

# **Looping Surveillance Cameras**

(like in the movies)

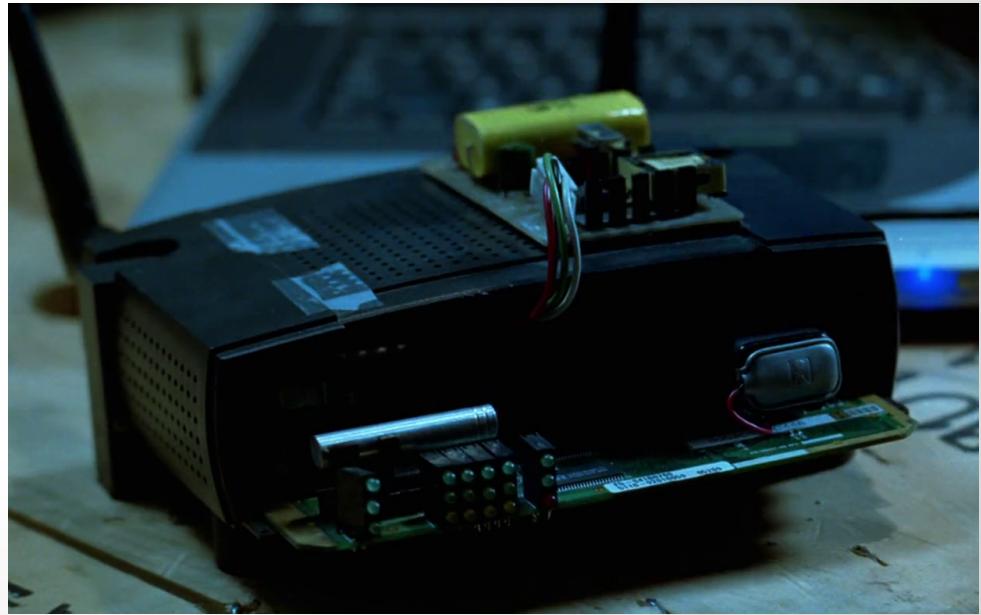
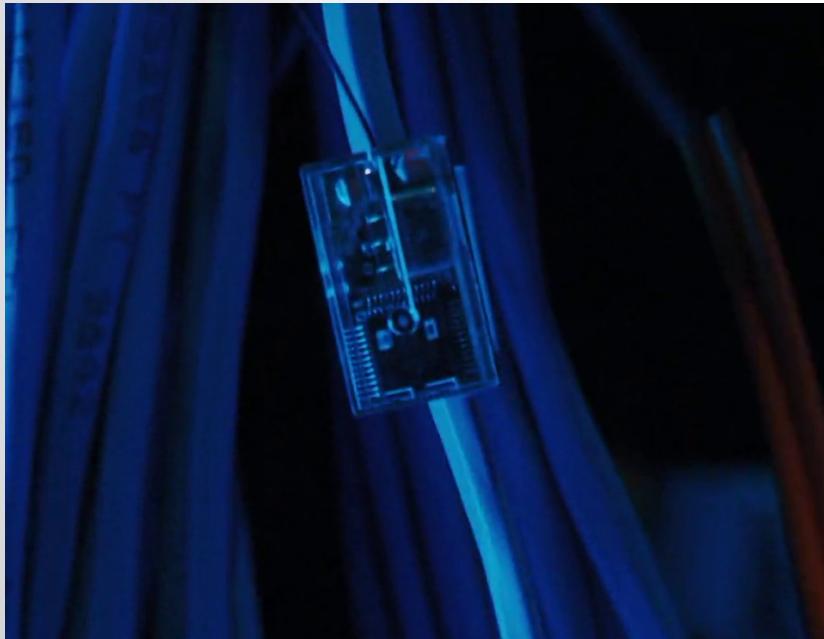
# Who are we?

- Ordinary law-abiding citizens
- Nothing to see here, move along
- Eric Van Albert <[eric@van.al](mailto:eric@van.al)>
- Zach Banks <[not-eric@van.al](mailto:not-eric@van.al)>

