OSINT Reverse Engineering of the ARFz

Marc Newlin / marc@bastille.net / @marcnewlin

Hack in the Box 2016 CommSec

Bastille

Marc Newlin

Security Researcher @ Bastille Networks

Agenda

1. Radio Primer
2. OSINT for ARFz
3. Mouse Reverse Engineering
4. MouseJack Demo

Radio Primer
(from a hacker’s perspective)

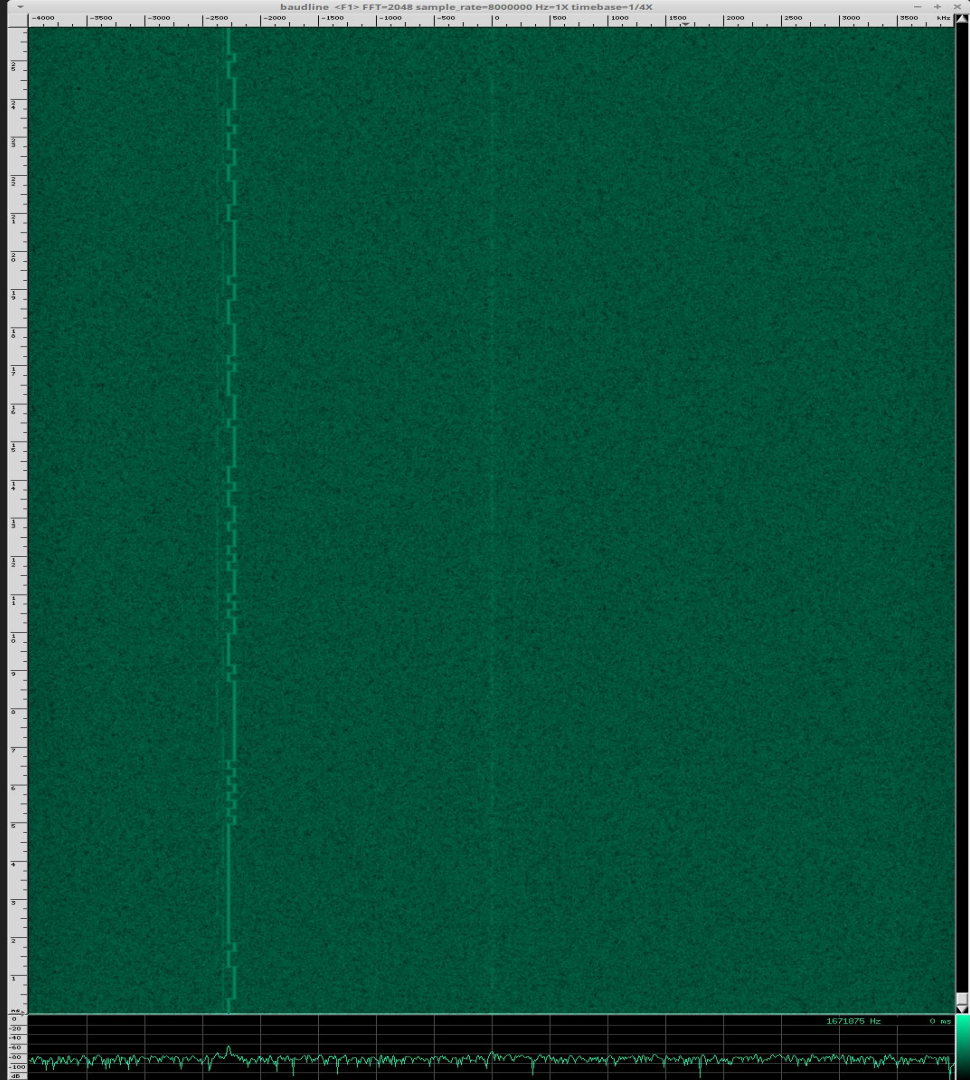
Hardware vs. Software Defined Radio

- Fixed functionality
- Really good at one thing
- Wifi card, wireless mouse dongle, bluetooth dongle, cellular modem, etc

- Reconfigurable on the fly
- Relies on computer or FPGA
- Lots of open source protocol stacks available
- USB and host computer timing limitations

Modulation

- Defines how the carrier wave is modified to encode the bits we are transmitting
- FSK - frequency shift keying
- OOK - on off keying
- ASK, QAM, OFDM, DSSS, FHSS
- Generally knowing the modulation is sufficient to decode w/ GNU Radio



Symbols and Samples

- Symbol is a fixed-length state that encodes one or more bits
- Sample is a single value resulting from quantizing radio data by the SDR
- Need at least two samples per symbol
- 1Mbps FSK needs a 2MS/s sample rate

Frequencies and Hopping

- Device can operate on one or more frequencies (channels)
- Frequency hopping -- actively hopping between channels
- Frequency agility -- opportunistically selecting channels for best performance
- Some devices stay on a single channel, regardless of transceiver support

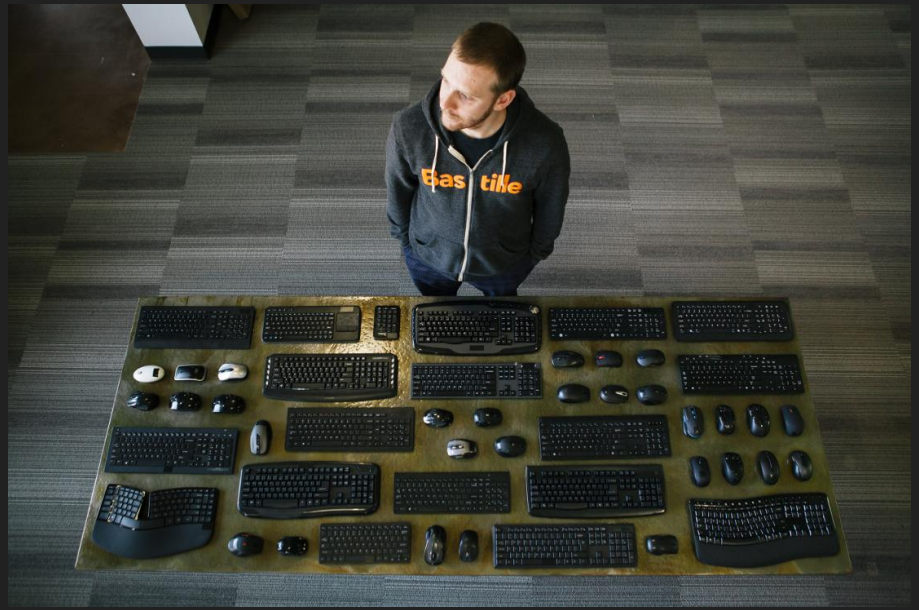
Channel Coding

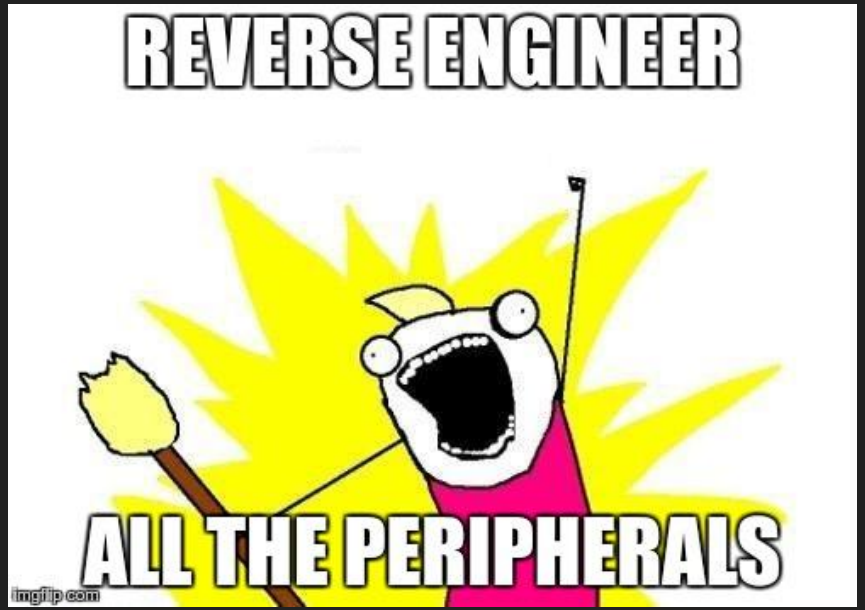
- Add redundancy to detect and correct errors during transmission
- Has the potential to alter payloads so they aren’t immediately recognizable
- FEC (Forward Error Correction)
- ARQ (Automatic Repeat Request)
- Repetition Coding
- Convolutional Coding

Reverse Engineering

MouseJack necessitated efficiency

- 50+ devices, 4 transceiver families





What are we trying learn?

- How does device work?
- Can we uniquely identify each device?
- What data can we passively sniff?
- Does its presence pose a risk?
- What does aberrant or malicious behavior look like?
- How can we fuzz the device/protocol?
- How quickly can we find all this out?

Standardized vs. Proprietary

1. Fully Standardized (Bluetooth, BLE, ZigBee, etc)
2. Partially Proprietary (nRF24L, TI-CC2544, etc)
3. Fully Proprietary (undocumented RFIC, SDR, etc)

RadioShack Wireless Mouse (and Dongle)

- Fully Proprietary
- 2.4GHz wireless mouse + dongle
- Need to do some OSINT!

Reverse Engineering Process

(Where da ARFz at??)

