MATT KNIGHT // BASTILLE NETWORKS

REVERSING LORA

EXPLORING NEXT-GENERATION WIRELESS

INTRODUCTION

- Matt Knight
- Software Engineer and Threat Researcher @ Bastille
- Background in electrical engineering, embedded software, etc.
- I love wireless!

matt@Bastille.net
@embeddedsec

BEFORE WE GET STARTED…

- No 0-days or exploits
- We're going to take apart a cutting edge protocol
- This is a software defined radio talk
- I'll assume you're technical, but no SDR experience required

WHY IS THIS RELEVANT

- Cisco IBSG: 50 billion devices by 2020
  - Fewer wires every year
- Wireshark wasn't always a thing
- Monitor mode wasn't always a thing
- Low-level access to interfaces is paramount for enabling comprehensive security research

AGENDA

1. Survey of IoT-focused wireless
2. Introduce LPWANs
3. Review technical radio concepts
4. Reverse engineer the LoRa PHY with SDR

IOT == EMBEDDED

EMBEDDED/IOT REALITIES

- Low intelligence (basic CPUs)
- Battery powered
- Hard-to-reach places
- Longevity/high endurance
- Must be easy to configure

EMBEDDED/IOT COMMUNICATION — REQUIREMENTS

- Given these constraints, an ideal IoT interface is…
  - Wireless
  - Easy on the battery
  - Capable of being installed anywhere
  - No configuration/easy to configure
  - Inexpensive!

EMBEDDED/IOT COMMUNICATION — REQUIREMENTS

- IoT interfaces do NOT require:
  - High throughput
  - Persistent/always-on connections

- Current IoT devices are grossly overserved

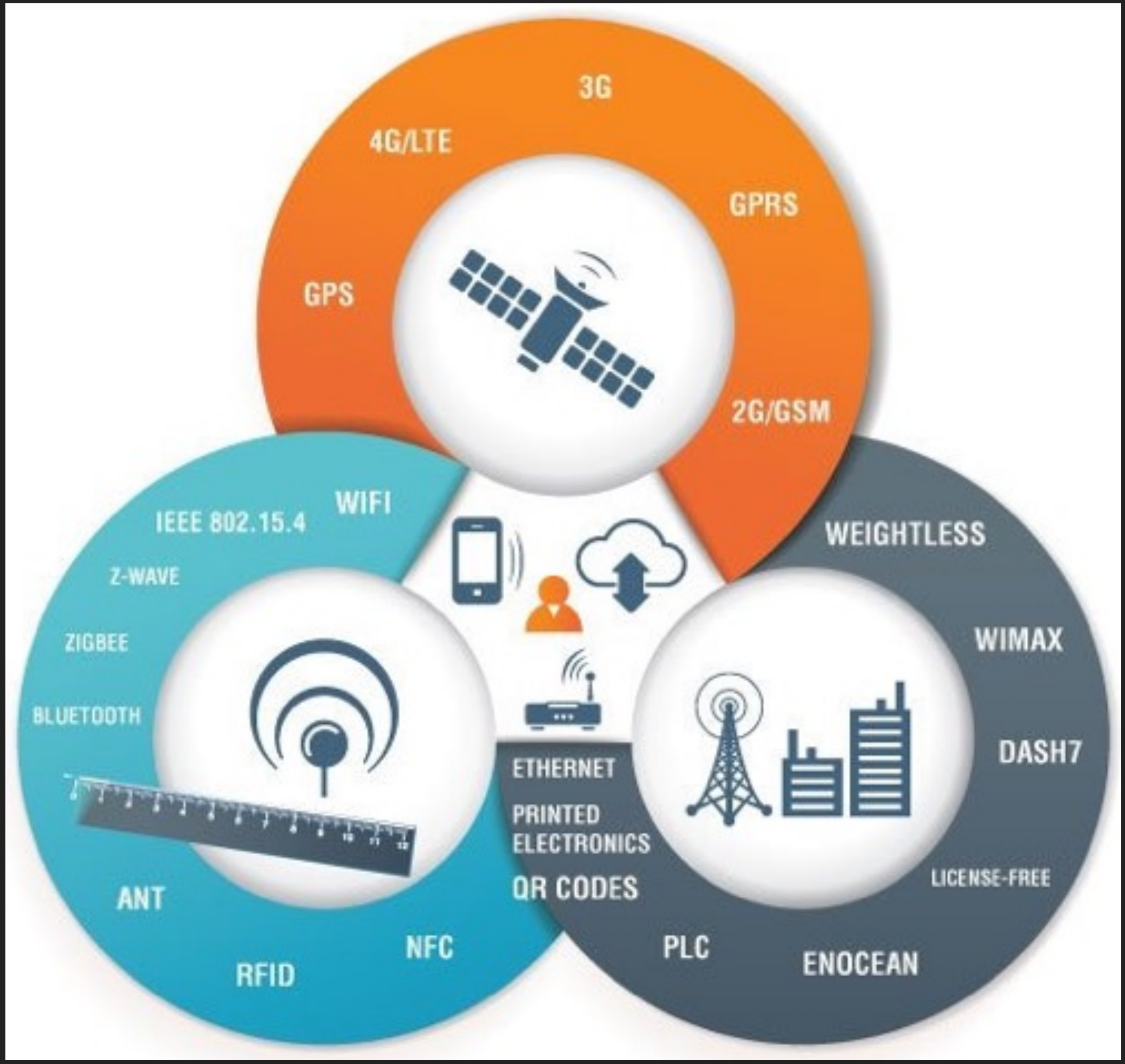
IOT INTERFACES

STATE OF THE ART

http://www.prismtechnology.com.au/wp-content/uploads/wireless.png

EXISTING IOT WIRELESS

- PAN-like technologies
  - 802.15.4 family
    - ZigBee, 6LoWPAN, Thread, etc.
  - Bluetooth
  - Bluetooth Low Energy
- Cellular
  - 2G (incl. GPRS/EDGE)
  - 3G
- Proprietary
  - OOK, ASK, FSK, etc.



http://postscapes.com/internet-of-things-technologies

EXISTING IOT WIRELESS

- PAN-like technologies
  - 802.15.4 family
    - ZigBee, 6LoWPAN, Thread, etc.
  - Bluetooth
  - Bluetooth Low Energy
- Cellular
  - 2G (incl. GPRS/EDGE)
  - 3G
- Proprietary
  - OOK, ASK, FSK, etc.

THEIR WEAKNESSES

- PAN-like technologies
  - Limited range
  - High-touch provisioning
- Cellular
  - Not inexpensive
  - Not fantastic on battery
- Proprietary
  - Some combination of the above
  - Insecurity by obscurity

WHICH IS BEST

DEPENDS

IF YOU NEED...

1. RANGE
2. COVERAGE
3. EASE OF INSTALLATION

EXISTING IOT WIRELESS

- PAN-like technologies
  - 802.15.4 family
    - ZigBee, 6LoWPAN, Thread, etc.
  - Bluetooth
  - Bluetooth Low Energy
- **Cellular**
  - 2G (incl. GPRS/EDGE)
  - 3G
- Proprietary
  - OOK, ASK, FSK, etc.

THEIR WEAKNESSES

- PAN-like technologies
  - Limited range
  - High-touch provisioning
- **Cellular**
  - Not inexpensive
  - Not fantastic on battery
- Proprietary
  - Some combination of the above
  - Insecurity by obscurity

Range, breadth, and ease

CELLULAR'S NOT GOING ANYWHERE*

*2G, HOWEVER, IS

DEPRECATION: A DEVELOPER'S CONUNDRUM

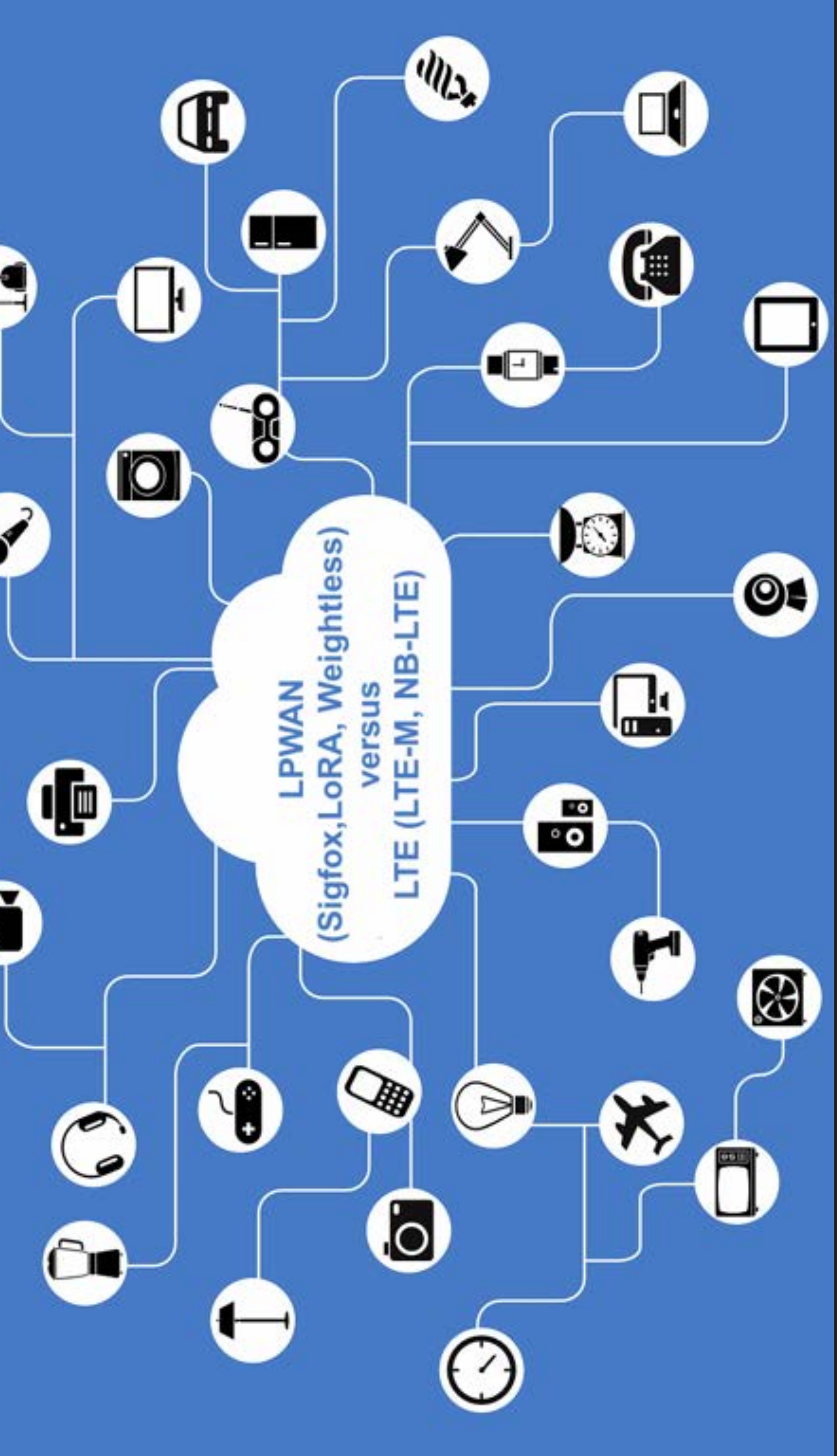
- AT&T to sunset 2G on January 1, 2017
- Other major carriers to follow

- 2G advantages: ubiquitous, battery-conscious, somewhat cheap
  - Exactly what IoT devices require

REPLACING 2G

- 3G
  - More expensive
  - Harder power requirements
- LTE-M/NB-LTE Release 13
  - IoT focused cellular protocols
  - Not ready by the sunset date, which means…

VOID IN MARKET



INTRODUCING

LPWANS

http://www.gsm-modem.de/M2M/wp-content/uploads/2015/10/LWPAN-versus-LTE.png

LPWAN

- LPWAN: Low Power Wide Area Network
- Like cellular, but optimized for IoT/M2M
  - Network of basestations worldwide
  - Star network to endpoints, UL/DL traffic
  - Range in miles

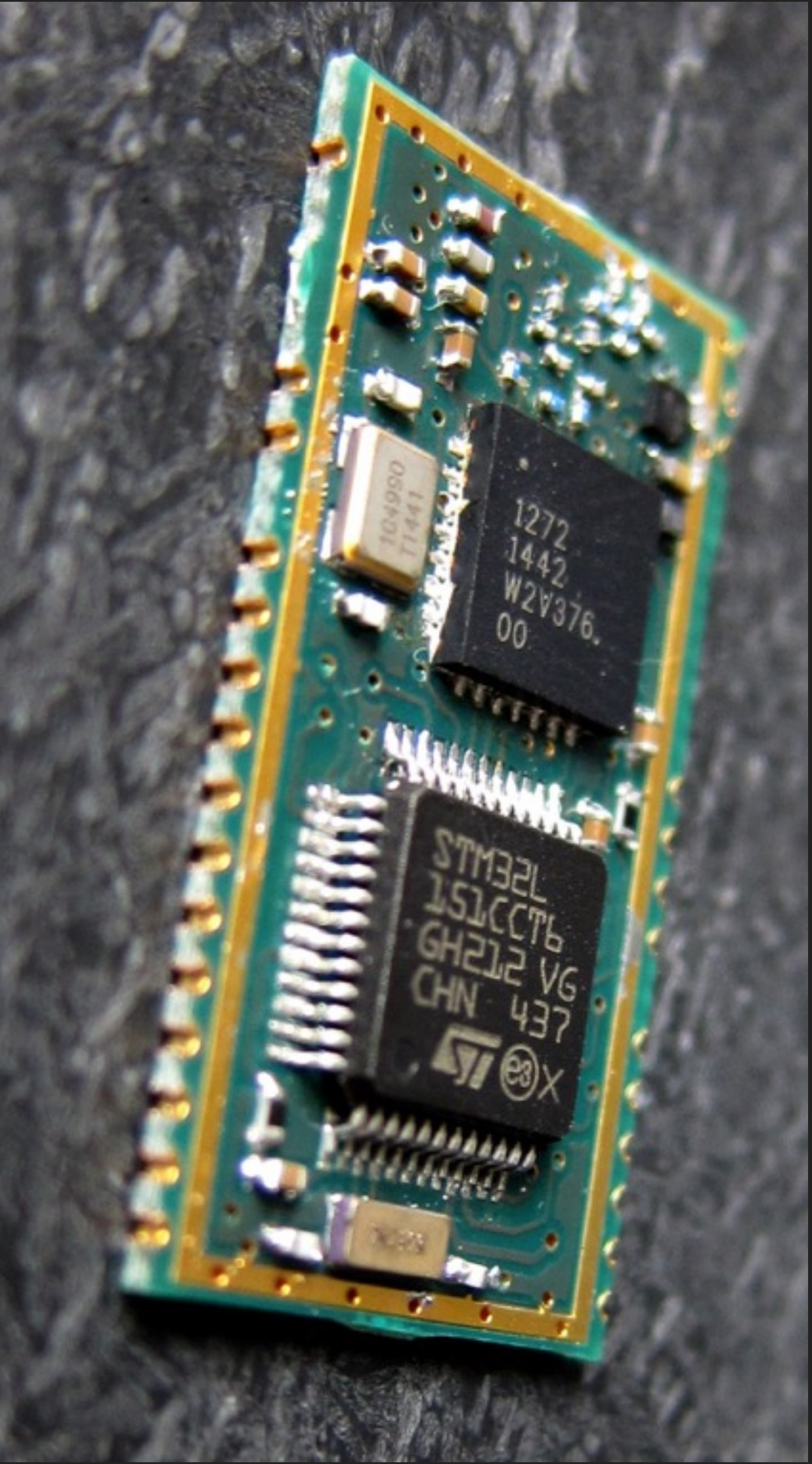
EMERGING STANDARDS

http://www.itwissen.info/bilder/lpwan-konzepte-fuer-das-internet-of-things-iot.png

AGGRESSIVE INVESTMENT

- SIGFOX raised 115MM last year
  - WSJ: Possible US IPO soon
- Senet and Actility, LoRa backers, raised a combined 51MM
- LoRa alliance membership tripled last year

http://img.scoop.it/pkne9jh9_fl-PFplrbFgL4XXXL4j3HpexhjNOf_P3YmryPKwJ94QGRtDb3Sbc6KY
https://pbs.twimg.com/profile_images/615790171116601345/Q1PU_OuW.png



LPWANS

THE STACKS

http://rethink-iot.com/wp-content/uploads/2015/10/MM002-resized.jpg

OPTIMIZED FOR IOT

- Battery-conscious
  - SIGFOX advertises 10 years on 1 AA battery
- Long range
  - LoRa advertises up to 13.6 miles

HOW!?

- Compare this with…
  - 2G: typically 1-2 miles, max 22 miles, a few days
  - 802.15.4: 10-100 meters, months-years
  - WiFi: 30 meters, a few days

COMPROMISES

EMBRACING COMPROMISE

- Conservative duty-cycling and listening
- Very sparse datagrams
- Highly rate-limited

EMBRACING COMPROMISE

- Examples
  - SIGFOX limits devices to 140 12-byte datagrams per day
  - Weightless-N is uplink-only
  - LoRa Class A devices can only receive downlink momentarily after uplinking

RADICALLY DIFFERENT

INTRODUCING

LORA

HISTORY

- LPWAN developed by Semtech
- PHY patented in June 2014
- LoRaWAN MAC/NWK stack released in January 2015
- Supported by LoRa Alliance

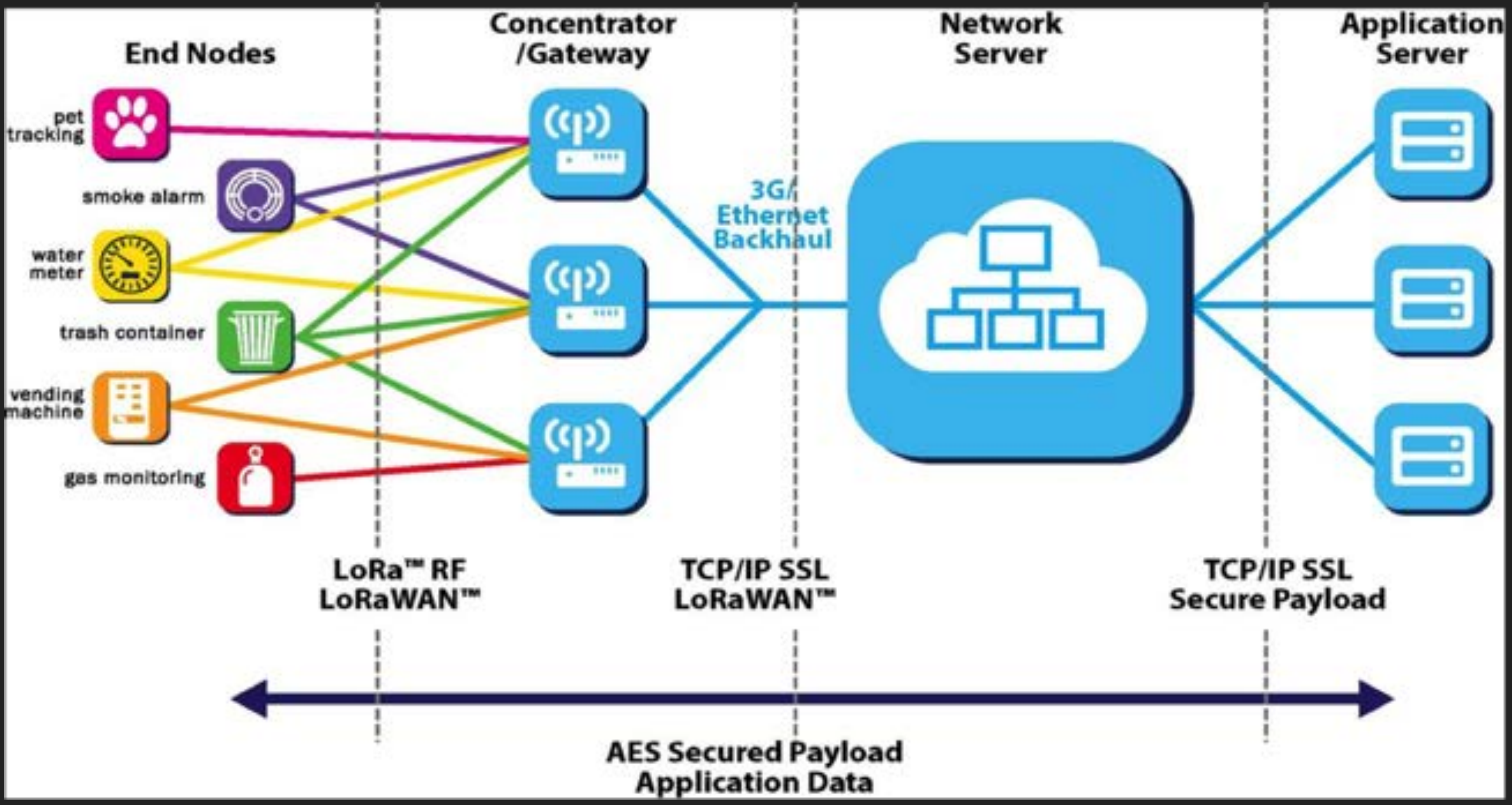
http://actility.thingpark.com/thingpark-store/87-thickbox_leoelec/thingpark-cloud-application-partner-kit-lora-sensors.jpg

NOMENCLATURE

- LoRa vs. LoRaWAN
  - LoRa: PHY layer
  - LoRaWAN: MAC, NWK, and APP built on LoRa

