Physical Attacks

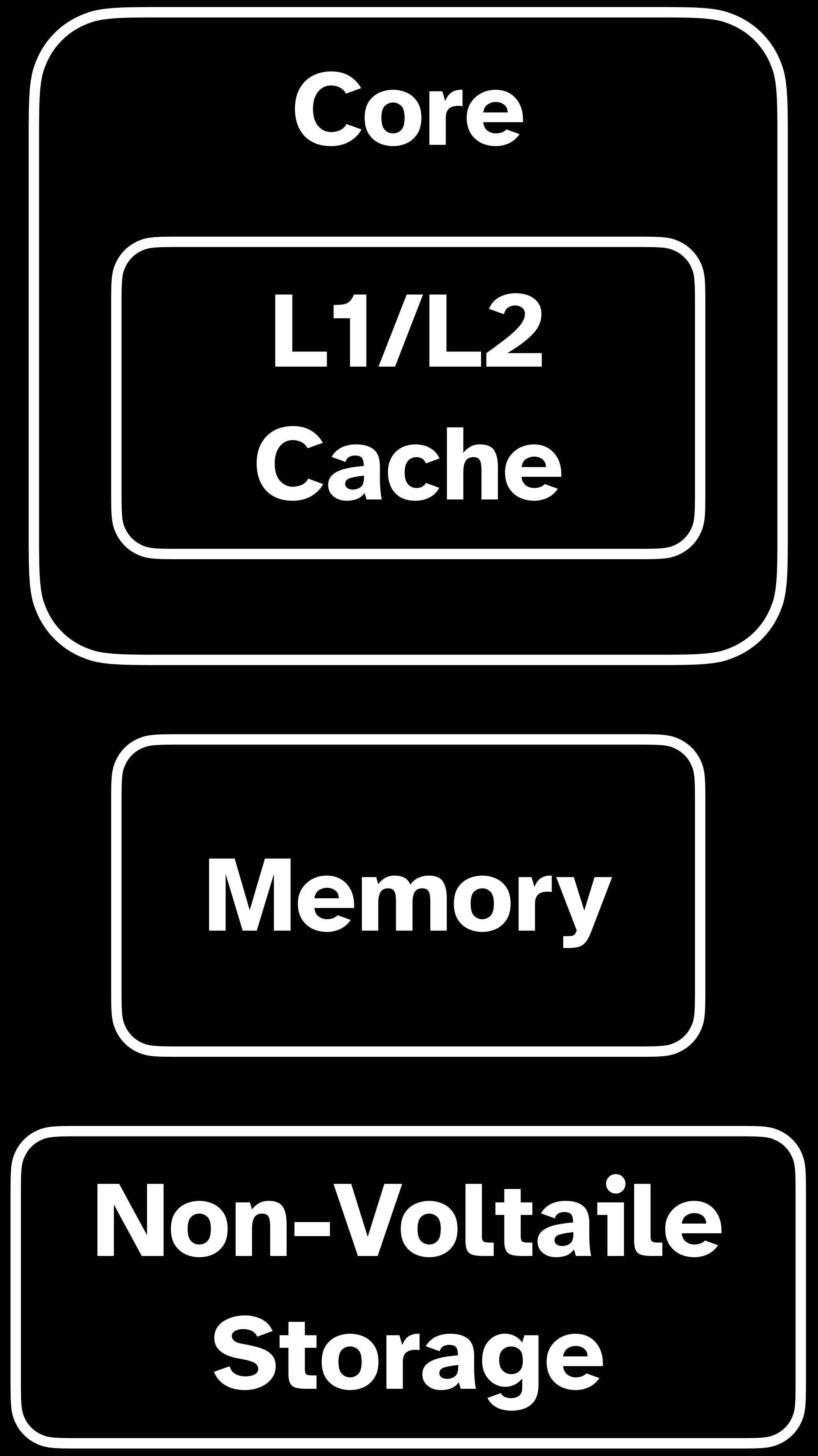
MIT Secure HW Design Spring 2023

Mengjia Yan & Joseph Ravichandran

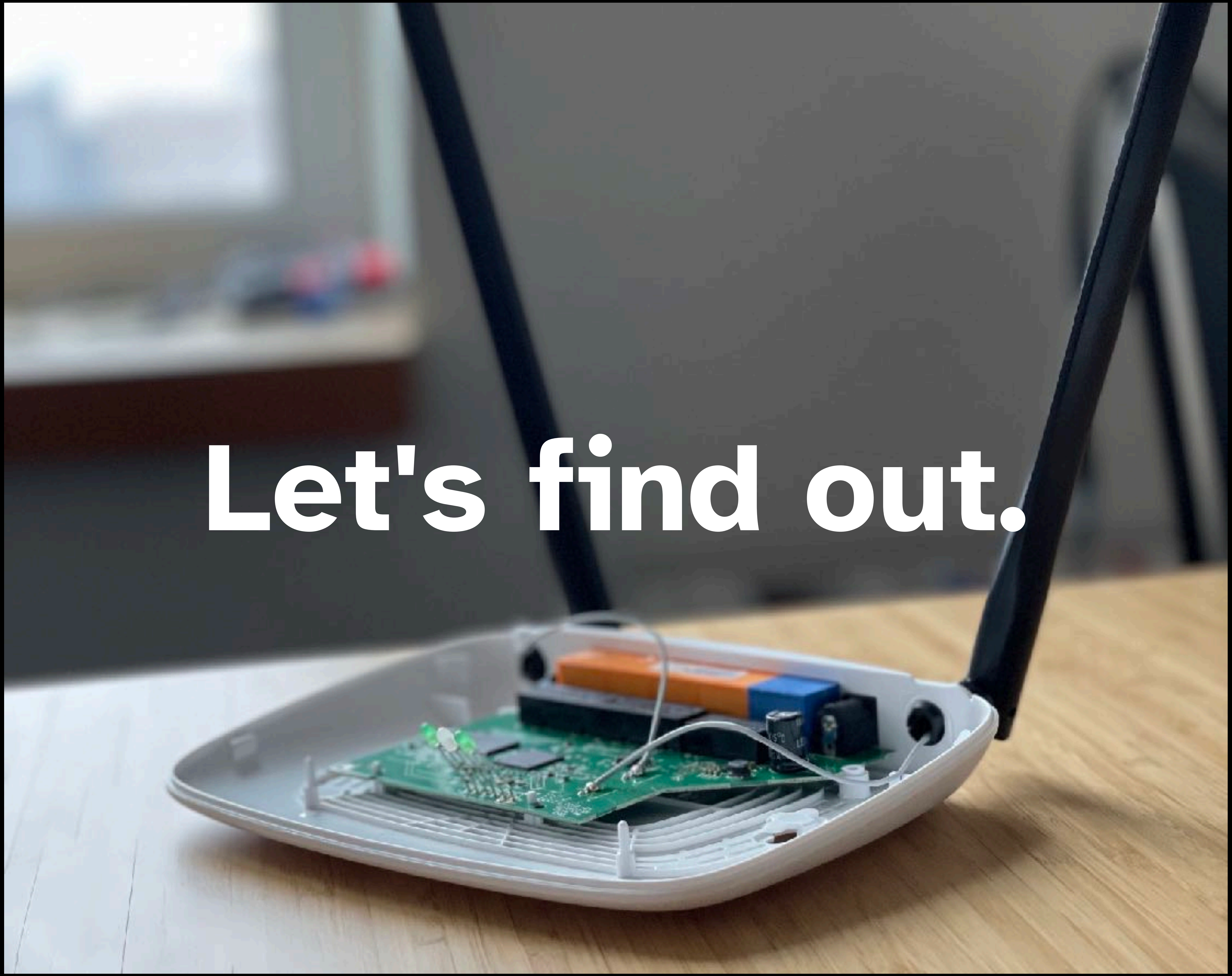
Image: Proto G Engineering, "Oscilloscope Art"

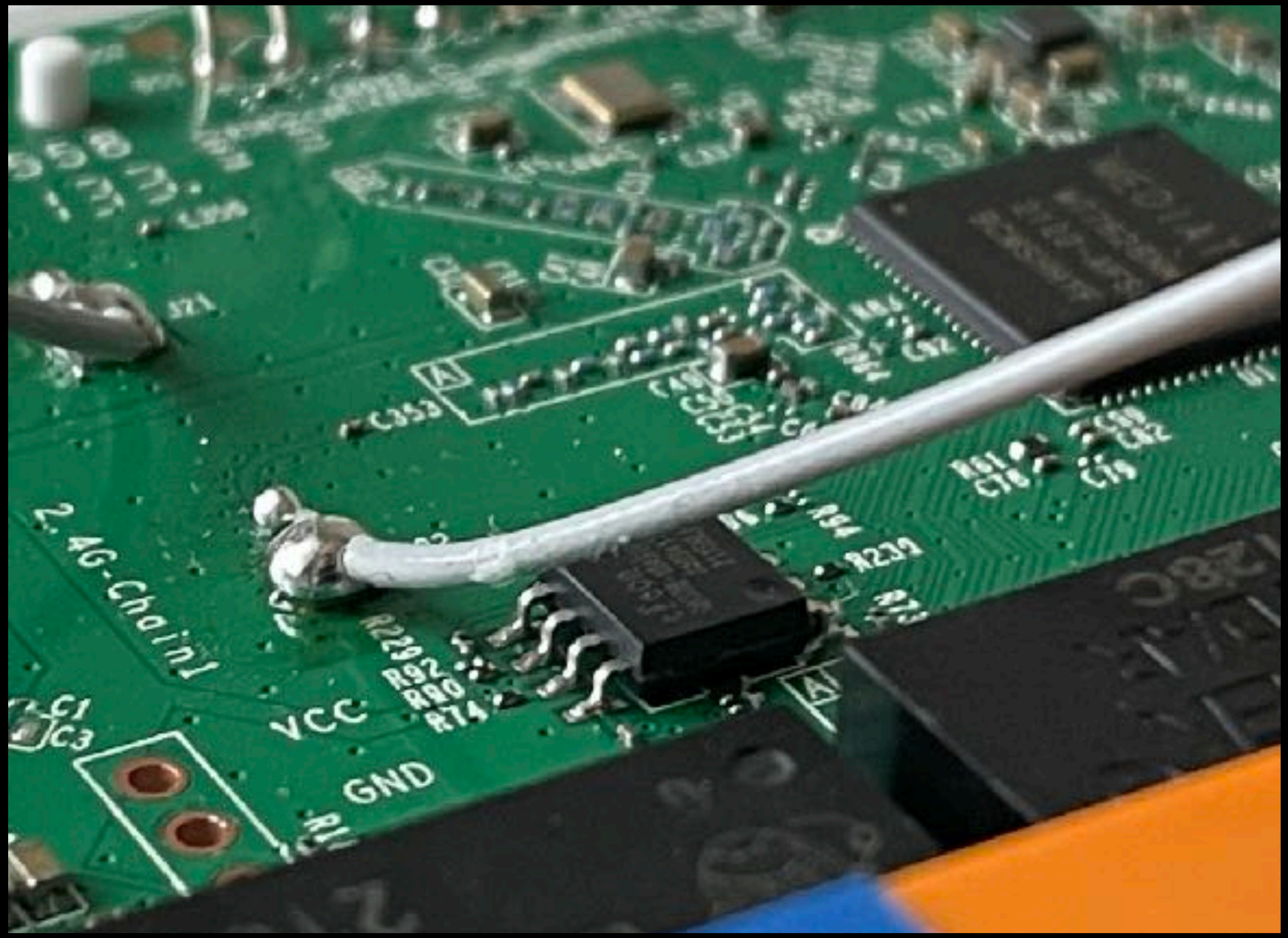
What's a Computer?

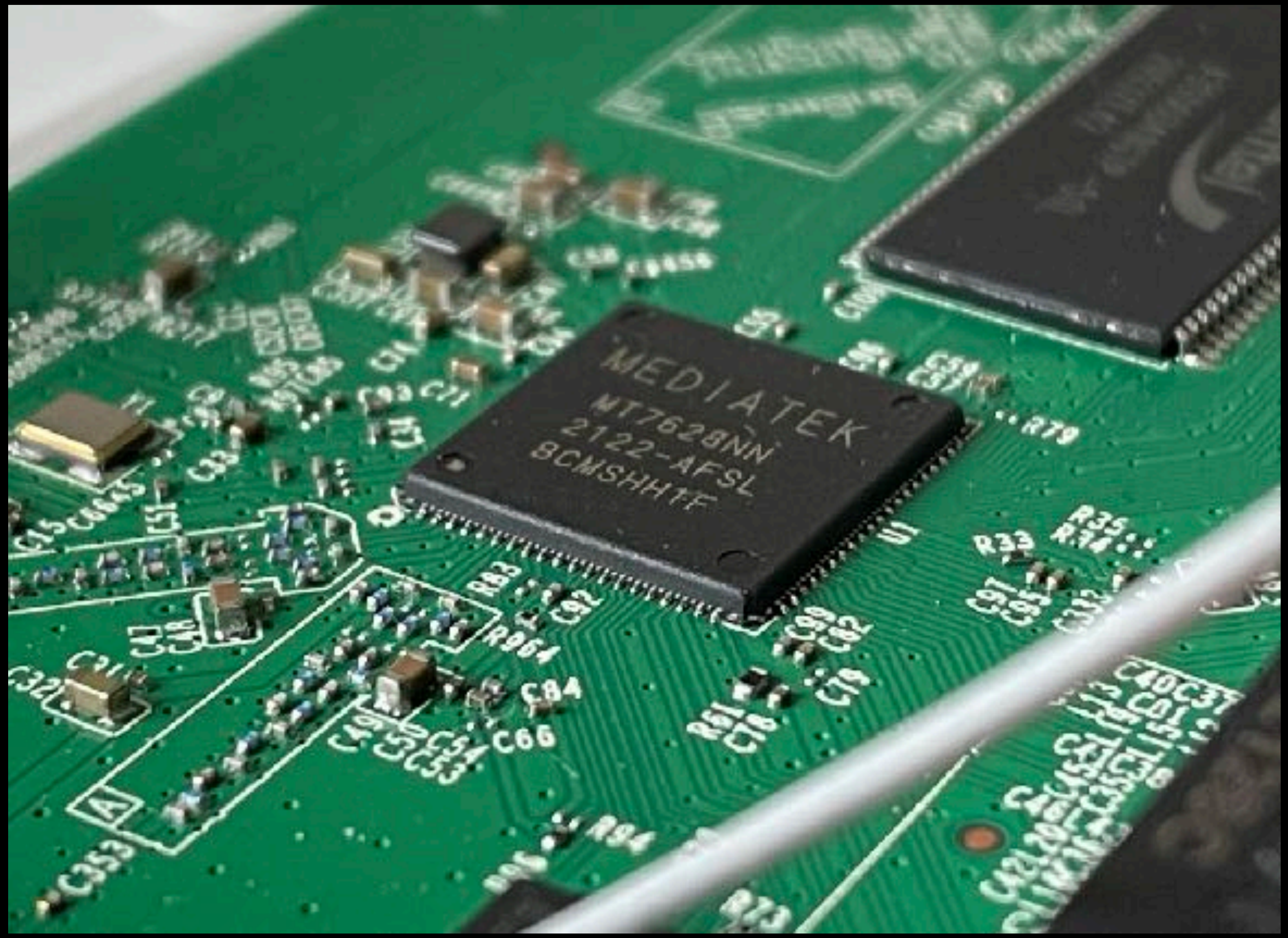
What's a Computer?




What's Inside?

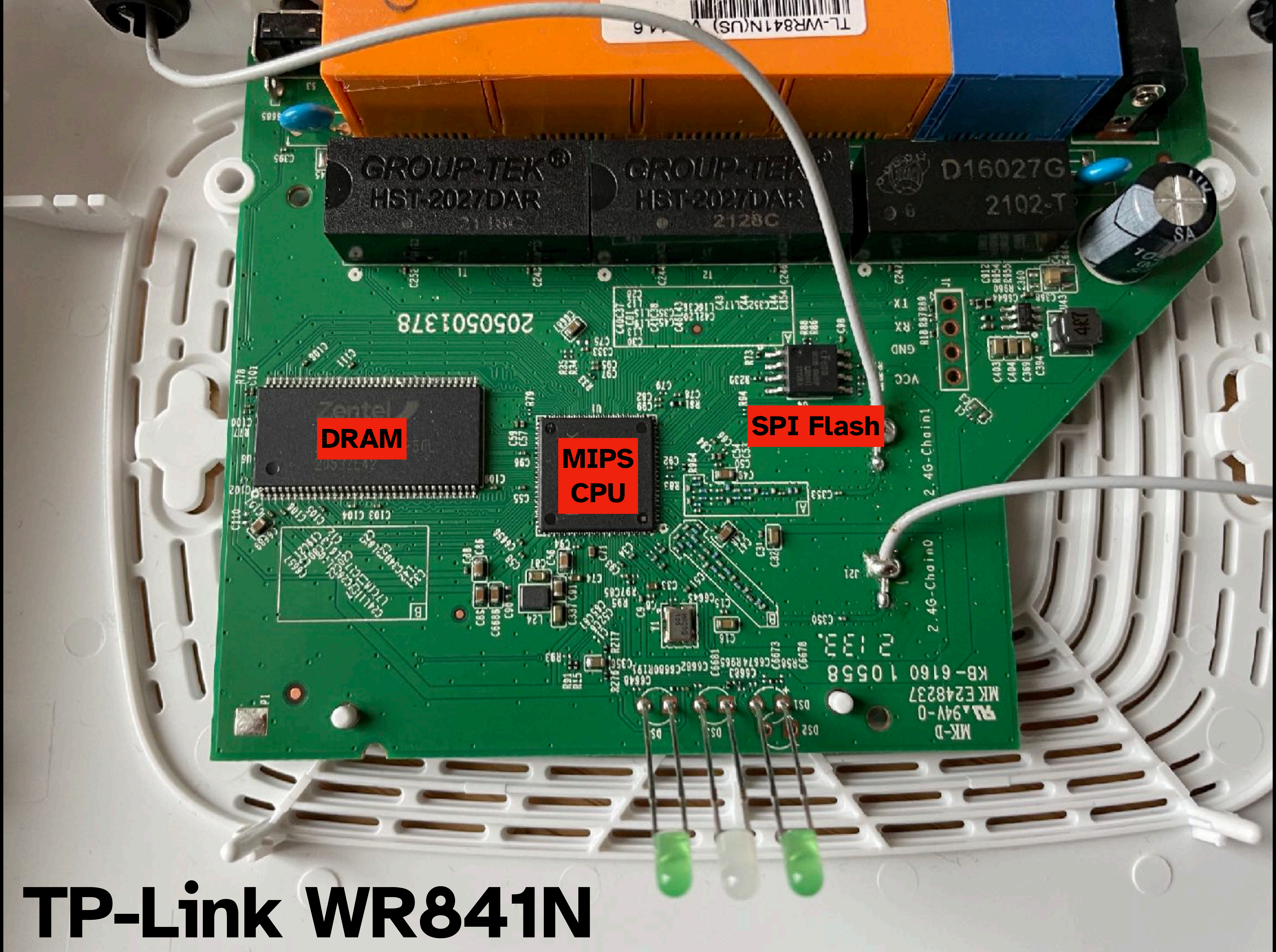

Let's find out.

