



MATT KNIGHT // MARC NEWLIN // BASTILLE NETWORKS

RADIO EXPLOITATION 101

CHARACTERIZING//CONTEXTUALIZING//CLASSIFYING RF ATTACKS

OFFENSIVE WIRELESS TECHNIQUES

PHY && MAC LAYERS

MOBILE

and

IoT

means

WIRELESS TECH IS

DIVERSE AND INCUMBENT

**ATTACK SURFACES
GALORE**

WHO ARE THESE GUYS

▶ Matt Knight

- ▶ Software Engineer and Security Researcher @ **Bastille**
- ▶ Reverse engineered the **LoRa** wireless protocol in 2016
- ▶ BE & BA from **Dartmouth**

matt@**Bastille**.net
@embeddedsec

▶ Marc Newlin

- ▶ Security Researcher @ **Bastille**
- ▶ Discovered **Mousejack** vulnerability in 2016
- ▶ Finished 3rd in DARPA Spectrum Challenge in 2012
- ▶ Finished 2nd in DARPA Shredder Challenge in 2010

marc@**Bastille**.net
@marcnewlin

AGENDA

1. Historical retrospective of wired and wireless security tech development
2. Key RF concepts
3. Methods of Wireless Exploitation
 - ▶ Techniques, impact, and defenses
 - ▶ Analogues to wired networks
 - ▶ Examples and demos (do it live!)
4. How to apply this information



HISTORICAL BACKGROUND

EVOLUTION OF NETWORK SECURITY

Packet sniffing in the

1990s

Protocols:

802.3

802.5

\$8,000+ (in 1990s dollars)



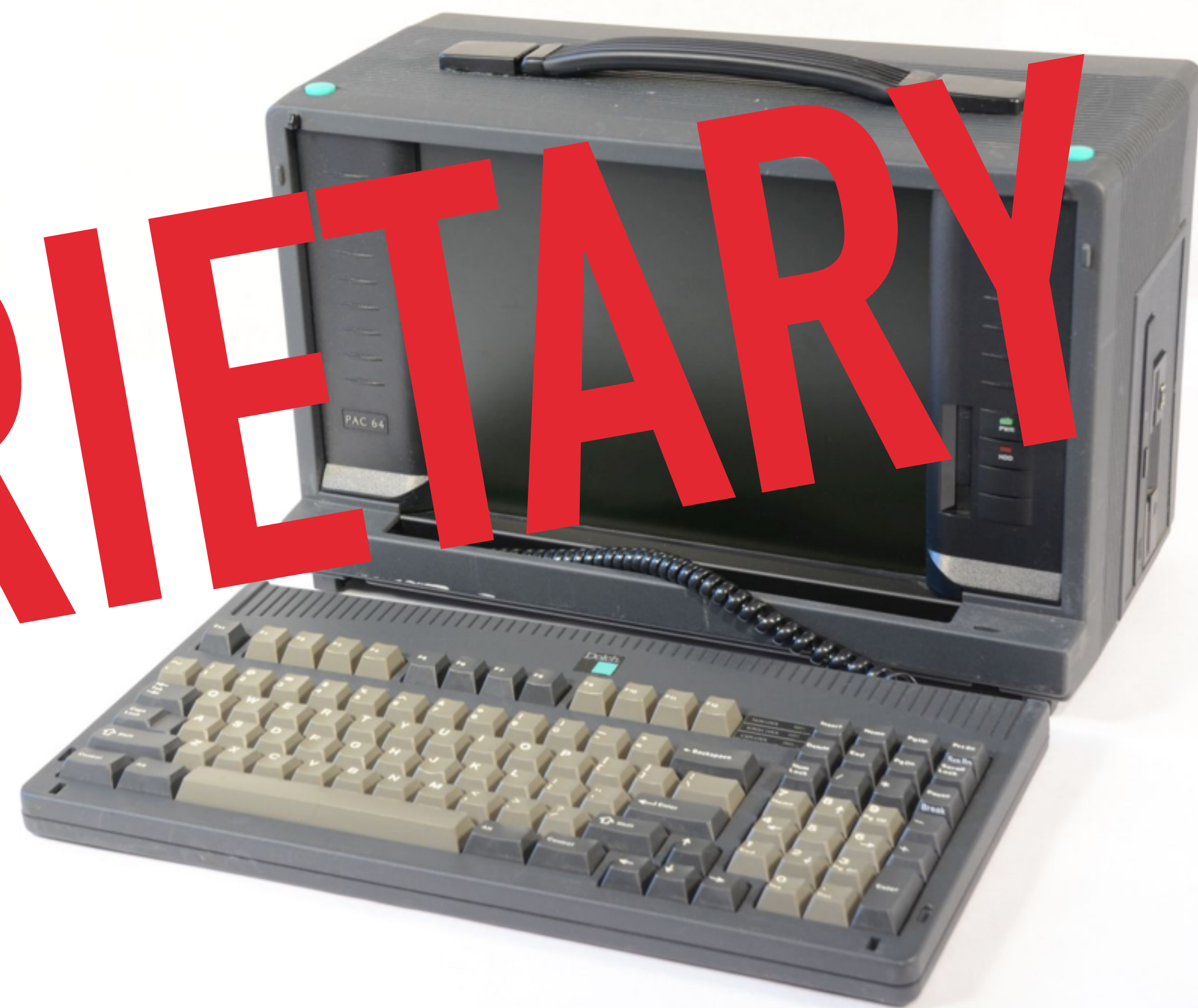
NETWORK GENERAL PACKET SNIFFER

Installed on a Dolch lunchbox computer



\$8,000+ (in 1990s dollars)

PROPRIETARY



NETWORK GENERAL PACKET SNIFFER

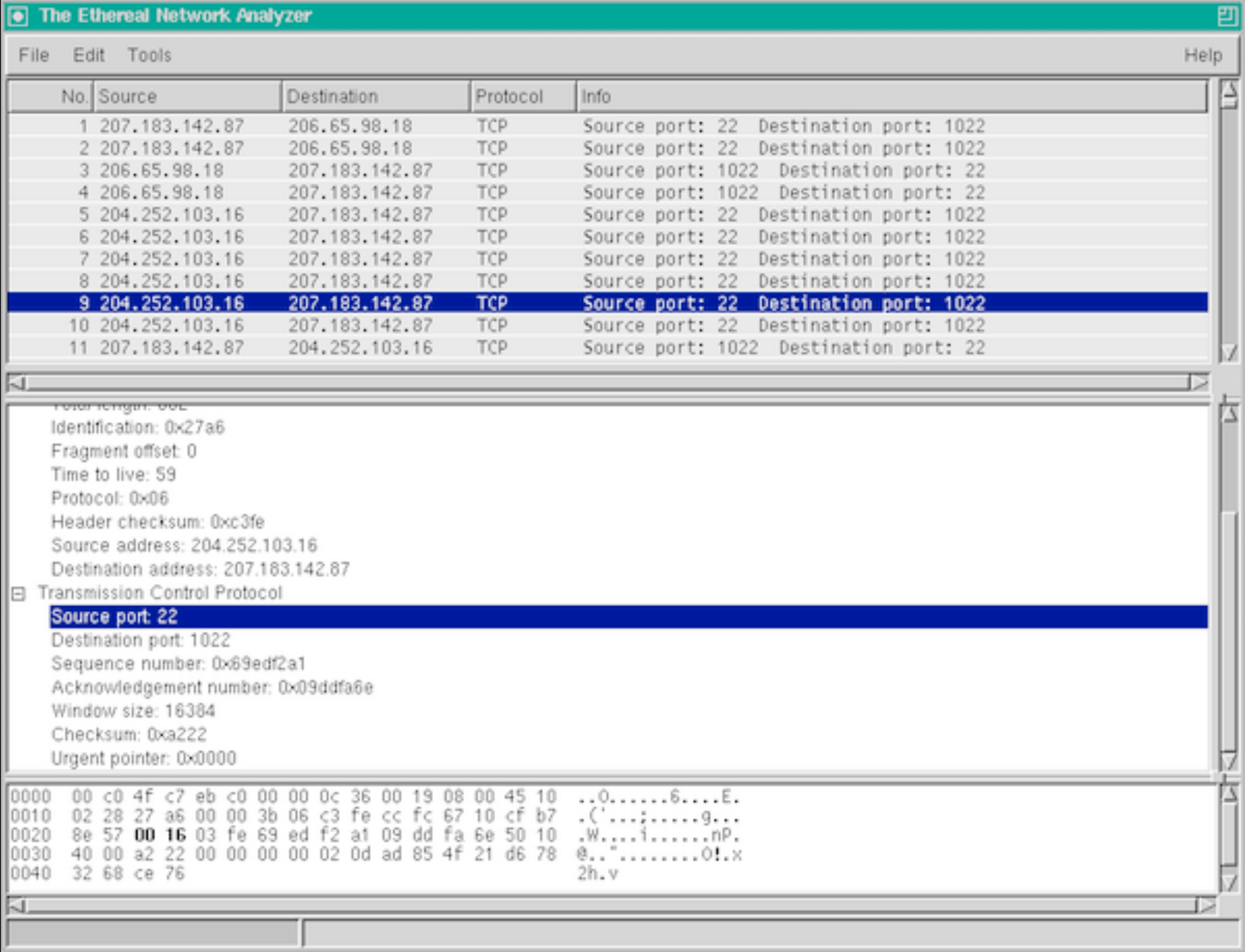
Installed on a Dolch lunchbox computer

Packet sniffing in

1998

\$0

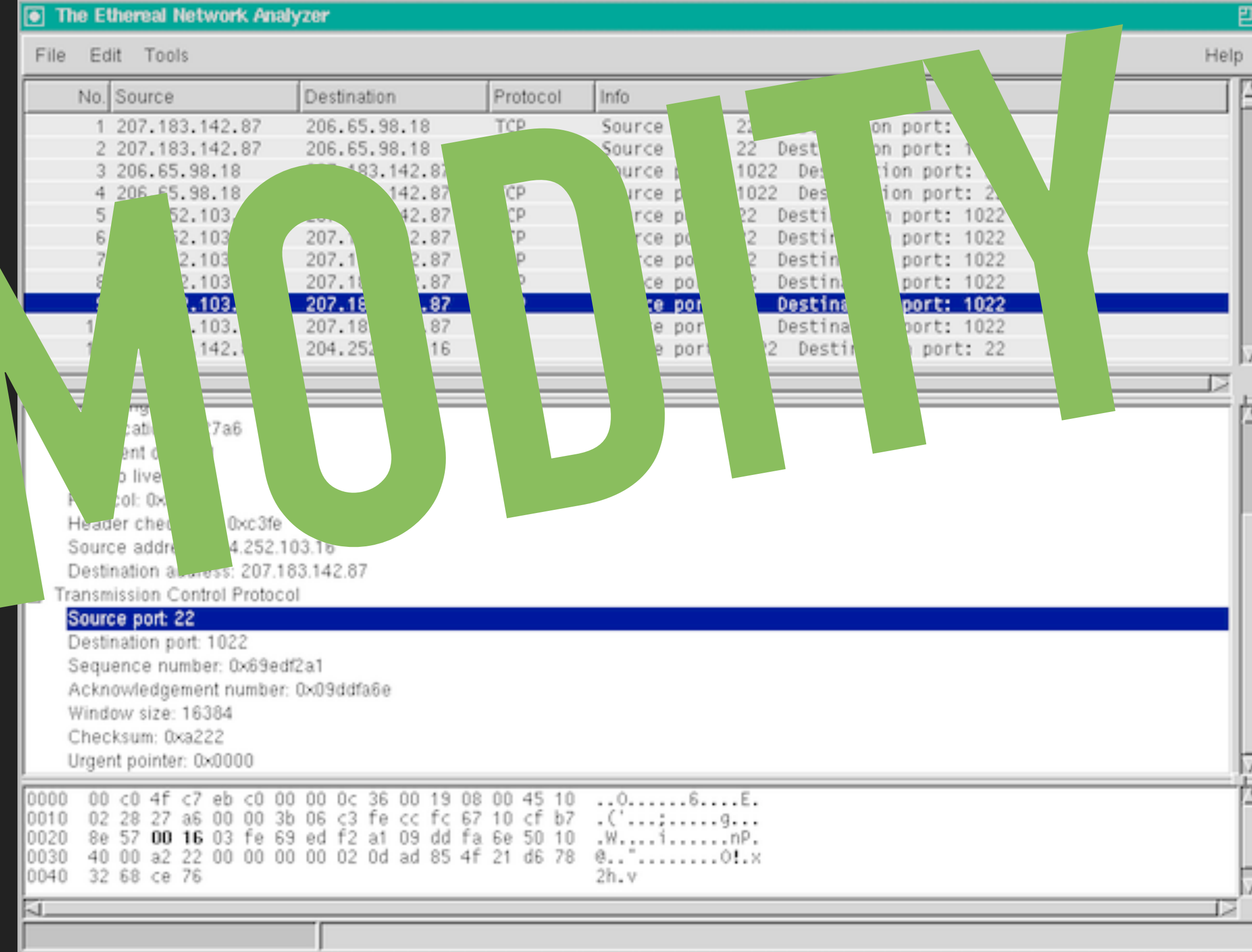
ETHERREAL // WIRESHARK





\$0

COMMODITY



ETHERREAL // WIRESHARK

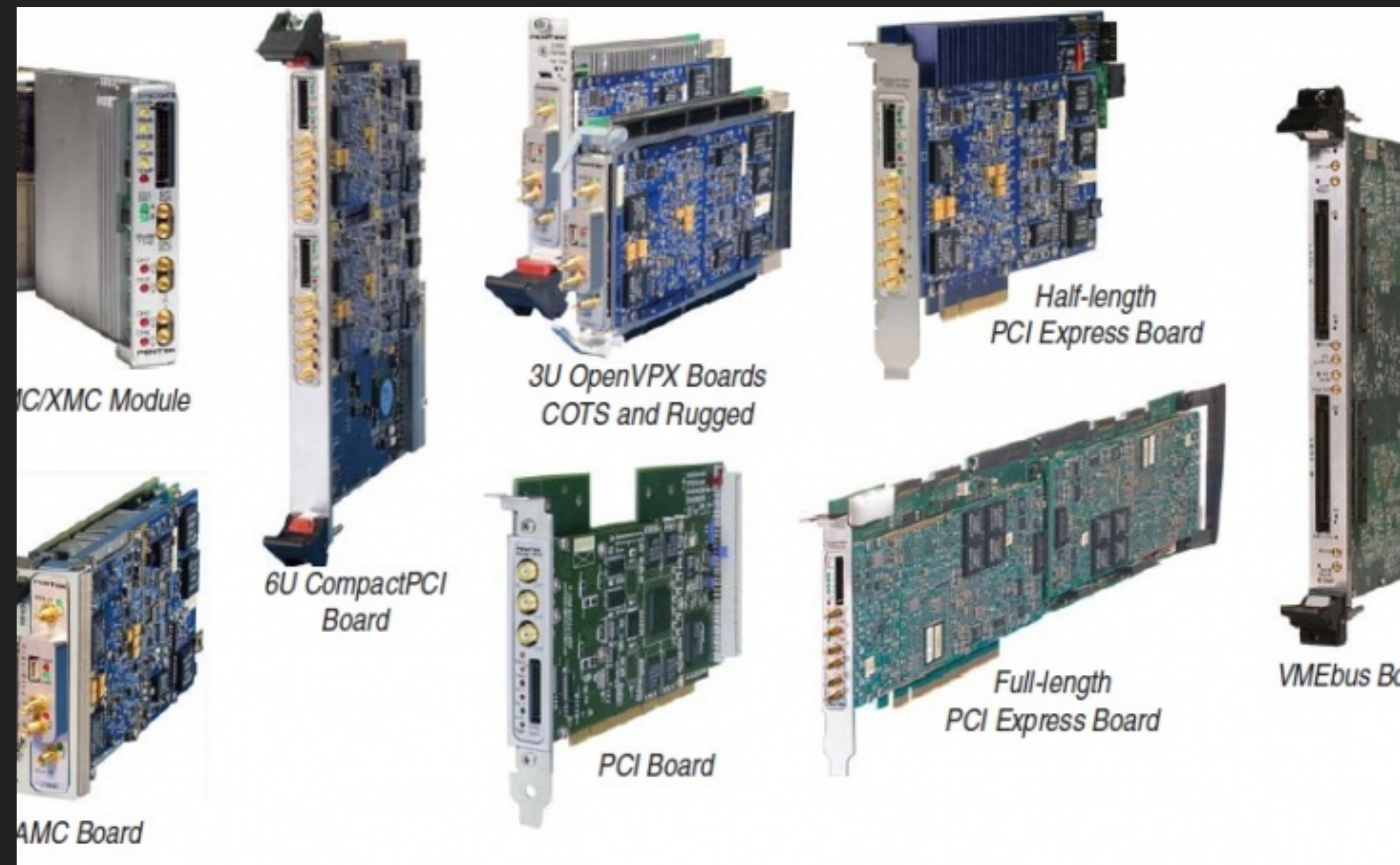
Packet sniffing since the

2000s

[illegible]