LORAWAN NETWORK TOPOLOGY

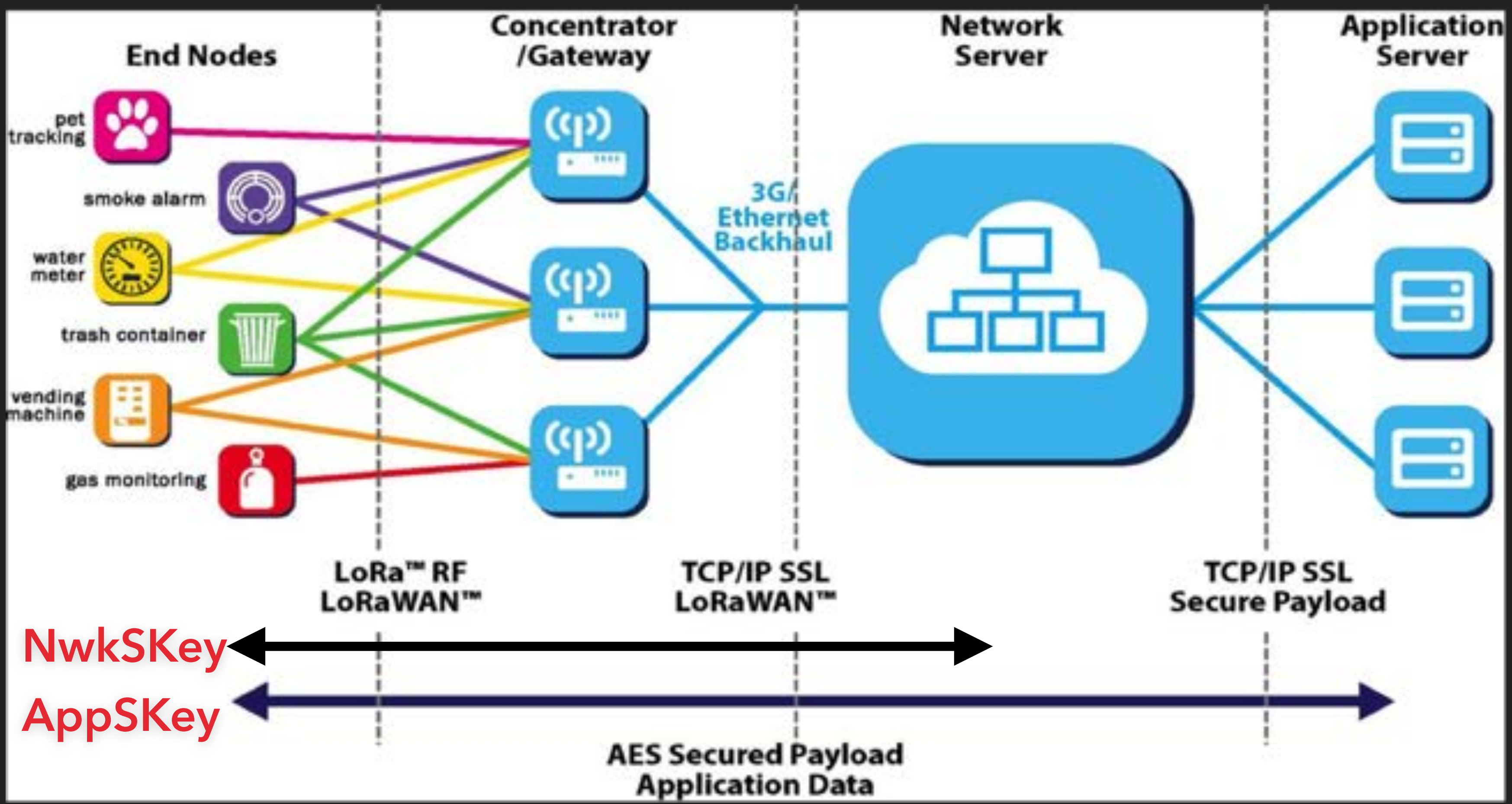
- Gateway: low-IQ basestation / LoRa concentrator
- Network Server: routes between gateways and application servers
  - Roaming being defined
- Application Servers



http://www.nnnco.com.au/uploads/6/0/7/9/60799127/4996599_orig.png

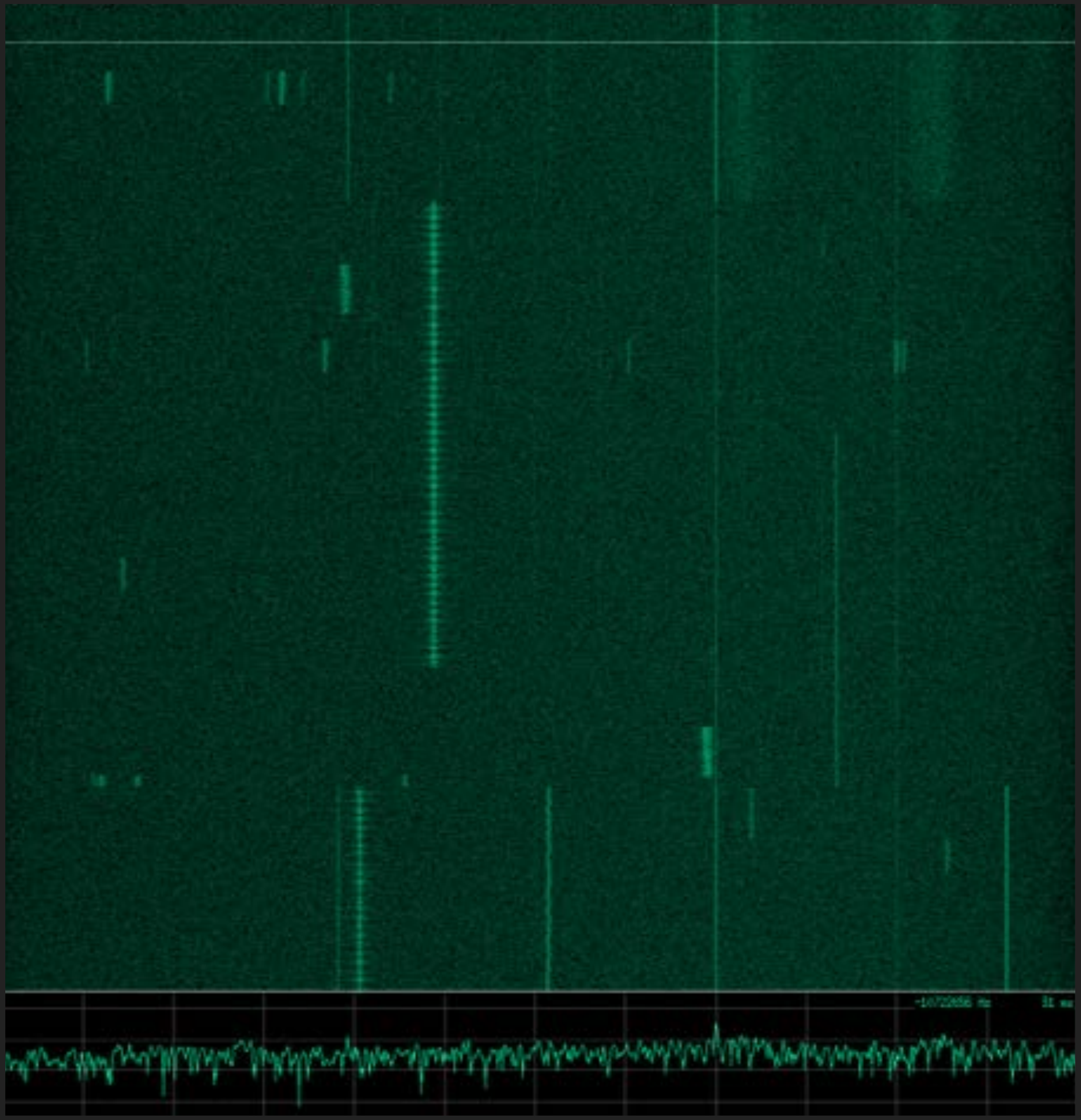
LORAWAN SECURITY ARCHITECTURE

- NwkSKey: end device to network server
- AppSKey: end device to application server
- Keys can be preconfigured or sent OTA



http://www.nnnco.com.au/uploads/6/0/7/9/60799127/4996599_orig.png

LICENSURE, OR LACK THEREOF



- 900 MHz ISM
  - US: 902-928 MHz
  - EU: 868 MHz
- No special license required

LICENSURE, OR LACK THEREOF

DMA[1]	Call Sign	Move Off-Air	Move to Low VHF	Move to High VHF
New York, NY	WABC-TV	[illegible]	351,539,055	NA
New York, NY	WCBS-TV	900,000,000	675,000,000	360,000,000
New York, NY	WDVB-CD	[illegible]	318,579,075	169,908,840
New York, NY	WEBR-CD	376,508,700	282,381,525	150,603,480
New York, NY	WEDW	448,103,700	336,077,775	179,241,480
New York, NY	WEPT-CD	129,438,900	97,079,175	51,775,560
New York, NY	WFTY-DT	438,170,400	328,627,800	175,268,160
New York, NY	WFUT-DT	788,135,400	591,101,550	315,254,160
New York, NY	WJLP	206,954,325	NA	NA
New York, NY	WLIW	672,446,700	504,335,025	268,978,680
New York, NY	WLNY-TV	483,707,700	362,780,775	193,483,080
New York, NY	WMBC-TV	805,447,800	604,085,850	322,179,120
New York, NY	WMBQ-CD	445,109,400	333,832,050	178,043,760
New York, NY	WMUN-CD	364,589,100	273,441,825	145,835,640
New York, NY	WNBC	869,160,600	651,870,450	347,664,240
New York, NY	WNET	443,269,260	258,573,735	NA
New York, NY	WNJB	475,608,780	277,438,455	NA
New York, NY	WNJN	775,742,400	581,806,800	310,296,960
New York, NY	WNJU	819,342,000	614,506,500	327,736,800
New York, NY	WNYE-TV	770,259,600	577,694,700	308,103,840
New York, NY	WNYJ-TV	625,505,400	469,129,050	250,202,160
New York, NY	WNYW	824,922,000	618,691,500	329,968,800
New York, NY	WPIX	435,535,920	254,062,620	NA
New York, NY	WPXN-TV	745,848,900	559,386,675	298,339,560
New York, NY	WRNN-TV	846,911,700	635,183,775	338,764,680
New York, NY	WTBY-TV	683,951,400	512,963,550	273,580,560
New York, NY	WVVH-CD	31,156,200	23,367,150	12,462,480
New York, NY	WWOR-TV	810,039,600	607,529,700	324,015,840
New York, NY	WXTV-DT	832,842,000	624,631,500	333,136,800
New York, NY	WZME	483,669,000	362,751,750	193,467,600
Los Angeles, CA	KABC-TV	305,469,360	178,190,460	NA
Los Angeles, CA	KAZA-TV	557,458,200	418,093,650	222,983,280
Los Angeles, CA	KBEH	617,519,700	463,139,775	247,007,880
Los Angeles, CA	KCAL-TV	299,840,400	174,906,900	NA
Los Angeles, CA	KCBS-TV	544,991,400	408,743,550	217,996,560
Los Angeles, CA	KCET	492,480,900	369,360,675	196,992,360
Los Angeles, CA	KCOP-TV	333,967,860	194,814,585	NA
Los Angeles, CA	KDOC-TV	539,584,200	404,688,150	215,833,680
Los Angeles, CA	KFTR-DT	573,801,300	430,350,975	229,520,520
Los Angeles, CA	KHTV-CD	457,959,600	343,469,700	183,183,840

- Compare this with cellular
  - FCC auctions cellular spectrum licenses for billions
  - Restricts building infrastructure to biggest telcos
- Left: opening bid list for FCC TV whitespace reverse auction

LORAWAN NETWORK PROVIDERS

- Senet
  - Commercial network
- The Things Network
  - Crowdsourced
- No licensed spectrum required…!!

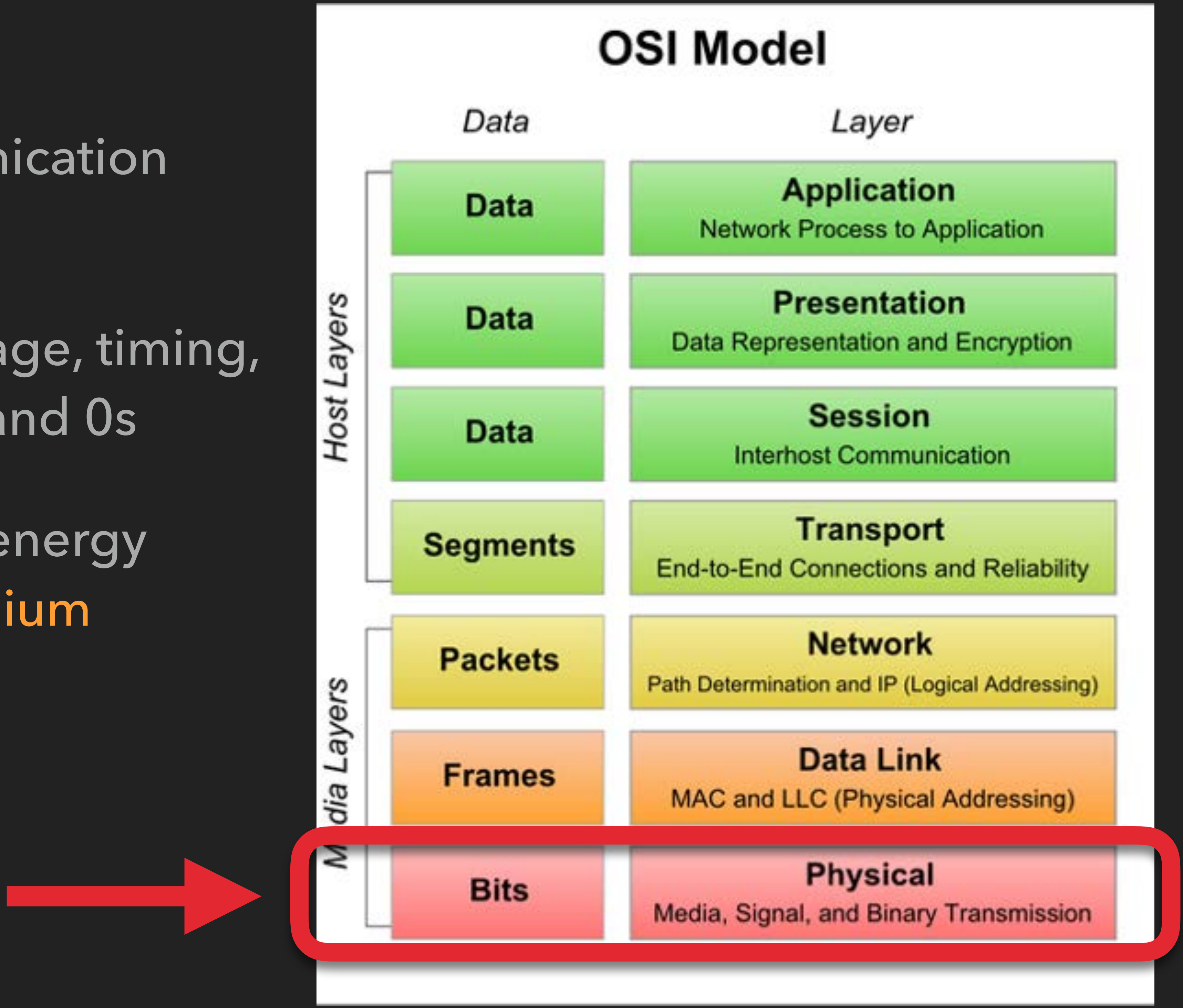
http://img.scoop.it/pkne9jh9_fl-PFplrbFgL4XXXL4j3HpexhjNOf_P3YmryPKwJ94QGRtDb3Sbc6KY
http://thenextweb.com/insider/2015/08/19/the-things-network-wants-to-make-every-city-smart-starting-with-amsterdam/#gref

OBSCENELY SHORT

RADIO CRASH COURSE

PHY LAYER

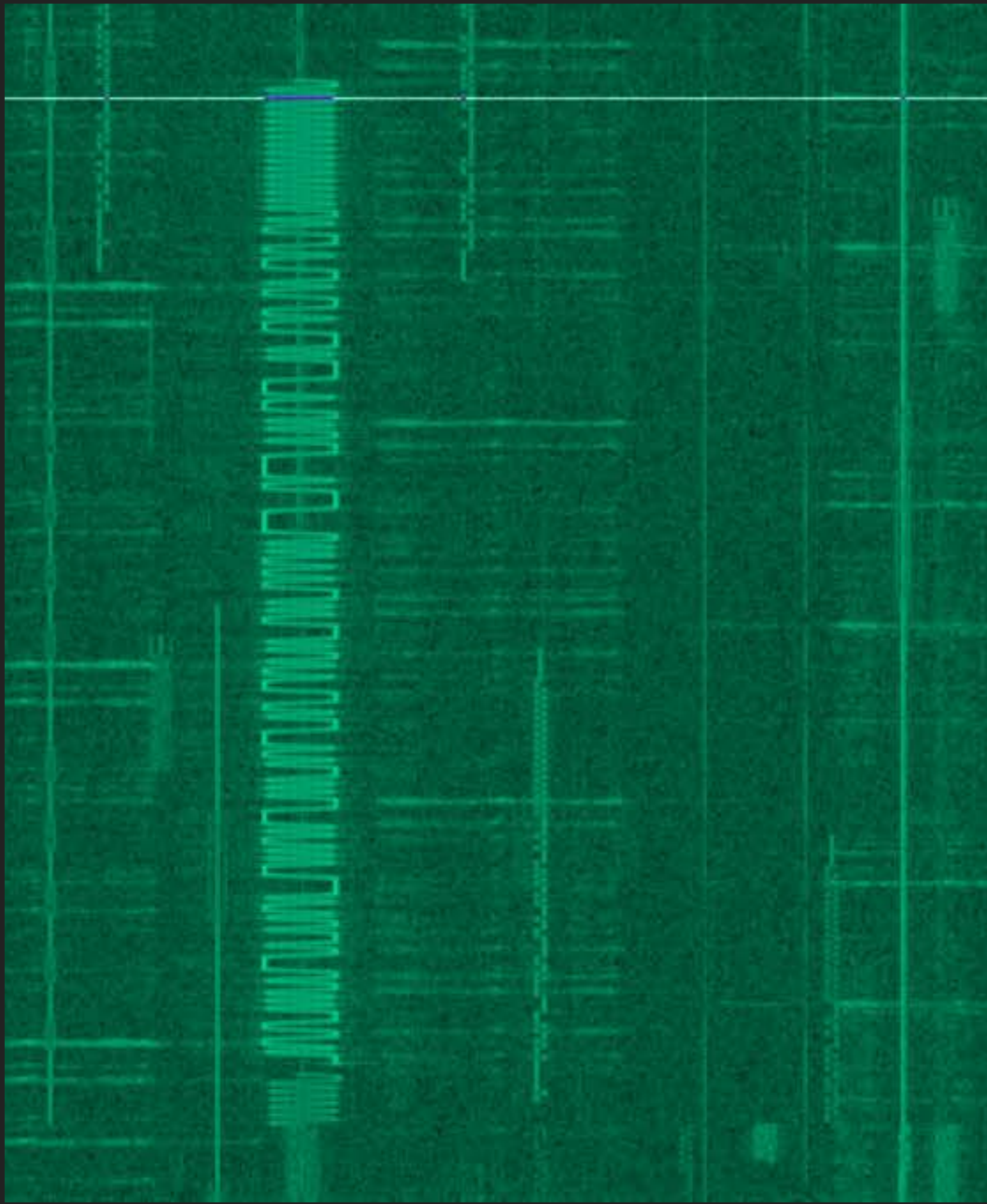
- Lowest layer in communication stack
- In wired protocols: voltage, timing, and wiring defining 1s and 0s
- In wireless: patterns of energy being sent over RF medium



http://www.tech-faq.com/wp-content/uploads/2009/01/osimodel.png

WHAT IS RF?

- "Radio Frequency"
- Electromagnetic waves
- Energy

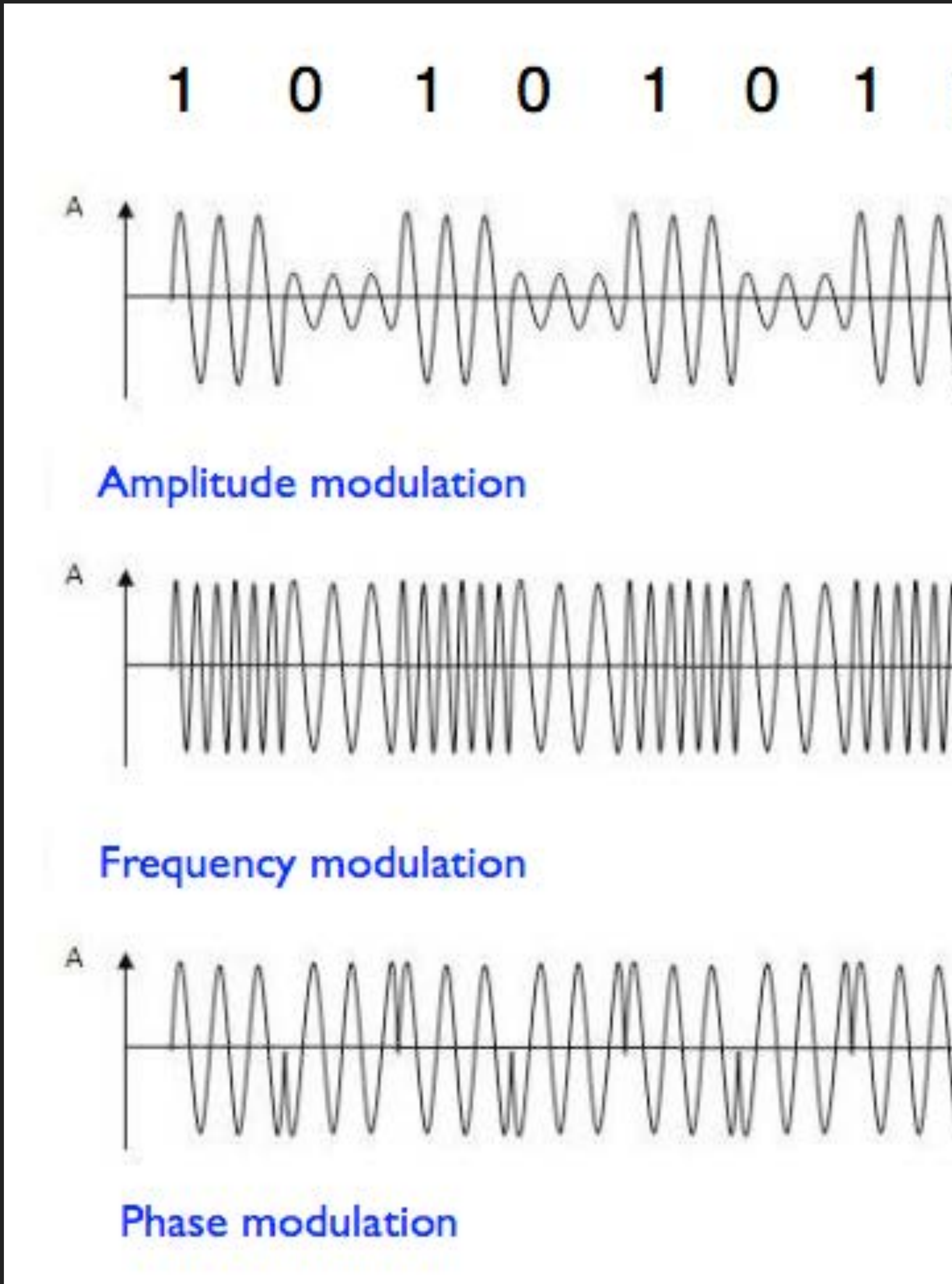


MANIPULATING RF

- Done with a radio
- Hardware defined
  - RF and protocol in silicon
- Software defined
  - Flexible silicon handles RF
  - Protocol-specific components implemented in software (CPU or FPGA)

