AFFORDABLE EMFI ATTACKS

AGAINST MODERN IOT CHIPS

DAVIDE TOLDO
SECURE MOBILE NETWORKING LAB - SEEMOO
TECHNICAL UNIVERSITY OF DARMSTADT, GERMANY

AGENDA

- **Motivation**
- **Introduction into EMFI**
- **Setup**
- **Exemplary Results**

MOTIVATION

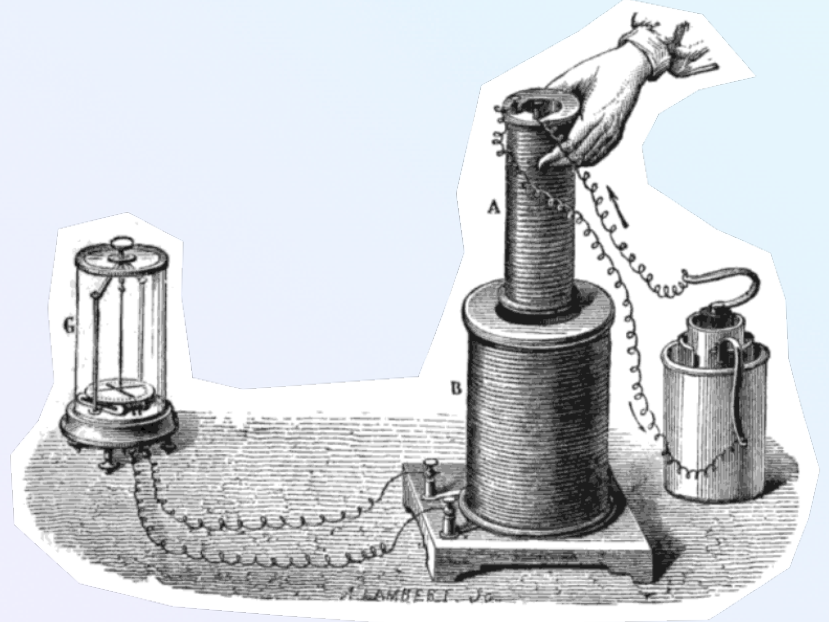
- Modern security features == classic attacks obsolete ⁉️😰
- Firmware experimentation severely limited
- Fault injection: bypass chosen instructions ➡️➡️➡️ Profit?

https://commons.wikimedia.org/wiki/File:Segger_J-Link_PRO.jpg

- Relatively new field, many devices have no countermeasures
- Tools and setup for such attacks not fully open-source / freely available / accessible

INTRODUCTION INTO EMFI

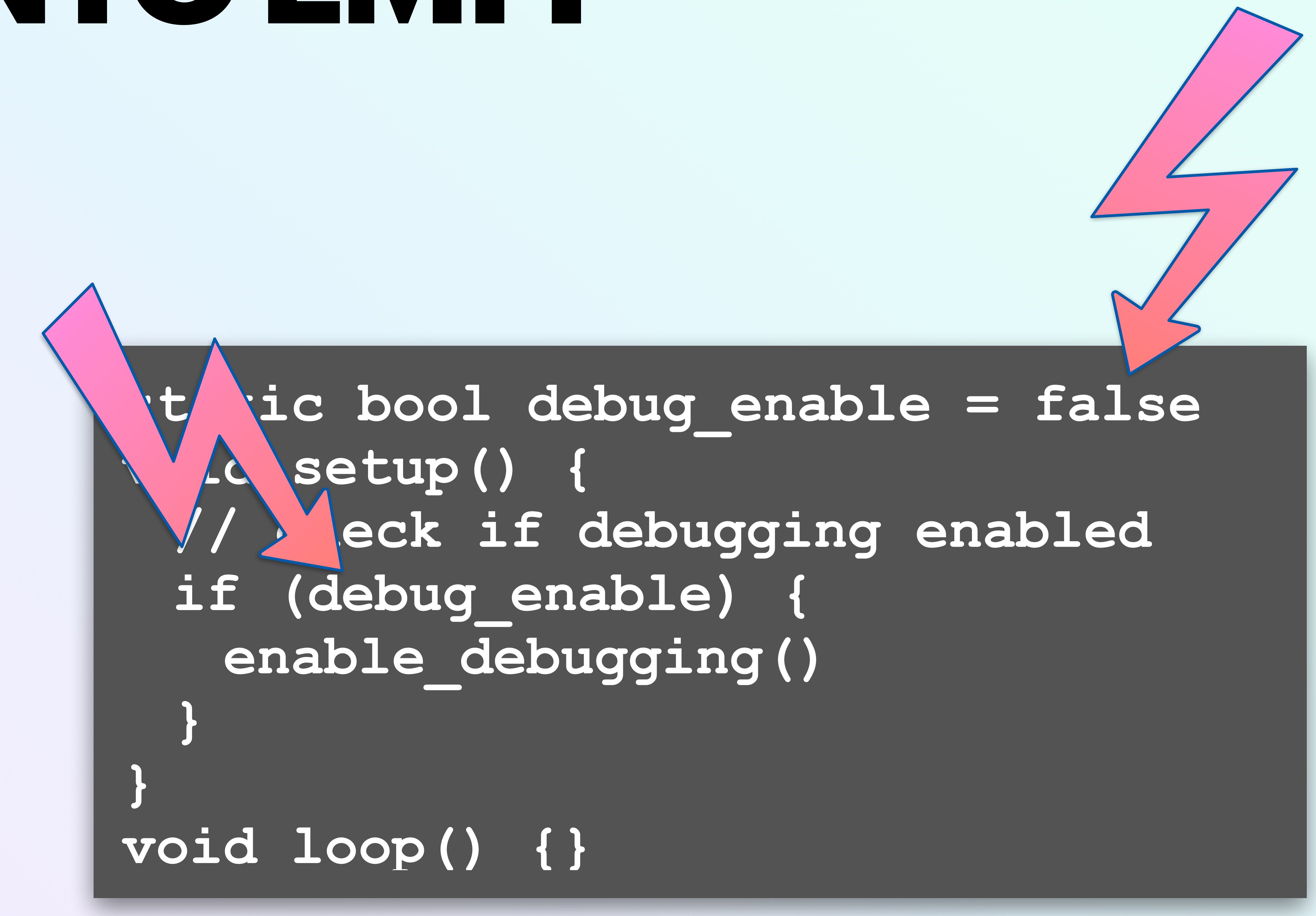
- Fault injection: introduce faults into a system to force it to behave in unintended way
- Physical FI: affect chip's internal behavior through external conditions
- 👾 EMFI: electromagnetic pulses 🧲 on SoC's and memory 👉 induce currents
  👉 affect transistor behavior
  👉 change execution path



(c) Arthur William Poyser (1892)

INTRODUCTION INTO EMFI

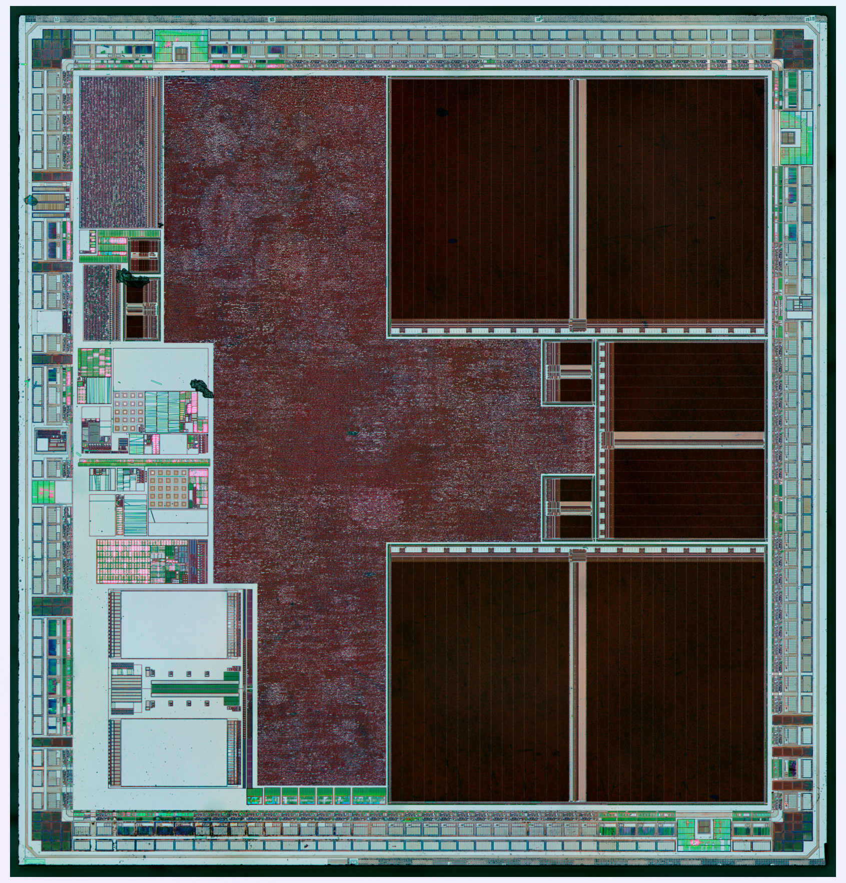
- Mitigations - if deployed at all - typically require expensive hardware changes (new revision of board / component)
- Impact of physical FI attacks is limited (require physical proximity to the target)
- However: well suited for firmware research on locked-down targets



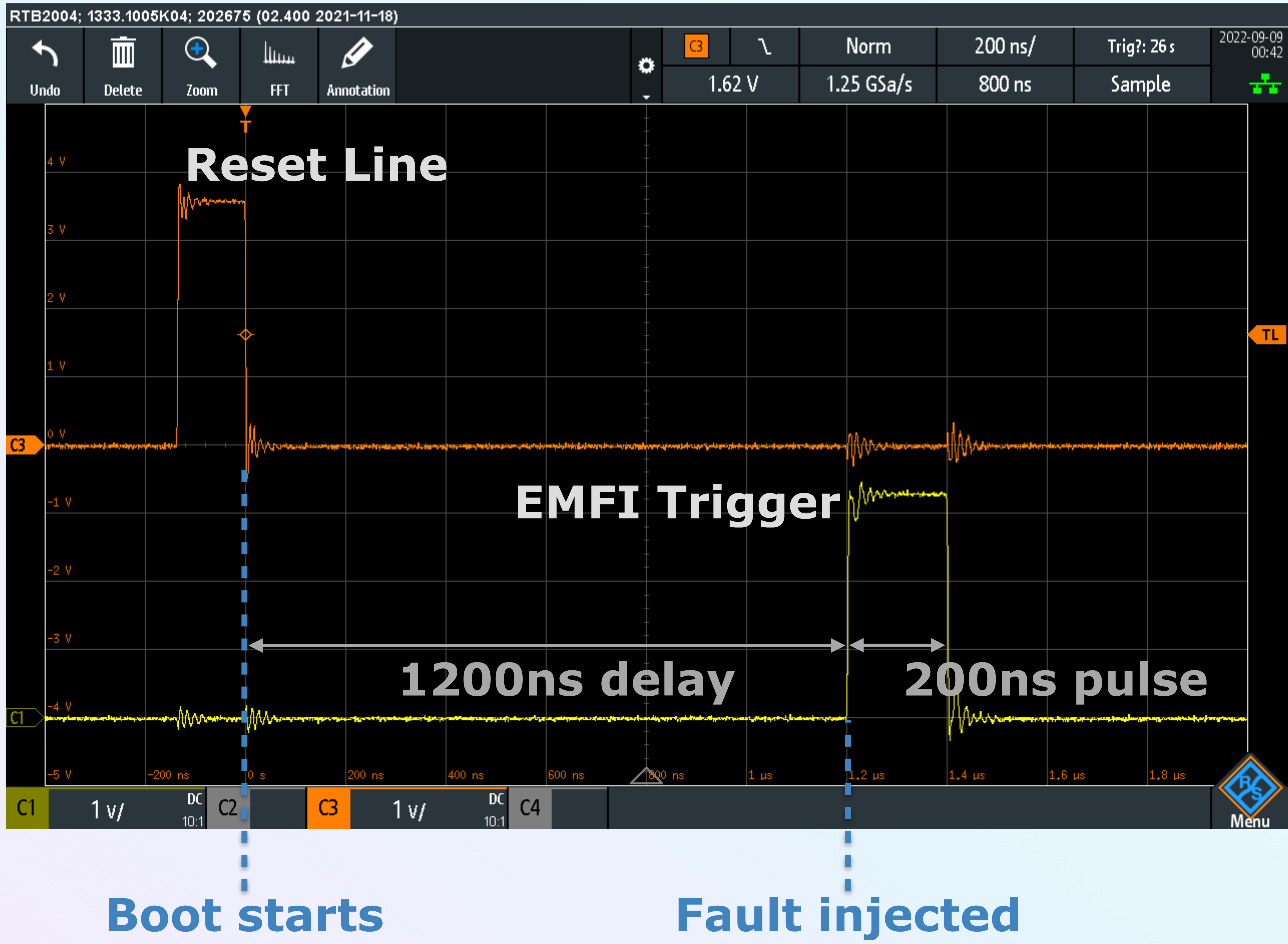
```
static bool debug_enable = false
void setup() {
  // check if debugging enabled
  if (debug_enable) {
    enable_debugging()
  }
}
void loop() {}
```