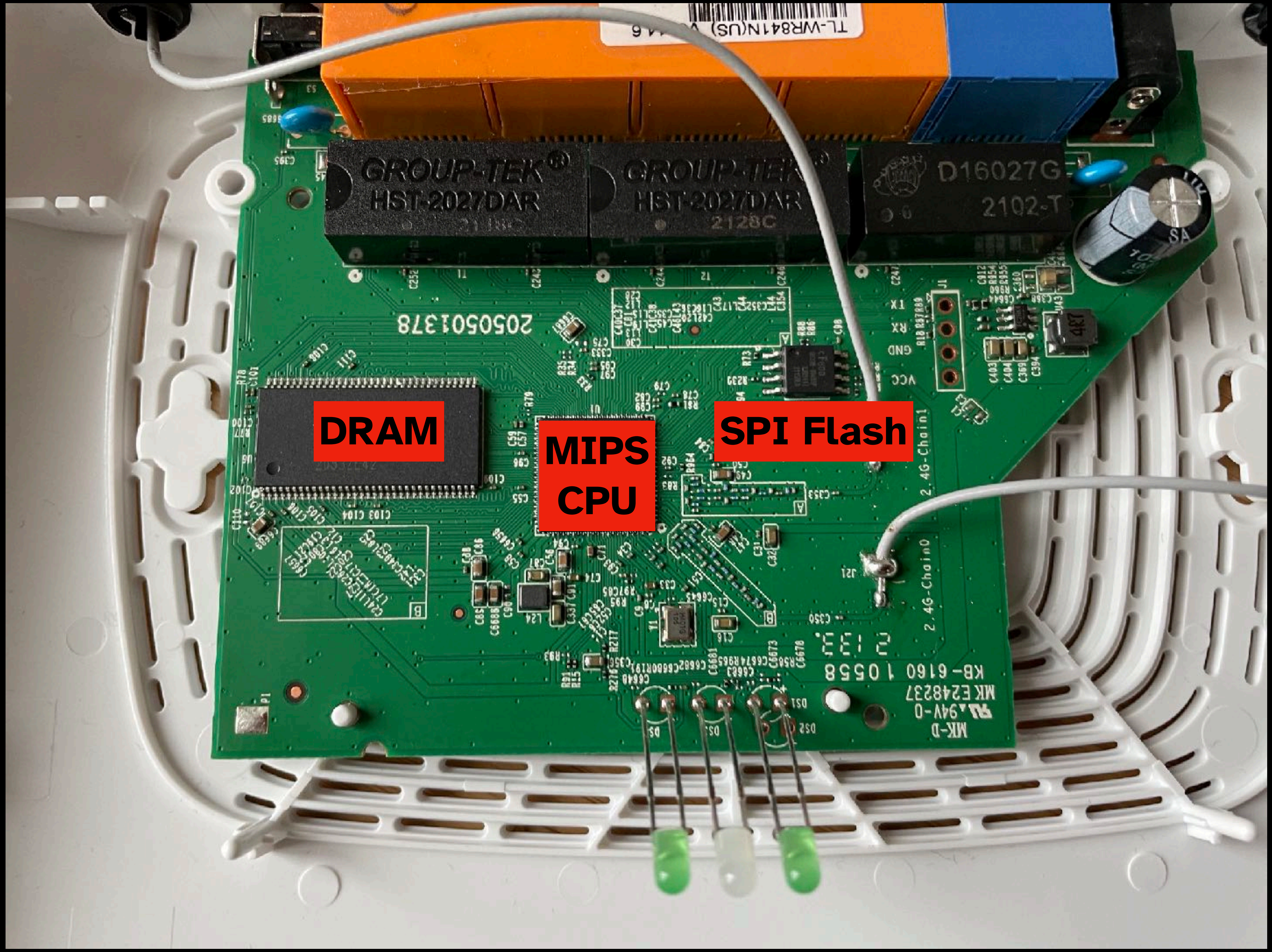

2.4G-Chain1
VCC
GND


MEDIATEK
MT7628NN
2122-AFSL

TP-Link WR841N

TP-Link WR841N



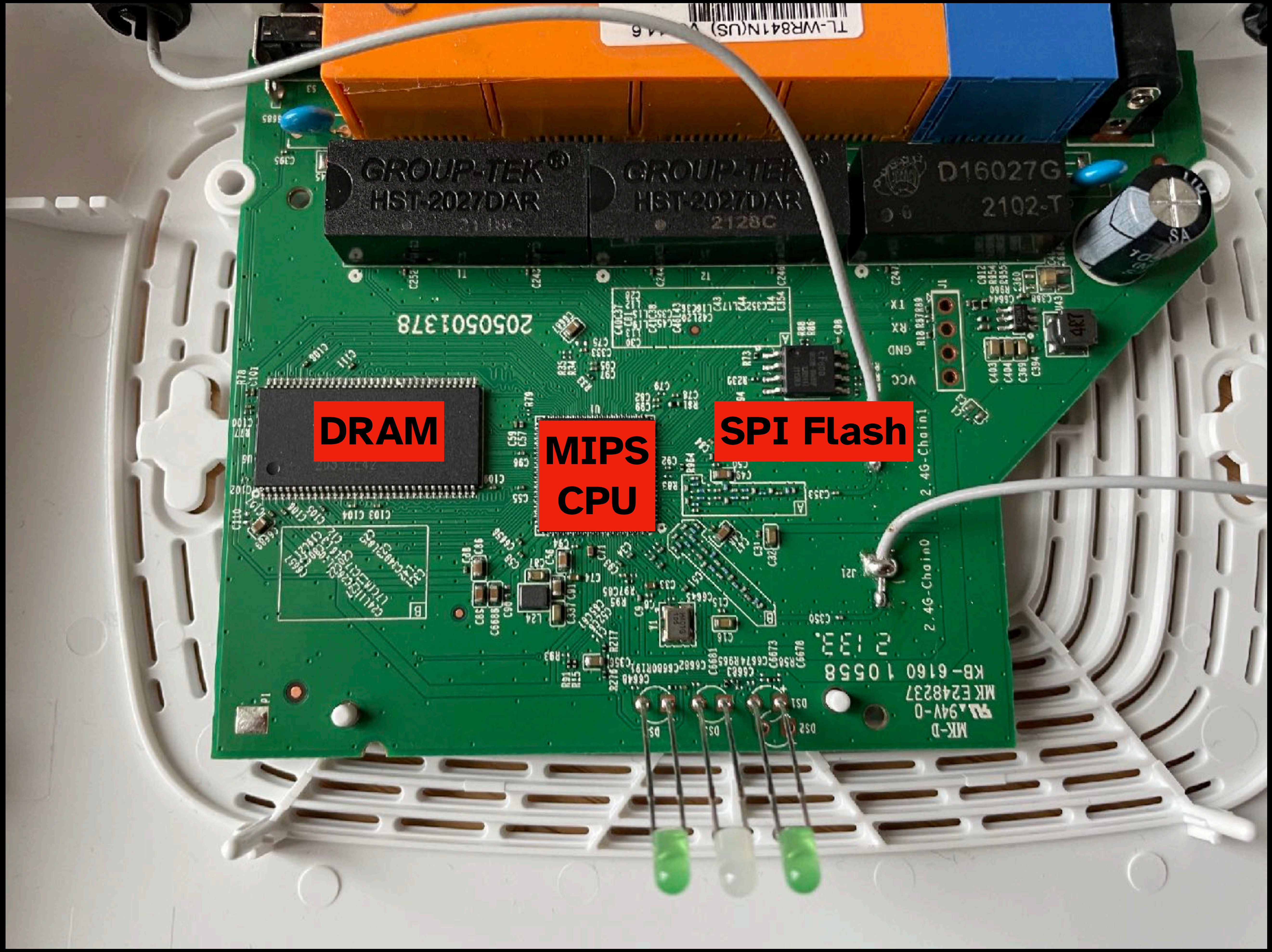


Core

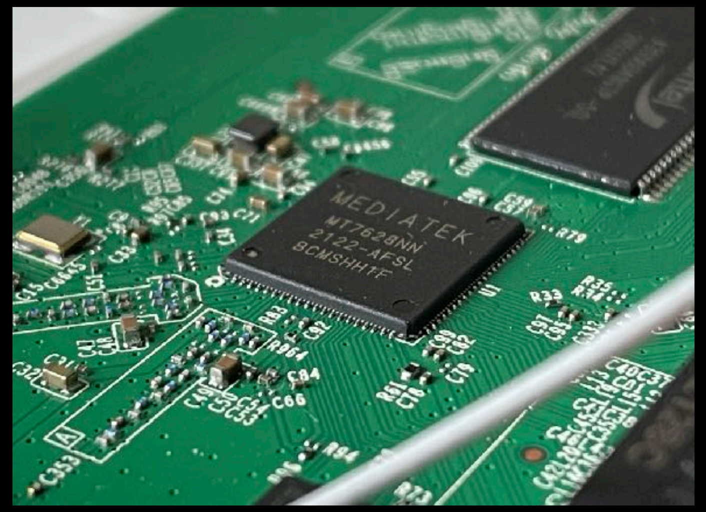
L1/L2 Cache

Memory

Non-Voltaile Storage

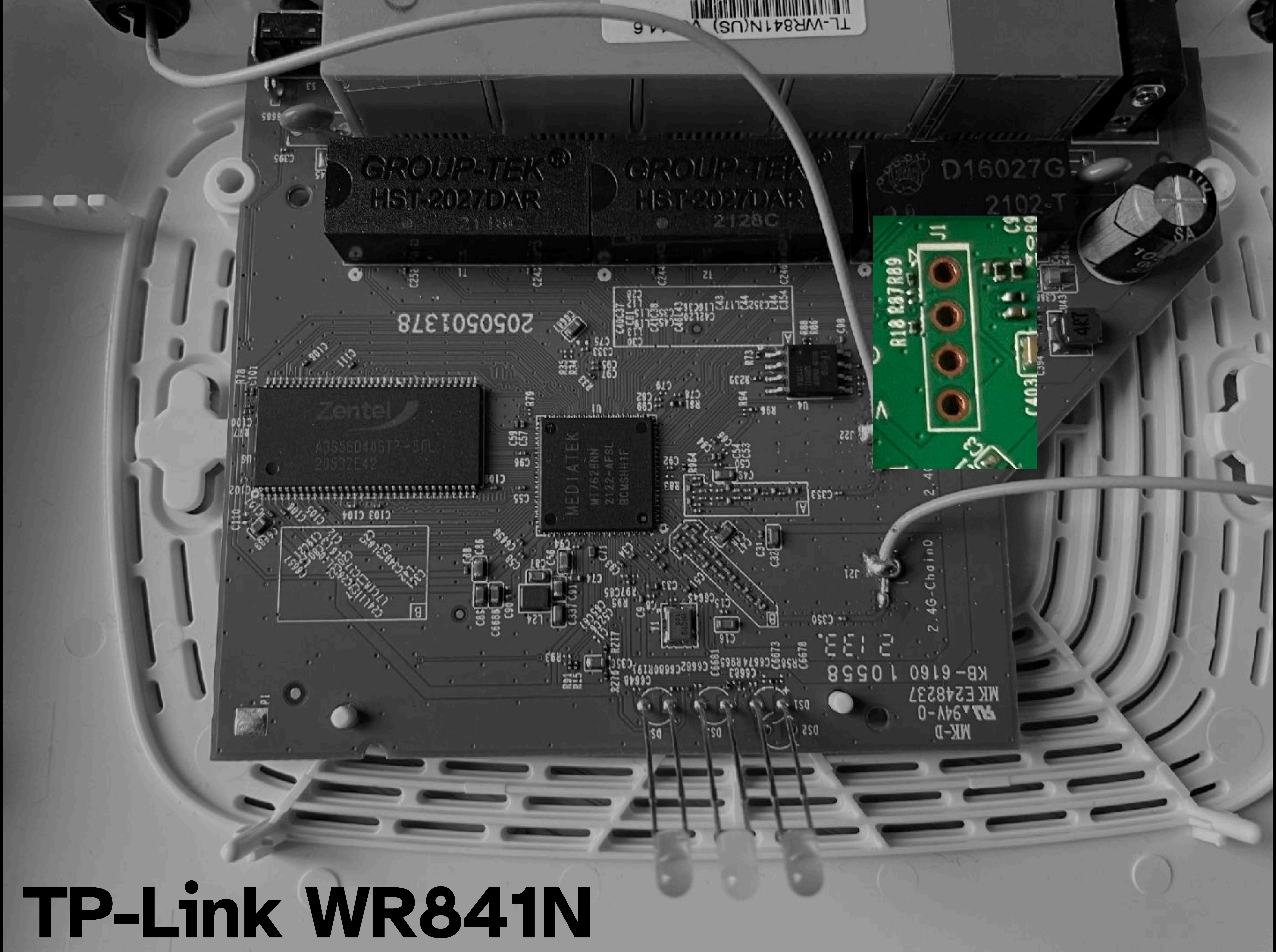


Core



Memory

SPI Flash



TP-Link WR841N

Hmmmm.....


2.4G-Chain1
VCC
GND
RX
TX
R18
R97
R89
R74
R90
R92
C98
C1
C3
C403
C404
C369

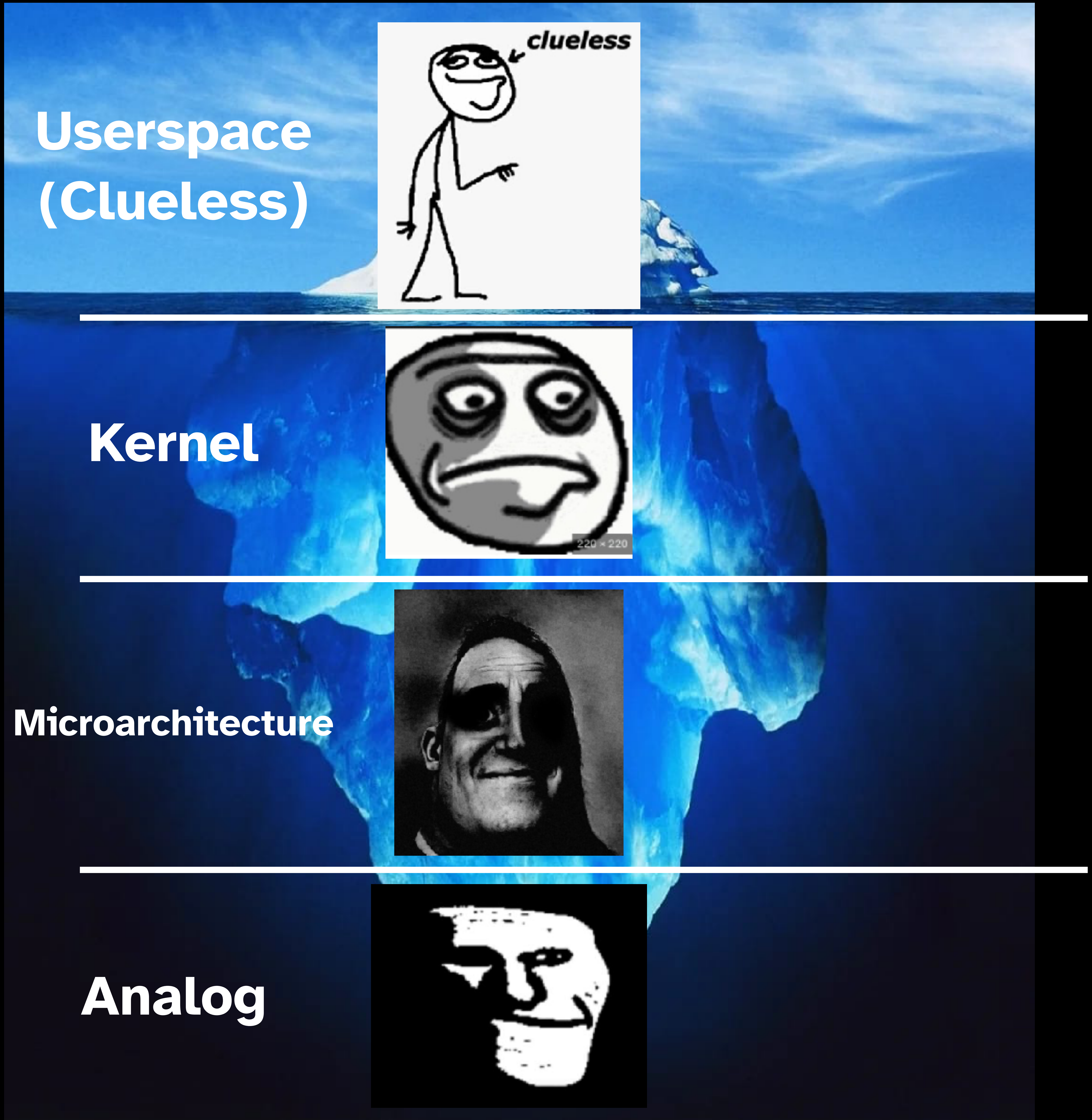
Demo 1

"What if the vendor just leaves the backdoor open?"

What other interfaces are out there?

- UART/ USART: Serial Console (usually root shell for free)
- JTAG/ SWD: Dump firmware, debug CPU, write your own firmware
- I2C/ SPI/ eMMC buses: Can sniff packets between flash and CPU to learn what the CPU is executing, even inject your own data!

The HW Security Iceberg


clueless
Userspace
(Clueless)
Kernel
Microarchitecture
Analog
Operating
System
ISA
Voltages
as 1s and 0s

Active

Inject new signals

Modify existing signals in new ways

Passive

No modification of signals

Only observe regular operation


Expensive
Focused Ion Beam
Optical Imaging
Laser Etching
Non-Invasive
Invasive
IC Decapping
Power Tracing
Fault Injection
Logical Interfaces
Cheap

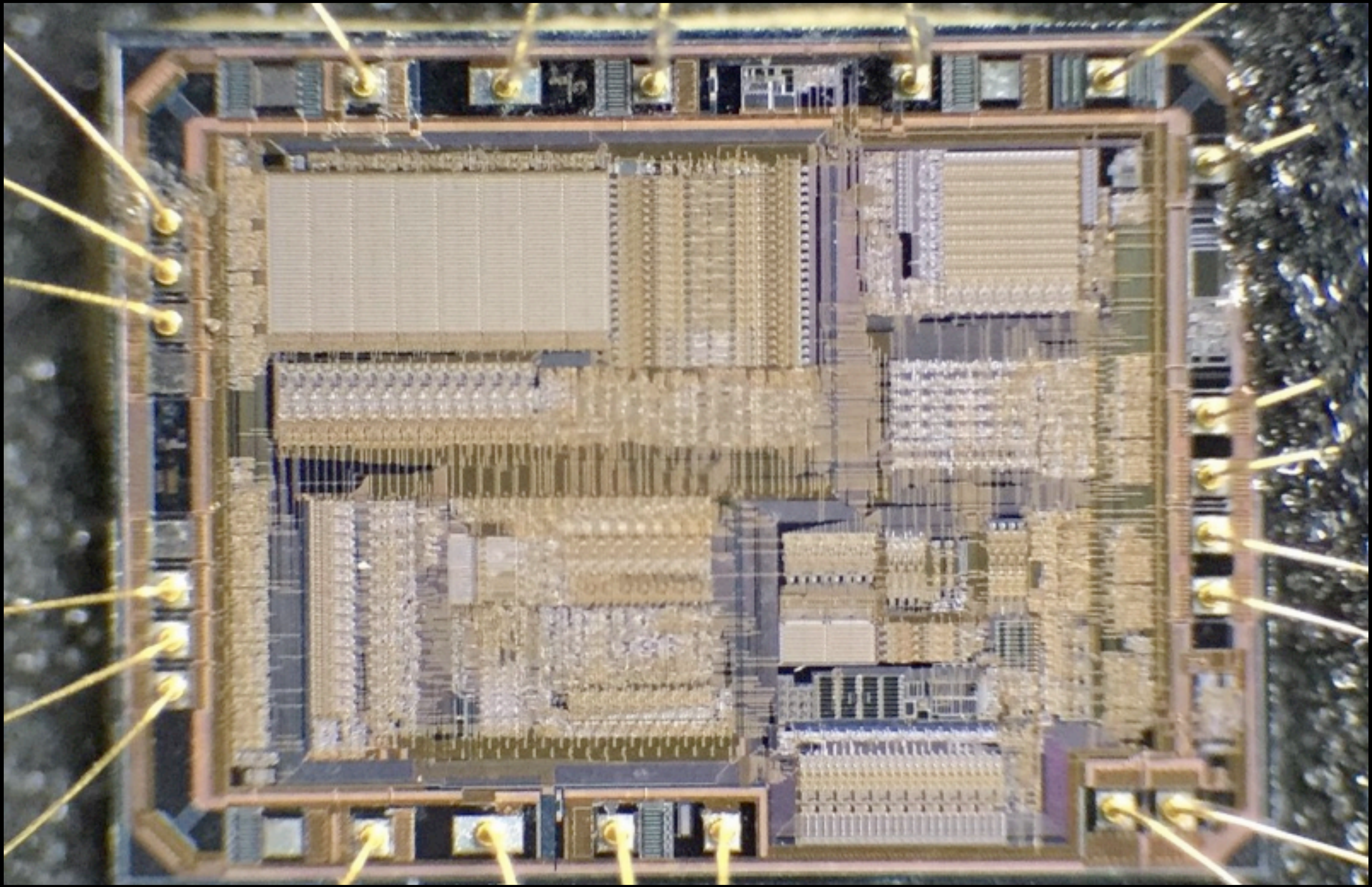


Image: Hackaday

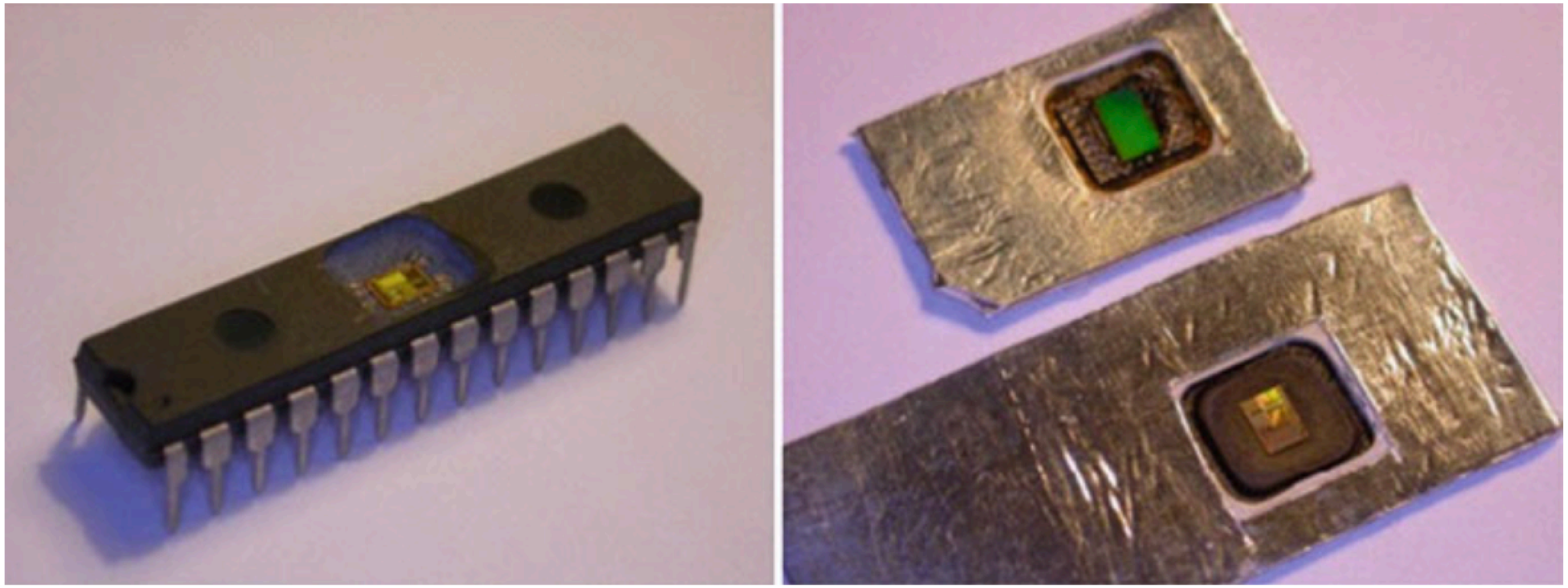


Fig. 7.3 Decapsulated chips

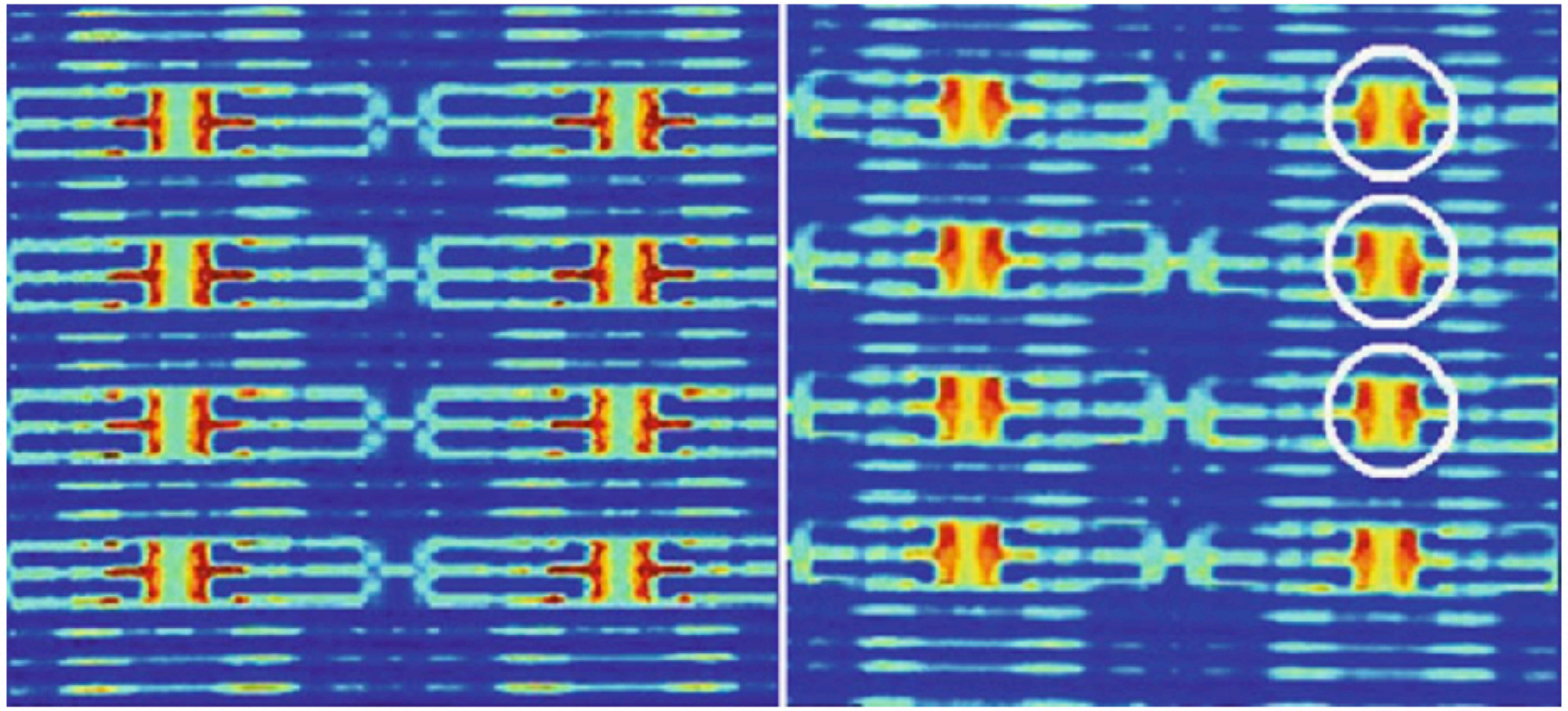


Fig. 7.6 Laser scan of unpowered and powered-up SRAM in PIC16F84 microcontroller

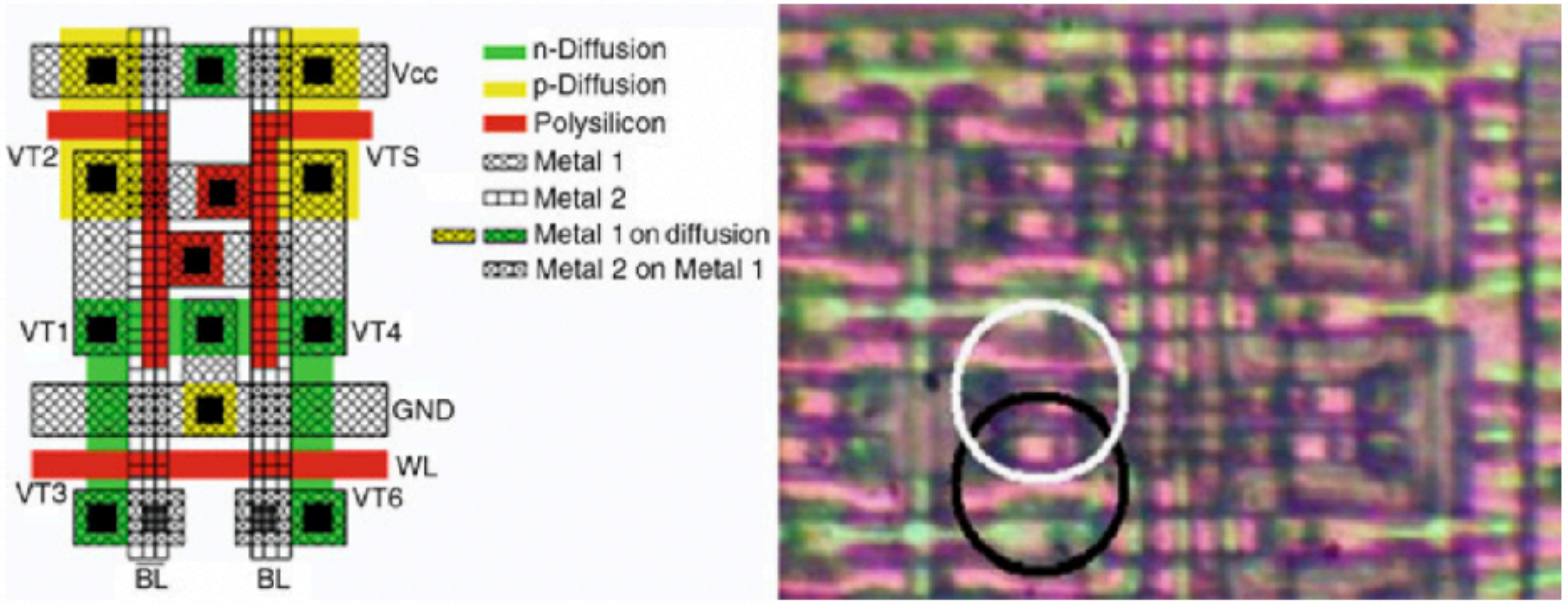


Fig. 7.7 Layout of SRAM cell and SRAM area in PIC16F84 microcontroller

4 Attacks

in this class.