PHY COMPONENTS

- Modulation
  - How digital values are mapped to RF energy
- RF parameters that can be modulated:
  - Amplitude
  - Frequency
  - Phase
  - some combination of the above



http://xenon.colorado.edu/spotlight/kb/gps_basics/modulations.001.png

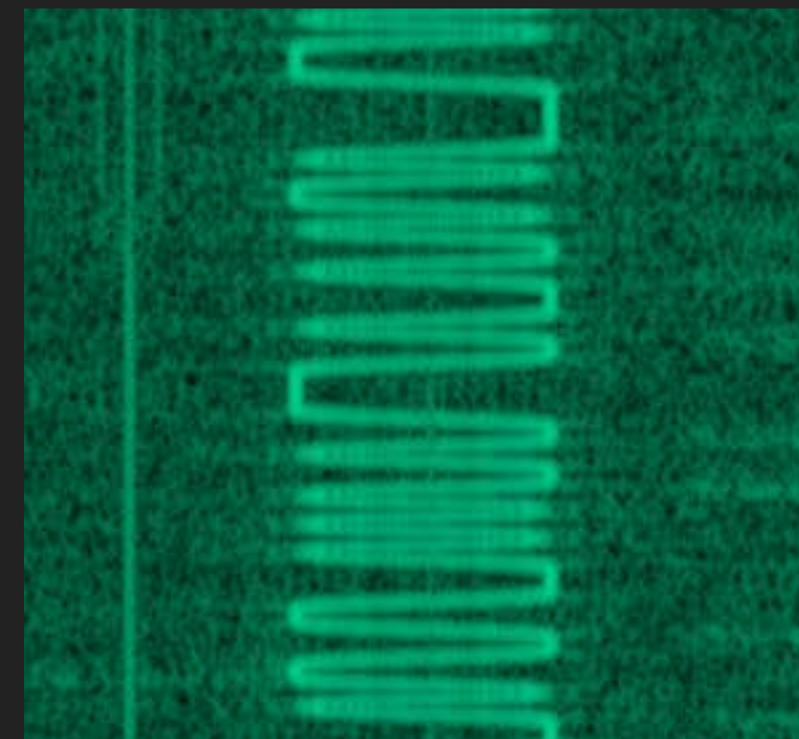
MODULATION

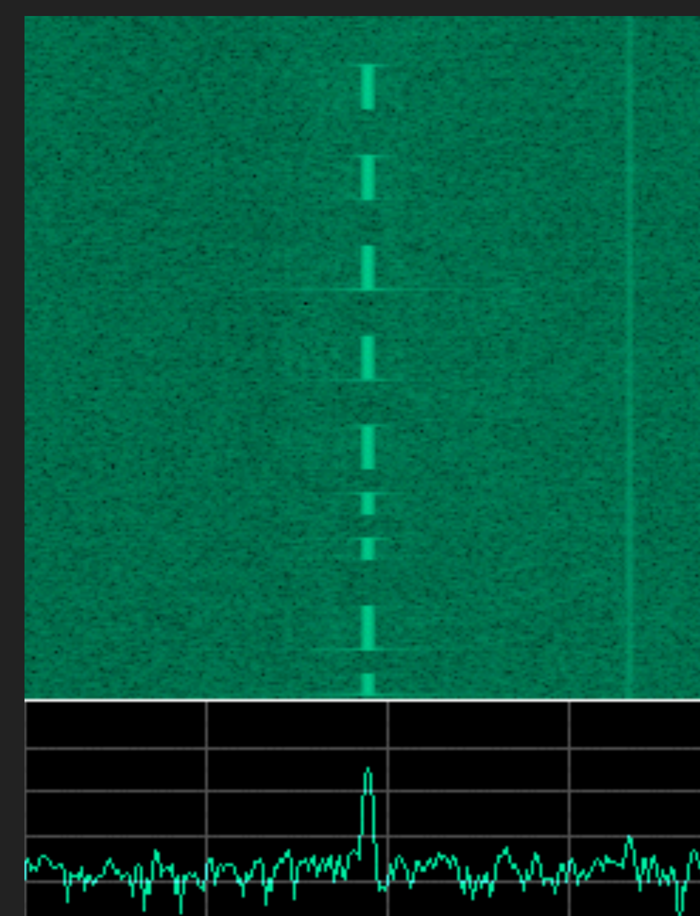
- Modulators can modulate analog or digital information

- Digital modulation
  - Symbols: RF energy state representing some quantity of information

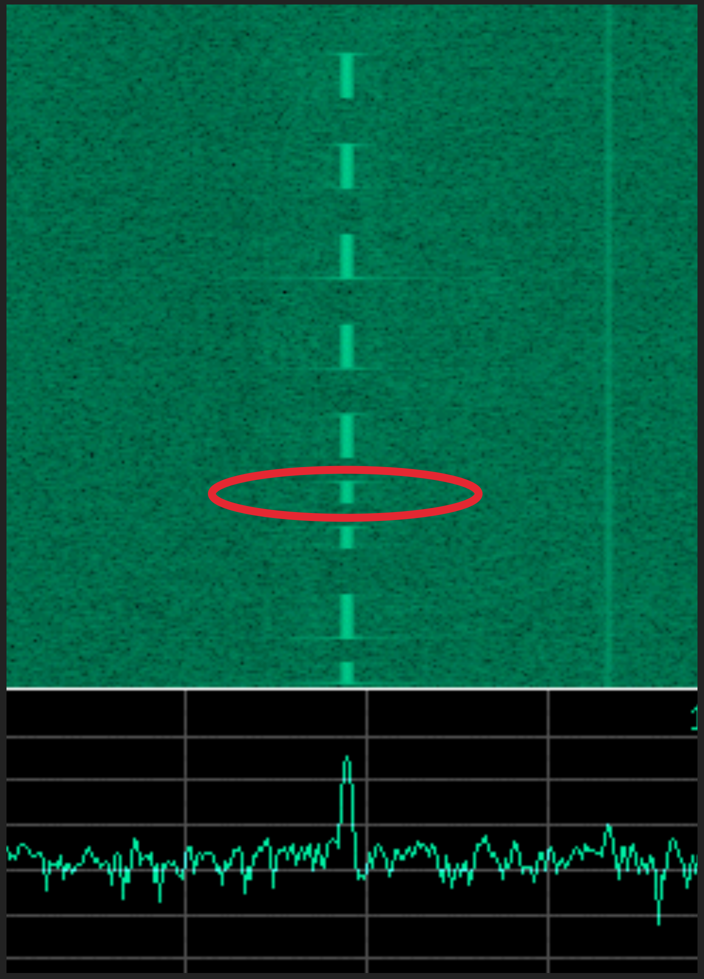
COMMON IOT PHYS

- FSK/GFSK
  - RF energy alternates between two frequencies to signify digital values
- ASK/OOK
  - Changes in RF power on a certain frequency signify digital values





http://3.bp.blogspot.com/-w66qwKucSJI/UgWKKmPUP2I/AAAAAAAADYA/B9NMGYzqJVk/s1600/Screenshot-2013-08-03-04-41-52.png



SYMBOLS ILLUSTRATED

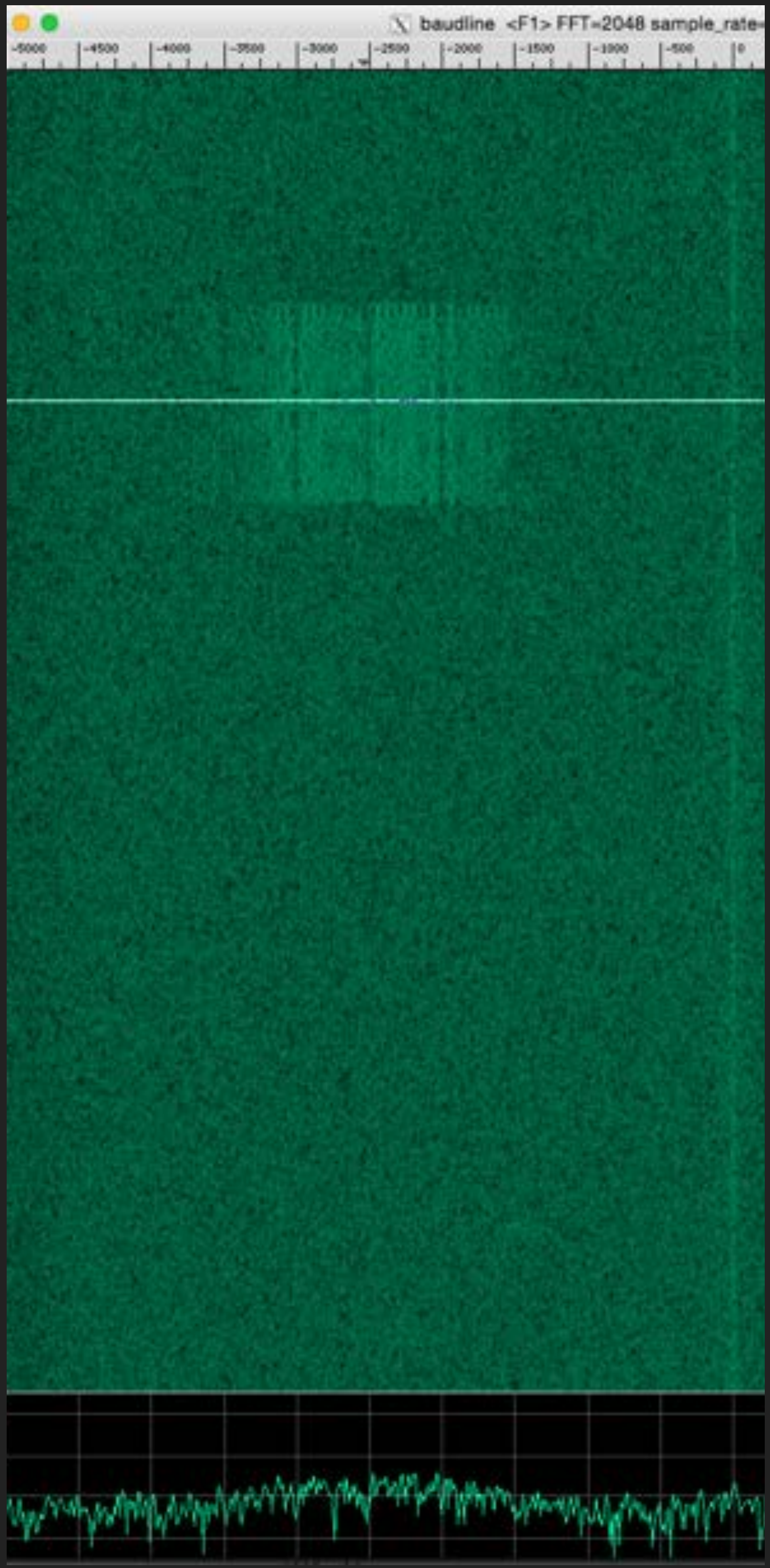
- Time domain
- Top: FSK
- Bottom: OOK/ASK

- Compare with analog modulation
  - Analog = infinite possible symbols
  - Digital = finite number of possible symbols, defined by modulation

http://3.bp.blogspot.com/-w66qwKucSJI/UgWKKmPUP2I/AAAAAAAADYA/B9NMGYzqJVk/s1600/Screenshot-2013-08-03-04-41-52.png

MORE COMPLICATED IOT PHYS

- Spread spectrum
  - Data bits are encoded at a higher rate and occupy more spectrum
  - Resilient to RF noise
- Examples
  - 802.15.4
  - Bluetooth
  - Bluetooth Low Energy

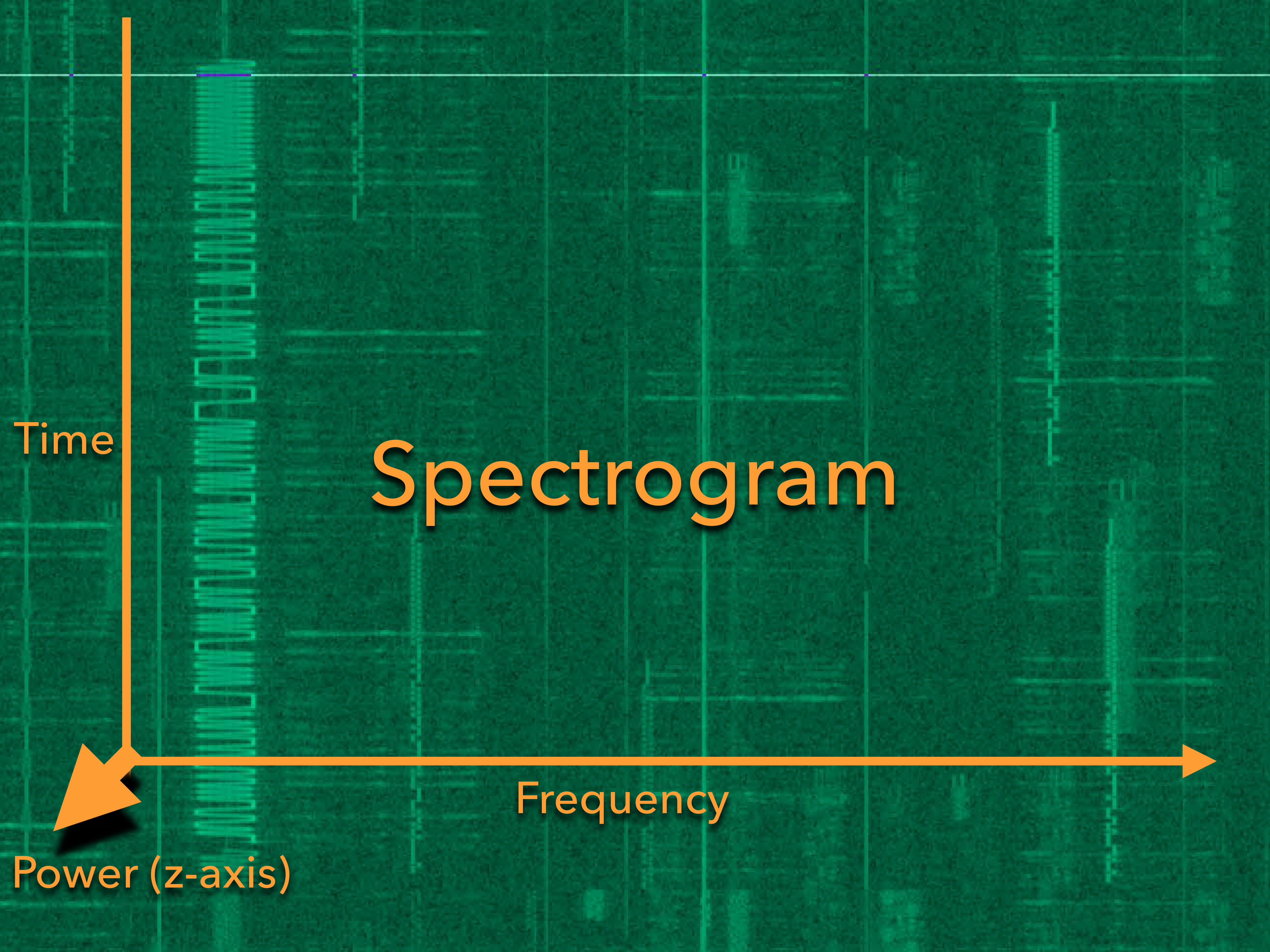


TOOLS USED IN THIS TALK

- Transmitter: Microchip LoRa RN2903 Module
  - Contains hardware-defined Semtech LoRa radio
- Receiver: Ettus B210
  - Software-defined radio
  - Python, GNURadio, and Baudline to process

LAST THING… FFT

- Fast Fourier Transform
- Decomposes a signal into its component frequencies
  - Any periodic signal can be modeled as the sum of harmonic sine waves
- Allows analysis and visualization of frequency domain



Spectrogram
Time
Frequency
Power (z-axis)

TECHNICAL DETAILS

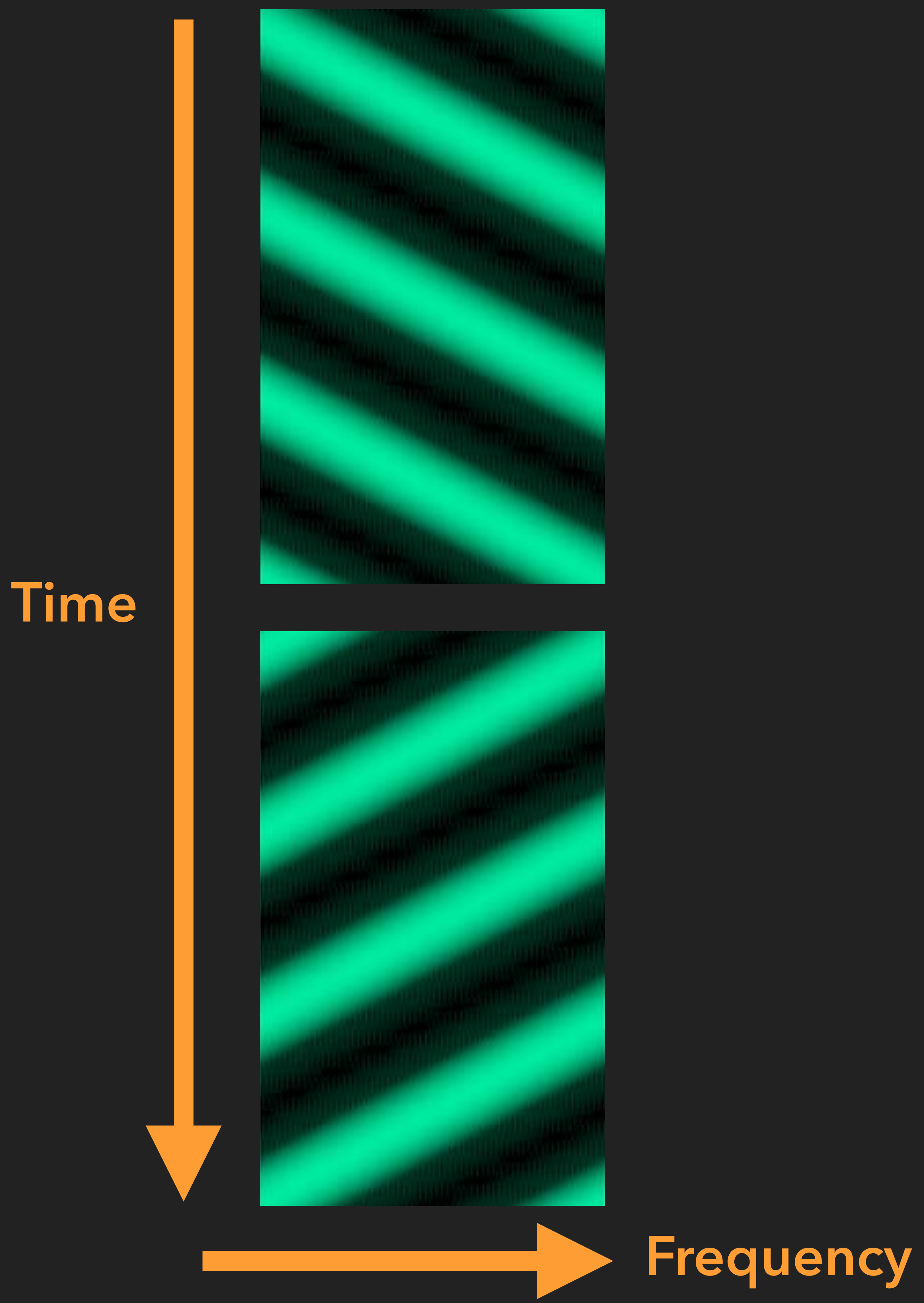
LORA

LORA'S PROPRIETARY PHY

- Modulation: Chirp Spread Spectrum (CSS)
- What's a chirp?
  - A signal of continuously increasing or decreasing frequency
  - i.e. a "sweeped tone"

CSS CHIRPS

- **Upchirp** (top)
  - Increasing frequency
- **Downchirp** (bottom)
  - Decreasing frequency



CSS ADVANTAGES

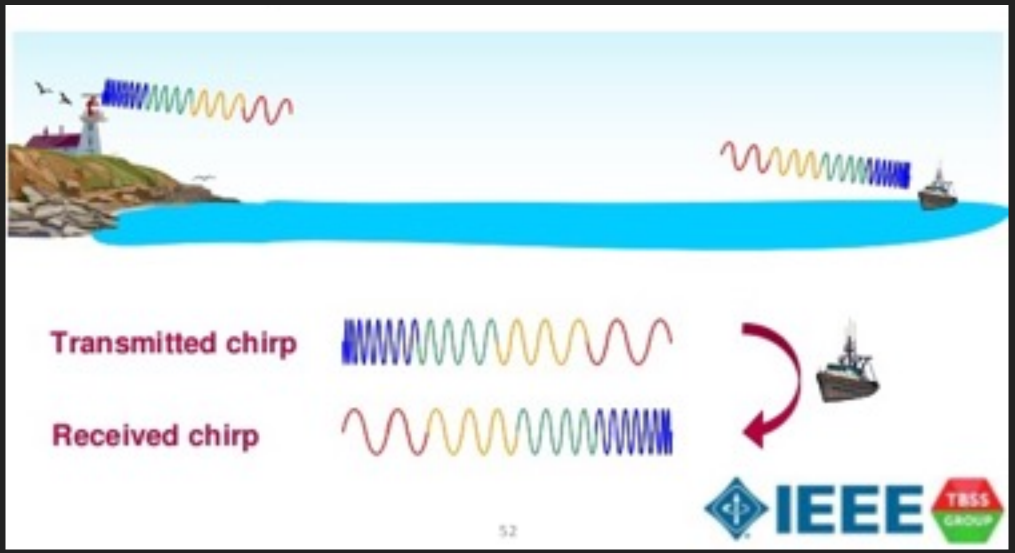
- Great link budget
  - Resilience to interference
  - Performance at low power
- Resistance to multi-path effects
- Resistance to Doppler effect (mobile applications)

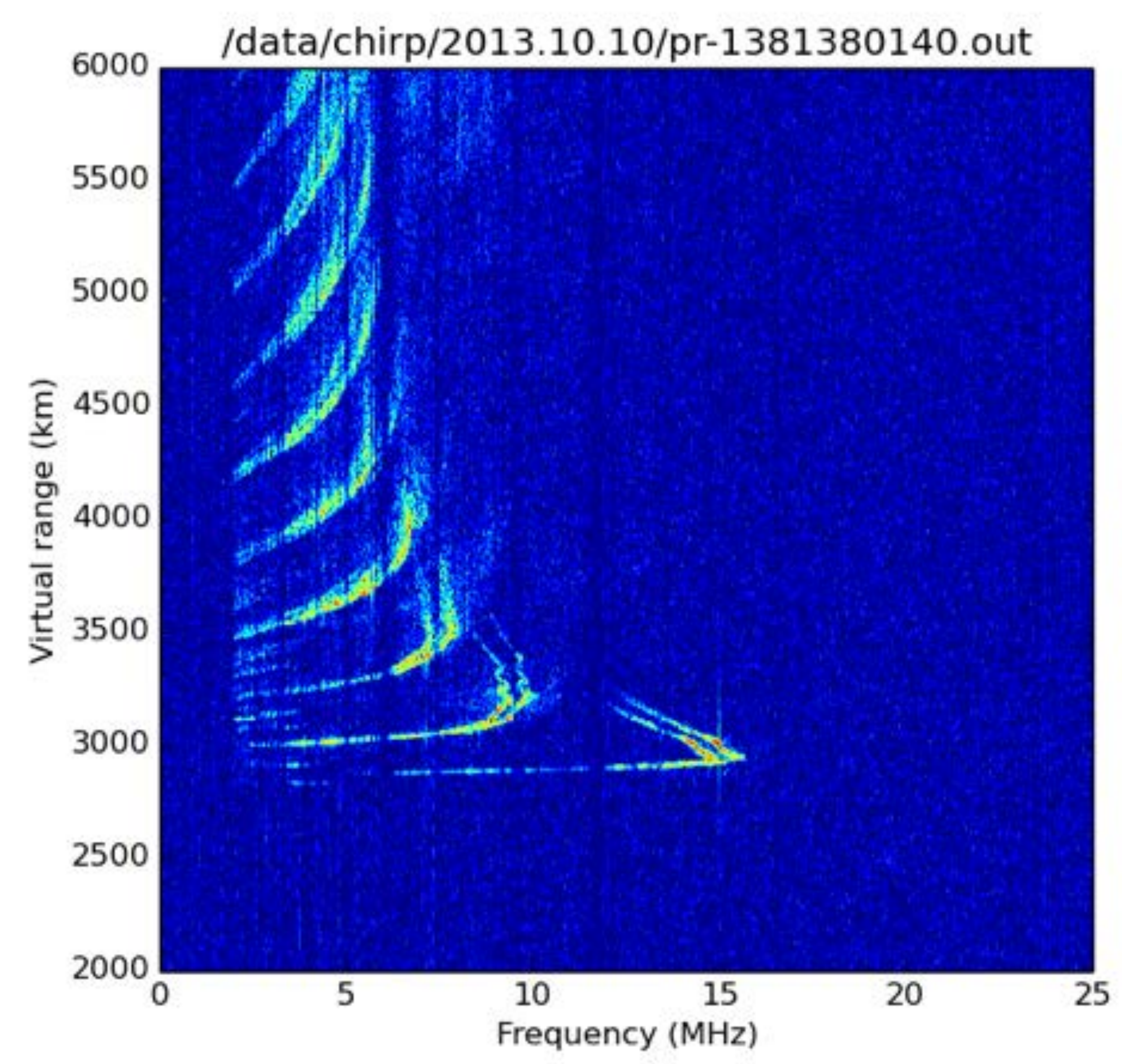
- Interesting set of pros… where else are chirps used?