TONS OF WIRELESS

>>\$100K

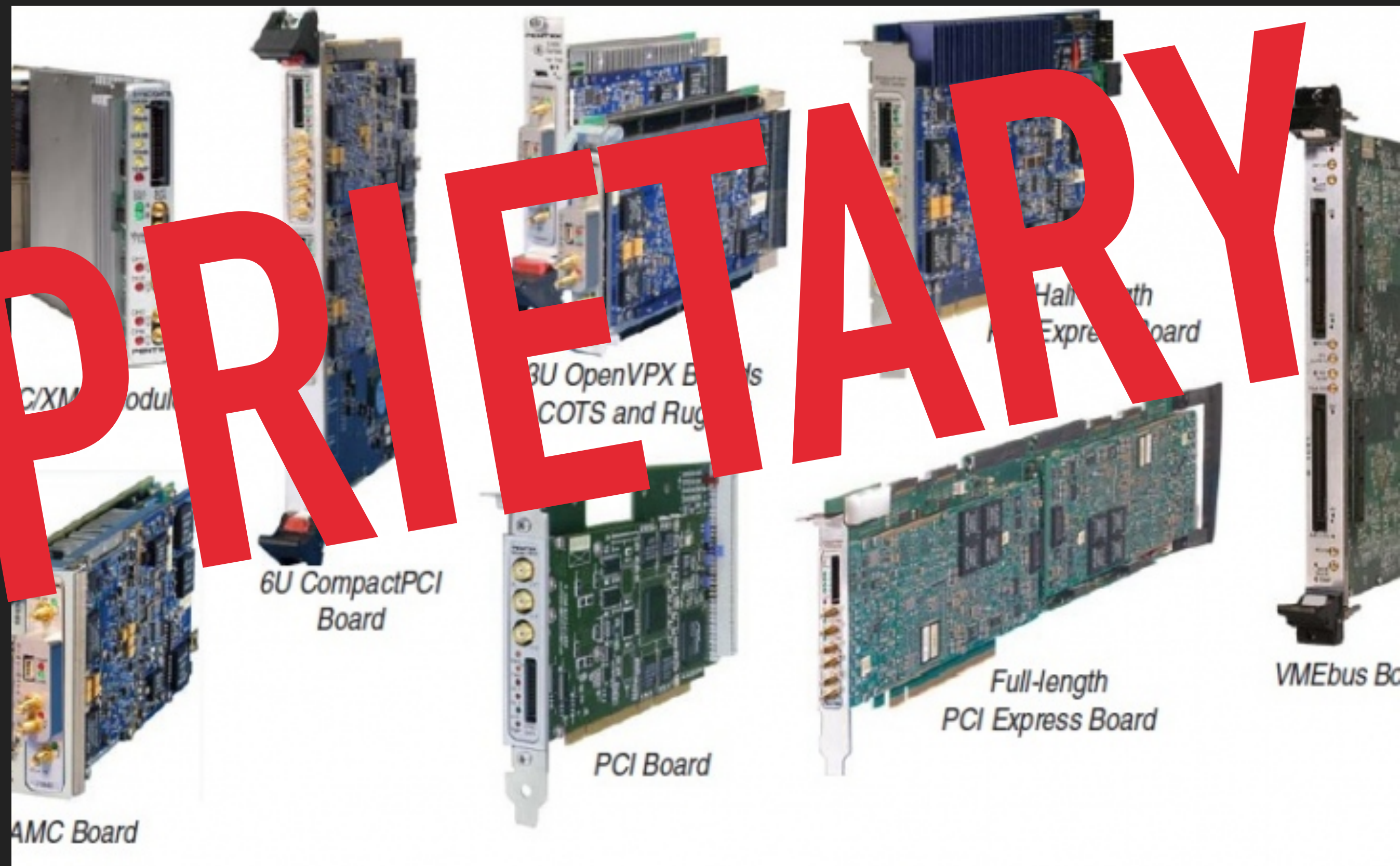


EARLY SDRS



>>\$100K

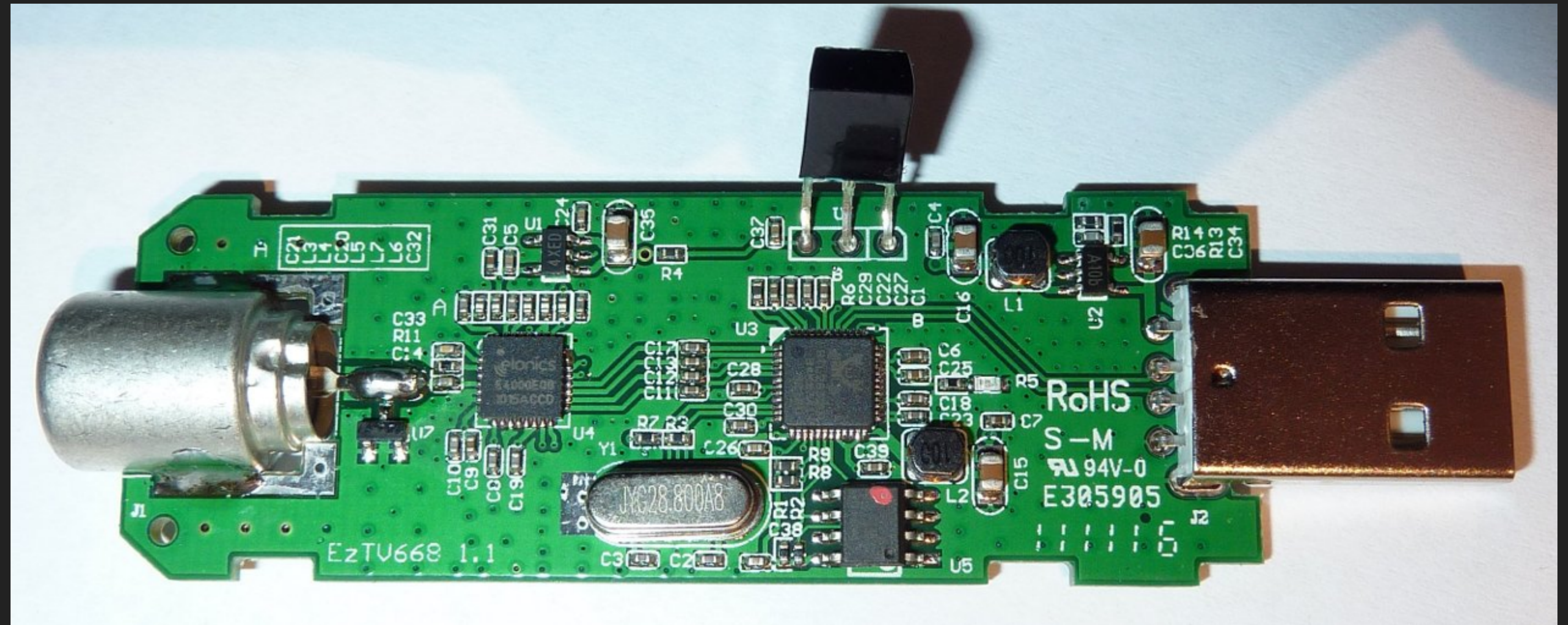
PROPRIETARY



EARLY SDRS

Wireless sniffing in

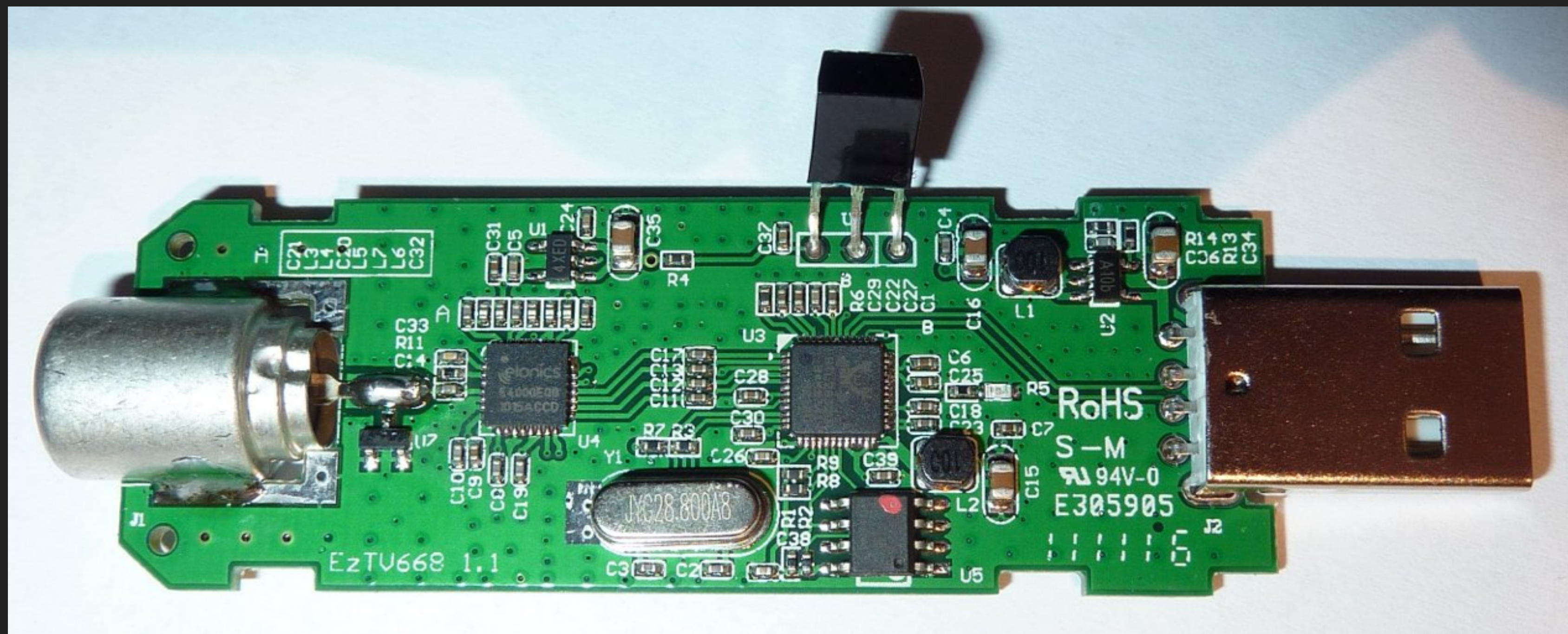
2012



\$8

RTL 2832 USB STICK

(not pictured: promiscuous mode driver)



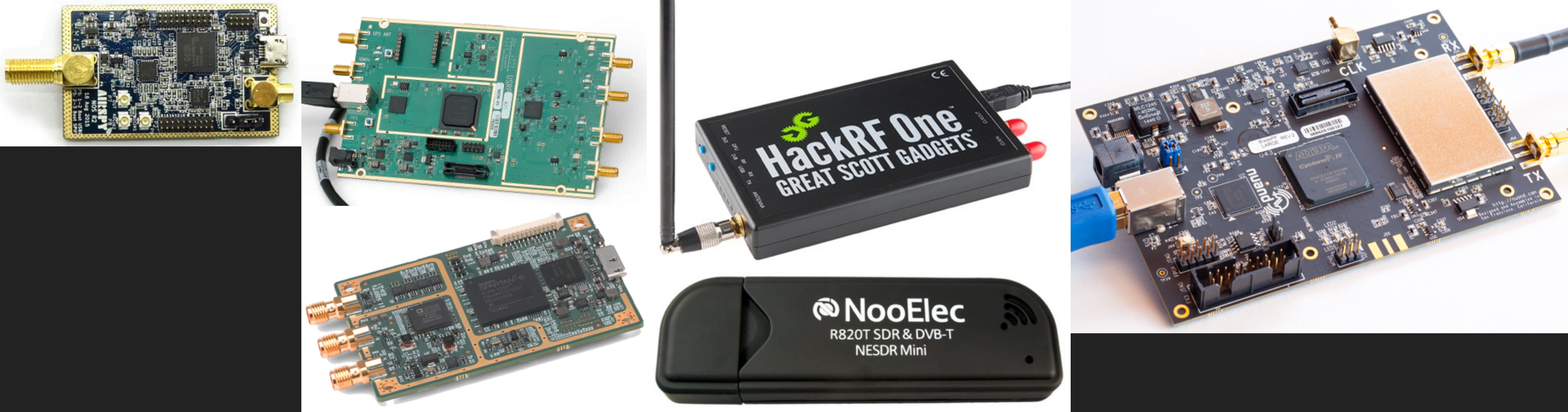
\$8

RTL 2832 USB STICK

(not pictured: promiscuous mode driver)

Wireless sniffing in

2017



\$8 -> \$1150

ALL THE SDRS

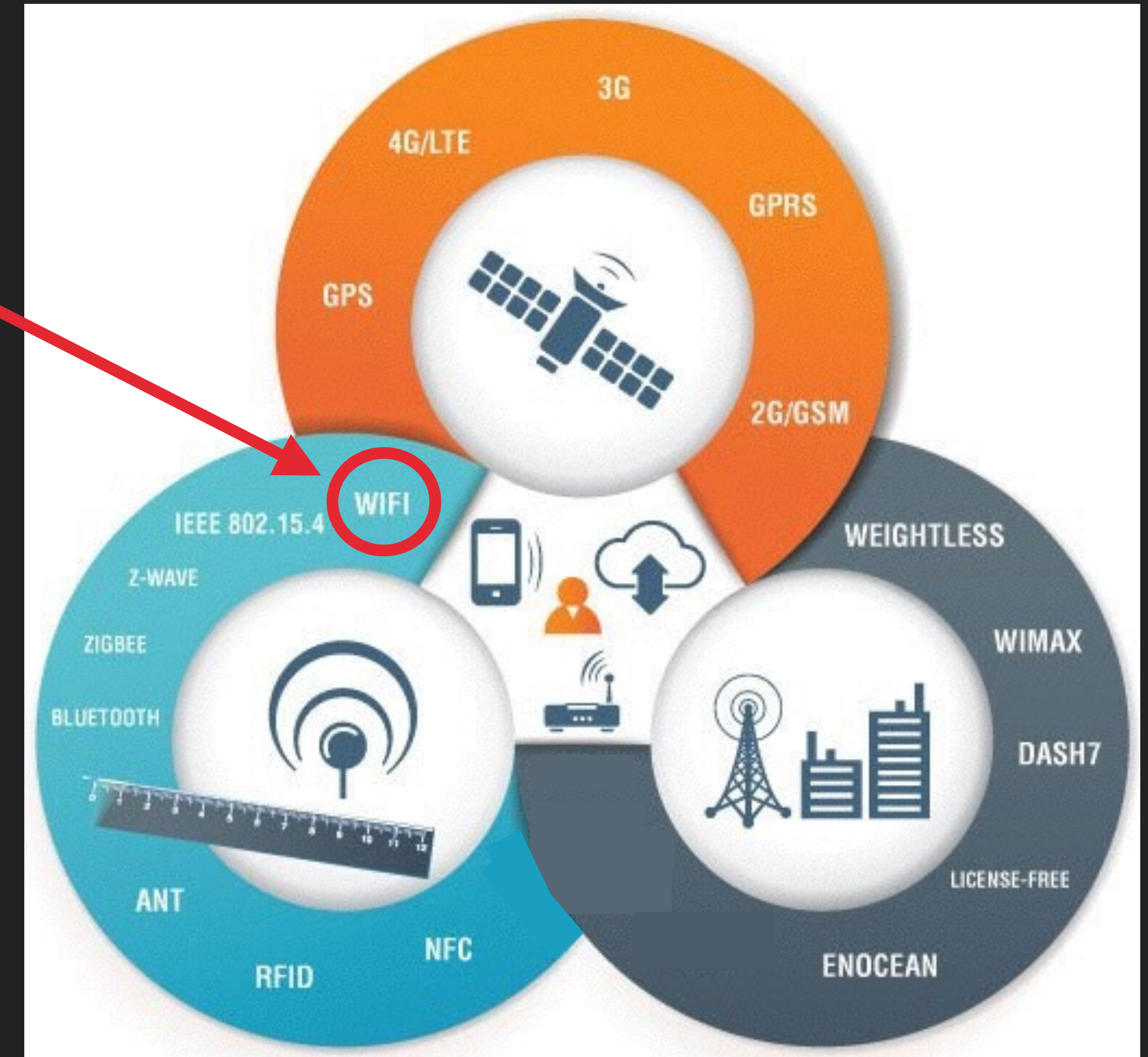


\$8 -> \$1150

ALL THE SDRS

WIRELESS IN 2017

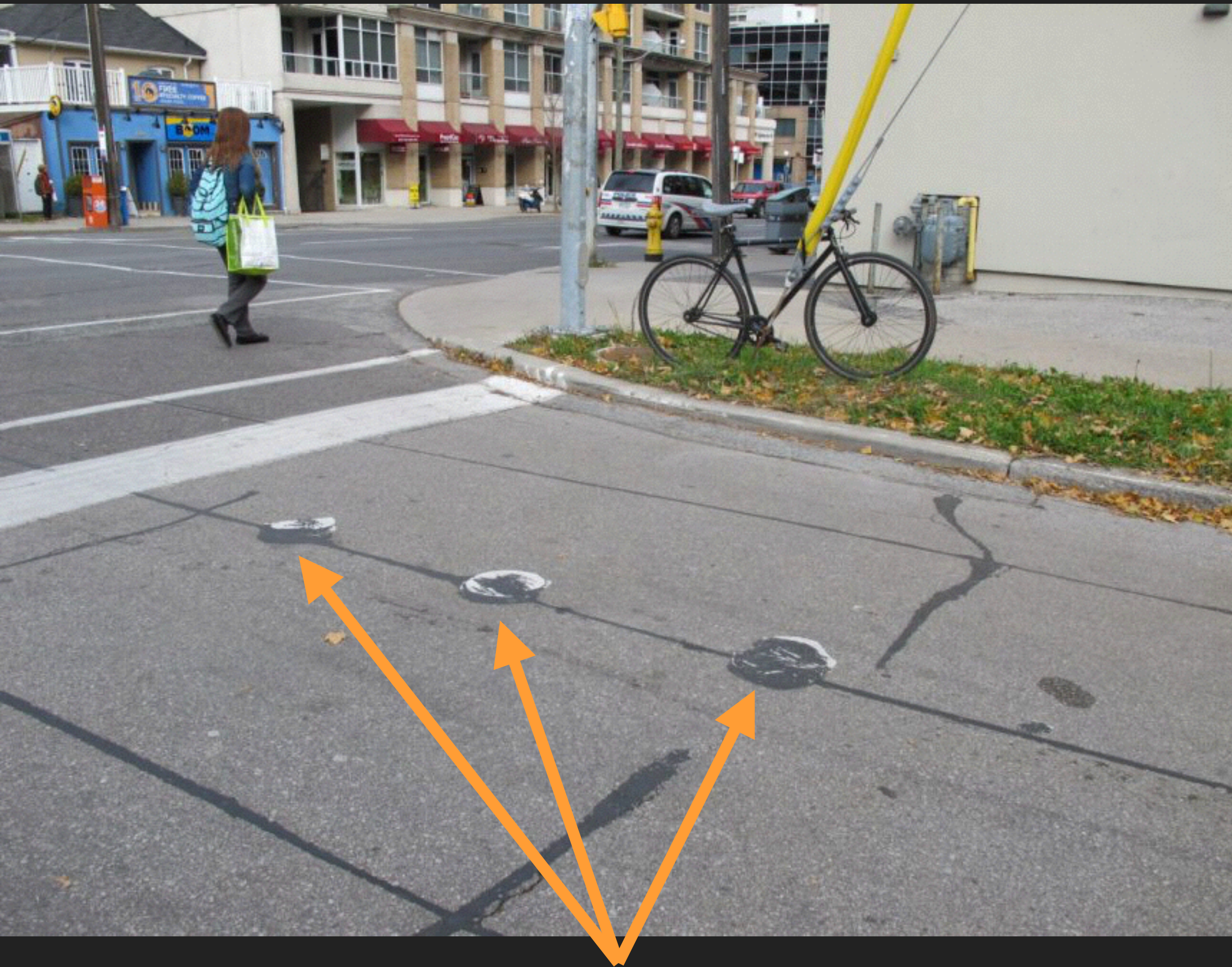
- ▶ 802.11 is just one piece of the puzzle
- ▶ Explosion of **IoT** and **Mobile** means...
 - ▶ There's a PHY for **every** use case
 - ▶ Embedded systems are **everywhere**



EMBEDDED == DESIGN BY COMPROMISE

(or is it Compromised by Design?)

- ▶ Battery powered
- ▶ Limited user interaction
- ▶ Lack of crypto
- ▶ Unsuitable pipes for firmware updates
- ▶ Performance, UX, cost, and delivery are more important than best practices



Literal embedded systems

Industry reliance on

SECURITY THROUGH OBSCURITY

means...

[PIÑATAS]

...TO HACK WIRELESS



SO WHAT DOES IT TAKE...