INTRODUCTION INTO EMFI

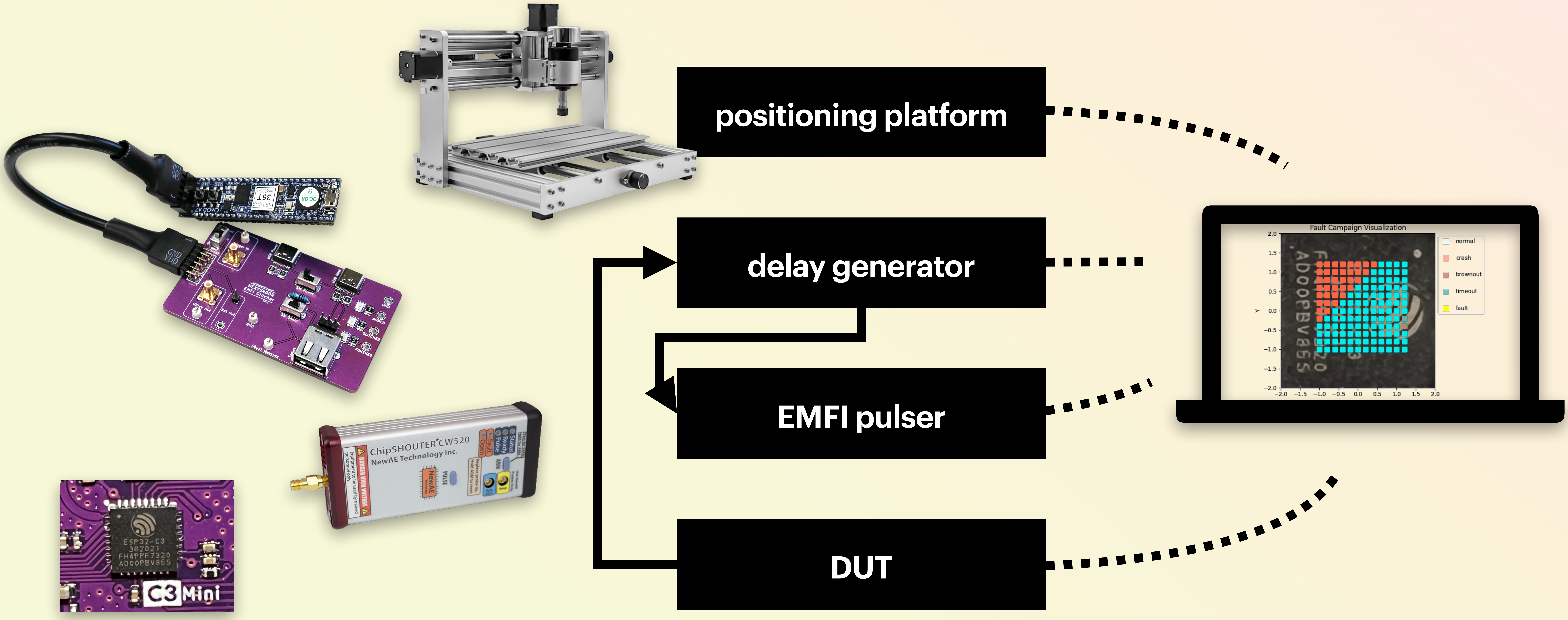
- Location & timing essential: fault exactly at the desired instruction and SoC area
- FPGA: 400MHz = 2.5ns steps
- Code, binary and power trace analysis help discover timing for potential fault