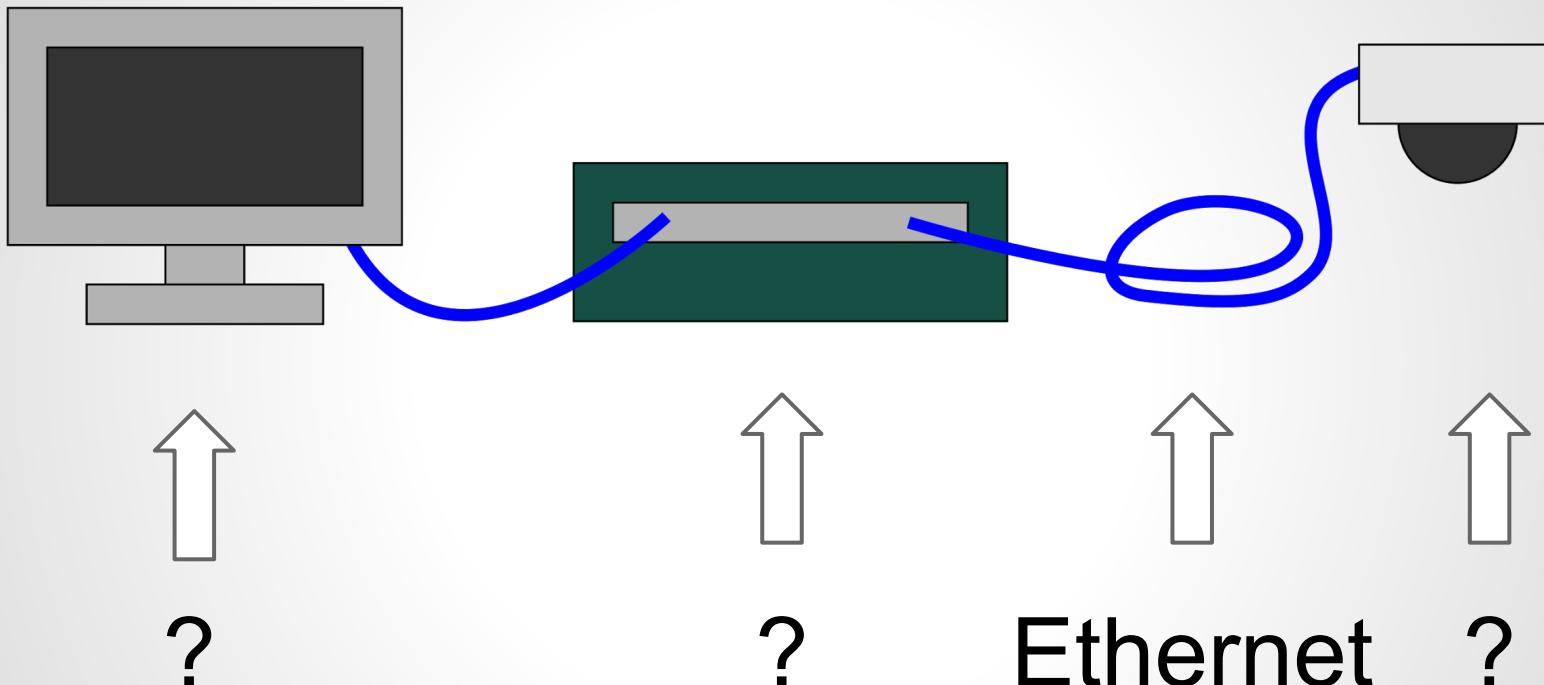
# Prior Art (what this isn't)



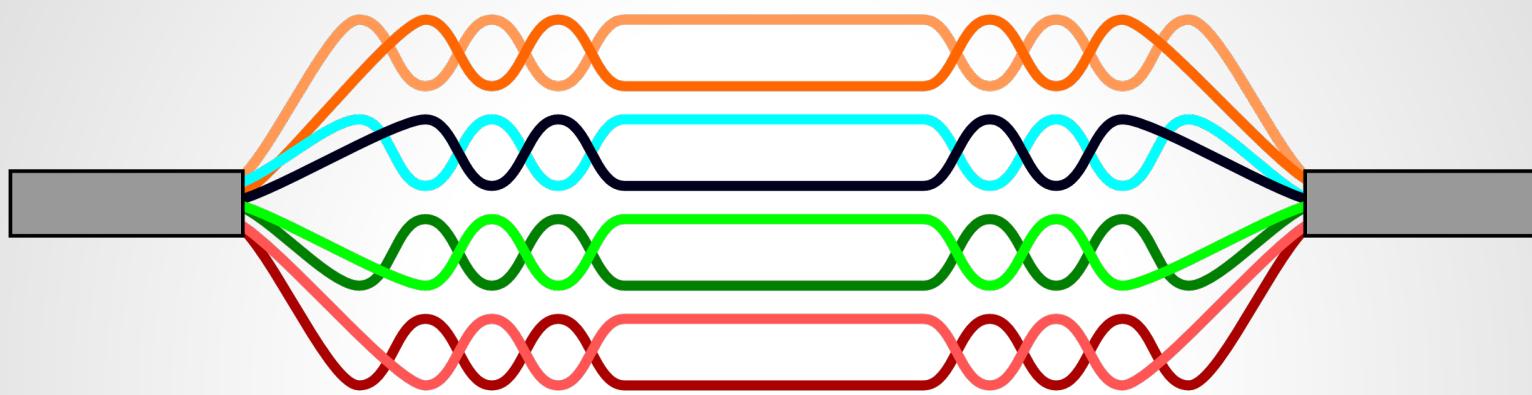
# Prior Art (what this is)



