Florent Fayollas

Antoine Vacher

TLS-SEC

DRONE PREDATOR

ONE DRONE TO RULE THEM ALL

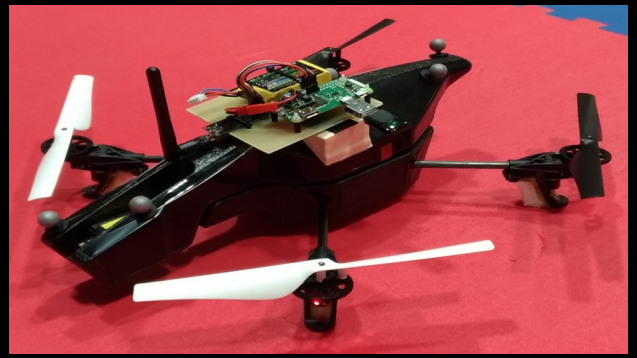
Project overall goal

- **Take control of multiple drones**
  - Some existing military solutions
  - No civil solution

Project overall goal

- **Take control of multiple drones**
  - Some existing military solutions
  - No civil solution

- **Our goals:**
  - Reduced cost
  - No drone falling implied
  - High number of compatible drones
  - Usable on public events, by civilians



Syma – Introduction

The goal of the attack is to highjack a Syma X5C-1 Drone in flight

<u>Documentation and markings tell us:</u>

- Not WIFI
- Frequency band 2.4 GHz
- 4 Channels

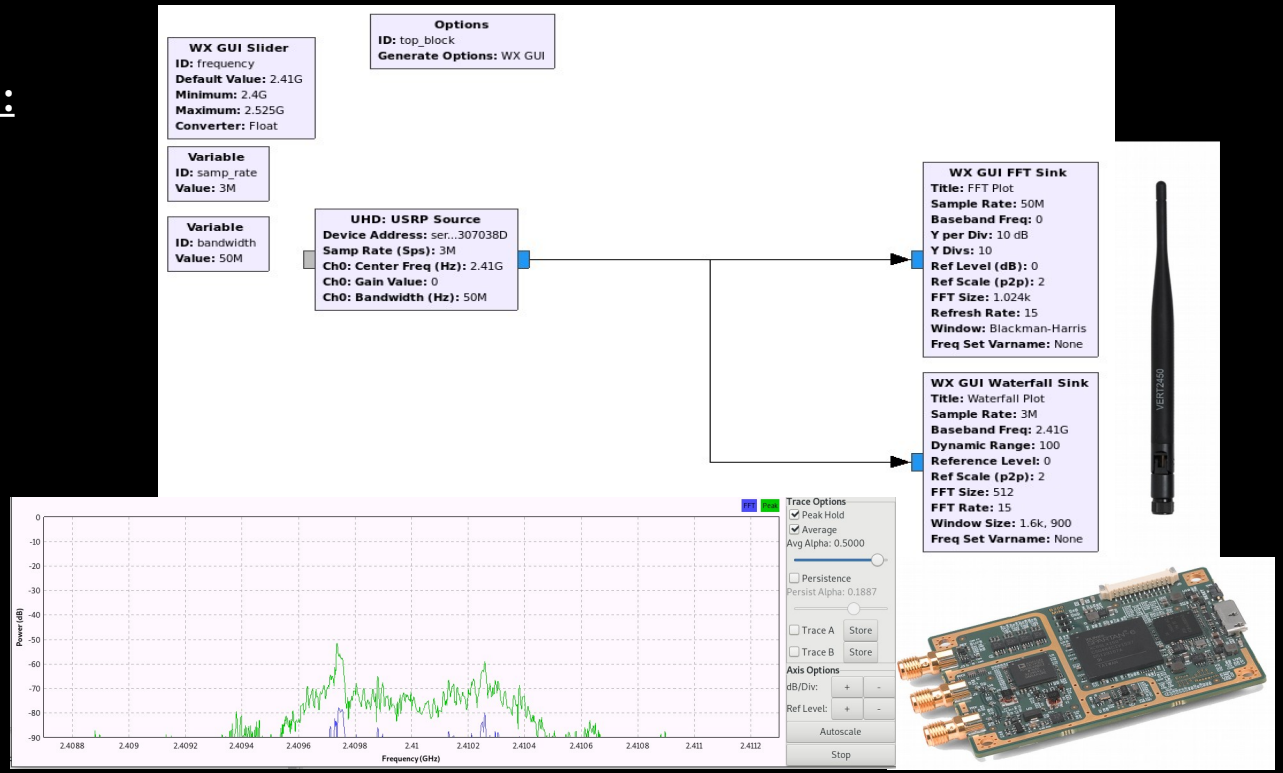
Syma – Step 1 – Finding Radio Channels

<u>Using an USRP B200 SDR with GnuRadio :</u>

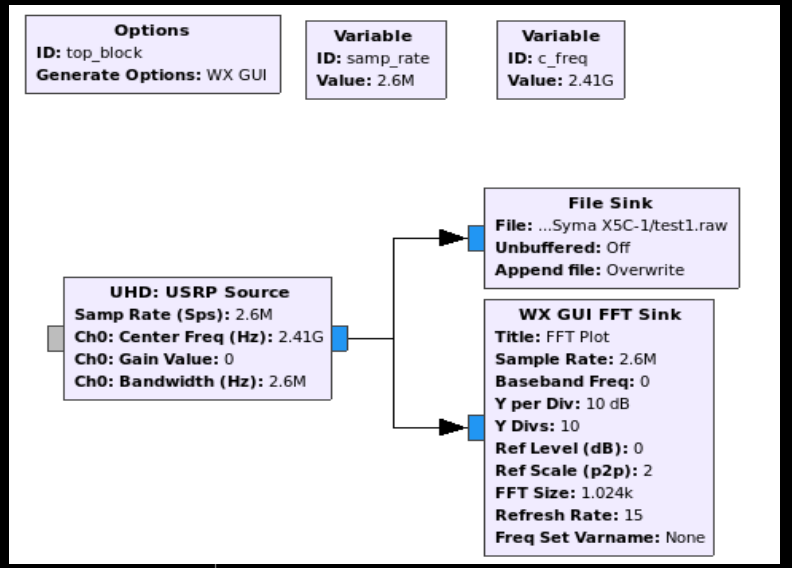
- USRP Source
- FFT and Waterfall Sinks
- Browse all channels by 1MHz steps

