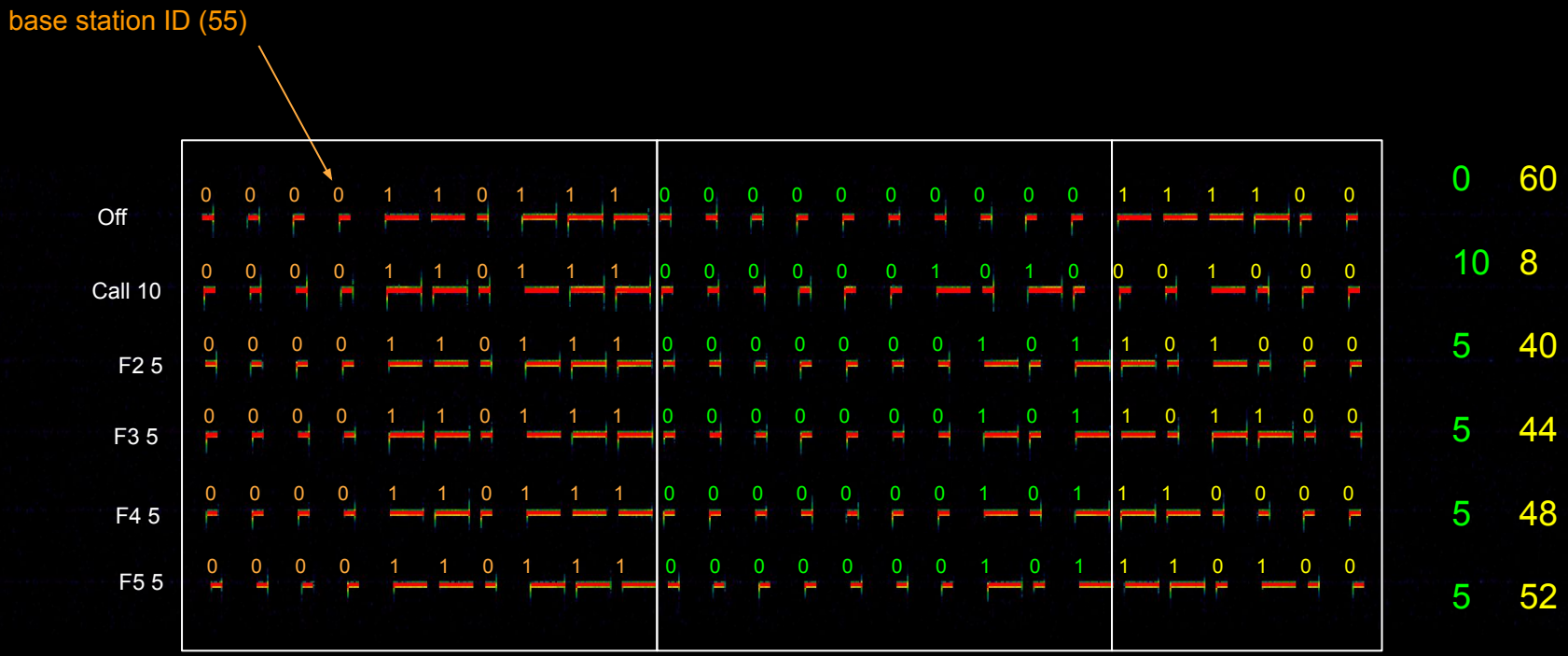

USRP B210
70 MHz - 6 GHz
RF A
RF B
TX/RX
RX2
ESD SENSITIVE

over and over and over and over and over and over and over and over and over and over and

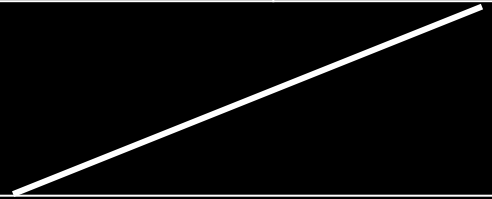

Off
Call 10
F2 5
F3 5
F4 5
F5 5

(I just did these by eye)




base station ID (55)
Off
Call 10
F2 5
F3 5
F4 5
F5 5
0 60
10 8
5 40
5 44
5 48
5 52

Base ID (10 bits)	Argument (10 bits)	Command (6 bits)



Call	0x8	Call a pager (pager ID in “argument”)
Off	0x3c	“Argument” set to all 0
F2	0x28	Program a pager to the pager ID in “argument”
F3	0x2c	Set the reminder duration, in seconds, specified in “argument”
F4	0x30	Set the beep mode, specified in “argument” - 0: disable, 1: slow, 2: medium, 3: fast
F5	0x34	Set the vibration mode specified in “argument” - 0: disable, 1: enable

modulation

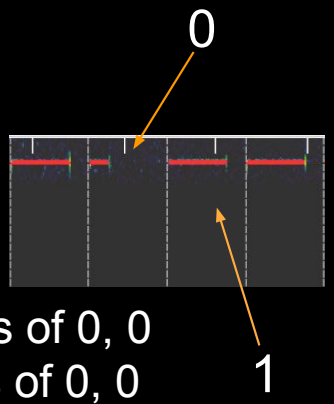
858us per symbol
214.5us for ¼ of a symbol

214.5us for 0
643.5us for 1

I'm going to generate something at 2 Msps (some devices have a lower limit on samples per second, also I just like 2Msps why not)

How many samples at 2 Msps is 214.5 us?