https://commons.wikimedia.org/wiki/File:GD32F103CBT6-Si-HD.jpg



EMFI SETUP 🎰



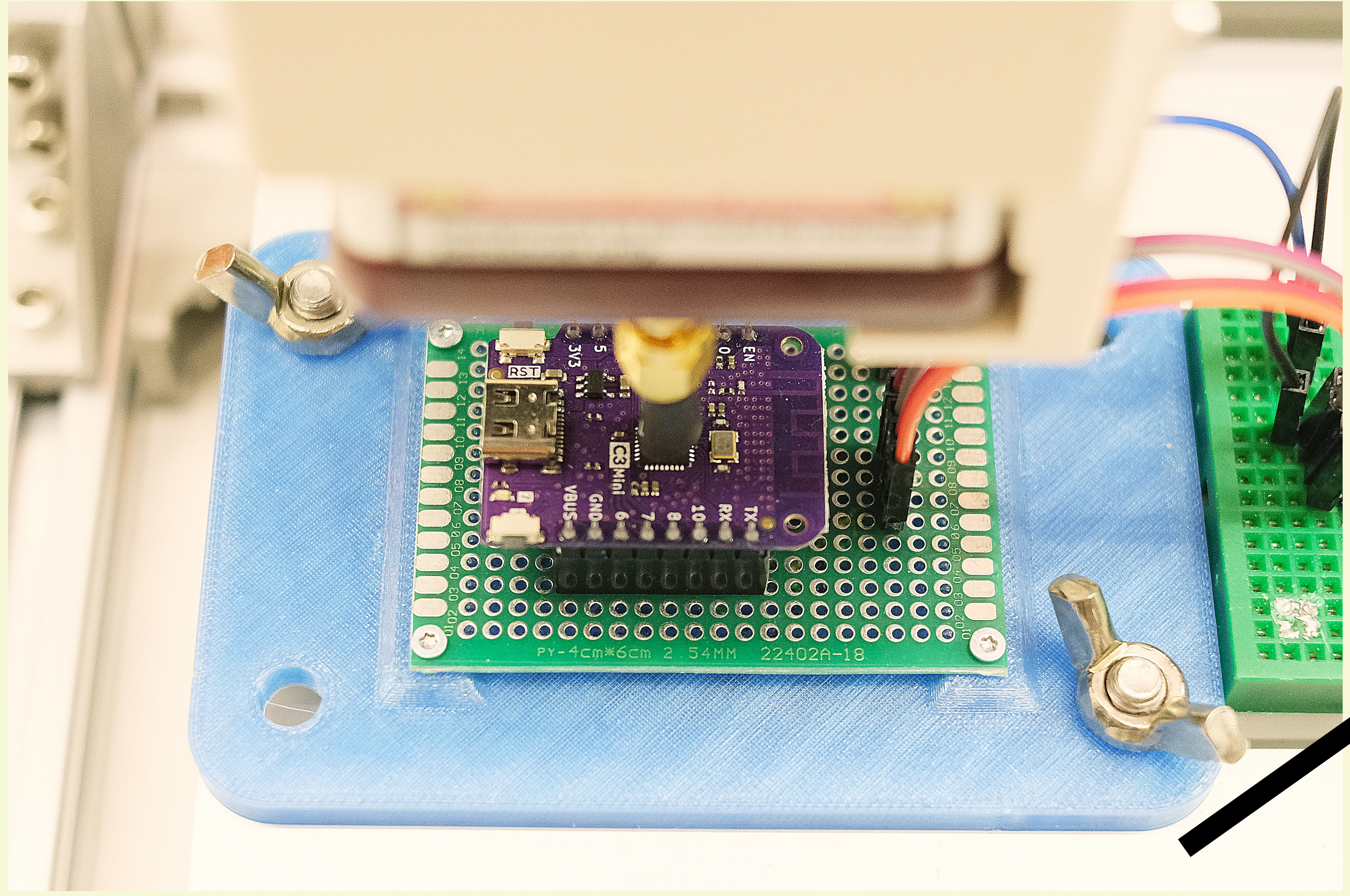
positioning platform

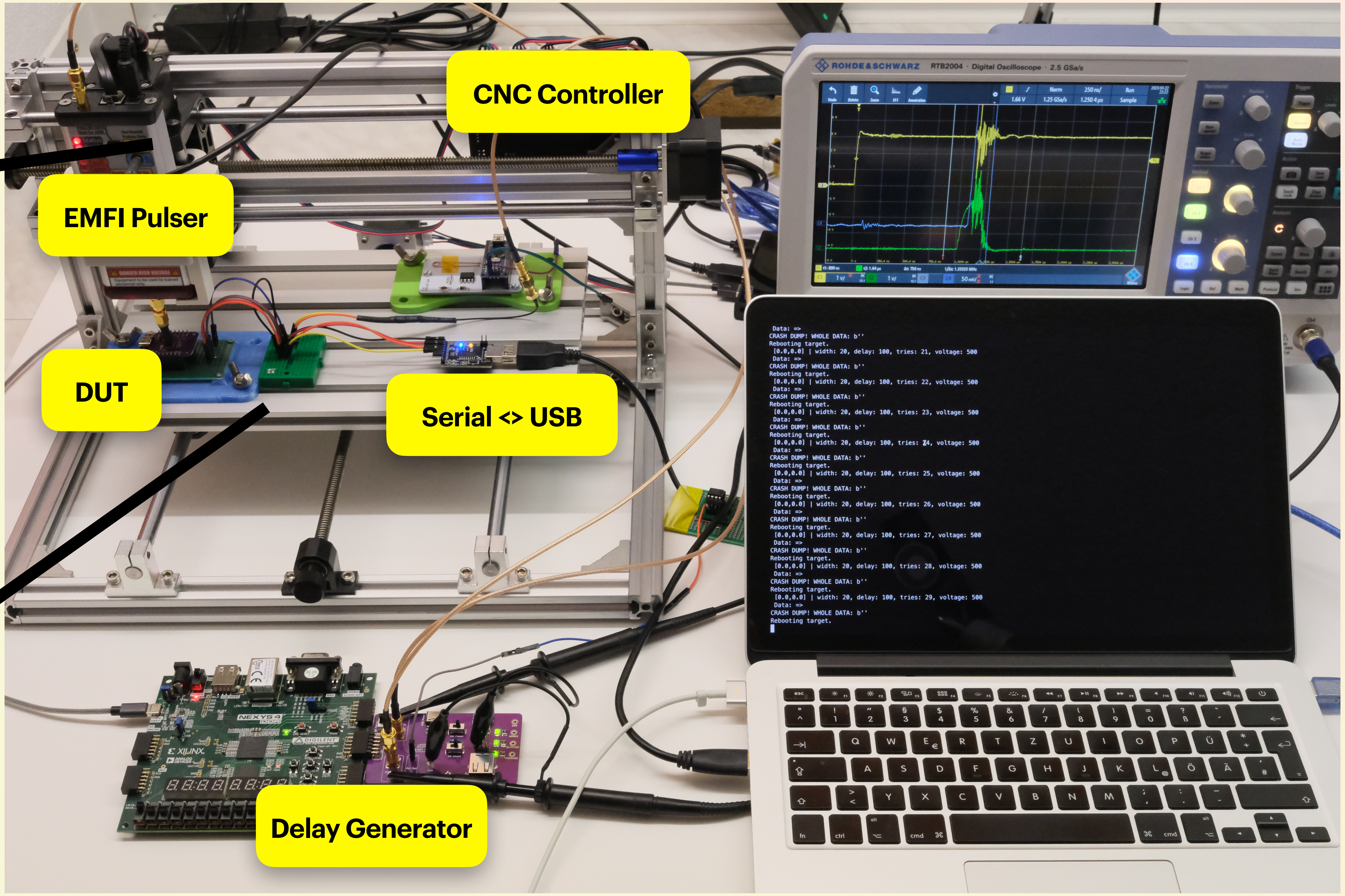
delay generator

EMFI pulser

DUT

EMFI SETUP 🎰



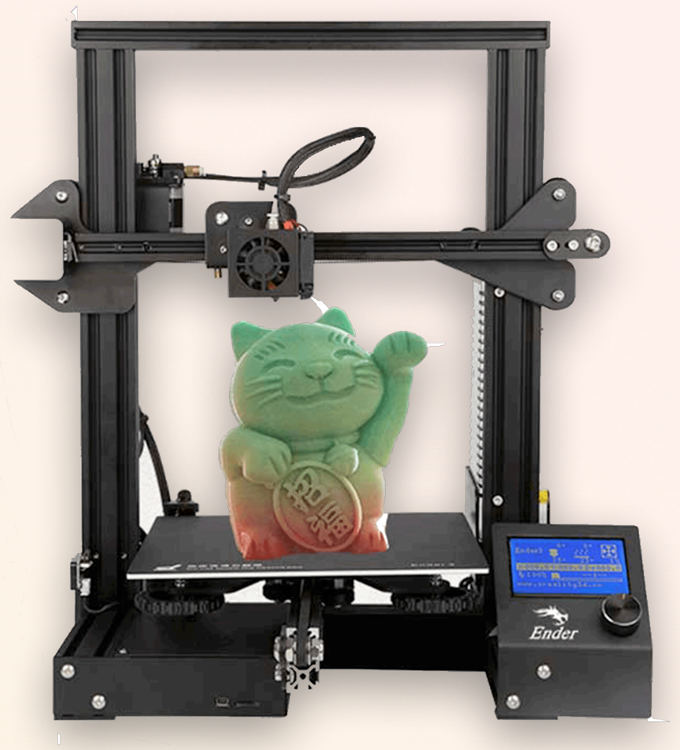


```
 Data: =>
CRASH DUMP! WHOLE DATA: b''
Rebooting target.
 [0.0,0.0] | width: 20, delay: 100, tries: 21, voltage: 500
 Data: =>
CRASH DUMP! WHOLE DATA: b''
Rebooting target.
 [0.0,0.0] | width: 20, delay: 100, tries: 22, voltage: 500
 Data: =>
CRASH DUMP! WHOLE DATA: b''
Rebooting target.
 [0.0,0.0] | width: 20, delay: 100, tries: 23, voltage: 500
 Data: =>
CRASH DUMP! WHOLE DATA: b''
Rebooting target.
 [0.0,0.0] | width: 20, delay: 100, tries: 24, voltage: 500
 Data: =>
CRASH DUMP! WHOLE DATA: b''
Rebooting target.
 [0.0,0.0] | width: 20, delay: 100, tries: 25, voltage: 500
 Data: =>
CRASH DUMP! WHOLE DATA: b''
Rebooting target.
 [0.0,0.0] | width: 20, delay: 100, tries: 26, voltage: 500
 Data: =>
CRASH DUMP! WHOLE DATA: b''
Rebooting target.
 [0.0,0.0] | width: 20, delay: 100, tries: 27, voltage: 500
 Data: =>
CRASH DUMP! WHOLE DATA: b''
Rebooting target.
 [0.0,0.0] | width: 20, delay: 100, tries: 28, voltage: 500
 Data: =>
CRASH DUMP! WHOLE DATA: b''
Rebooting target.
 [0.0,0.0] | width: 20, delay: 100, tries: 29, voltage: 500
 Data: =>
CRASH DUMP! WHOLE DATA: b''
Rebooting target.
```

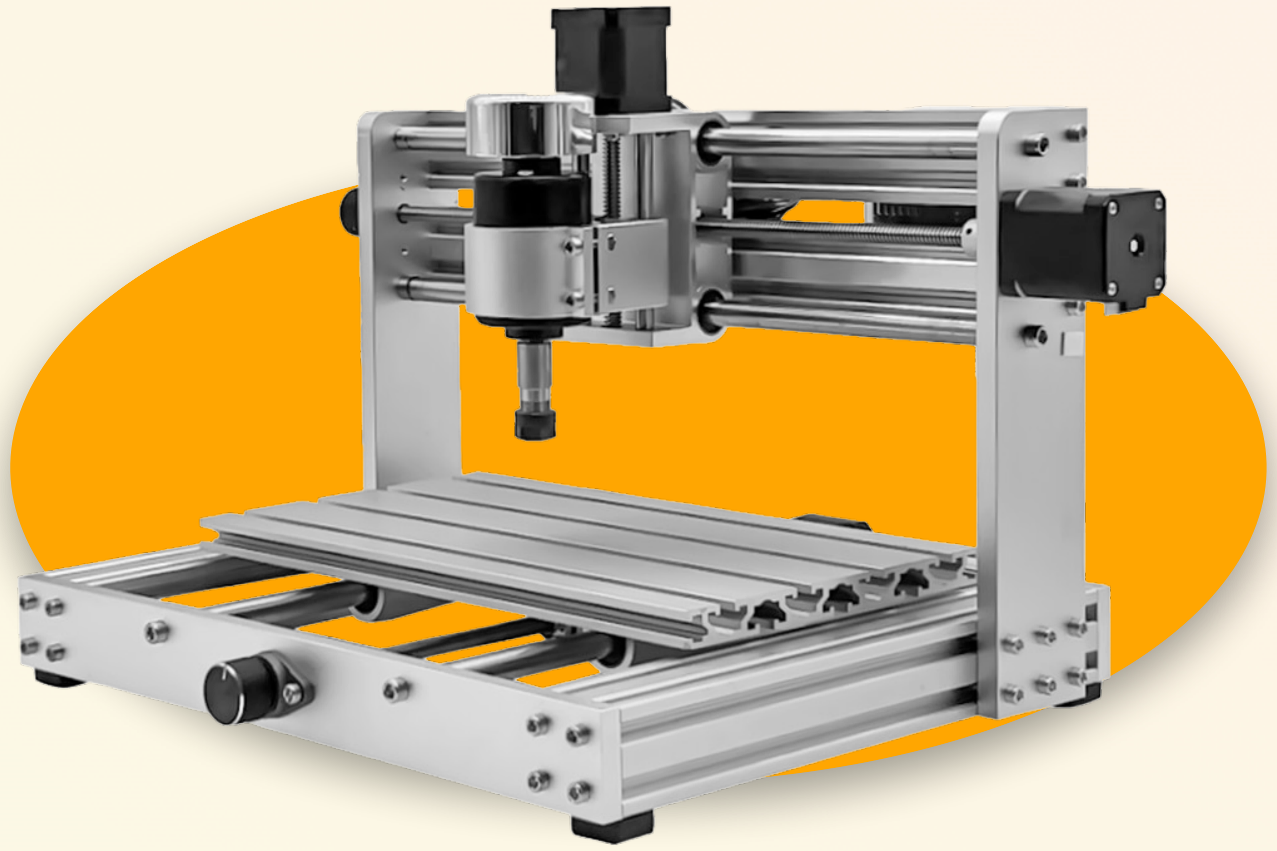
EMFI SETUP 🎰

POSITIONING PLATFORM

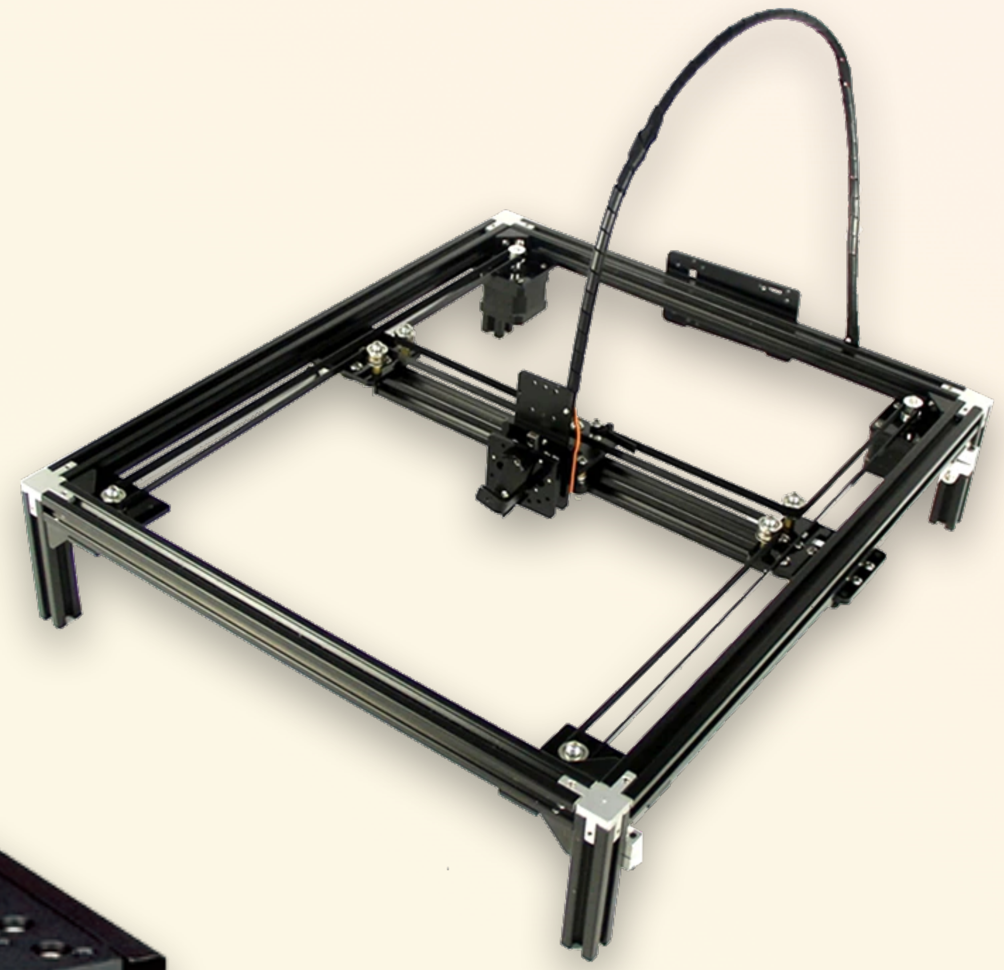
- CNCs or 3D printers can be used interchangeably due to GCODE
- Both are available for very low cost, come with motor controllers and everything needed
- Lead screws have approx. 10x more backlash ➡️ if budget allows, use belts
- Motorized XY stages offer small benefit for the price and IoT target



