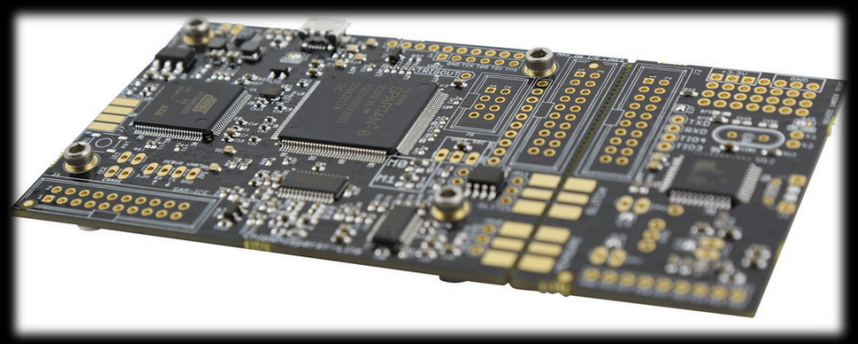
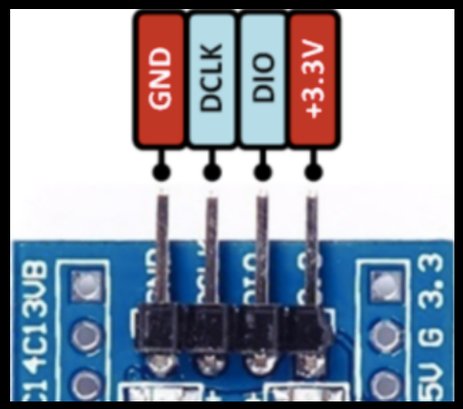


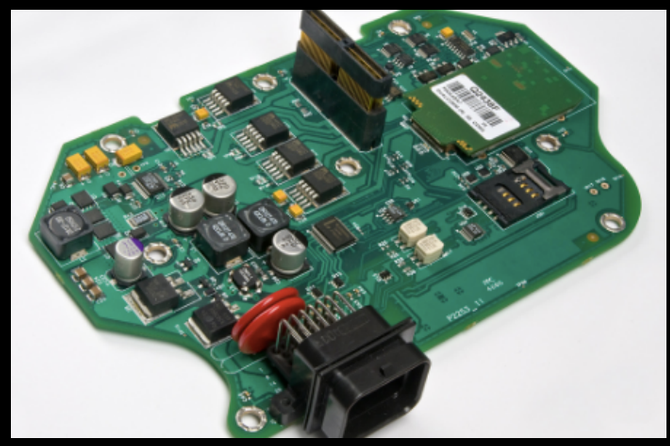
Beginner Electronics for Hardware Hacking

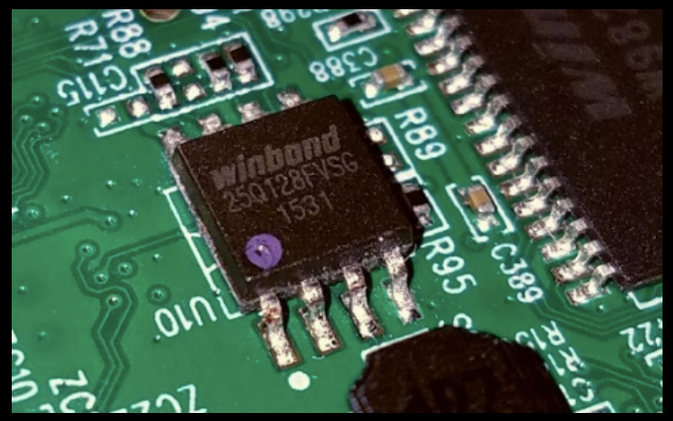
Or why physical access breaks most stuff…



What This Talk Is

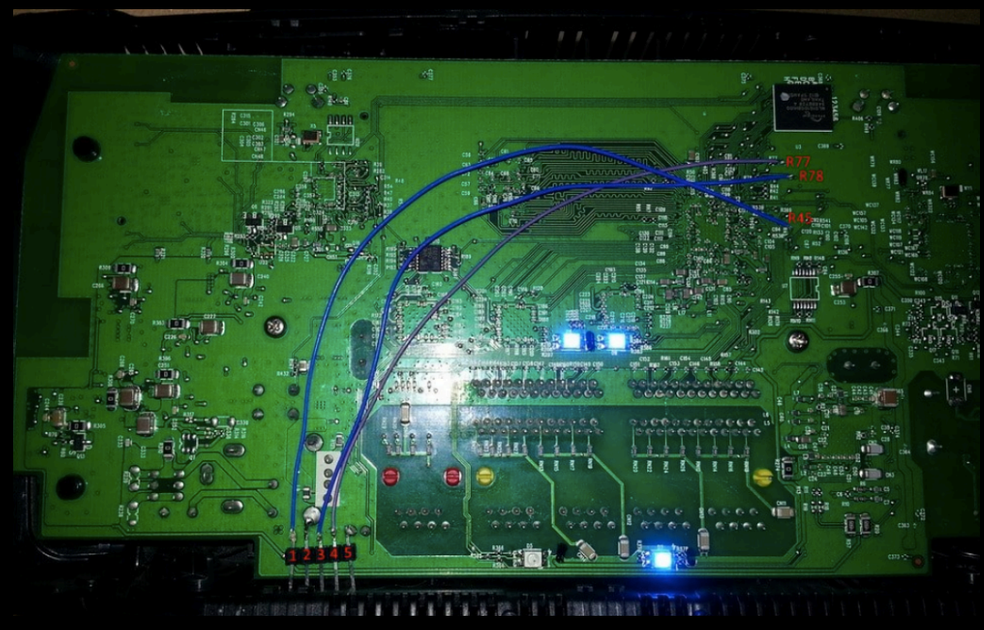
- Very brief intro to electronics
    - Examples of hardware hacking techniques
- Also look at some things to look out for
- Focus on embedded / cheap devices
- Apologies to anyone with experience





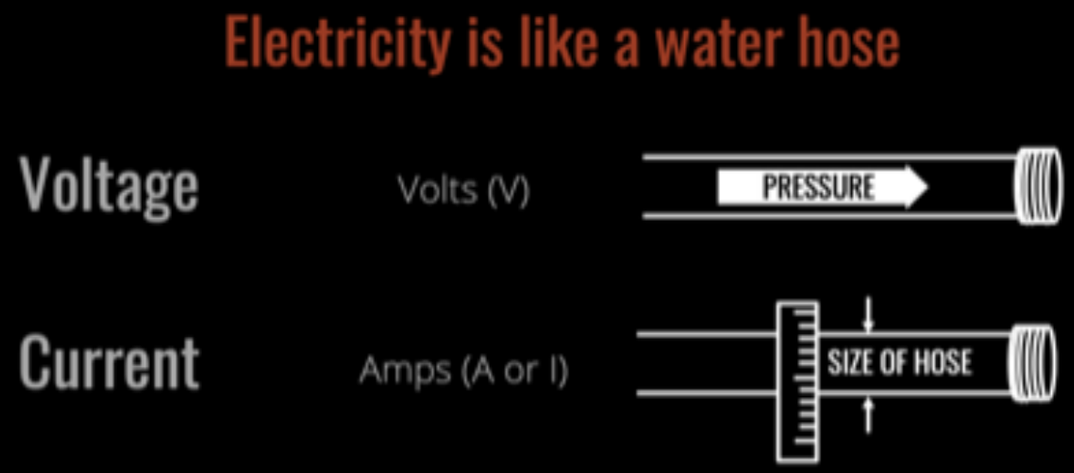
Motivation

- Hardware is commonly overlooked
- Its not as difficult as it can seem
- Physical access pwns