Active

Fault Injection

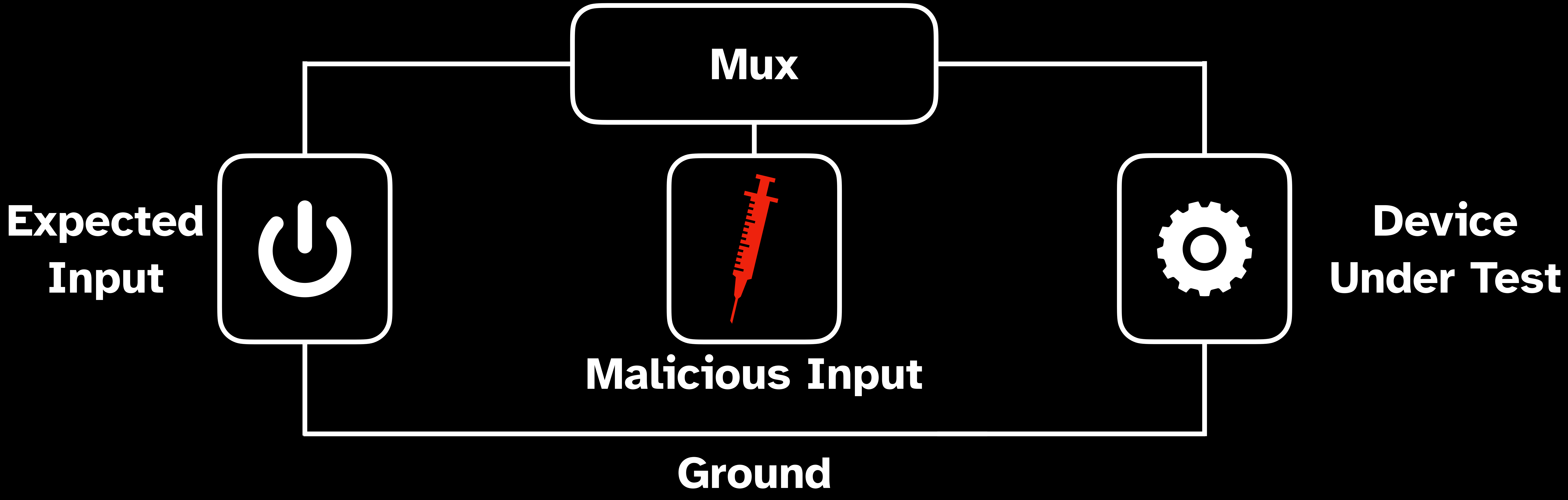
UART

Passive

Power Analysis

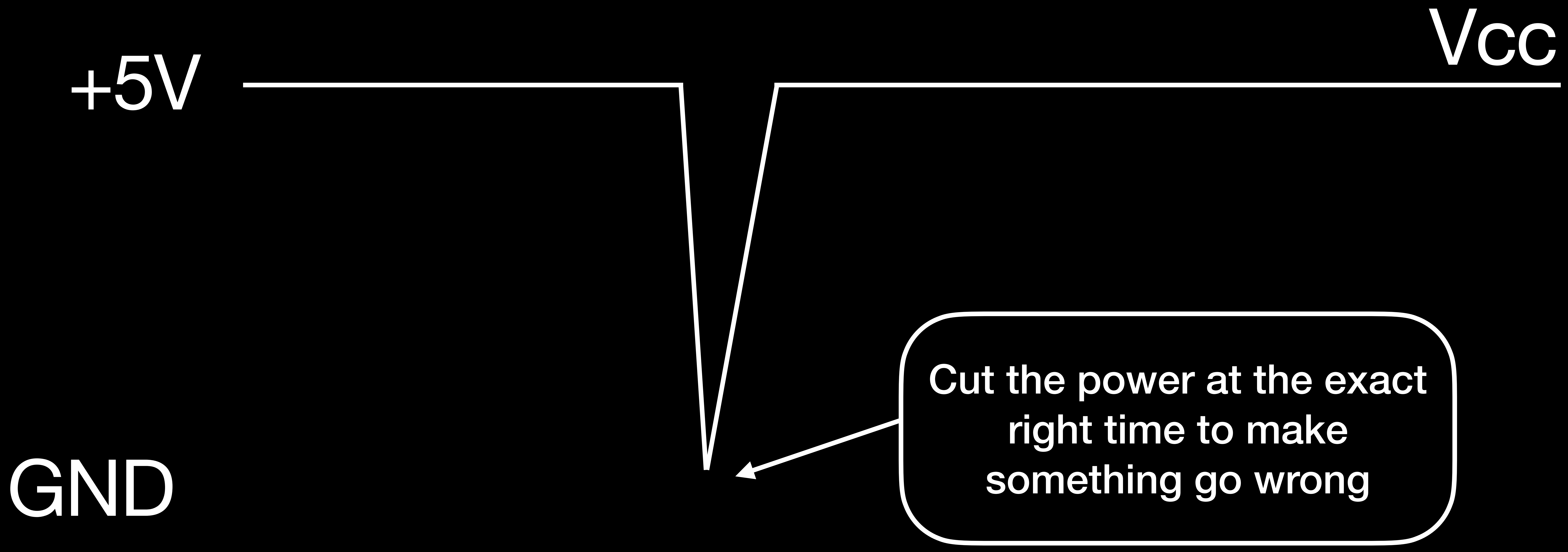
Timing Analysis

Fault Injection



Mux
Expected Input
Device Under Test
Malicious Input
Ground

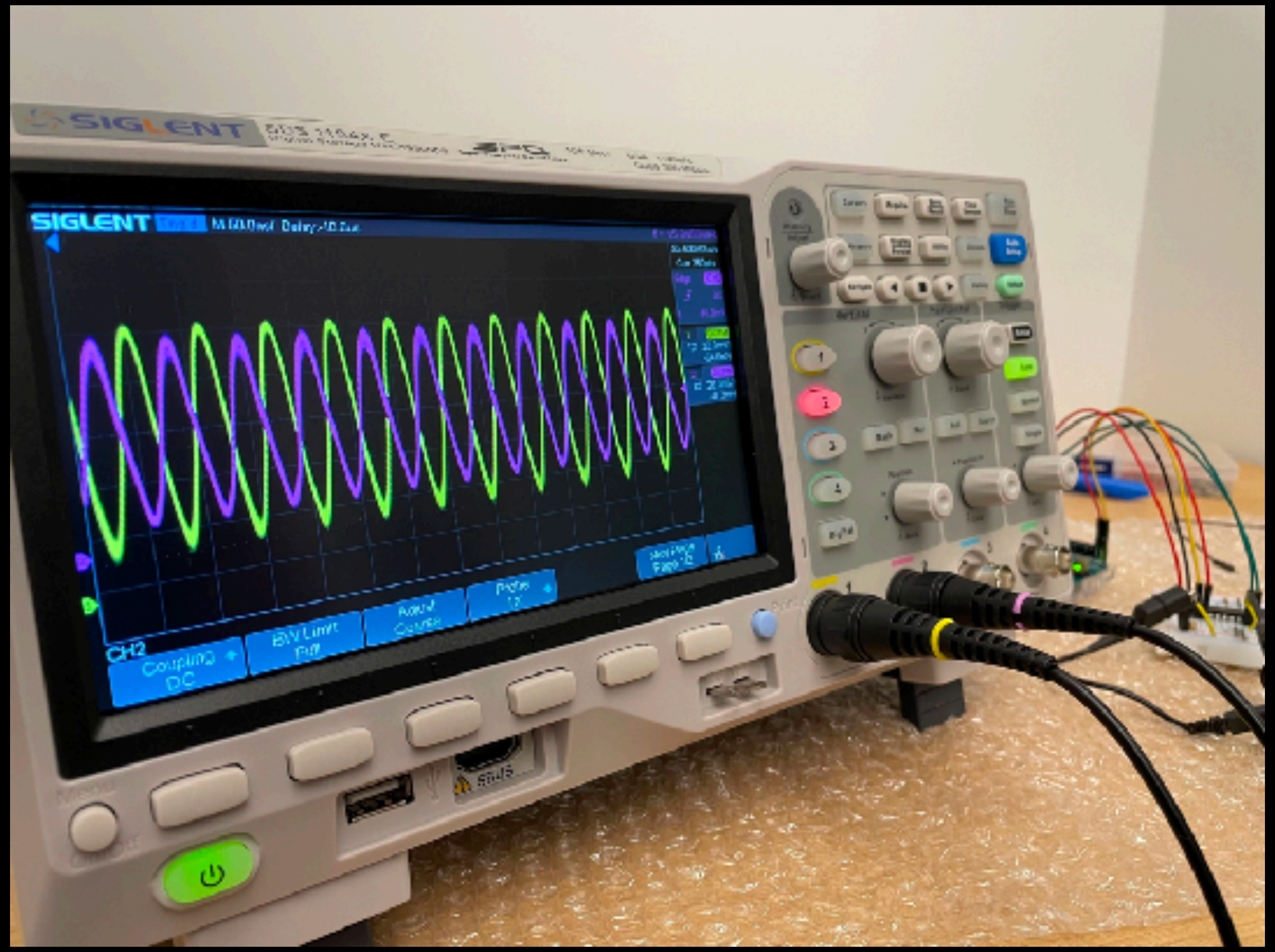
Voltage Glitching



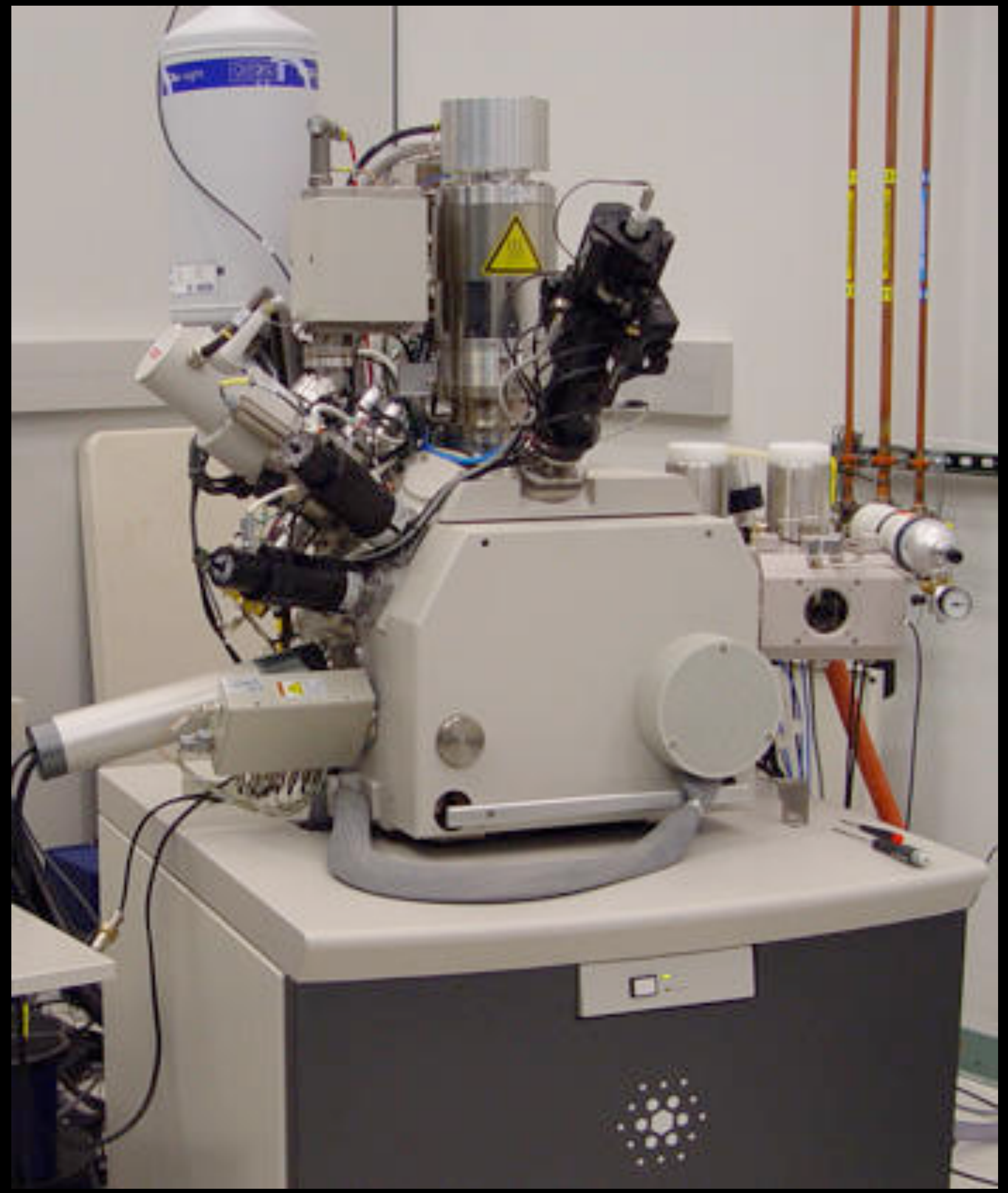
Tools

Cheap

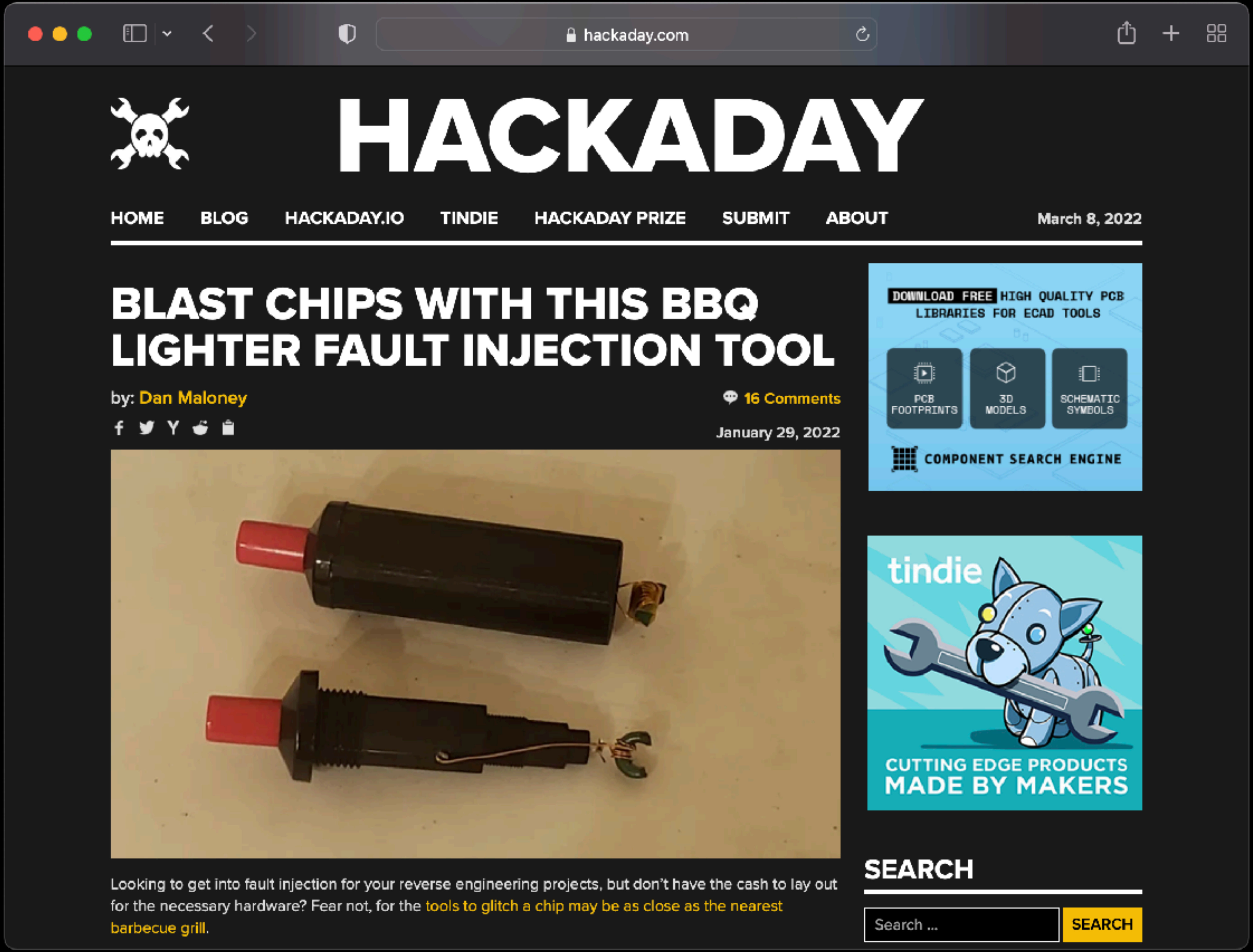
Affordable



Crazy Expensive

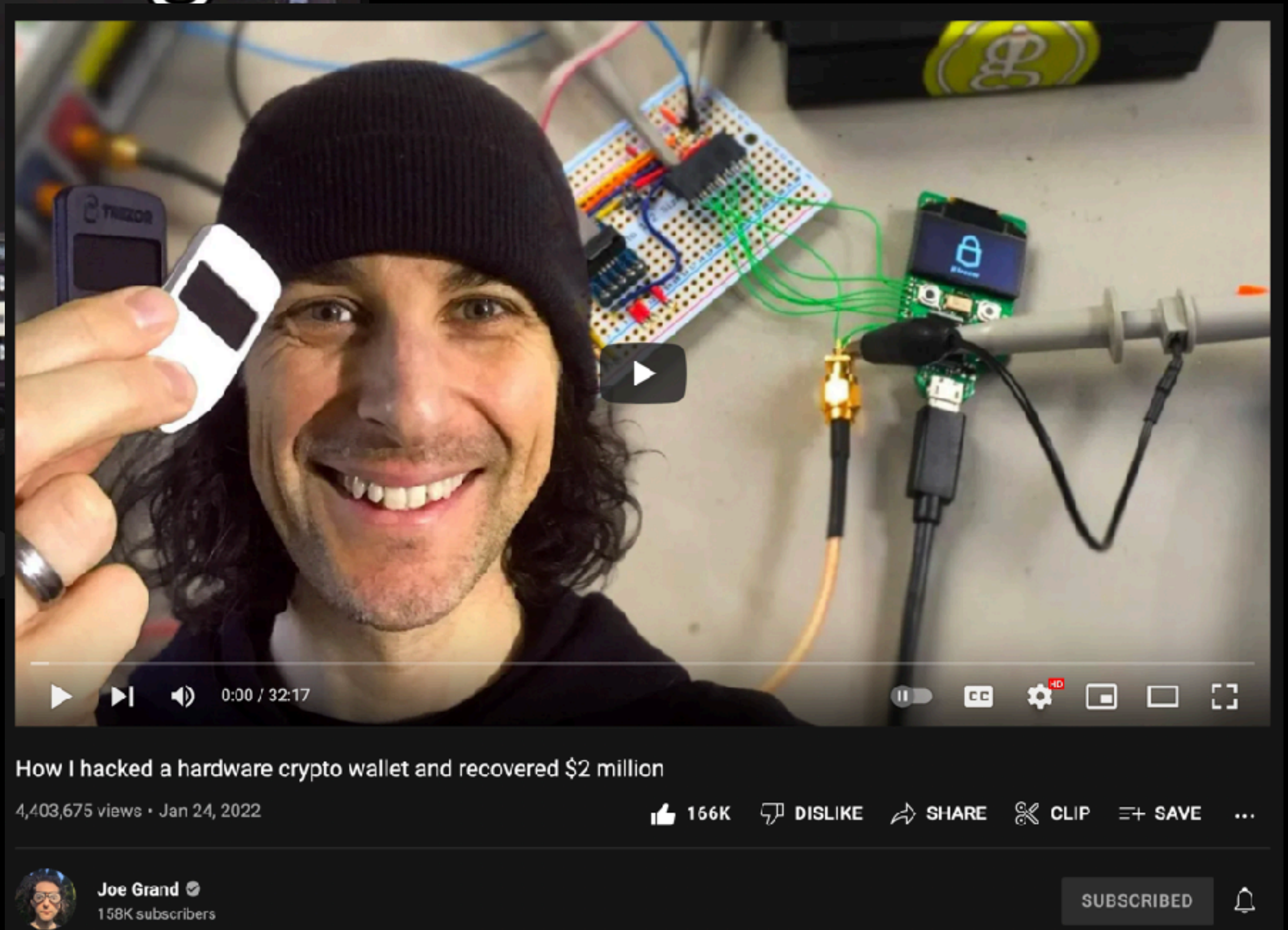


Yes, Really



hackaday.com
HACKADAY
HOME
BLOG
HACKADAY.IO
TINDIE
HACKADAY PRIZE
SUBMIT
ABOUT
March 8, 2022
BLAST CHIPS WITH THIS BBQ LIGHTER FAULT INJECTION TOOL
by: Dan Maloney
16 Comments
January 29, 2022
Looking to get into fault injection for your reverse engineering projects, but don't have the cash to lay out for the necessary hardware? Fear not, for the tools to glitch a chip may be as close as the nearest barbecue grill.
DOWNLOAD FREE HIGH QUALITY PCB LIBRARIES FOR ECAD TOOLS
PCB FOOTPRINTS
3D MODELS
SCHEMATIC SYMBOLS
COMPONENT SEARCH ENGINE
tindie
CUTTING EDGE PRODUCTS MADE BY MAKERS
SEARCH
Search ...
SEARCH

Notable Examples



What is "Firmware"?