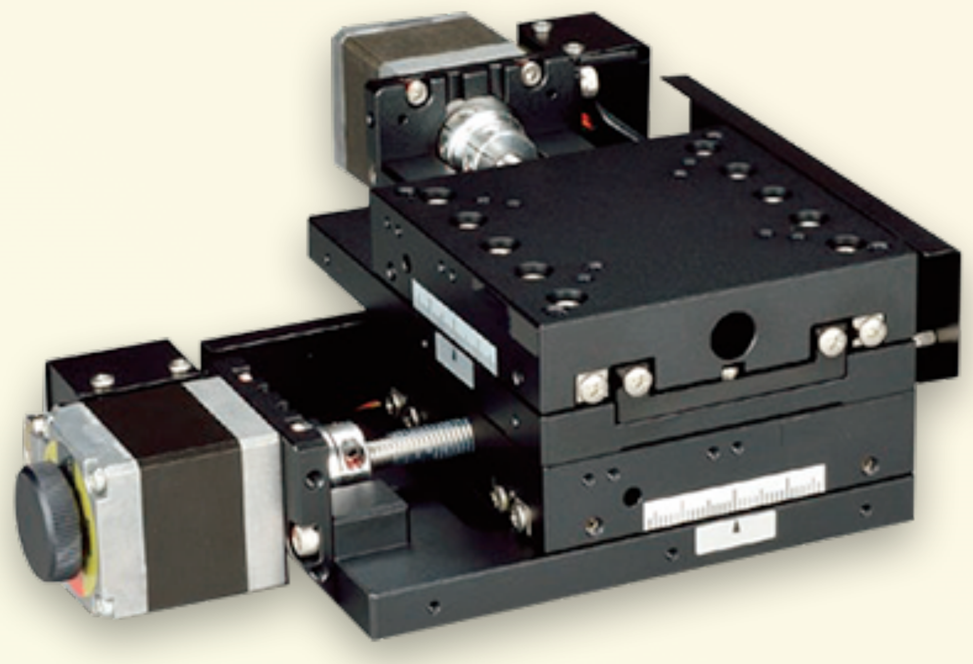
https://aliexpress.com



https://aliexpress.com



https://aliexpress.com

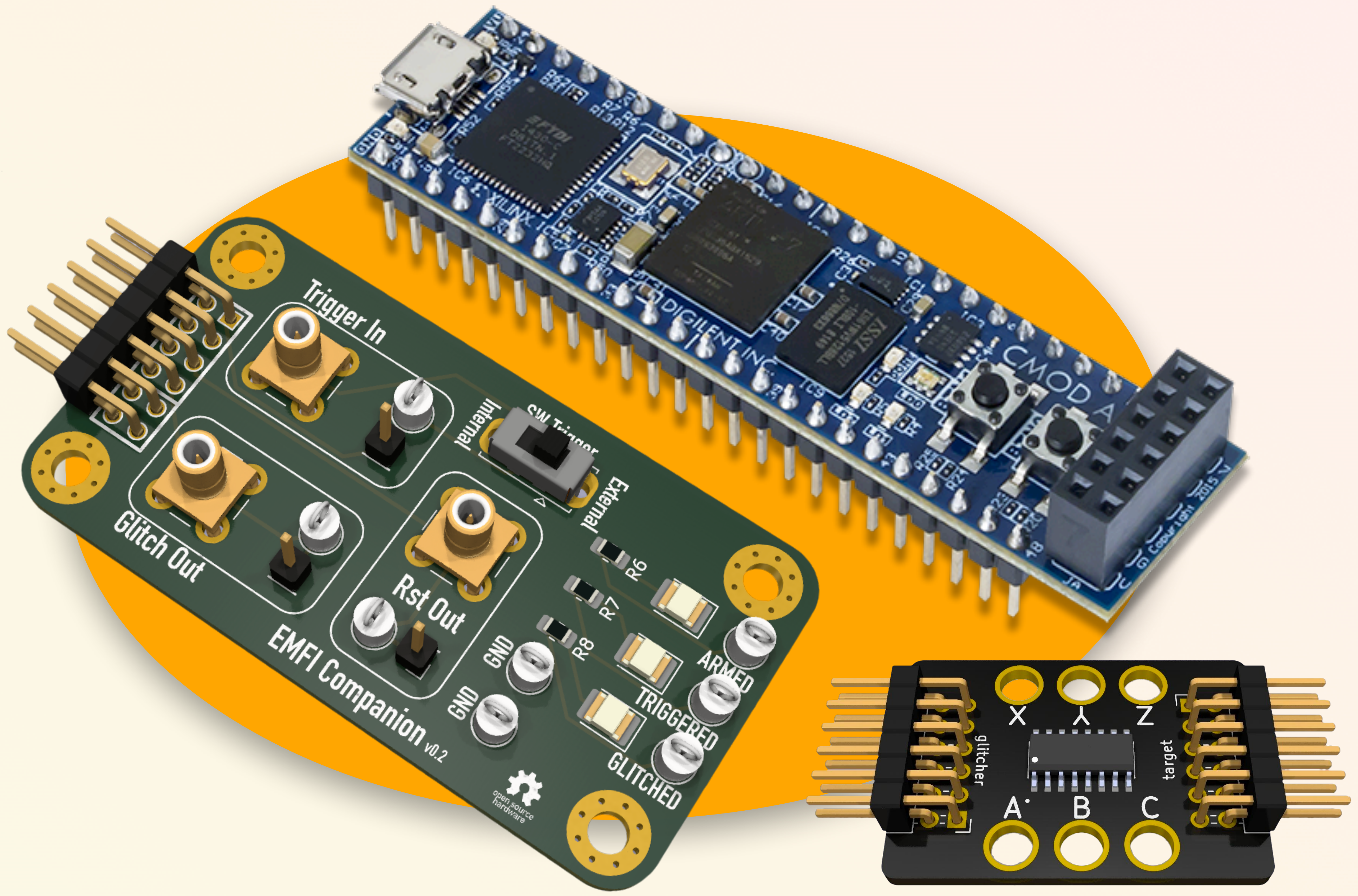


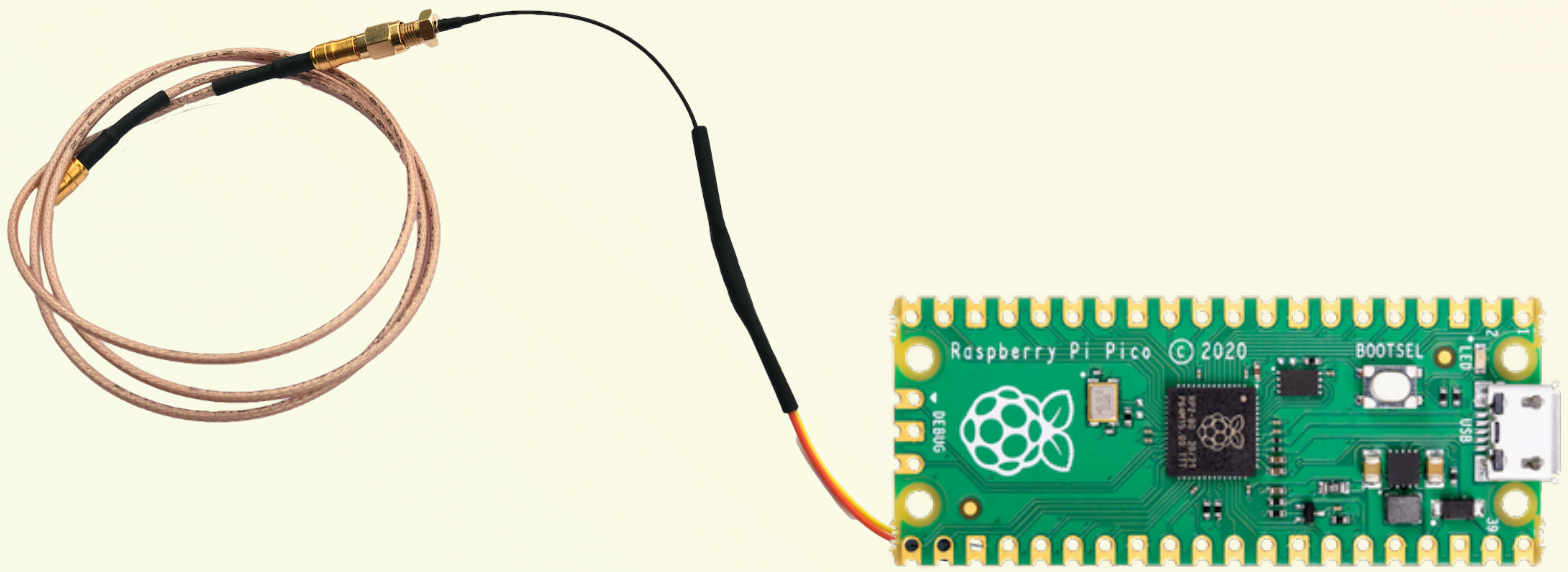
https://www.thk.com

EMFI SETUP 🎰

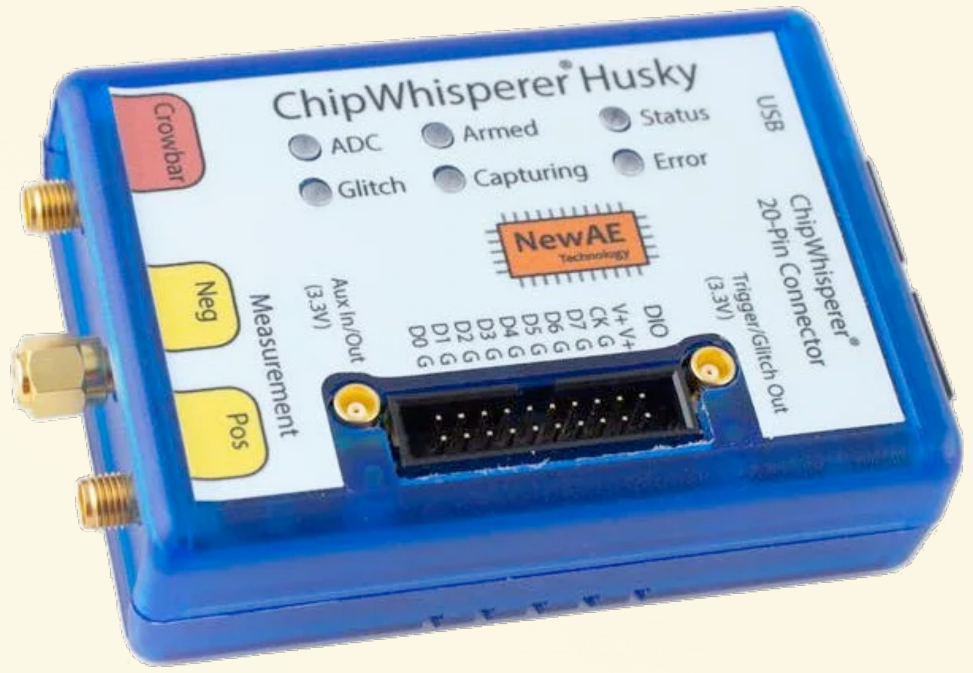
DELAY GENERATOR

- FPGA: chip.fail FOSS bitstream @stacksmashing
- Raspberry Pi Pico: custom firmware (WiP)
- ChipWhisperer @colinoflynn