1. OSINT
2. Verify w/ Spectrum Analyzer
3. ARFz to Bytes
4. Packet Formats
5. Payloads and Protocol

1. OSINT for ARFz

OSINT Resources

- FCC documents
- RFIC product specifications
- The Google

Best case, what will we find?

- Modulation
- Symbol Rate / Data Rate
- Frequencies
- Frequency Hopping Behavior
- Channel Coding
- Whitening / Scrambling
- Packet Format
- Protocol Behavior

FCC Certification Process

1. Device is manufactured
2. Test lab evaluates the device
3. Telecommunications certification body issues a grant of certification
4. Test report, application, and related exhibits published in FCC database
5. Some exhibits are confidential (temporarily or permanently)

Finding FCC Exhibits

- Lookup FCC ID @ https://www.fcc.gov/general/fcc-id-search-page
- Click on the ‘Detail’ link on the results page

OET Exhibits List

10 Matches found for FCC ID JNZMR0054

View Attachment	Exhibit Type	Date Submitted to FCC	Display Type	Date Available
Confidentiality Request.pdf	Cover Letter(s)	12/11/2015	pdf	12/15/2015
Cover Letter - Agent Authorization.pdf	Cover Letter(s)	12/11/2015	pdf	12/15/2015
External Photos.pdf	External Photos	12/11/2015	pdf	05/10/2016
Label ID Label Location Information.pdf	ID Label/Location Info	12/11/2015	pdf	12/15/2015
Internal Photos.pdf	Internal Photos	12/11/2015	pdf	05/10/2016
RF Exposure Information (MPE).pdf	RF Exposure Info	12/11/2015	pdf	12/15/2015
Test Report.pdf	Test Report	12/11/2015	pdf	12/15/2015
Test Setup Photos.pdf	Test Setup Photos	12/11/2015	pdf	05/10/2016
User Manual (Statements).pdf	Users Manual	12/11/2015	pdf	05/10/2016
User Manual.pdf	Users Manual	12/11/2015	pdf	05/10/2016

Relevant Exhibit Types

Test Reports	Behavior of RF emissions
Internal Photos	What’s in the box?!?
User Manuals	Varying levels of technical detail
Schematics	Rare, but useful
Operational Descriptions	Also rare, and only sometimes useful

Test Reports

- Does the device meet FCC guidelines?
  - Transmit power
  - Bandwidth
  - Frequencies
  - Duty cycle
- 2498 authorized test labs
- Each lab has one or more report formats
- Each lab provides a varying degree of detail

Test Report from Bureau Veritas (test lab)

Product	2.4GHz Cordless Mouse
Brand	Logitech
Test Model	M-R0054
Status of EUT	ENGINEERING SAMPLE
Power Supply Rating	DC 3.7V from battery DC 5V from USB interface
Modulation Type	GFSK
Modulation Technology	DTS
Transfer Rate	Up to 2Mbps
Operating Frequency	2402MHz ~ 2481MHz
Number of Channel	8
Output Power	3.899 mW
Antenna Type	PCB printed antenna with 1.32 dBi gain
Antenna Connector	NA
Accessory Device	NA
Data Cable Supplied	USB Charging cable (Shielded, 1.8m with one core)

RadioShack Mouse Test Report

- 2408-2474 MHz
- 67 channels
- **Intertek** test lab

1.1 Product Description

The Equipment Under Test (EUT) is a Wireless Optional Mouse. It can pair with a corresponding dongle. The 2.4GHz module in the EUT is operating in the frequency range from 2408MHz to 2474MHz (67 channels with 1MHz channel spacing). The EUT is powered by 1.5VDC “AA” size batteries.

The Model: 2603754, 2603755, 2603770, DM-3652RM and DM-3752RM are the same as the Model: 2603752 in hardware aspect except different color and cosmetic details. The difference in model number serves as marketing strategy. Only model: 2603752 is tested.

Antenna Type : Internal, Integral

RadioShack Dongle Test Report

- 2408-2474 MHz
- 34 channels, 2 MHz spacing
- GFSK modulation
- 1Mbps data rate
- **Neutron Engineering** test lab

Frequency Channel							
Channel	Frequency (MHz)	Channel	Frequency (MHz)	Channel	Frequency (MHz)	Channel	Frequency (MHz)
01	**2408**	10	2426	19	2444	28	2462
02	2410	11	2408	21	2446	29	2464
03	2412	12	2430	21	2448	30	2466
04	2414	13	2432	22	2450	31	2468
05	2416	14	2434	23	2452	32	2470
06	2418	15	2436	24	2454	33	2472
07	2420	16	2438	25	2456	**34**	**2474**
08	2422	**17**	**2440**	26	2458		
09	2424	18	2442	27	2460		

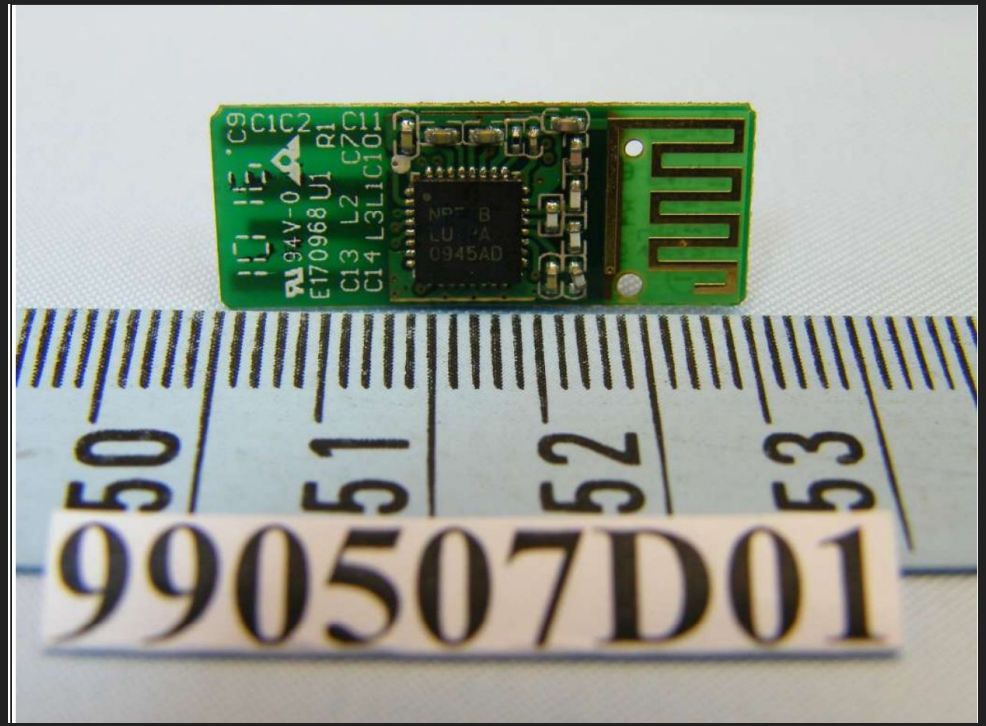
Equipment	2.4GHz Nano Transceiver
Brand Name	Fujitsu
Model Name.	DR-9053RM; DR-9055RM
Model Difference	Only difference is model name.
Product Description	The EUT is a 2.4GHz Nano Transceiver. Product Type: Low Power Communication Device Operation Frequency: 2408~2474 MHz Modulation Technology: GFSK Data rate: 1Mbps Number of Channel: 34CH .Please see note 2. (Page 9) Antenna Gain(Peak): Please see note 3.(Page 9). Field Strength: 68.87 dBuV/m (AV Max.) Based on the application, features, or specification exhibited in User's Manual, the EUT is considered as an ITE/Computing Device. More details of EUT technical specification. Please refer to the User's Manual.
Power Source	DC voltage supplied from system.
Power Rating	I/P AC 120V/60Hz O/P DC 5V
Connecting I/O Port(s)	Please refer to the User's Manual

Internal Photos

- Varying degree of resolution
- Some vendors blackout RFIC markings
- No standardization

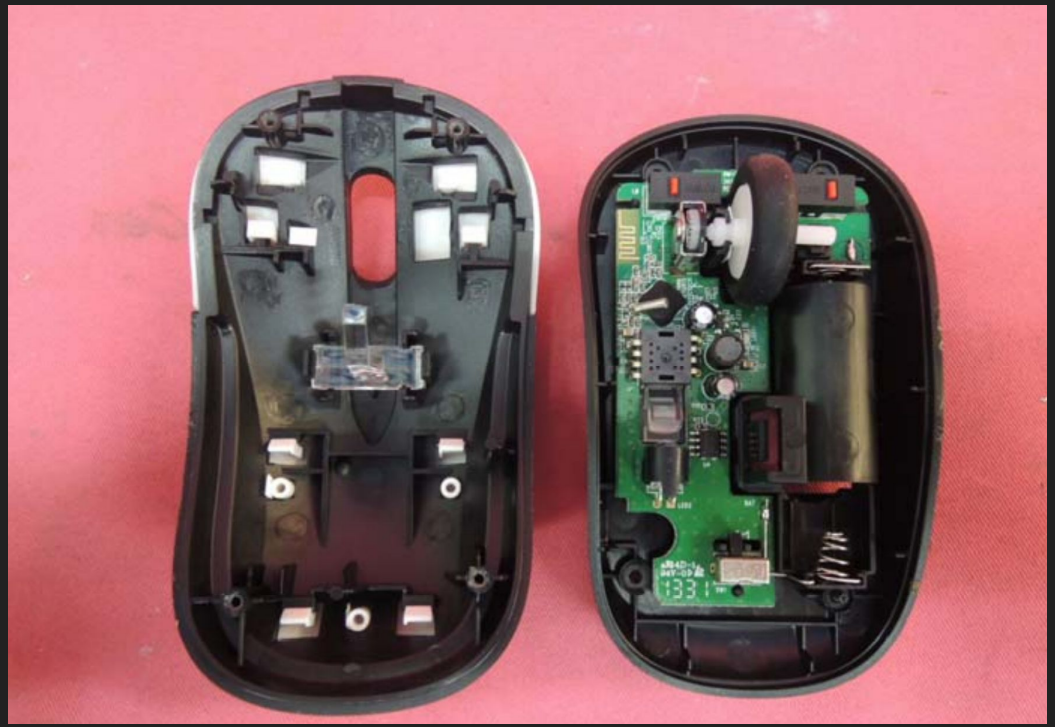
Internal photo of a Microsoft dongle

- nRF24LU+
- Partially blacked out markings
- Well documented RFIC
- (easy mode)