- It's just software running on an embedded device.
- Can be bare metal, real-time OS, or even Linux.

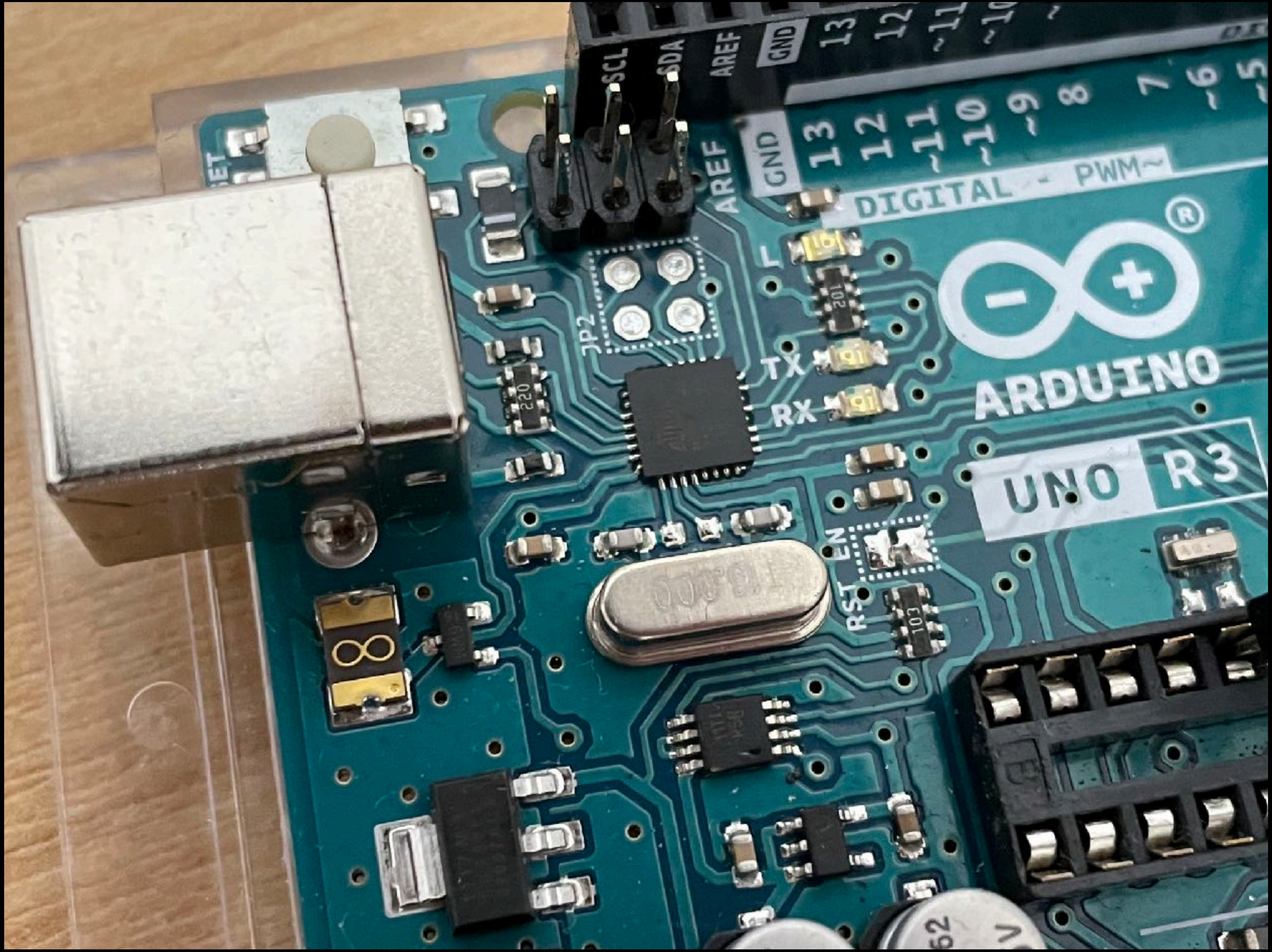
Pseudocode

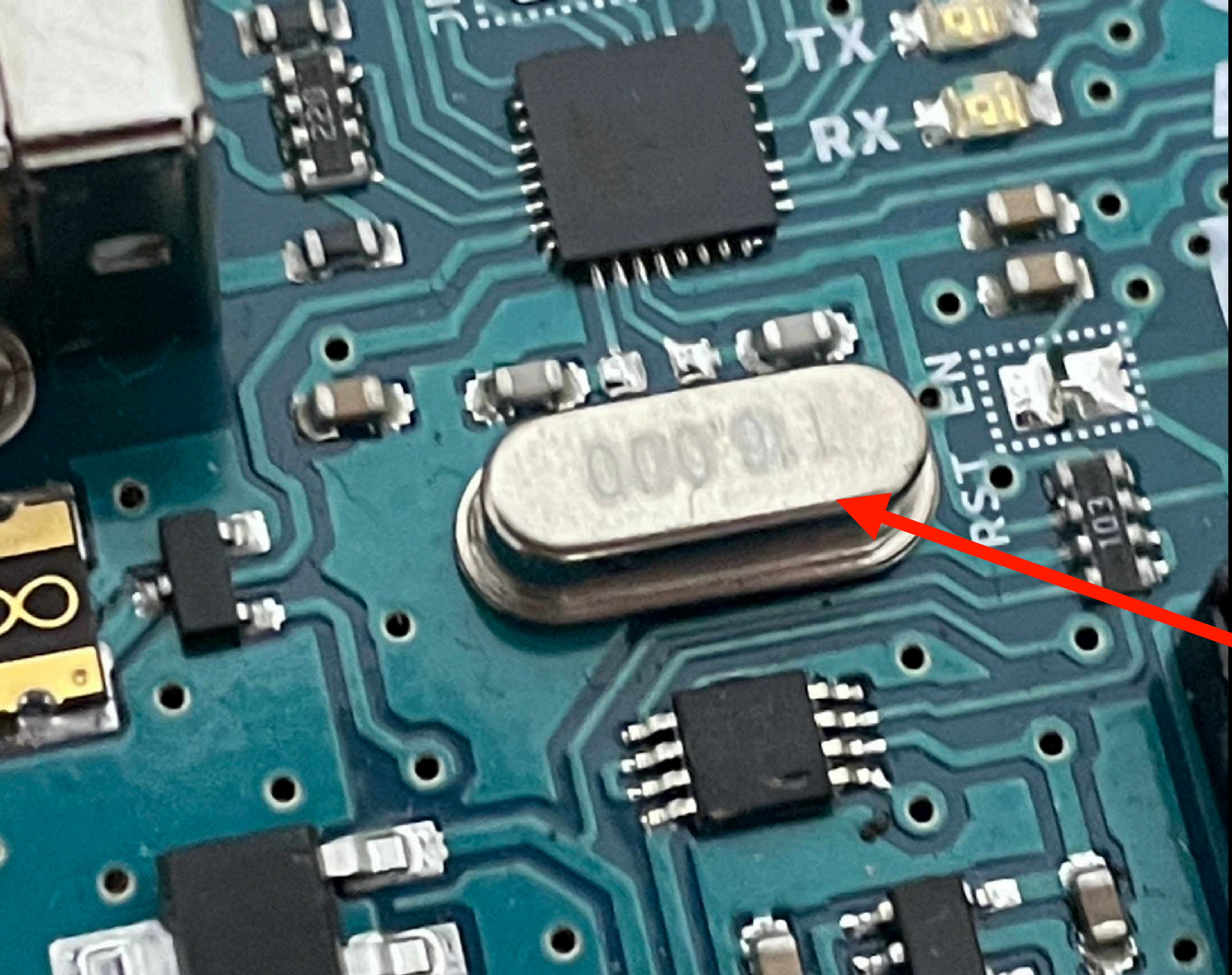
```
while(chksum == CORRECT_CHECKSUM) {
    chksum = compute_checksum();
    print("Locked! %d %d", chksum, iter);
    iter++;
}
print("MIT{flag}");
```

Pseudocode

```
while(chksum == CORRECT_CHECKSUM) {
    chksum = compute_checksum();
    print("Locked! %d %d", chksum, iter);
    iter++;
}
print("MIT{flag}");
```