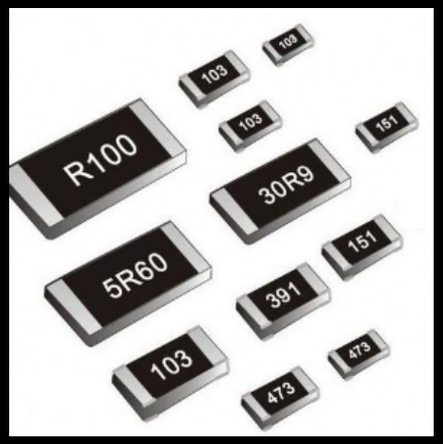


Terminology

- Voltage is a measure of difference in electrical energy
  - Measured in Volts (V)
- Current is a measure of *flow* of electrical charge in a circuit
  - Measured in amperes (amps, A)
  - Also denoted I (e.g. I_{max} for max current through a component)
- V_{cc} (V_{dd}) - Supply voltage
- GND - Ground

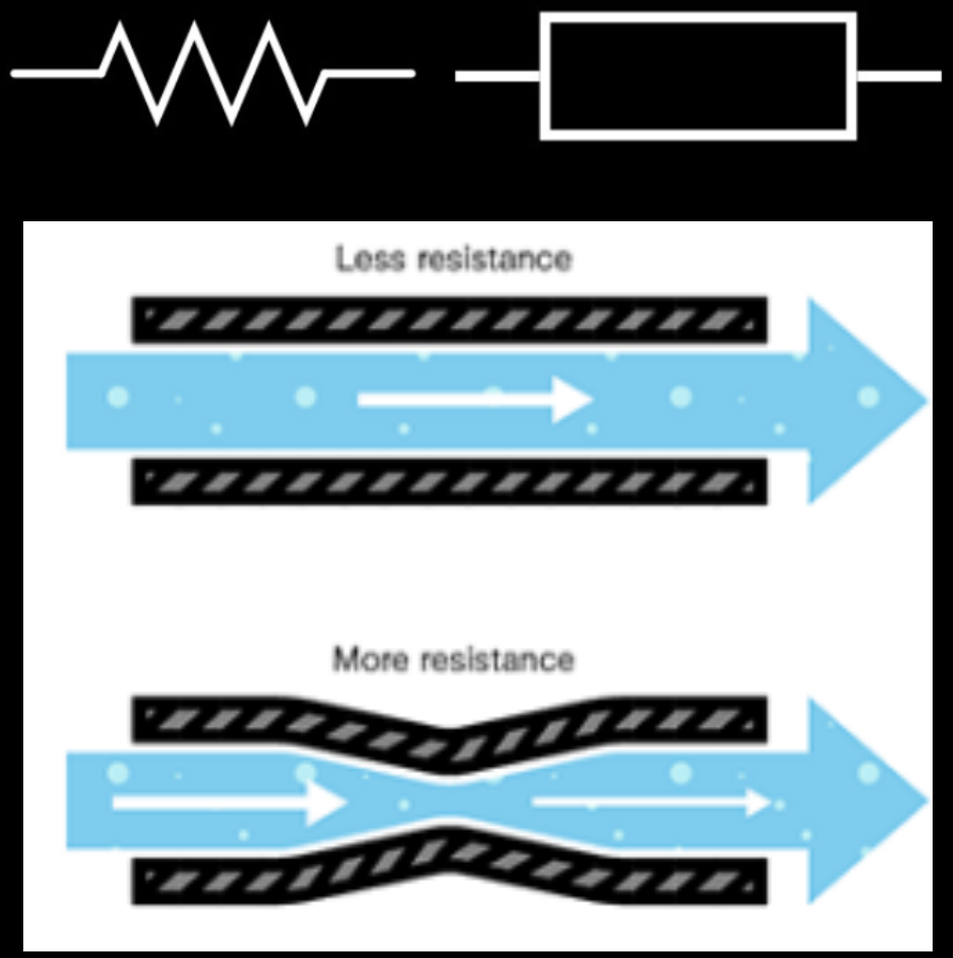


Component I



Resistors

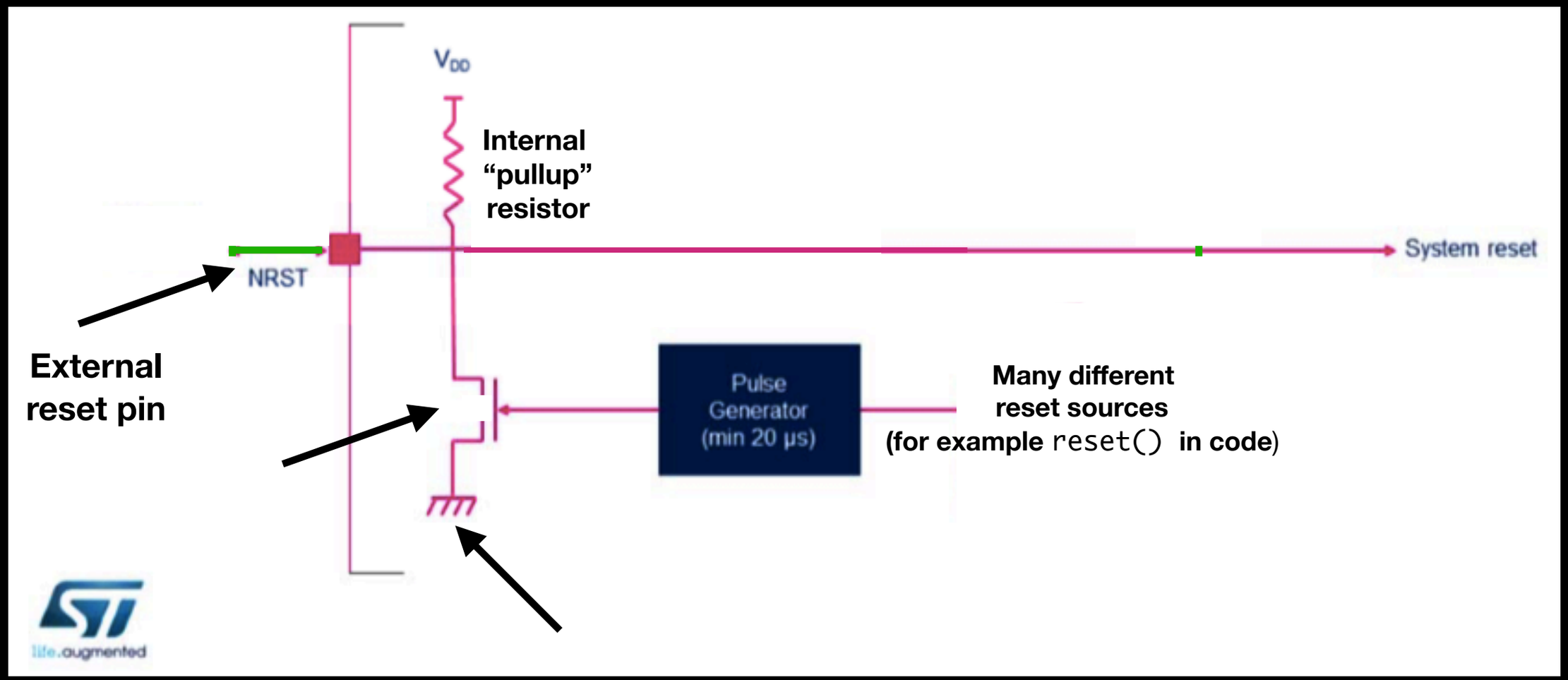
- Resist current
- Most common component
- Many uses
  - Voltage divider
  - Current limiting
  - Pullups / pulldowns (assert boolean values)