IP NETWORK SNIFFING IS EASY

- ▶ Interfacing with an IP network is **trivial**
- ▶ Commodity NICs
- ▶ Monitor mode
- ▶ **Known** Layer 2 // MAC frame **protocols**
- ▶ 802.3 // Ethernet for wired IP traffic
- ▶ 802.11 // Wi-Fi for wireless IP traffic

Wi-Fi: en0

Apply a display filter ... < %/>

No.	Time	Length	Source	Destination	Protocol	Info
1	0.000000	323	Pegatron_2f:5e:60	Broadcast	802.11	Beacon frame, SN=2877, FN=0, Flags=.....C, BI=100
2	0.000493	46		Apple_ef:58:1b ...	802.11	Clear-to-send, Flags=.....C
3	0.000569	64	ArrisGro_60:e8:40 ...	Apple_ef:58:1b ...	802.11	802.11 Block Ack, Flags=.....C
4	0.000645	64	ArrisGro_60:e8:40 ...	Apple_ef:58:1b ...	802.11	802.11 Block Ack, Flags=.....C
5	0.002632	263	62:86:8c:60:e8:40	Broadcast	802.11	Beacon frame, SN=3541, FN=0, Flags=.....C, BI=100
6	0.002738	46		Apple_ef:58:1b ...	802.11	Clear-to-send, Flags=.....C
7	0.002839	64	ArrisGro_60:e8:40 ...	Apple_ef:58:1b ...	802.11	802.11 Block Ack, Flags=.....C
8	0.003137	64	ArrisGro_60:e8:40 ...	Apple_ef:58:1b ...	802.11	802.11 Block Ack, Flags=.....C
9	0.003216	64	ArrisGro_60:e8:40 ...	Apple_ef:58:1b ...	802.11	802.11 Block Ack, Flags=.....C
10	0.003347	46		Apple_ef:58:1b ...	802.11	Acknowledgement, Flags=.....C
11	0.003610	46		Apple_a2:7c:fb ...	802.11	Acknowledgement, Flags=.....C
12	0.005019	66	Pegatron_2f:5e:60	Apple_a2:7c:fb	802.11	Action, SN=617, FN=0, Flags=.....C
13	0.015792	278	AsustekC_69:91:d8	Broadcast	802.11	Beacon frame, SN=1102, FN=0, Flags=.....C, BI=100
14	0.018874	46		Apple_a0:c7:f6 ...	802.11	Clear-to-send, Flags=.....C
15	0.019147	64	Pegatron_f6:37:7a ...	Apple_a0:c7:f6 ...	802.11	802.11 Block Ack, Flags=.....C
16	0.020334	52	ArrisGro_60:e8:40 ...	Apple_ef:58:1b ...	802.11	Request-to-send, Flags=.....C
17	0.021202	336	ArrisGro_60:e8:47	Apple_ef:58:1b	802.11	QoS Data, SN=795, FN=0, Flags=p....F..
18	0.024139	254	Pegatron_2f:5e:63	Broadcast	802.11	Beacon frame, SN=562, FN=0, Flags=.....C, BI=100
19	0.026608	304	ArrisGro_60:e8:40	Broadcast	802.11	Beacon frame, SN=2060, FN=0, Flags=.....C, BI=100
20	0.027003	52	ArrisGro_60:e8:40 ...	Apple_ef:58:1b ...	802.11	Request-to-send, Flags=.....C
21	0.027806	296	ArrisGro_60:e8:47	Apple_ef:58:1b	802.11	QoS Data, SN=796, FN=0, Flags=p....F..
22	0.028386	52	ArrisGro_60:e8:40 ...	Apple_ef:58:1b ...	802.11	Request-to-send, Flags=.....C
23	0.029055	194	ArrisGro_60:e8:47	Apple_ef:58:1b	802.11	QoS Data, SN=797, FN=0, Flags=p....F..
24	0.029550	52	ArrisGro_60:e8:40 ...	Apple_ef:58:1b ...	802.11	Request-to-send, Flags=.....C
25	0.030198	189	ArrisGro_60:e8:47	Apple_ef:58:1b	802.11	QoS Data, SN=798, FN=0, Flags=p....F..
26	0.030700	52	ArrisGro_60:e8:40 ...	Apple_ef:58:1b ...	802.11	Request-to-send, Flags=.....C
27	0.031431	241	ArrisGro_60:e8:47	Apple_ef:58:1b	802.11	QoS Data, SN=799, FN=0, Flags=p....F..
28	0.031923	52	ArrisGro_60:e8:40 ...	Apple_ef:58:1b ...	802.11	Request-to-send, Flags=.....C

▶ Frame 1: 323 bytes on wire (2584 bits), 323 bytes captured (2584 bits) on interface 0

▶ PPI version 0, 32 bytes

▼ 802.11 radio information

PHY type: 802.11g (6)

Proprietary mode: None (0)

Data rate: 1.0 Mb/s

Channel: 6

Frequency: 2437 MHz

Signal strength (dBm): -51 dBm

Noise level (dBm): -95 dBm

TSF timestamp: 1481797185

▶ [Duration: 2520 us]

▼ IEEE 802.11 Beacon frame, Flags:C

Type/Subtype: Beacon frame (0x0000)

▼ Frame Control Field: 0x0000

.... 00 = Version: 0

.... 00.. = Type: Management frame (0)

1000 = Subtype: 8

▼ Flags: 0x00

.... 00 = DS status: Not leaving DS or network is operating in AD-HOC mode (To DS: 0 From DS: 0) (0x0)

.... 0... = Retrv: Frame is not being retransmitted

```

0020 80 00 00 00 ff ff ff ff ff ff 10 05 01 2f 5e 60 ...../^`
0030 10 05 01 2f 5e 60 d0 b3 1c d5 d4 5c 20 01 00 00 .../^`...
0040 64 00 31 04 00 0d 43 42 43 49 2d 46 42 42 42 2d d.1...CB CI-FBBB-
0050 32 2e 34 01 08 82 84 8b 96 8c 12 98 24 03 01 06 2.4....$....
0060 05 04 00 01 00 08 07 06 55 53 20 01 0b 1e 2a 01 .....US...*
0070 00 32 04 b0 48 60 6c 2d 1a ad 01 1b ff ff ff 00 .2..H`l- .....
0080 00 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 .....
0090 00 00 00 3d 16 06 00 05 00 00 00 00 00 00 00 00 ...=.....
00a0 00 00 00 00 00 00 00 00 00 00 00 4a 0e 14 00 0a .....J....
00b0 00 2c 01 c8 00 14 00 05 00 19 00 7f 08 01 00 00 .....

```


WIRELESS* NETWORK SNIFFING IS HARD

*non-802.11

- ▶ Network interface is **totally non-trivial**
 - ▶ Your Wi-Fi NIC can't sniff wireless traffic from your home security system
- ▶ **Arbitrary Layer 1 // PHYs**
 - ▶ There are many ways to make a PHY
 - ▶ 802.11 // Wi-Fi is just one example
- ▶ How does one speak arbitrary RF?

Enter...

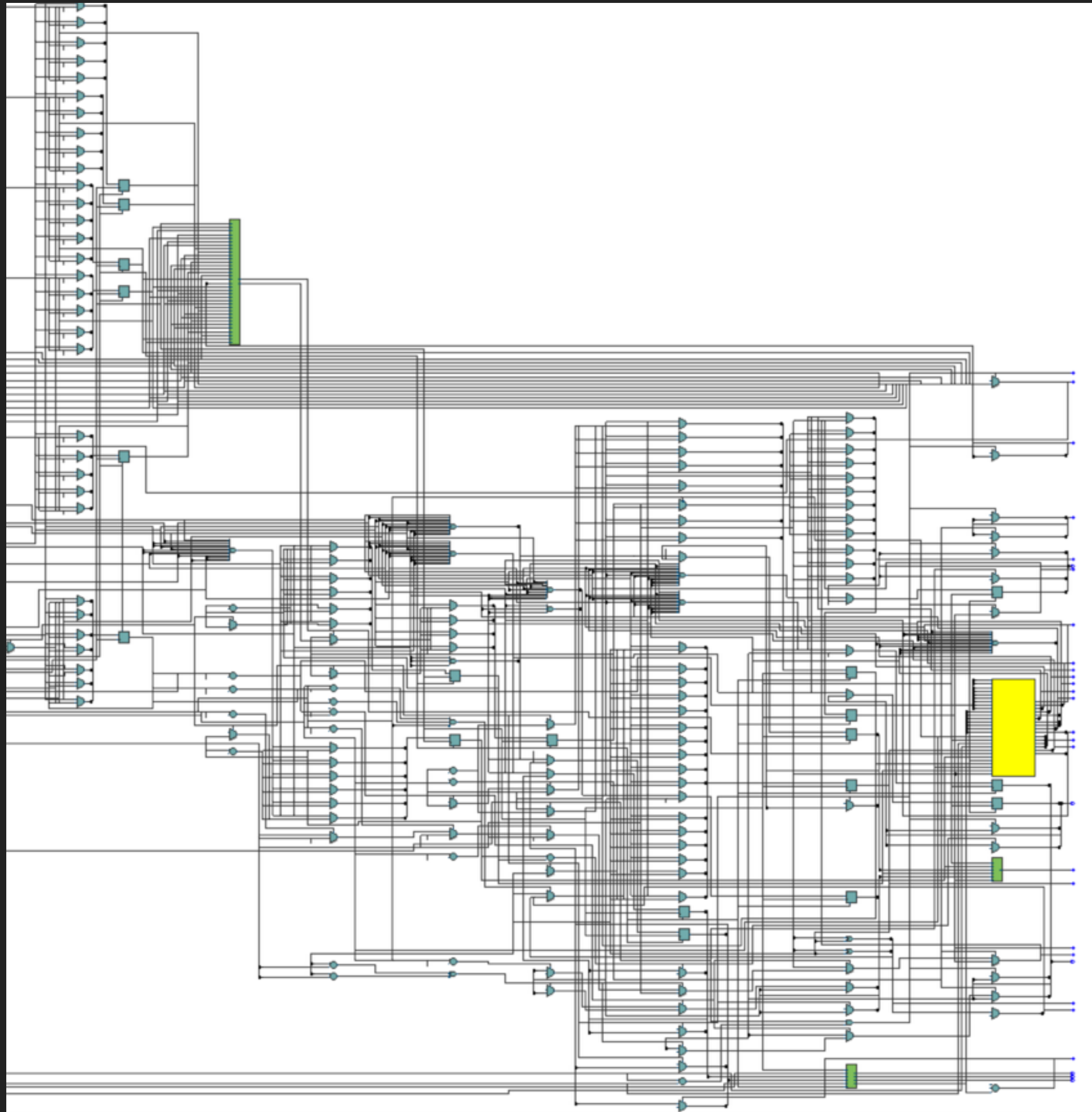
SOFTWARE DEFINED RADIO

SOFTWARE DEFINED RADIO (SDR)

- ▶ Board with **flexible** wideband RF frontend
- ▶ Captures raw radio spectrum
- ▶ Shuttles RF I/Q samples to DSP or host
- ▶ Implement **arbitrary** PHYs in:
 - ▶ Software
 - ▶ FPGA HDL

Fast iteration!
Flexible!



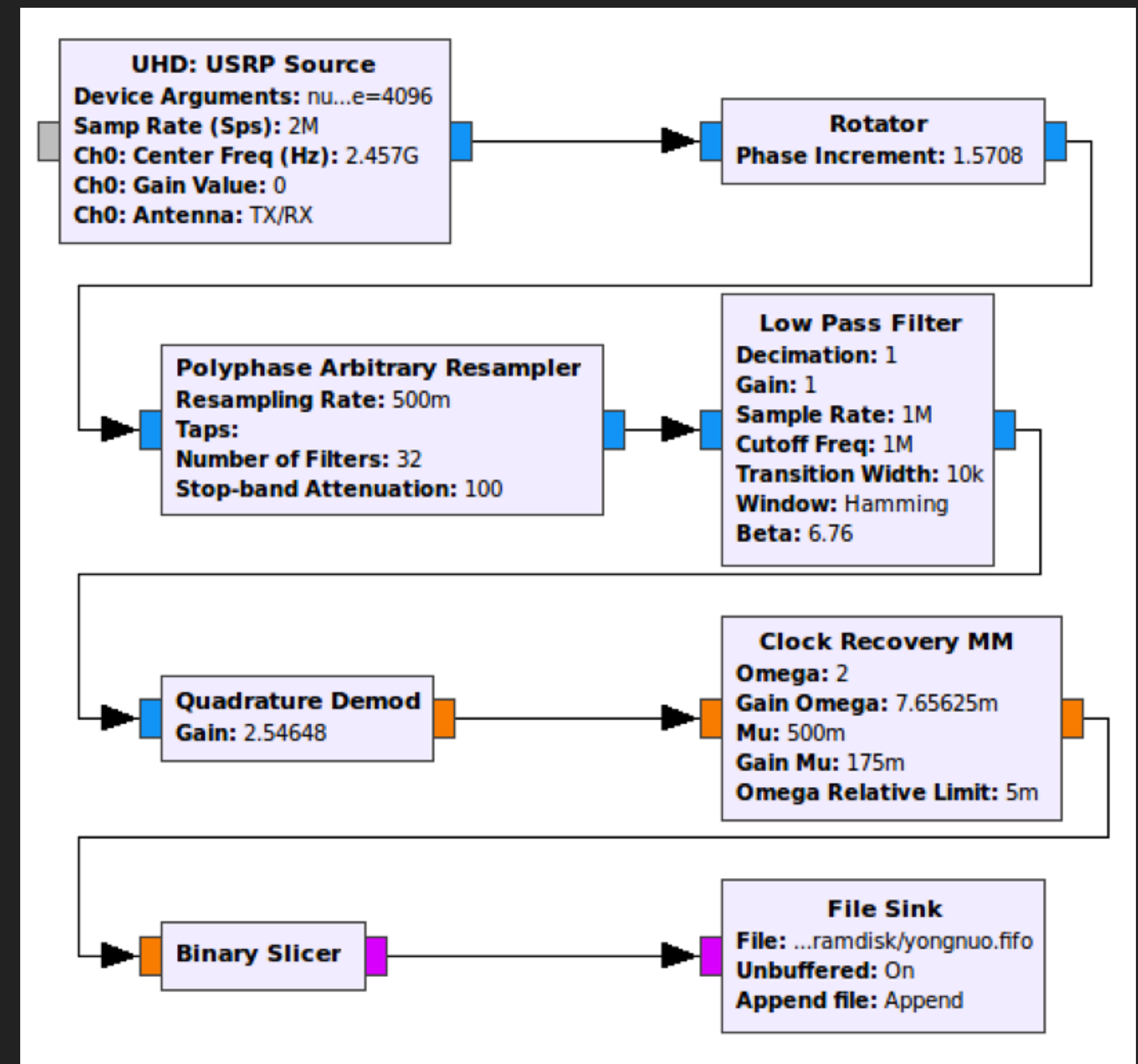


SDR DOWNSIDES

- ▶ Requires:
 - ▶ Lots of computing power, power hungry
 - ▶ Complex and esoteric domain knowledge
- ▶ Left: RTL for one of Matt's 802.15.4 decoders
- ▶ To learn about SDR and PHYs:
 - ▶ "So You Want to Hack Radios" series from Shmoocon, Troopers, and HITB2017AMS

OPEN SOURCE SOFTWARE RADIO

- ▶ Vibrant open-source SDR community
- ▶ GNU Radio
 - ▶ Modular signal processing framework
 - ▶ Abstracts away hard math
- ▶ Ossmann and Balint's video tutorials
- ▶ Wireless Village here at DEF CON

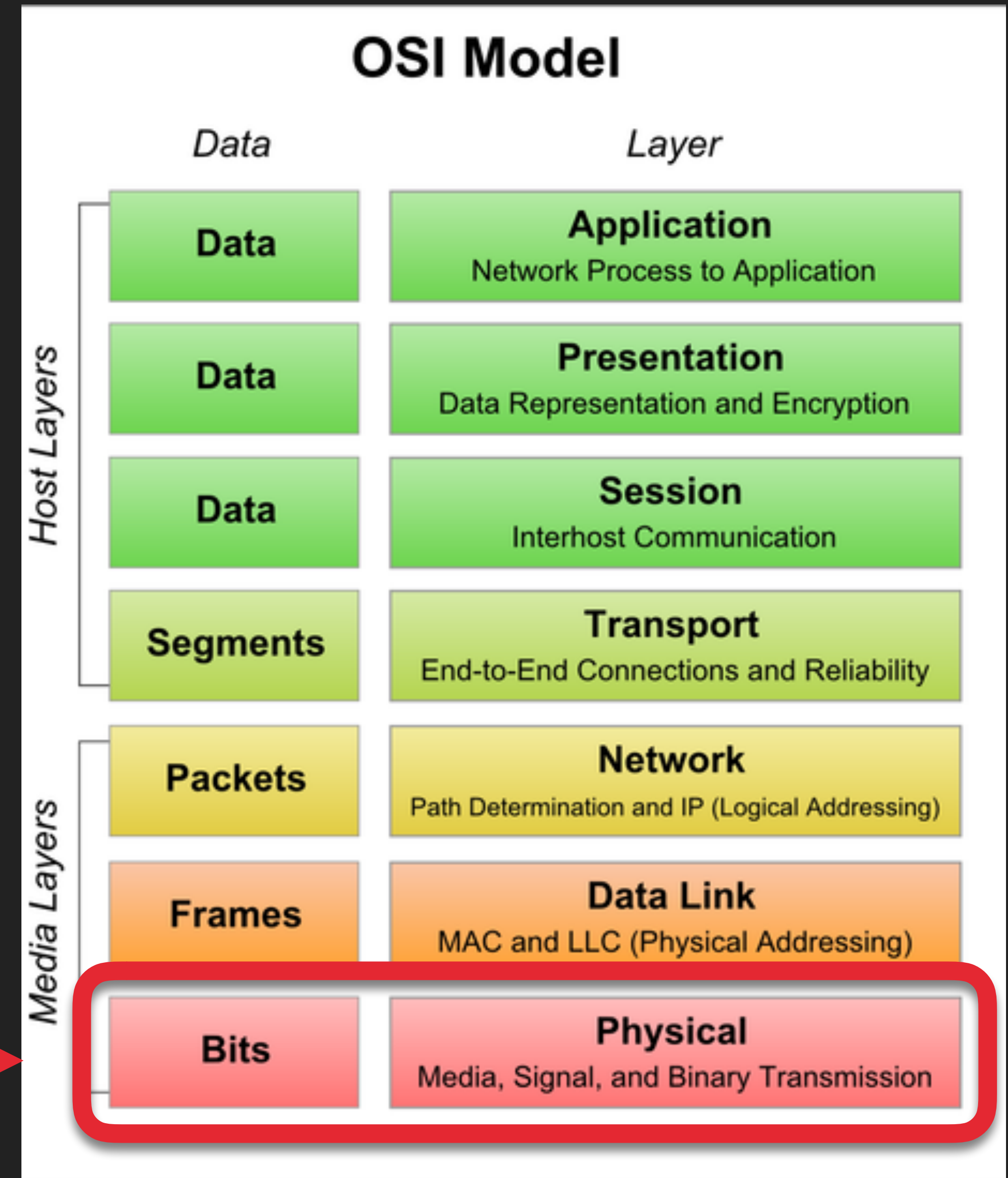


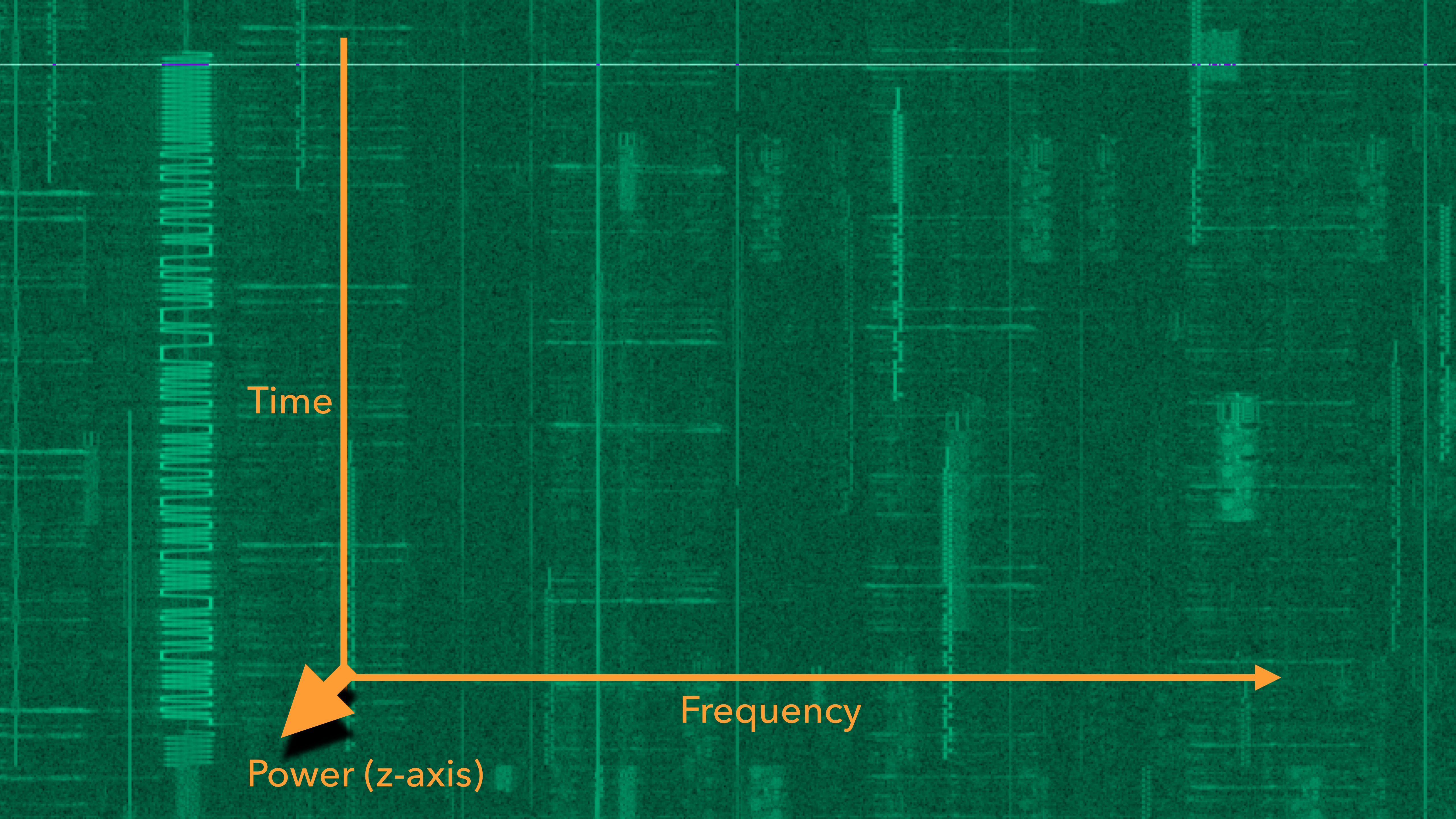
GROSSLY SIMPLIFIED

RF CONCEPTS

PHYSICAL LAYER (PHY)