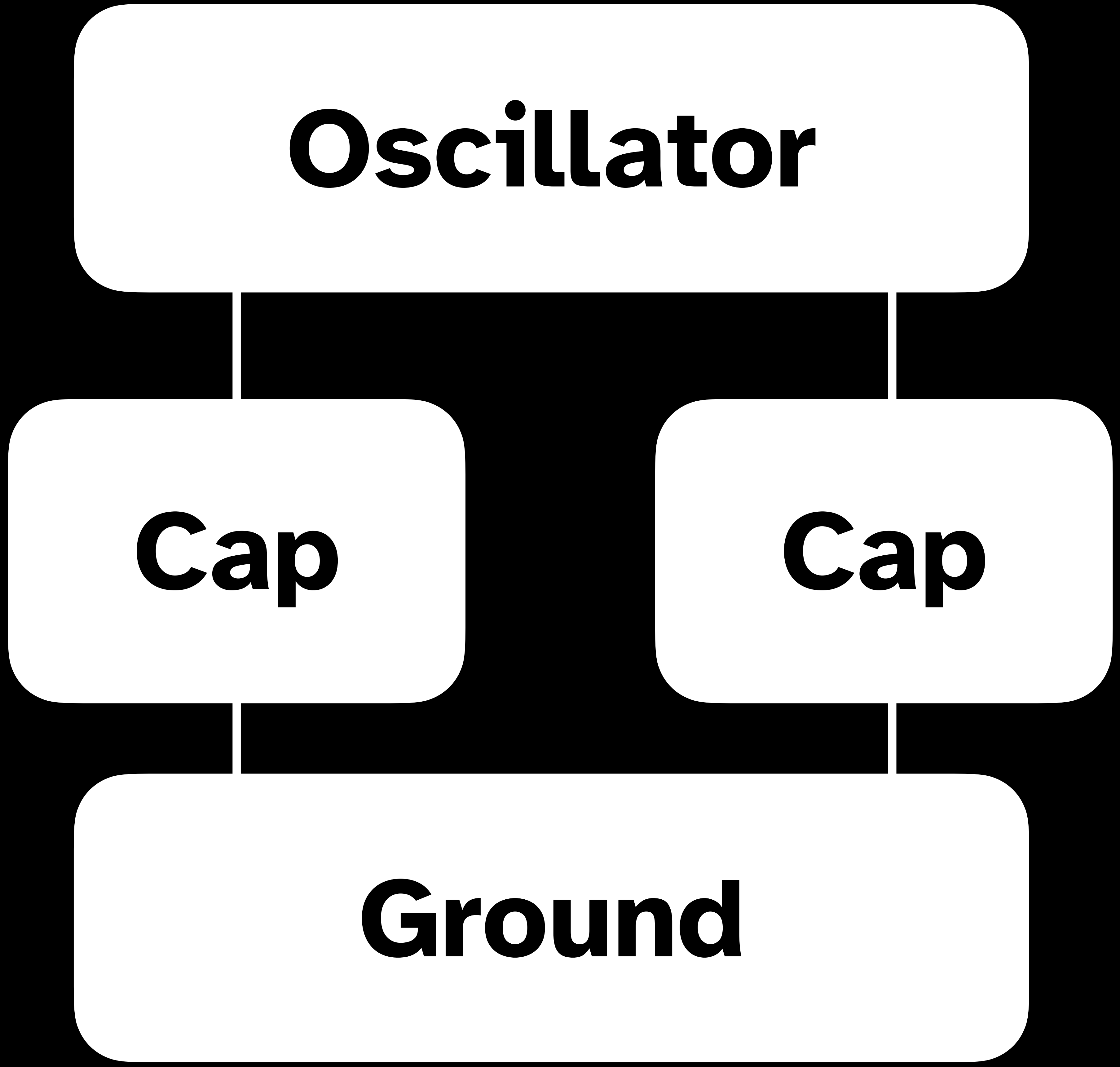
Inject Fault here

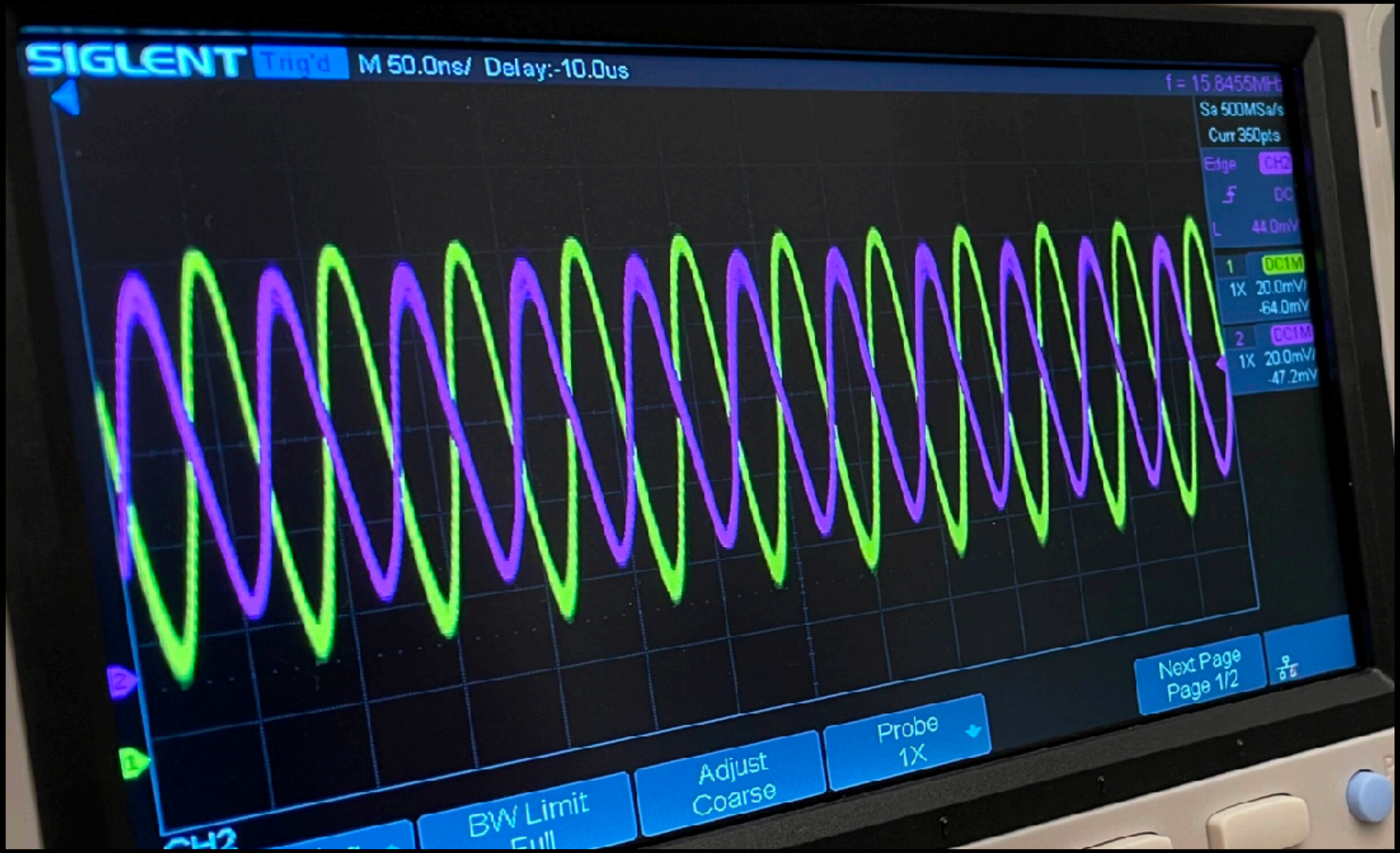

SCL
SDA
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
DIGITAL - PWM~
L
TX
RX
JP2
ARDUINO
UNO
R3
EN
RST



Crystal Oscillator

Clock Glitching





Crystal Oscillator

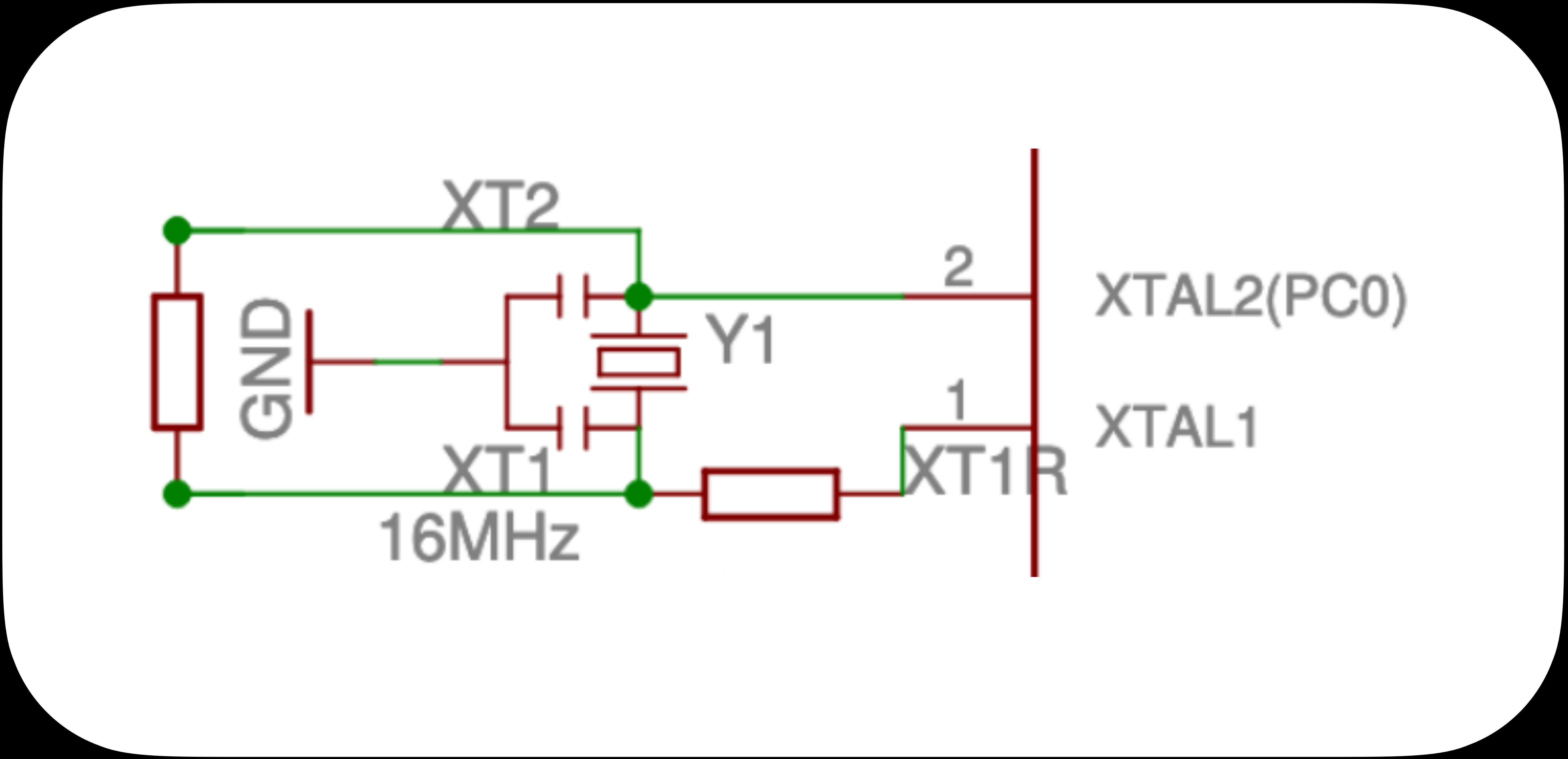


Image: Arduino Uno R3 Reference Design

Crystal Oscillator

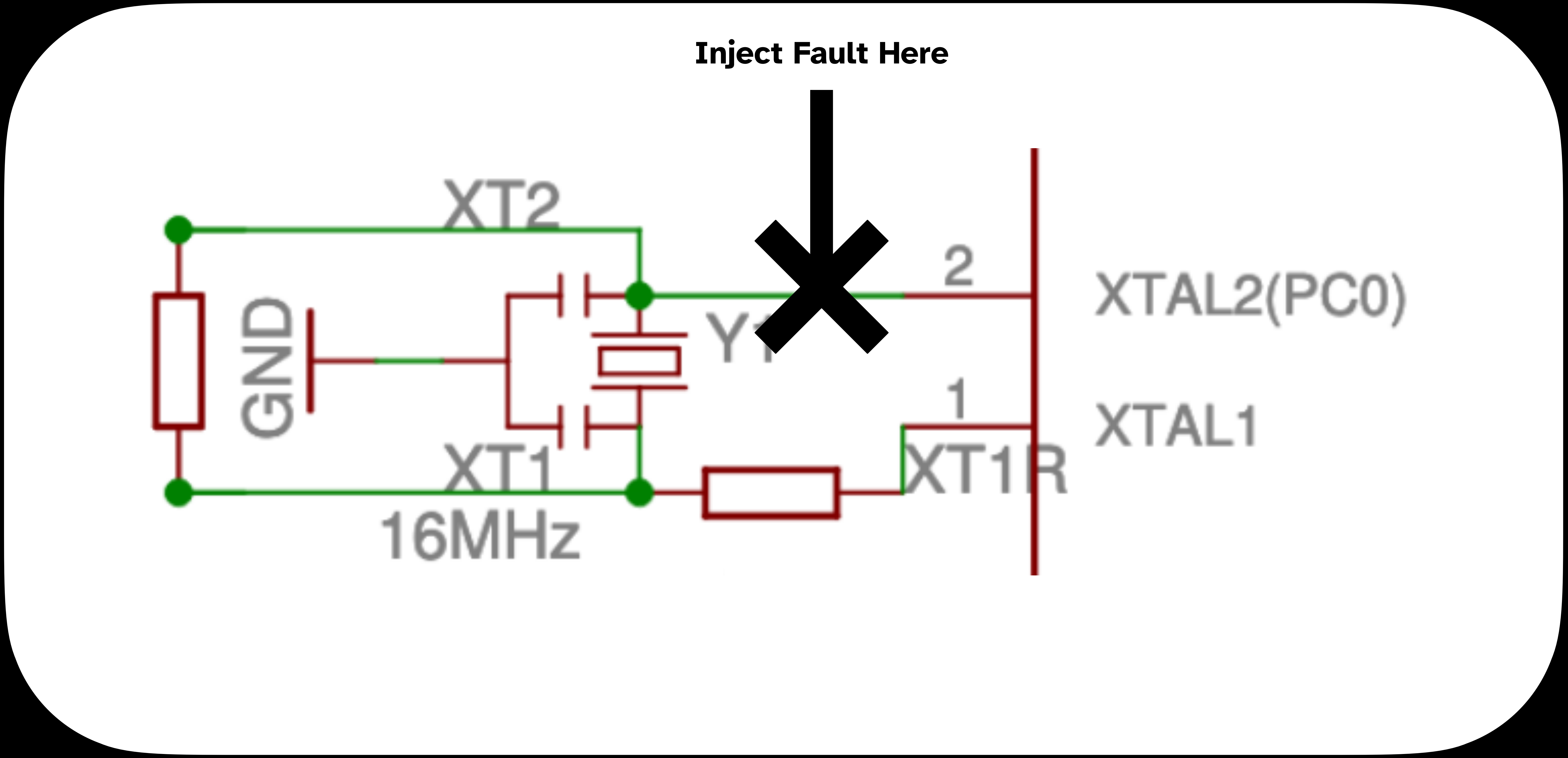
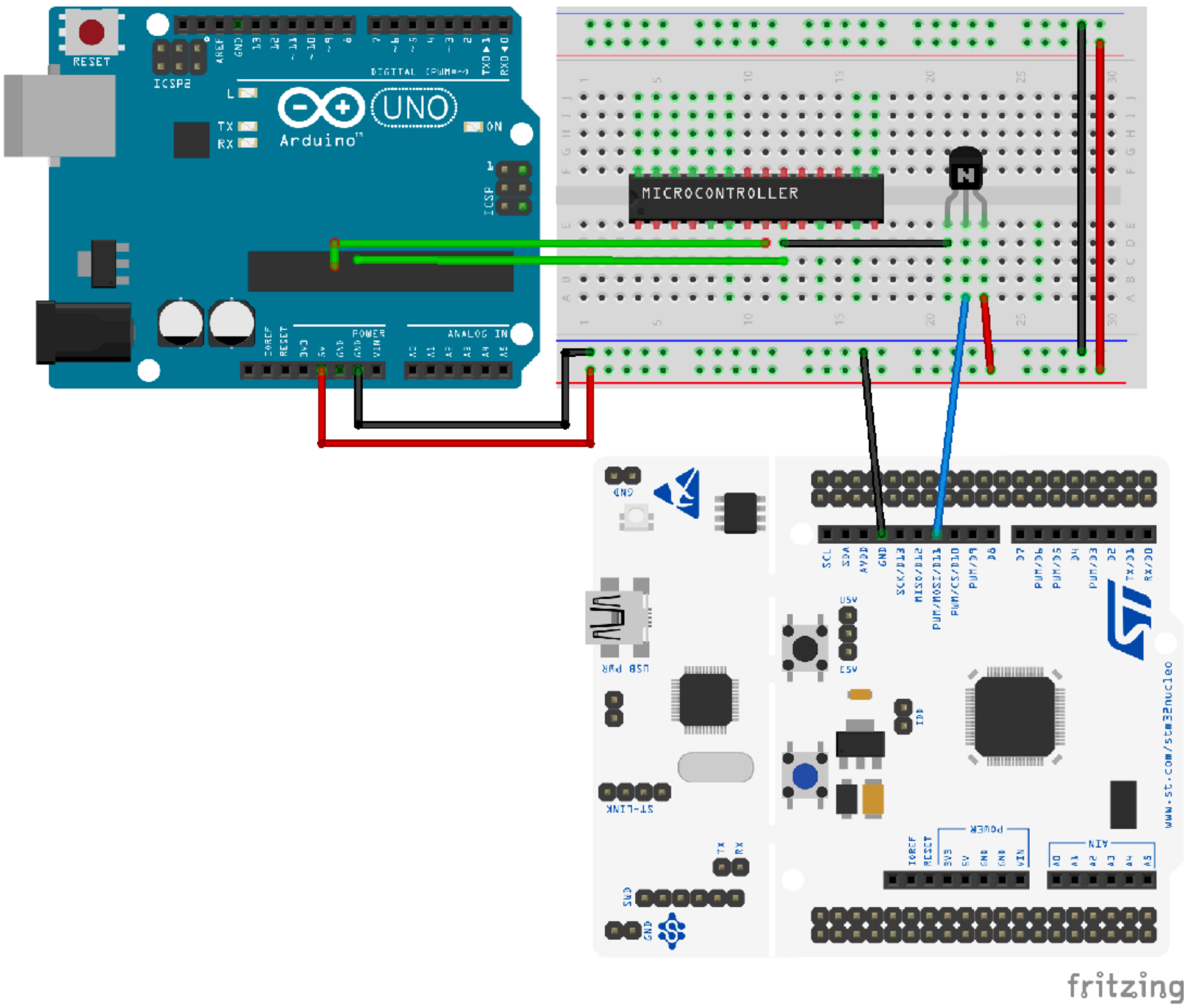
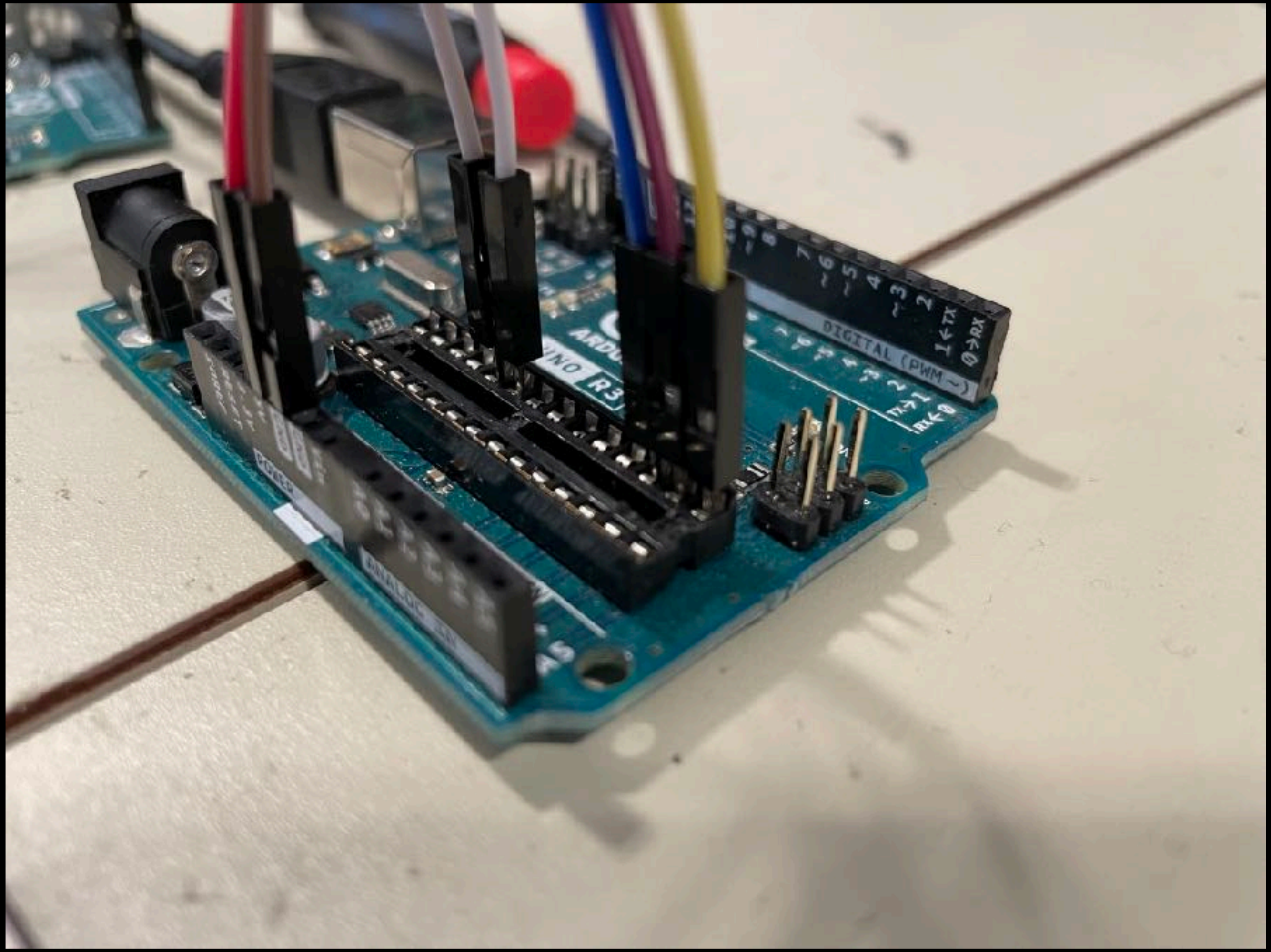
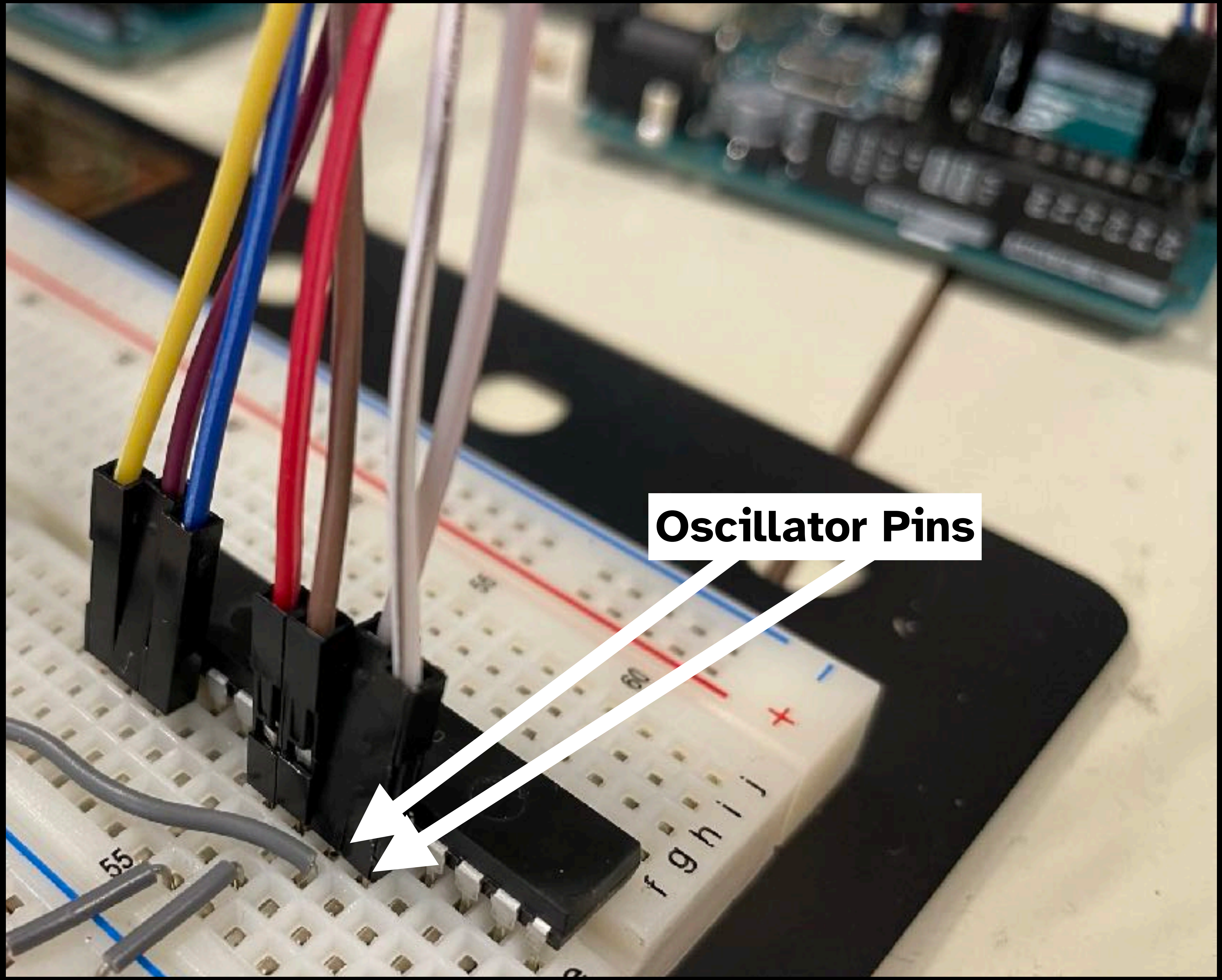


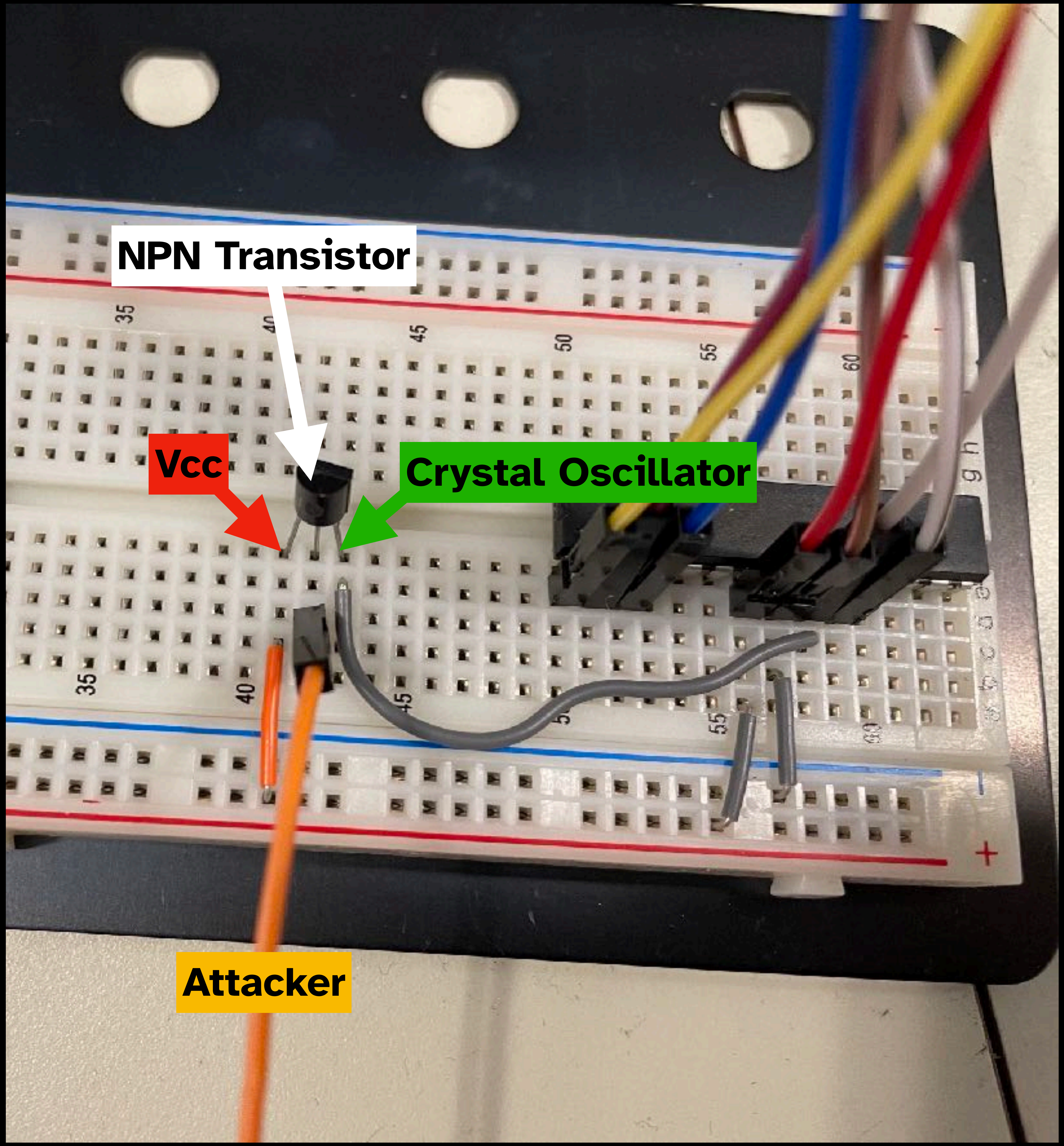
Image: Arduino Uno R3 Reference Design

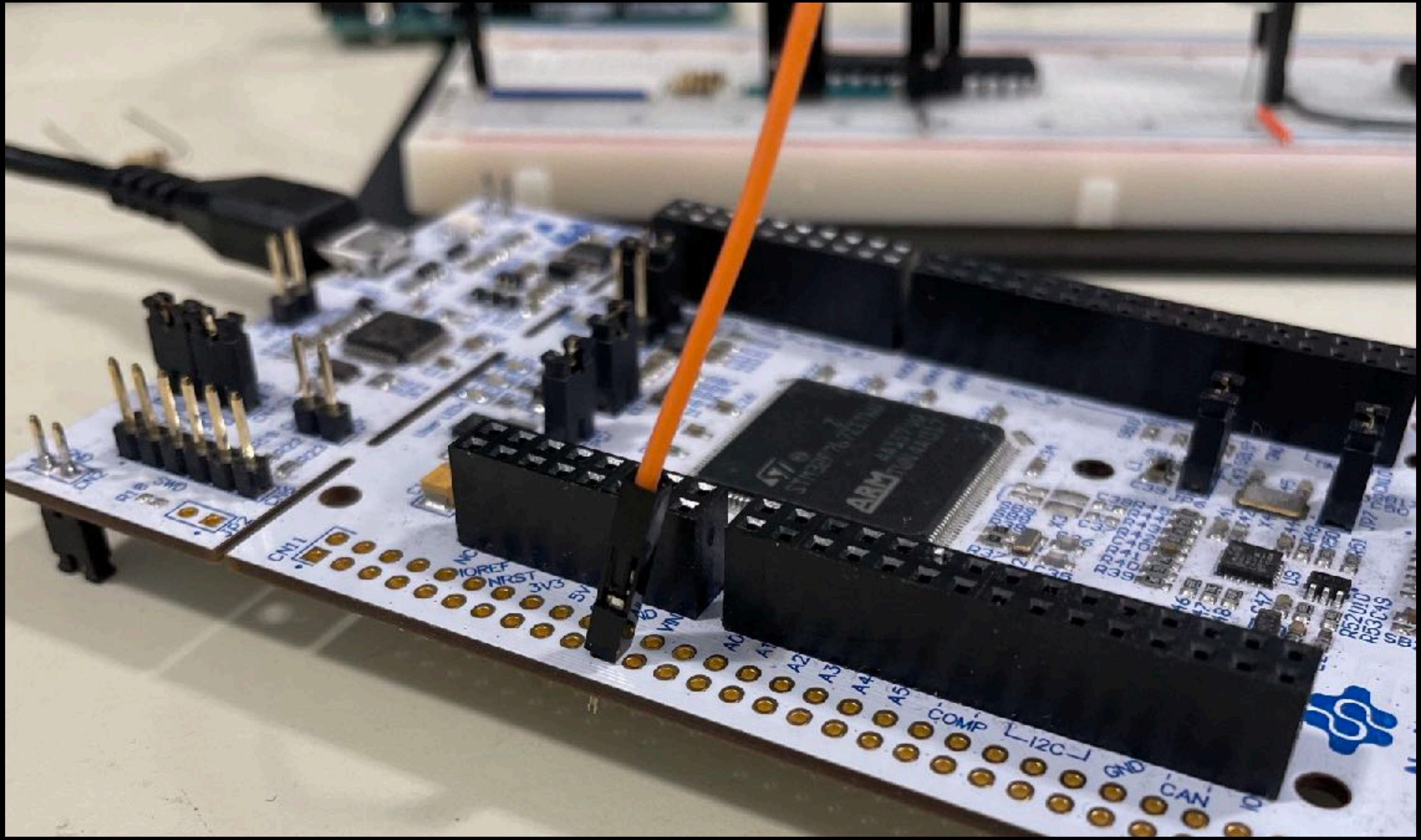


RESET
ICSP2
UNO
Arduino
ON
TX
RX
ICSP
DIGITAL (PWM=~)
POWER
ANALOG IN
MICROCONTROLLER
ST-LINK
fritzing




Oscillator Pins


NPN Transistor
Vcc
Crystal Oscillator
Attacker


CN11
IOREF
NRST
3V3
5V
A2
A3
A4
A5
COMP
I2C
GND
CAN
ARM

Pseudocode

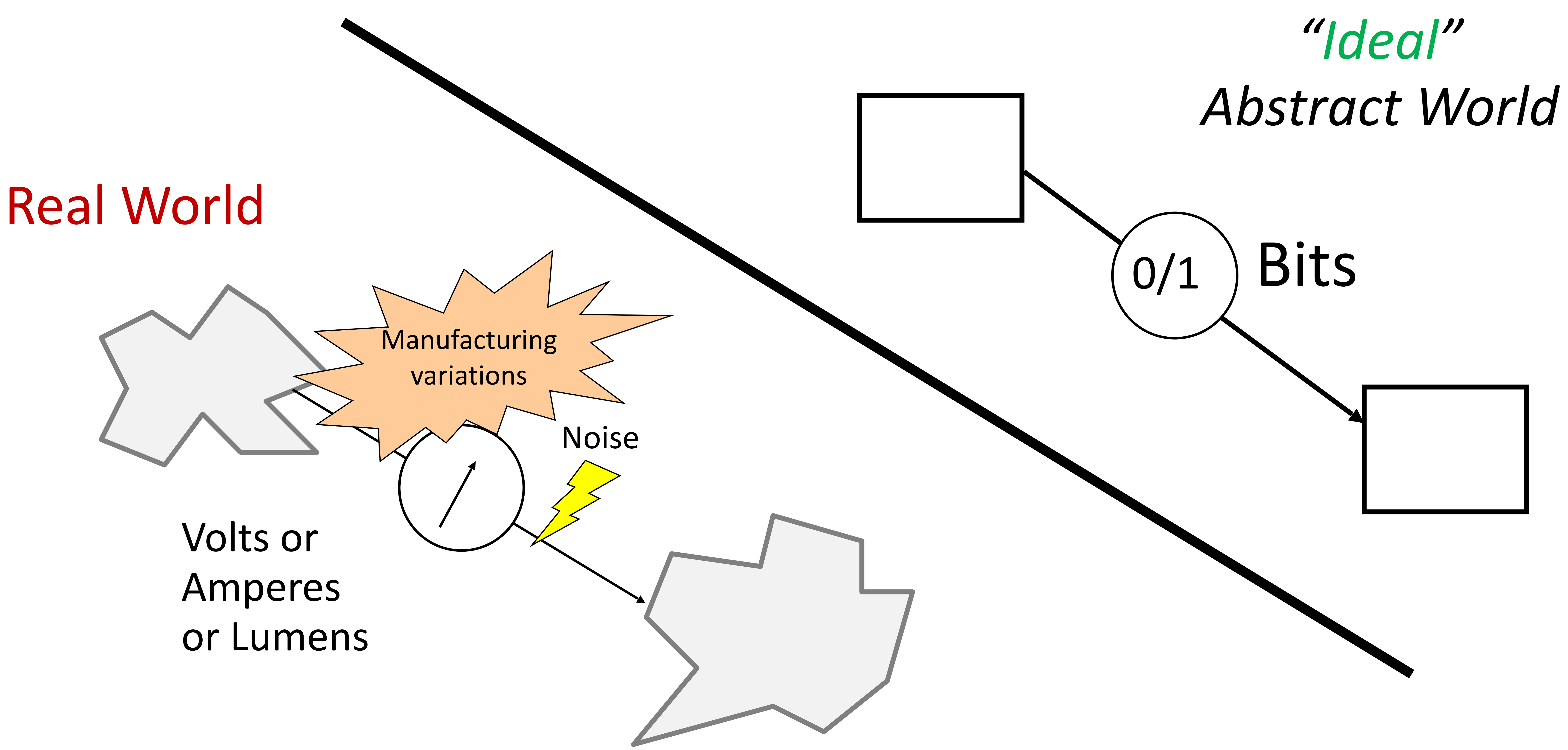
```
while(chksum == CORRECT_CHECKSUM) {
    chksum = compute_checksum();
    print("Locked! %d %d", chksum, iter);
    iter++;
}
print("MIT{flag}");
```

Demo 2

"What if we intentionally violate the chip's expected operating conditions?"