<u>→ Tx power received on channels :</u>

- 10 : 2.410 GHz
- 31 : 2.431 GHz
- 42 : 2.442 GHz
- 66 : 2.466 GHz



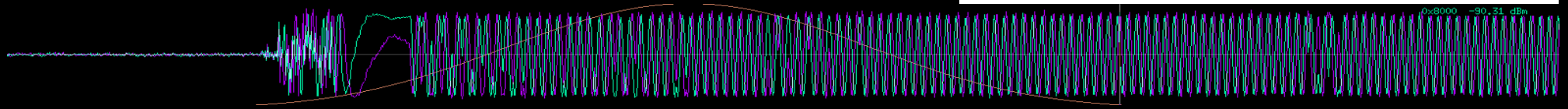
Syma – Step 2 – Finding Radio Modulation



<u>Using an USRP B200 SDR with GnuRadio :</u>

- USRP Source
- File Sink

<u>Opening file with Baudline :</u>

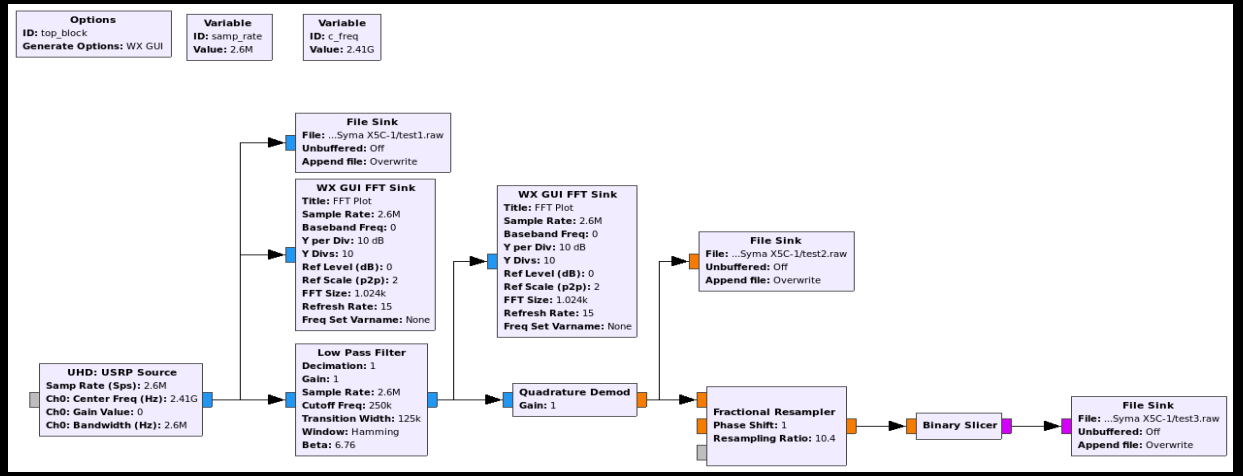


Constant amplitude → Not ASK. Could be (G)FSK or PSK

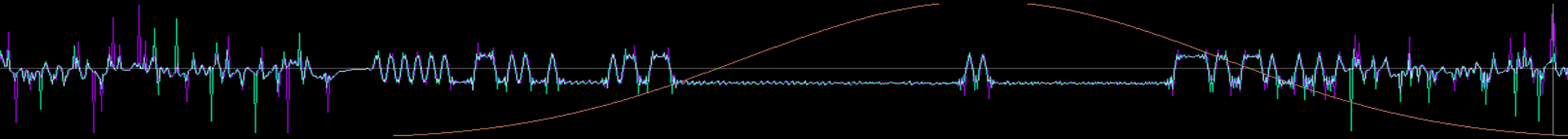
Syma – Step 2 – Finding Radio Modulation

Trying GFSK as it is widely used for small electronics

- USRP Source
- Low Pass Filter
- Quadrature Demod
- File Sink
- Save bit stream

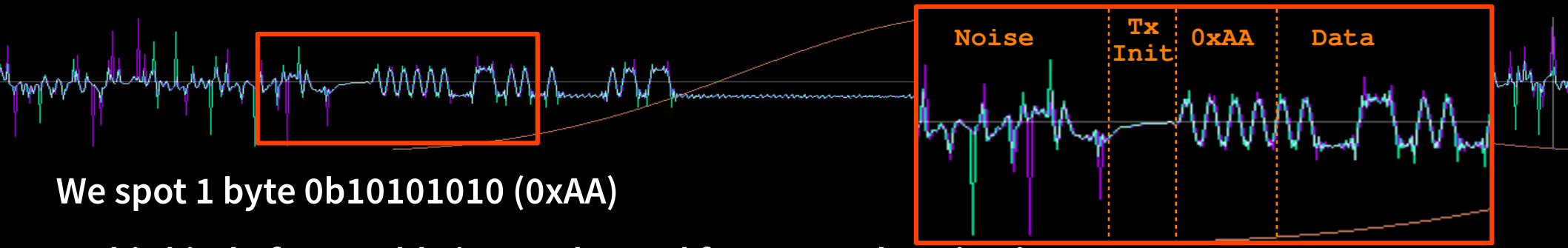


Opening file with Baudline :



Signal looked clean. Demodulation seemed to be correct : GFSK 250 Kbit/s

Syma – Step 3 – Understanding Data Link Layer



We spot 1 byte 0b10101010 (0xAA)