https://www.raspberrypi.com/products/raspberry-pi-pico/

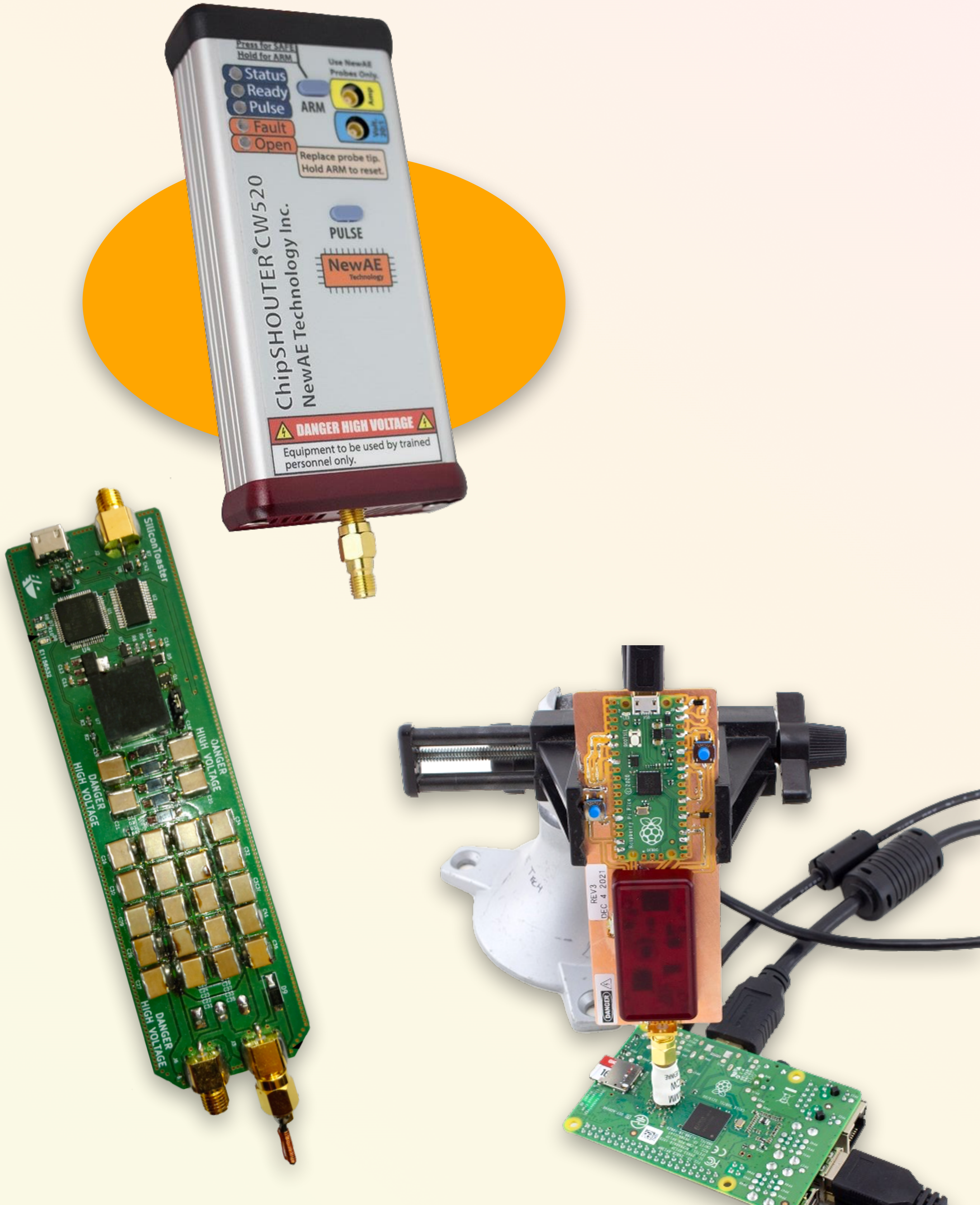


https://www.newae.com/products/NAE-CWHUSKY

EMFI SETUP 🎰

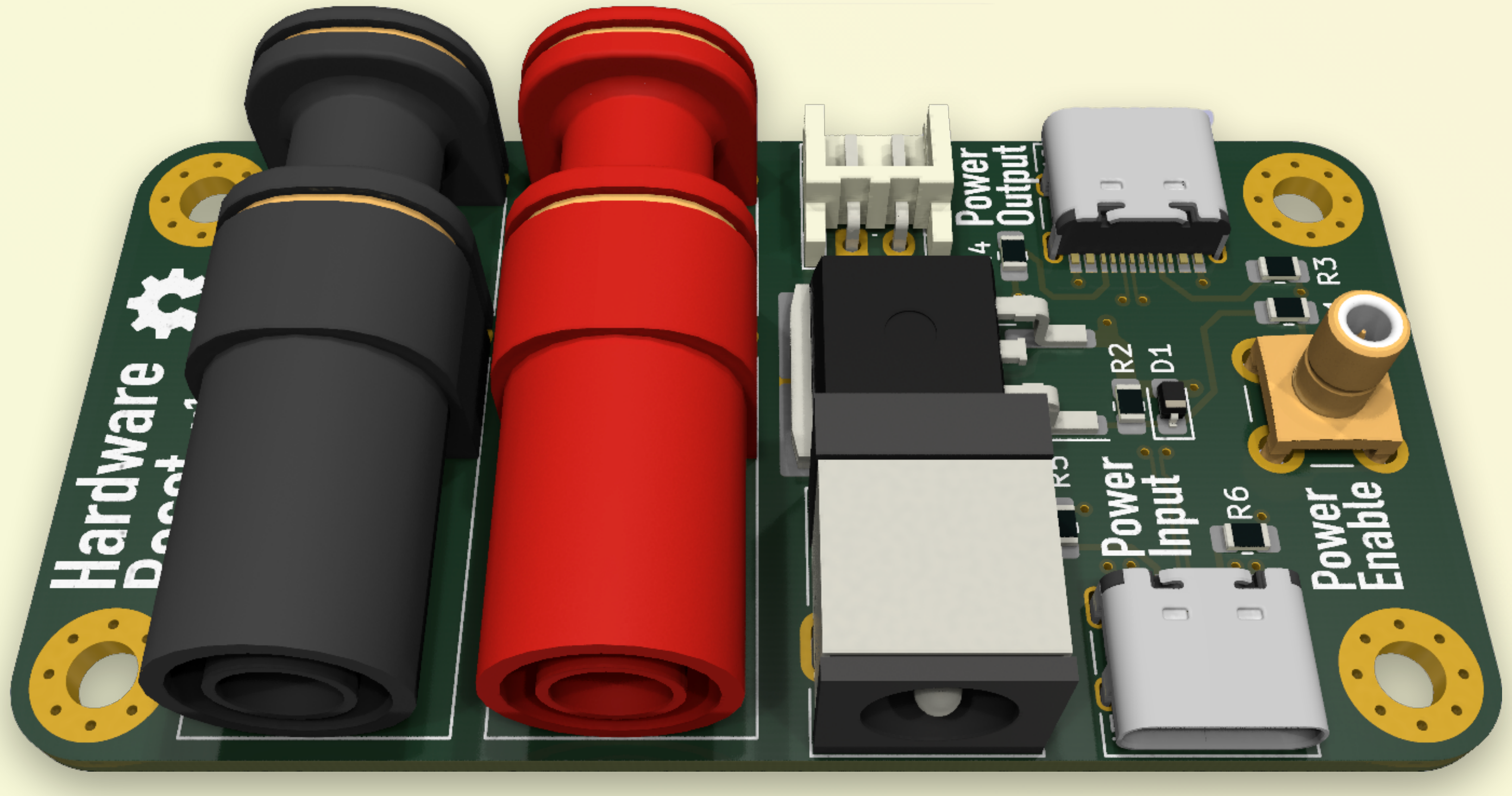
EMFI PULSER

- ChipSHOUTER by @colinoflynn
- PicoEMP by @colinoflynn, @stacksmashing et al.
- SiliconToaster by Ledger