Example - STM32 Reset Pin to Stop CPU Resetting

- STM32 - very common line of ARM microcontrollers - cheap and ubiquitous
- Why stop resets? - Sometimes devices will panic reset
  - If they are in an undefined state
  - Intrusion detection
- Stopping resets can be useful. How?

Example - STM32 Reset Pin to Stop CPU Resetting



https://www.st.com/content/ccc/resource/training/technical/product_training/group0/c8/9e/ff/ac/7a/75/42/d1/STM32F7_System_RCC/files/STM32F7_System_RCC.pdf/_jcr_content/translations/en.STM32F7_System_RCC.pdf

Example - STM32 Reset Pin to Stop CPU Resetting

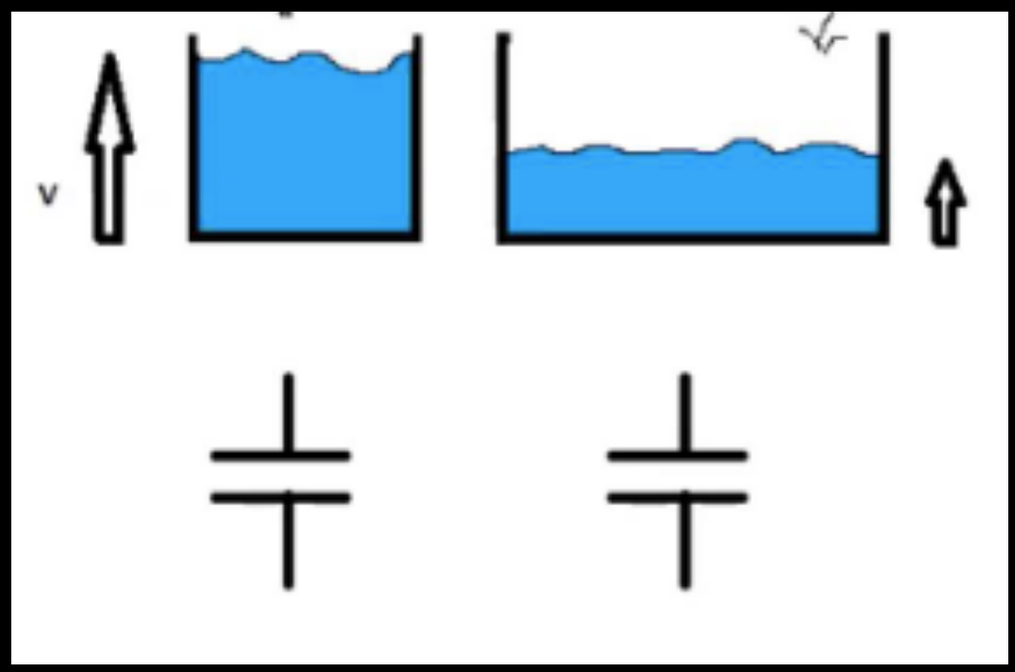
- Clamping external reset pin high
  - Disables resets
    - Allows for arbitrarily long debug sessions
- These techniques are a means to an end
  - Dumping flash
  - Changing internal state