# System

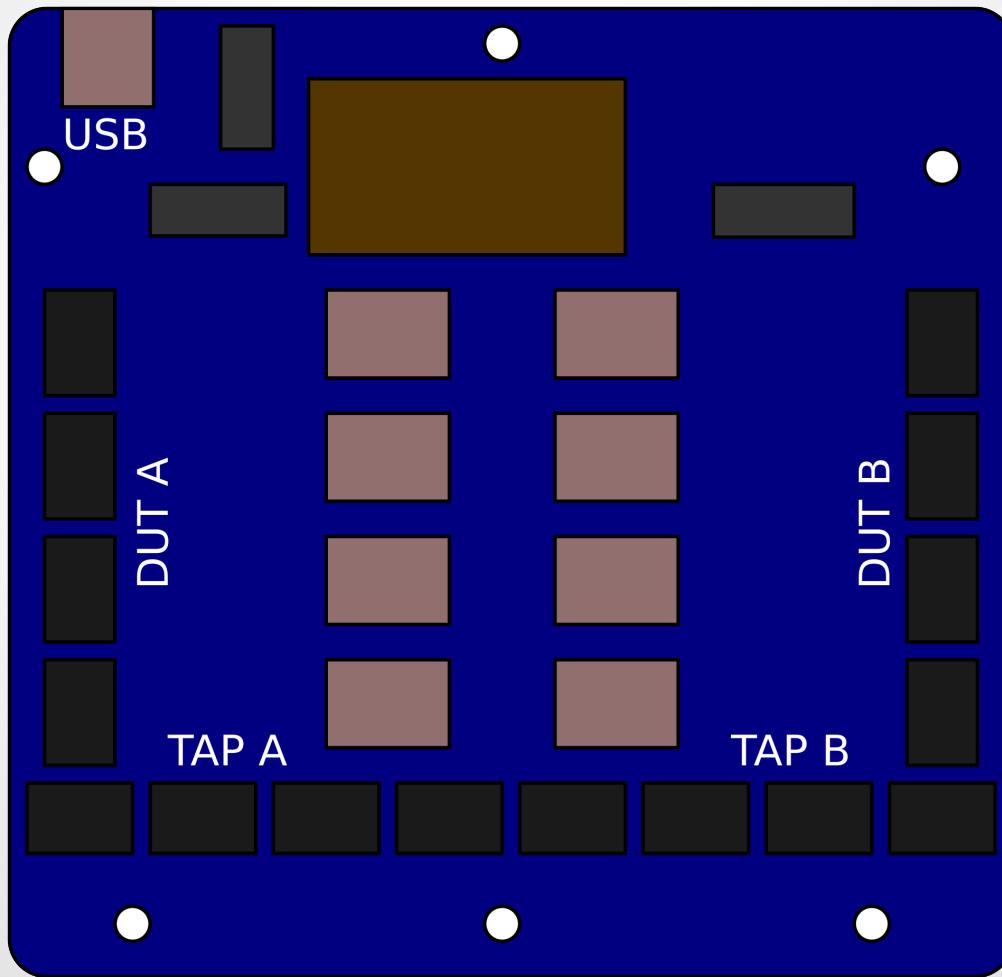


# Ethernet Anatomy



- Four twisted pairs
- All may be carrying data
- Wide variety of electrical standards