RADAR

CHIRPS IN RADAR

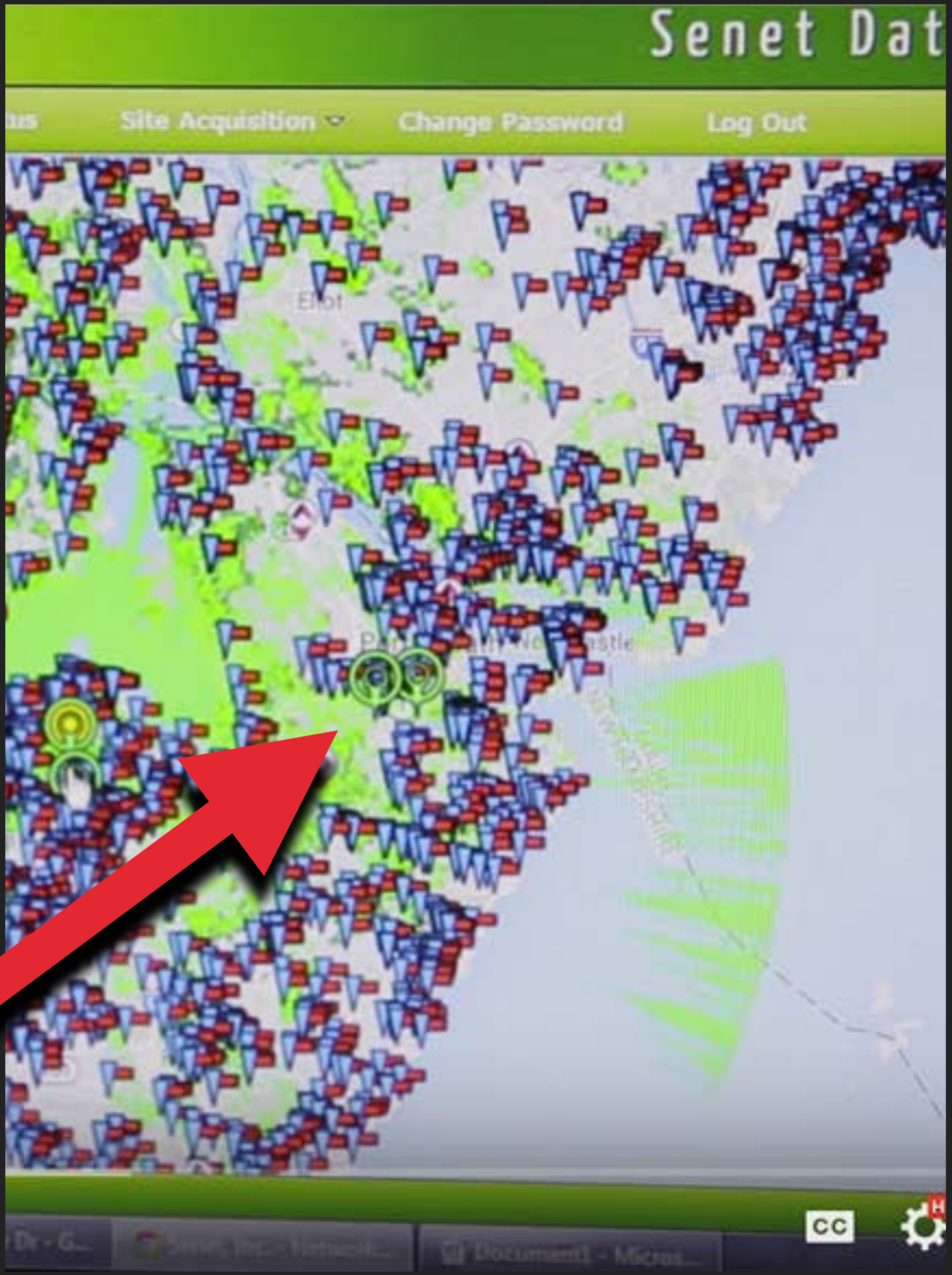
- Various military and marine radars
  - Wideband and pulse compression
- Open source GNU Chirp Sounder
  - Ionospheric radars
  - Space weather





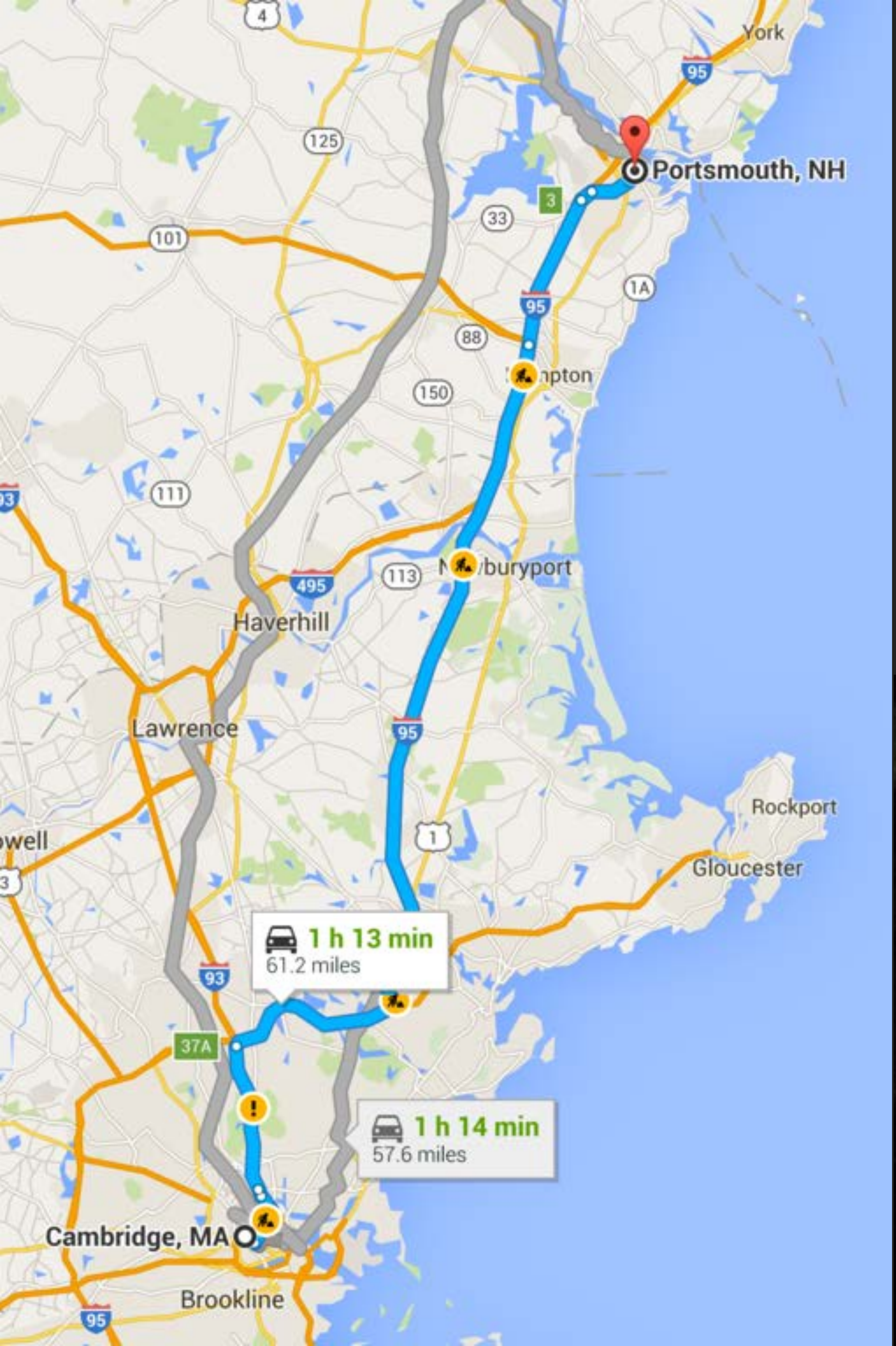
SEEKING LORA

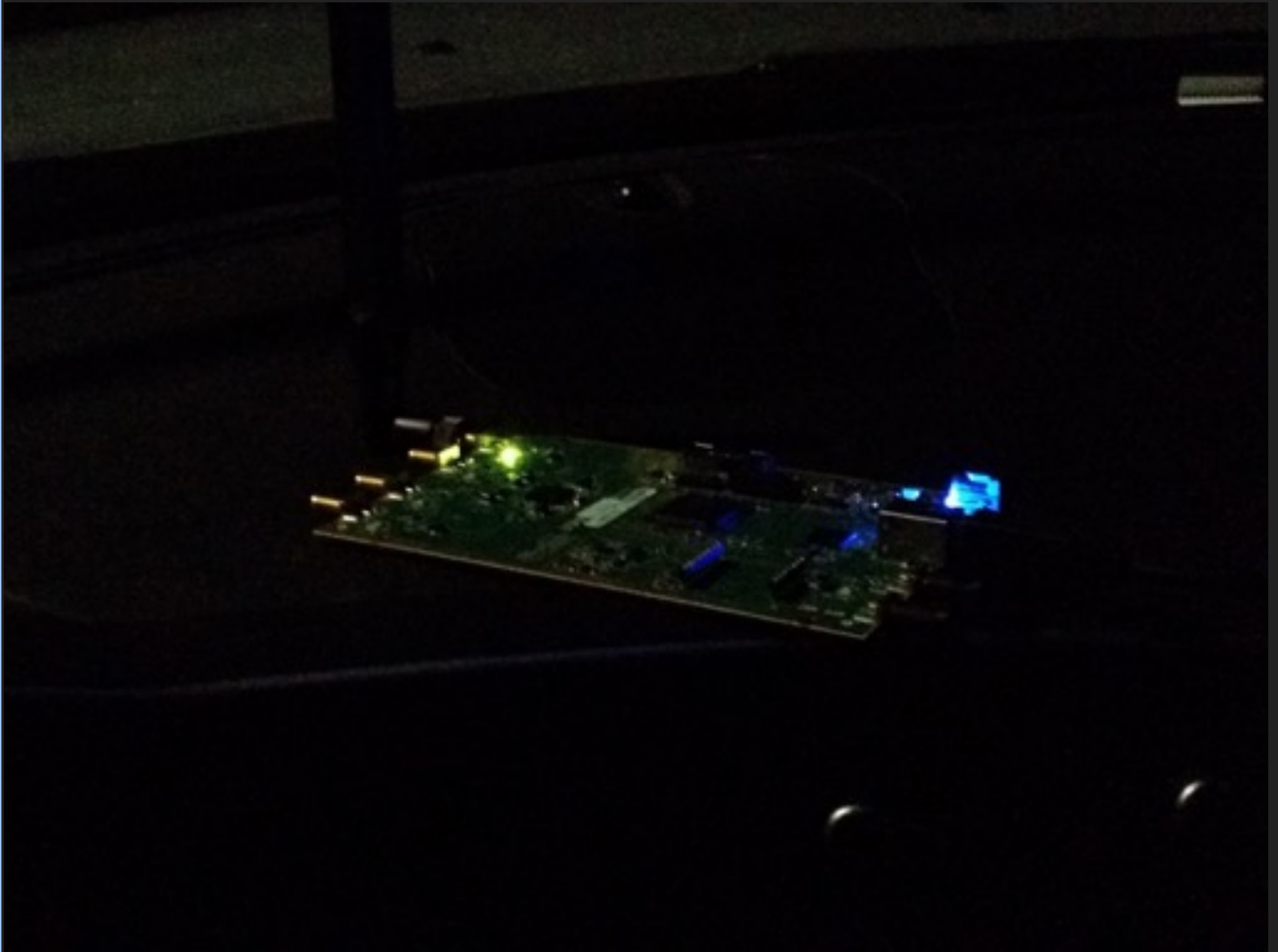
- December 2015: No LoRa sightings in Boston, Atlanta, San Francisco, or New York
- I encountered Senet at a Meetup event in Cambridge, MA
- While watching one of their marketing videos…



Portsmouth, NH!

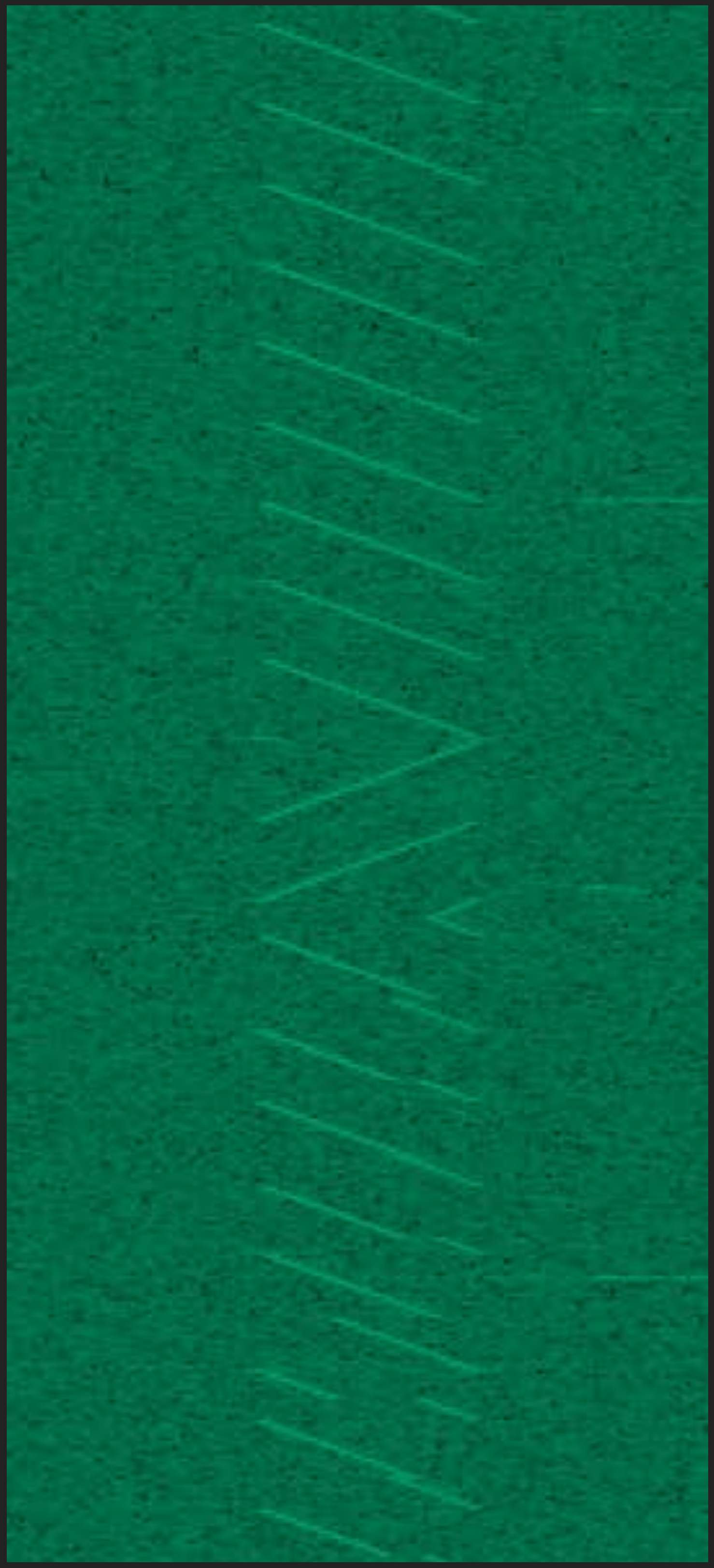
ROAD TRIP


York
Portsmouth, NH
Haverhill
Lawrence
Rockport
Gloucester
1 h 13 min
61.2 miles
1 h 14 min
57.6 miles
Cambridge, MA
Brookline



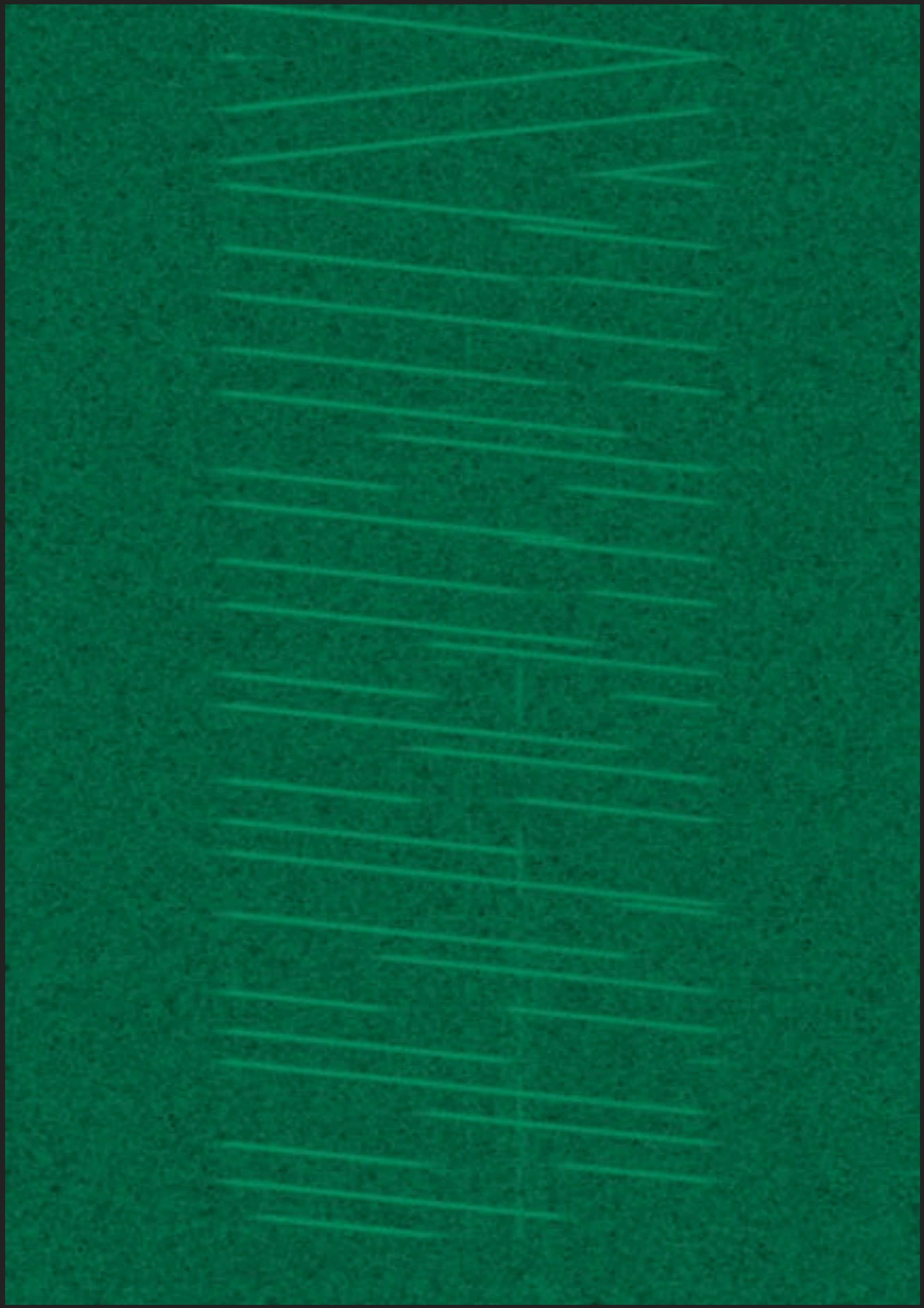
EXAMINING THE LORA PHY FRAME

- Repeated upchirps
  - Preamble/Training Sequence
- Two downchirps
  - Start of frame delimiter (SFD)
- Choppy upchirps of varying length
  - Data!



PHY DATA UNIT STRUCTURE

- Chirp frequency is static
- Chirp “jumps” throughout band
- Instantaneous frequency changes are result of data being modulated onto the chirps
  - Chirp “phase”



DEMODULATING

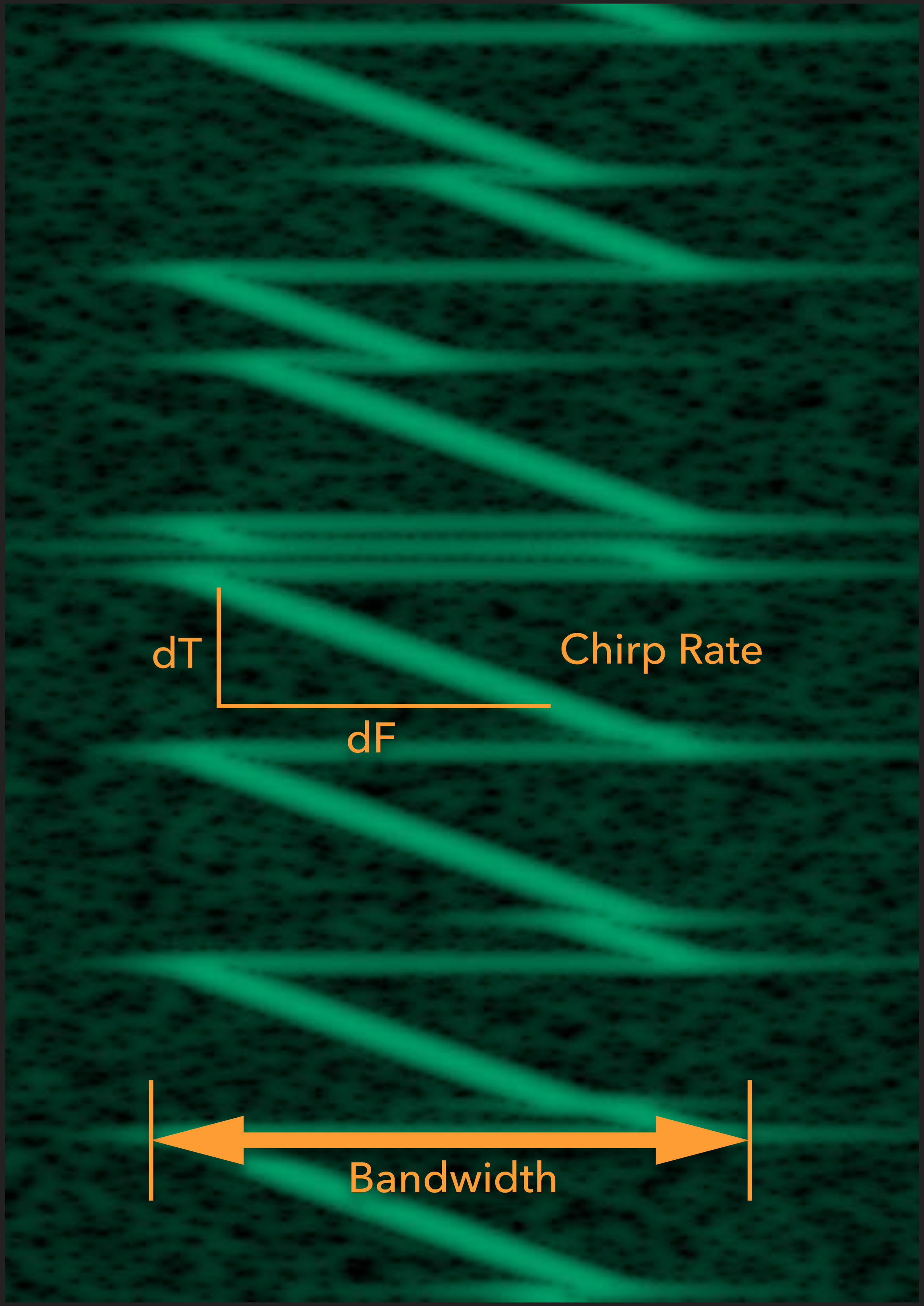
LORA

BEFORE WE GET STARTED… WHAT DO WE KNOW?