Component II

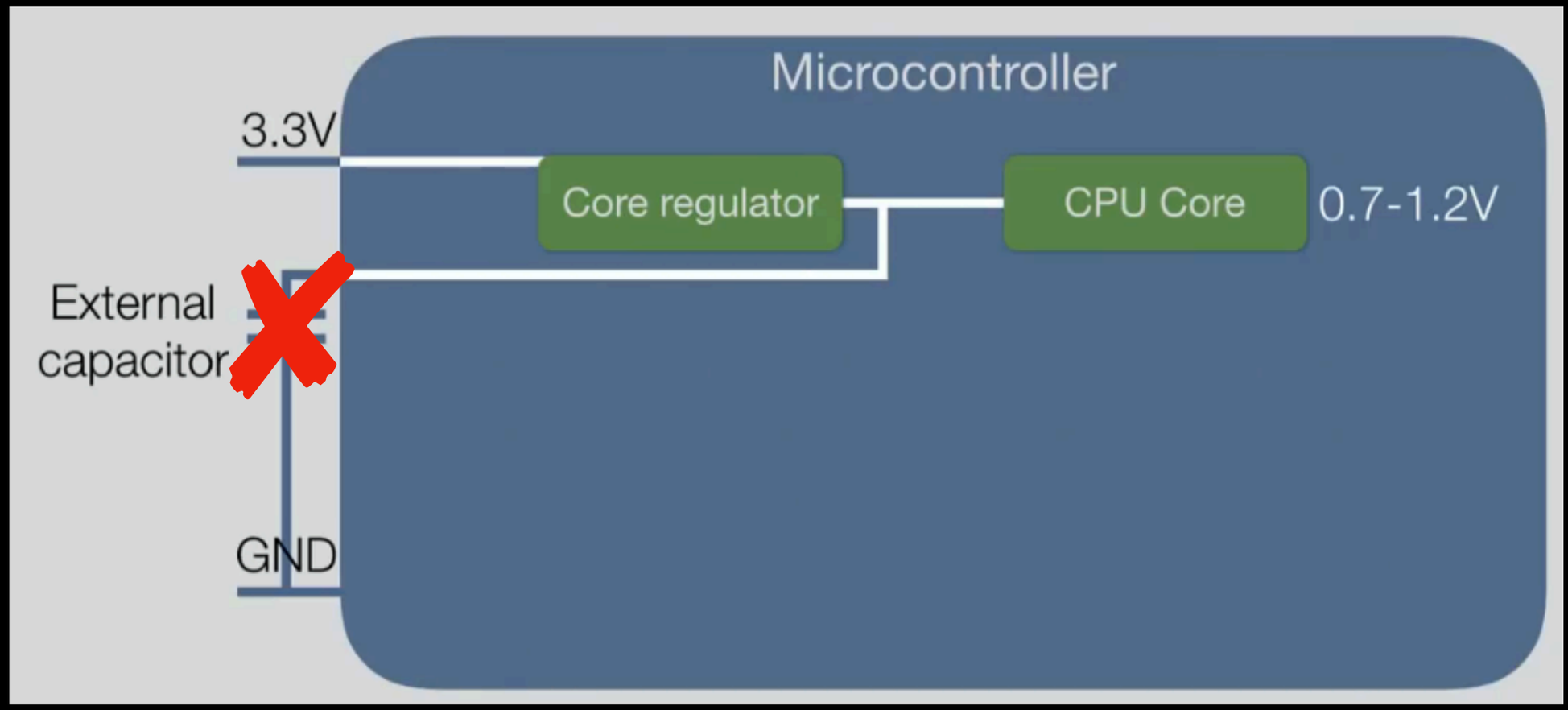
Capacitors

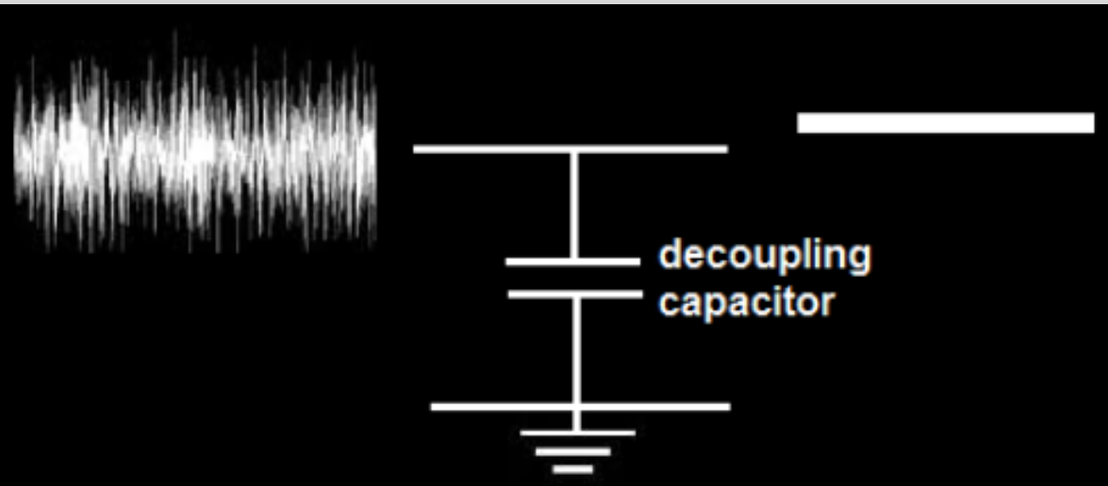
- Store charge

- Resist changes in voltage
- Used for decoupling (smoothing noise in voltage)
- Second most common component