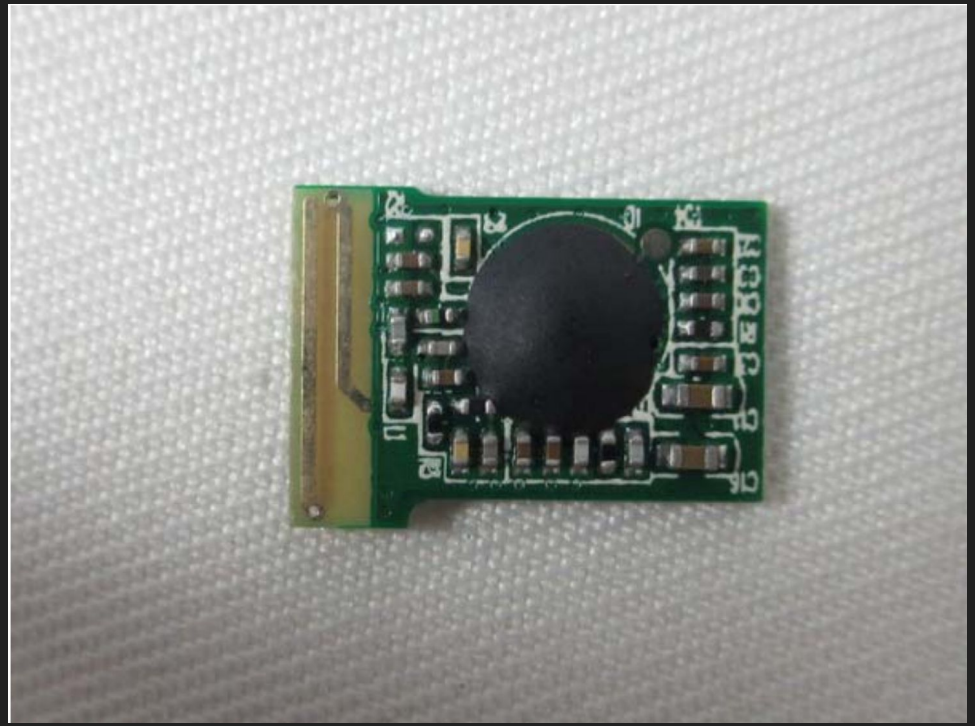
RadioShack Mouse Internal Photos

- Low resolution
- Nothing useful



RadioShack Dongle Internal Photos

- Better picture
- Still nothing useful

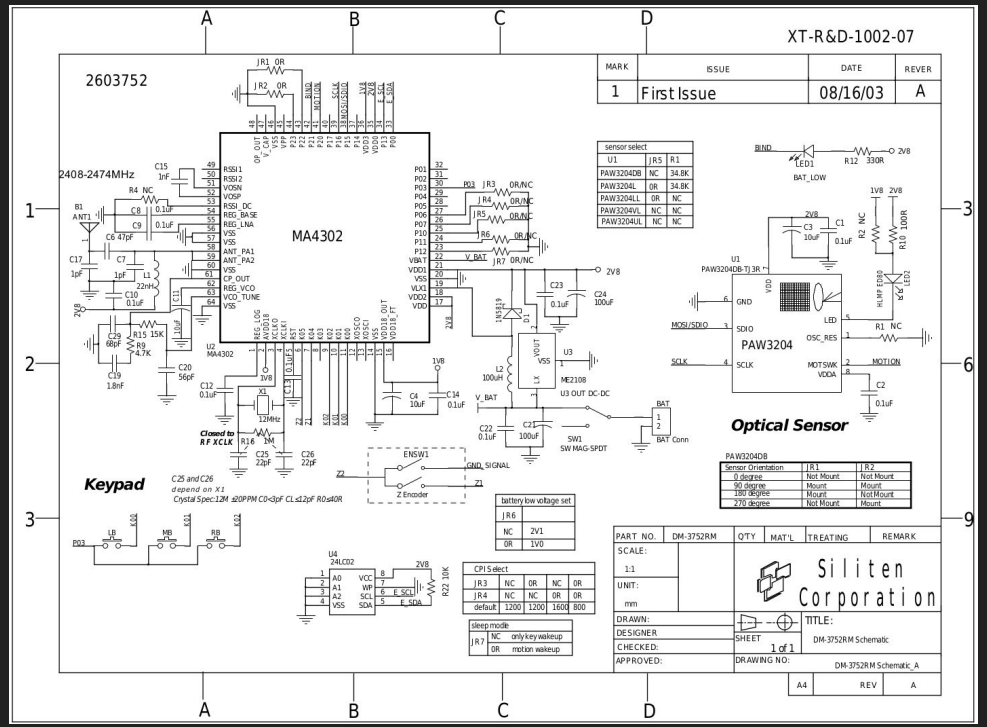


Schematics

- Most vendors request permanent confidentiality on schematics
- More common with lesser known manufacturers
- When available, extremely useful to learn RFIC specifics

RadioShack Mouse Schematic

- MA4302 RFIC



Operational Descriptions and User Manuals

- Describes the device behavior in an undefined format
- Hit or miss, but potentially fruitful
- Some vendors include useful technical details

RadioShack Mouse Operational Description

- Bluetooth !?!?
- FSK modulation
- 2408-2474 MHz
- 67 channels
- MA4302 RFIC

Technical Description

The Equipment Under Test (EUT) is a Wireless Optical Mouse. It can pair with a corresponding dongle. The 2.4GHz module in the EUT is operating in the frequency range from 2408MHz to 2474MHz (67 channels with 1MHz channel spacing). The EUT is powered by 3.0VDC (2 x 1.5VDC “AAA” size batteries).

2.4GHz Bluetooth Module:
Modulation Type: FSK
Antenna Type: Integral, Internal (PCB Trace)
Frequency Range: 2408MHz - 2474MHz, 1MHz channel spacing, 67 channels

Nominal field strength is 96.0dBμV/m @ 3m
Production Tolerance of field strength is +/- 3dB
Antenna gain is 1.1dBi

The functions of main ICs are mentioned below.

1. 2.4GHz module MA4302
1) MA4302 acts as the 2.4GHz radio core of 2.4GHz module
2) 12MHz crystal (X1) provides clock for MA4302
3) U4 (24LC02) is serial EEPROM for parameter backup of MA4302

RadioShack Dongle User Manual

- 2408-2472 MHz
- GFSK modulation
- 1Mbps data rate
- human house only!!!

3. Wireless specifications

2.408-2.474GHz frequency coverage.

GFSK RF transceiver

High Speed RF link data rate Max. 1M bit/s

Caution

Please use the Transceiver in human house only and keep away water.

Children don’t to install the Transceiver.

MA4302 mouse via Google-fu

- Marketing material from another mouse with the same RFIC
- FSK modulation
- FHSS
- Mosart MA4302 RFIC
- More Google-fu, still no RFIC spec sheet

STANDARDS SUPPORTED
Low-power FHSS FSK wireless technology

BATTERY AND POWER
Batteries: 1 x AA
Battery life: up to 500 hours continuous use

ADVANCED FEATURES
2 mouse buttons
Scroll/Click wheel
Select DPI button

ADDITIONAL SPECIFICATIONS
Optical Sensor type: IR (Avago A3000)
Chipset type: Mosart MA4302
Sensor resolution: 1000, 1500, 2000 DPI (native)
3 slide skids
Tracking speed: 30 ips max.
Acceleration: 20 g max.
Latency: < 16 ms

SUPPORTED OPERATING SYSTEMS
Microsoft® Windows® XP, Windows Vista®, Windows® 7, MAC® OSX™

WIRELESS RANGE
Up to 10 metres

FREQUENCY RANGE
2.4 GHz ISM band

So what do we “know”?

- GFSK modulation
- 2408-2474 MHz frequency range
- 34 or 67 channels
- 1Mbps data rate
- Maybe FHSS?
- Maybe Bluetooth?