- Technical documentation
  - Semtech European patent application 13154071.8
  - LoRa Alliance LoRaWAN spec (MAC/NWK only, not a PHY spec)
  - Semtech app notes AN1200.18 and AN1200.22
- Prior art
  - Partial implementation in open source rtl-sdrangelove
  - Observations at https://revspace.nl/DecodingLora

SOME DEFINITIONS…

- Bandwidth: width of spectrum occupied by chirp
- Spreading factor: number of bits encoded per symbol (RF state, remember?)
- Chirp rate: first derivative of chirp frequency



SOME DEFINITIONS…

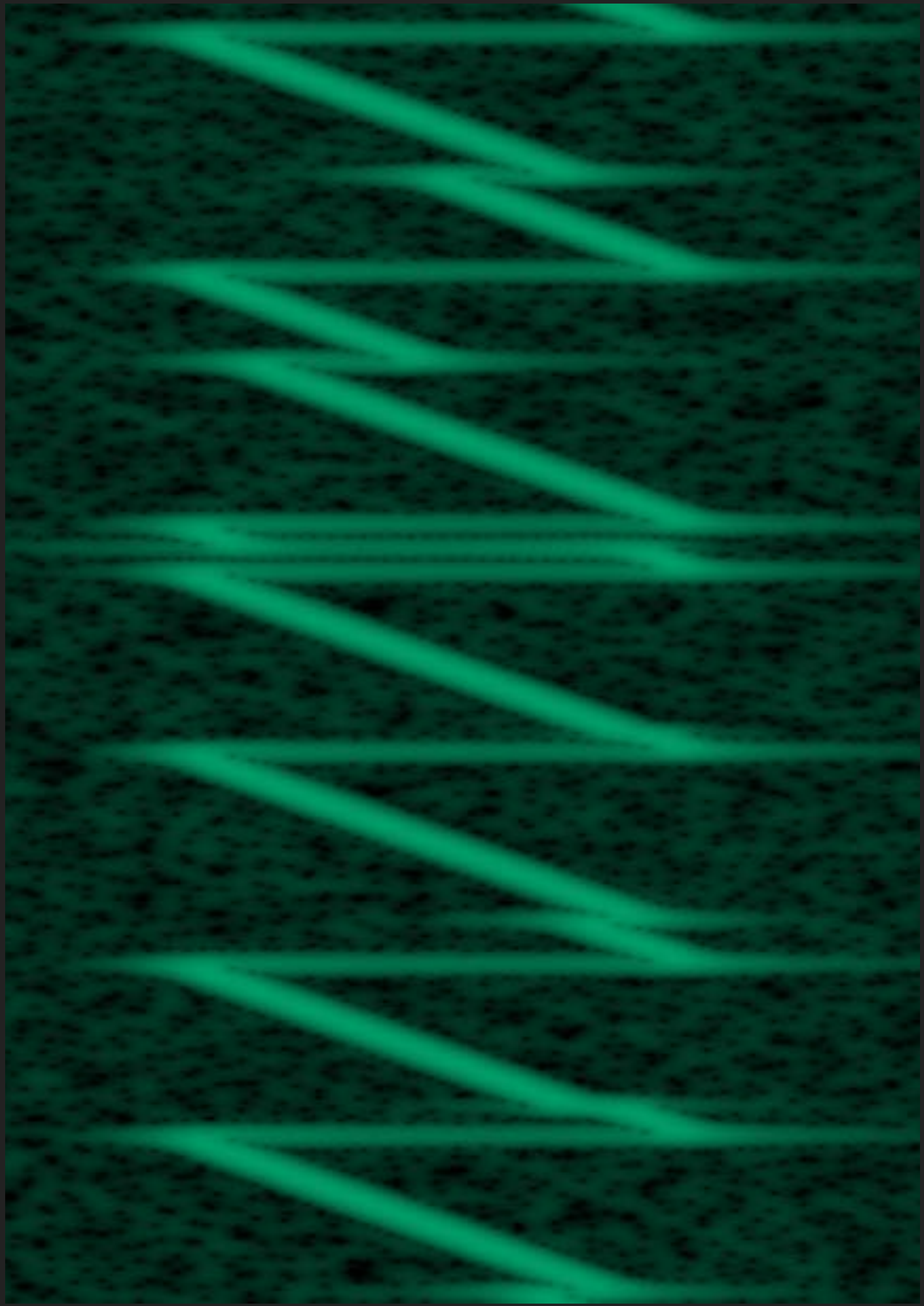
- Bandwidth: width of spectrum occupied by chirp
- Spreading factor: number of bits encoded per symbol (RF state, remember?)
- Chirp rate: first derivative of chirp frequency

SOME NUMBERS…

- US: 125kHz, 250kHz, 500kHz
- US: [7-12] bits per symbol
- bandwidth/(2**spreading_factor)

SO WHAT'S A SYMBOL?

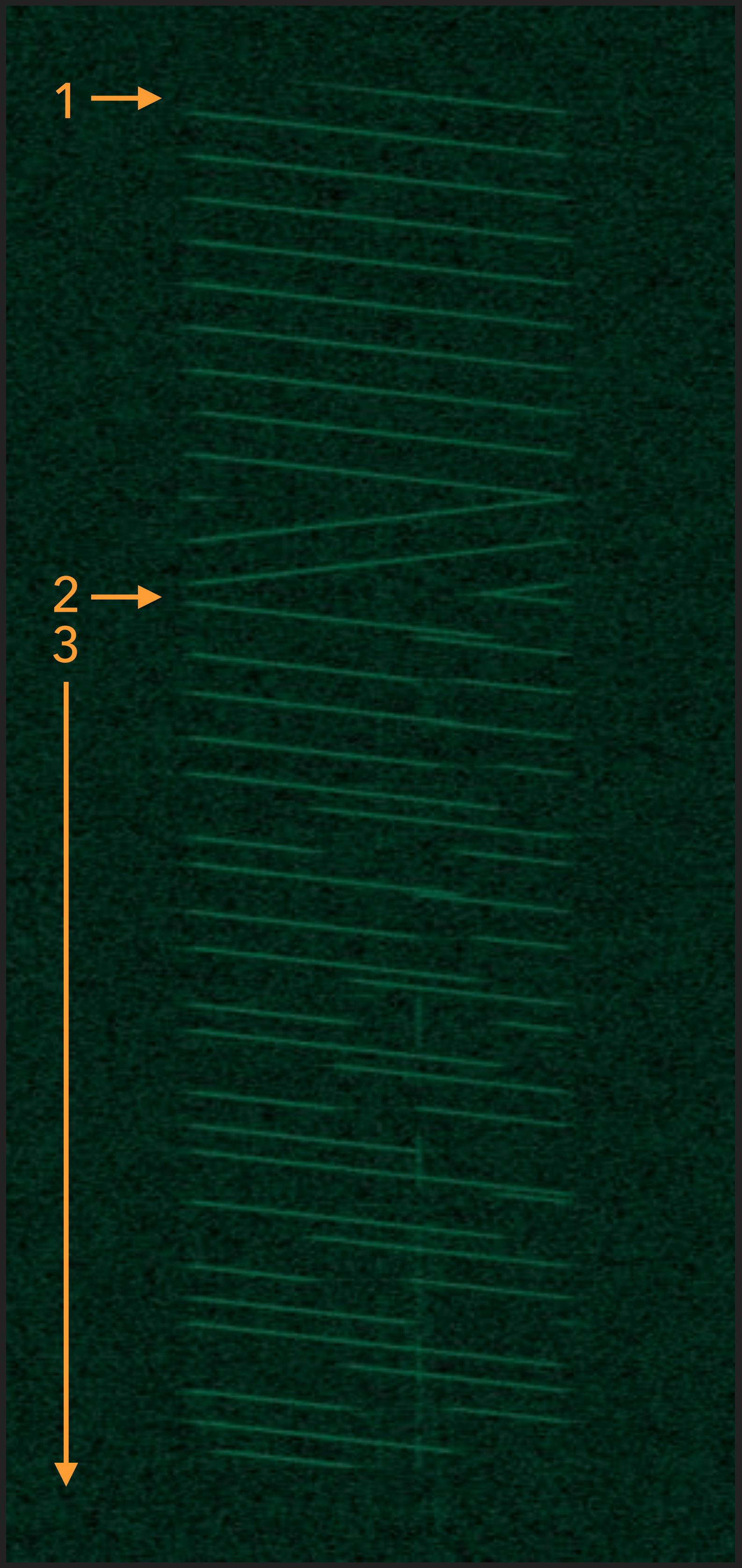
- Instantaneous change in frequency
- FM modulated chirps

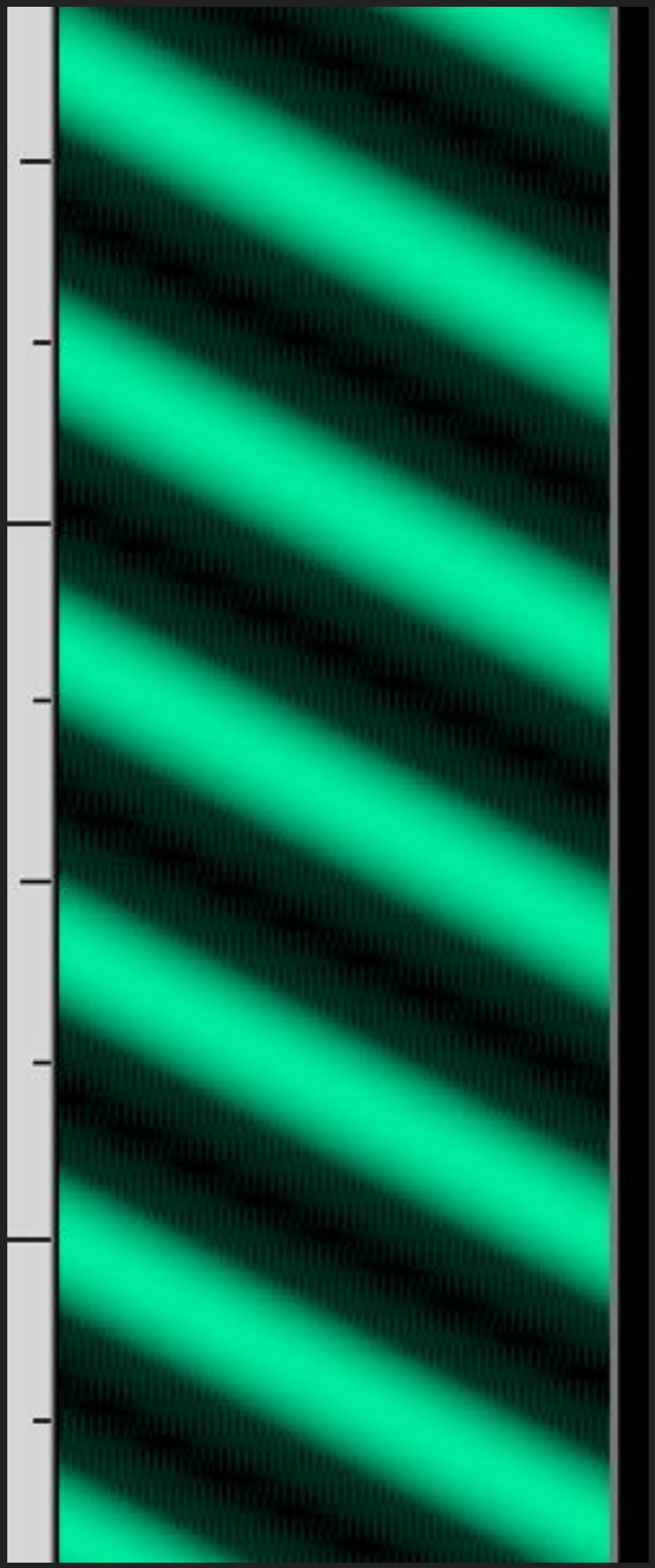


DEMODULATING THE PHY

1. Identify the beginning of a frame
2. Find the beginning of the PHY data unit
3. Extract data from instantaneous frequency transitions

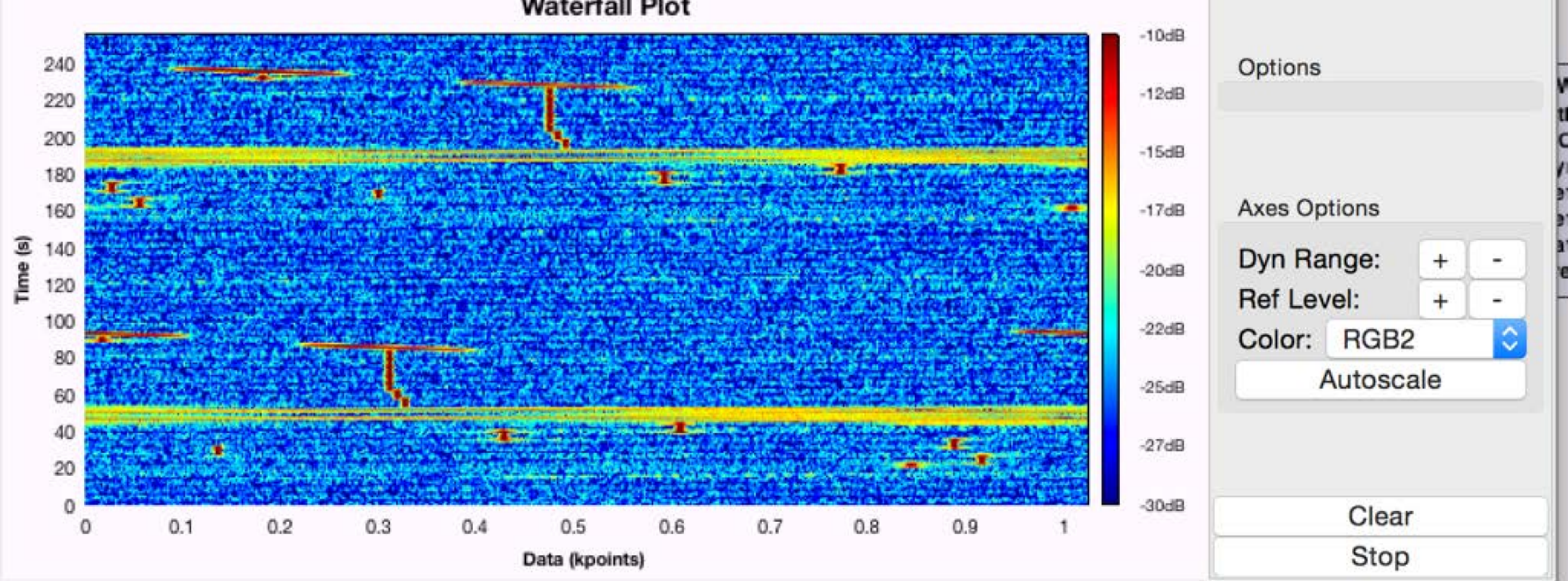
- How? We need to quantify the frequency transitions

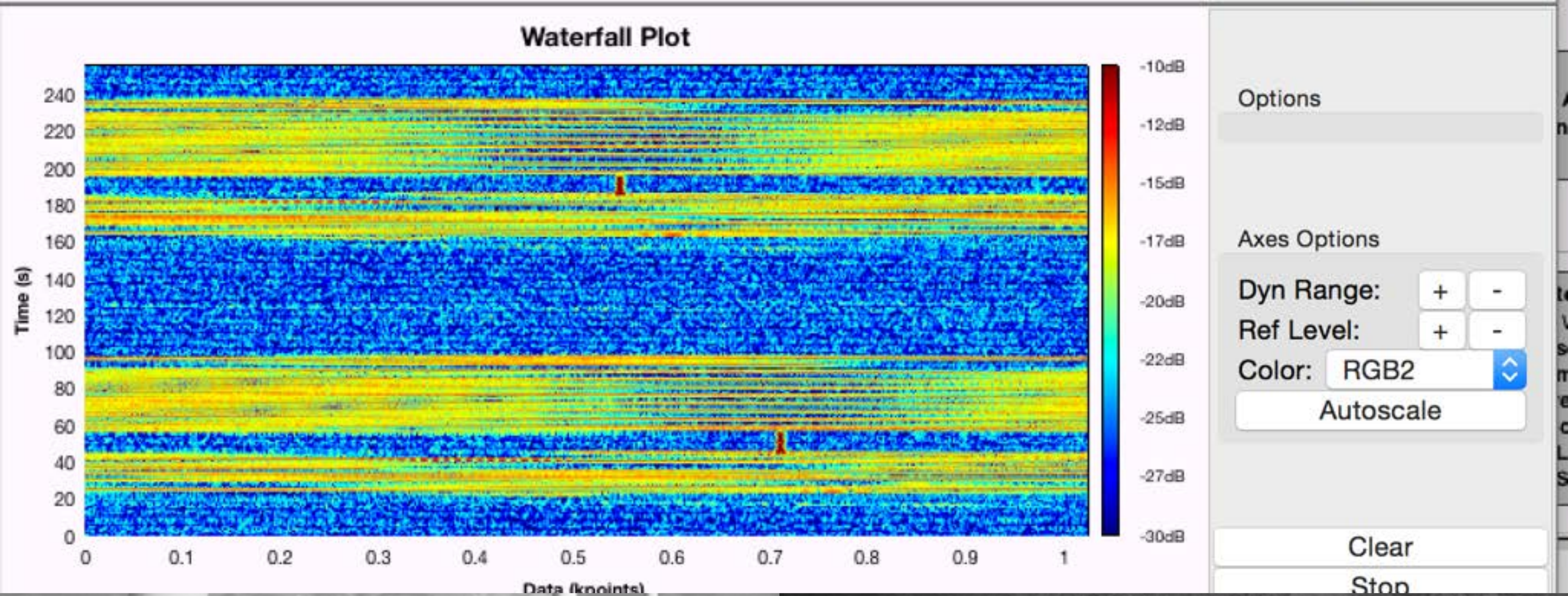




TRANSFORMING THE SIGNAL

- De-chirping the signal makes analysis easier
- Generate local upchirp and downchirp at the appropriate chirp rate
- Multiply each against the signal and something interesting happens…


Waterfall Plot
Time (s)
Data (kpoints)
-10dB
-12dB
-15dB
-17dB
-20dB
-22dB
-25dB
-27dB
-30dB
Options
Axes Options
Dyn Range:
Ref Level:
Color:
RGB2
Autoscale
Clear
Stop

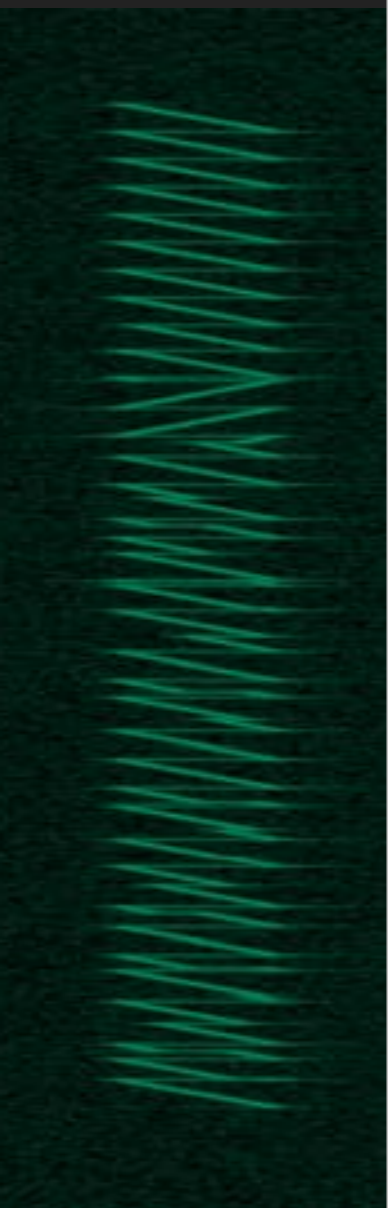

Waterfall Plot
Time (s)
Data (kpoints)
-10dB
-12dB
-15dB
-17dB
-20dB
-22dB
-25dB
-27dB
-30dB
Options
Axes Options
Dyn Range:
Ref Level:
Color:
RGB2
Autoscale
Clear
Stop