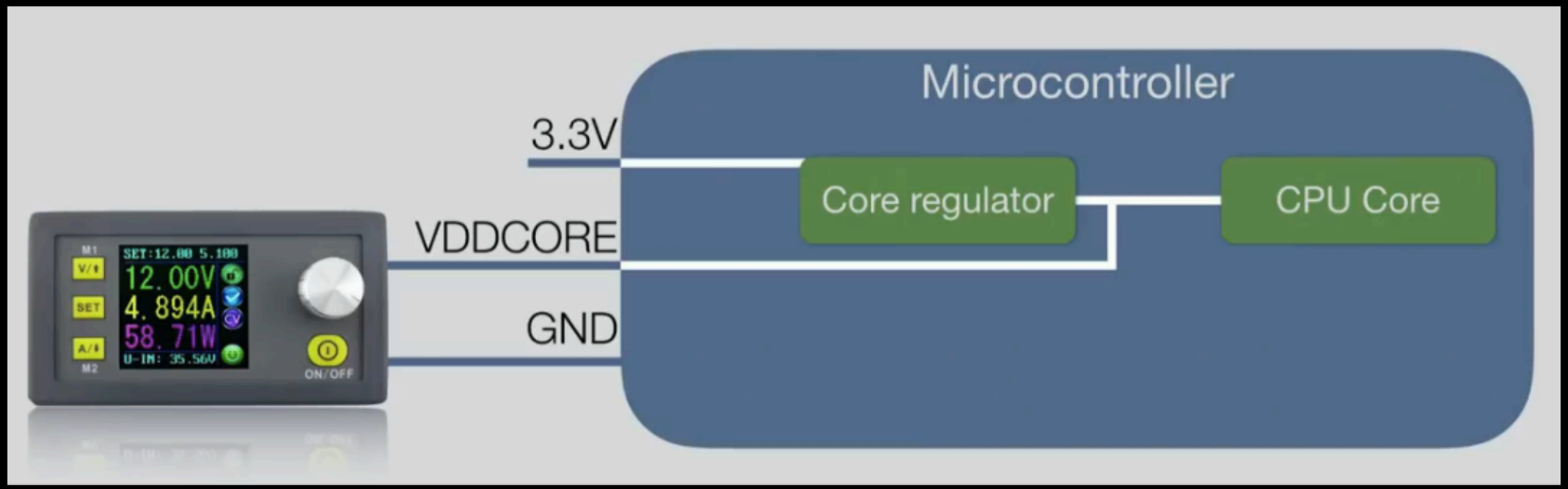
Example - CPU Glitching





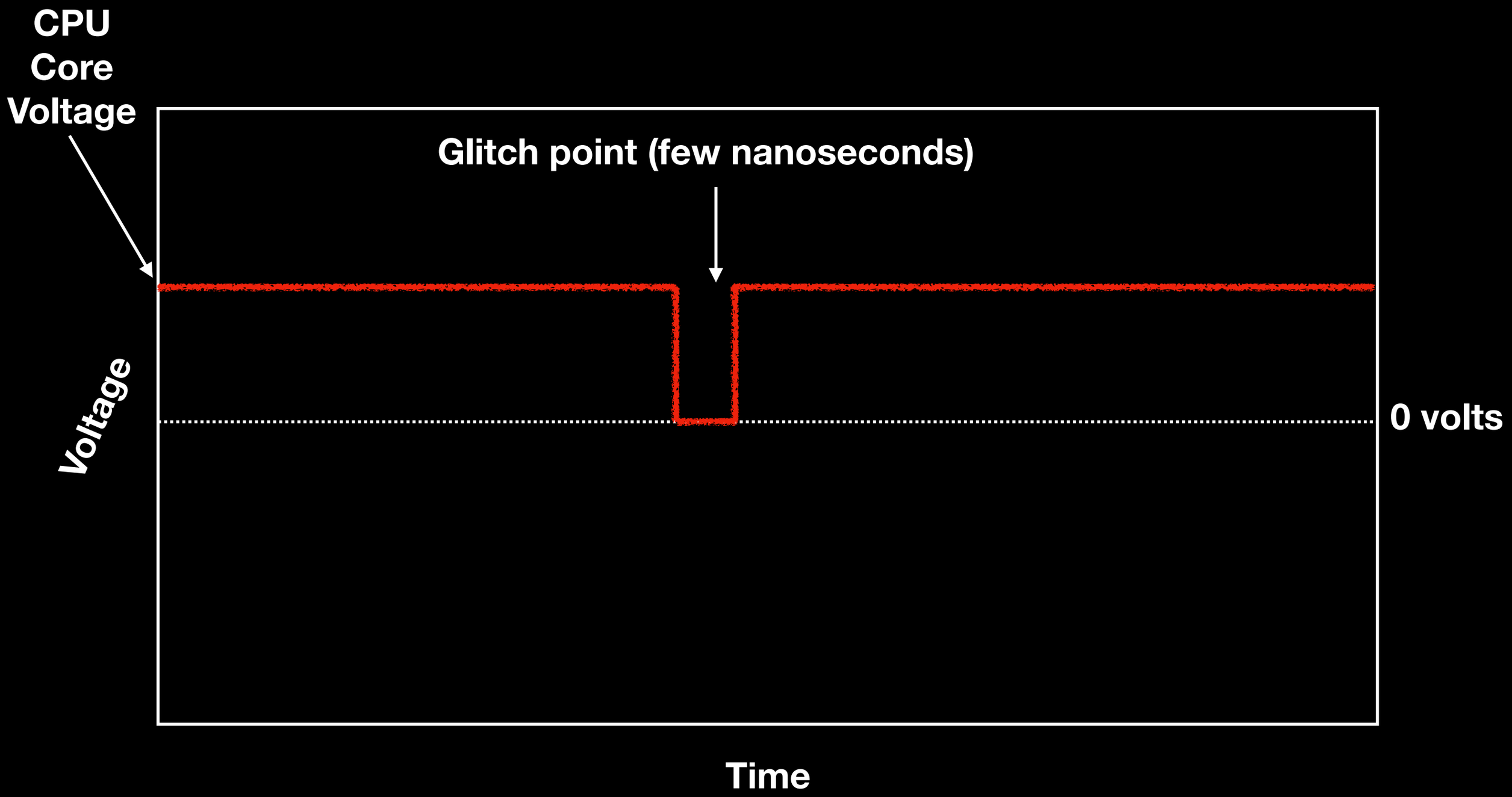
https://www.youtube.com/watch?v=4u6BAH8mEDw

Example - CPU Glitching



https://www.youtube.com/watch?v=4u6BAH8mEDw

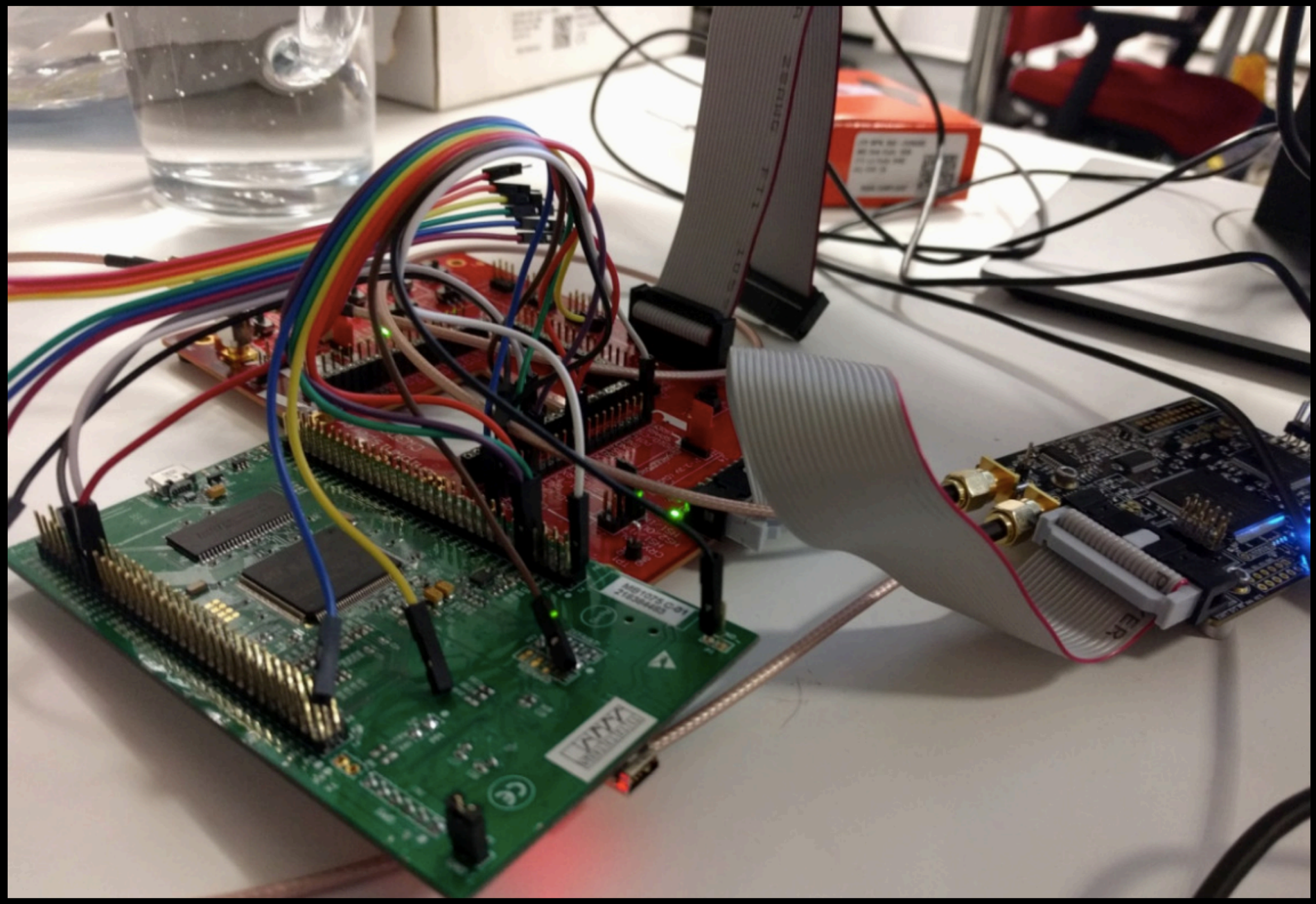
Example - CPU Glitching



Example - CPU Glitching

```
int main(){
  Config config = get_stored_config();
  bool DEBUG = true;

  if (config.isDebugEnabled == false){ //Instruction level: cmp reg 0x0
    DEBUG = false;
  }

  ...

  if (DEBUG){
    print("Super secret");
  }
}
```