→ This kind of preamble is mostly used for Rx synchronisation

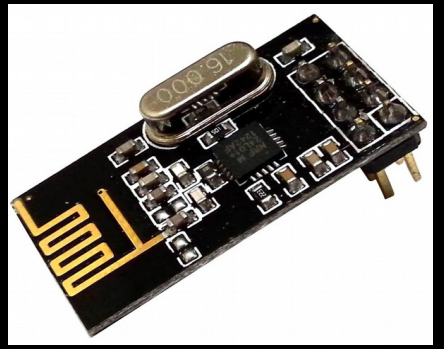
Message is 18 bytes (including preamble)

```
000007f0  00 00 00 01 01 01 00 00  01 00 01 00 01 00 01 00  |................|
00000800  01 01 01 01 00 00 00 00  01 01 00 00 00 00 01 00  |................|
00000810  00 00 00 01 00 00 01 01  01 00 00 01 01 00 00 01  |................|
00000820  01 00 00 01 00 01 00 00  00 00 00 01 00 01 01 00  |................|
00000830  01 01 00 00 01 01 01 01  00 00 00 00 01 00 01 00  |................|
00000840  00 00 00 00 01 01 00 00  00 01 01 01 00 01 01 01  |................|
00000850  01 01 00 00 00 01 00 01  01 00 01 00 01 01 00 00  |................|
00000860  00 00 01 01 01 01 00 00  00 01 00 00 00 01 00 00  |................|
00000870  01 01 01 01 01 01 00 00  01 00 01 01 00 01 01 01  |................|
00000880  01 01 01 00 01 00 00 01  00 00 01 01 01 01 01 00  |................|
00000890  01 01 01 01 01 01 00 01  00 01 00 00 01 00 00 00  |................|
```

hexdump -C test3.raw (recorded bit stream)

Syma – Step 3 – Understanding Data Link Layer

Introducing the nRF24l01+ module :



- Transceiver GFSK 2.4GHz on SPI bus
- Very popular module (on drones, RF mouses, DIY projects, …)
- Come with 2 predefined packet based datalink layers (Basic or Enhanced Shockburst)
- Using 0xAA or 0x55 as preamble
- Using 3 to 5 bytes address
- Using variable payload length (up to 32 bytes in basic mode)
- Can use CRC on 1 or 2 bytes (optional)

Preamble (1 byte)	Address (3-5 bytes)	Payload (0-32 bytes)	CRC (0-2 bytes)

Drone preamble is 0xAA which may mean that it is using a nR24l01+

→ Need to see if the rest of the frame matches

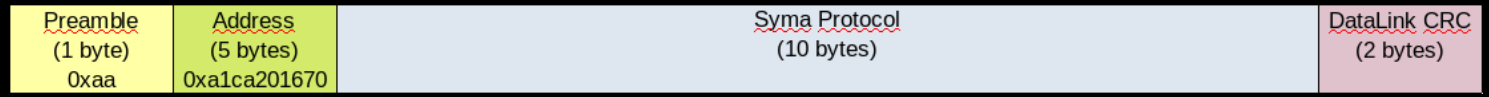
Syma – Step 3 – Understanding Data Link Layer

With the help of a python script, we guessed the field sizes from the bit stream recorded with GnuRadio

https://github.com/chopengauer/nrf_analyze/blob/master/nrf24_analyzer.py

```
[root@tigrou Syma X5C-1]# ../../../nrf_analyze/nrf24_analyzer.py
Start
0Position 483   offset 483      Address a1ca201670      Data 00000000002000000176       CRC c711        Len 10 preamb = aa
0Position 9226  offset 8743     Address a1ca201670      Data 00000000002000000003       CRC da12        Len 10 preamb = aa
0Position 16875 offset 7649     Address a1ca201670      Data 00000000002000000176       CRC c711        Len 10 preamb = aa
0Position 17968 offset 1093     Address a1ca201670      Data 00000000002000000176       CRC c711        Len 10 preamb = aa
0Position 25617 offset 7649     Address a1ca201670      Data 00000000002000000176       CRC c711        Len 10 preamb = aa
0Position 26710 offset 1093     Address a1ca201670      Data 00000000002000000003       CRC da12        Len 10 preamb = aa
0Position 34359 offset 7649     Address a1ca201670      Data 00000000002000000176       CRC c711        Len 10 preamb = aa
0Position 35452 offset 1093     Address a1ca201670      Data 00000000002000000176       CRC c711        Len 10 preamb = aa
0Position 44195 offset 8743     Address a1ca201670      Data 00000000002000000003       CRC da12        Len 10 preamb = aa
0Position 51845 offset 7650     Address a1ca201670      Data 00000000002000000176       CRC c711        Len 10 preamb = aa
0Position 52937 offset 1092     Address a1ca201670      Data 00000000002000000176       CRC c711        Len 10 preamb = aa
0Position 60588 offset 7651     Address a1ca201670      Data 00000000002000000176       CRC c711        Len 10 preamb = aa
0Position 61681 offset 1093     Address a1ca201670      Data 00000000002000000003       CRC da12        Len 10 preamb = aa
0Position 69331 offset 7650     Address a1ca201670      Data 00000000002000000176       CRC c711        Len 10 preamb = aa
0Position 79166 offset 9835     Address a1ca201670      Data 00000000002000000003       CRC da12        Len 10 preamb = aa
0Position 87909 offset 8743     Address a1ca201670      Data 00000000002000000176       CRC c711        Len 10 preamb = aa
```

In a nutshell :



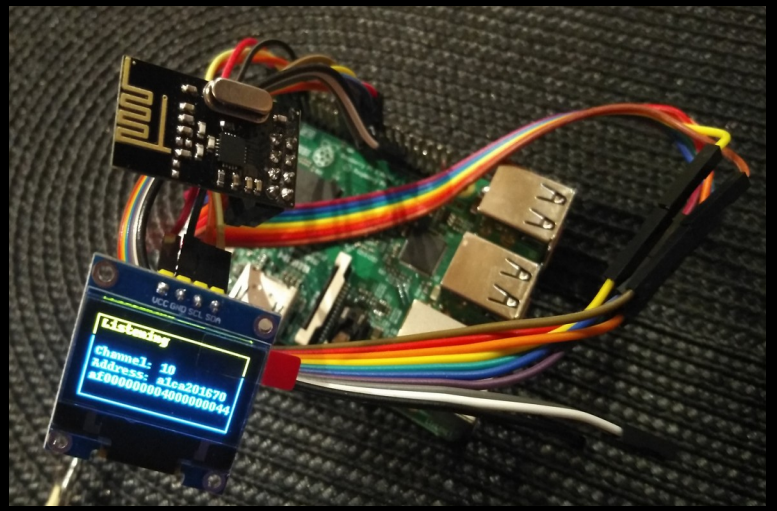
Note : If we would not have had a SDR, we could have used a pseudo-promiscuous mode of the nRF24l01+ by tuning the module out of its specification. This is quite tricky. Refer to the full report for more information.