2000000 * 0.0002145 = 429
^ samples/sec ^ How long (in seconds) a sample is



Therefore!

A one bit is `(429*3)` IQ samples of 127, 0, followed by `249` IQ samples of 0, 0
A zero bit is `429` IQ samples of 127, 0, followed by (`429*3`) IQ samples of 0, 0

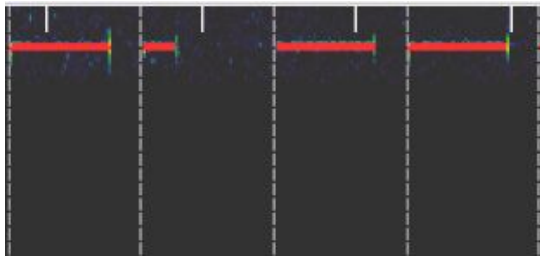
Modulating a bit in code

Given that:

one is encoded as `(429*3)` IQ samples of 1+0i, followed by `249` IQ samples of 0+0i
zero is encoded as `429` IQ samples of 1+0i, followed by (`429*3`) IQ samples of 0+0i

```
carrier_off = [0, 0] * 429
carrier_on = [127, 0] * 429

one = (carrier_on * 3) + carrier_off
zero = carrier_on + (carrier_off * 3)
```



Modulating bits in code

Iterate over all bits, and return the bytes for a “one” symbol or a “zero” symbol, MSB first.

```
one = …
zero = …

def bits2iq(bits, len):
    message = []
    for bit in range(len, 0, -1):       # for each bit
        if bits & (1 << (bit-1)) != 0: # check if the bit is set
            message += one
        else:
            message += zero
    return message
```

Preamble

```
def preamble():
    return bits2iq(0, 26)
```

Modulating a packet

```
# see last slide
def bits2iq(bits, len):

def message2iq(*, base_id, argument, command):
    return bits2iq(base_id, 10) \
            + bits2iq(argument, 10) \
            + bits2iq(command, 6)
```

Base ID (10 bits)	Argument (10 bits)	Command (6 bits)

Making our “library”

```
# see last slide
def message2iq(*, base_id, argument, command):

COMMAND_OFF = 60
COMMAND_CALL = 8
COMMAND_F2 = 40
COMMAND_F3 = 44
COMMAND_F4 = 48
COMMAND_F5 = 52
```

Call	0x8	Call a pager (pager ID in “argument”)
Off	0x3c	“Argument” set to all 0
F2	0x28	Program a pager to the pager ID in “argument”
F3	0x2c	Set the reminder duration, in seconds, specified in “argument”
F4	0x30	Set the beep mode, specified in “argument” - 0: disable, 1: slow, 2: medium, 3: fast
F5	0x34	Set the vibration mode specified in “argument” - 0: disable, 1: enable

Making our “library”

```
def call(*, base_id, pager_id):
    return message2iq(base_id=base_id,
                      argument=pager_id,
                      command=COMMAND_CALL)
```

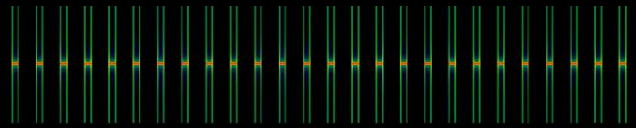
Modulating a full payload

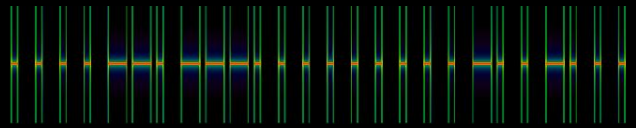
Let’s write our IQ data to stdout, where we can pipe it around (and transmit it!)

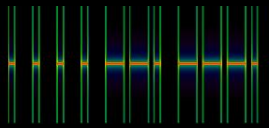
```
def full_burst(message):
    blank = (_carrier_off * 4) * 6
    packet = blank + preamble() + blank
    for _ in range(0, 5):
        packet += message + blank
    return packet

import sys
command = call(base_id=55, pager_id=1)
packet = full_burst(command)
sys.stdout.buffer.write(bytes(packet))
```

How’s it look in inspectrum?







Looks good! Preamble followed by the same data as we saw over the air!

live demo time

```
python3 paging-all.py | \
  hackrf_transfer -t -
    -f 433920000
    -a 1
    -x 47
    -s 2000000
```

Questions!

Want to try to do this yourself? I have gear with me and other radios!

Come to federal A/B!

https://k3xec.com/parch/

paultag@gmail.com
@paul@soylent.green
github.com/paultag / pault.ag / k3xec.com