Example - CPU Glitching



https://www.youtube.com/watch?v=4u6BAH8mEDw

Example - CPU Glitching

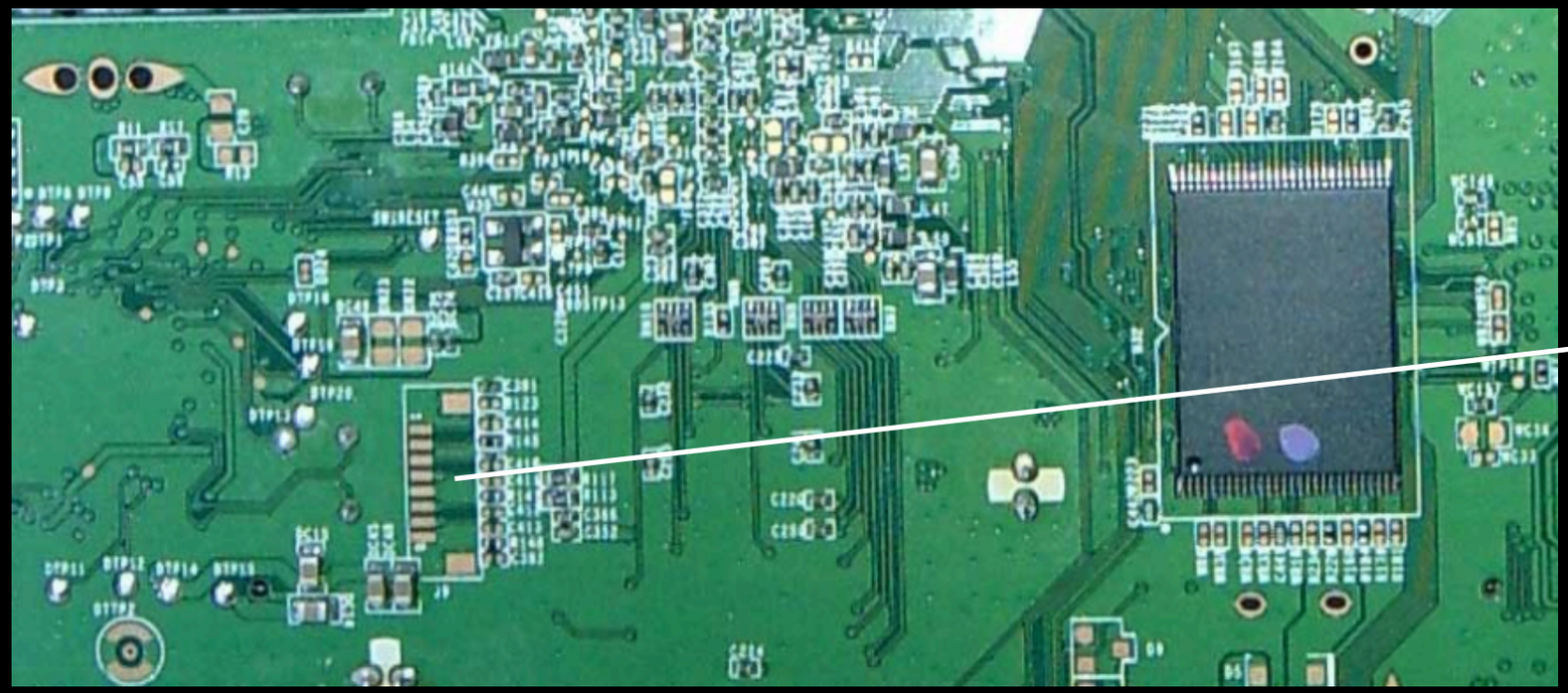
- Is very hard to get the right timing
- You need some sort of success indicator
    - Perhaps debug session could be used?
- Means to an end
    - Skip debug disable instruction to allow debugging access
    - Cause undefined behaviour - dump flash / encryption keys
- Mitigations: brown-out / glitch detector

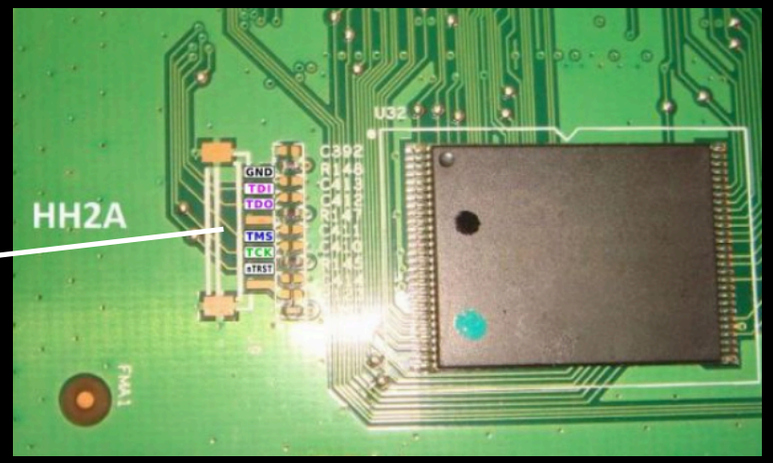
Things to Look Out For

Debug Ports

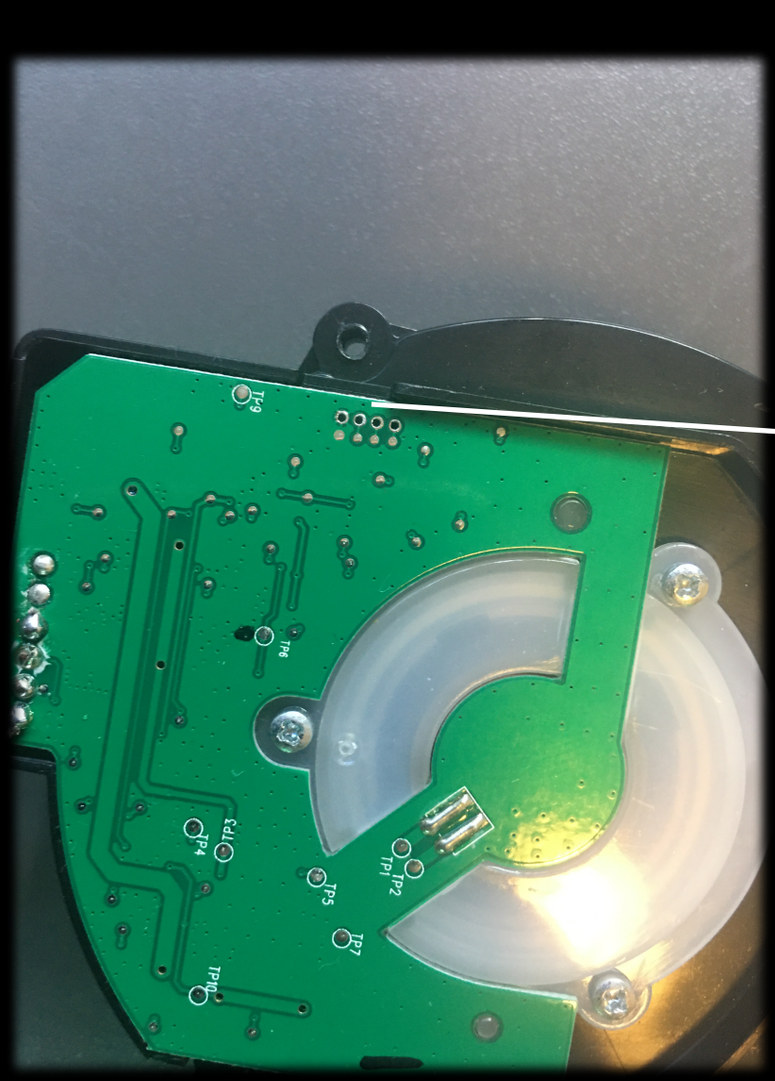
- OEMs regularly leave low hanging fruits
  - JTAG ports
  - Serial ports
  - SWD ports (STM microcontrollers)
- Easy access for debugging
  - Halt CPU / step through instructions
  - Set breakpoints
  - See memory / register contents
  - Dump flash
- Left in for production