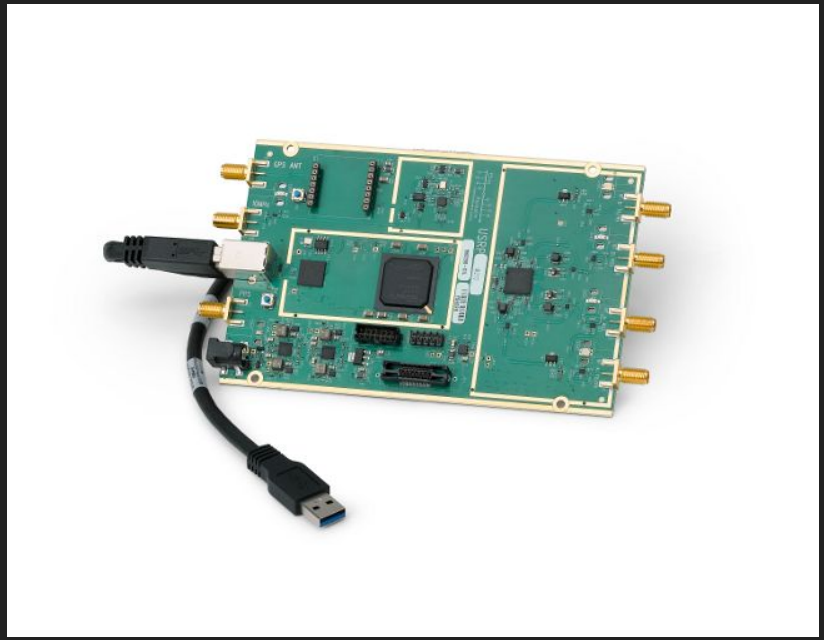
2. Verify w/ Spectrum Analyzer

Tools and Equipment

- Software Defined Radio
- RF Test Enclosure
- GNU Radio
- gr-fosphor
- baudline
- RadioShack mouse and dongle

Software Defined Radio

1. Streams raw RF data to a host computer
2. Reconfigurable bandwidth and center frequency
3. Lots of popular options (USRP, BladeRF, HackRF, RTL-SDR, LimeSDR, etc)

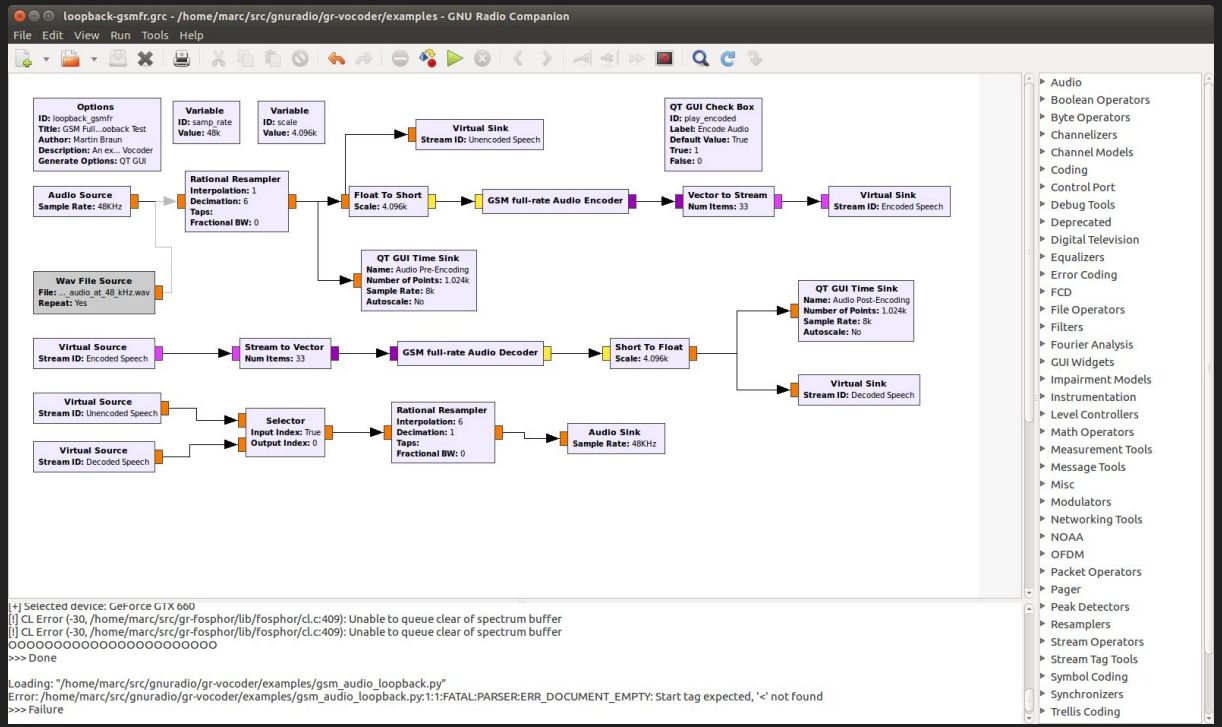


GNU Radio

- Open source SDR toolkit written in C/C++ and Python
- Large selection of signal processing libraries
- Hardware support for common SDR platforms
- Efficient prototyping

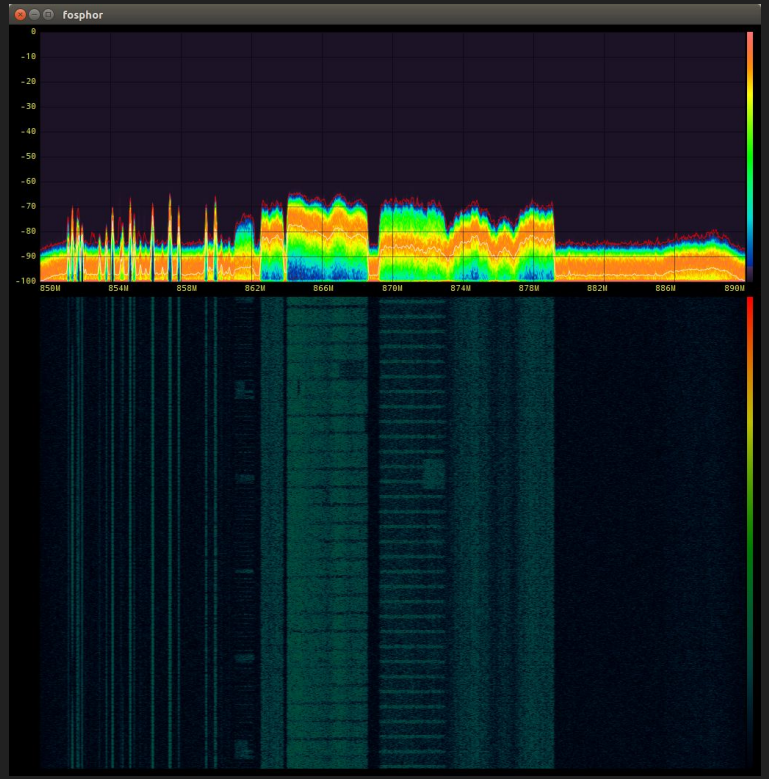
GNU Radio Companion

- Drag and drop flow graph creator
- Quick and easy



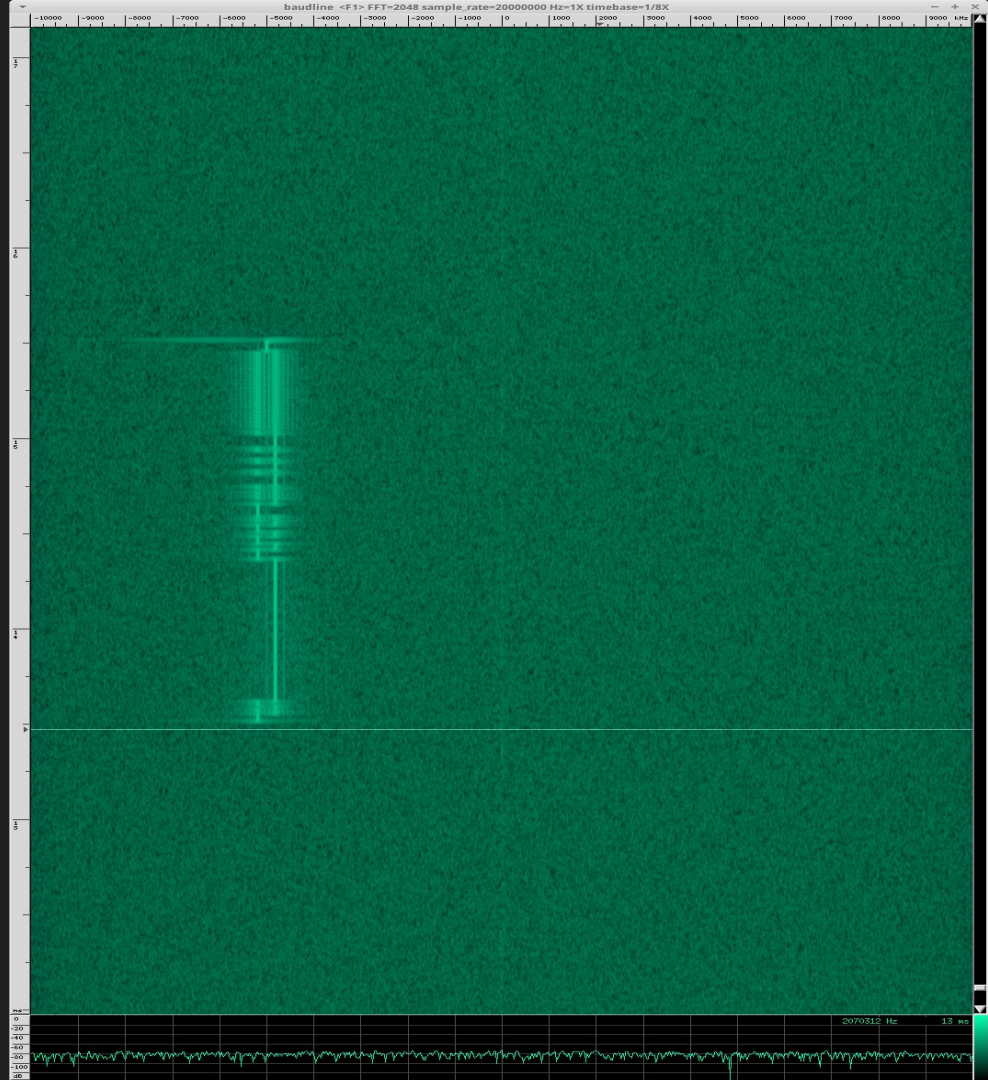
gr-fosphor

- OpenCL/OpenGL accelerated spectrum visualization tool
- Out-of-tree GNU Radio module



baudline / gr-baz

- Spectrum visualization tool
- Excellent for analyzing signals
- GNU Radio block in gr-baz