Demystify
Fault Injection Attacks

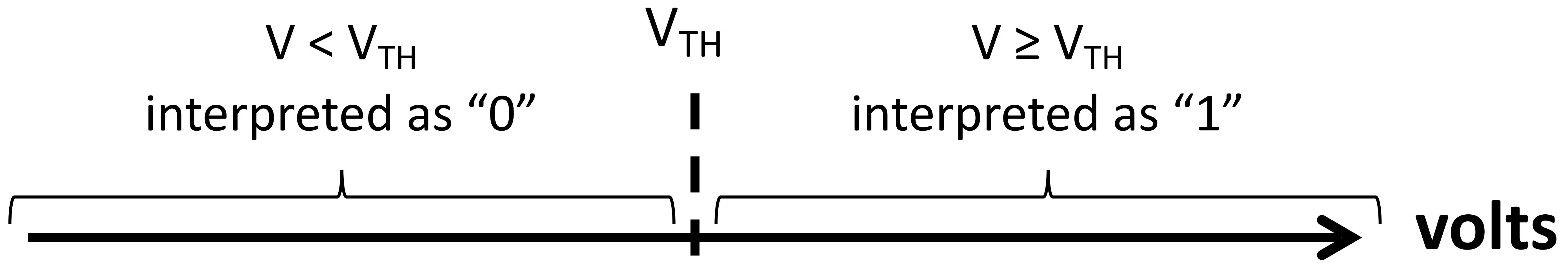
The Digital Abstraction



Using Voltages “Digitally”

Attempt #1:

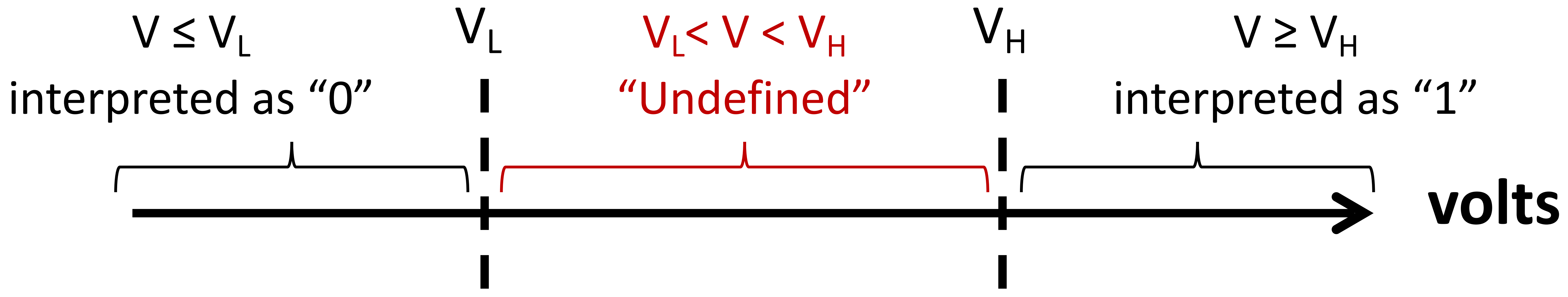
✗



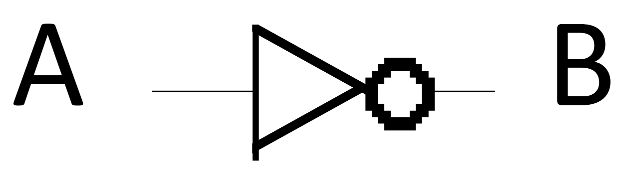
Not quite correct. Why? *Hard to distinguish* $V_{TH}-\varepsilon$ *from* $V_{TH}+\varepsilon$

Attempt #2:

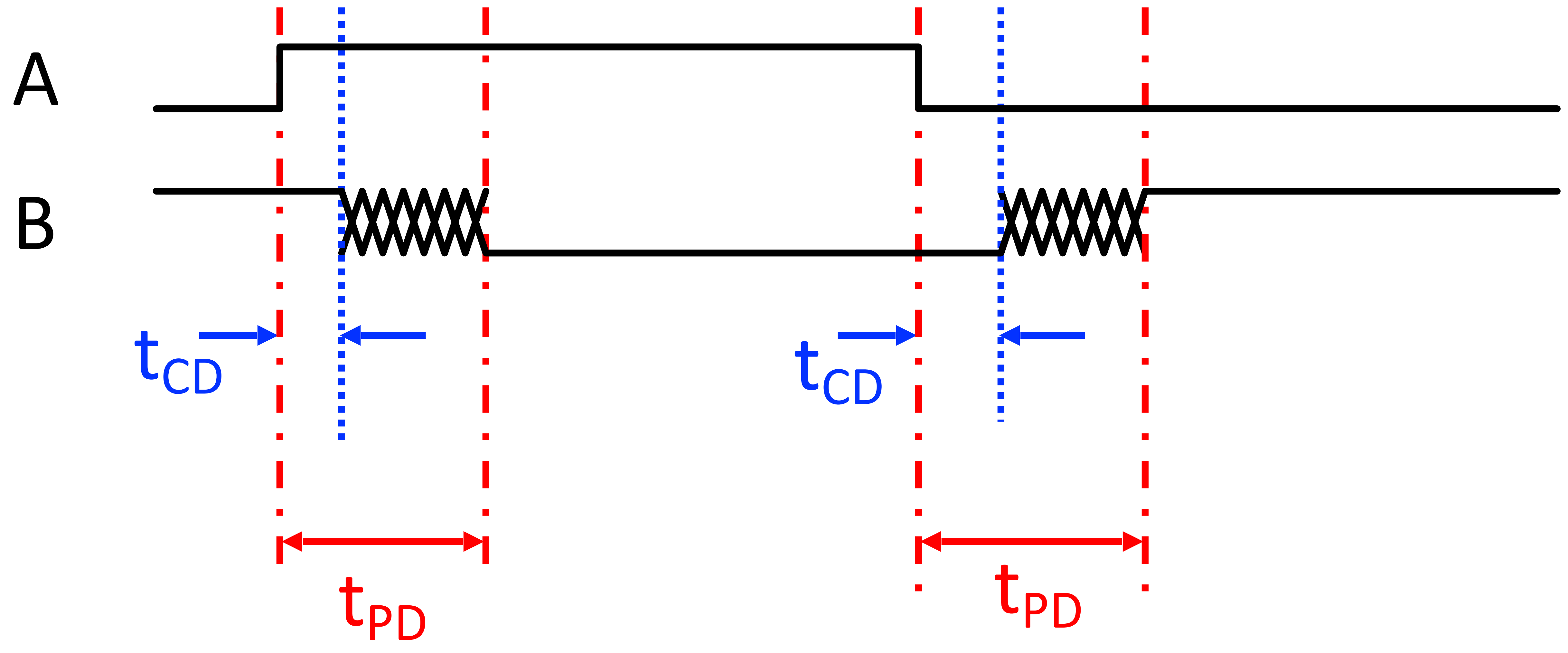
✓



Combinational Circuit Timing

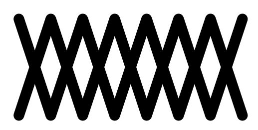


A	B
0	1
1	0

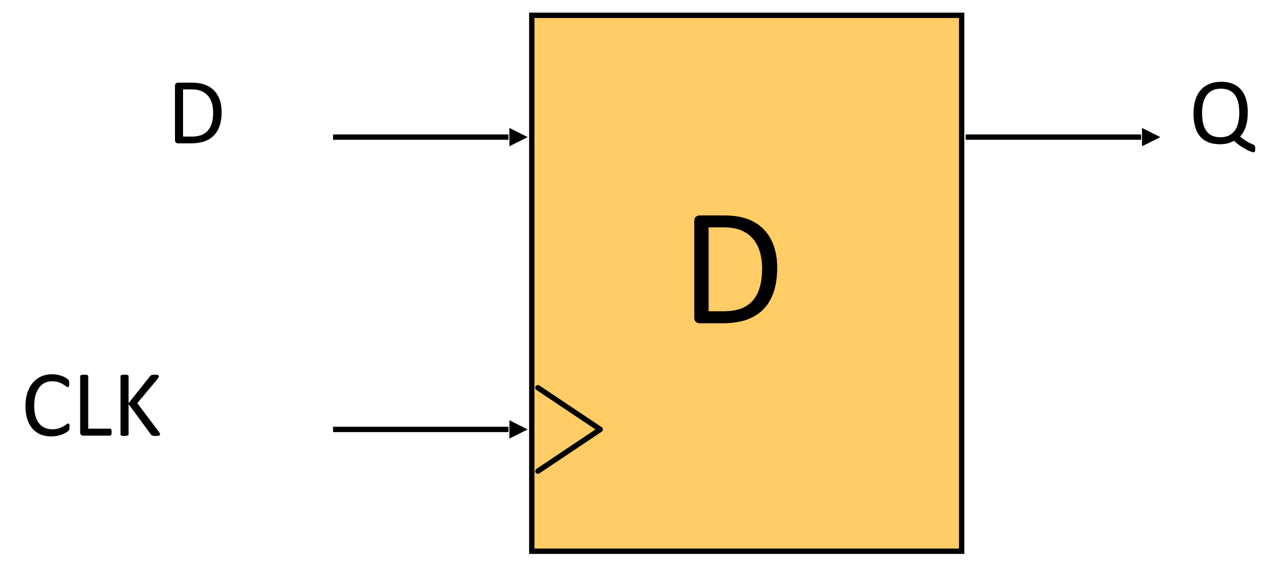


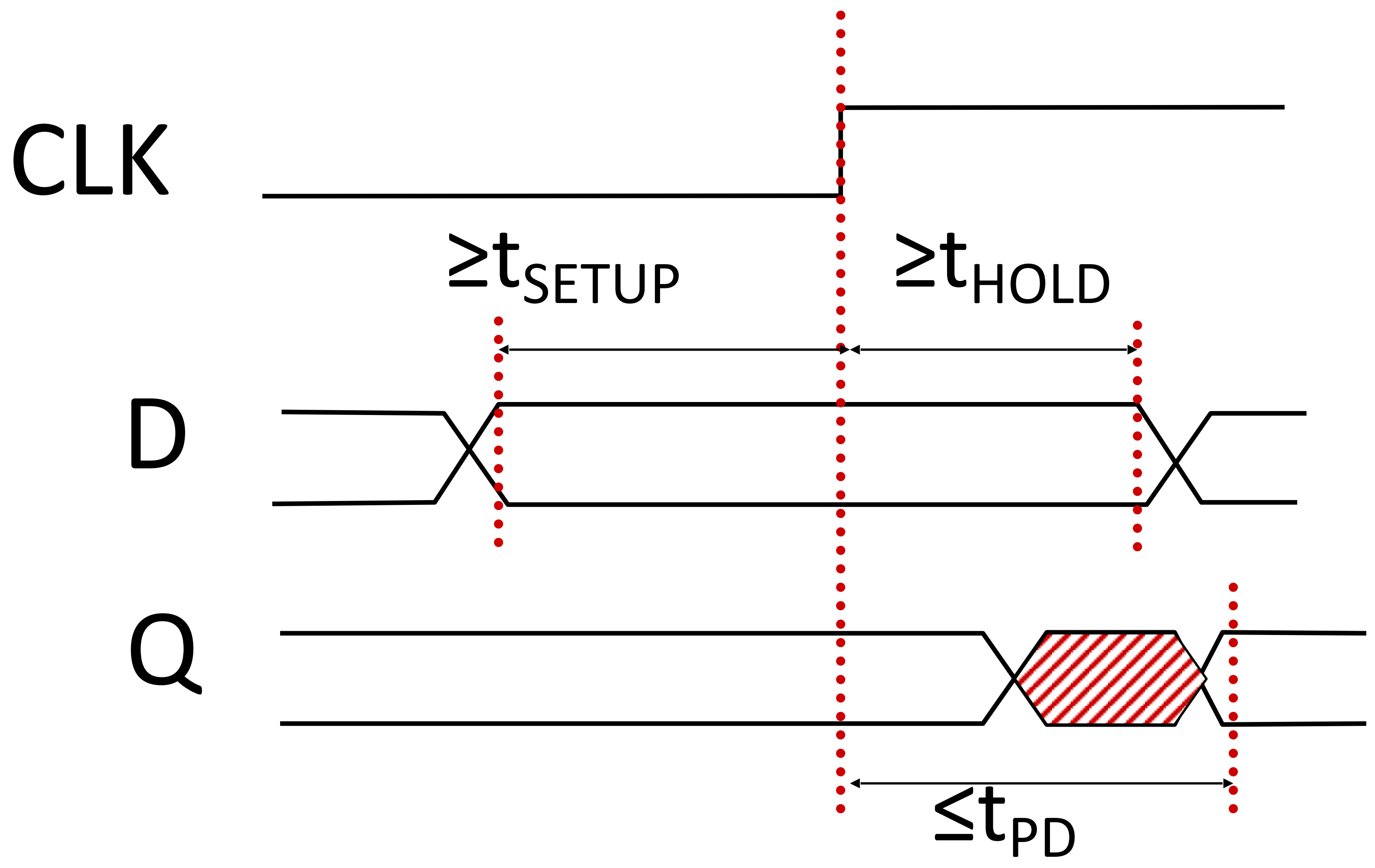
t_{PD} propagation delay

t_{CD} contamination delay

No Promises during 

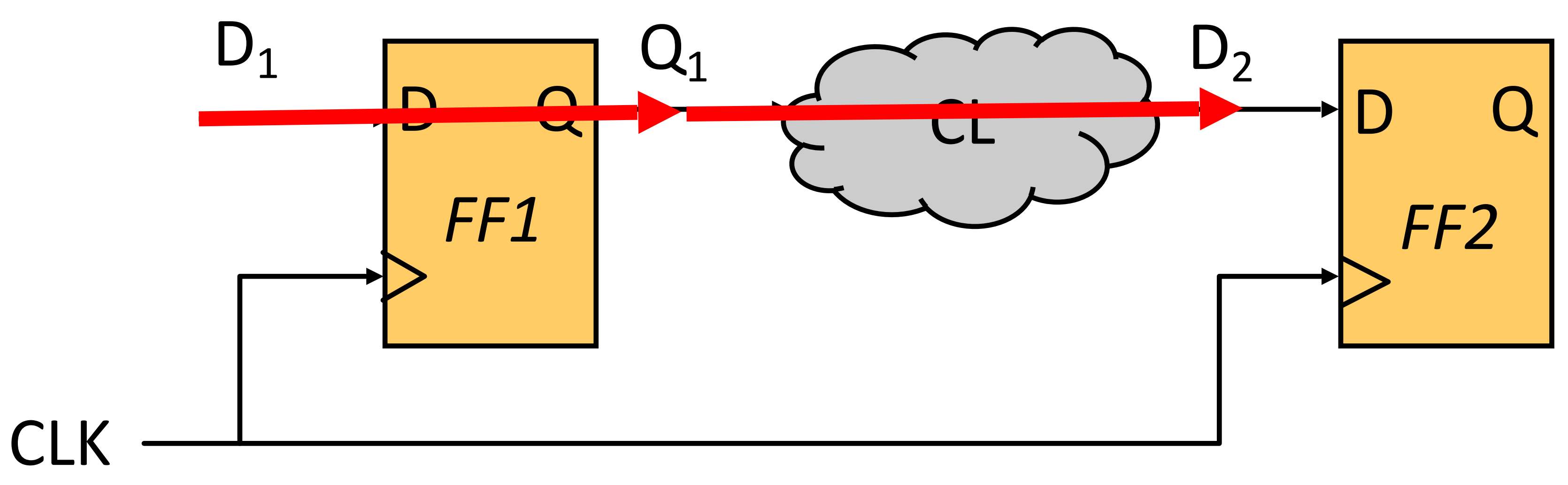
D Flip-Flop Timing (CLK Edge Trigger)

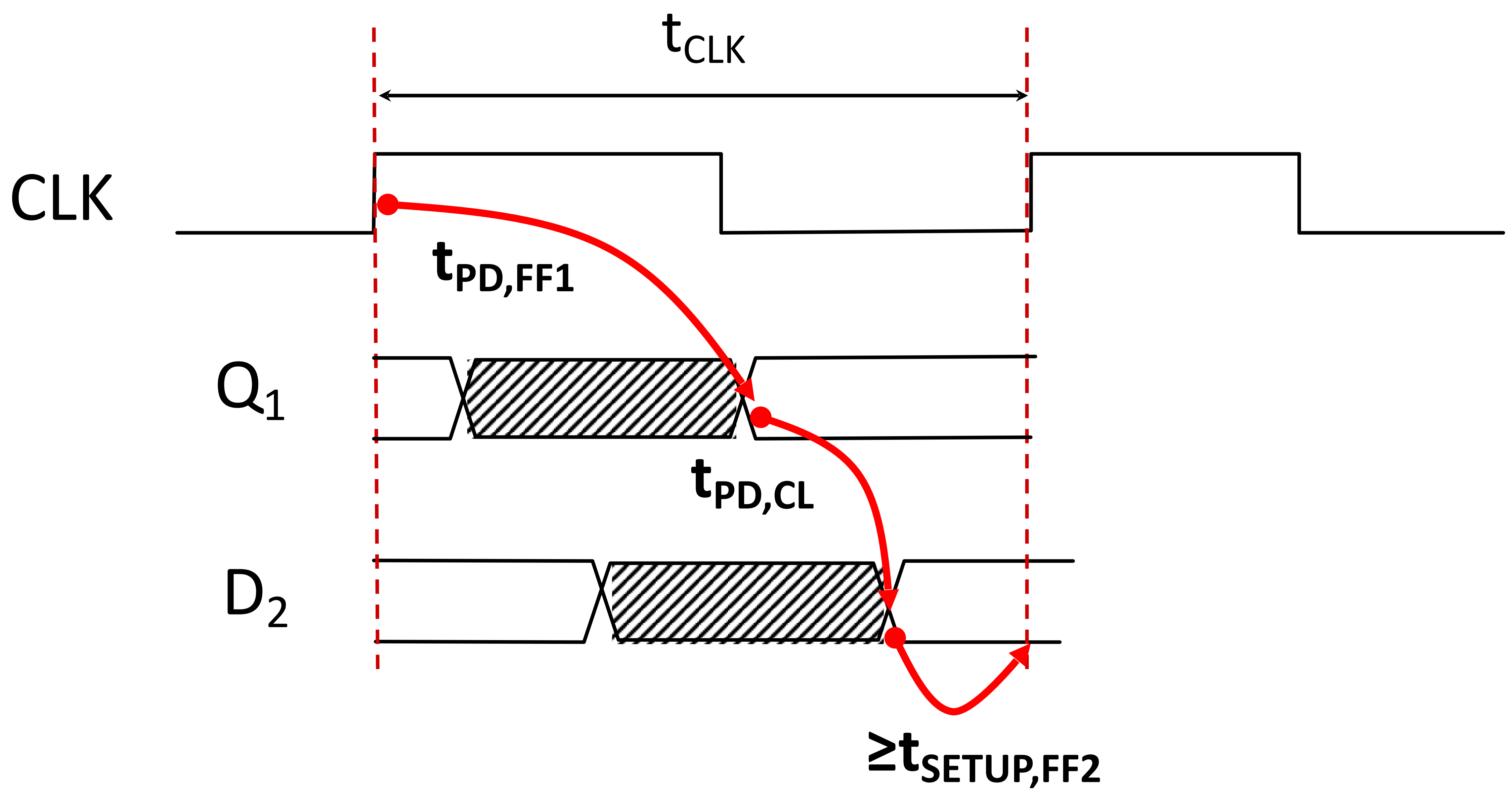




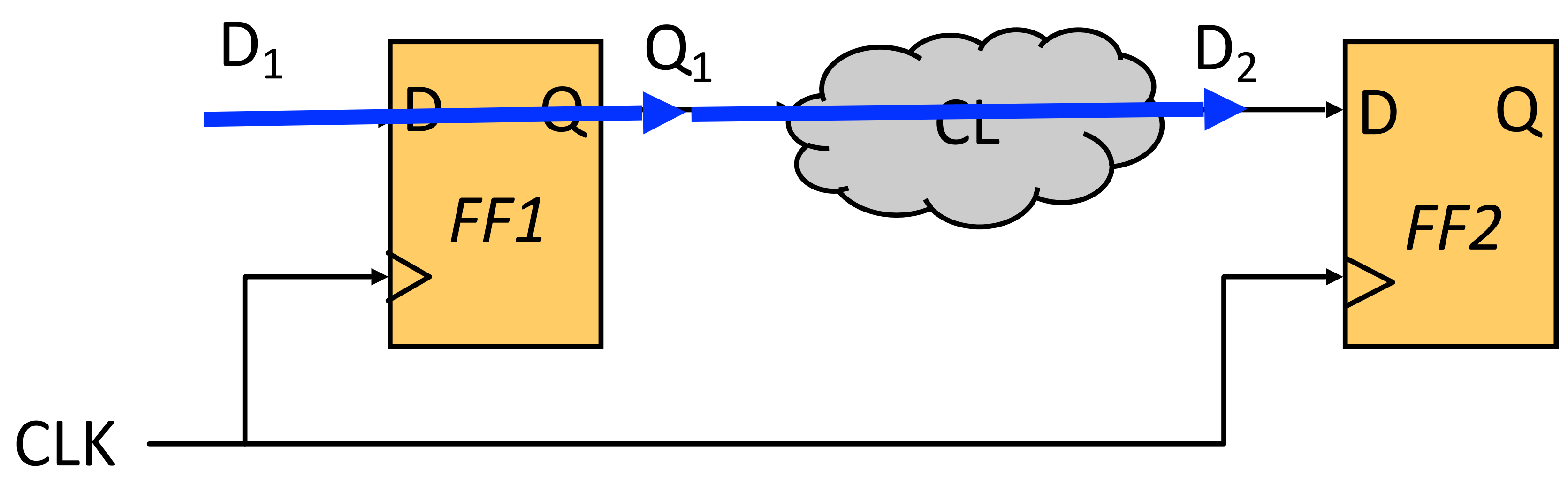
- Flip-flop input D should not change around the rising edge of the clock to avoid ***metastability***
- Formally, D should be a stable and valid digital value:
  - For at least t_{SETUP} before the rising edge of the clock
  - For at least t_{HOLD} after the rising edge of the clock
- Violating the timing constraints leaves the circuit in a metastability state. A contaminated value will be loaded into the register.