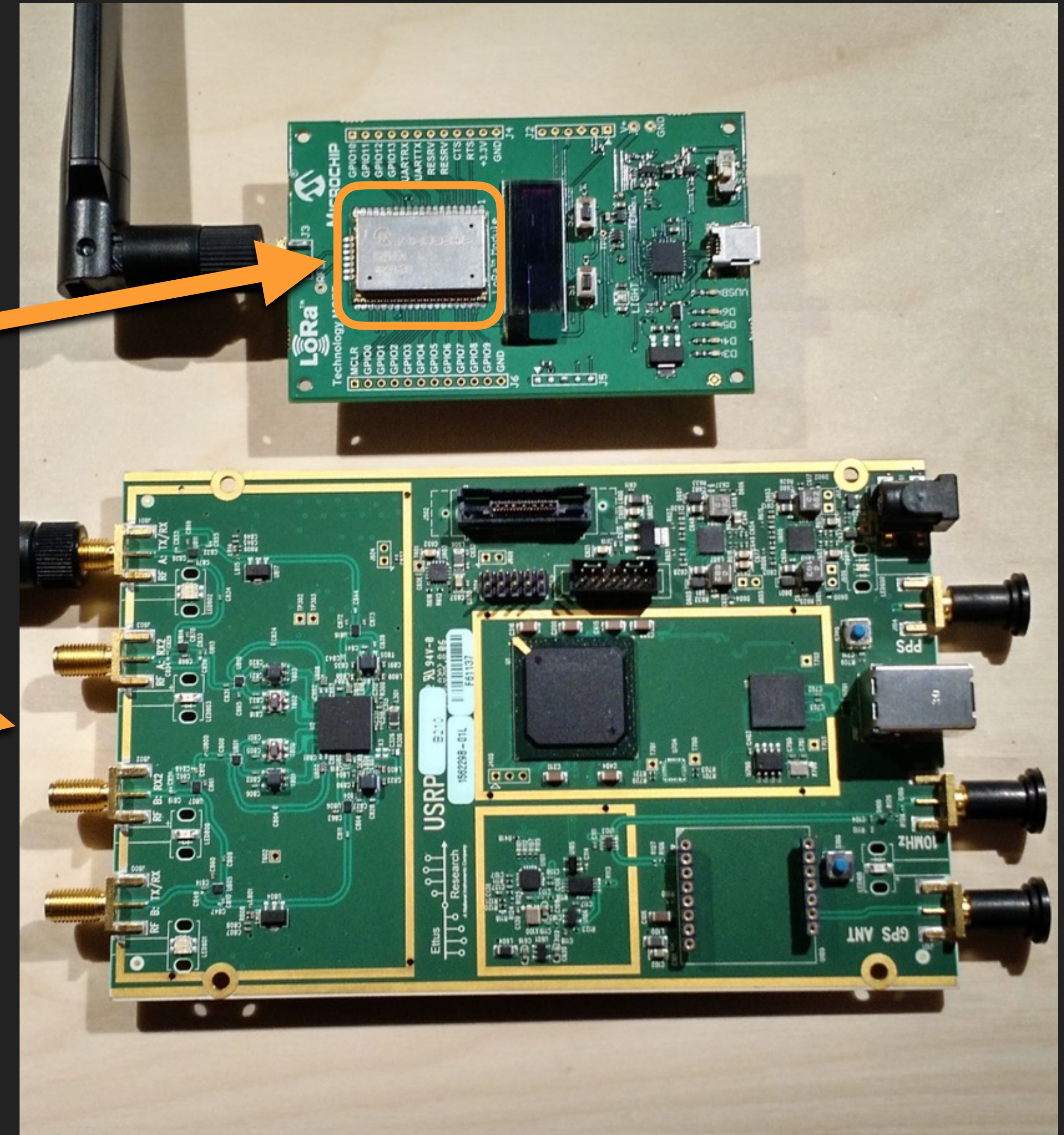
- ▶ Lowest layer in communication stack
- ▶ In wired protocols: 1s and 0s defined in voltage, timing, and wiring
- ▶ In wireless: patterns of energy being sent over **RF medium**



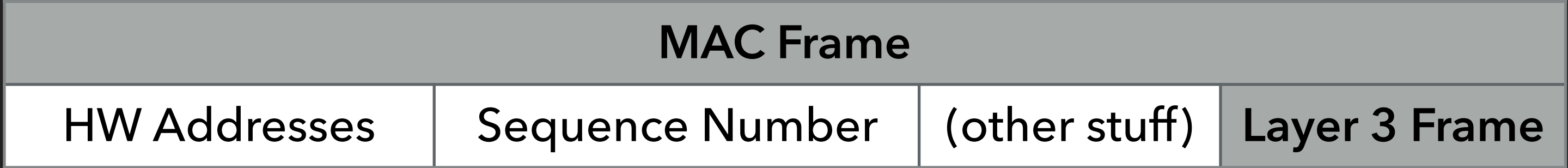


MANIPULATING RF

- ▶ ... is done with a radio
- ▶ Hardware defined
- ▶ Software defined radio (SDR)
- ▶ Modulation
 - ▶ Function: how digital values are mapped to RF energy



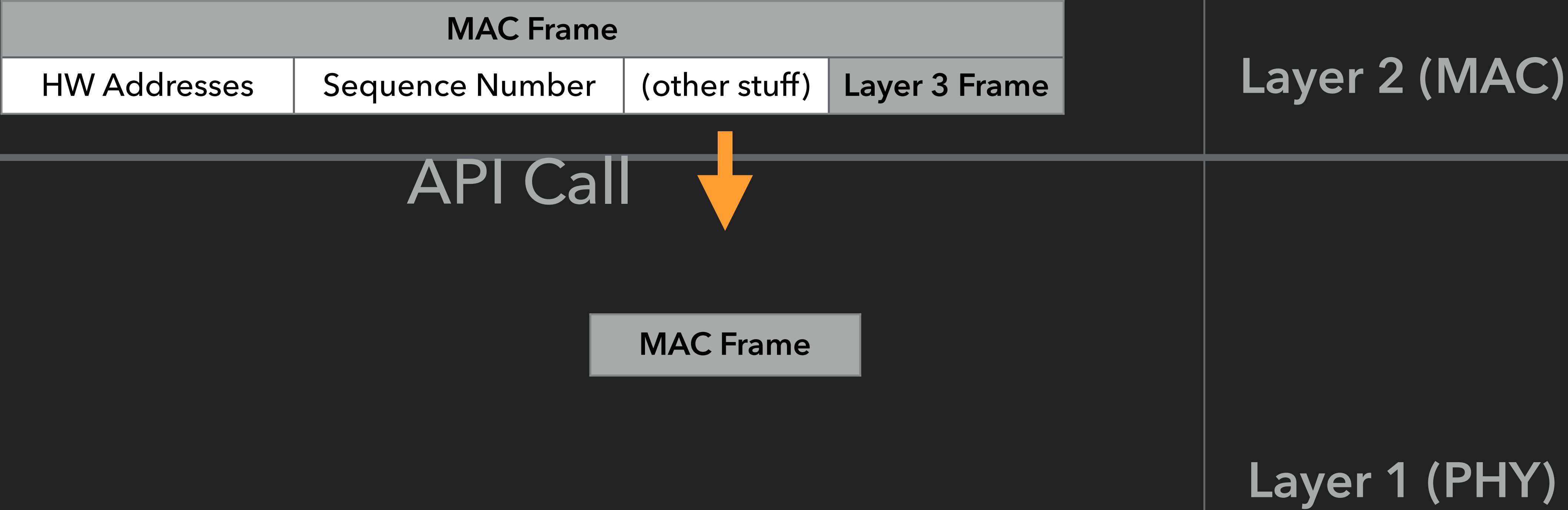
HOW TRANSMITTING WORKS



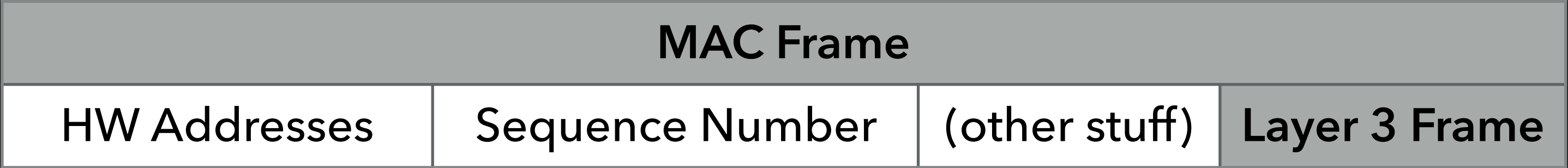
Layer 2 (MAC)

Layer 1 (PHY)

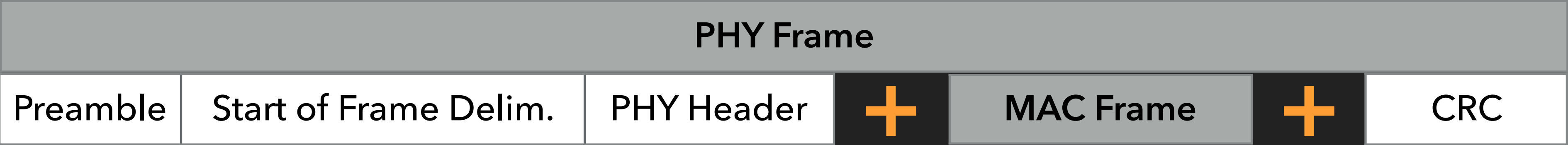
HOW TRANSMITTING WORKS



HOW TRANSMITTING WORKS

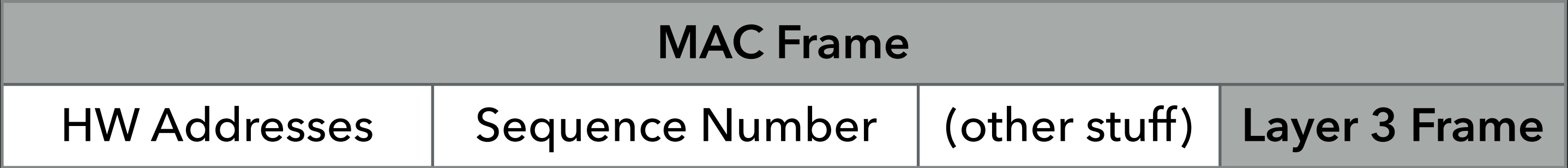


Layer 2 (MAC)



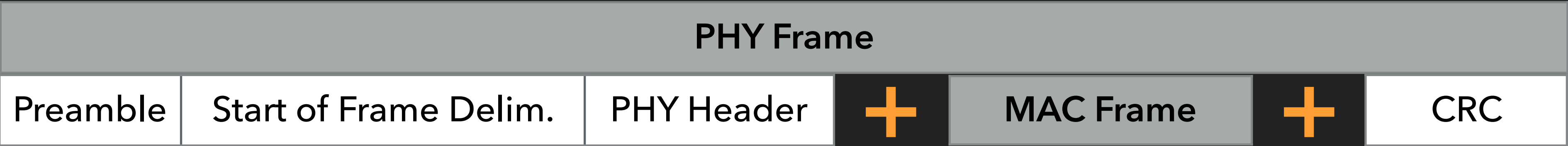
Layer 1 (PHY)

HOW TRANSMITTING WORKS



Layer 2 (MAC)

API Call



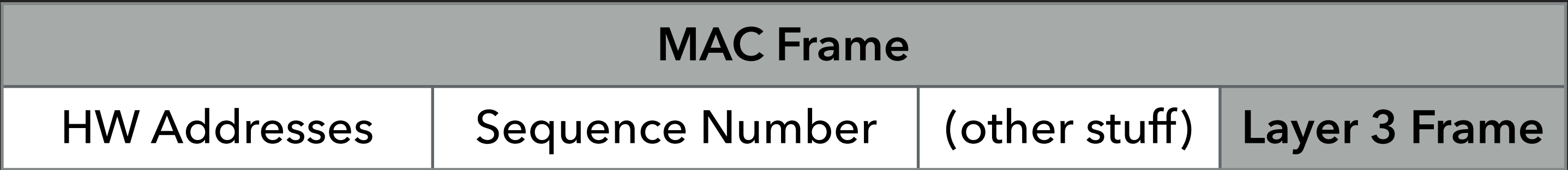
Modulation



(Maps 1s and 0s to
electrical phenomena)

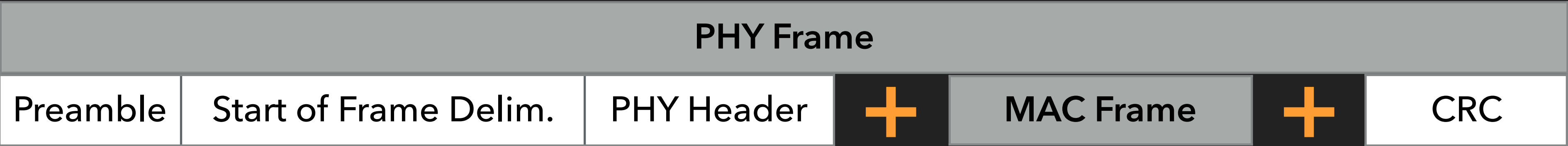
Layer 1 (PHY)

HOW TRANSMITTING WORKS



Layer 2 (MAC)

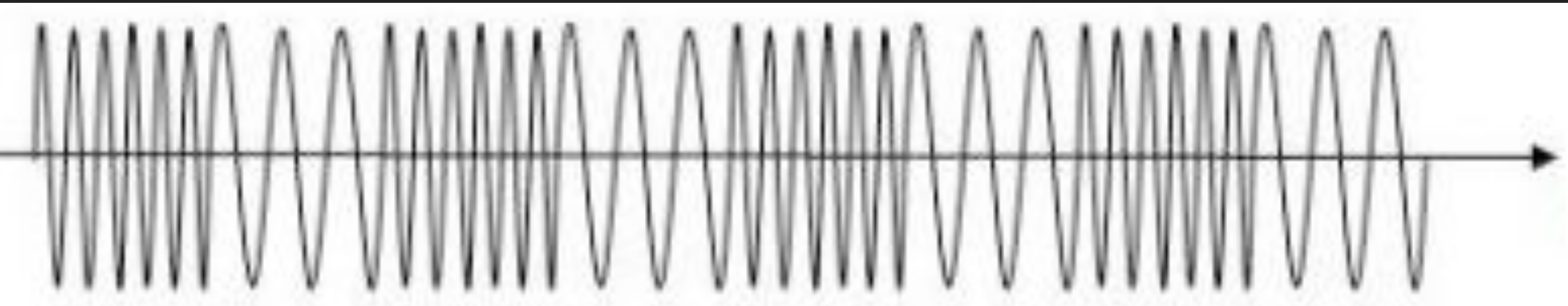
API Call



Modulation

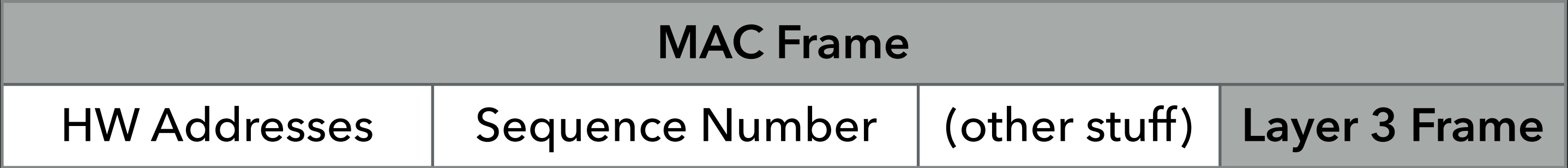


(Maps 1s and 0s to electrical phenomena)



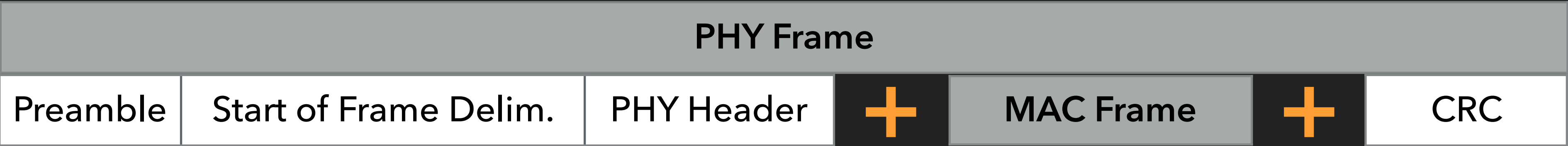
Layer 1 (PHY)

HOW TRANSMITTING WORKS



Layer 2 (MAC)

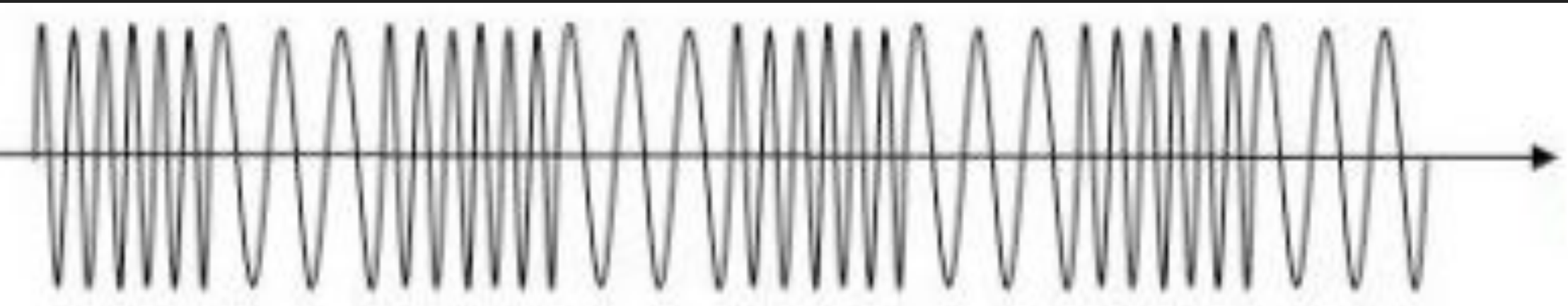
API Call



Modulation



(Maps 1s and 0s to electrical phenomena)



(to antenna/RF frontend)



Layer 1 (PHY)

HOW RECEIVING WORKS

Layer 2 (MAC)

Layer 1 (PHY)

 (from antenna)