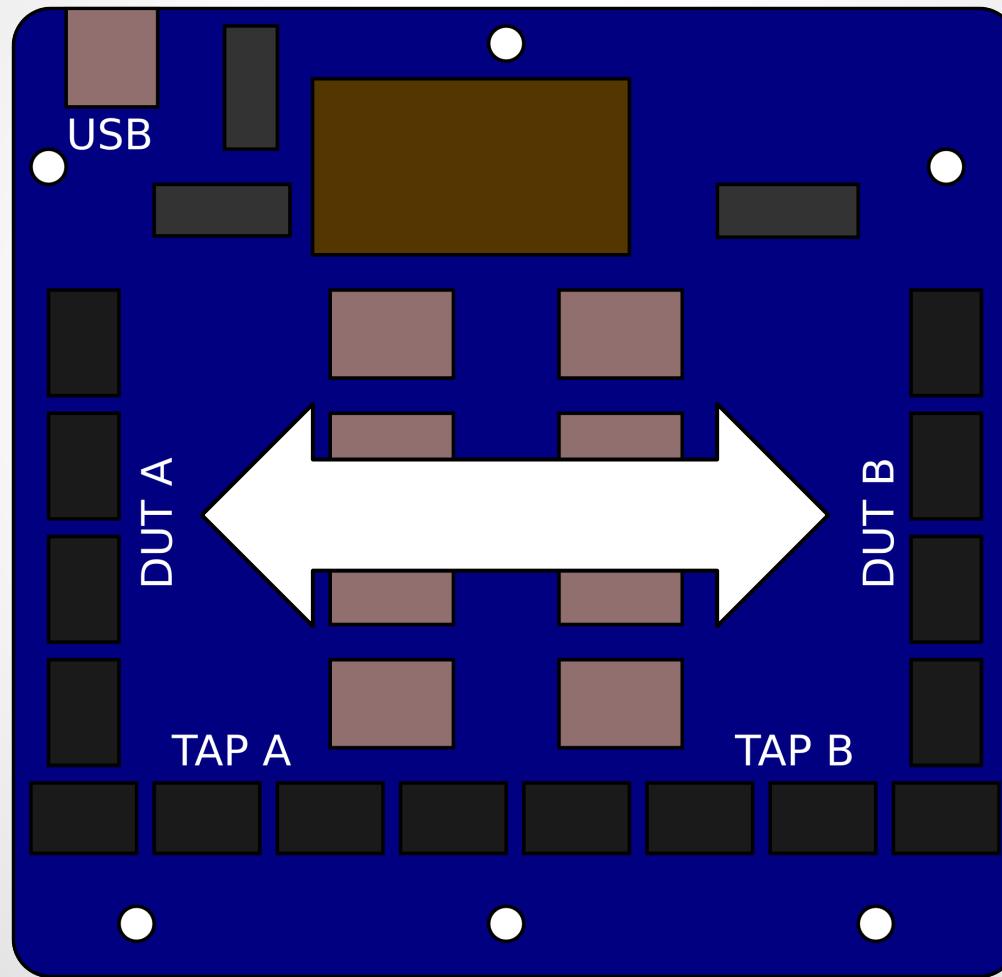
# The Tap Board



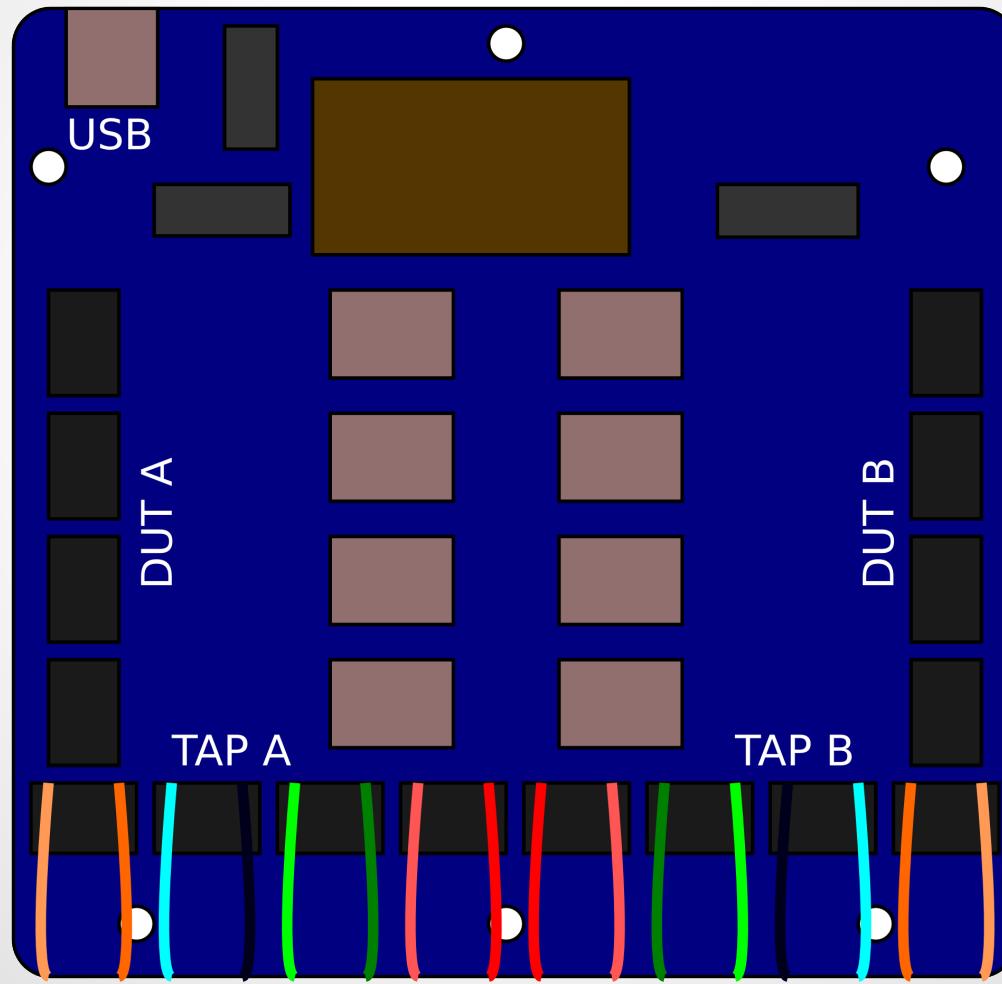
# The Tap Board

- Eight DPDT latching relays
  - Rated for 1 GHz
- Punch-down connectors
- Impedance-matched traces
- Powered and controlled over USB