Syma – Step 4 – Understanding Syma Protocol

We built a live analyser

- With Raspberry PI and nRF24l01+
- OLED Screen added for THCon

→ By moving the RC controls, we could observe the payload.



Finally, Syma Protocol is quite simple...*

*Except Byte 10 : A pseudo CRC built as a XOR of bytes 1 to 9 and added to 0x55

Preamble (1 byte) 0xaa	Address (5 bytes) 0xa1ca201670	Altitude (1 byte)	Pitch (1 byte)	Roll (1 byte)	Yaw (1 byte)	Other Stuff (5 bytes)	Syma « CRC » (1 byte)	DataLink CRC (2 bytes)

… Now we were ready to implement !

… so we added a USB Gamepad, tuned the code …

… and finally attacked !

Syma – Attack Characteristics

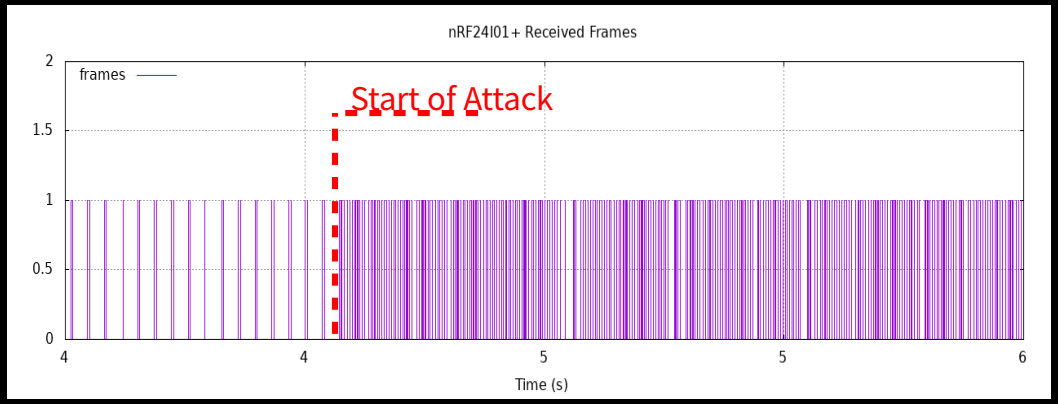
- Attack is only working when the drone is already paired with a controller

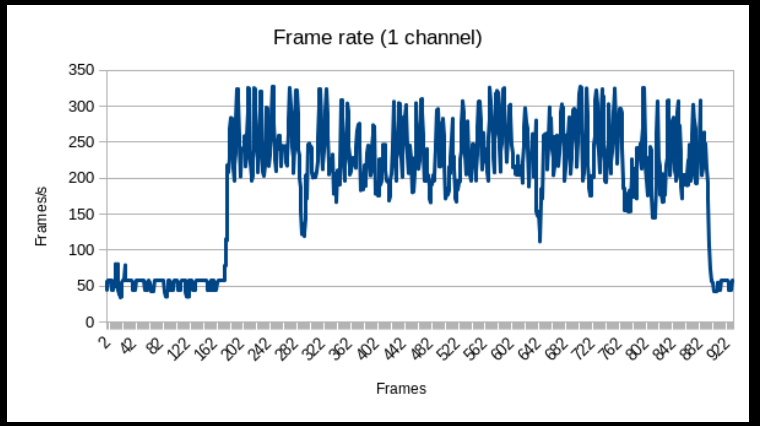
  → Some binding protocol exists (out of scope)

- Drone receives orders both from legit pilot and from predator

  → Attack succeeds because the predator emits a lot more frames than the official remote

- We created a logger with RPI and nRF24l01+ to visualize the difference





Syma – Conclusion

- **Syma Protocol quite simple (1 byte per command, no encryption, ...)**
- **Highjacking a Syma X5C-1 requires very few ressources :**
  - 1 Raspberry Pi Zero W (~10€)
  - 1 nRF24l01+ (~0.5€)
  - A few cables
  - 1 USB Gamepad (~5€)
- **Attack is :**
  - Hard to detect by the pilot
  - Easy to detect by an engineer (SDR, ...)
  - Almost impossible to stop without jamming and thus loosing the drone

The Parrot AR.Drone 2.0

- **Developed by a French company**
- **Released in 2012**
- **Features:**
  - OS: Linux 2.6.32
  - CPU: ARM Cortex A8
  - Autonomy: 12”
  - Optional GPS
- **Controlled by WiFi**
  - Open WiFi network
  - iOS/Android app

The Parrot AR.Drone 2.0

- **Developed by a French company**
- **Released in 2012**
- **Features:**
  - OS: Linux 2.6.32
  - CPU: ARM Cortex A8
  - Autonomy: 12”
  - Optional GPS
- **Controlled by WiFi**
  - Open WiFi network
  - iOS/Android app

Vulnerable!

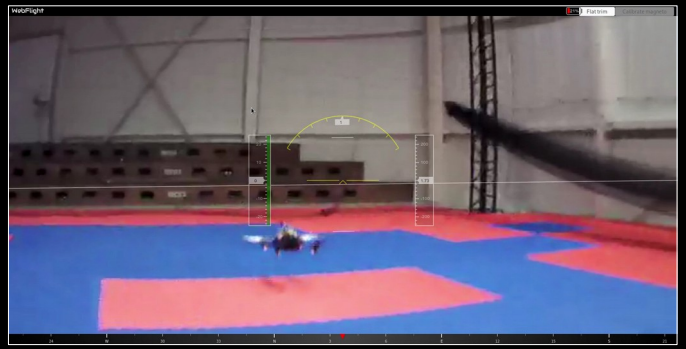
Parrot – Attack principle

- **Based on a existing attack (Samy Kamkar)**
  - Discover drones' WiFi network and clients