HOW RECEIVING WORKS

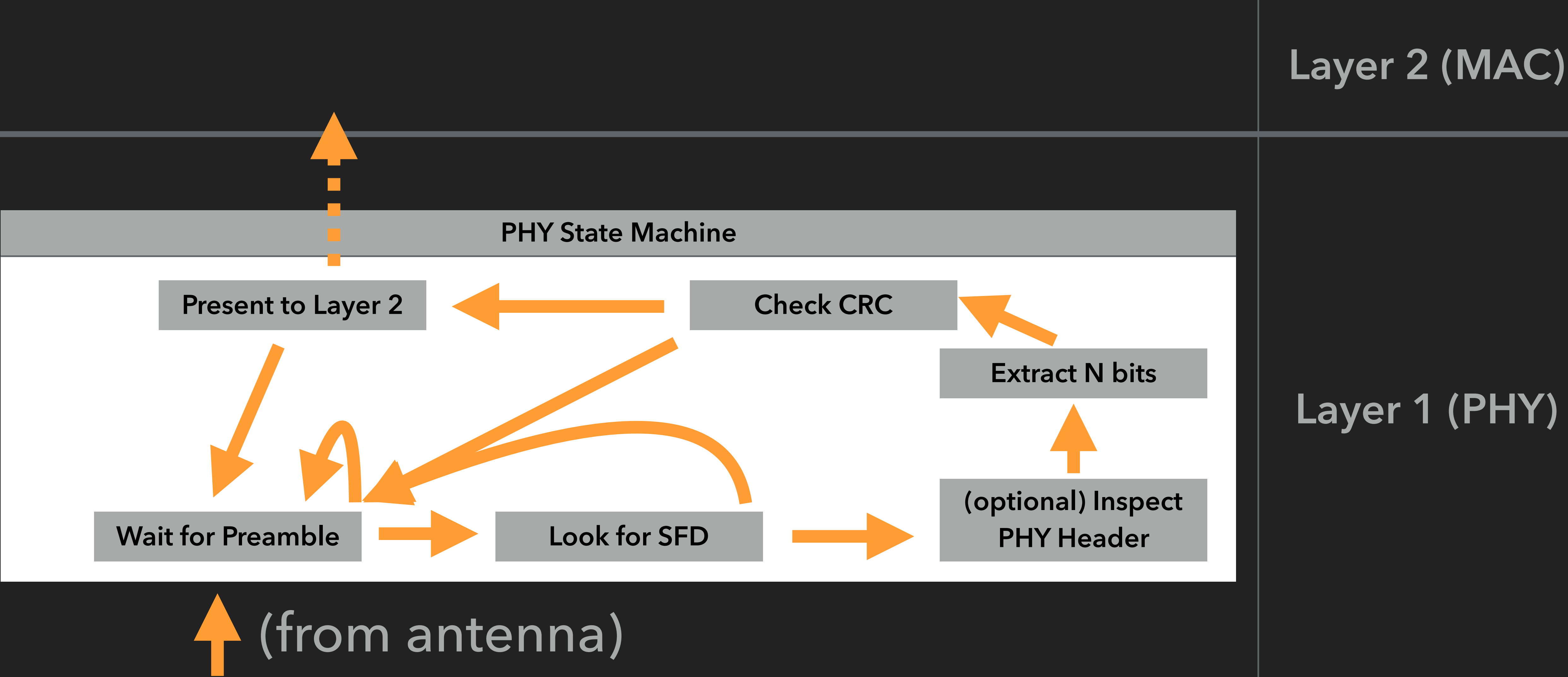
Layer 2 (MAC)

PHY State Machine

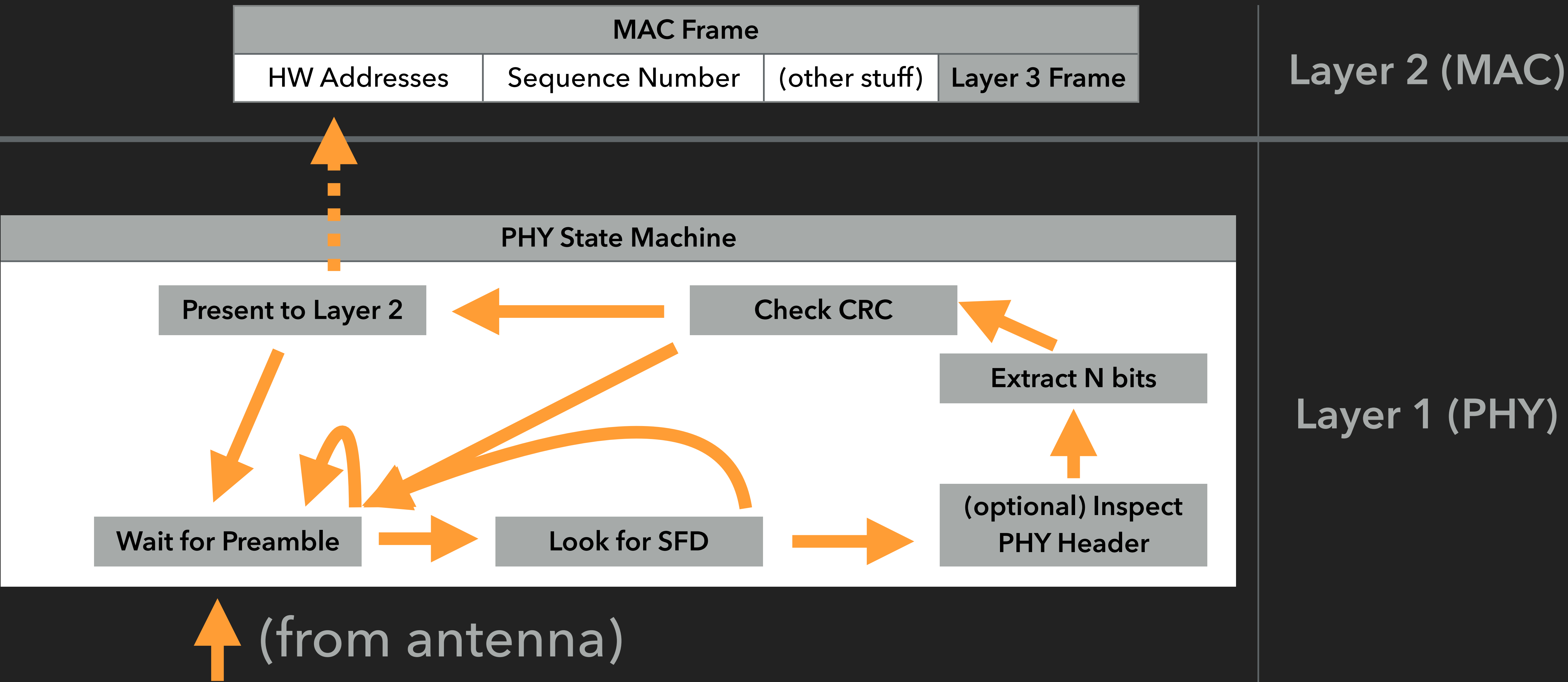
Layer 1 (PHY)

↑ (from antenna)

HOW RECEIVING WORKS



HOW RECEIVING WORKS



KEY CONCEPTS

- ▶ Radios are **state machines**
 - ▶ They are **deterministic**: not magic
- ▶ Complicated RF environments **inform radio state machine design**
 - ▶ Tons of contention
 - ▶ Radios have features to compensate
- ▶ Opportunities for **tactical abuse**

CLASSIFYING RF ATTACK METHODS

METHODS OF EXPLOITATION

CLASSIFYING RF ATTACK METHODS

- ▶ For each attack category, we'll show:
 1. Method: **how** the attack is performed
 2. Impact: **what** the attack enables
 3. Analogue: **equivalent attack** on wired/IP network, if one exists
 4. Limitations: **mitigations**, whether incidental or intentional
 5. Example: **relevant examples** of this type of attack
 6. Proof: **demo**

RECONNAISSANCE // DATA LEAKAGE

SNIFFING



SNIFFING OVERVIEW

- ▶ Method
 - ▶ **Observe traffic** from nearby networks
- ▶ Impact
 - ▶ Data loss // leakage
 - ▶ Reconnaissance for further development
- ▶ Wired Analogue
 - ▶ **None!** Promiscuity is a unique characteristic of RF

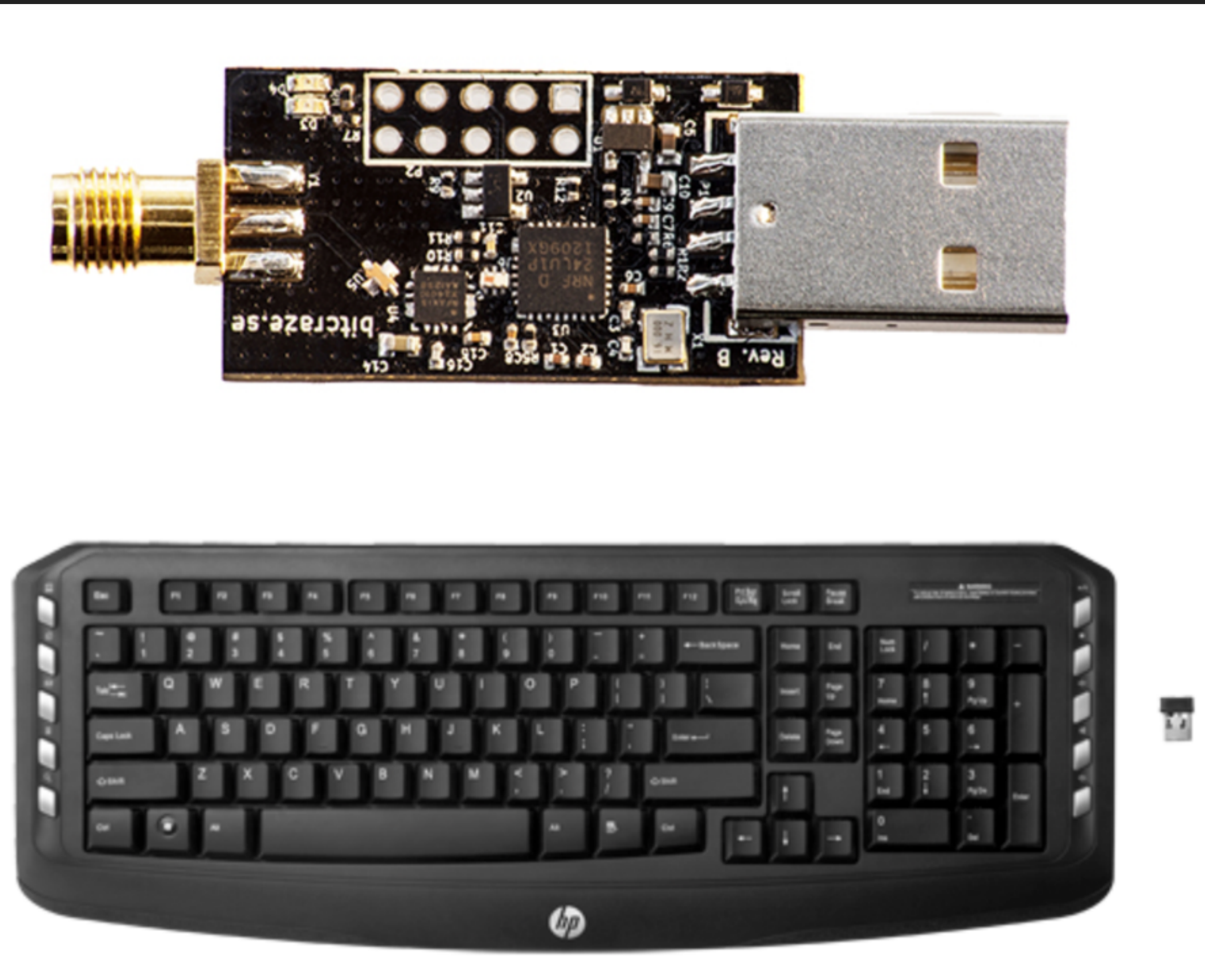


SNIFFING APPLIED

- ▶ Limitations
 - ▶ Physical range, availability of PHY interface
- ▶ Example
 - ▶ Key extraction during ZigBee lock pairing
 - ▶ Keystrokes from unencrypted keyboards



SNIFFING DEMO



- ▶ HP Classic Wireless Desktop
 - ▶ Unencrypted wireless keyboard
 - ▶ Keystroke sniffing demo
- ▶ **Proprietary** 2.4 GHz wireless protocol
 - ▶ MOSART Semiconductor transceiver
 - ▶ No encryption
 - ▶ No official protocol documentation
 - ▶ Over-the-air compatible with nRF24L USB dongles

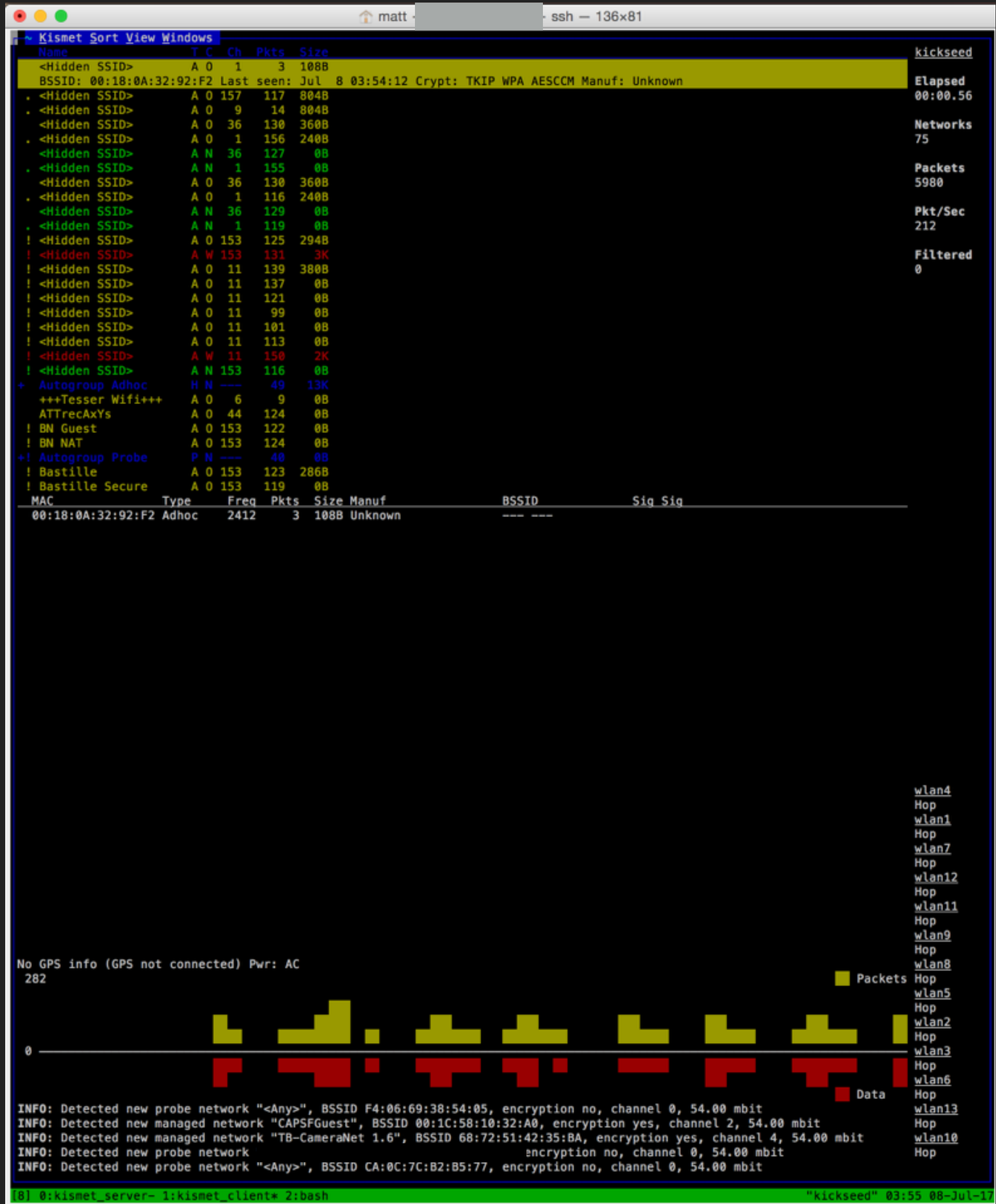
A photograph of a dark-colored car at night. The car's interior is visible through the windows, showing a steering wheel and a laptop computer on the dashboard displaying a map. On the roof of the car, there are several antennas and electronic equipment, suggesting it is a specialized vehicle for wireless operations. The background is dark, with some faint lights visible.

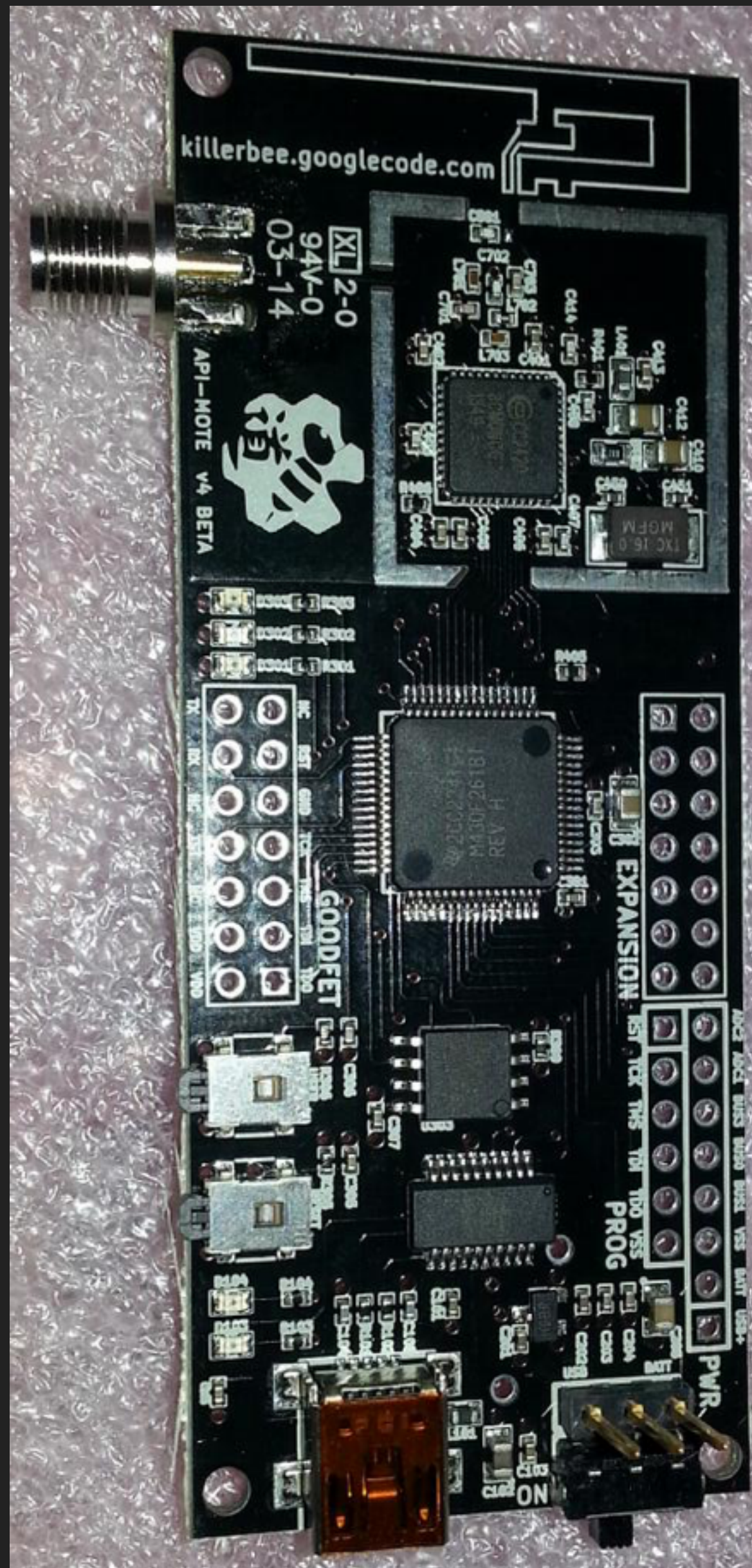
DEVICE DISCOVERY // ATTACK DEVELOPMENT

WARDRIVING

WARDRIVING OVERVIEW

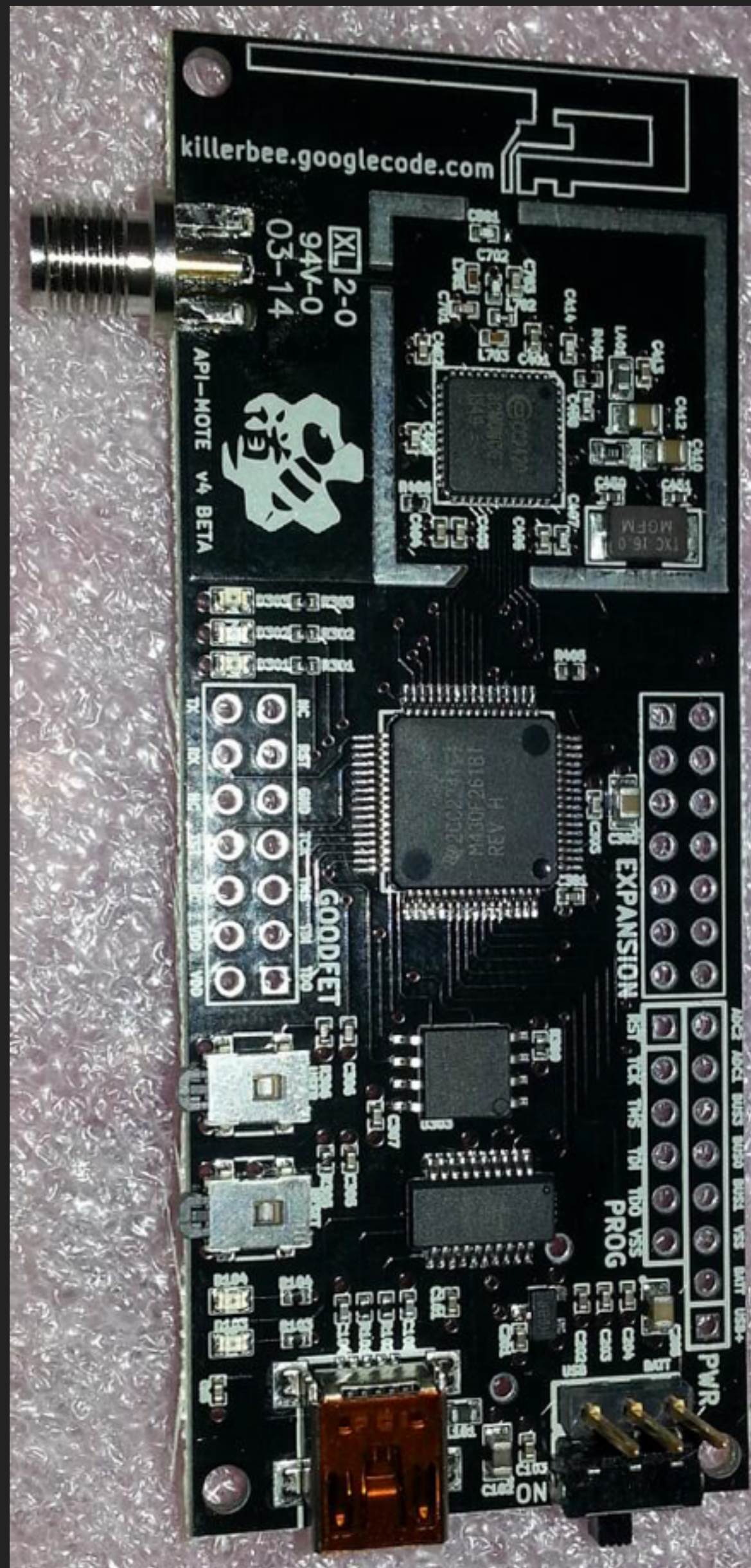
- ▶ Method
 - ▶ **Scan** for identifying features on a protocol of interest
 - ▶ (Optional) **actively beacon** to induce traffic
- ▶ Impact
 - ▶ **Discovery** of exploitable devices or networks
- ▶ Wired Analogue
 - ▶ Port scanning, service discovery