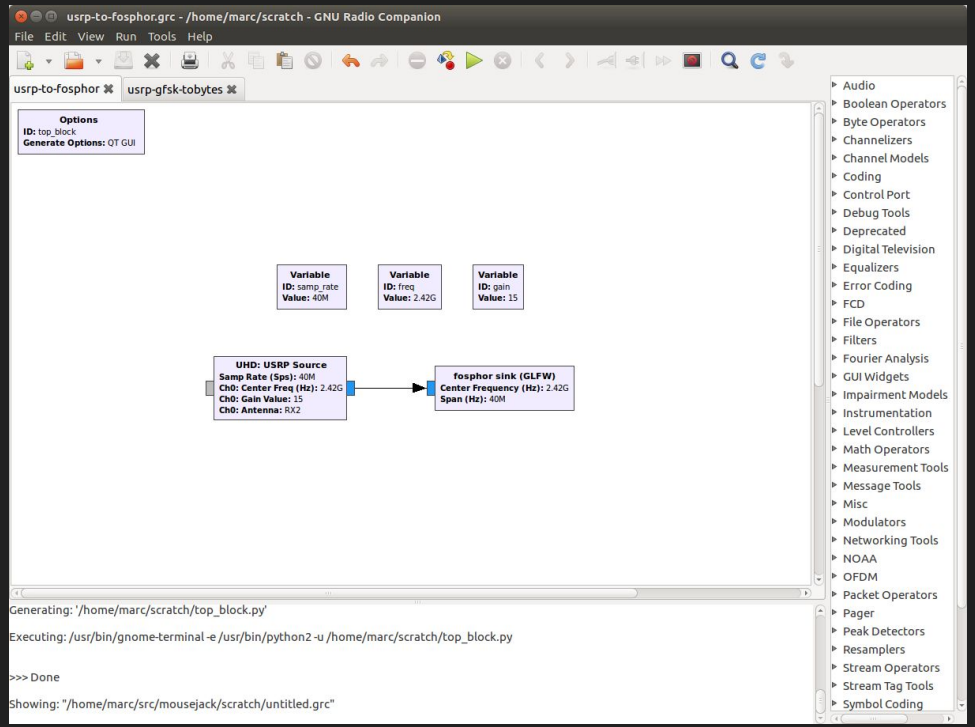


RF Test Enclosure (Faraday Cage)

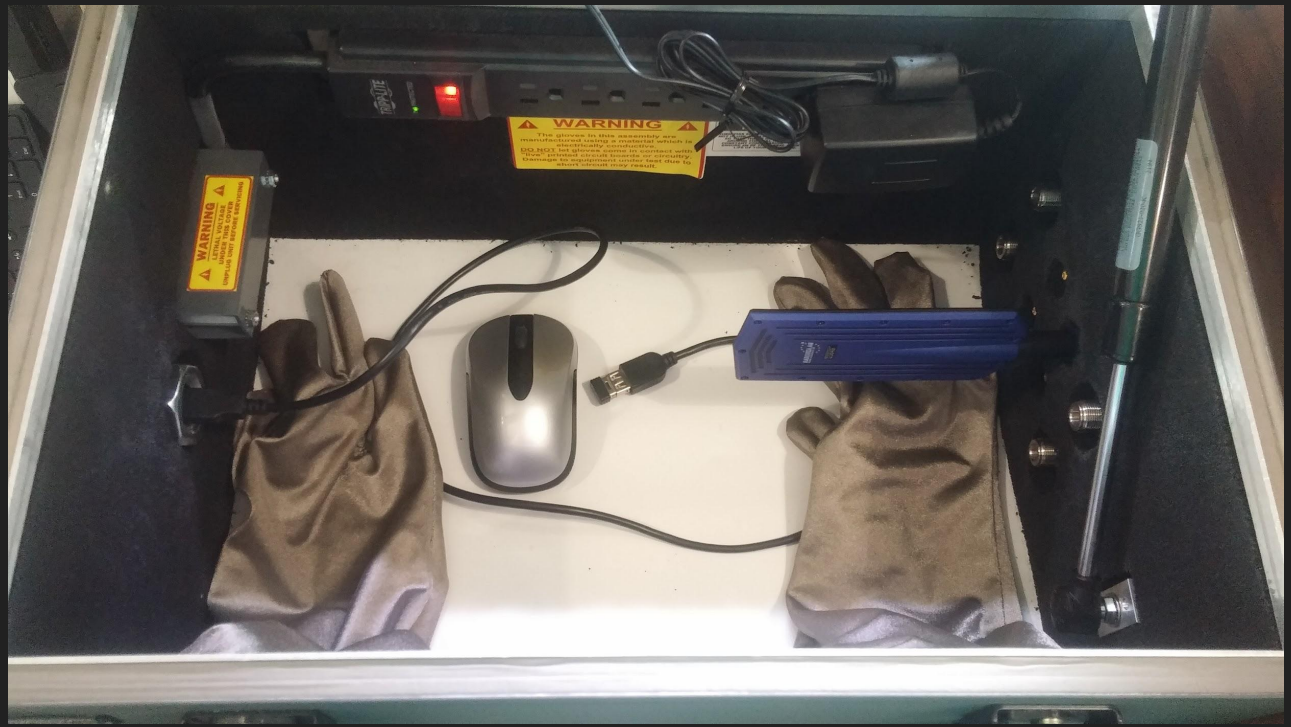
- Attenuates the ARFz
- Isolate devices for reverse engineering
- Prevent unintended side effects of fuzzing
- Keeps the FCC happy :)

SDR to gr-fosphor flow graph

- USRP data source
- gr-fosphor data sink
- 40 MHz bandwidth
- 2420 MHz center frequency
- 15 dB antenna gain