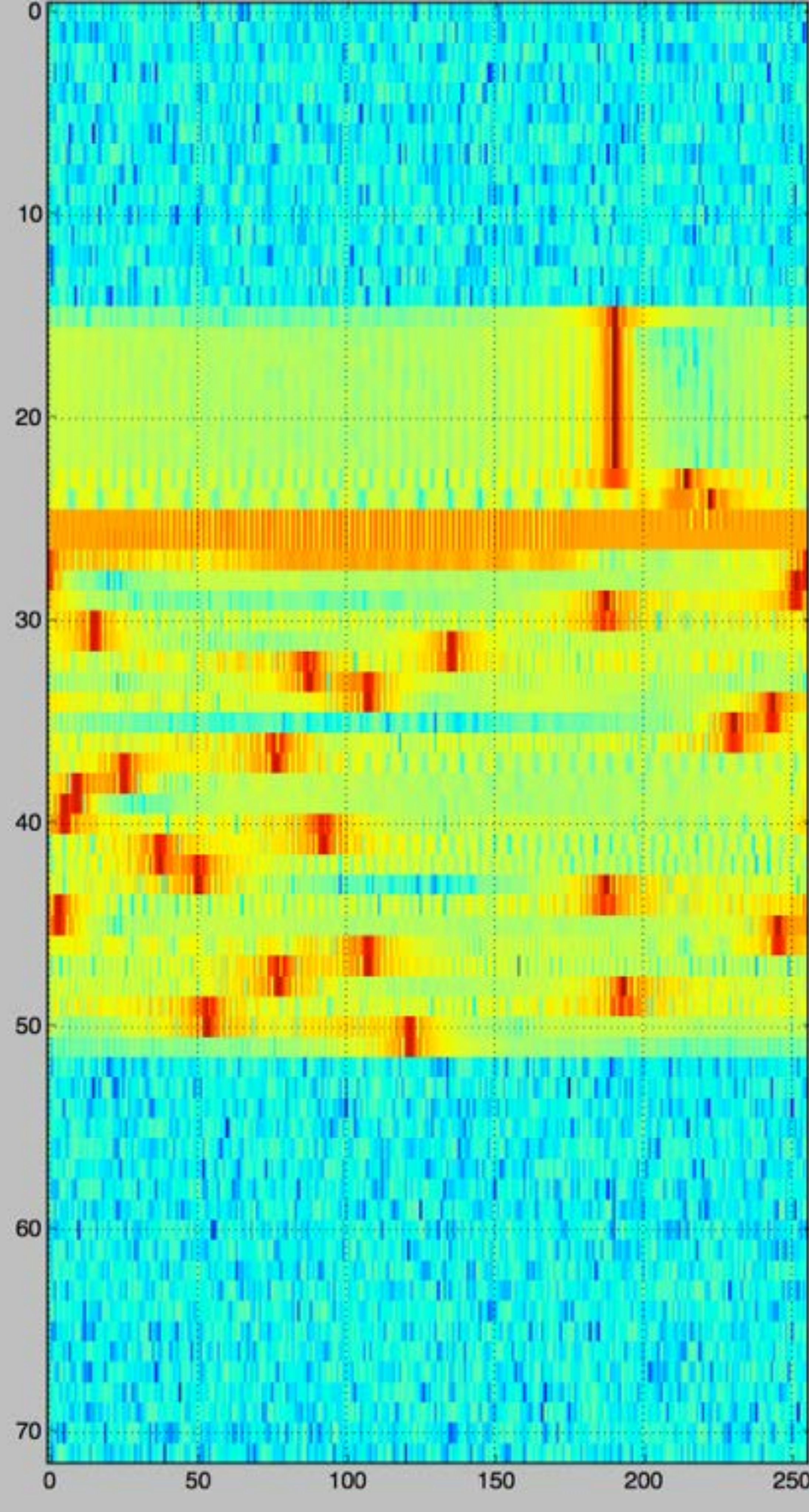
REMEMBER SYMBOLS

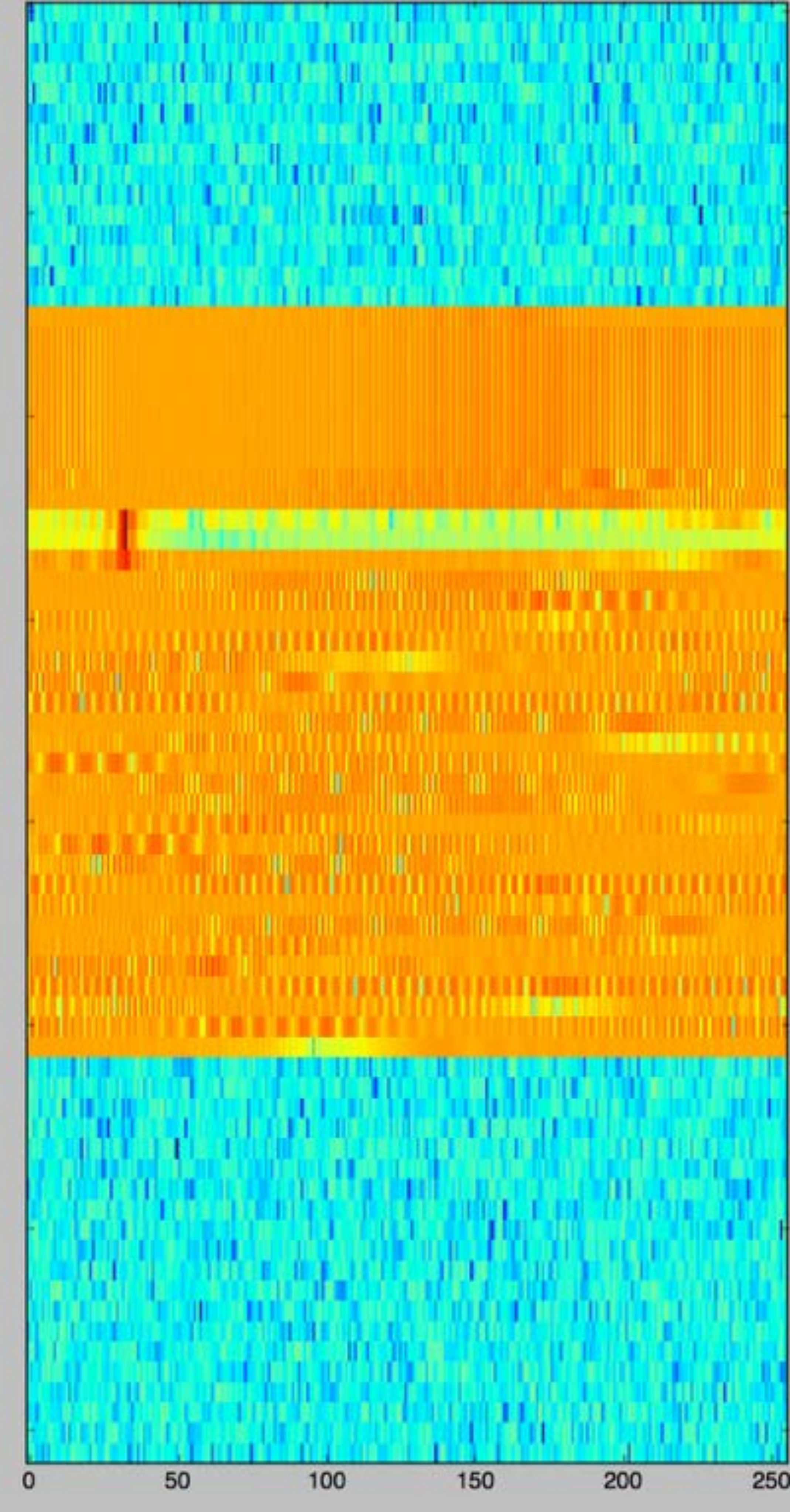
- Symbol: RF state representing some quantity of information
- LoRa spreading factor: number of bits encoded into each symbol
- How many possible symbols are there?
  - 2**spreading_factor

EXTRACTING SYMBOLS

- Length of FFT == number of component frequencies to be extracted
- Channelize and resample signal to chirp bandwidth
- Make the length of the FFT equal to the number of possible symbols
- Most powerful component in each FFT is the symbol!

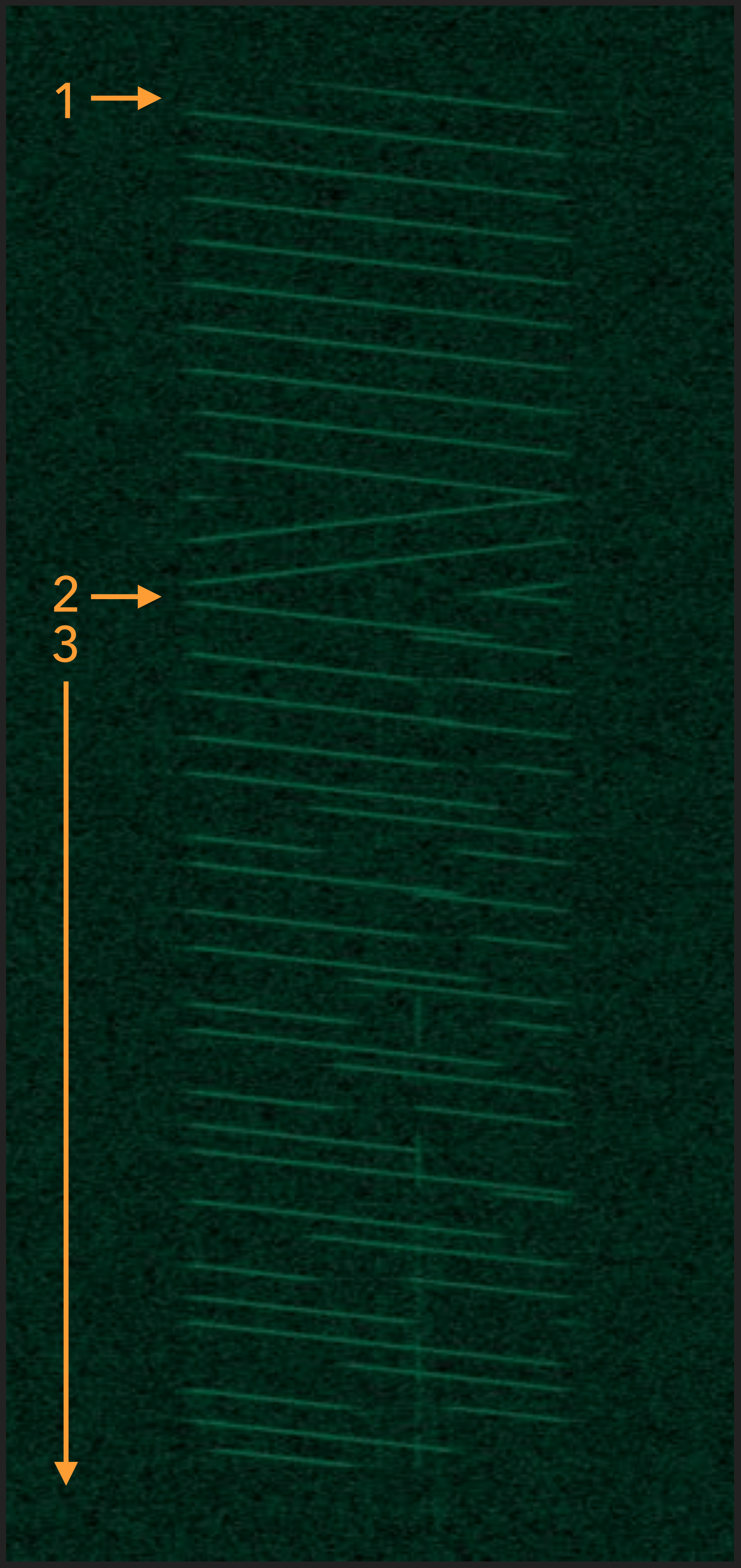



0
10
20
30
40
50
60
70
0
50
100
150
200
250


0
50
100
150
200
250

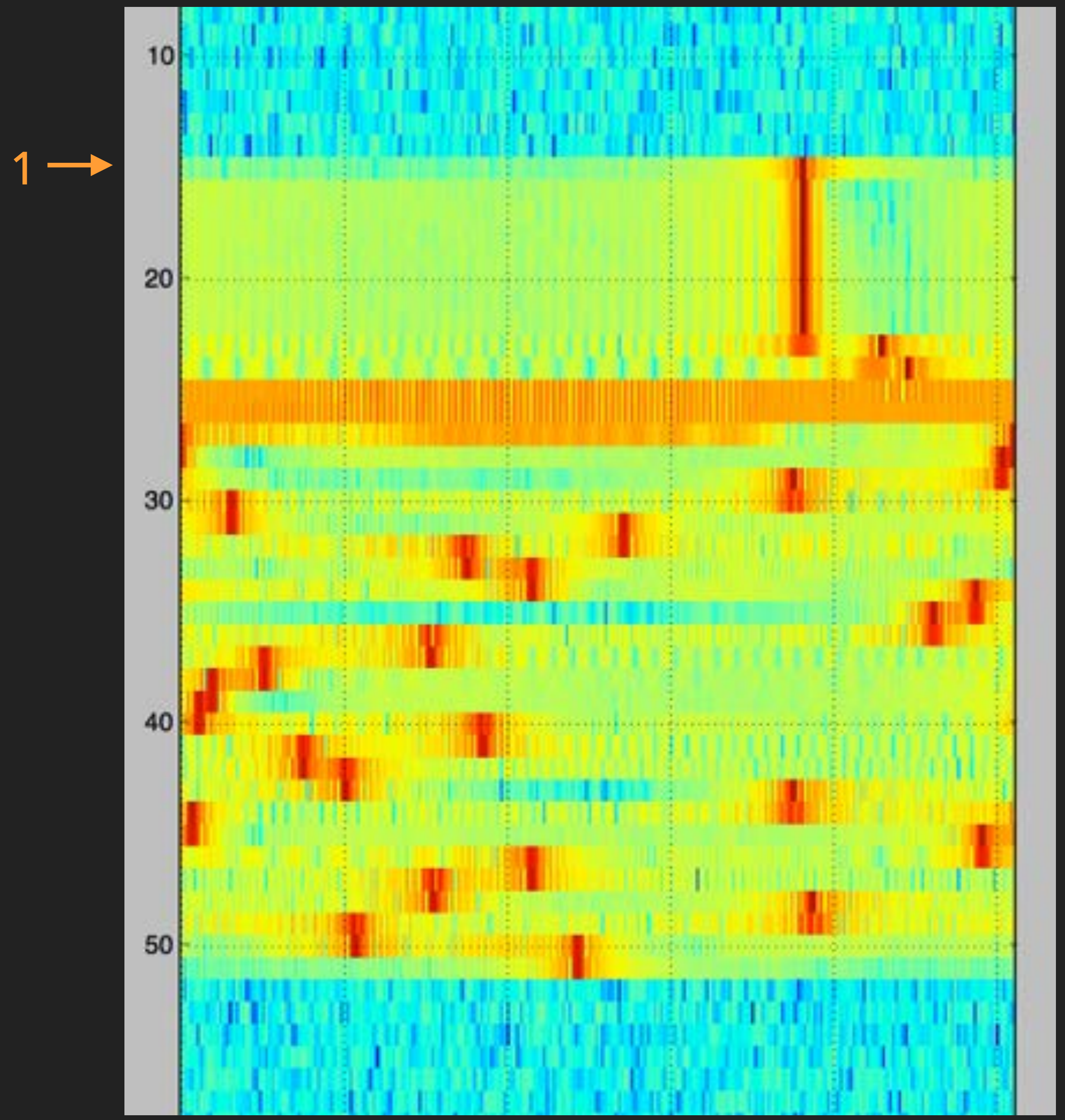
DEMODULATION SUMMARY

1. Identify the beginning of a frame
2. Find the beginning of the PHY data unit
3. Extract data from instantaneous frequency transitions



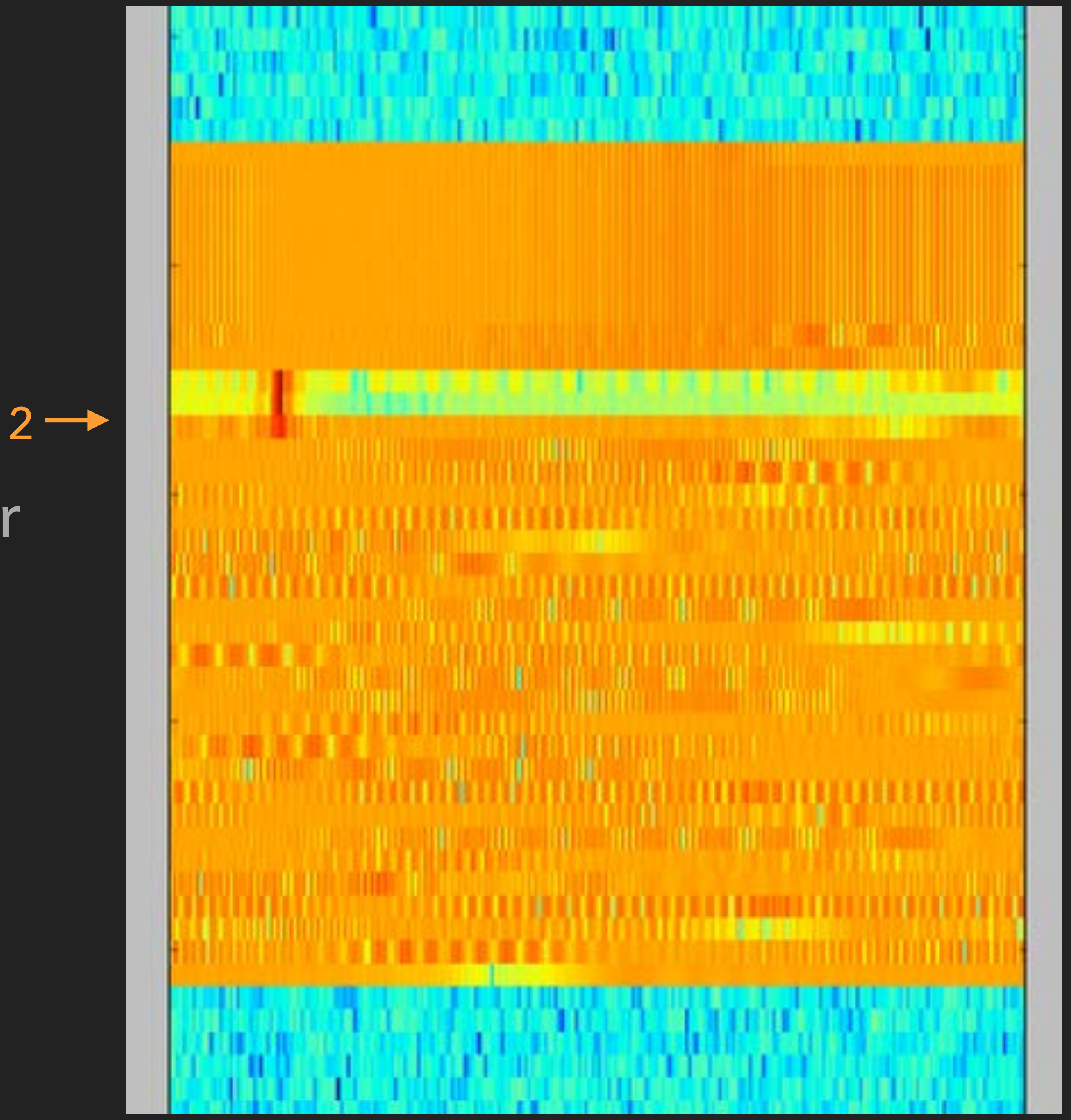
DEMODULATION SUMMARY

1. Identify the beginning of a packet
   - Preamble signified by continuous up-chirp
   - == same symbol being transmitted over and over
   - Look for some number of consecutive FFTs with maximum power in the same bin



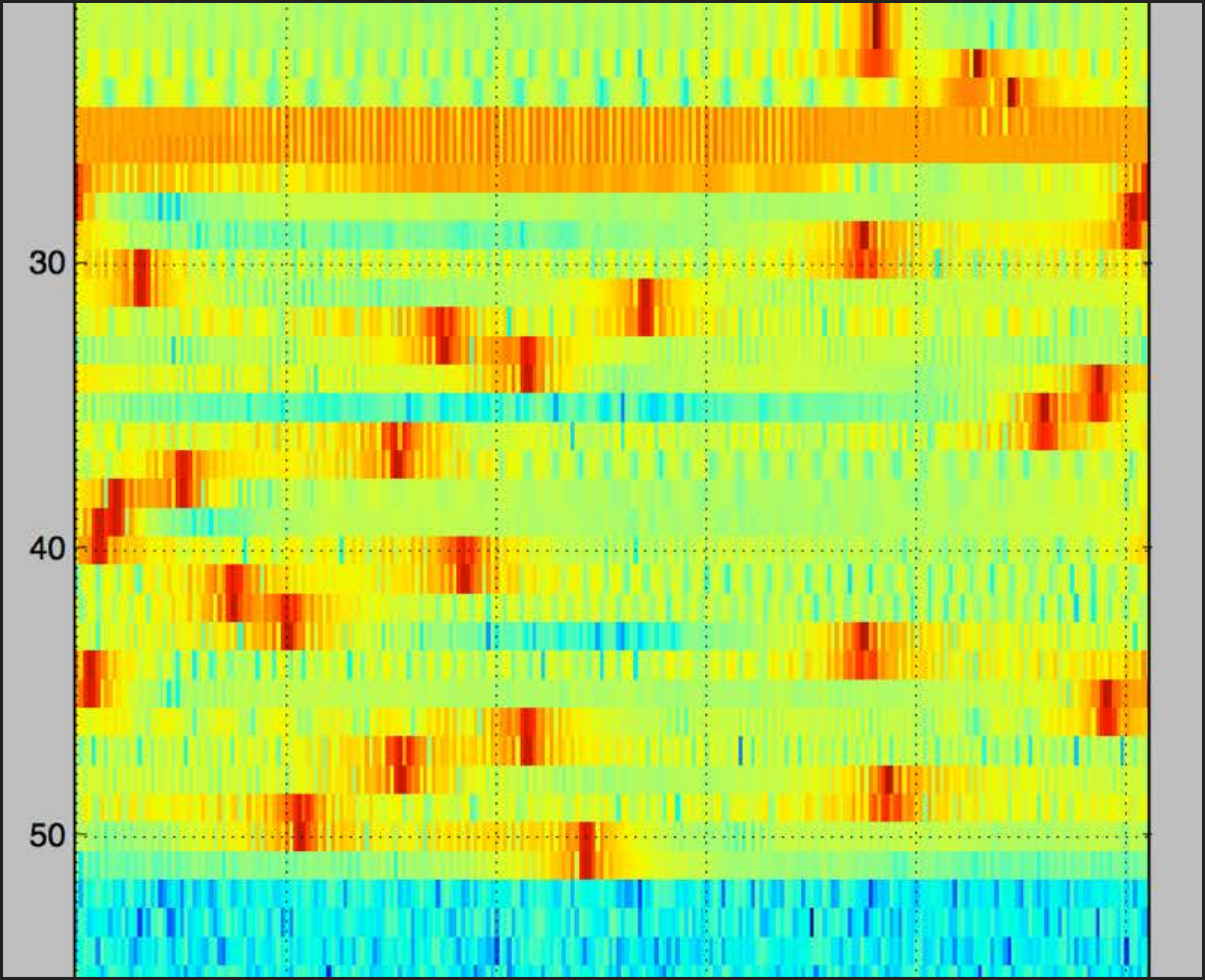
DEMODULATION SUMMARY

2. Find the beginning of the PHY data unit
   - Repeat same process looking for SFD down-chirps
   - Down-chirp is complex conjugate of the up-chirp
   - PHY data unit begins 2 symbols after the SFD



BUT WAIT!

- Accurately finding SFD is essential for receiver synchronization
- Bad sync can spread symbol energy between adjacent FFTs
  - == incorrect data!