WARDRIVING APPLIED

- ▶ Limitations
 - ▶ Physical range, number of channels, easy to spot if defender knows to look for it
- ▶ Examples
 - ▶ 802.11 AP discovery
 - ▶ Beaconsing for an 802.15.4/ZigBee network coordinator



WARDRIVING DEMO

- ▶ 802.15.4 wardriving
- ▶ **Killerbee** exploitation framework
 - ▶ Crafts broadcast beacon requests
- ▶ **ApiMote** hardware radio board
 - ▶ Sends out requests
 - ▶ Records responses from 802.15.4 network coordinators



DEVICE CONTROL // STATE MANIPULATION

REPLAY

REPLAY ATTACK OVERVIEW

- ▶ Method
 - ▶ **Re-transmit** a previously captured PHY frame
- ▶ Impact
 - ▶ Command injection: change the state of a device or something on the network
- ▶ Wired Analogue
 - ▶ Same (replays exists on wired networks too)

Ctrl+C

Ctrl+V



REPLAY ATTACKS APPLIED

- ▶ Limitations
 - ▶ Defeated by **freshness** (sequence number) or **authentication** (cryptography/trust)
- ▶ Example
 - ▶ 2017 Dallas tornado emergency siren attack

REPLAY ATTACK DEMO



- ▶ Fortress Security Safeguard panic button
 - ▶ Tornado siren surrogate 😊
- ▶ 433 MHz on-off keying
 - ▶ No **freshness** or **authentication**
 - ▶ Raw IQ replay OR decode/resynthesize

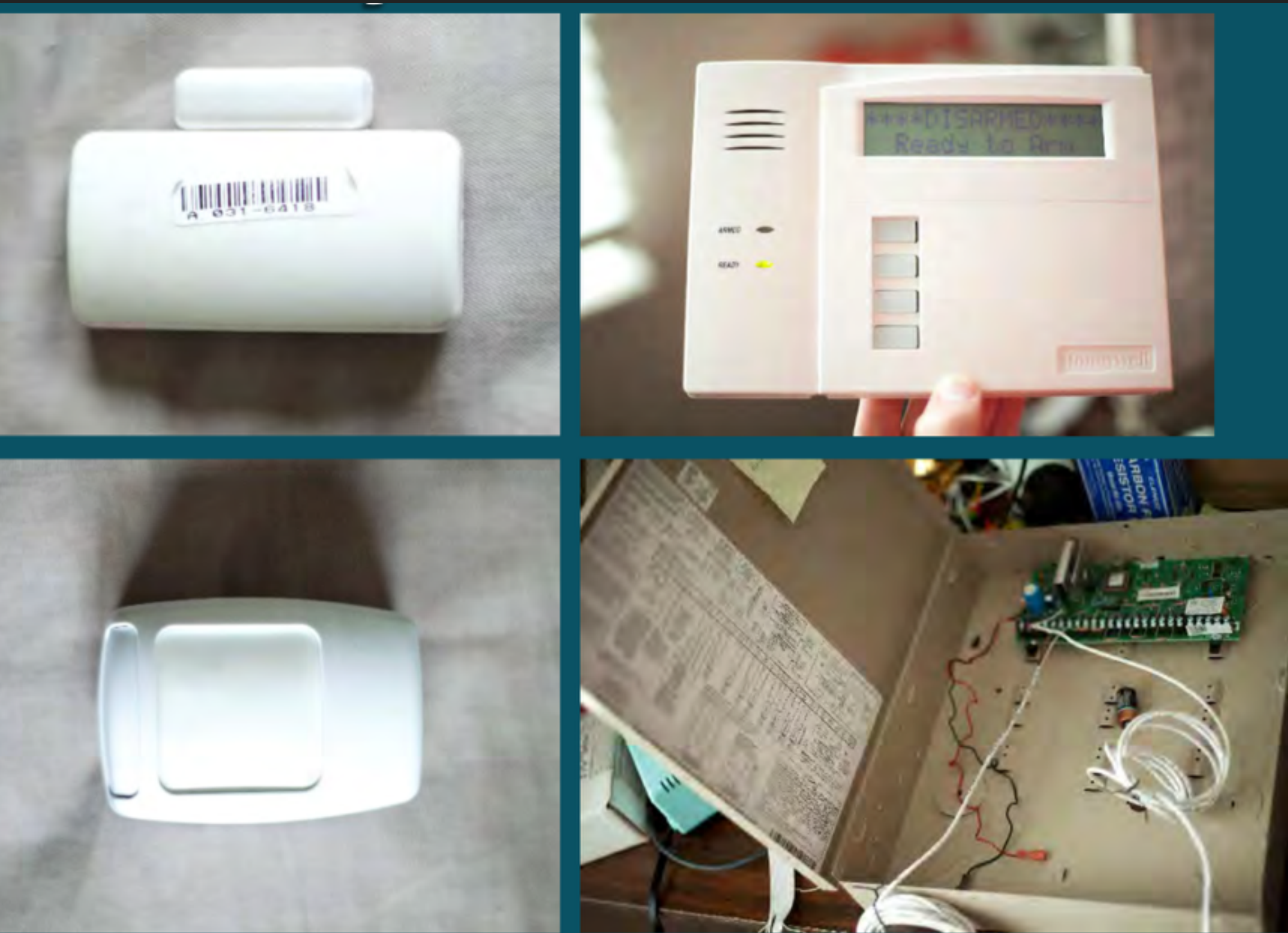
DENIAL OF SERVICE // NETWORK STATE DISRUPTION

JAMMING

JAMMING OVERVIEW

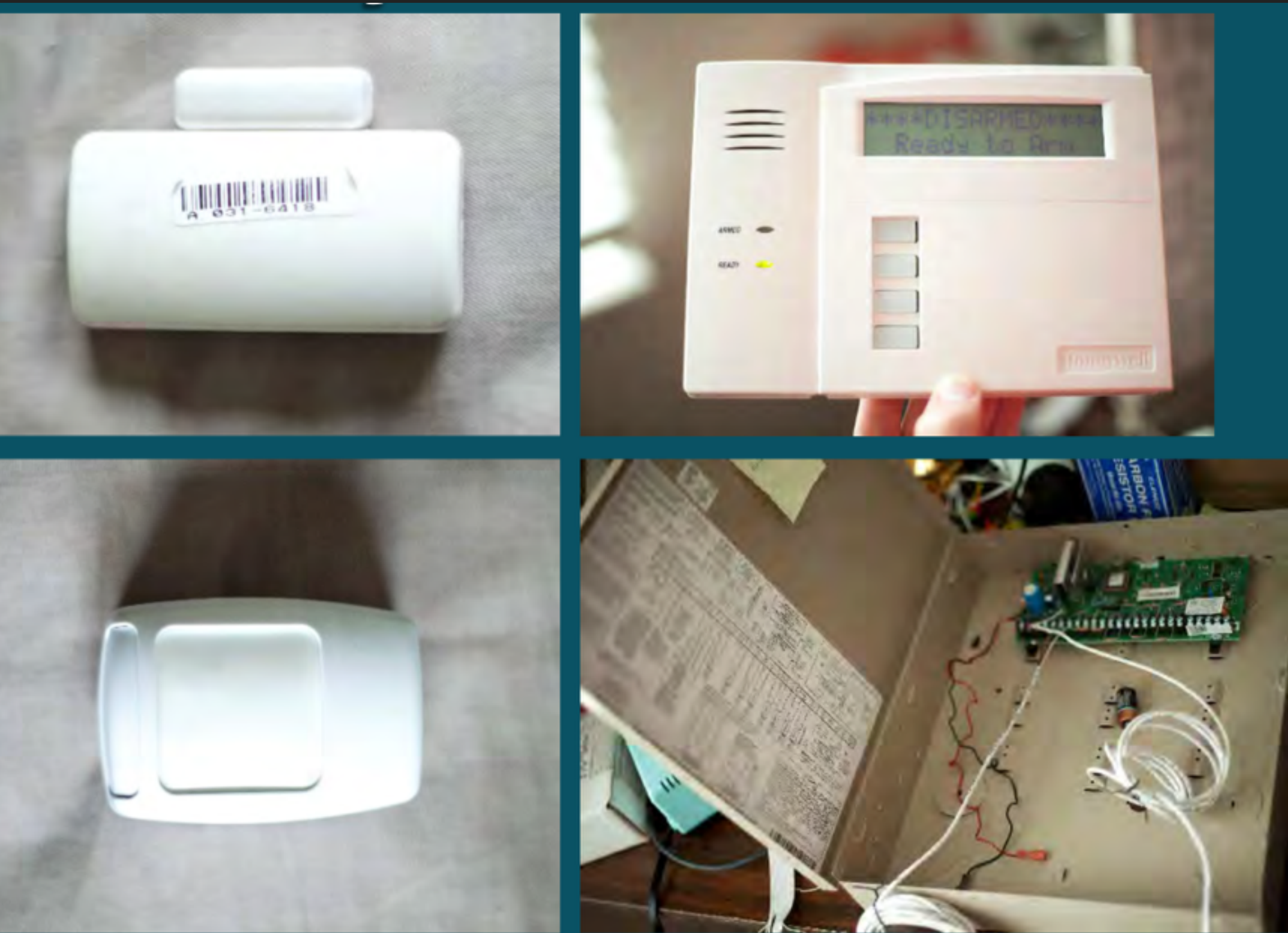
- ▶ Method
 - ▶ **Transmit** noise or conflicting traffic within target network's RF channel (same frequency)
- ▶ Impact
 - ▶ Blocks traffic on network
 - ▶ Network state disruption
- ▶ Wired Analogue
 - ▶ Denial of Service

JAMMING APPLIED



- ▶ Limitations
 - ▶ Jam detection mechanisms
 - ▶ Self-denial: difficult to simultaneously jam and monitor network traffic
- ▶ Example
 - ▶ Home security system jamming

JAMMING DEMO



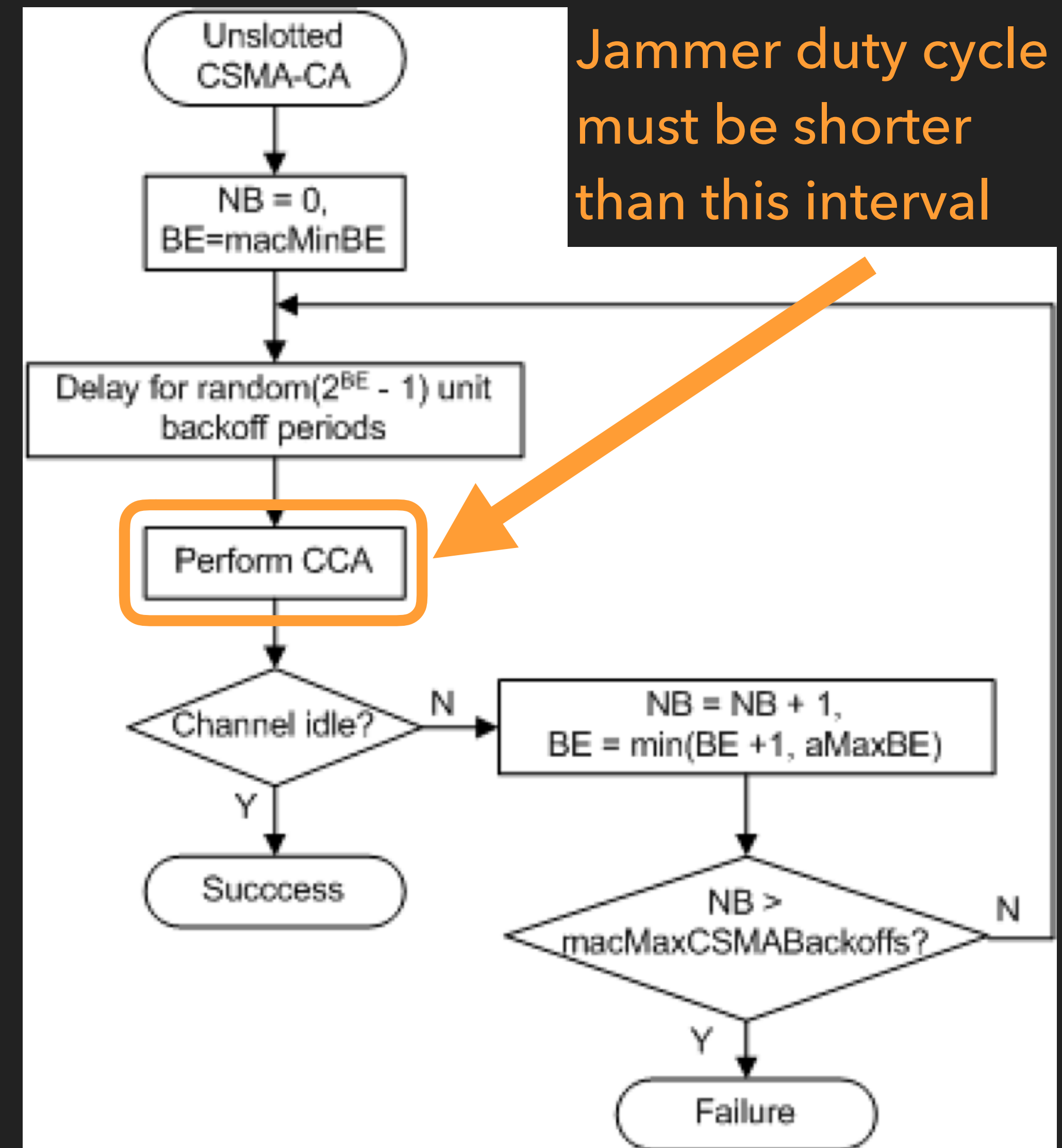
- ▶ Honeywell home security system
- ▶ 345 MHz on-off keying protocol
- ▶ Transmit **wideband noise** at 345 MHz
- ▶ Device **jam detection** mechanisms will detect after several seconds, so...

EVADING DETECTION 🤗

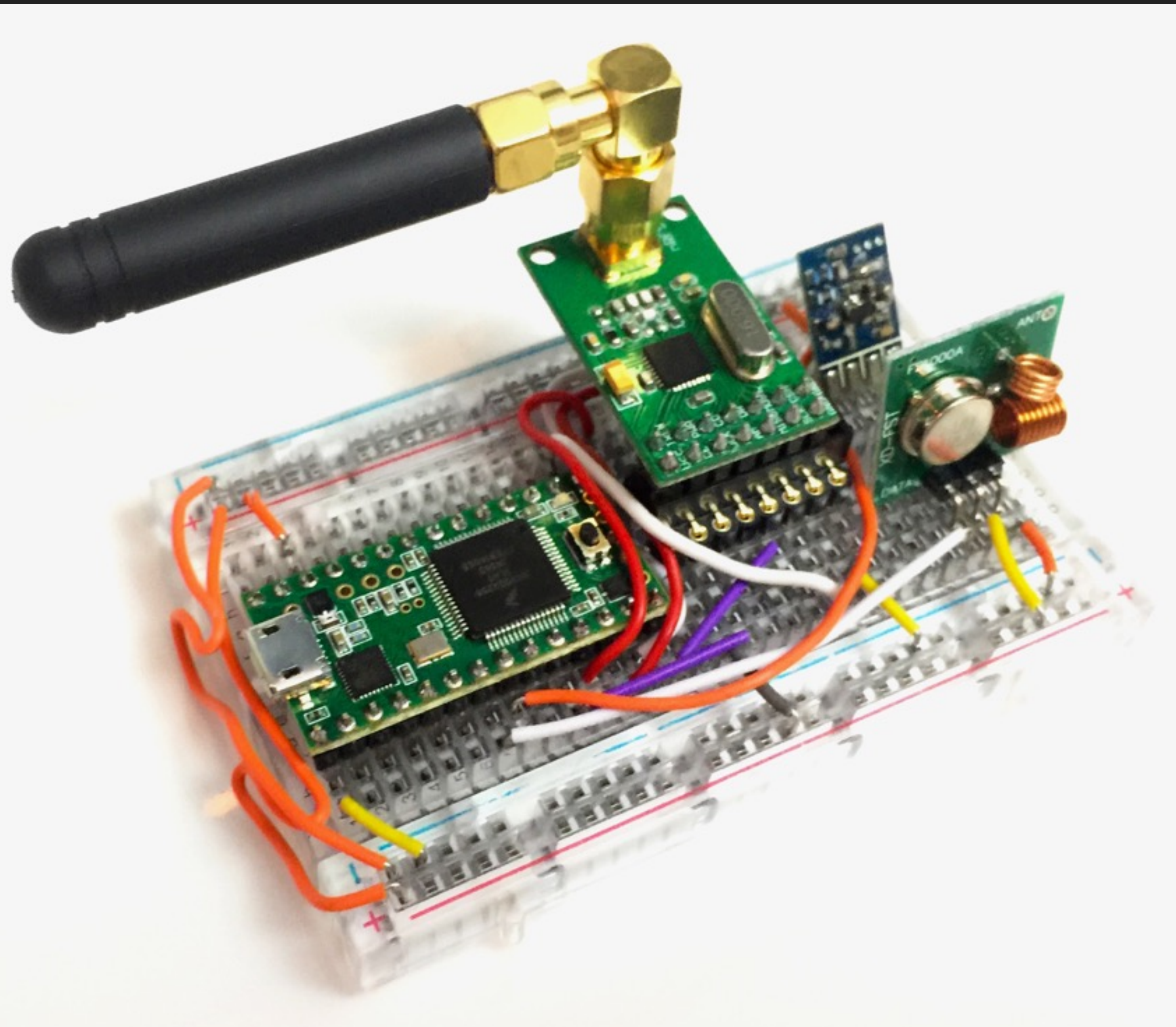
SMART JAMMING

DUTY CYCLED JAMMING

- ▶ Problem: Hardware radios implement “clear channel” detection features to avoid talking over other radios
- ▶ Sampling CCA is a zero-cost jam detector
- ▶ Solution: **pulse jammer on and off** at appropriate rate to evade jam detection functions
- ▶ Examples: Matt’s done this to defeat 802.15.4 jam detection, but doesn’t know of any public examples