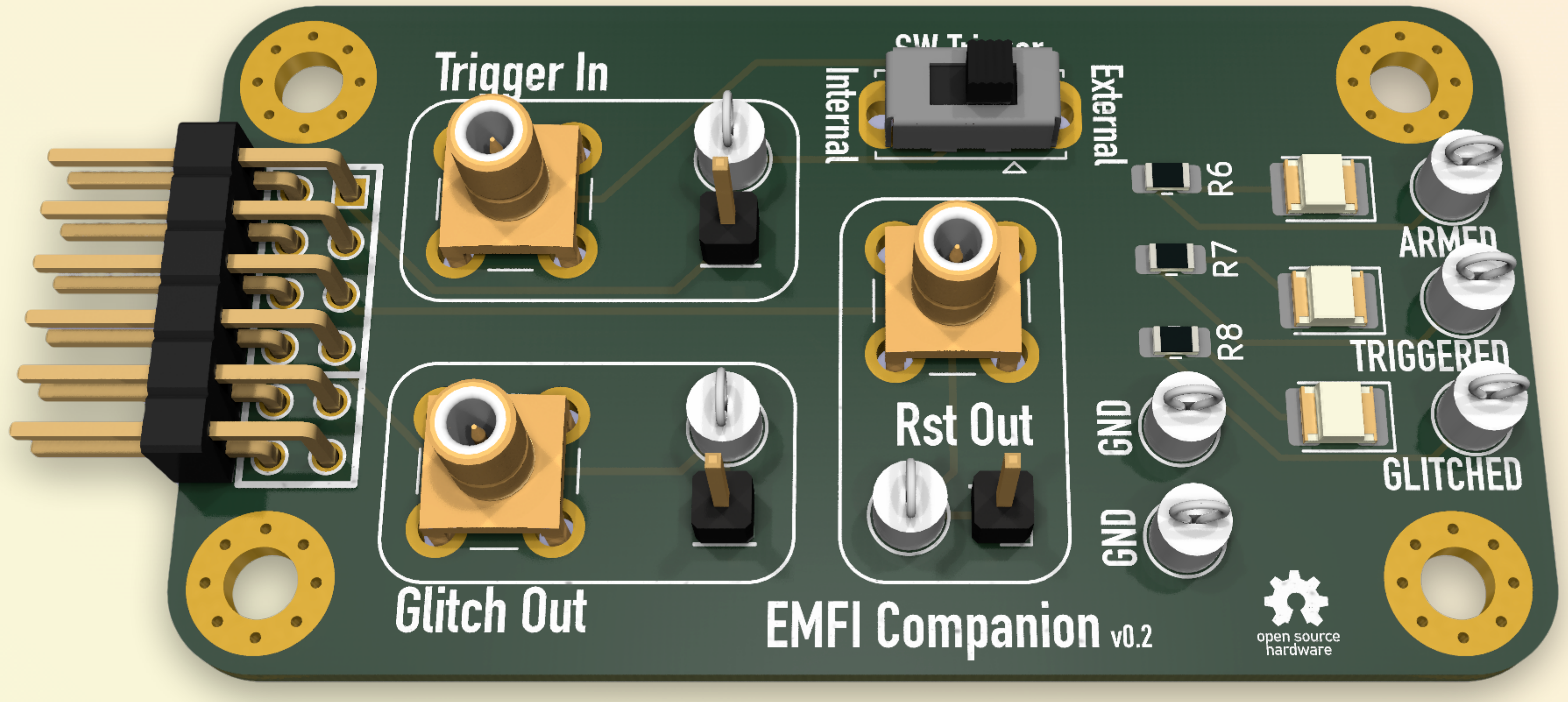


EMFI SETUP 🎰

CUSTOM HARDWARE & SOFTWARE



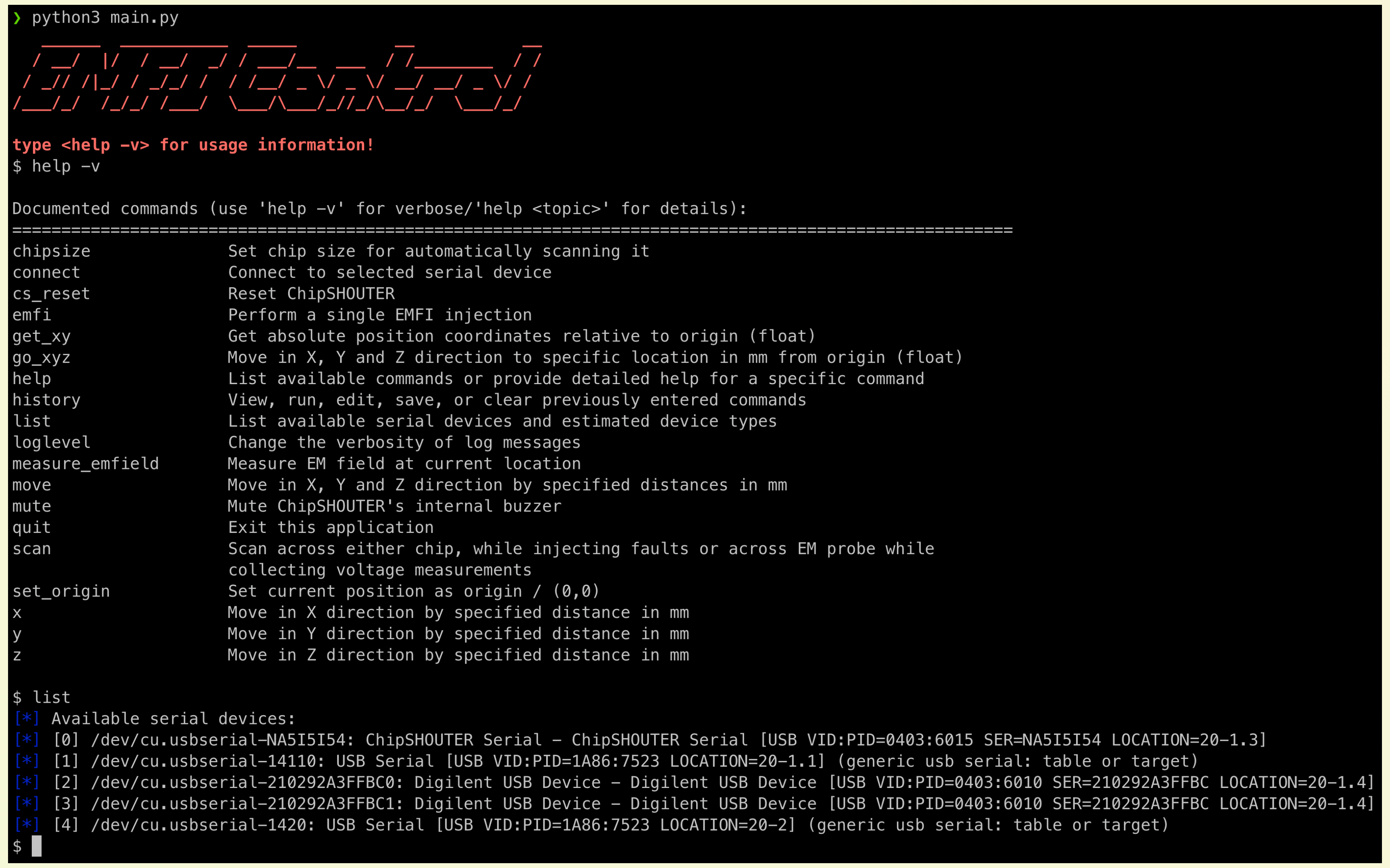
HARDWARE RESET



FPGA BREAKOUT

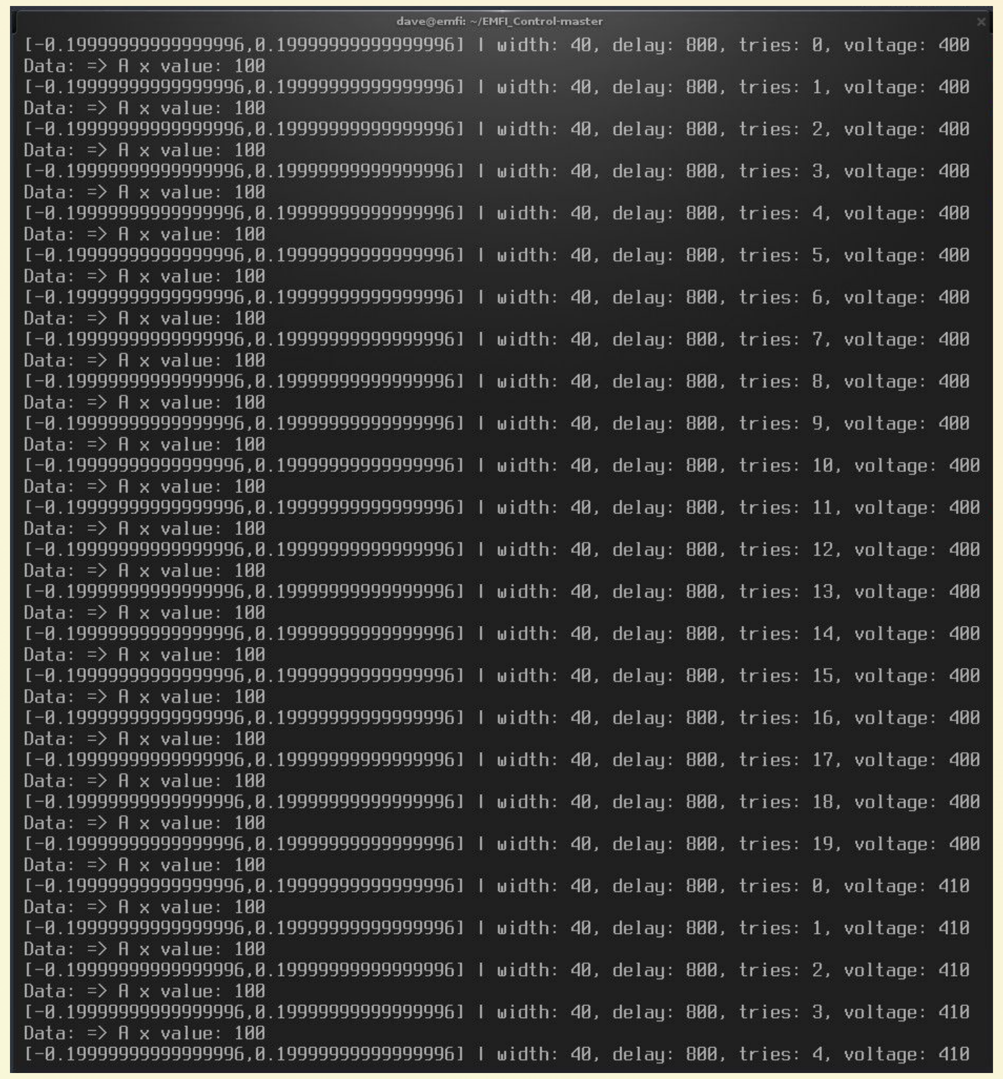
EMFI SETUP

CUSTOM HARDWARE & SOFTWARE



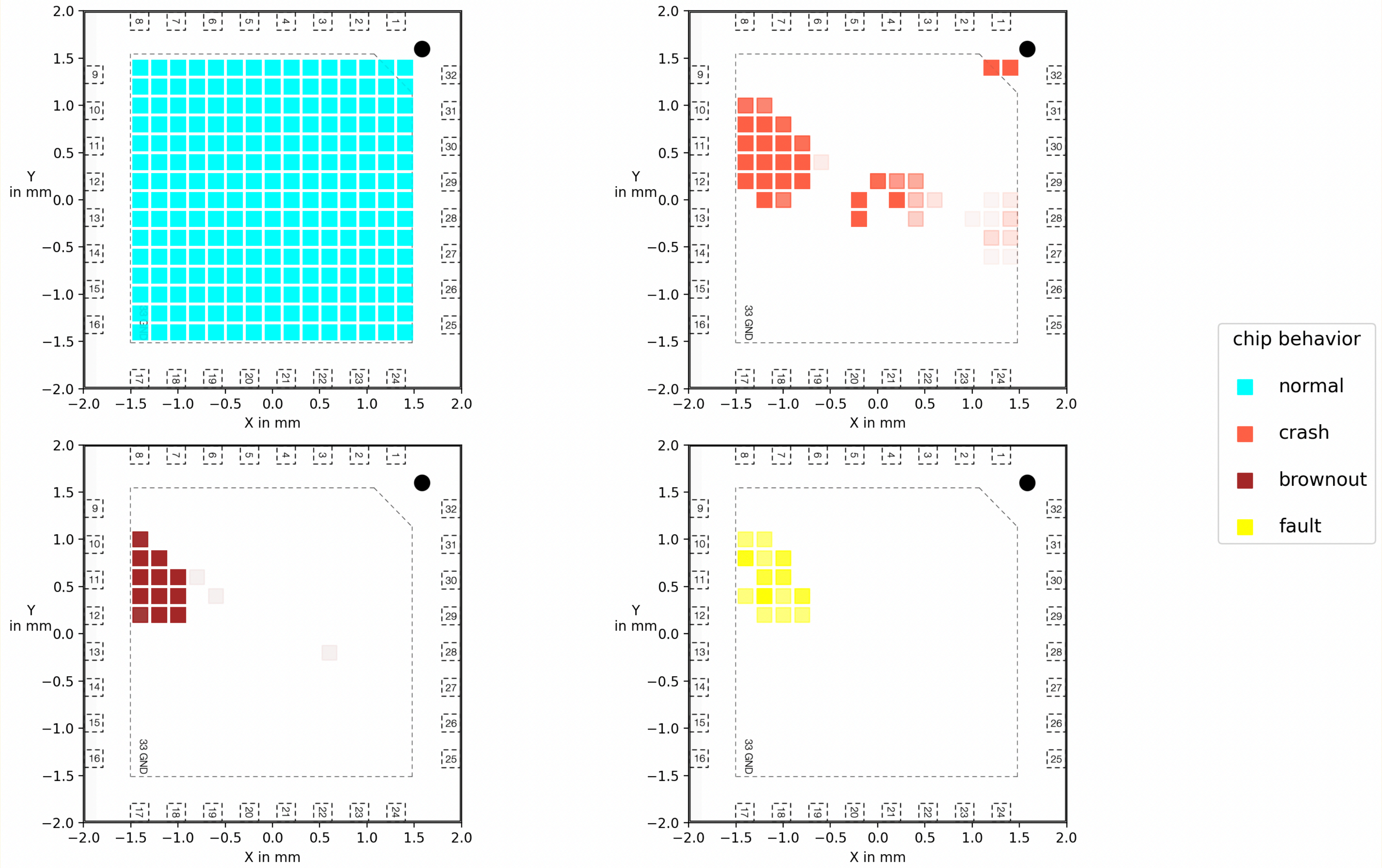
```
❯ python3 main.py
   ______  __________  _____           __         __
  / __/  |/  / __/  _/ / ___/__  ___  / /_______  / /
 / _// /|_/ / _/_/ /  / /__/ _ \/ _ \/ __/ __/ _ \/ /
/___/_/  /_/_/ /___/  \___/\___/_//_/\__/_/  \___/_/

type <help -v> for usage information!
$ help -v

Documented commands (use 'help -v' for verbose/'help <topic>' for details):
==========================================================================================================
chipsize              Set chip size for automatically scanning it
connect               Connect to selected serial device
cs_reset              Reset ChipSHOUTER
emfi                  Perform a single EMFI injection
get_xy                Get absolute position coordinates relative to origin (float)
go_xyz                Move in X, Y and Z direction to specific location in mm from origin (float)
help                  List available commands or provide detailed help for a specific command
history               View, run, edit, save, or clear previously entered commands
list                  List available serial devices and estimated device types
loglevel              Change the verbosity of log messages
measure_emfield       Measure EM field at current location
move                  Move in X, Y and Z direction by specified distances in mm
mute                  Mute ChipSHOUTER's internal buzzer
quit                  Exit this application
scan                  Scan across either chip, while injecting faults or across EM probe while
                      collecting voltage measurements
set_origin            Set current position as origin / (0,0)
x                     Move in X direction by specified distance in mm
y                     Move in Y direction by specified distance in mm
z                     Move in Z direction by specified distance in mm

$ list
[*] Available serial devices:
[*] [0] /dev/cu.usbserial-NA5I5I54: ChipSHOUTER Serial - ChipSHOUTER Serial [USB VID:PID=0403:6015 SER=NA5I5I54 LOCATION=20-1.3]
[*] [1] /dev/cu.usbserial-14110: USB Serial [USB VID:PID=1A86:7523 LOCATION=20-1.1] (generic usb serial: table or target)
[*] [2] /dev/cu.usbserial-210292A3FFBC0: Digilent USB Device - Digilent USB Device [USB VID:PID=0403:6010 SER=210292A3FFBC LOCATION=20-1.4]
[*] [3] /dev/cu.usbserial-210292A3FFBC1: Digilent USB Device - Digilent USB Device [USB VID:PID=0403:6010 SER=210292A3FFBC LOCATION=20-1.4]
[*] [4] /dev/cu.usbserial-1420: USB Serial [USB VID:PID=1A86:7523 LOCATION=20-2] (generic usb serial: table or target)
$ 
```