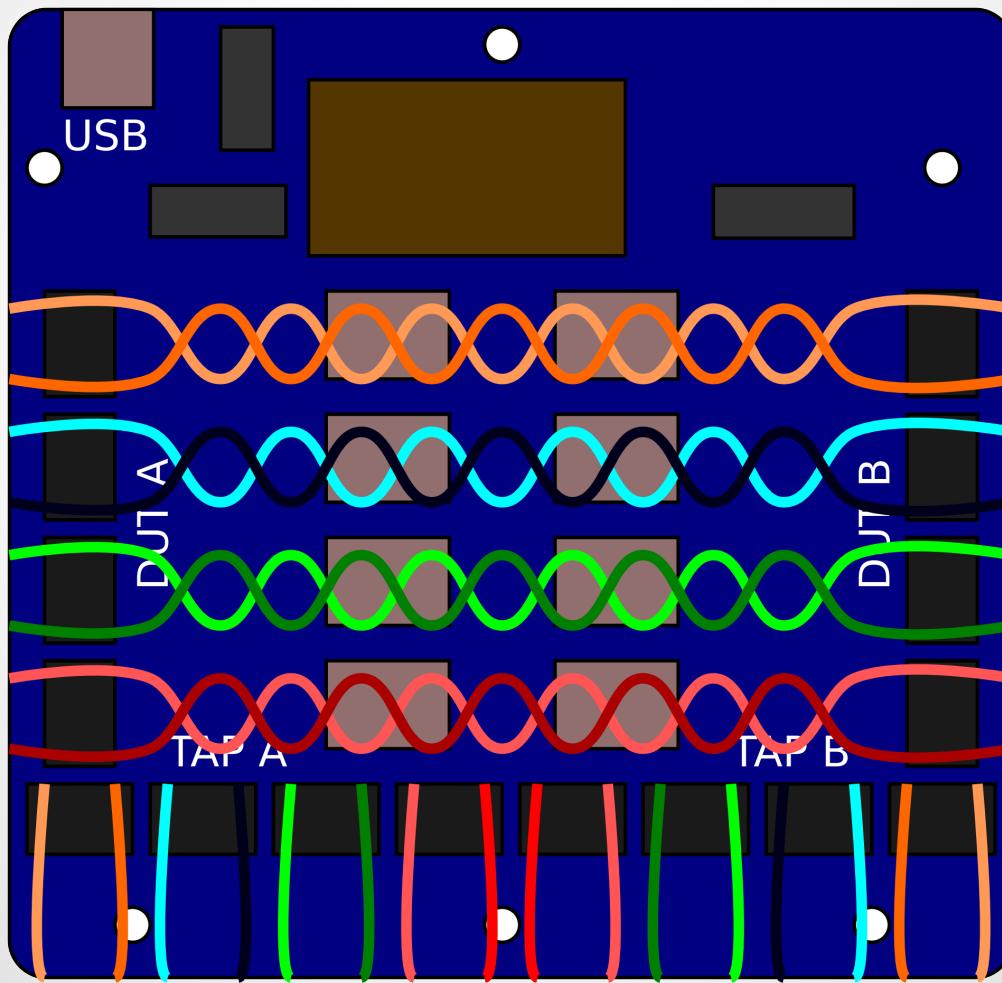
# The Tap Board



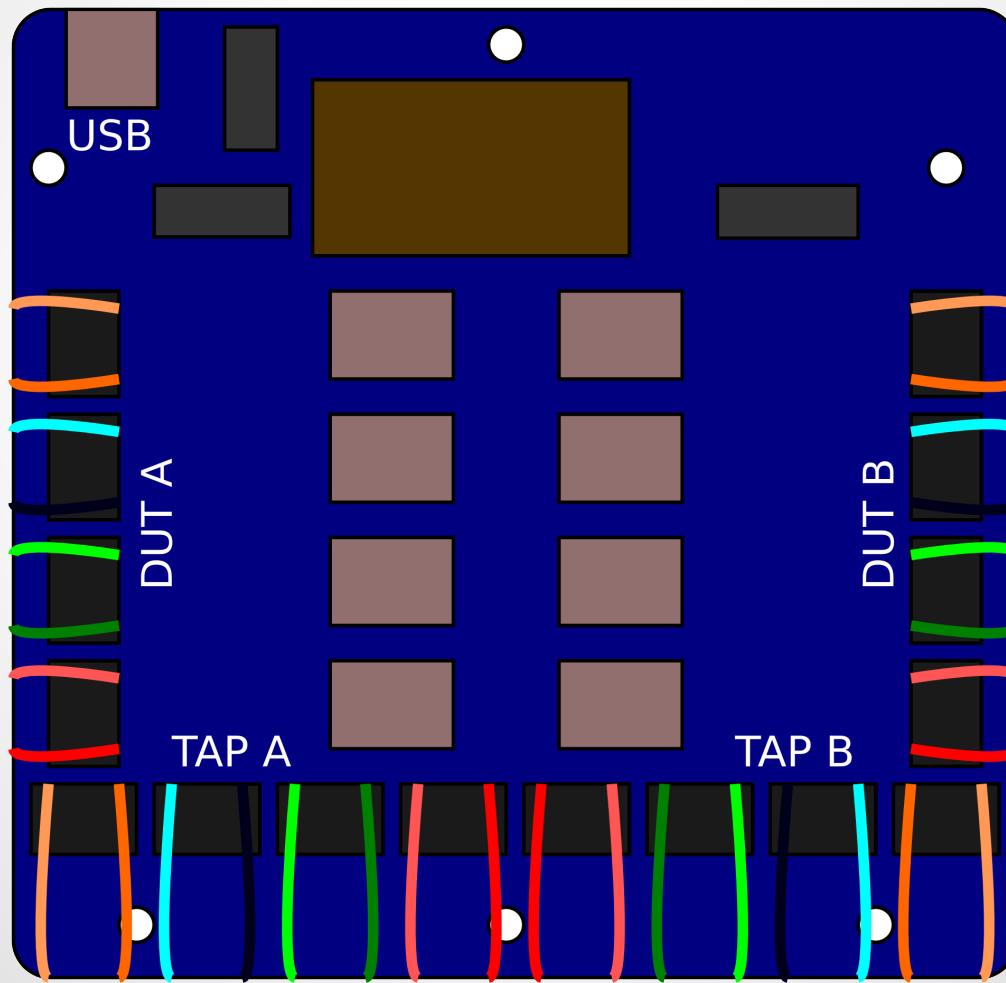
# Splicing Ethernet



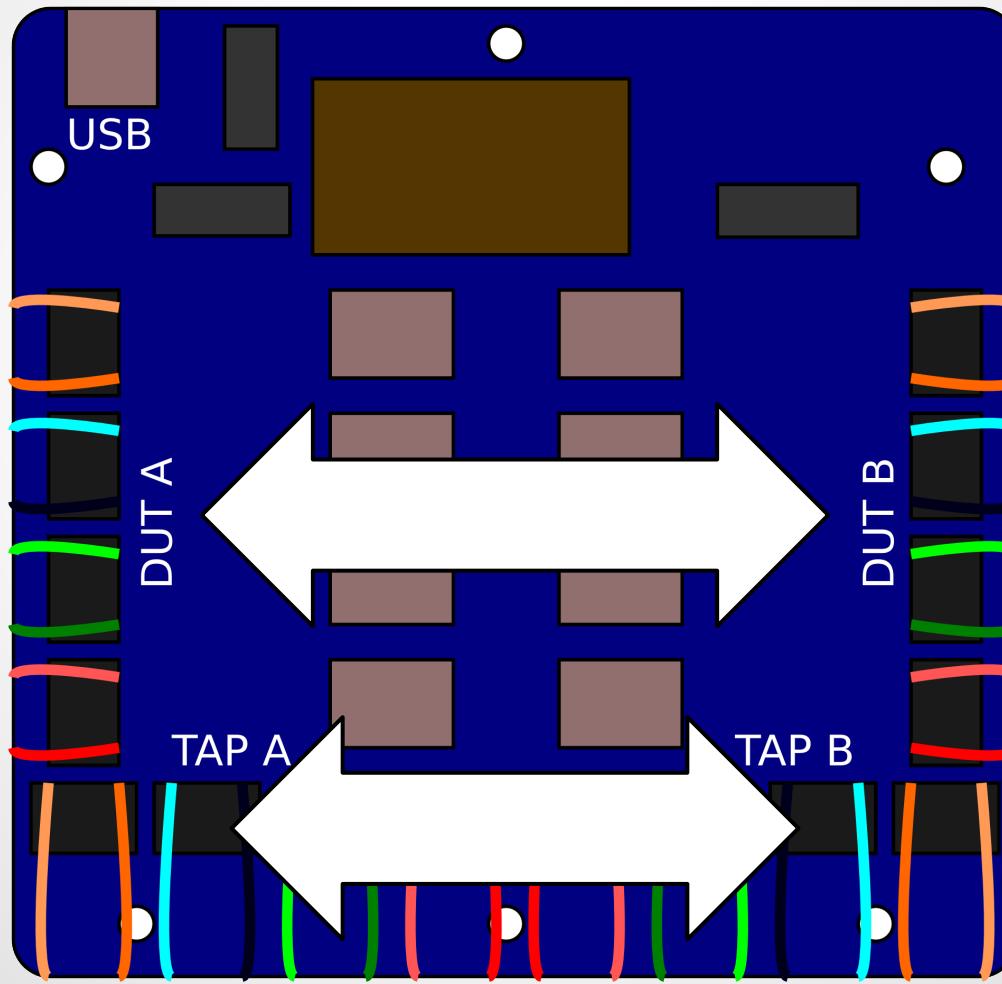
# Splicing Ethernet



# Splicing Ethernet



# Splicing Ethernet

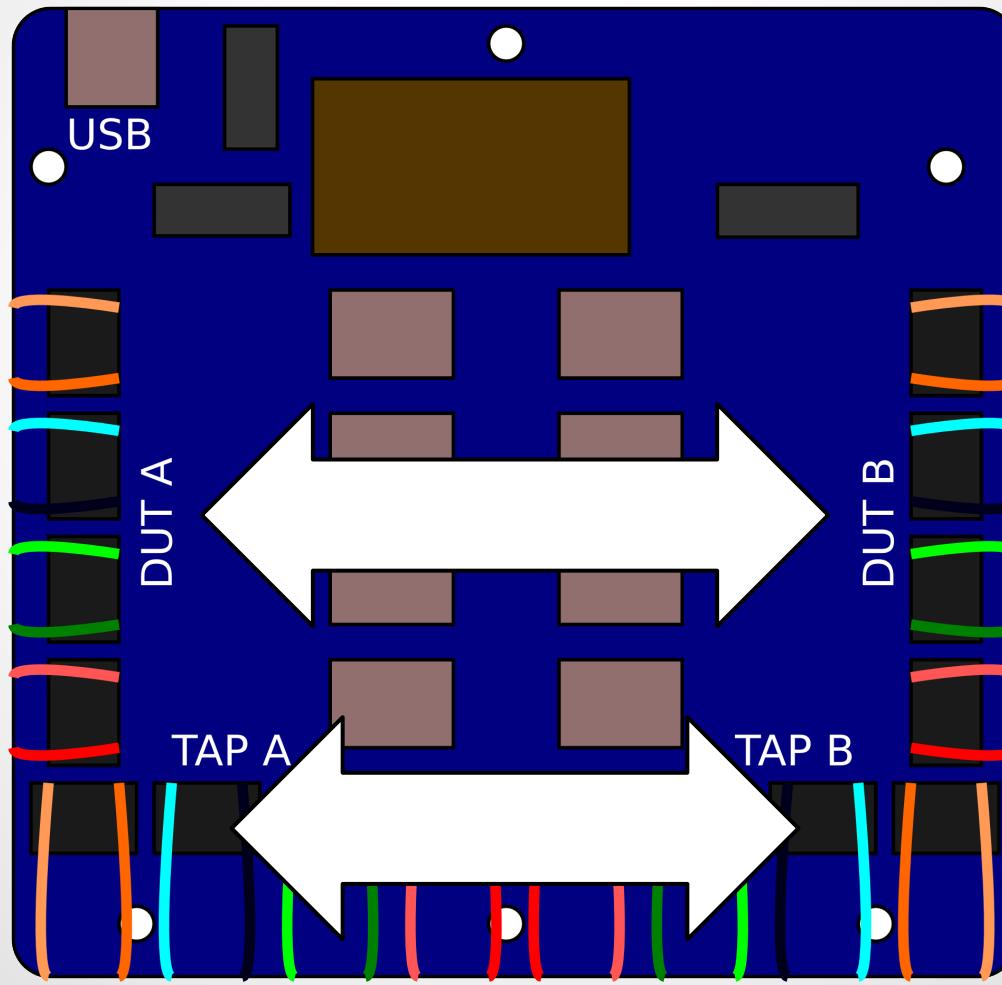


# Advanced Tap Board Features

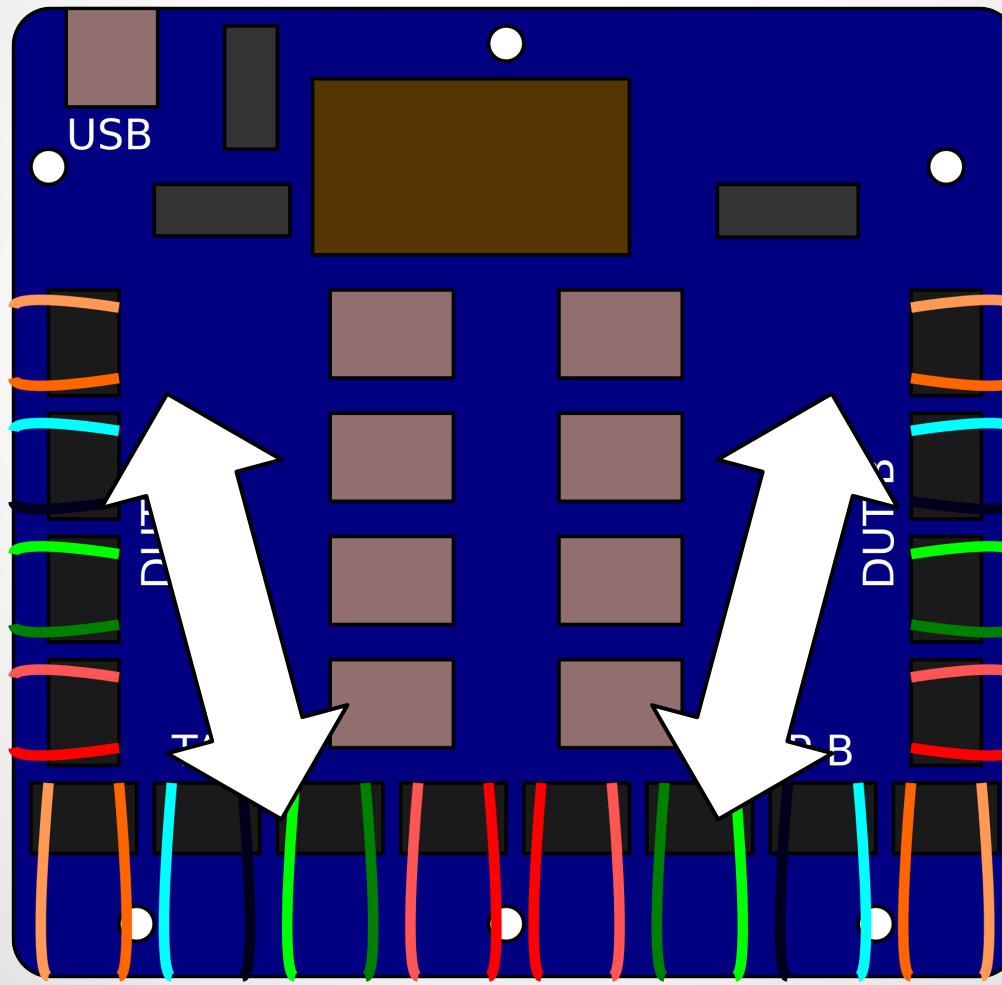


- Tamper-evident
- Fail-safe heartbeat
- Fail-safe power loss

# Switching to Active Tap



# Switching to Active Tap



# The Network Stack

```
Filter: Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
2148 12.725980000 192.168.1.10 192.168.1.2 RTP 1452 PT=DynamicRTP-Type-96, SSRC=0x0, Seq=2078, Time=16739174
2149 12.725989000 192.168.1.10 192.168.1.2 RTP 1452 PT=DynamicRTP-Type-96, SSRC=0x0, Seq=2079, Time=16739174