REFLEXIVE JAMMING



- ▶ Problem: Continuously jamming makes offensive network monitoring hard
- ▶ Jamming denies both the attacker and the defender
- ▶ Solution: detect beginning of frame and **reflexively jam** to target either specific packets or trailing checksums
- ▶ Examples: Samy Kamkar's RollJam (left), reflexive jamming built into Killerbee//802.15.4 ApiMote

DENIAL OF SERVICE

MAC LAYER RESERVATION

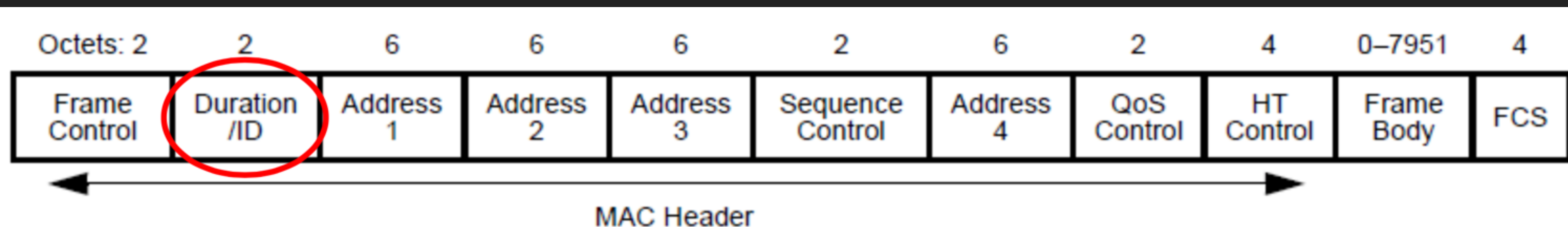
LINK LAYER CONGESTION OVERVIEW

▶ Method

- ▶ Abuse **channel reservation features** to reserve channel... forever
- ▶ See 802.11's virtual carrier sensing

▶ Impact

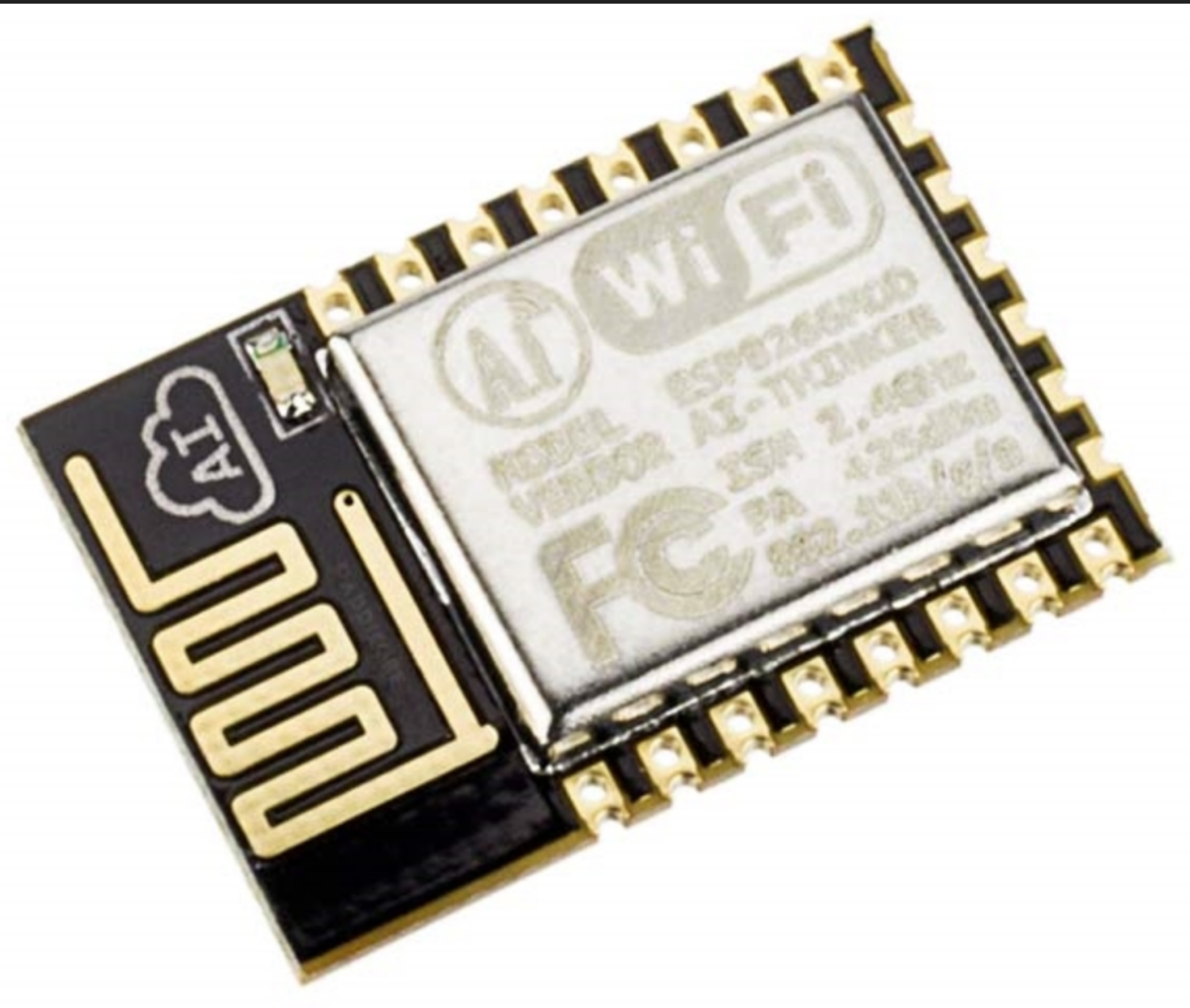
- ▶ Denies legitimate Wi-Fi devices access to the RF channel
- ▶ Wired Analogue
 - ▶ None (virtual carrier sense is unique to wireless protocols)



Left: 802.11 MAC header

LINK LAYER CONGESTION APPLIED

- ▶ Limitations
 - ▶ Network allocation vector constraints
 - ▶ [802.11]: Max 32ms effect per malicious packet
- ▶ Example
 - ▶ No recent examples of a virtual carrier sense abuse attack



\$5 attack platform (ESP8266)

LINK LAYER CONGESTION DEMO

```
(172.217.1.46): icmp_seq=7 ttl=50 time=95.9 ms
(172.217.1.46): icmp_seq=8 ttl=50 time=94.6 ms
(172.217.1.46): icmp_seq=9 ttl=50 time=87.6 ms
(172.217.1.46): icmp_seq=10 ttl=50 time=146 ms
(172.217.1.46): icmp_seq=11 ttl=50 time=18147 ms
(172.217.1.46): icmp_seq=12 ttl=50 time=17150 ms
(172.217.1.46): icmp_seq=13 ttl=50 time=16150 ms
(172.217.1.46): icmp_seq=14 ttl=50 time=15153 ms
(172.217.1.46): icmp_seq=15 ttl=50 time=14153 ms
(172.217.1.46): icmp_seq=16 ttl=50 time=13153 ms
(172.217.1.46): icmp_seq=17 ttl=50 time=12164 ms
(172.217.1.46): icmp_seq=18 ttl=50 time=11164 ms
(172.217.1.46): icmp_seq=19 ttl=50 time=10164 ms
(172.217.1.46): icmp_seq=20 ttl=50 time=9164 ms
(172.217.1.46): icmp_seq=21 ttl=50 time=8165 ms
(172.217.1.46): icmp_seq=22 ttl=50 time=7165 ms
(172.217.1.46): icmp_seq=23 ttl=50 time=64.8 ms
(172.217.1.46): icmp_seq=24 ttl=50 time=104 ms
(172.217.1.46): icmp_seq=25 ttl=50 time=101 ms
(172.217.1.46): icmp_seq=26 ttl=50 time=77.6 ms
(172.217.1.46): icmp_seq=27 ttl=50 time=79.7 ms
(172.217.1.46): icmp_seq=28 ttl=50 time=76.7 ms
(172.217.1.46): icmp_seq=29 ttl=50 time=78.3 ms
(172.217.1.46): icmp_seq=30 ttl=50 time=162 ms
```

- ▶ 802.11 virtual carrier sense abuse
 - ▶ One-line Scapy script
- ▶ 802.11 frame with:
 - ▶ Empty payload
 - ▶ **Maximum frame duration**
- ▶ Prevents other 802.11 devices from attempting to transmit on the channel

NETWORK MEMBERSHIP // IMPERSONATION // ROUTING

EVIL TWIN

EVIL TWIN OVERVIEW

- ▶ Method
 - ▶ **Impersonate** a network participant by assuming its address and configuration
 - ▶ (Optional) **actively deny** the real thing
- ▶ Impact
 - ▶ **Man in the Middle**: Intercept and actively tamper with network traffic
 - ▶ Tamper with routing, if eligible
- ▶ Wired Analogue
 - ▶ ARP spoofing/cache poisoning



(evil facial hair)

EVIL TWIN APPLIED

- ▶ Limitations
 - ▶ Defeated by **authentication** (cryptography/trust)
 - ▶ Attacker may need to deny legitimate twin
- ▶ Example
 - ▶ WiFi Pineapple
 - ▶ IMSI Catchers



EVIL TWIN DEMO

- ▶ 2G **GSM/GPRS** IMSI catcher/man in the middle
- ▶ OpenBTS/Range Networks GSM BTS
 - ▶ Creates an arbitrarily-configured network
 - ▶ Phones downgrade and connect if they believe it's the best option



Faraday cage

China Mobile

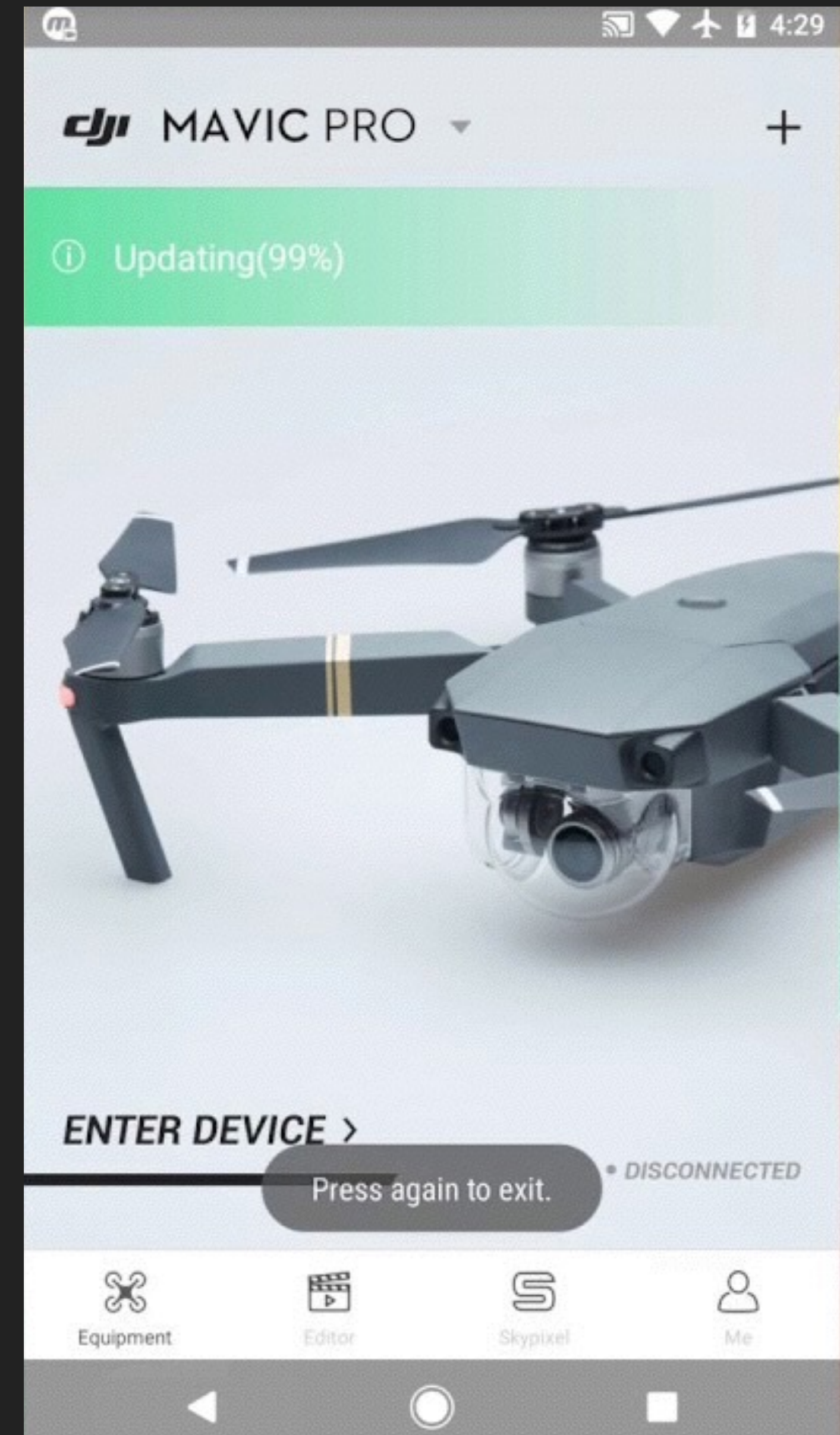
PERSISTENCE // ADDED VALUE

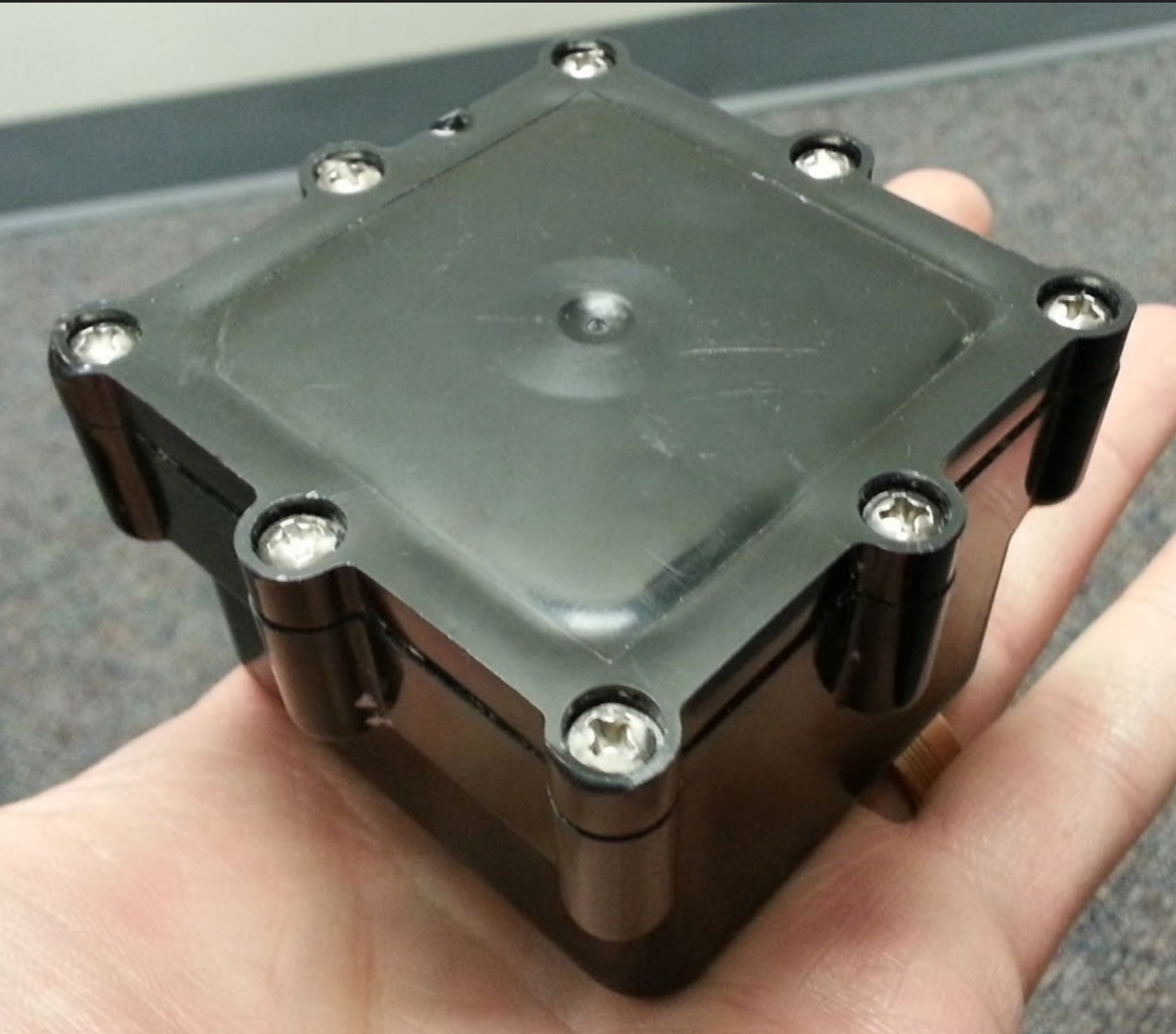
FIRMWARE UPDATES



FIRMWARE ATTACK OVERVIEW

- ▶ Method
 - ▶ Modify firmware image to your liking
 - ▶ Deliver it via **OTA firmware update mechanism**
- ▶ Impact
 - ▶ “Added value”: modifying device behavior
 - ▶ Persistence
 - ▶ Denial of Service: see BrickerBot
 - ▶ Self-propagation
- ▶ Wired Analogue
 - ▶ Traditional endpoint malware, worms, etc.