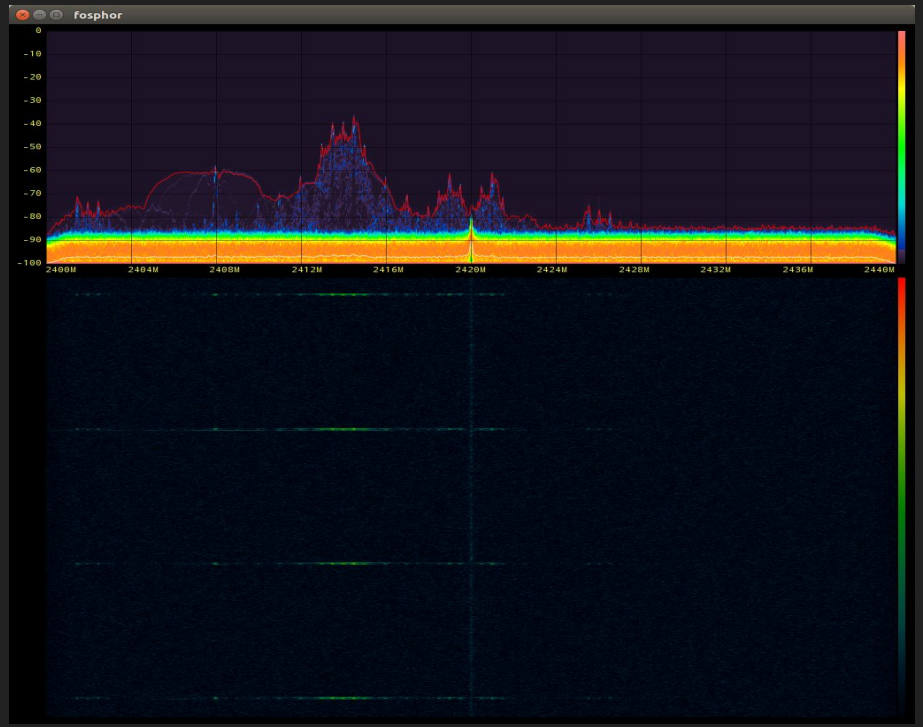


RF Test Enclosure



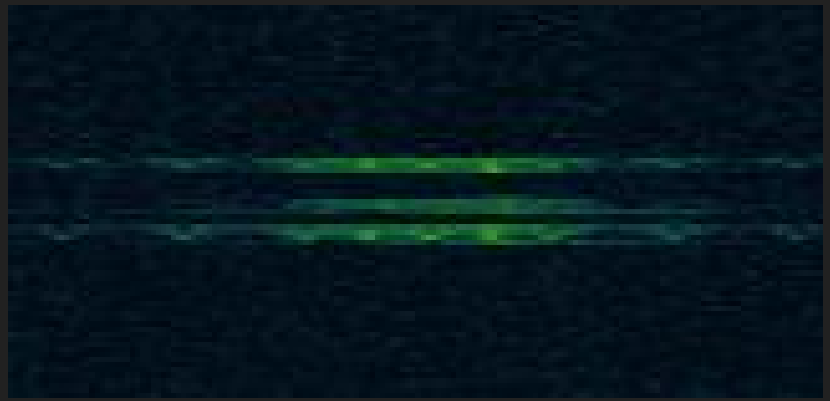
gr-fosphor: RadioShack dongle

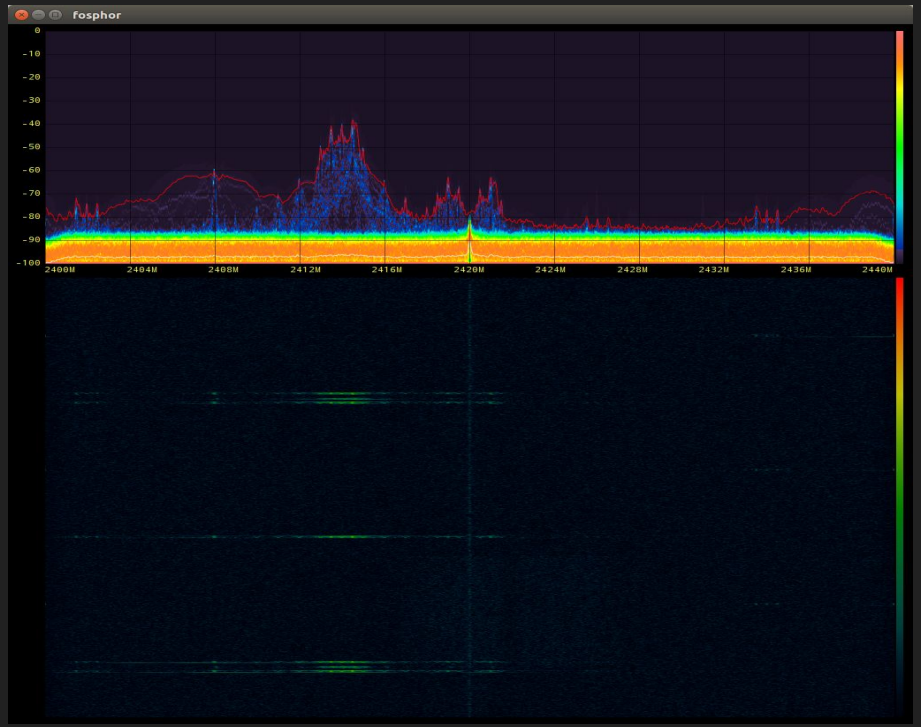
- 2414 MHz
- TX at regular intervals
- Looks about 1 MHz wide
- Sync packet?



gr-fosphor: RadioShack mouse + dongle

- Also camped at 2414 MHz
- 3 packet sequence with mouse movement
- Sync + data + ack?

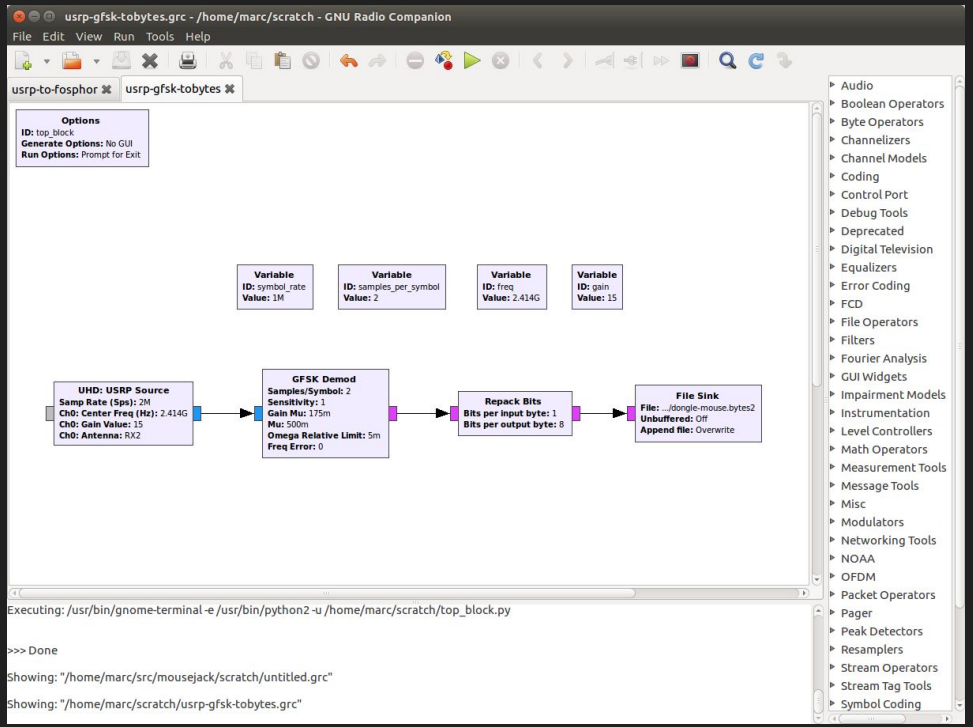




3. ARFz to Bytes

Flowgraph to demodulate mouse/dongle traffic

- USRP data source at 2414 MHz
- GFSK demodulator
- 2 MHz sample rate
- 2 SPS (1Mbps data rate)
- Bits to bytes
- File data sink



Capture some packets

- Need to generate repeated packets
- Capture dongle alone to isolate the sync packets
- Capture mouse + dongle, repeatedly clicking the mouse

4. Packet Format

Anatomy of an RF packet

Preamble	Clock correction / synchronization
Sync field	Start of payload delimiter, can be static value or the address
Address	Receiver address
Header	Describes the packet, depending on protocol complexity
Payload	The actual data being transmitted
Checksum	Checksum, CRC, etc
Postamble	End of frame delimiter, more clock correction

Prep the data (binary to hex)

```
$ xxd -p dongle.bytes | tr -d '\n' > dongle.bytes.hex
```

```
$ xxd -p dongle-mouse.bytes | tr -d '\n' > dongle-mouse.bytes.hex
```

Standard command line tools enable quick and dirty analysis.

- grep
- xxd
- sort
- uniq

Byte boundaries mean we only see a subset of the packets.

Find the preamble (dongle)

```
$ grep -Po "(00|ff|aa|55)+" dongle.bytes.hex | sort | uniq -c | sort -nr
    528 5555555555
    514 ffff
    468 aaaaaaaaaa
    392 ffff5555555555
    349 ffffaaaaaaaaaaaa
    281 55ff
    243 aaaa
    226 5555
    158 aa55
    156 55aa
```

We grep for a tone (0x00 or 0xFF), or alternating 1’s and 0’s (0xAA or 0x55).

Find the longest repeated sequences (dongle)

```
$ grep -Po "(ffff[a5]{12}).{16}" dongle.bytes.hex | sort | uniq -c | sort -nr
    392 ffffaaaaaaaaaaaa1116e8d14b782aff
      1 ffff5aaaaaaaaaaa1116e8d14b782aff
```

- Look for shifted preamble variants (FFFF followed by 12 A’s or 5’s)
- Increase the number of bytes after the preamble until it no longer repeats
- The most repeated sequence is likely the dongle sync packet

Sanity check the packets (dongle)

```
$  grep -Pob "(ffffaaaaaaaaaaaa1116e8d14b782aff)+" dongle.bytes.hex | head -n 10
28215:ffffaaaaaaaaaaaa1116e8d14b782aff
32221:ffffaaaaaaaaaaaa1116e8d14b782aff
44253:ffffaaaaaaaaaaaa1116e8d14b782aff
46255:ffffaaaaaaaaaaaa1116e8d14b782aff
56291:ffffaaaaaaaaaaaa1116e8d14b782aff
58297:ffffaaaaaaaaaaaa1116e8d14b782aff
80365:ffffaaaaaaaaaaaa1116e8d14b782aff
84377:ffffaaaaaaaaaaaa1116e8d14b782aff
98420:ffffaaaaaaaaaaaa1116e8d14b782aff
126506:ffffaaaaaaaaaaaa1116e8d14b782aff
```

Packet offsets are multiples of ~2000 bytes, or 16ms. Looks good!

Isolate the mouse packets

```
$ sed -i "s/ffffaaaaaaaaaaaa1116e8d14b782aff//g" dongle-mouse.bytes.hex
```

Remove the dongle packets from the mouse + dongle capture to isolate the mouse packets.

Find the preamble! (mouse)

```
$ grep -Po "(00|ff|aa|55)+" dongle.bytes.hex | sort | uniq -c | sort -nr
   2898 ffff
    765 5555555555
    666 aaaaaaaaaa
    578 ff00
    357 55ff
    280 aaaa
    272 5555
    215 55aa
    204 aa55
```

We grep for a tone (0x00 or 0xFF), or alternating 1’s and 0’s (0xAA or 0x55).

No repeated occurrences of the dongle preamble, so we’ll try ‘em all!

Find the longest repeated sequences (mouse)

```
$ grep -Po "aaaa.{20}" dongle-mouse.bytes.hex | sort | uniq -c | sort -nr | head -n 10
     14 aaaa1116e8d126dbfa706aff
     11 aaaa1116e8d121dbfae0eff
     10 aaaa1116e8d12edbfad1c3ff
      8 aaaa1116e8d12fdbfae1f4ff
      7 aaaa1116e8d12ddbfa819aff
      7 aaaa1116e8d129dbfa4146ff
      7 aaaa1116e8d128dbfa7171ff
      6 aaaa1116e8d123dbfa8081ff
      6 aaaa1116e8d122dbfab0b6ff
      5 aaaa1116e8d12ddbfa819aff
```

Many repeated payloads, which may point to a sequence number.