EMFICONTROL



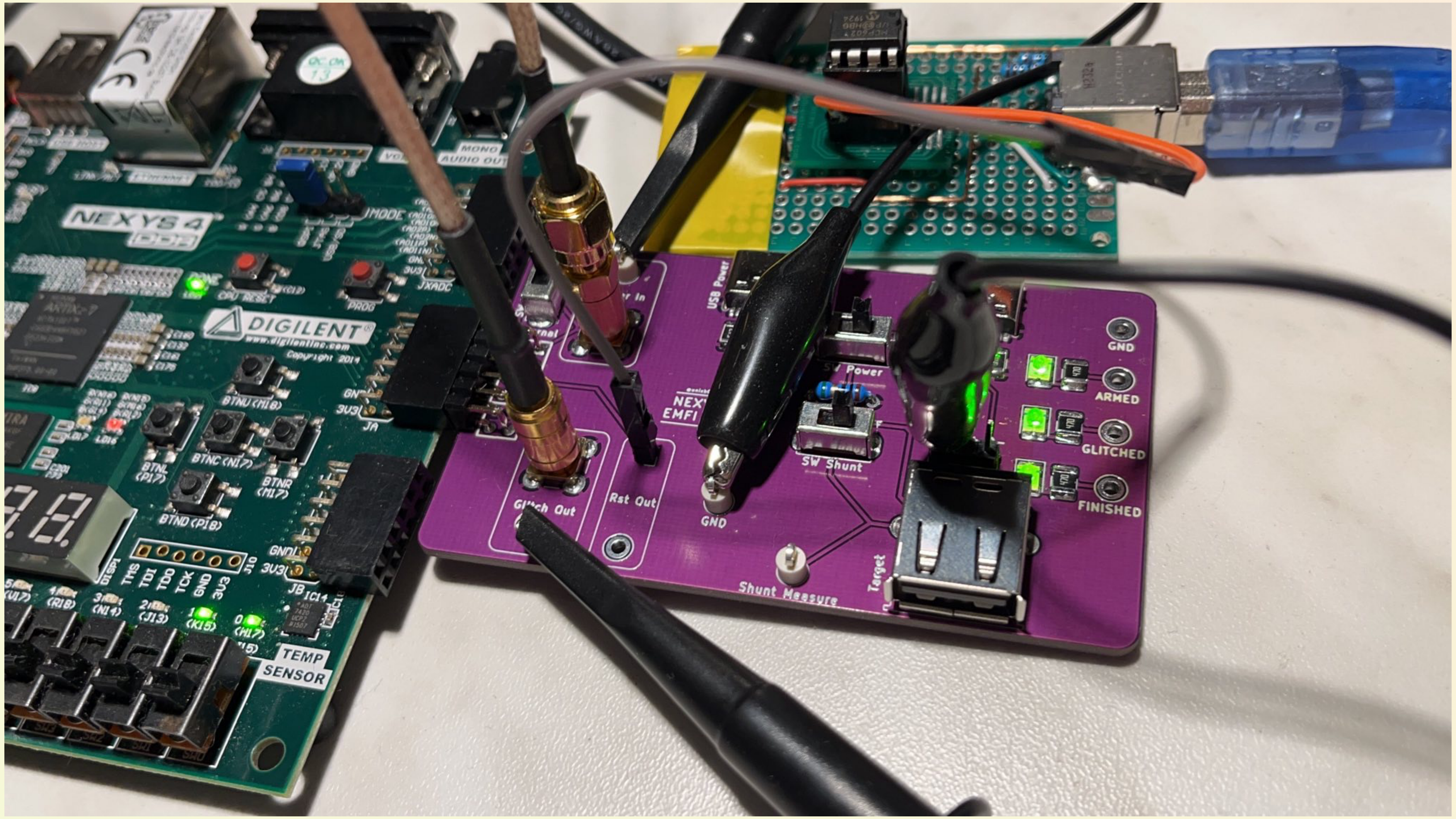
```
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 0, voltage: 400
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 1, voltage: 400
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 2, voltage: 400
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 3, voltage: 400
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 4, voltage: 400
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 5, voltage: 400
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 6, voltage: 400
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 7, voltage: 400
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 8, voltage: 400
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 9, voltage: 400
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 10, voltage: 400
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 11, voltage: 400
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 12, voltage: 400
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 13, voltage: 400
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 14, voltage: 400
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 15, voltage: 400
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 16, voltage: 400
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 17, voltage: 400
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 18, voltage: 400
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 19, voltage: 400
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 0, voltage: 410
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 1, voltage: 410
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 2, voltage: 410
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 3, voltage: 410
Data: => A x value: 100
[-0.19999999999999996,0.19999999999999996] | width: 40, delay: 800, tries: 4, voltage: 410
```

FAULT CAMPAIGN



RESULT VISUALIZATION

EMFI SETUP 🎰

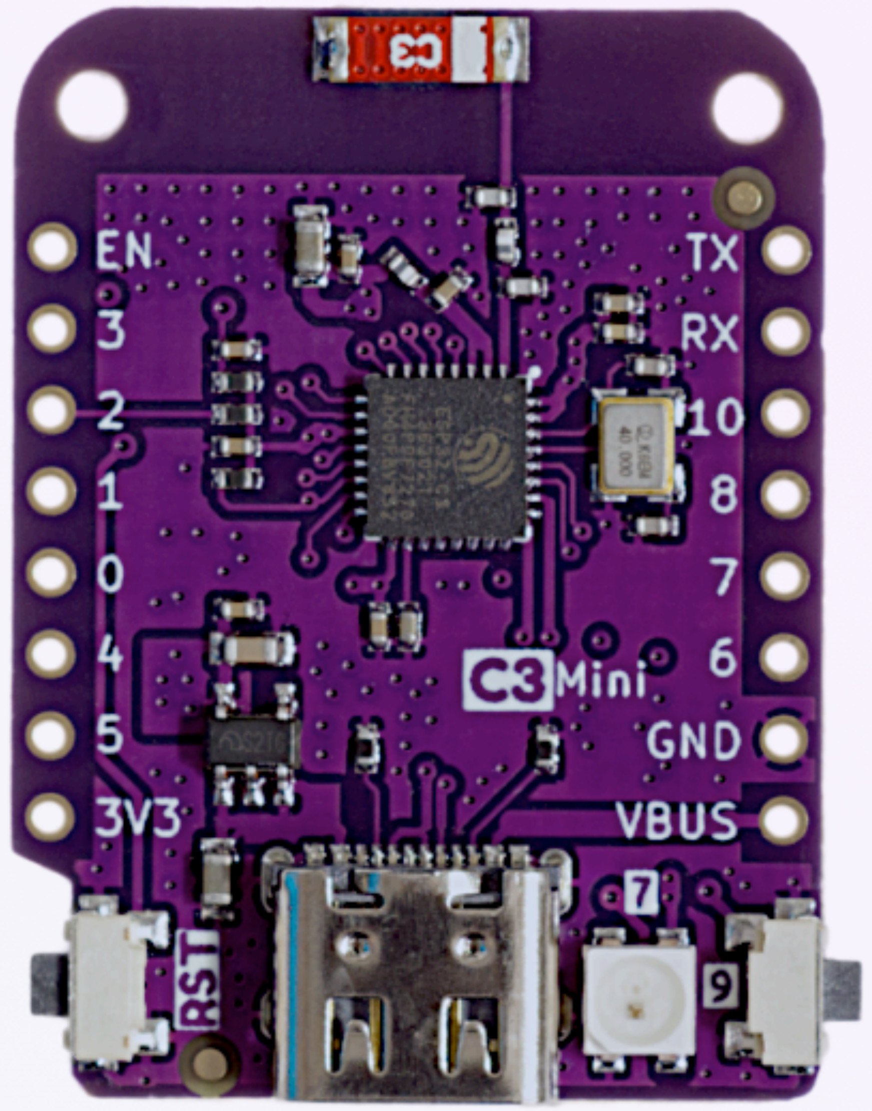

NEXYS 4 DDR
DIGILENT
CPU RESET
PROG
BTNU (M18)
BTNL (P17)
BTNC (N17)
BTNR (M17)
BTND (P18)
TEMP SENSOR
JXADC
JA
JB
MONO AUDIO OUT
USB Power
Power
NEXYS EMFI
Glitch Out
Rst Out
GND
SW Shunt
Shunt Measure
Target
GND
ARMED
GLITCHED
FINISHED

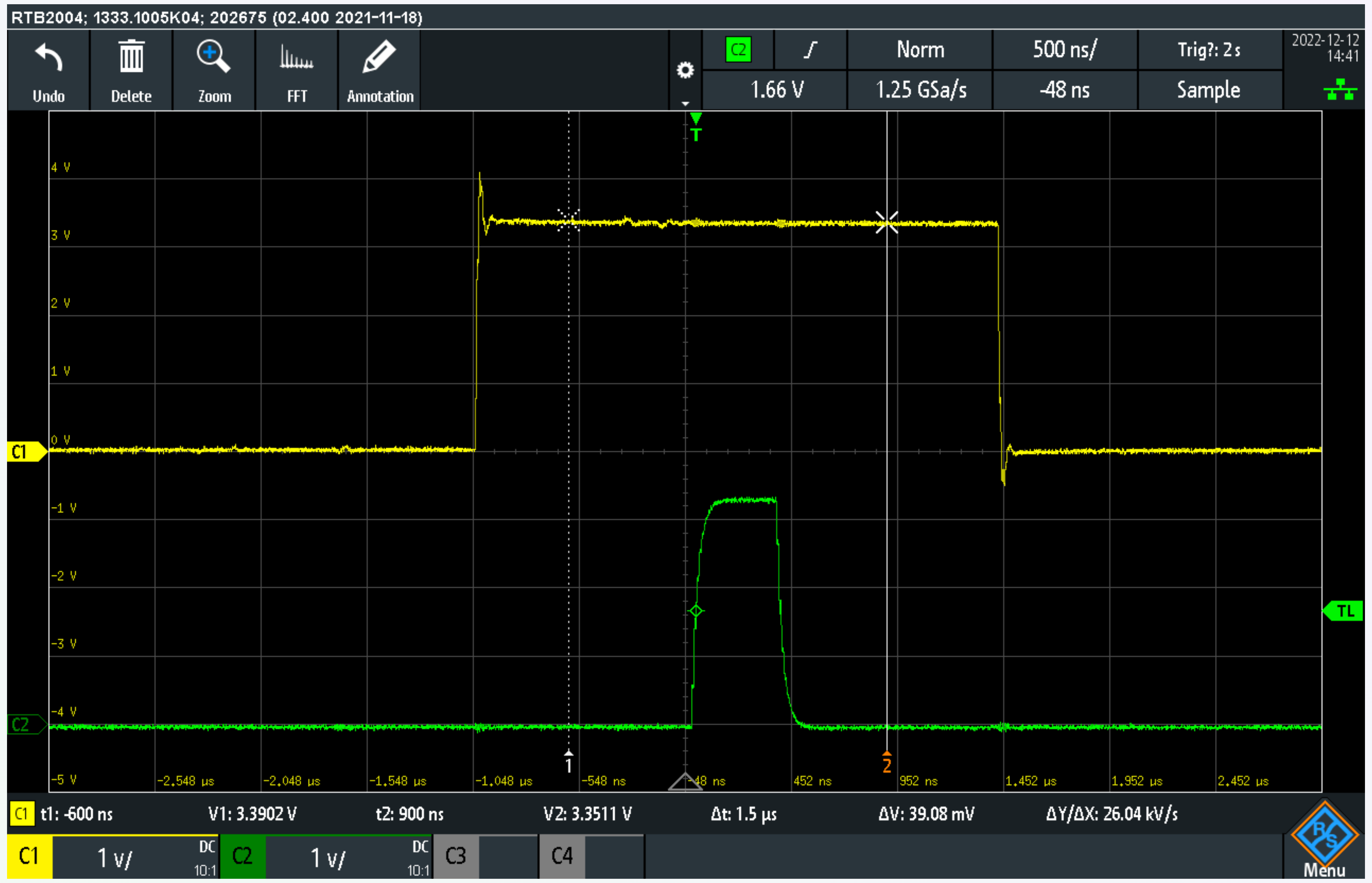
RESULTS ⁉️🤨

VFI PROTECTED IOT CHIP

- Secure boot, RISC-V single core processor, no published glitching research on this version yet.

- “Simple loop” test:
  - GPIO high 👉 100 additions 👉 GPIO low 👉 check result.
  - Glitch in the middle & see if the result changes.