FIRMWARE ATTACKS APPLIED

- ▶ Limitations
 - ▶ Defeated by code signing, network encryption, etc.
- ▶ Example
 - ▶ Cesar Cerrudo's hypothetical traffic light sensor worm (also ZigBee)
 - ▶ Phillips Hue ZigBee Light Link worm

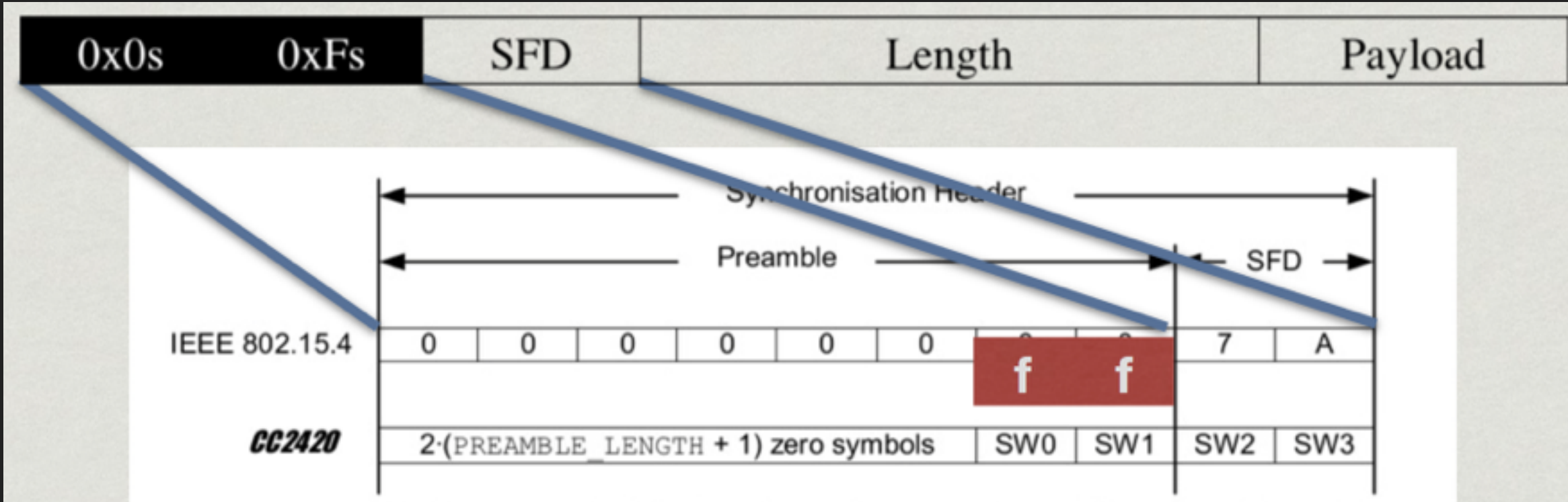
FIRMWARE ATTACK DEMO



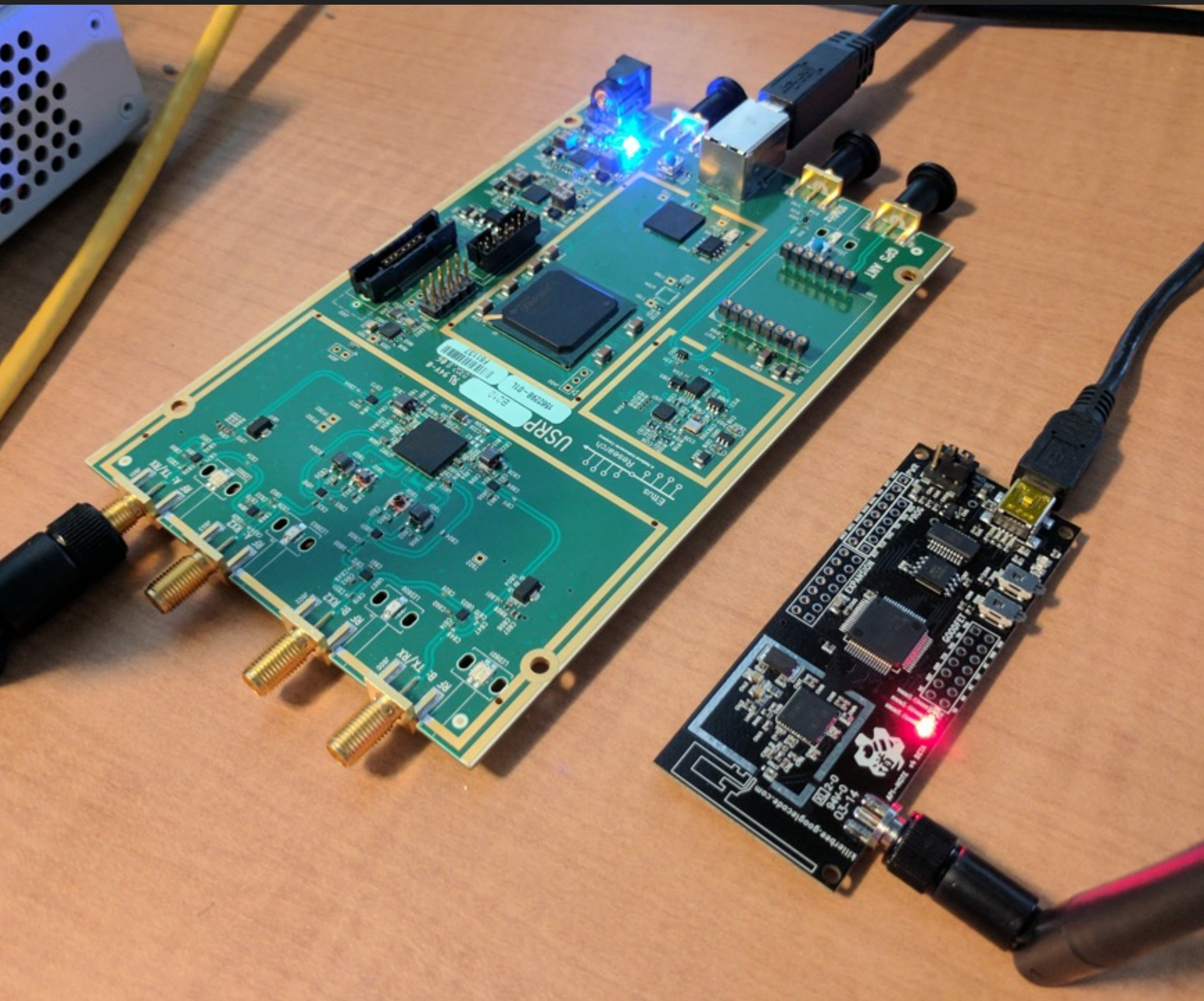
- ▶ Phillips Hue Worm (ZigBee Light Link OTA)
 - ▶ Eyal Ronen, Colin O'Flynn, Adi Shamir ("S" from RSA) and Achi-Or Weingarten
- ▶ No demo for this one
 - ▶ Great paper and video at <http://iotworm.eyalro.net/>
 - ▶ Best part is firmware signing key recovery through a side channel

PHY SELECTIVE TARGETING OVERVIEW

- ▶ Method
 - ▶ Chipsets implement PHY standards differently – various degrees of error tolerance
 - ▶ Send **standards-noncompliant transmissions** that exploit corner cases in specific PHY state machines
- ▶ Impact
 - ▶ Targeted receiver evasion (IDS evasion)
 - ▶ Device fingerprinting
- ▶ Wired Analogue
 - ▶ Same (demonstrated on 802.3 chipsets)
 - ▶ **Far more practical in RF domain**

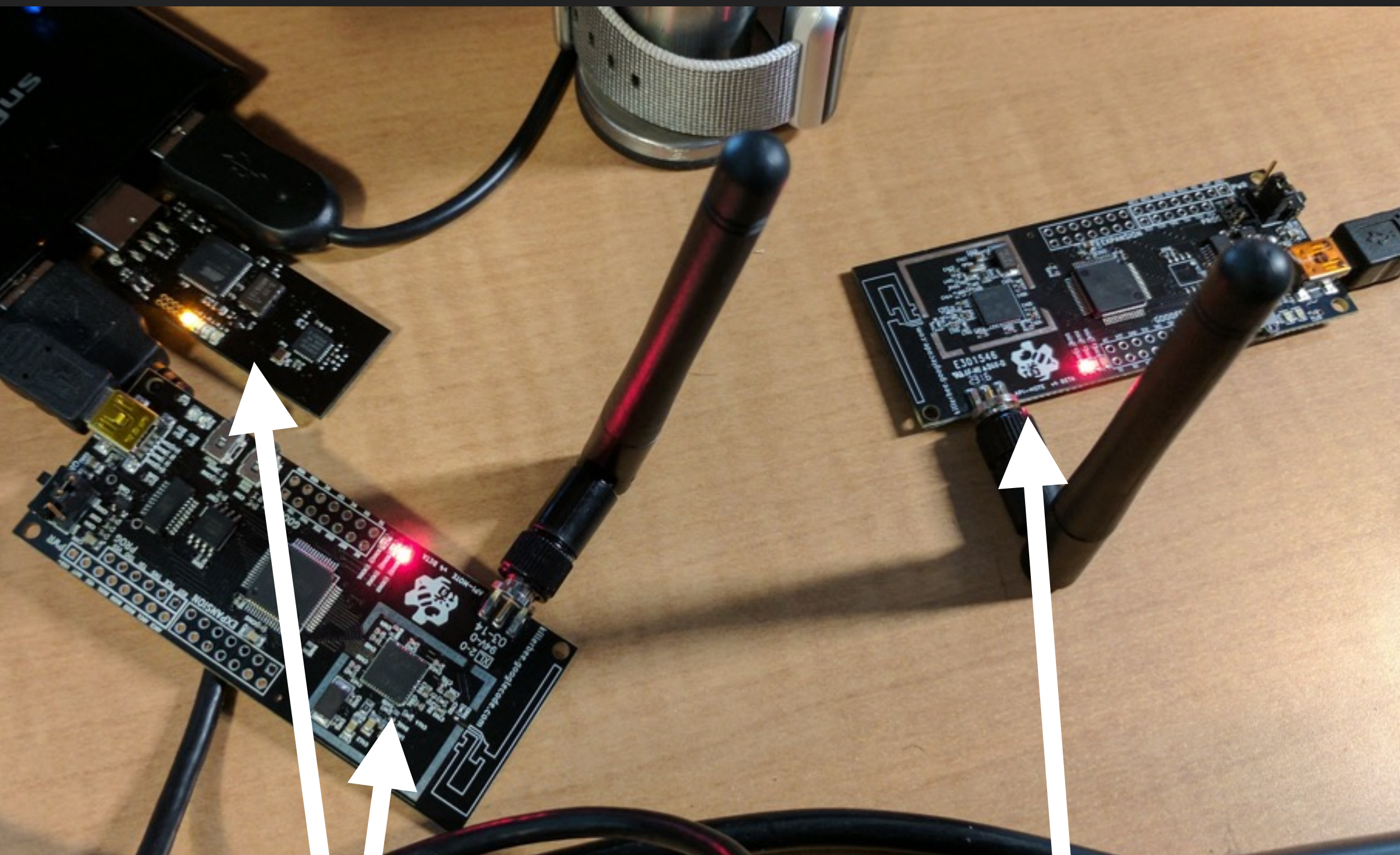


Preamble	RZUSB Observed	ApiMote Observed
00 00 00 00	672	1000
00 00 00 ff	991	0
00 00 ff ff	990	0
00 ff ff ff	855	1
ff ff ff ff	4	0



PHY SELECTIVE TARGETING APPLIED

- ▶ Limitations
 - ▶ Network participants must be on different chipsets
 - ▶ Not all chipsets are vulnerable
- ▶ Example
 - ▶ 802.15.4 receiver evasion

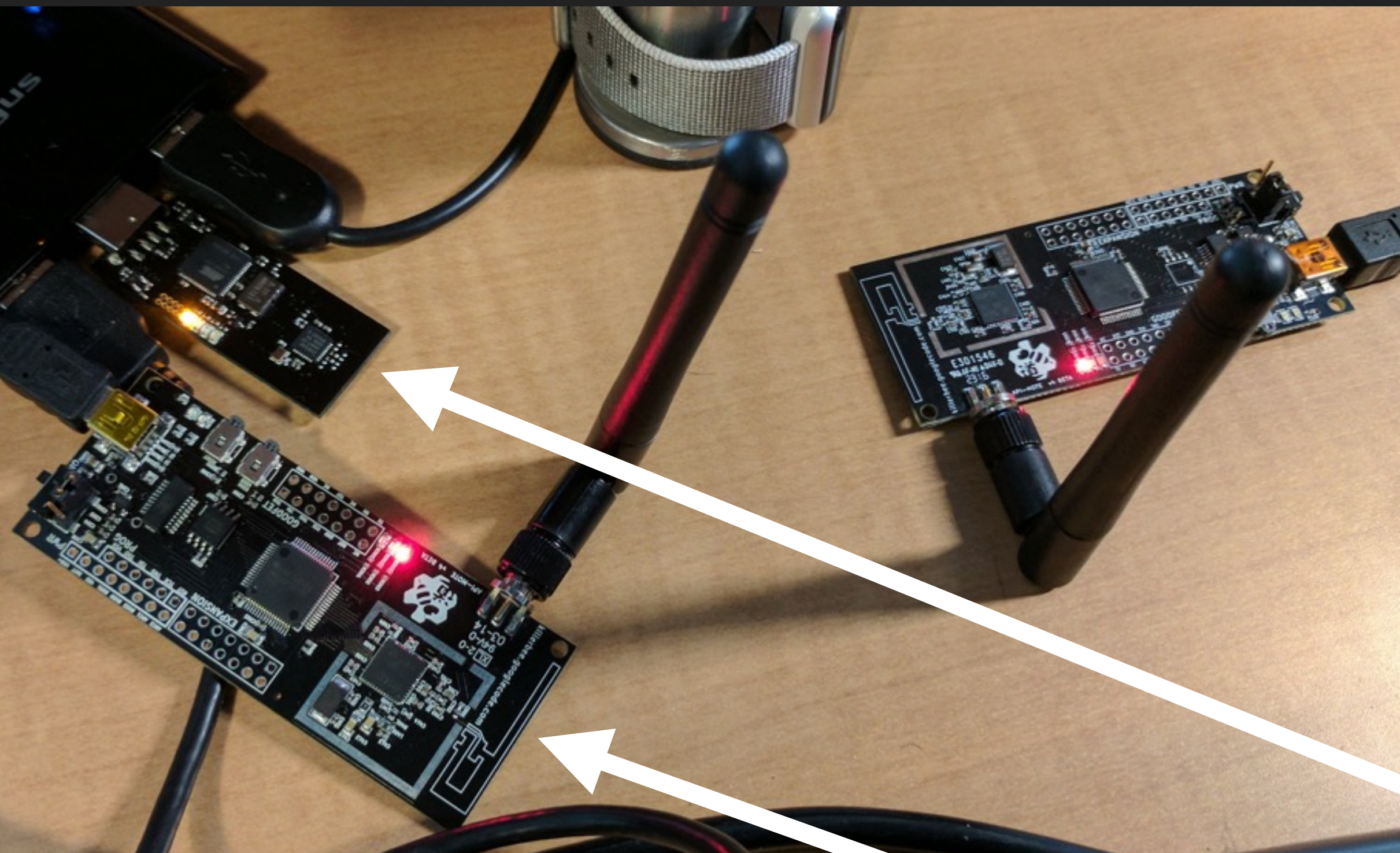


Receivers

Transmitter

PHY SELECTIVE TARGETING DEMO

- ▶ Selectively evasive 802.15.4 packets
- ▶ Transmitter: ApiMote w/ CC2420
- ▶ Receivers: ApiMote w/ CC2420
RZUSB stick w/ AT86RF230
- ▶ Both receivers receive everything, until they don't... 😊



PHY SELECTIVE TARGETING DEMO

- ▶ 802.15.4 preamble and SFD:
 - ▶ 0x00000000A7: 4 0x00s + 1 0xA7
- ▶ What if we screw with this?
 - ▶ 0x00000000FFA7: extra symbols in preamble
 - ▶ 0x00000000A7: short preamble

CHARACTERIZING WIRELESS ATTACK METHODS

CONCLUSIONS

WIRELESS ATTACK METHODS SUMMARY

	Analogue	Complexity	Ease of Mitigation
Sniffing	Unique!	Easy	Hard
Wardriving	Port Scanning	Easy	Hard
Replay	[same]	Easy	Moderate
Jamming	Denial of Service	Easy	Hard
Link Layer Congestion	Unique!	Moderate	Moderate
Evil Twin	ARP Spoofing	Hard	Moderate
Firmware Attack	Malware	Hard	Moderate
PHY Abuse	[same]	Hard	Hard

NON-EXHAUSTIVE LIST

[OBVIOUSLY]

AS ATTACKERS...

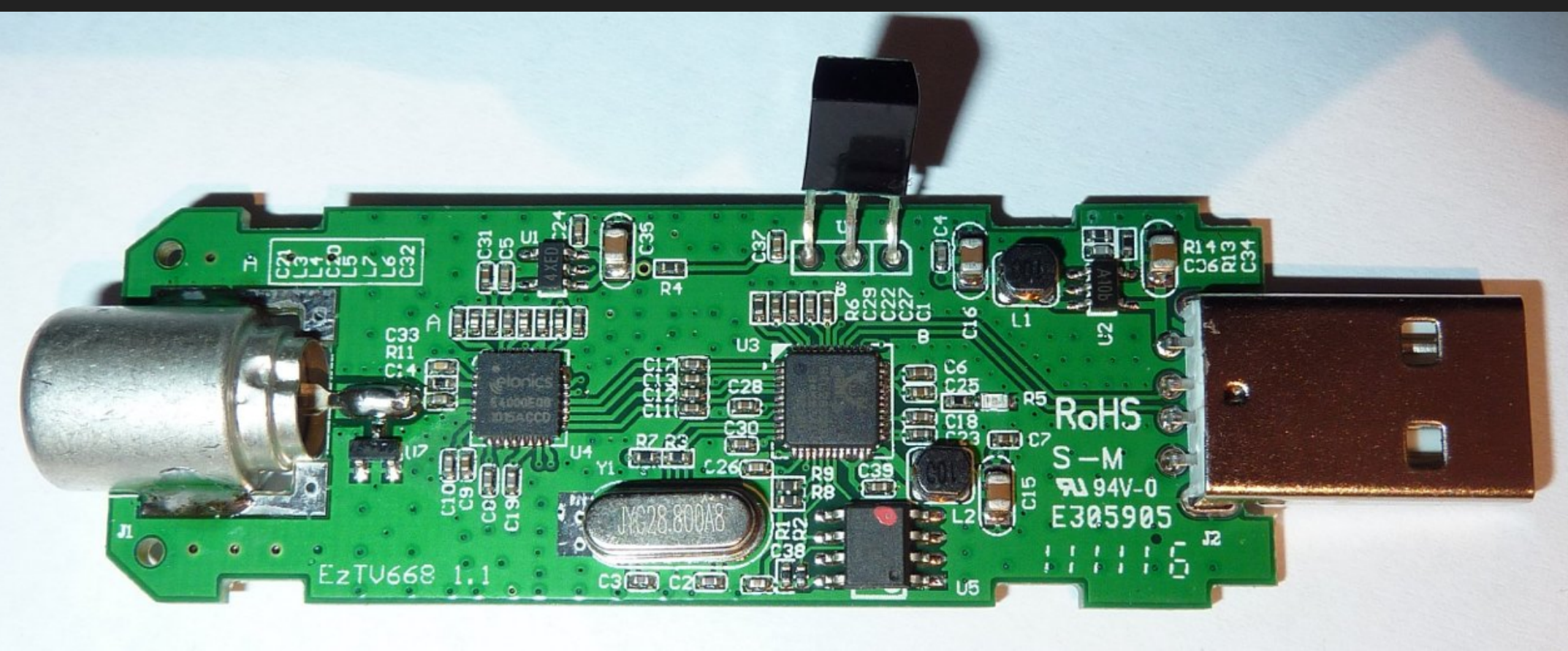
- ▶ Look for **low-hanging fruit** first
 - ▶ Unencrypted comms, replay attacks, cleartext key exchanges, etc.
 - ▶ Complexity goes up in a hurry
- ▶ Lean on your existing wired/IP network skill set
 - ▶ Analogues exist!

AS ATTACKERS... CONTINUED

- ▶ Leverage Open Source Intelligence (OSINT):
 - ▶ FCC regulatory filings
 - ▶ Data sheets
- ▶ It will make your life easy
- ▶ Marc gave an entire talk on this at HITB2016AMS

AS DEVELOPERS...

- ▶ This is the **Golden Age of RF Hacking**
 - ▶ Software Defined Radio has been commodity for >5 years
 - ▶ **Every RF PHY** is in scope now



**TIME TO OWN
YOUR AIRWAVES**

ADDITIONAL RADIO RESOURCES

- ▶ “So You Want to Hack Radios” series (all about RF Physical Layers)
 - ▶ Shmoocon: <https://www.youtube.com/watch?v=L3udJnRe4vc>
 - ▶ Troopers: <https://www.youtube.com/watch?v=OFRwqpH9zAQ>
 - ▶ HITB2017AMS Commsec: <https://www.youtube.com/watch?v=QeoGQwT0Z1Y>
- ▶ Matt’s LoRa research
 - ▶ 33c3: https://media.ccc.de/v/33c3-7945-decoding_the_lora_phy
- ▶ Marc’s OSINT techniques
 - ▶ HITB2016AMS Commsec: <https://www.youtube.com/watch?v=JUAiav674D8>
- ▶ Dallas siren attack research
 - ▶ White paper: <https://www.bastille.net/blogs/2017/4/17/dallas-siren-attack>

ACKNOWLEDGEMENTS

- ▶ Balint and Logan from **Bastille**'s Threat Research Team
- ▶ **Bastille** at large
- ▶ DEF CON 25 team!
- ▶ See you at DEF CON 50!





THANKS

github.com/BastilleResearch

marc@**Bastille**.net
@marcnewlin

matt@**Bastille**.net
@embeddedsec



QUESTIONS?

github.com/BastilleResearch

marc@**Bastille**.net
@marcnewlin

matt@**Bastille**.net
@embeddedsec