    Scapy `sniff` instead of `airodump-ng`
  - Disconnect client from network

    `aireplay-ng`
  - Connect to network and control the drone

    `ardrone-webflight`



Real connection

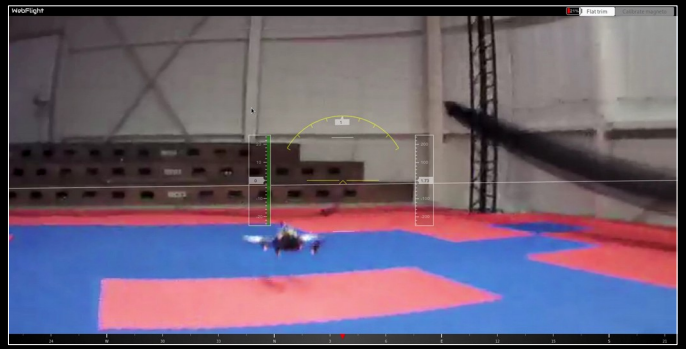
Parrot – Attack principle

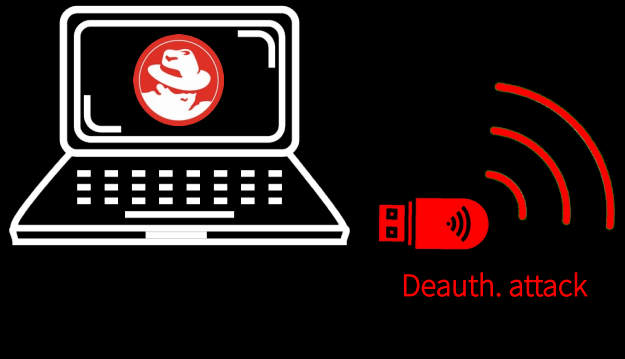
- **Based on a existing attack (Samy Kamkar)**
  - Discover drones' WiFi network and clients

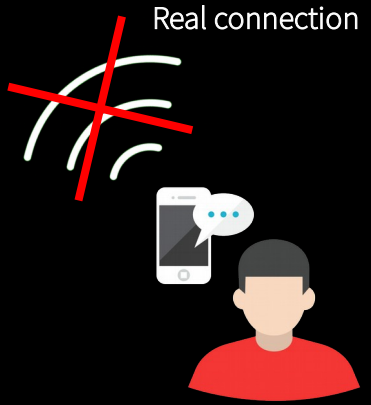
    Scapy `sniff` instead of `airodump-ng`
  - Disconnect client from network

    `aireplay-ng`
  - Connect to network and control the drone

    `ardrone-webflight`







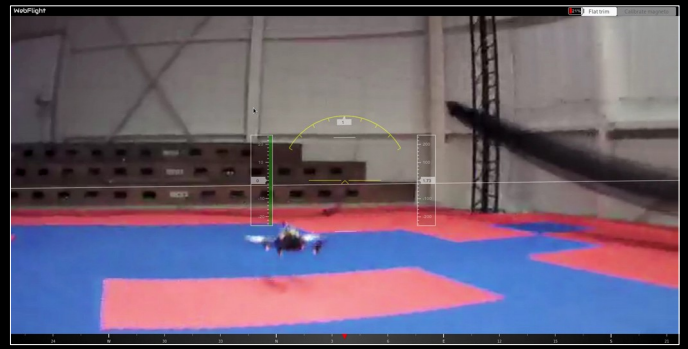
Parrot – Attack principle

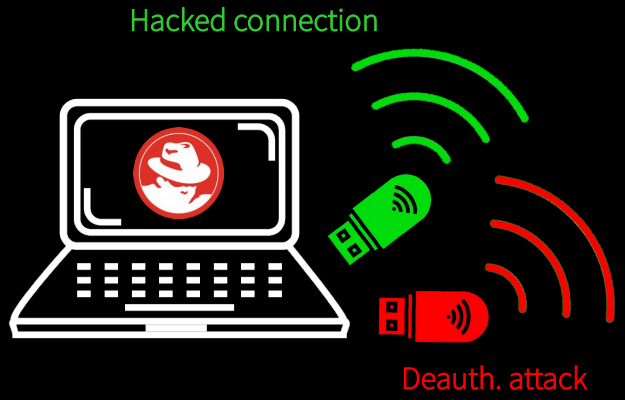
- **Based on a existing attack (Samy Kamkar)**
  - Discover drones' WiFi network and clients

    Scapy `sniff` instead of `airodump-ng`
  - Disconnect client from network

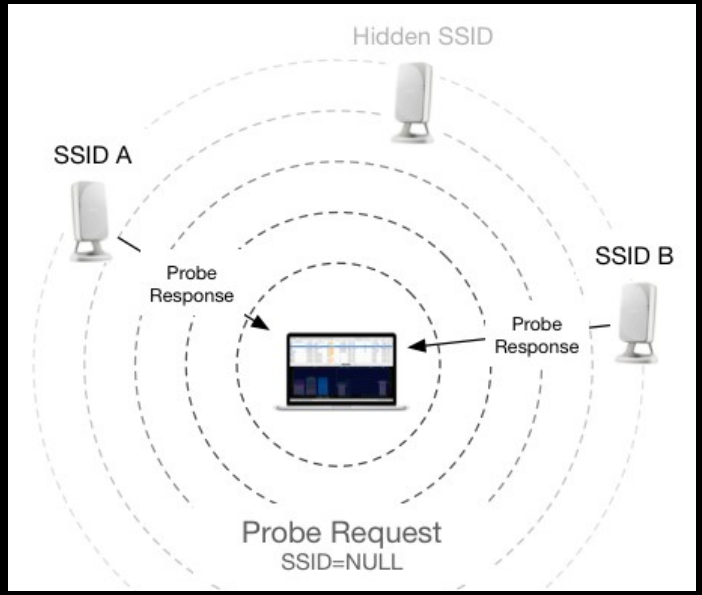
    `aireplay-ng`
  - Connect to network and control the drone