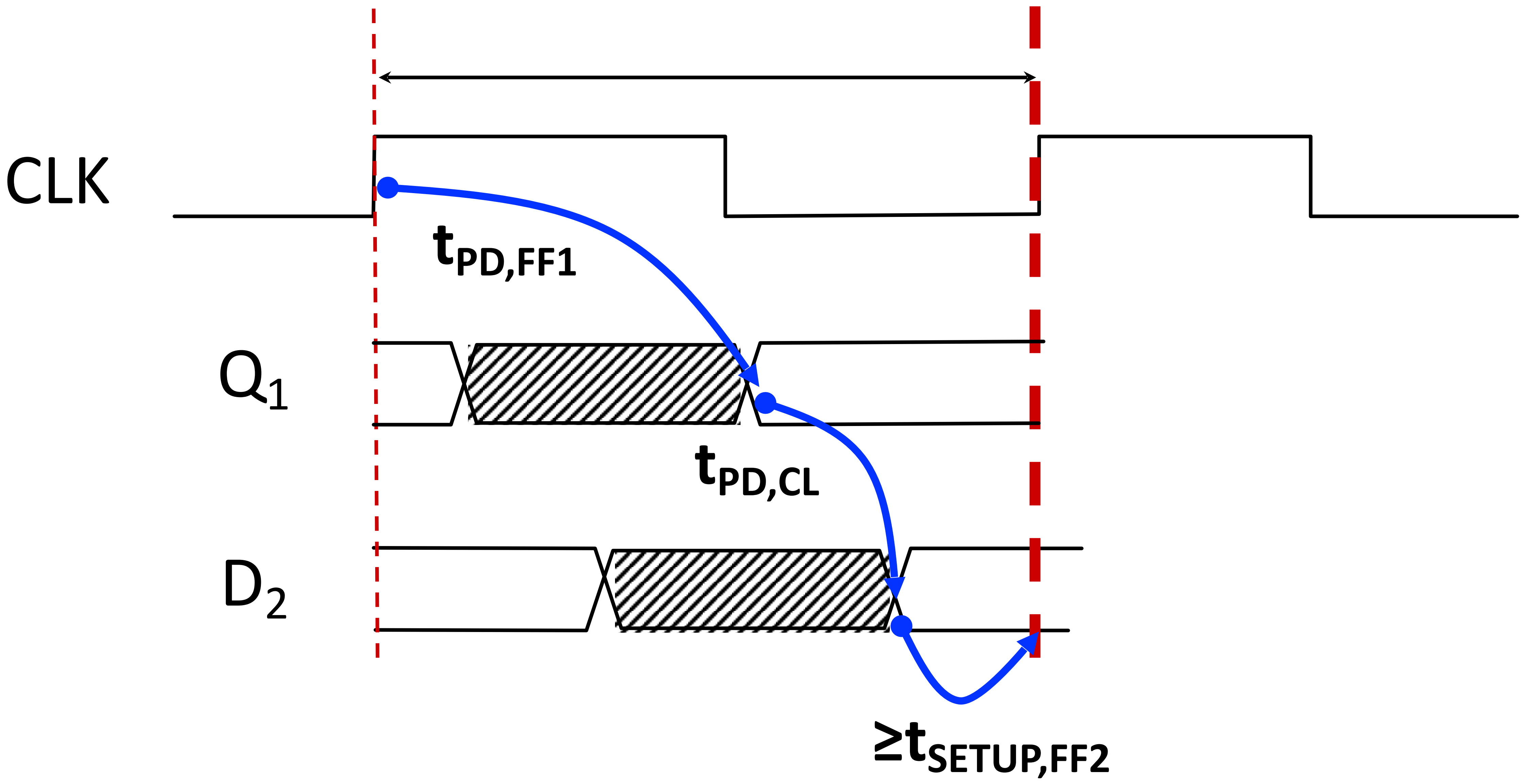
Sequential Circuit Timing (Setup Time)





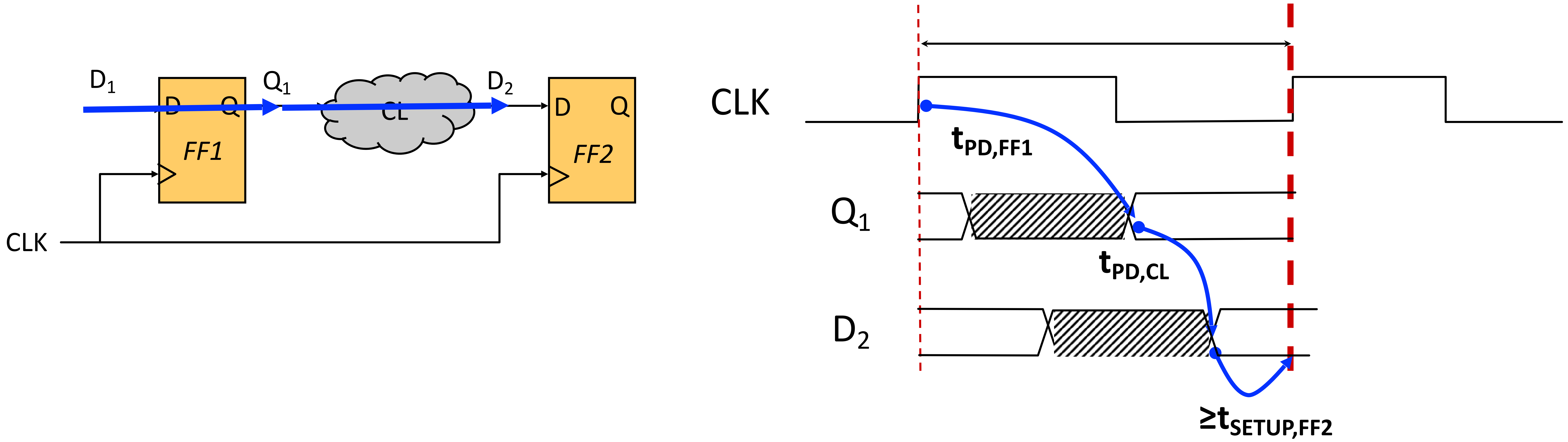
Fault Injection Attacks





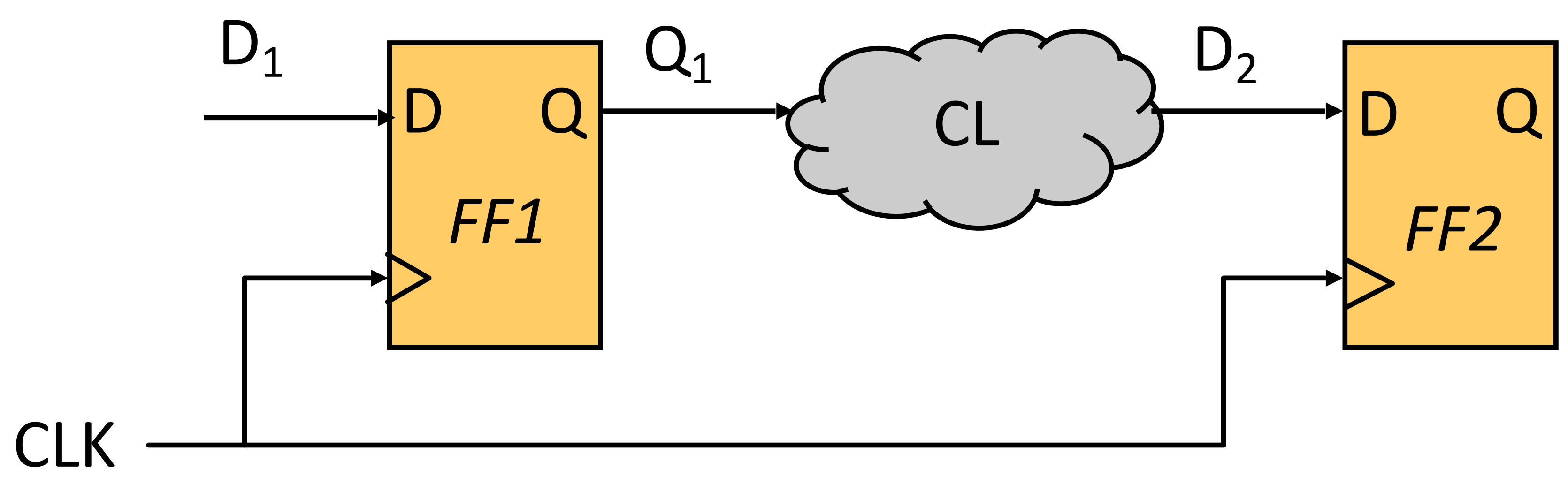
- What will happen if switch to a faster clock?

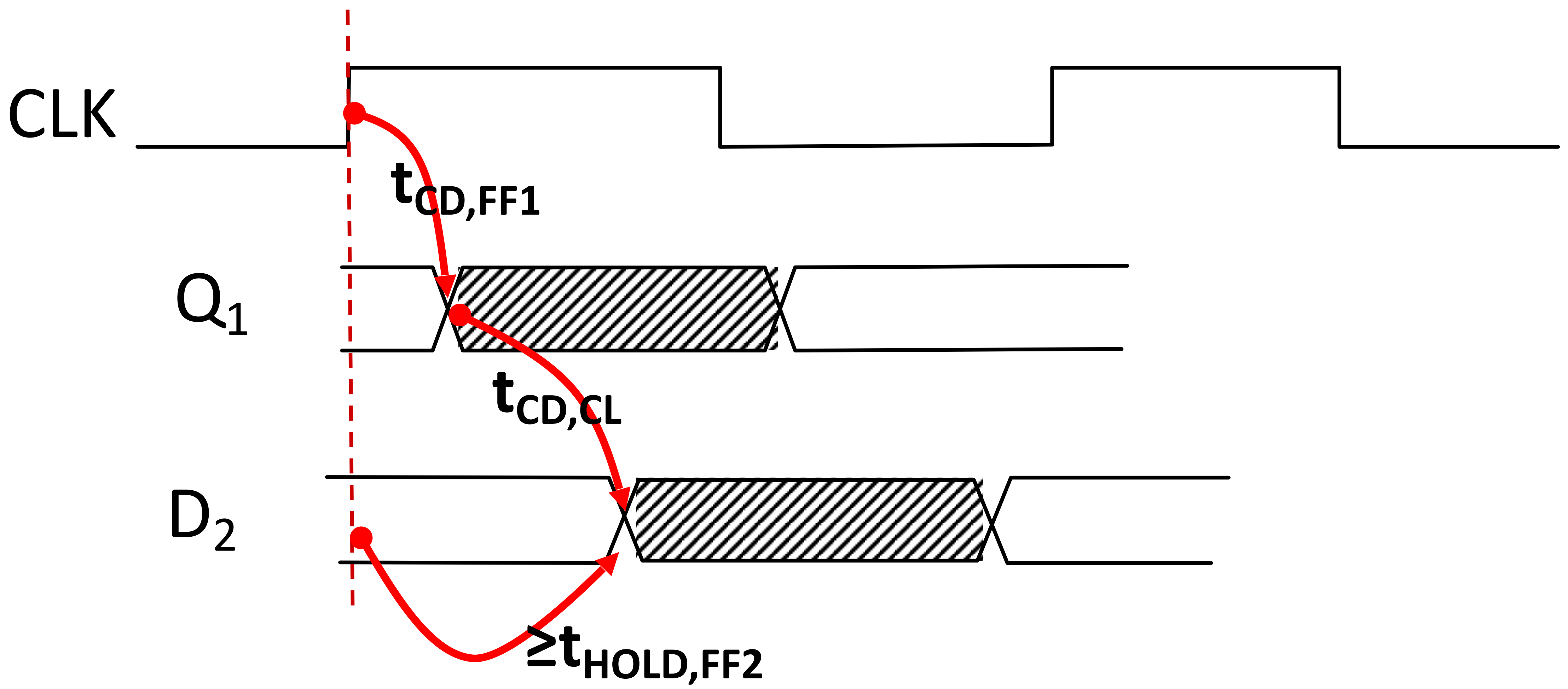
Fault Injection Attacks



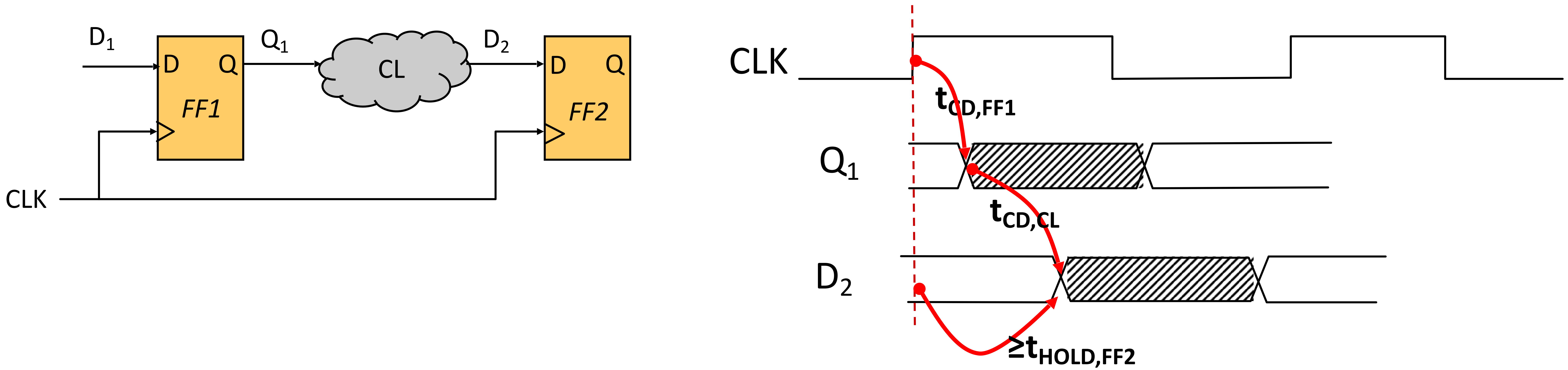
- What if when you decrease the voltage, the propagation delay becomes longer?

Sequential Circuit Timing (Hold Time)





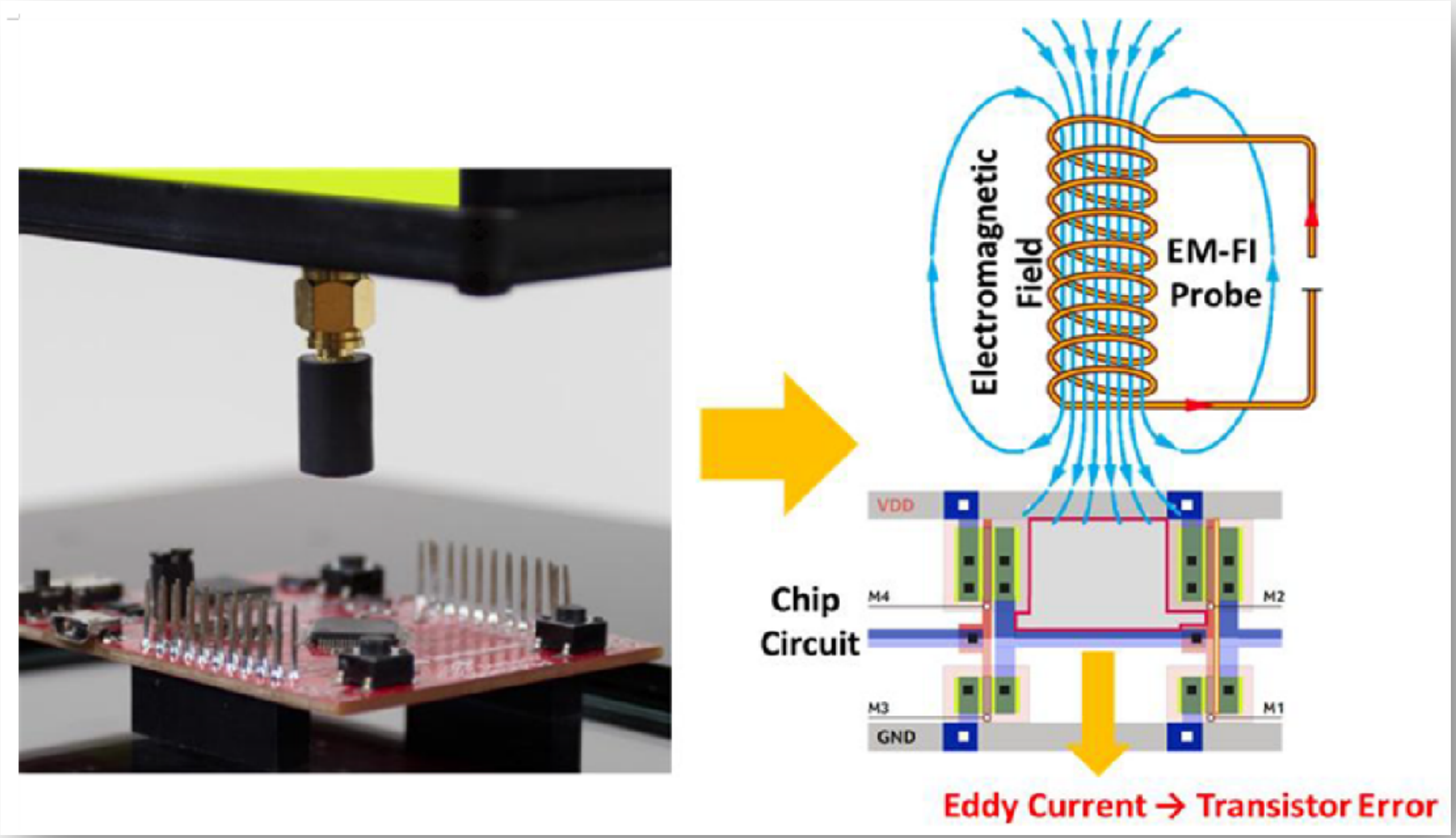
Voltage Glitching Attacks



- What if when increasing voltage, contamination time becomes shorter?

Other Variations

- Faults can also be triggered by EM and photonic signals.



Lim et al. Novel Fault Injection Attack without Artificial Trigger. Applied Science

Real-world Example and Challenges

- attack Xbox 360 with Reset Glitch attack
  - Goal: load our own kernel/hypervisor
  - Problem: the bootloader checks the hash of the kernel.
  - How: On Xbox360, a pin labeled as CPU_PLL_BYPASS to make CPU runs at a slower speed: 520kHz. When cpu runs at a slower speed, insert a short spike on the reset line of the CPU can cause fault to bypass the check.
  - Challenge: Need to know when to trigger the fault
    - Side channel
    - Reverse engineering the code
    - Keep trying

Mitigations?

Mitigations

- Reliability issues, so redundancy can rescue
  - Redundancy: detect a fault or recover from the fault -> two cores running the same thing
  - Example: Google OpenTitan, some automotive but for different reasons...
  - Problem: Expensive

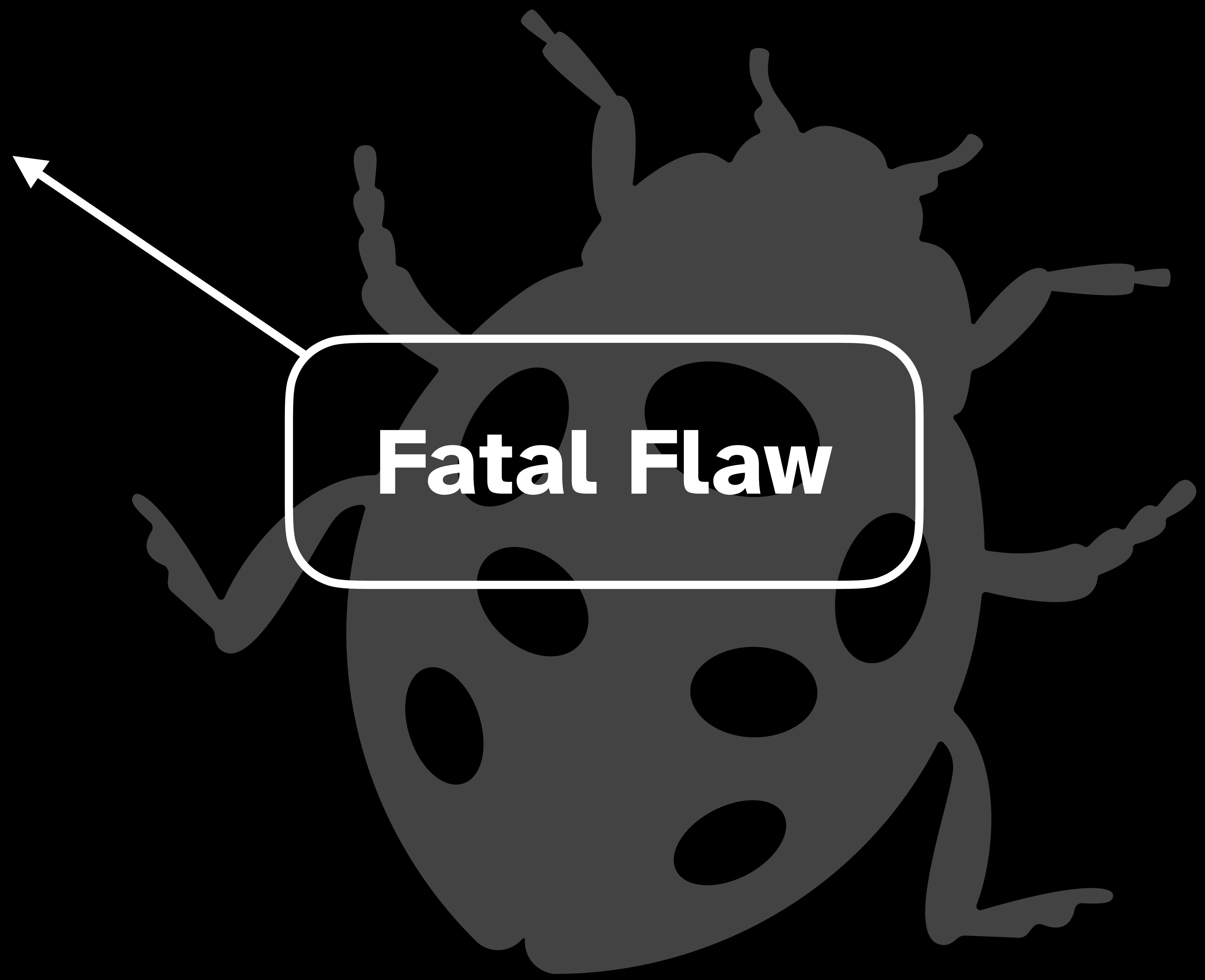
- The attack requires precise timing, so make it even more difficult
  - Non-deterministic: add randomization, so it becomes difficult for the attacker to know when to trigger the fault
  - Benefit: increase the time cost, also reduce the scalability of the attack.