2150 12.725989000 192.168.1.10 192.168.1.2 RTP 1452 PT=DynamicRTP-Type-96, SSRC=0x0, Seq=2080, Time=16739174
2151 12.725990000 192.168.1.10 192.168.1.2 RTP 1452 PT=DynamicRTP-Type-96, SSRC=0x0, Seq=2081, Time=16739174
2152 12.725990000 192.168.1.10 192.168.1.2 RTP 687 PT=DynamicRTP-Type-96, SSRC=0x0, Seq=2082, Time=16739174, Mark
2153 12.765543000 192.168.1.10 192.168.1.2 RTP 1452 PT=DynamicRTP-Type-96, SSRC=0x0, Seq=2083, Time=16743264

►Frame 2150: 1452 bytes on wire (11616 bits), 1452 bytes captured (11616 bits) on interface 0
►Ethernet II, Src: Plus 87:df:ac (00:12:12:87:df:ac), Dst: Celink_02:ad:c8 (a0:c0:c8:02:ad:c8)
►Internet Protocol Version 4, Src: 192.168.1.10 (192.168.1.10), Dst: 192.168.1.2 (192.168.1.2)
▼User Datagram Protocol, Src Port: safetynetp (40000), Dst Port: 41522 (41522)
  Source port: safetynetp (40000)
  Destination port: 41522 (41522)
  Length: 1416
  ►Checksum: 0x8ccb [validation disabled]
▼Real-Time Transport Protocol
  ►[Stream setup by RTSP (frame 46)]
    10. .... = Version: RFC 1889 Version (2)
    ..0. .... = Padding: False
    ...0 .... = Extension: False
    .... 0000 = Contributing source identifiers count: 0
    0... .... = Marker: False
    Payload type: DynamicRTP-Type-96 (96)
    Sequence number: 2080
    [Extended sequence number: 67616]
    Timestamp: 16739174
    Synchronization Source identifier: 0x00000000 (0)
    Payload: 7c01ef6e953cd3b28b2d8552f3657bc869447e559394038...
```