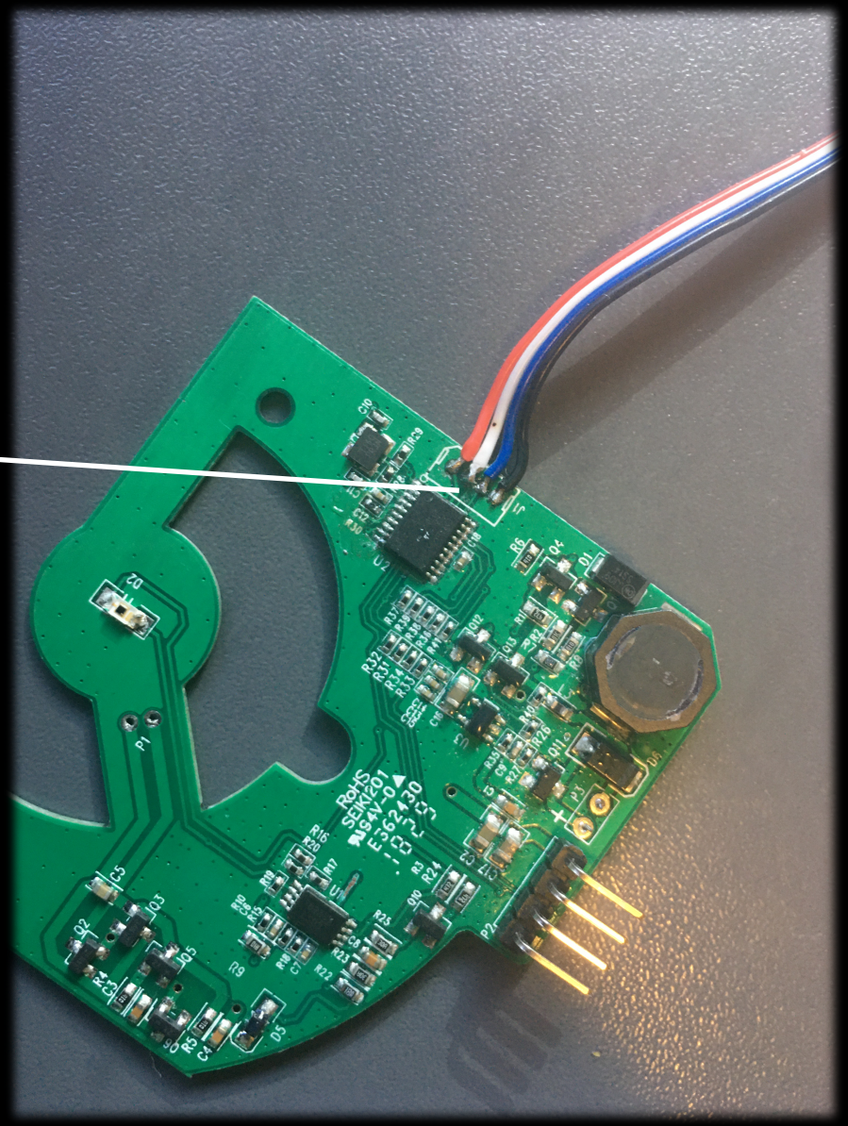
Things to Look Out For - JTAG



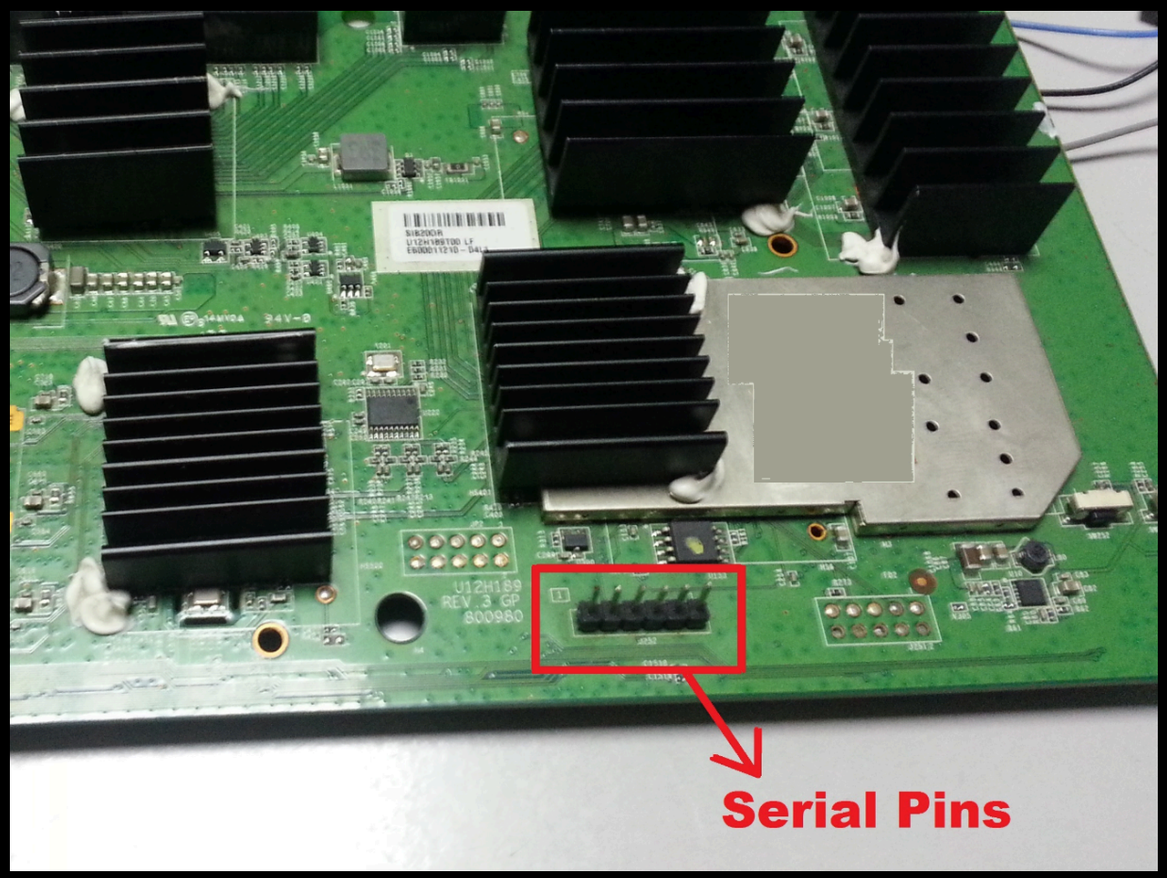


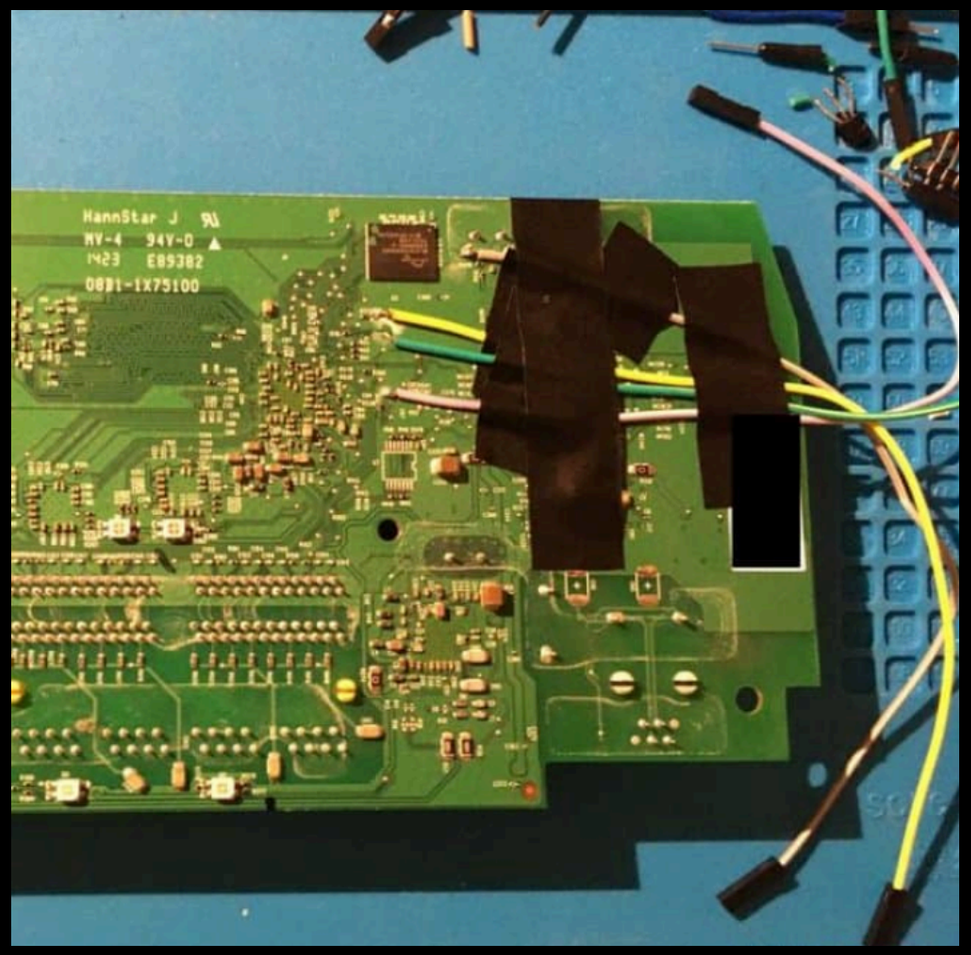
Things to Look Out For - SWD





Things to Look Out For - Serial Ports





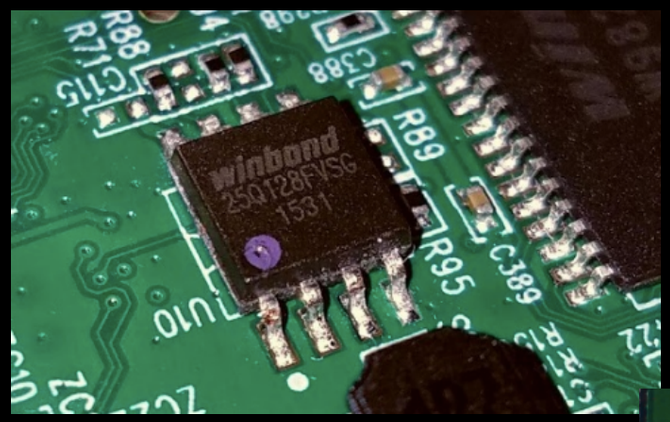
Things to Look Out For - Serial Ports

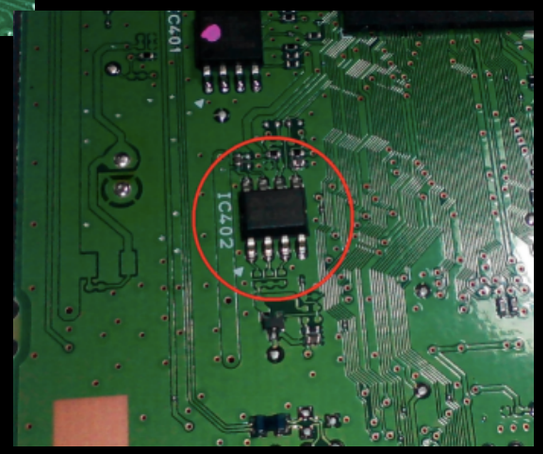
```
DDR2 test..
VCDL test.

Decompressing Bootloader.............................

Version BL: 1.0.2
Reading cpu info..........bcm96358 tp0 revision 1
MIPS is in Big endian mode
Icache : 32Kb            Icachelinesize : 16 bytes
Dcache : 16Kb            Dcachelinesize : 16 bytes
BCM config reg CP0 : e30e1006
                Instruction cache enabled
                Data cache enabled
                Data cache is nonblocking
C0_CONFIG reg = 80008083
                standard TLB config
                Cacheable, write-back

Multicore enable; Booting Linux kernel

pfuncjmp = A0001840
Reading cpu info..........bcm96358 tp0 revision 1
MIPS is in Big endian mode
Icache : 16Kb            Icachelinesize : 16 bytes
Dcache : 16Kb            Dcachelinesize : 16 bytes
BCM config reg CP0 : 230e1006
                Data cache is nonblocking
C0_CONFIG reg = 80008082
                standard TLB config
                Noncacheable

JTAG select tp1
BOOTING THE THOMSON LINUX KERNEL
..
..
Press enter to start serial prompt.
root@homehub [~/] #
```

- Embedded Linux devices
- Gives **root** terminal prompt
  - Usually without password
  - Common password / find online
- Means to an end
  - Complete control of device - install software backdoor etc.
  - Dump RAM / flash
    - Recover passwords, encryption keys

Things to Look Out For - EEPROMs

- Electronically Erasable Programmable (Read Only) Memory
- Used for many things
  - Configuration storage
  - Debug log storage (Tesla)
  - Storing filesystems
- Means to an end
  - Dump flash -> edit -> reflash
  - Recover user configuration (emails, passwords…)
  - Firmware analysis to find remote exploits





Security Hardening

- Modern, more expensive hardware is better
- OEMs can use easy techniques
  - Disabling debug
  - Removing serial ports from circuit boards
- Hardware trip switches render devices inoperable
- Chips designed for security - chain of trust from chip manufacturer to OEM

Takeaways

- (Cheap) hardware isn’t designed with security in mind
- There are many approaches to hardware hacking
- OEMs want to save money / time
- Secure hardware is predictably harder to hack

Thanks :)