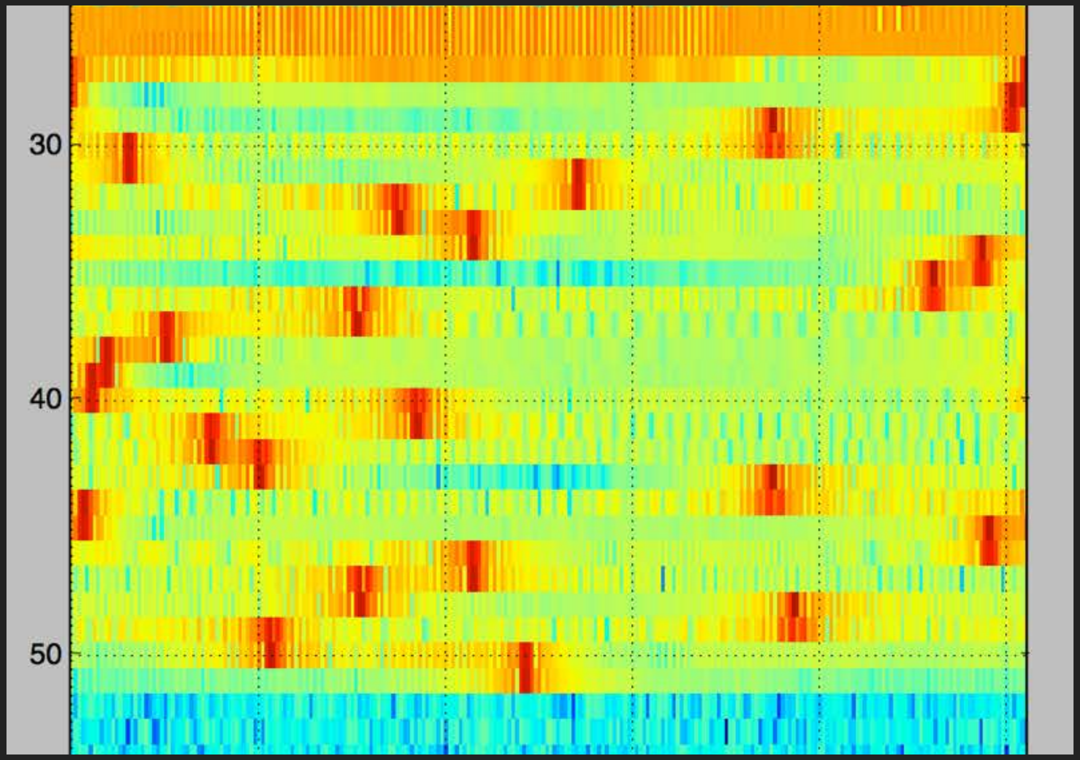

30
40
50

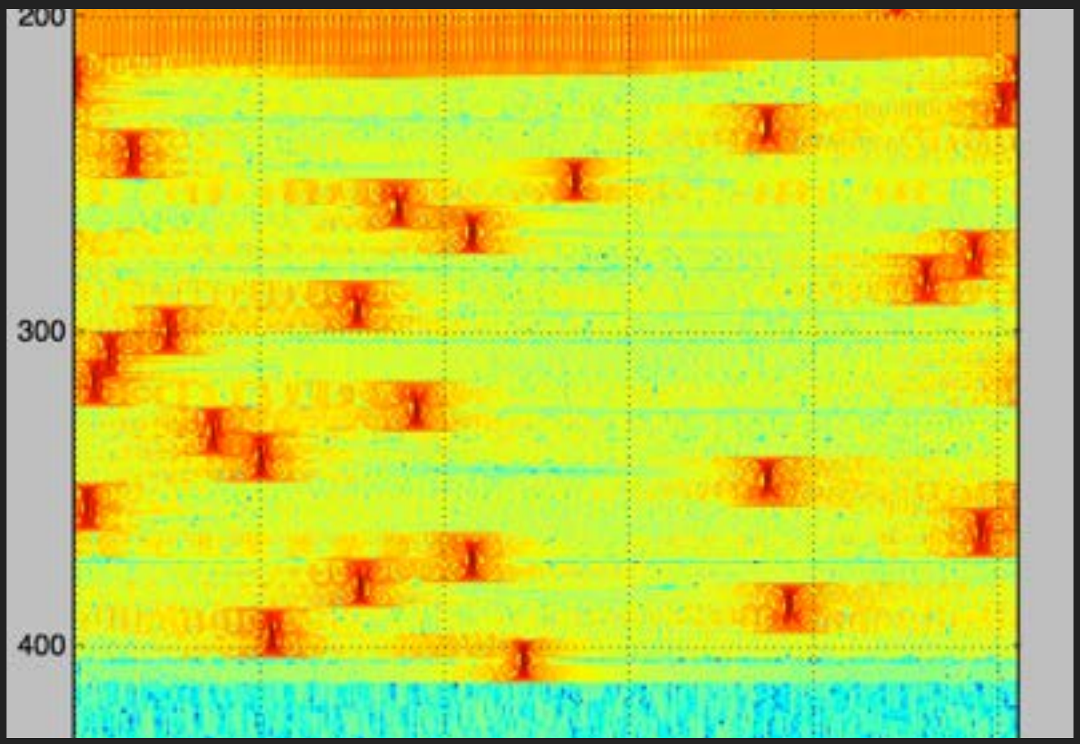
SFD SYNC SOLUTION

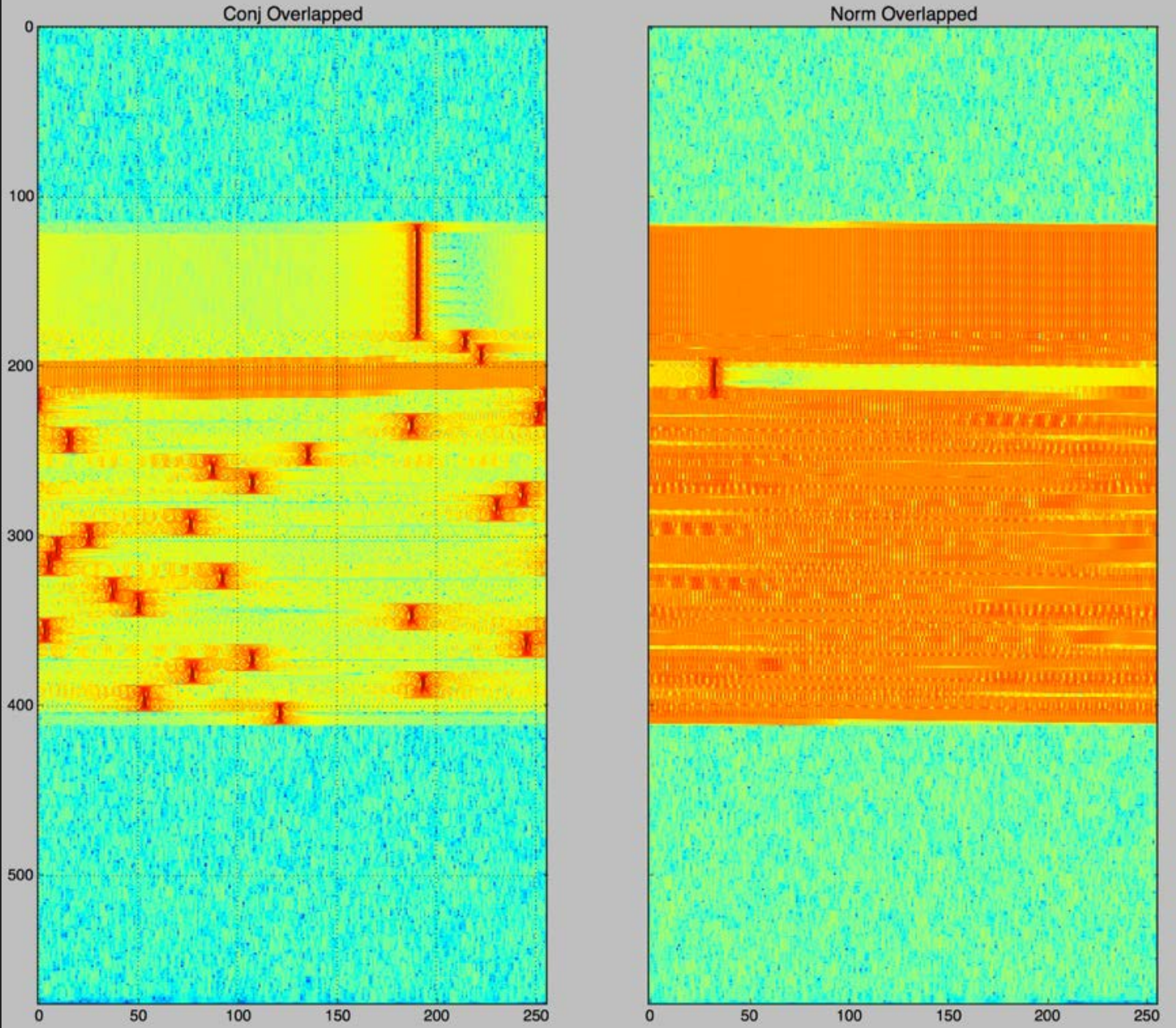
- Increase FFT time-based precision once preamble is found
- Overlapping FFT sample buffers!

OVERLAPPING FFTS

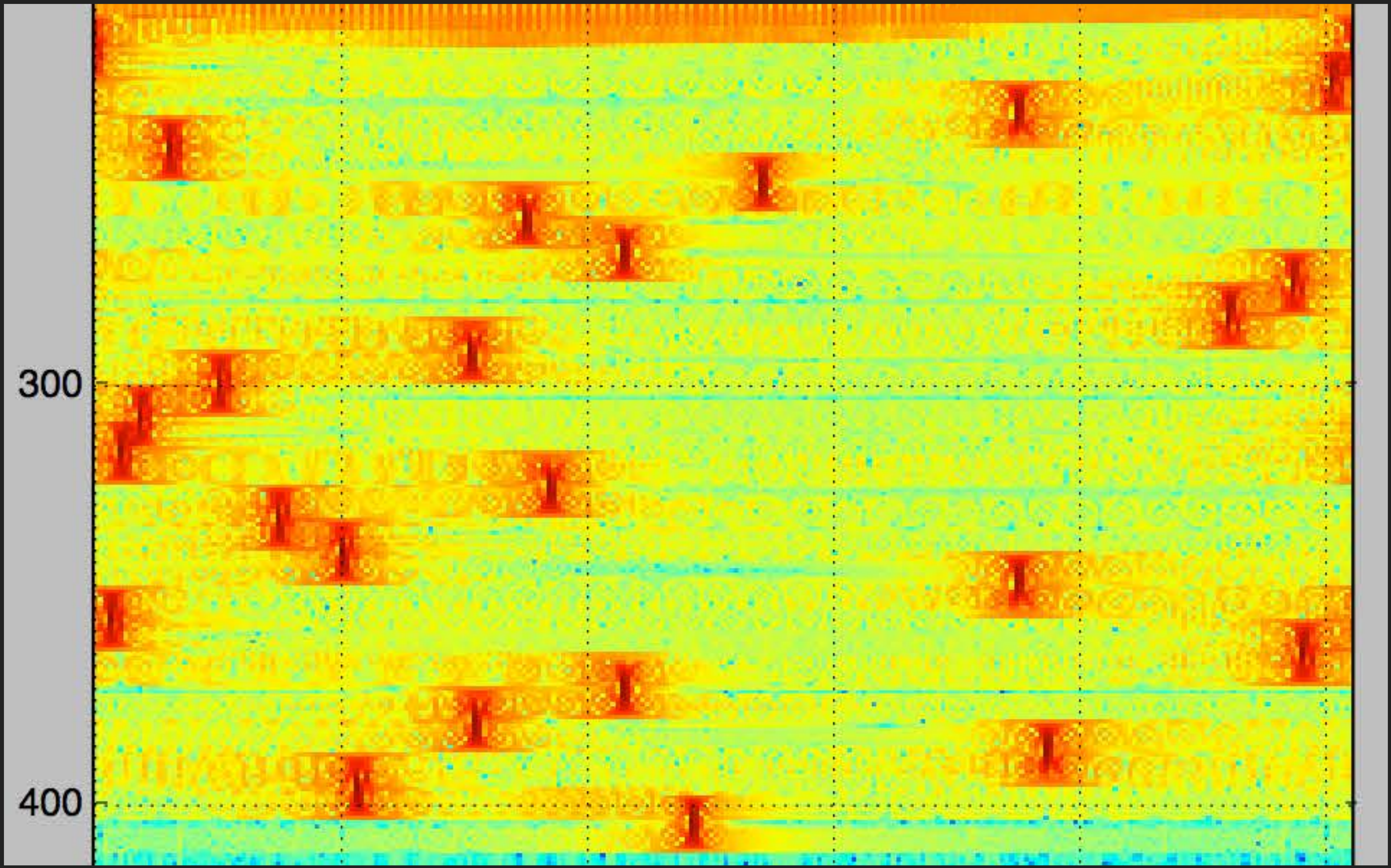
- Top: non-overlapping FFT
  - Each sample processed exactly once
  - [i*fft_len:(i+1)*fft_len-1]
- Bottom: overlapping FFT
  - Samples shifted across multiple FFTs
  - [i*(fft_len/n_overlaps]:(i+1)*(fft_len/n_overlaps)-1]





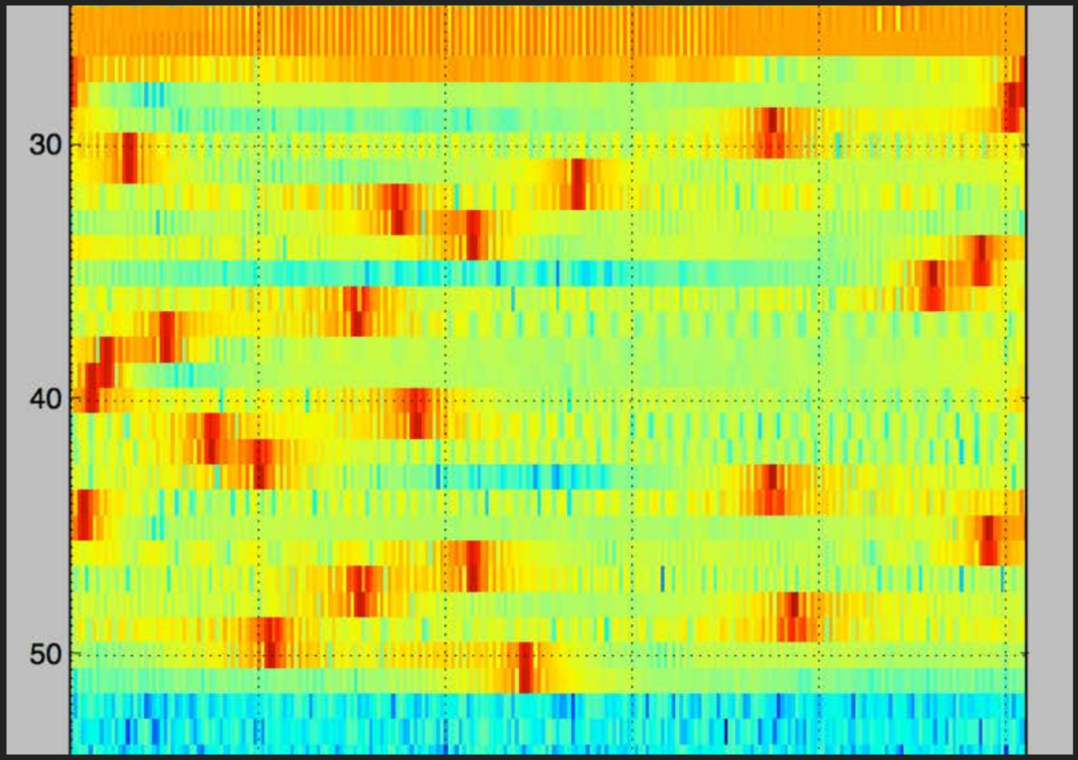


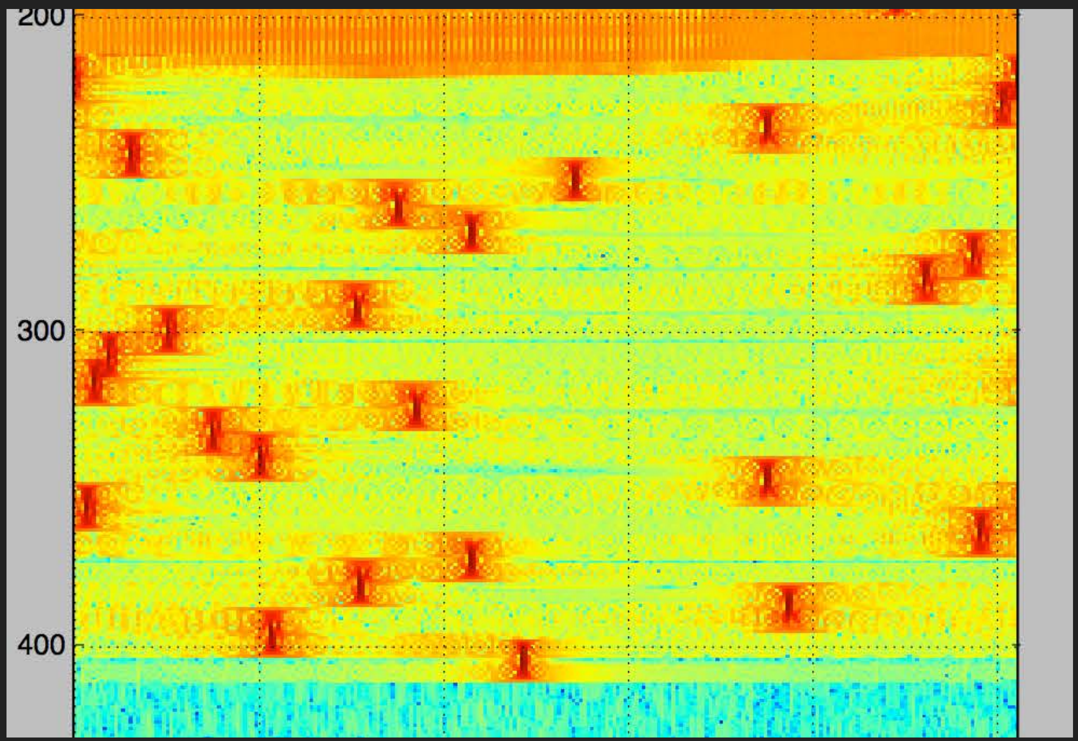
Conj Overlapped
Norm Overlapped
0
100
200
300
400
500
0
50
100
150
200
250
0
50
100
150
200
250

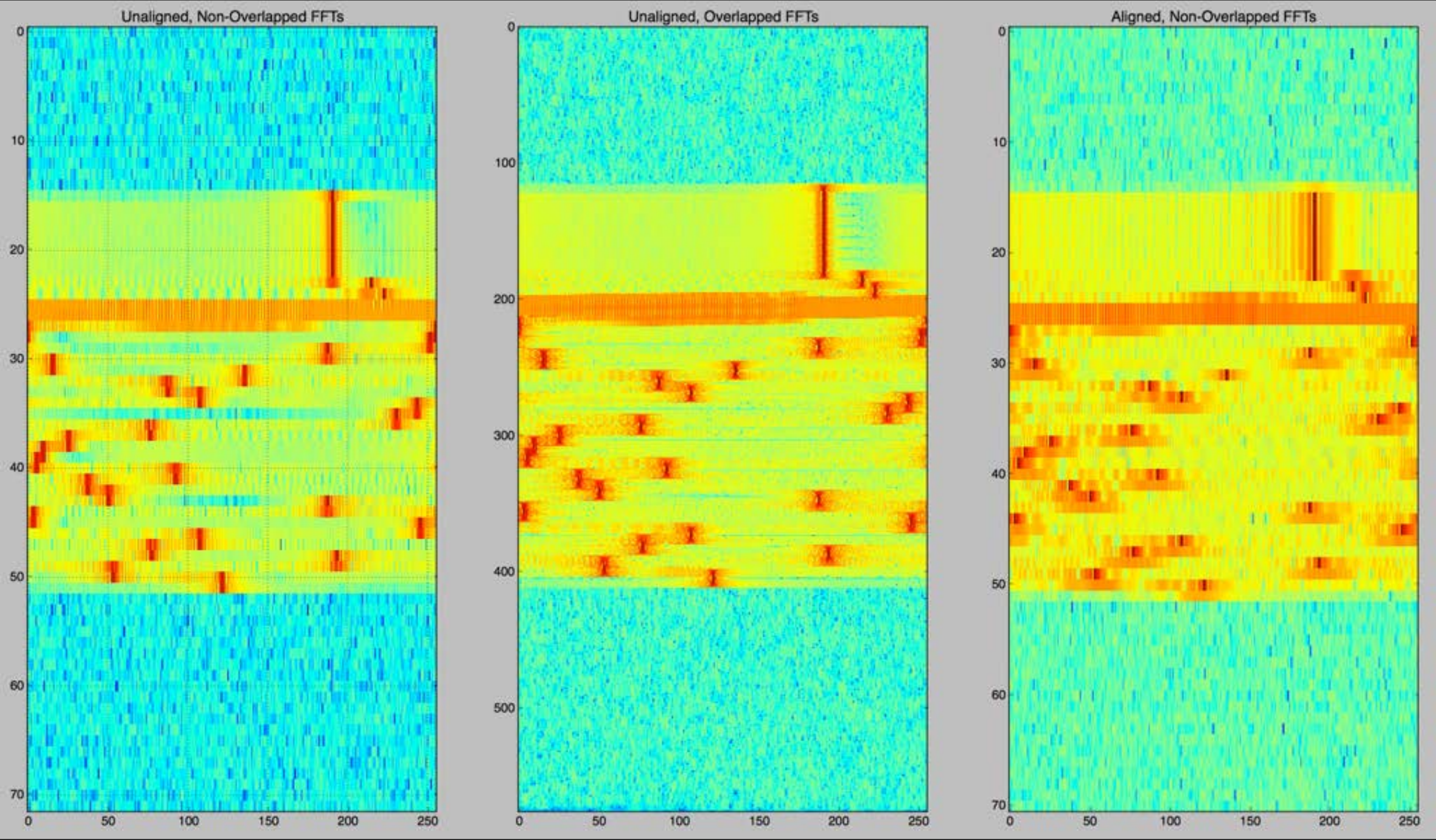

300
400

OVERLAPPING FFTS

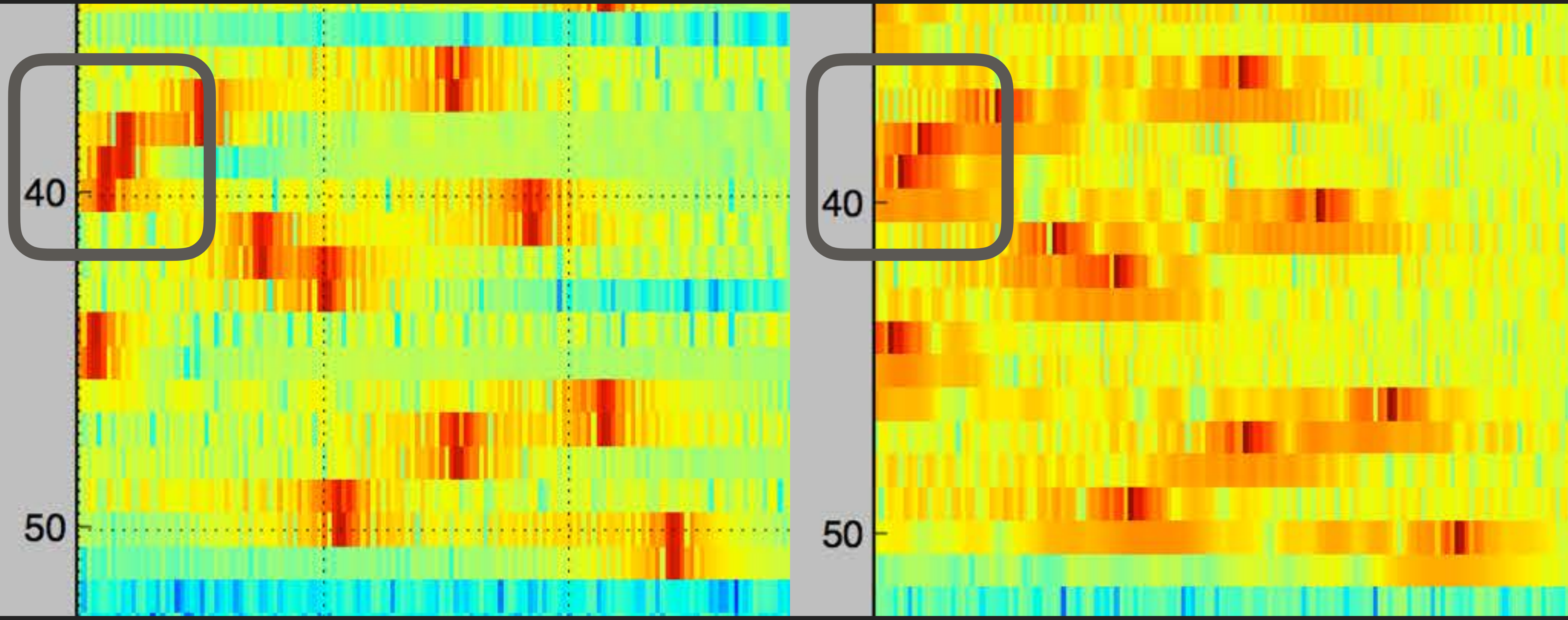
- Use overlapping FFTs to synchronize to first sample in the first SFD symbol
- Re-compute with non-overlapping FFTs to get your data!