    `ardrone-webflight`

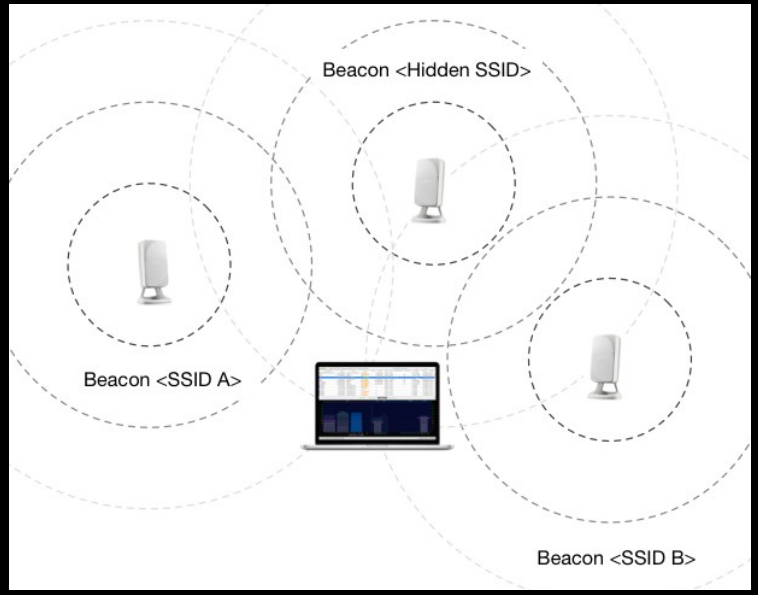




Parrot – WiFi network discovering with Scapy

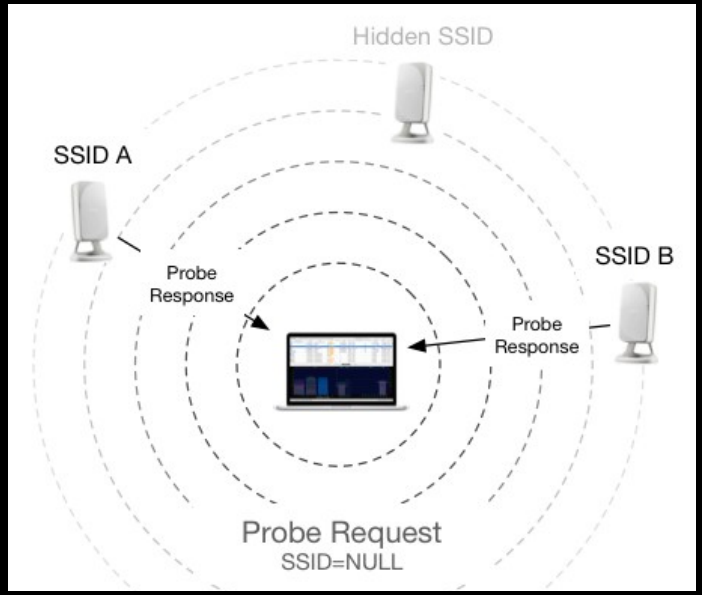


Active scan

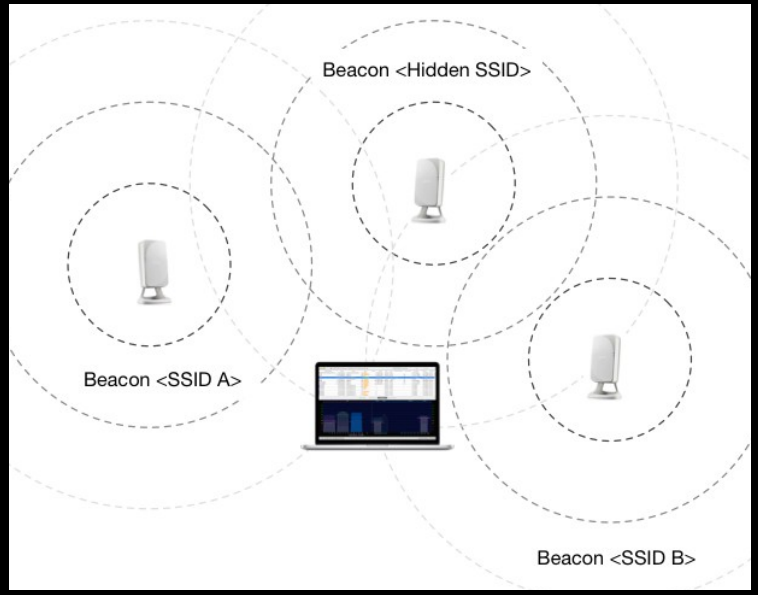


Passive scan

Parrot – WiFi network discovering with Scapy



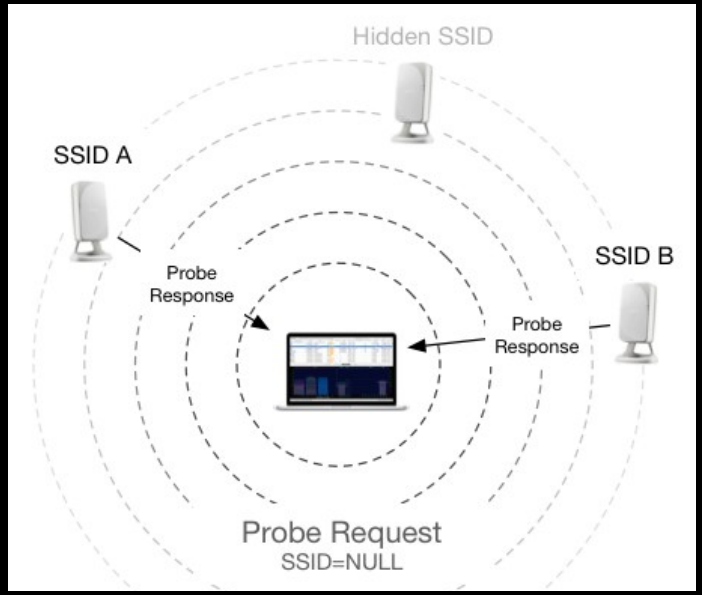
Active scan



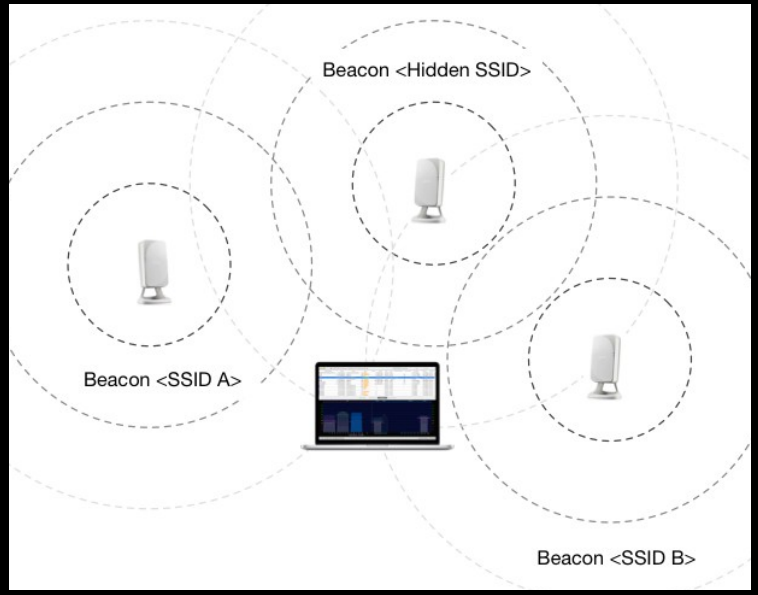
Passive scan

90:03: b7:c8 :68: d0

Parrot – WiFi network discovering with Scapy



Active scan

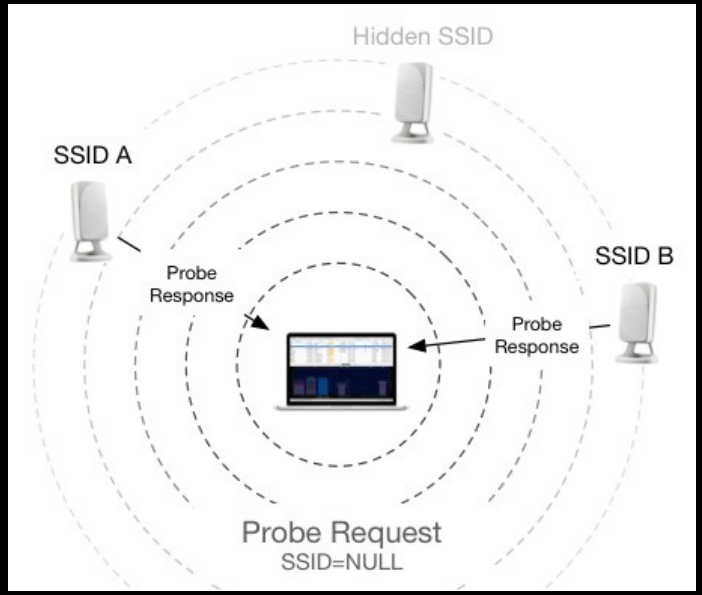


Passive scan

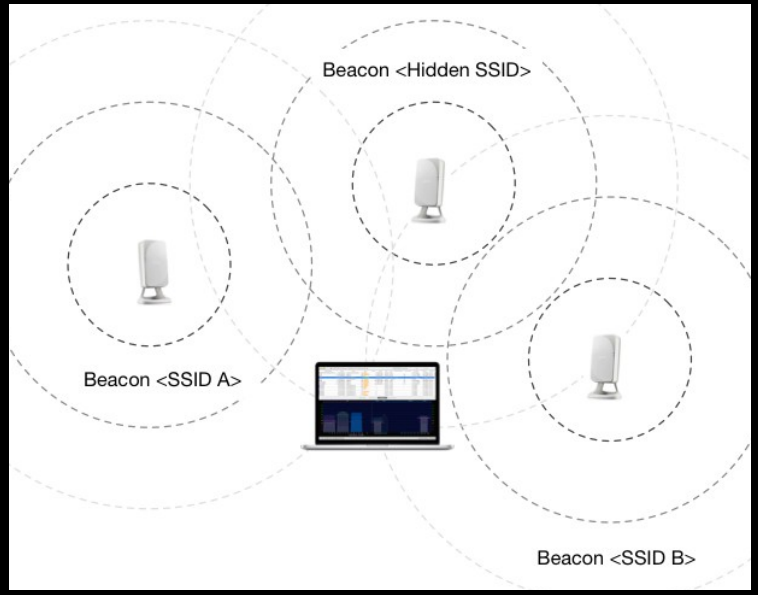
90:03: b7:c8 :68: d0

Organisationally Unique Identifier (OUI) | Network Interface Controller (NIC) specific