Sanity check the packets (mouse)

```
$ grep -Pob "(...)" dongle-mouse.bytes.hex | head -n 10
167823:aaaa1116e8d126dbfa706aff
263746:aaaa1116e8d122dbfab0b6ff
303715:aaaa1116e8d126dbfa706aff
423469:aaaa1116e8d121dbfae0efff
455379:aaaa1116e8d127dbfa405dff
591291:aaaa1116e8d124dbfa1004ff
691083:aaaa1116e8d121dbfae0efff
738884:aaaa1116e8d128dbfa7171ff
878869:aaaa1116e8d129dbfa4146ff
1170597:aaaa1116e8d12edbfad1c3ff
```

Packet offsets are multiples of ~2000 bytes and more spaced out than the dongle packets. Looks good!

Packet candidates

Dongle packet:

```
ffffaaaaaaaaaaaa1116e8d14b782aff
```

Address / sync word?:

```
1116e8d1
```

Preambles / postambles:

```
ffffaaaaaaaaaaaa
aaaa
ff
```

Mouse packets:

```
aaaa1116e8d126dbfa706aff
aaaa1116e8d121dbfae0efff
aaaa1116e8d12edbfad1c3ff
aaaa1116e8d12fdbfae1f4ff
aaaa1116e8d12ddbfa819aff
aaaa1116e8d129dbfa4146ff
aaaa1116e8d128dbfa7171ff
aaaa1116e8d123dbfa8081ff
aaaa1116e8d122dbfab0b6ff
aaaa1116e8d12ddbfa819aff
```

Checksum / CRC

- 3 byte dongle payload
- 5 byte mouse payload
- potentially an 8 or 16 bit CRC (if any)
- check dongle payloads with CRC RevEng

```
$ reveng -w 16 -s 26dbfa706a 21dbfae0ef 2edbfad1c3 2fdbfae1f4 2ddbfa819a
29dbfa4146 28dbfa7171 23dbfa8081 22dbfab0b6 2ddbfa819a

reveng: no models found
```

No dice :/

How about whitening?

- Some guesswork is required
- XOR’ing with some value?
- Reverse byte order?

```
$ ./reveng -w 16 -s 7c81a0302a 7b81a0b5ba 7481a0998b 7581a0aebb 7781a0c0db
7381a01c1b 7281a02b2b 7981a0dbda 7881a0ecea 7781a0c0db

width=16  poly=0x1021  init=0x0000  refin=false  refout=false  xorout=0x0000
check=0x31c3  name="XMODEM"
```

Success!! Payloads are whitened by XOR’ing with 0x5A repeated, and the CRC is in reversed byte order. Dongle payload appears to have no CRC.

Mouse click packet format

Preamble	`AAAA`
Address	`1116E8D1`
Payload	`3 bytes`
Checksum	`CRC-16 XMODEM`
Postamble	`FF`

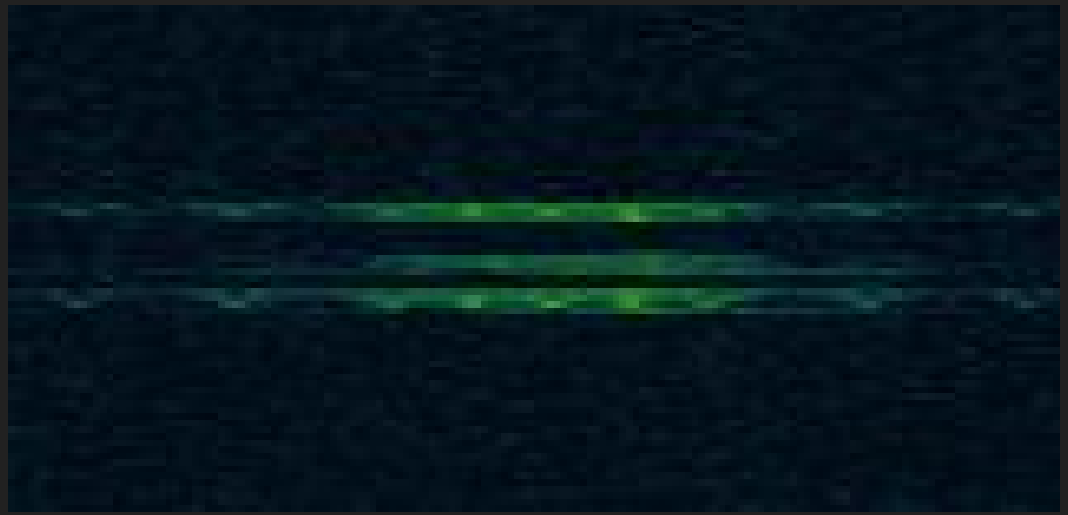
5. Payloads and Protocol

Build out the model with additional test data

- Test second mouse/dongle set to identify static vs. dynamic values
- Dongle sync packets are identical across devices, appear to be unprotected by CRCs
- Sync field is unique across devices, so it is indeed an address
- Second mouse/dongle set camps at 2426 MHz
- Mouse movement packets are 5 bytes in length

What about ACKs?

```
$ grep -Pob "1116e8d1.{12}" dongle-mouse.bytes.rev.hex
1932509:1116e8d120dbfad0d8 - mouse click payload
1932592:1116e8d14b78ff2752 - ACK(?), ~300us later
```



TDMA timing

- Dongle transmits sync packets every 16ms
- Mouse transmits packets following sync packets
- Dongle ACKs mouse packets

Reverse Engineering Payloads

- Generate RF traffic with known expected behavior
- Mouse clicks, scrolling, movement
- What changes over the air?

Mouse Payload Formats

Movement

```
4D 08 07 06 05
```

```
4  | Frame Type
D  | Sequence Number
08 | X1
07 | X2
06 | Y1
05 | Y2
```

Scroll

```
7E 81 FF
```

```
7  | Frame Type
E  | Sequence Number
81 | "Button" State
F  | Button Type (Scroll Wheel)
F  | Scroll Motion (Down 1)
```

Click

```
7A 81 A1 // left down
7A 01 A1 // left up
```

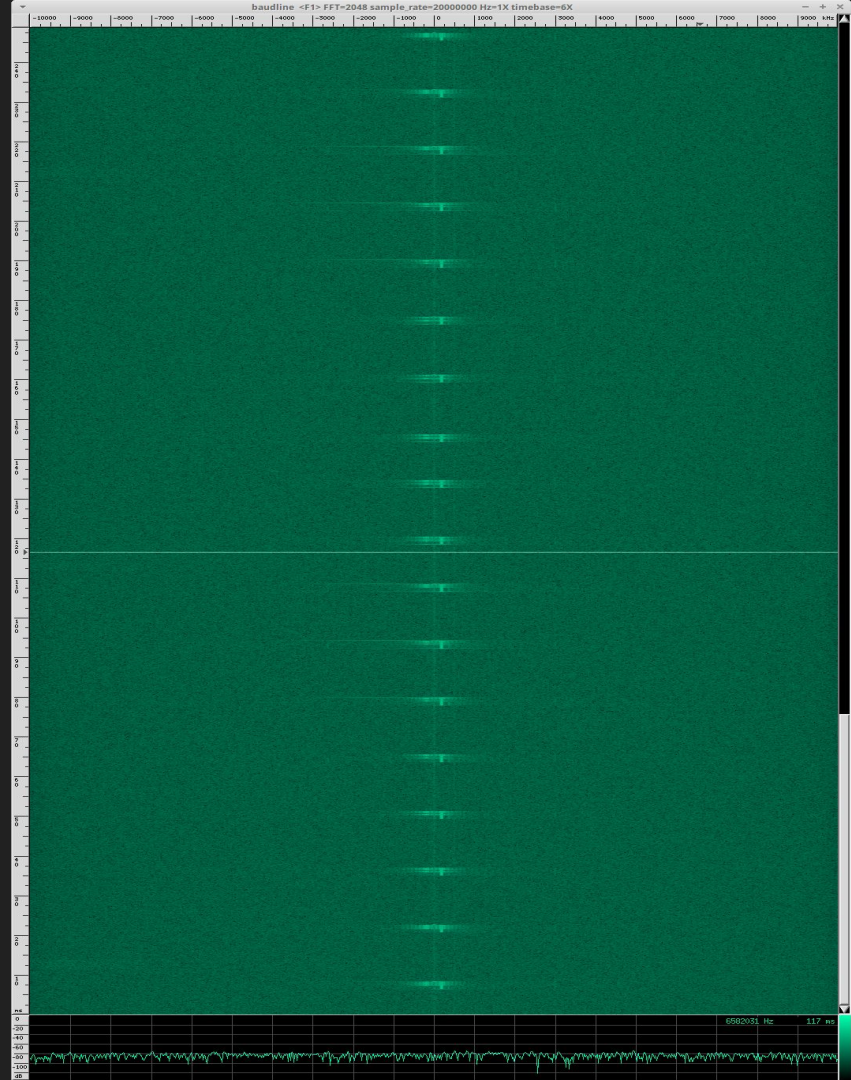
```
7  | Frame Type
A  | Sequence Number
81 | Button State
A  | Button Type (Button)
1  | Button (Left)
```

So what have we learned about the mouse?