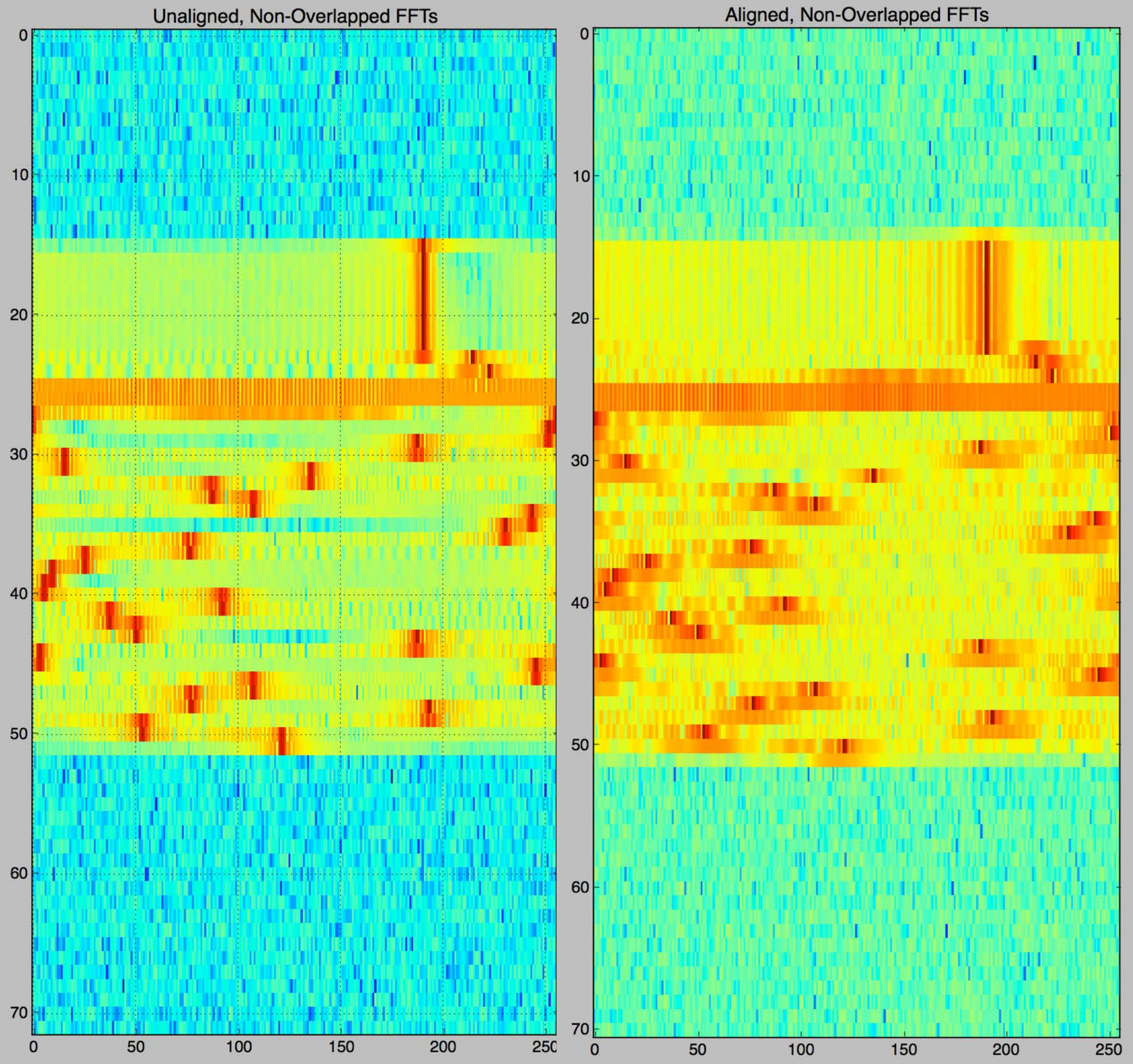






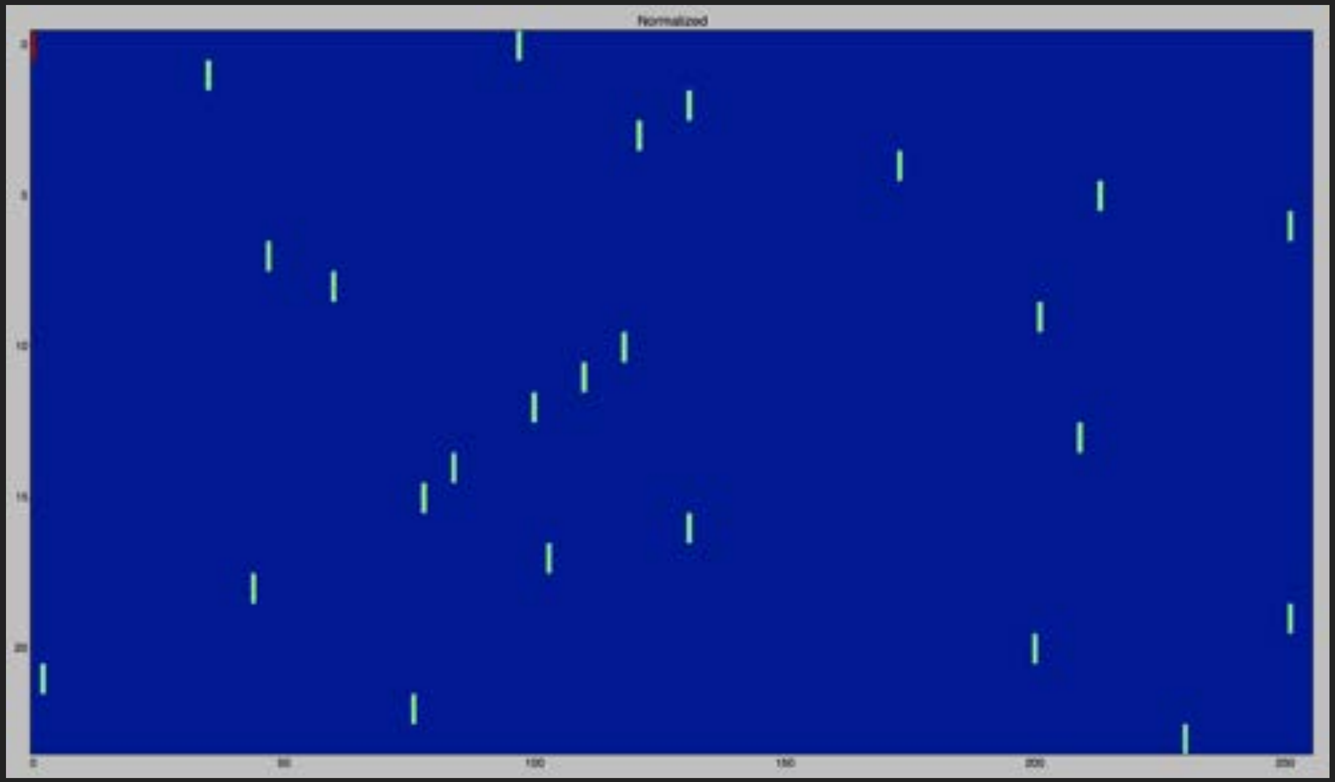
Unaligned, Non-Overlapped FFTs
Unaligned, Overlapped FFTs
Aligned, Non-Overlapped FFTs
0
10
20
30
40
50
60
70
100
200
300
400
500
0
50
100
150
200
250


40
50
40
50



Unaligned, Non-Overlapped FFTs
Aligned, Non-Overlapped FFTs
0
10
20
30
40
50
60
70
0
50
100
150
200
250

DEMODULATION SUMMARY



3. Extract data from chirp phase transitions

- Use described FFT method
- Normalize data about the value of the preamble (always value 0)

WE'RE DONE, RIGHT?

NO.

DATA ENCODING

- Symbols represent encoded data
  - What?
- Data is transformed before it is transmitted
  - Why?
- Encoding increases OTA resiliency

"ARFZ BE LIKE IT IS."

Marc Newlin

RF IS A BRUTAL ENVIRONMENT

- All systems see interference from weather, geomagnetic activity, etc.
- Some systems have protected/reserved frequencies
- LoRa is ISM – TONS OF CHATTER
  - RF contention/collision is guaranteed
- Encoding scrambles and replicates data within frame

WHAT KIND OF ENCODING?

- Semtech European patent application clues:
  1. Symbol “gray indexing” — Adds error tolerance
  2. Data whitening — Induces randomness
  3. Interleaving — Scrambles bits within frame
  4. Forward Error Correction — Adds correcting parity bits
- 4 distinct operations to reverse!

FORWARD ERROR CORRECTION

- Parity bits that can repair bit errors
- Hamming(M,N)
  - M: total bits per codeword [5:8]
  - N: number of bits which are data bits [4]
- Hamming error correction rule
  - (2**num_parity_bits) >= (num_parity_bits + num_data_bits + 1)

HAMMING(M,4) CAPABILITIES

- (5,4): Parity
- (6,4): Parity
- (7,4): Single bit error correction
- (8,4): Single bit error correction, double error detection

GRAY INDEXING

- Originally read as gray-coded before transmission
- Actually de-grayed before transmission

WHITENING

- Transmitter XORs frame with a pseudorandom sequence
- Receiver XORs RX'd frame with same sequence
  - Returns original frame, b/c XOR is its own inverse

- Why? Randomizing data helps receiver synchronization
  - Similar to Manchester encoding
  - Manchester reduces bit rate, whitening does not

FINDING THE WHITENING SEQUENCE

- Several whitening algos defined in Semtech AN1200.18
- Other examples in rtl-sdrangelove
- None of them worked

DERIVING THE WHITENING SEQUENCE

- data XOR 0 == data
- Transmitting a frame of all 0s actually transmits the whitening sequence!
- Hamming (8,4) of 0b0000 = 0b00000000
- Non-additive interleaving of 8x 0b00000000 = 8x 0b00000000

FINDING THE INTERLEAVER

- Semtech European patent application defines diagonal interleaver
  - data(byte,bit) = symbol(bit,(bit+byte)%len_word)
- Also doesn’t work!

DERIVING THE INTERLEAVER

- This was hard
- Exploit properties of Hamming FEC to reveal patterns
  - Most (8,4) codewords contain 4 set bits, however…
  - (8,4) 0b0000 = 0b00000000
  - (8,4) 0b1111 = 0b11111111

DERIVING THE INTERLEAVER, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: 0x0000000F	Payload: 0x000000F0	Payload: 0x00000F00	Payload: 0x0000F000
00100011	11000000	00001001	11010000
00010011	00100101	00000111	00001001
00001001	00010001	00000011	00000101
00000111	00001101	00000011	00000110
00000000	00001100	01000010	00001000
00000100	00000000	10000001	01000010
01000011	00000001	00100001	10000000
10000101	01000111	00010000	00100101

Payload: 0x000F0000	Payload: 0x00F00000	Payload: 0x0F000000	Payload: 0xF0000000
00000011	01000100	01000001	00001000
00000011	00000011	10000010	01000101
01000001	00000000	00100001	10000011
10000010	01000101	00010010	00100011
00100010	10001001	00001010	00010011
00010001	00100010	00000111	00001011
00001001	00010000	00000011	00000111
00000000	00001111	00000101	00000111

DERIVING THE INTERLEAVER, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

```
Payload: 0x0000000F
 00100011
 00010011
 00001001
 00000111
 00000000
 00000100
 01000011
 10000101
```

```
Payload: 0x000000F0
 11000000
 00100101
 00010001
 00001101
 00001100
 00000000
 00000001
 01000111
```

```
Payload: 0x00000F00
 00001001
 00000111
 00000011
 00000011
 01000010
 10000001
 00100001
 00010000
```

```
Payload: 0x0000F000
 11010000
 00001001
 00000101
 00000110
 00001000
 01000010
 10000000
 00100101
```

```
Payload: 0x000F0000
 00000011
 00000011
 01000001
 10000010
 00100010
 00010001
 00001001
 00000000
```

```
Payload: 0x00F00000
 01000100
 00000011
 00000000
 01000101
 10001001
 00100010
 00010000
 00001111
```

```
Payload: 0x0F000000
 01000001
 10000010
 00100001
 00010010
 00001010
 00000111
 00000011
 00000101
```

```
Payload: 0xF0000000
 00001000
 01000101
 10000011
 00100011
 00010011
 00001011
 00000111
 00000111
```

ALIGNING CODEWORDS

- Mapping diagonals returns bits in each code word
- However bits in each word are scrambled
- Solution: Transmit known words, read out diagonals, and look for recognizable Hamming code words

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

```
bit 76543210
    00110011
    10111110
    11111010
    11011101
    10000010
    10000111
    11000000
    10000010
```

	Top							Bot
D								
E								
A								
D								
B								
E								
E								
F								

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

```
bit 76543210
    00110011
    10111110
    11111010
    11011101
    10000010
    10000111
    11000000
    10000010
```

	Top							Bot
D	1	0	1	0	0	0	0	1
E								
A								
D								
B								
E								
E								
F								

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

```
bit 76543210
    00110011
    10111110
    11111010
    11011101
    10000010
    10000111
    11000000
    10000010
```

	Top							Bot
D	1	0	1	0	0	0	0	1
E	0	1	1	1	0	1	0	0
A								
D								
B								
E								
E								
F								

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

```
Payload: 0xDEADBEEF

bit 76543210
    00110011
    10111110
    11111010
    11011101
    10000010
    10000111
    11000000
    10000010
```

	Top							Bot
D	1	0	1	0	0	0	0	1
E	0	1	1	1	0	1	0	0
A	0	1	0	1	1	0	0	0
D								
B								
E								
E								
F								

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

```
bit 76543210
    00110011
    10111110
    11111010
    11011101
    10000010
    10000111
    11000000
    10000010
```

	Top							Bot
D	1	0	1	0	0	0	0	1
E	0	1	1	1	0	1	0	0
A	0	1	0	1	1	0	0	0
D	1	0	1	1	0	0	0	0
B								
E								
E								
F								

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

bit 76543210
001**1**0011
1011**1**110
11111**0**10
110111**0**1
1000001**0**
1**0**000111
11000000
10**0**00010

	Top						Bot
D	1	0	1	0	0	0	0	1
E	0	1	1	1	0	1	0	0
A	0	1	0	1	1	0	0	0
D	1	0	1	1	0	0	0	0
B	1	1	0	0	0	0	1	0
E								
E								
F								

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

```
Payload: 0xDEADBEEF

bit 76543210
    00110011
    10111110
    11111010
    11011101
    10000010
    10000111
    11000000
    10000010
```

	Top							Bot
D	1	0	1	0	0	0	0	1
E	0	1	1	1	0	1	0	0
A	0	1	0	1	1	0	0	0
D	1	0	1	1	0	0	0	0
B	1	1	0	0	0	0	1	0
E	0	1	1	1	0	1	0	0
E								
F								

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

```
Payload: 0xDEADBEEF

bit 76543210
    00110011
    10111110
    11111010
    11011101
    10000010
    10000111
    11000000
    10000010
```

	Top							Bot
D	1	0	1	0	0	0	0	1
E	0	1	1	1	0	1	0	0
A	0	1	0	1	1	0	0	0
D	1	0	1	1	0	0	0	0
B	1	1	0	0	0	0	1	0
E	0	1	1	1	0	1	0	0
E	0	1	1	1	0	1	0	0
F								

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

```
Payload: 0xDEADBEEF

bit 76543210
    00110011
    10111110
    11111010
    11011101
    10000010
    10000111
    11000000
    10000010
```

	Top							Bot
D	1	0	1	0	0	0	0	1
E	0	1	1	1	0	1	0	0
A	0	1	0	1	1	0	0	0
D	1	0	1	1	0	0	0	0
B	1	1	0	0	0	0	1	0
E	0	1	1	1	0	1	0	0
E	0	1	1	1	0	1	0	0
F	1	1	1	1	1	1	1	1

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

```
Payload: 0xDEADBEEF

bit 76543210
    00110011
    10111110
    11111010
    11011101
    10000010
    10000111
    11000000
    10000010
```

	Top							Bot
D	1	0	1	0	0	0	0	1
E	0	1	1	1	0	1	0	0
A	0	1	0	1	1	0	0	0
D	1	0	1	1	0	0	0	0
B	1	1	0	0	0	0	1	0
E	0	1	1	1	0	1	0	0
E	0	1	1	1	0	1	0	0
F	1	1	1	1	1	1	1	1

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

`Payload: 0xDEADBEEF`

	Bin	(8,4) Parity
0xD	1101	1000
0xE	1110	0001
0xA	1010	1010
0xD	1101	1000
0xB	1011	0100
0xE	1110	0001
0xE	1110	0001
0xF	1111	1111

	Top							Bot
D	1	0	1	0	0	0	0	1
E	0	1	1	1	0	1	0	0
A	0	1	0	1	1	0	0	0
D	1	0	1	1	0	0	0	0
B	1	1	0	0	0	0	1	0
E	0	1	1	1	0	1	0	0
E	0	1	1	1	0	1	0	0
F	1	1	1	1	1	1	1	1

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

	Bin	(8,4) Parity
0xD	1101	1000
0xE	1110	0001
0xA	1010	1010
0xD	1101	1000
0xB	1011	0100
0xE	1110	0001
0xE	1110	0001
0xF	1111	1111

	Bot							Top
D	1	0	0	0	0	1	0	1
E	0	0	1	0	1	1	1	0
A	0	0	0	1	1	0	1	0
D	0	0	0	0	1	1	0	1
B	0	1	0	0	0	0	1	1
E	0	0	1	0	1	1	1	0
E	0	0	1	0	1	1	1	0
F	1	1	1	1	1	1	1	1

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

`Payload: 0xDEADBEEF`

	Bin	(8,4) Parity
0xD	1101	1000
0xE	1110	0001
0xA	1010	1010
0xD	1101	1000
0xB	1011	0100
0xE	1110	0001
0xE	1110	0001
0xF	1111	1111

	Bot							Top
D	1	0	0	0	0	1	0	1
E	0	0	1	0	1	1	1	0
A	0	0	0	1	1	0	1	0
D	0	0	0	0	1	1	0	1
B	0	1	0	0	0	0	1	1
E	0	0	1	0	1	1	1	0
E	0	0	1	0	1	1	1	0
F	1	1	1	1	1	1	1	1

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

	Bin	(8,4) Parity
0xD	1101	1000
0xE	1110	0001
0xA	1010	1010
0xD	1101	1000
0xB	1011	0100
0xE	1110	0001
0xE	1110	0001
0xF	1111	1111

	p1	p2			d1	d2	d3	d4
	Bot							Top
D	1	0	0	0	0	1	0	1
E	0	0	1	0	1	1	1	0
A	0	0	0	1	1	0	1	0
D	0	0	0	0	1	1	0	1
B	0	1	0	0	0	0	1	1
E	0	0	1	0	1	1	1	0
E	0	0	1	0	1	1	1	0
F	1	1	1	1	1	1	1	1

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

	Bin	(8,4) Parity
0xD	1101	1000
0xE	1110	0001
0xA	1010	1010
0xD	1101	1000
0xB	1011	0100
0xE	1110	0001
0xE	1110	0001
0xF	1111	1111

	p1	p2	p4	p3	d1	d2	d3	d4
	Bot							Top
D	1	0	0	0	0	1	0	1
E	0	0	1	0	1	1	1	0
A	0	0	0	1	1	0	1	0
D	0	0	0	0	1	1	0	1
B	0	1	0	0	0	0	1	1
E	0	0	1	0	1	1	1	0
E	0	0	1	0	1	1	1	0
F	1	1	1	1	1	1	1	1

ALIGNING CODEWORDS, ILLUSTRATED

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

	Bin	(8,4) Parity
0xD	1101	1000
0xE	1110	0001
0xA	1010	1010
0xD	1101	1000
0xB	1011	0100
0xE	1110	0001
0xE	1110	0001
0xF	1111	1111

	p1	p2	p4	p3	d1	d2	d3	d4
	Bot							Top
D	1	0	0	0	0	1	0	1
E	0	0	1	0	1	1	1	0
A	0	0	0	1	1	0	1	0
D	0	0	0	0	1	1	0	1
B	0	1	0	0	0	0	1	1
E	0	0	1	0	1	1	1	0
E	0	0	1	0	1	1	1	0
F	1	1	1	1	1	1	1	1

ALMOST THERE

APPLY FORWARD ERROR CORRECTION

- spreading_factor = 8, code_rate=4/8

`Payload: 0xDEADBEEF`

	Bin	(8,4) Parity
0xD	1101	1000
0xE	1110	0001
0xA	1010	1010
0xD	1101	1000
0xB	1011	0100
0xE	1110	0001
0xE	1110	0001
0xF	1111	1111

	p1	p2	p4	p3	d1	d2	d3	d4
	Bot							Top
D	1	0	0	0	0	1	0	1
E	0	0	1	0	1	1	1	0
A	0	0	0	1	1	0	1	0
D	0	0	0	0	1	1	0	1
B	0	1	0	0	0	0	1	1
E	0	0	1	0	1	1	1	0
E	0	0	1	0	1	1	1	0
F	1	1	1	1	1	1	1	1

APPLY FORWARD ERROR CORRECTION

- spreading_factor = 8, code_rate=4/8

Payload: **0xDEADBEEF**

	Bin	(8,4) Parity
0xD	1101	1000
0xE	1110	0001
0xA	1010	1010
0xD	1101	1000
0xB	1011	0100
0xE	1110	0001
0xE	1110	0001
0xF	1111	1111

	p1	p2	p4	p3	d1	d2	d3	d4
	Bot							Top
D	1	0	0	0	1	1	0	1
E	0	0	1	0	1	1	1	0
A	1	0	0	1	1	0	1	0
D	1	0	0	0	1	1	0	1
B	0	1	0	0	1	0	1	1
E	0	0	1	0	1	1	1	0
E	0	0	1	0	1	1	1	0
F	1	1	1	1	1	1	1	1

APPLY FORWARD ERROR CORRECTION

- spreading_factor = 8, code[illegible]/8

`Payload: 0xDEAD[illegible]`

BOOM

[illegible]	B[illegible]	(8,4[illegible]ity
[illegible]	11[illegible]	10[illegible]
[illegible]	[illegible]	00[illegible]
0[illegible]	[illegible]1010	101[illegible]
0x[illegible]	1101	1000
0x[illegible]	1011	[illegible]0100
0xE	11[illegible]	0001
0xE	[illegible]	0001
0xF	1111	1111

[illegible]	[illegible]	p2	[illegible]	p3	[illegible]1	[illegible]	d3	[illegible]4
[illegible]	[illegible]	0	[illegible]	0	[illegible]	1	0	[illegible]
[illegible]	0	0	1	0	[illegible]	1	1	[illegible]
[illegible]	1	[illegible]	[illegible]	1	[illegible]	0	1	0
[illegible]	1	[illegible]	[illegible]	0	1	1	0	1
[illegible]	0	1	0	0	1	0	1	1
E	0	0	1	0	1	1	1	0
E	0	0	1	0	1	1	1	0
F	1	1	1	1	1	1	1	1

TO CONCLUDE

- LPWANs have momentum and are proliferating
- RF stacks are becoming more diverse
  - Wireless is not just WiFi anymore
- Shown how to go from obscure RF –> bits

ACKNOWLEDGEMENTS

- Balint Seeber, Bastille Threat Research Team
- hexameron and Bertrik Sikken, open source contributors
- Johan Stokking, The Things Network
- Jailbreak for hosting!

THANKS

matt@Bastille.net
@embeddedsec

QUESTIONS?

matt@Bastille.net
@embeddedsec