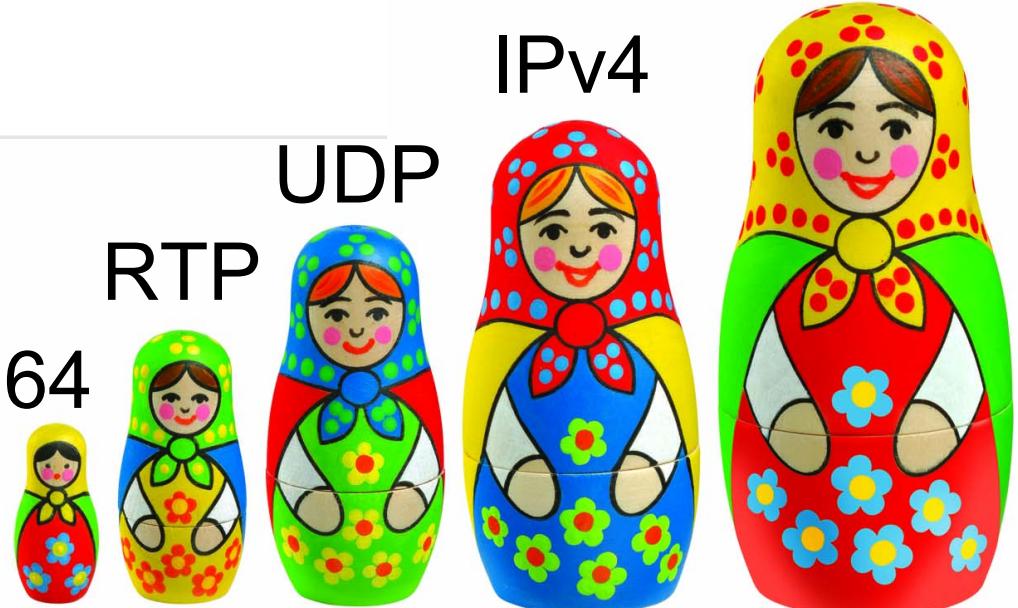
Ethernet

IPv4

UDP

RTP

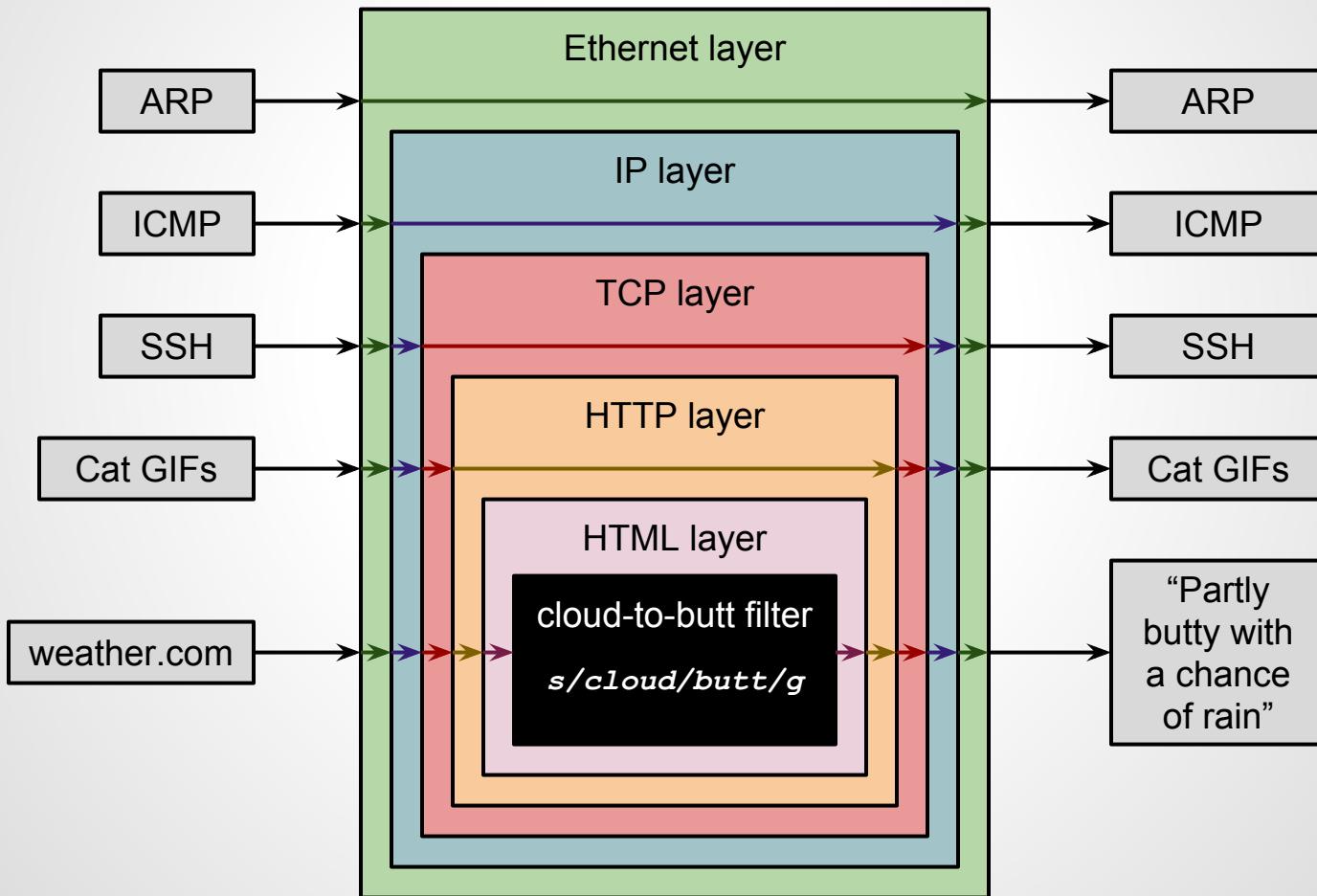
H.264



# *lens* Overview

- MitM-centric network stack in Python
  - Ethernet, IPv4, TCP, UDP, HTTP, RT\*P...
- Designed to be as transparent as possible
  - We need to be able to forge packets as necessary and have them blend in
- Allows for additional “layers” to filter data
  - Ex: turn a video stream into a loop

# *lens* Implementation



# Looping Video

- RTP: Real Time Protocol
  - RTSP: Streaming session; TCP
  - RTCP: Codec information, statistics; UDP\*
  - RTP: Video data; UDP\*
- **ffmpeg** solves all of your (video) problems
  - Looping, masks, transforms, and more!

# Looping Video

1. Read video stream from camera over RTP
2. Create new stream using `ffmpeg`
3. Forge packets from camera of new stream
4. ???
5. Profit!

# Potential Extensions

- HTTPS
  - Incredibly tricky to get right in embedded systems
- USB
- HDMI

# Check it out!

- Hardware
- Firmware
- Mechanical
- Code

<http://github.com/ervanalb/lens>