- 4 packet formats
  - Dongle sync
  - Dongle ACK
  - Mouse movement
  - Mouse click
- GFSK modulation, 1Mbps data rate
- Device pair camps on a single channel
- Dongle transmits timing and frequency synchronization packets
- Mouse times its transmissions based on the dongle
- Likely 34 channels, spaced at 2 MHz, between 2408-2474 MHz
- Definitely not Bluetooth
- XMODEM variant of CRC-CCITT

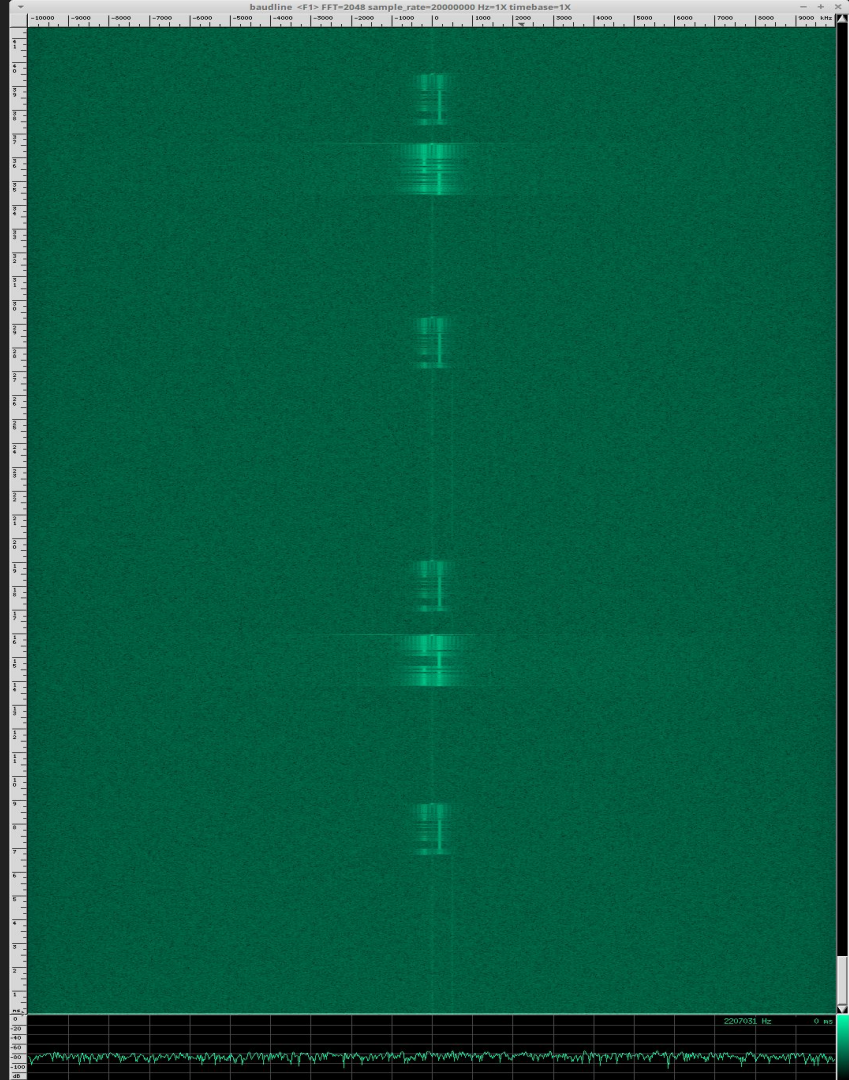
Quadcopter Visual Analysis

- Controller Only
- Transmitting every ~15ms



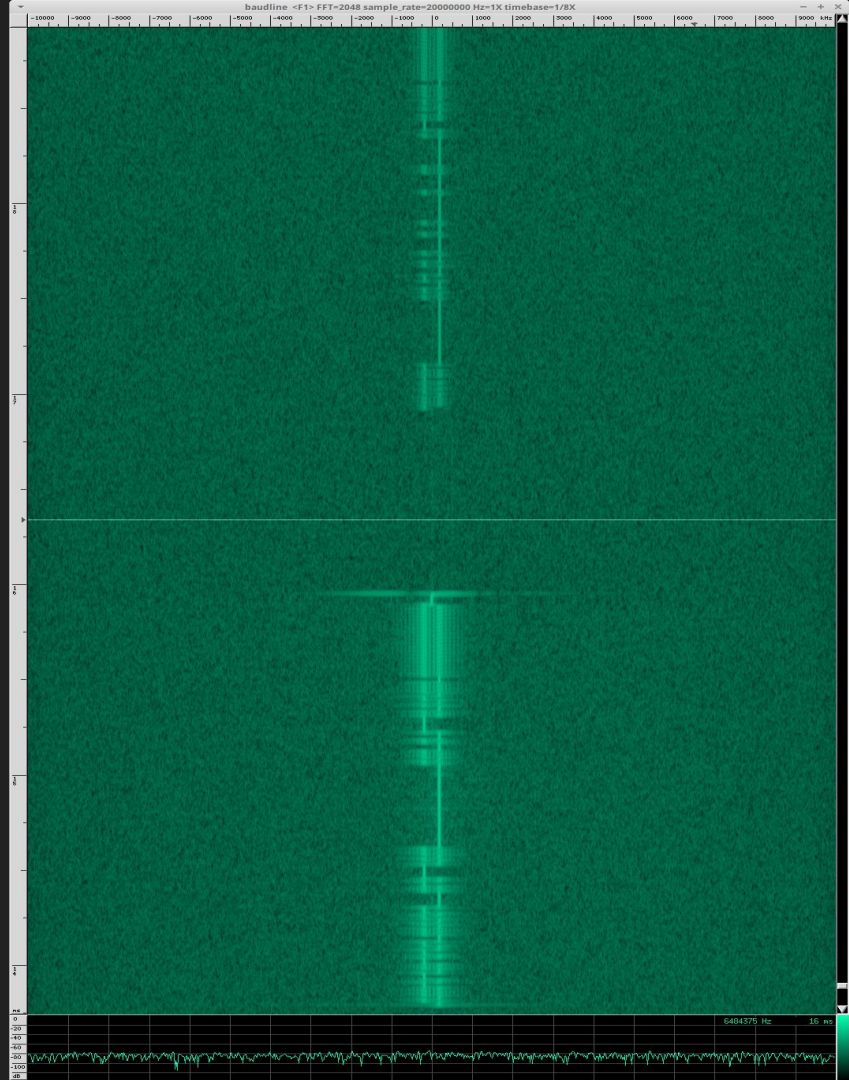
Quadcopter Visual Analysis

- Quadcopter Connected to Controller
- Quadcopter perhaps ACKs controller packets (higher power packets)



Quadcopter Visual Analysis

- Quadcopter Connected to Controller
- Zoomed in view
- FSK symbols are clearly visible

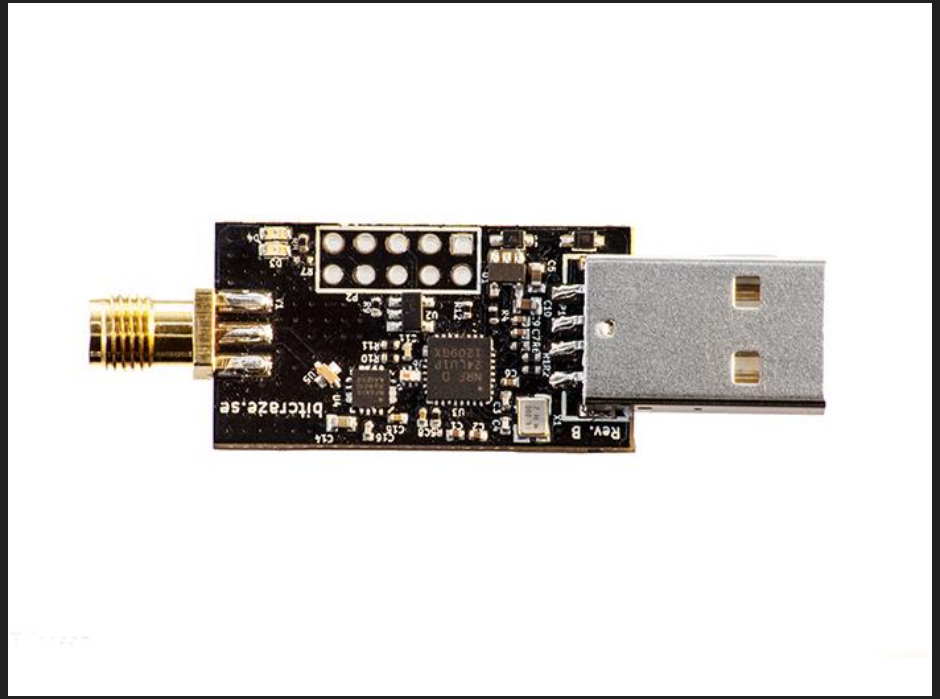


Process Recap

1. OSINT
2. Verify w/ Spectrum Analyzer
3. ARFz to Bytes
4. Packet Formats
5. Payloads and Protocol

MouseJack Demo - CrazyRadio Dongle

- nRF24LU1+ based dongle
- Part of the CrazyFlie project
- Open source
- 225 meter injection range with yagi antenna



MouseJack Demo - Logitech

- Forced pairing
- Disguise keyboard as mouse
- Unencrypted keystroke injection into keyboard address
- Firmware patch issued by Logitech

Questions?

Marc Newlin

marc@bastille.net

@marcnewlin