Parrot – WiFi network discovering with Scapy



Active scan



Passive scan

90:03: b7:c8 :68: d0

Manufacturer filtering ←

Organisationally Unique Identifier (OUI)

Network Interface Controller (NIC) specific

Parrot – Embedding tool on a predator drone

- **Raspberry Pi Zero W**
  - Bluetooth adapter
  - WiFi adapter
  - SPI bus
  - Lightweight

Parrot – Embedding tool on a predator drone

- **Raspberry Pi Zero W**
  - Bluetooth adapter
  - WiFi adapter
  - SPI bus
  - Lightweight

- **Tried reducing network adapter count**

Parrot – Embedding tool on a predator drone

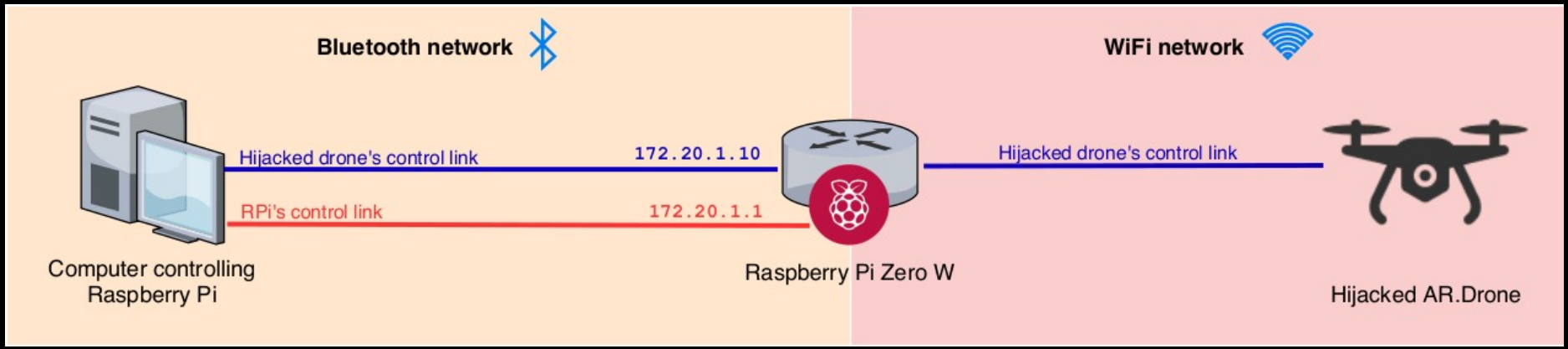
- **Raspberry Pi Zero W**
  - Bluetooth adapter
  - WiFi adapter
  - SPI bus
  - Lightweight