Spot the Bug

```
bool memcmp (char *buf1, char *buf2, size_t len) {
    for (int i = 0; i < len; i++) {
        if (buf1[i] != buf2[i]) {
            return false;
        }
    }
    return true;
}
```

Spot the Bug

```
bool memcmp (char *buf1, char *buf2, size_t len) {
    for (int i = 0; i < len; i++) {
        if (buf1[i] != buf2[i]) {
            return false;
        }
    }
    return true;
}
```



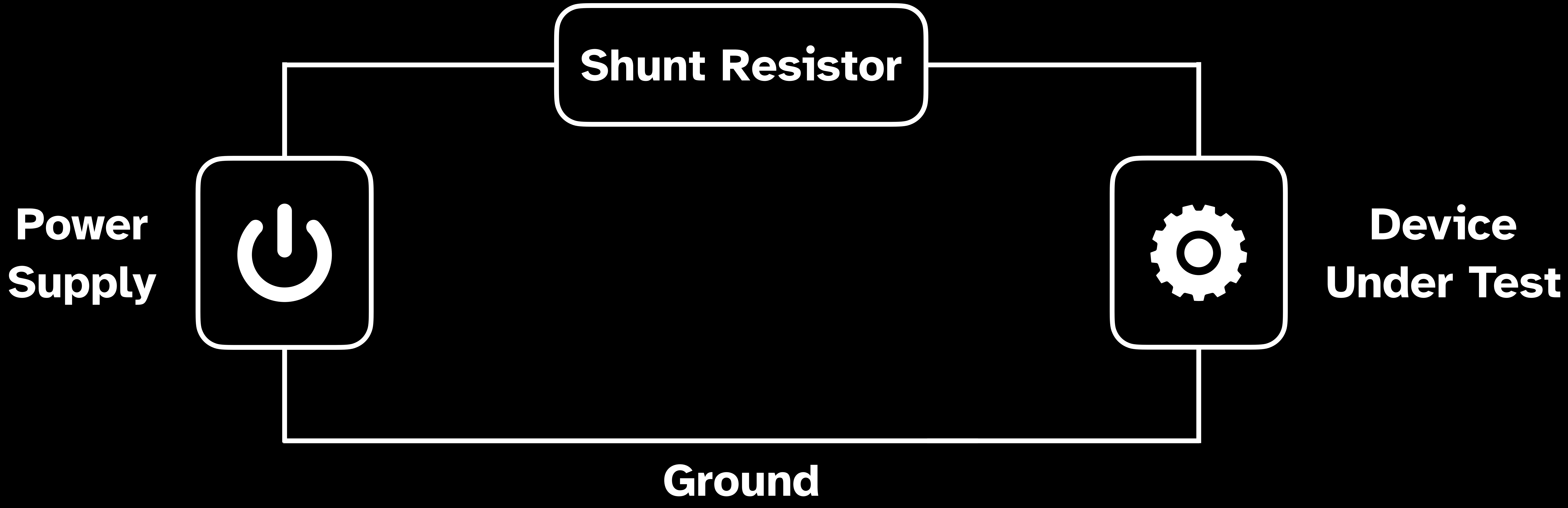
Demo 3

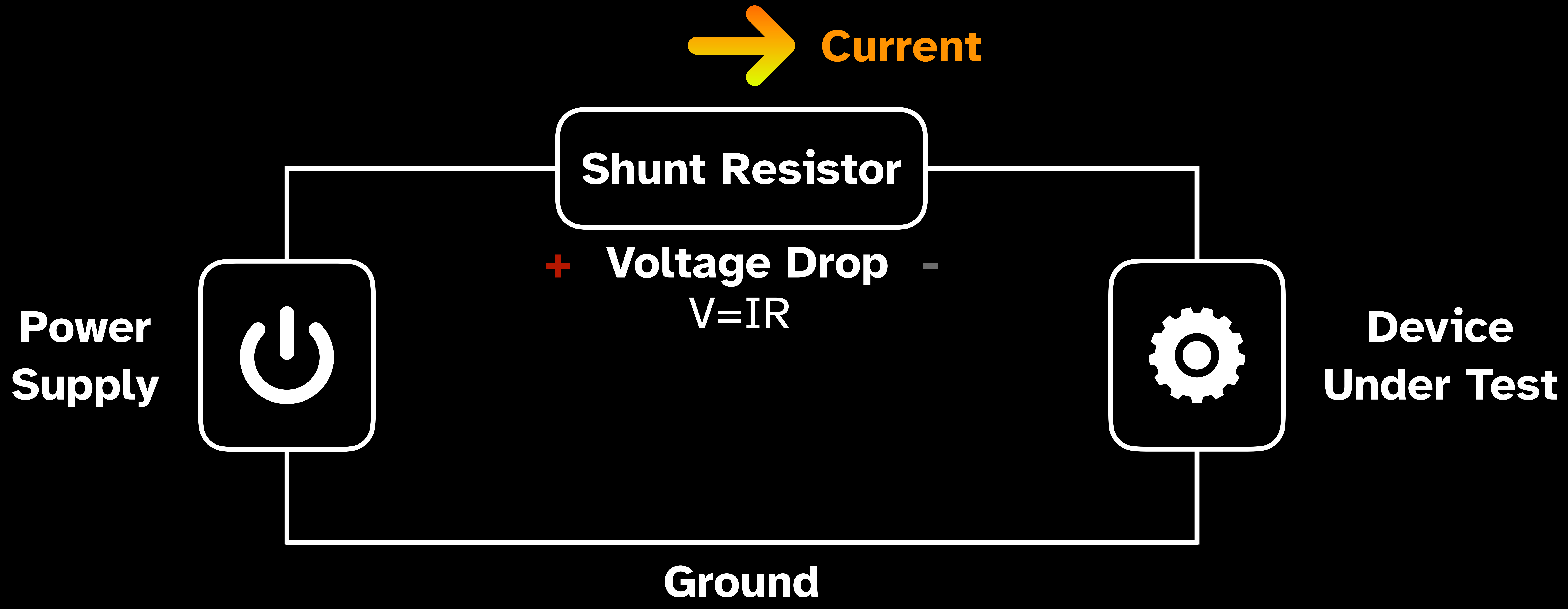
"What if we closely inspect the timing of a **`memcmp`**?"

No Demo:
You will do this in recitation next week!

Power Analysis

Power = Voltage x Current


Shunt Resistor
Power Supply
Device Under Test
Ground


Current
Shunt Resistor
+ Voltage Drop -
V=IR
Power Supply
Device Under Test
Ground

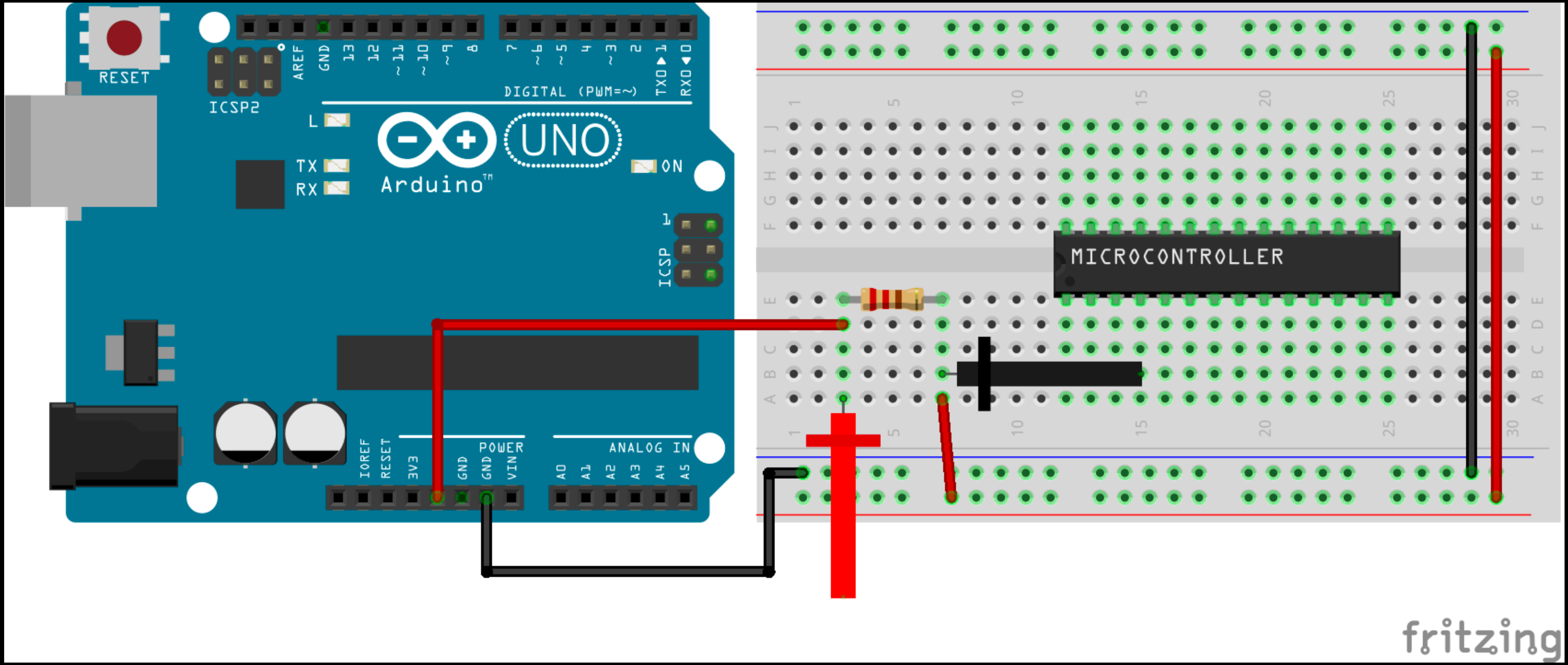
How can you measure current on an oscilloscope?

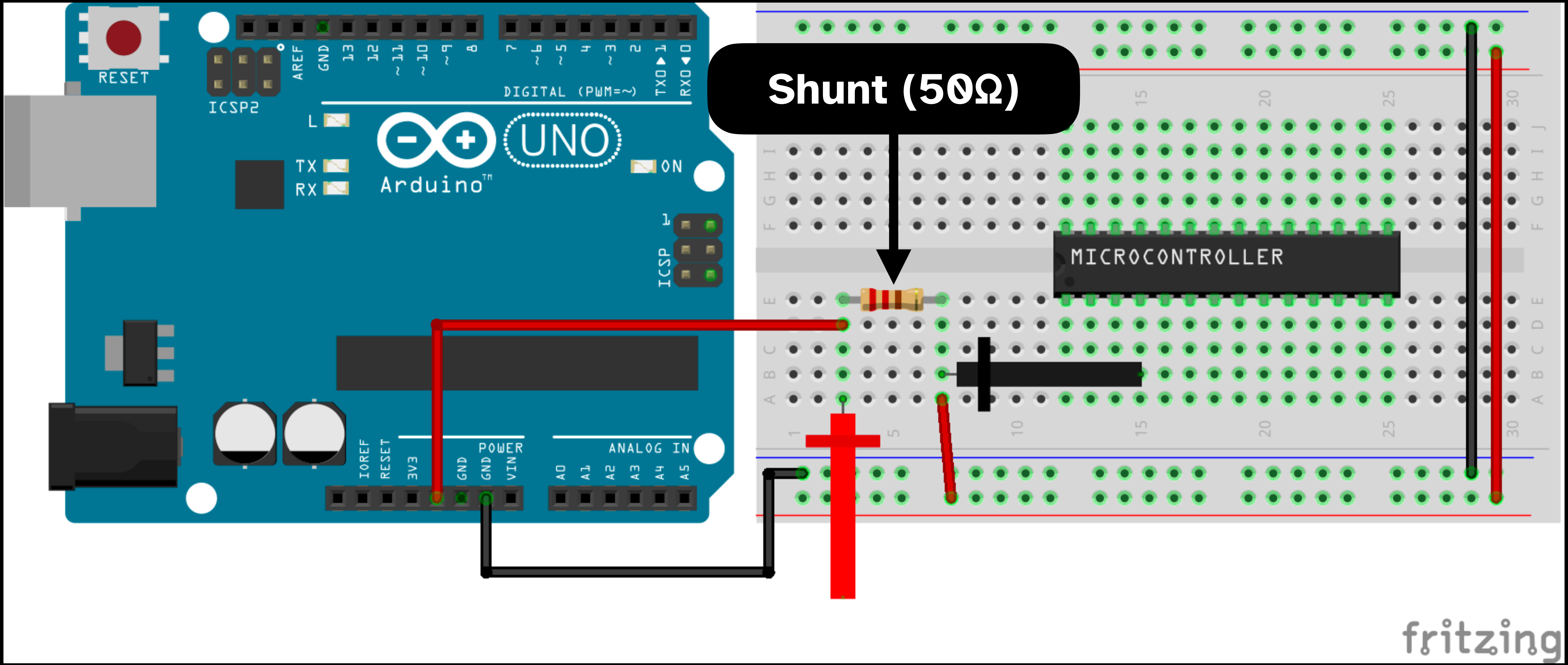
Apply Ohm's Law

Voltage (V) = Current (I) * Resistance (R)

Or in other words,

$$I = V / R$$


RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX▶1
RX◀0
DIGITAL (PWM=~)
L
TX
RX
UNO
Arduino™
ON
ICSP
1
IOREF
RESET
3V3
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
MICROCONTROLLER
fritzing


Shunt (50Ω)
RESET
ICSP2
AREF
GND
DIGITAL (PWM=~)
L
TX
RX
UNO
Arduino
ON
ICSP
IOREF
RESET
3V3
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
MICROCONTROLLER
fritzing

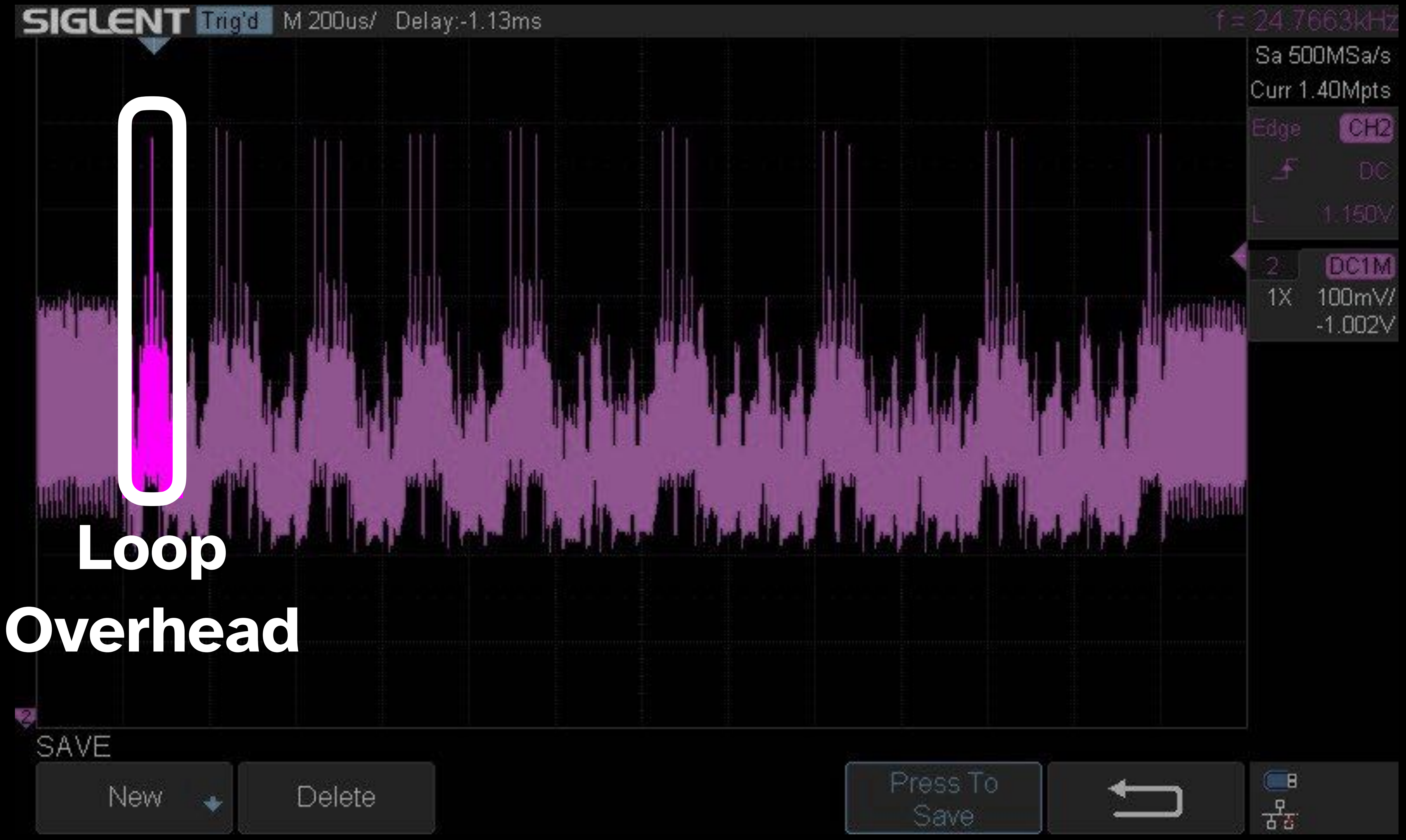
RSA Modular Exponentiation


```
int rsa_modExp(int b, int e, int m) {
  int product = 1;
  b = b % m;
  while ( e > 0){
    if (e & 1){
      product = modmult(product, b, m);
    }
    b = modmult(b, b, m);

    e >>= 1;
  }
  return product;
}
```

RSA Modular Exponentiation


```
int rsa_modExp(int b, int e, int m) {
  int product = 1;
  b = b % m;
  while ( e > 0){
    if (e & 1){
      product = modmult(product, b, m);
    }
    b = modmult(b, b, m);

    e >>= 1;
  }
  return product;
}
```

RSA Modular Exponentiation




```
int rsa_modExp(int b, int e, int m) {
  int product = 1;
  b = b % m;
  while ( e > 0){
    if (e & 1){
      product = modmult(product, b, m);
    }
    b = modmult(b, b, m);

    e >>= 1;
  }
  return product;
}
```

RSA Modular Exponentiation

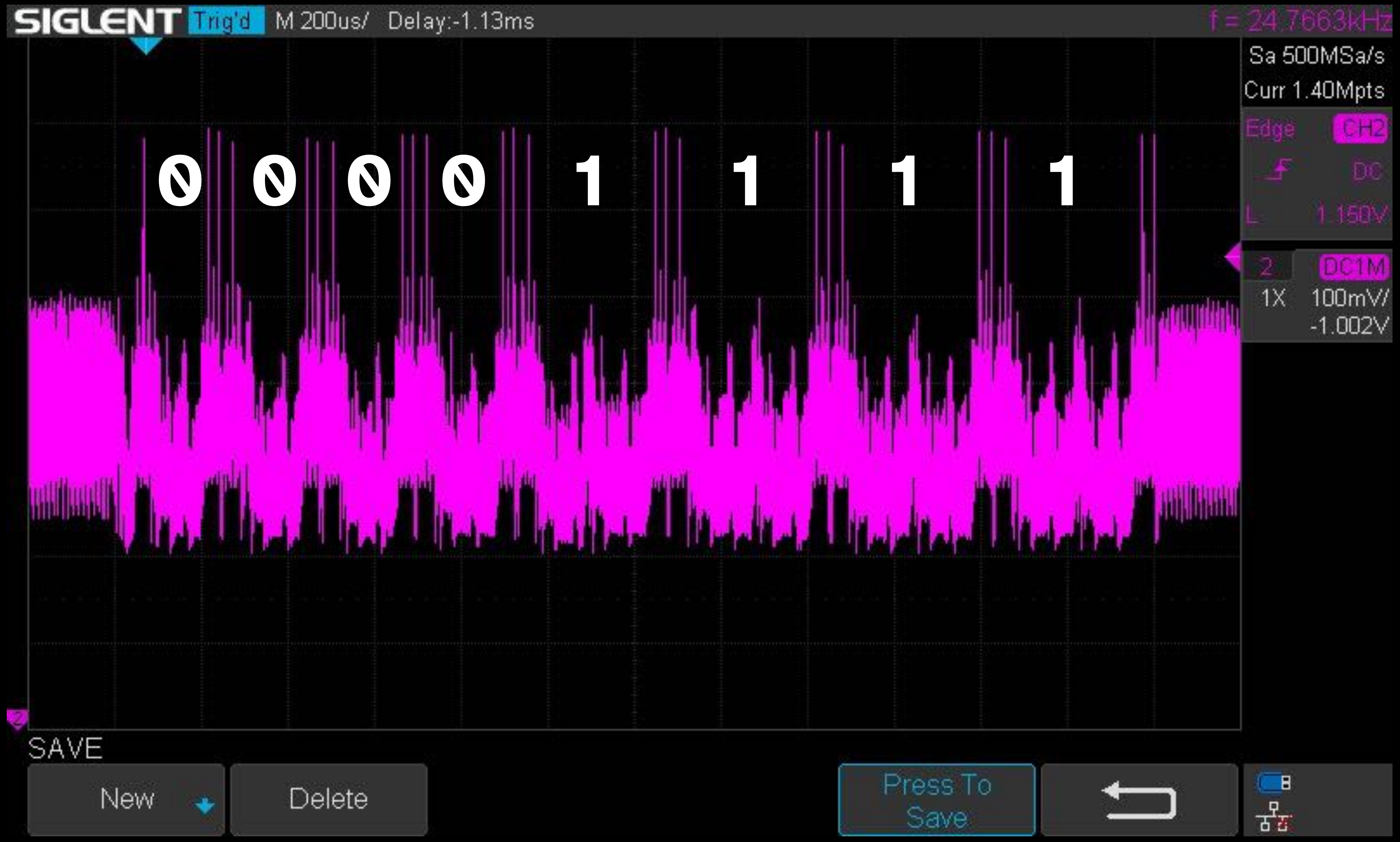
1 call
to modmult

```
int rsa_modExp(int b, int e, int m) {
  int product = 1;
  b = b % m;
  while ( e > 0){
    if (e & 1){
      product = modmult(product, b, m);
    }
    b = modmult(b, b, m);

    e >>= 1;
  }
  return product;
}
```

RSA Modular Exponentiation


```
int rsa_modExp(int b, int e, int m) {
  int product = 1;
  b = b % m;
  while ( e > 0){
    if (e & 1){
      product = modmult(product, b, m);
    }
    b = modmult(b, b, m);

    e >>= 1;
  }
  return product;
}
```

RSA Modular Exponentiation




```
int rsa_modExp(int b, int e, int m) {
  int product = 1;
  b = b % m;
  while ( e > 0){
    if (e & 1){
      product = modmult(product, b, m);
    }
    b = modmult(b, b, m);

    e >>= 1;
  }
  return product;
}
```

e = 0xf0

Demo 4

"What if we watch the chip's current draw?"

Physical Attack Mitigation Case Study

- IBM 4758
- Satisfy FIPS 140-1 Level 4

1.4 Security Level 4

Security Level 4 provides the highest level of security. Although most existing products do not meet this level of security, some products are commercially available which meet many of the Level 4 requirements. Level 4 physical security provides an envelope of protection around the cryptographic module. Whereas the tamper detection circuits of lower level modules may be bypassed, the intent of Level 4 protection is to detect a penetration of the device from any direction. For example, if one attempts to cut through the enclosure of the cryptographic module, the attempt should be detected and all critical security parameters should be zeroized. Level 4 devices are particularly useful for operation in a physically unprotected environment where an intruder could possibly tamper with the device.

Photo of IBM 4758 Cryptographic Coprocessor (courtesy of Steve Weingart) from *https://www.cl.cam.ac.uk/~rnc1/descrack/ibm4758.html*

Physical Tamper Resistance

- Make it difficult for the attackers to get access to PCB

Tampering Detection

Robust metal enclosures.
Open the lid → disconnect power supply

↓

Drill through the lid

↓

Photocells and tilt devices

Tampering Evident

“potting" the device in a block of epoxy resin

↓

Patience: Scrape away the epoxy

↓

tamper-sensing barriers:
nichrome wire wound around the device

IBM 4758 Secure Co-Processor

- Clock glitching:
  - use phase locked loops and independently generated internal clocks
- Voltage glitching:
  - Add detection and monitor circuits to watch voltage changes
- X-ray fault injection
  - a radiation sensor
- Power side channels
  - Solid aluminium shielding and a low-pass filter (a Faraday cage)

Photo of IBM 4758 Cryptographic Coprocessor (courtesy of Steve Weingart) from *https://www.cl.cam.ac.uk/~rnc1/descrack/ibm4758.html*

Expensive. Other secure processors only focus on a limited set of physical attacks.


Expensive
Focused Ion Beam
Optical Imaging
Laser Etching
Non-Invasive
Invasive
IC Decapping
Power Tracing
Fault Injection
Logical Interfaces
Cheap

Next:
Physical Attacks CTF