- **Tried reducing network adapter count**
- **Network to control board:**
  - Bluetooth PAN
  - SSH connection

Parrot – Embedding tool on a predator drone

- **Raspberry Pi Zero W**
  - Bluetooth adapter
  - WiFi adapter
  - SPI bus
  - Lightweight

- **Tried reducing network adapter count**
- **Network to control board:**
  - Bluetooth PAN
  - SSH connection
- **`ardrone-webflight` moved to attacker's computer**

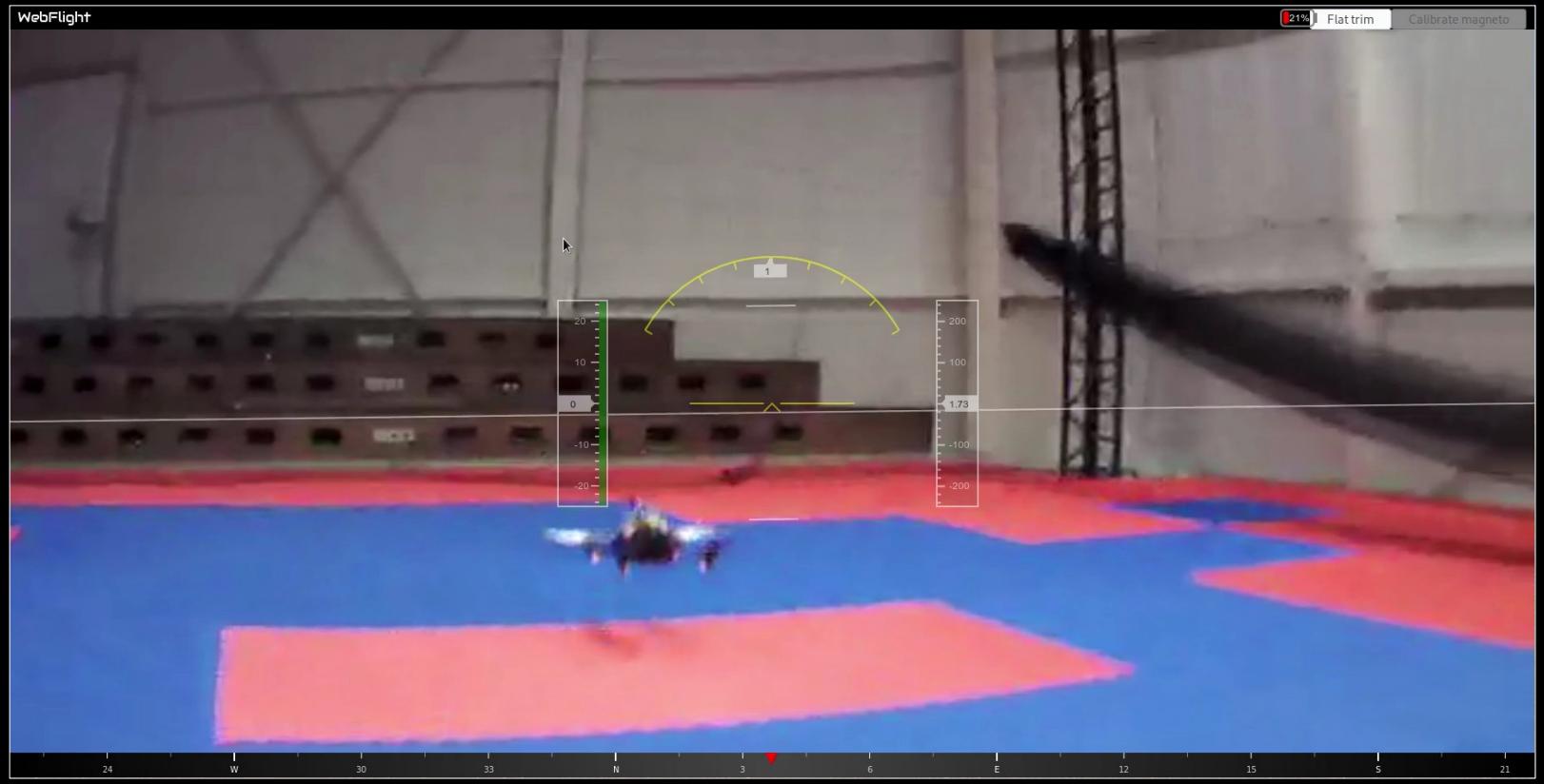


Parrot – Protection means

- **Use Parrot's solution**
  - Associate MAC address of real pilot
  - Bypassed by MAC spoofing
- **Do not send beacon frames**
  - Active search of WiFi networks
  - Eventually brute-force searching
- **Encrypt WiFi trafic**
  - Using WPA
  - Parrot Bepop example

Parrot Bepop

Drone-predator hacking Parrot – Demo



Conclusion

- Objective reached
- Learned a lot of things (WiFi, RF, Electronics, …)
- Ressources from n7 and ENAC
- And it is always fun to fly drones :-)

Questions

QUESTIONS

Promiscous mode with nRF24l01+

Preamble (1 byte)	Address (3-5 bytes)	Payload (0-32 bytes)	CRC (0-2 bytes)

No promiscuous mode <u>but</u> :

- Receiver does not check for preamble (0xAA), just checking the address

- Part before preamble is often interpreted as 0x00

- Configuration register can be set at ‘00’ even if it is out of spec. In this case the addess is 2 bytes

03	SETUP_AW				Setup of Address Widths (common for all data pipes)
	Reserved	7:2	000000	R/W	Only '000000' allowed
	AW	1:0	11	R/W	RX/TX Address field width '00' - Illegal '01' - 3 bytes '10' - 4 bytes '11' – 5 bytes LSByte is used if address width is below 5 bytes

→ We can configure pseudo-promiscous mode by setting 0x00AA as a receiver address.