RTB2004; 1333.1005K04; 202675 (02.400 2021-11-18)

C2 Norm 500 ns/ Trig?: 2 s 2022-12-12 14:41
1.66 V 1.25 GSa/s -48 ns Sample

C1 t1: -600 ns V1: 3.3902 V t2: 900 ns V2: 3.3511 V Δt: 1.5 µs ΔV: 39.08 mV ΔY/ΔX: 26.04 kV/s

C1 1 V/ DC 10:1 C2 1 V/ DC 10:1 C3 C4

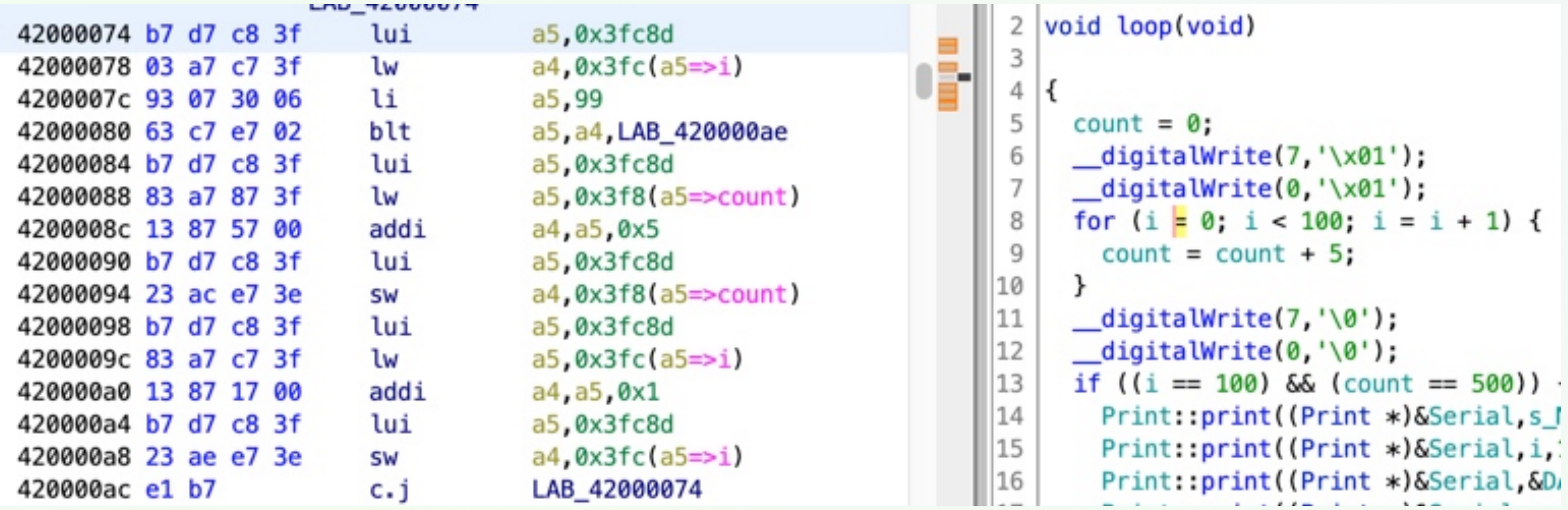
DAVIDE TOLDO | MAY 2023 | HWIO USA

FAULT CHARACTERIZATION

1️⃣ Toolchain compiles to FreeRTOS with brownout detection (although not dedicated for EMFI detection) and other safety checks.

2️⃣ Loop was optimized away; even without optimizations no success; so: inline assembly!

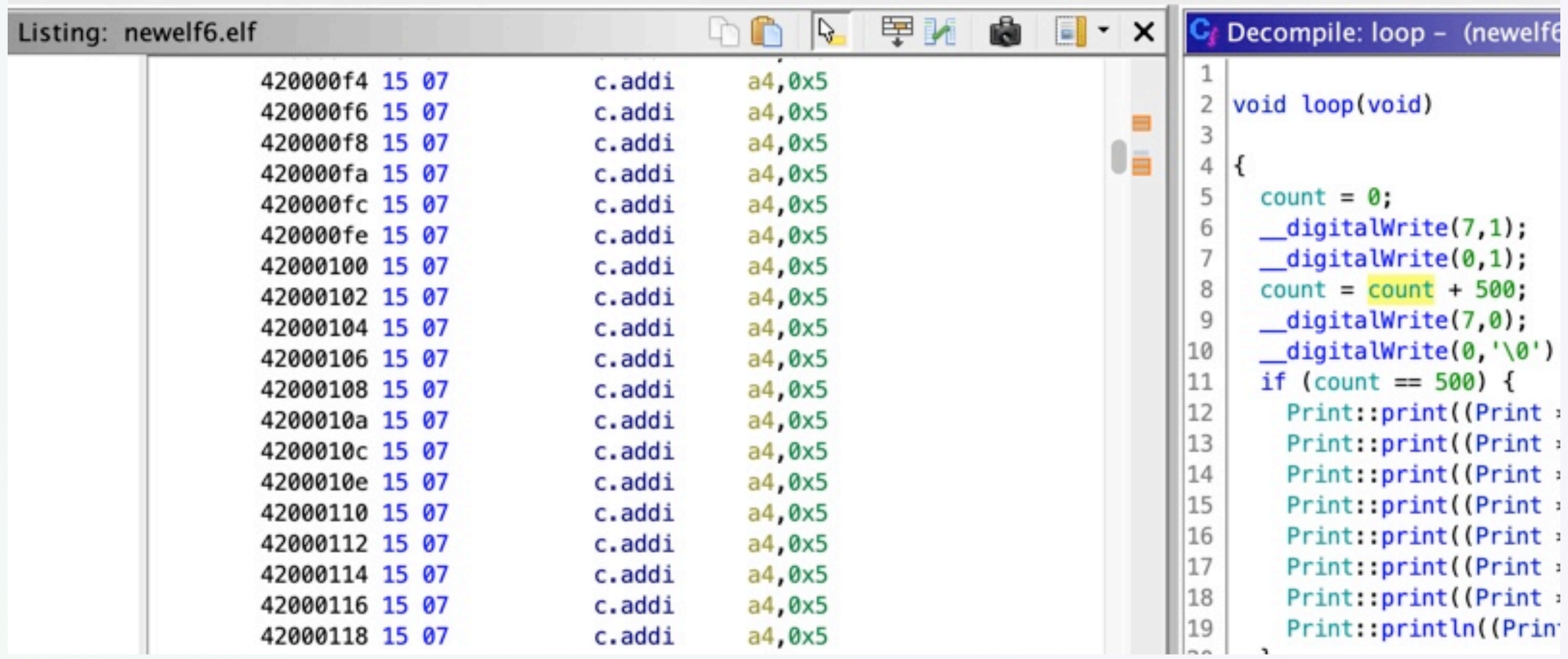
3️⃣ UART transfers corrupted by FI - switch from built-in USB-UART interface to UART via GPIO pins.



```
42000074 b7 d7 c8 3f     lui         a5,0x3fc8d
42000078 03 a7 c7 3f     lw          a4,0x3fc(a5=>i)
4200007c 93 07 30 06     li          a5,99
42000080 63 c7 e7 02     blt         a5,a4,LAB_420000ae
42000084 b7 d7 c8 3f     lui         a5,0x3fc8d
42000088 83 a7 87 3f     lw          a5,0x3f8(a5=>count)
4200008c 13 87 57 00     addi        a4,a5,0x5
42000090 b7 d7 c8 3f     lui         a5,0x3fc8d
42000094 23 ac e7 3e     sw          a4,0x3f8(a5=>count)
42000098 b7 d7 c8 3f     lui         a5,0x3fc8d
4200009c 83 a7 c7 3f     lw          a5,0x3fc(a5=>i)
420000a0 13 87 17 00     addi        a4,a5,0x1
420000a4 b7 d7 c8 3f     lui         a5,0x3fc8d
420000a8 23 ae e7 3e     sw          a4,0x3fc(a5=>i)
420000ac e1 b7           c.j         LAB_42000074
```

```
void loop(void)

{
  count = 0;
  __digitalWrite(7,'\x01');
  __digitalWrite(0,'\x01');
  for (i = 0; i < 100; i = i + 1) {
    count = count + 5;
  }
  __digitalWrite(7,'\0');
  __digitalWrite(0,'\0');
  if ((i == 100) && (count == 500))
    Print::print((Print *)&Serial,s_
    Print::print((Print *)&Serial,i,
    Print::print((Print *)&Serial,&D
```

➡️



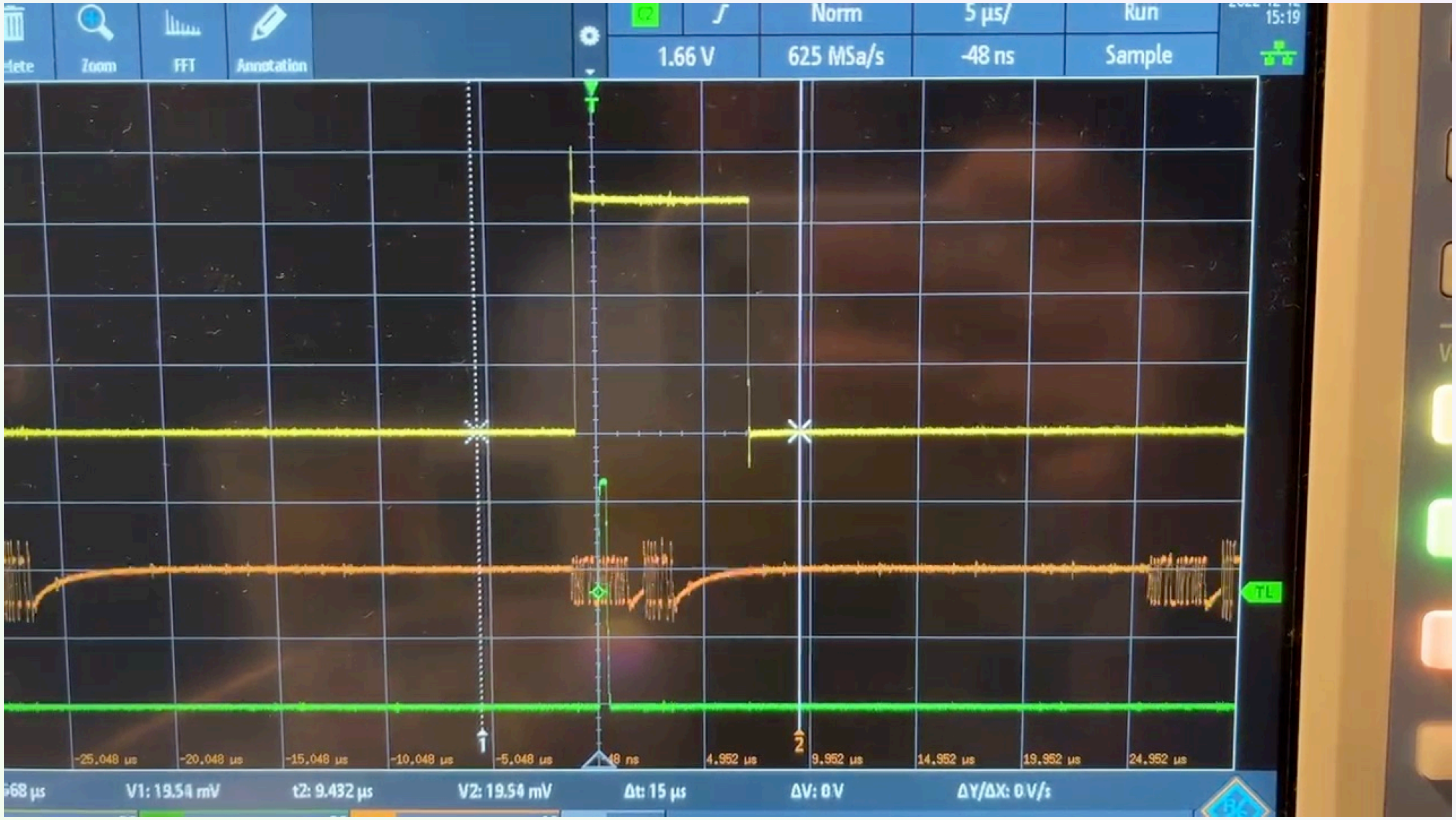
Listing: newelf6.elf

```
420000f4 15 07       c.addi     a4,0x5
420000f6 15 07       c.addi     a4,0x5
420000f8 15 07       c.addi     a4,0x5
420000fa 15 07       c.addi     a4,0x5
420000fc 15 07       c.addi     a4,0x5
420000fe 15 07       c.addi     a4,0x5
42000100 15 07       c.addi     a4,0x5
42000102 15 07       c.addi     a4,0x5
42000104 15 07       c.addi     a4,0x5
42000106 15 07       c.addi     a4,0x5
42000108 15 07       c.addi     a4,0x5
4200010a 15 07       c.addi     a4,0x5
4200010c 15 07       c.addi     a4,0x5
4200010e 15 07       c.addi     a4,0x5
42000110 15 07       c.addi     a4,0x5
42000112 15 07       c.addi     a4,0x5
42000114 15 07       c.addi     a4,0x5
42000116 15 07       c.addi     a4,0x5
42000118 15 07       c.addi     a4,0x5
```

Decompile: loop – (newelf6

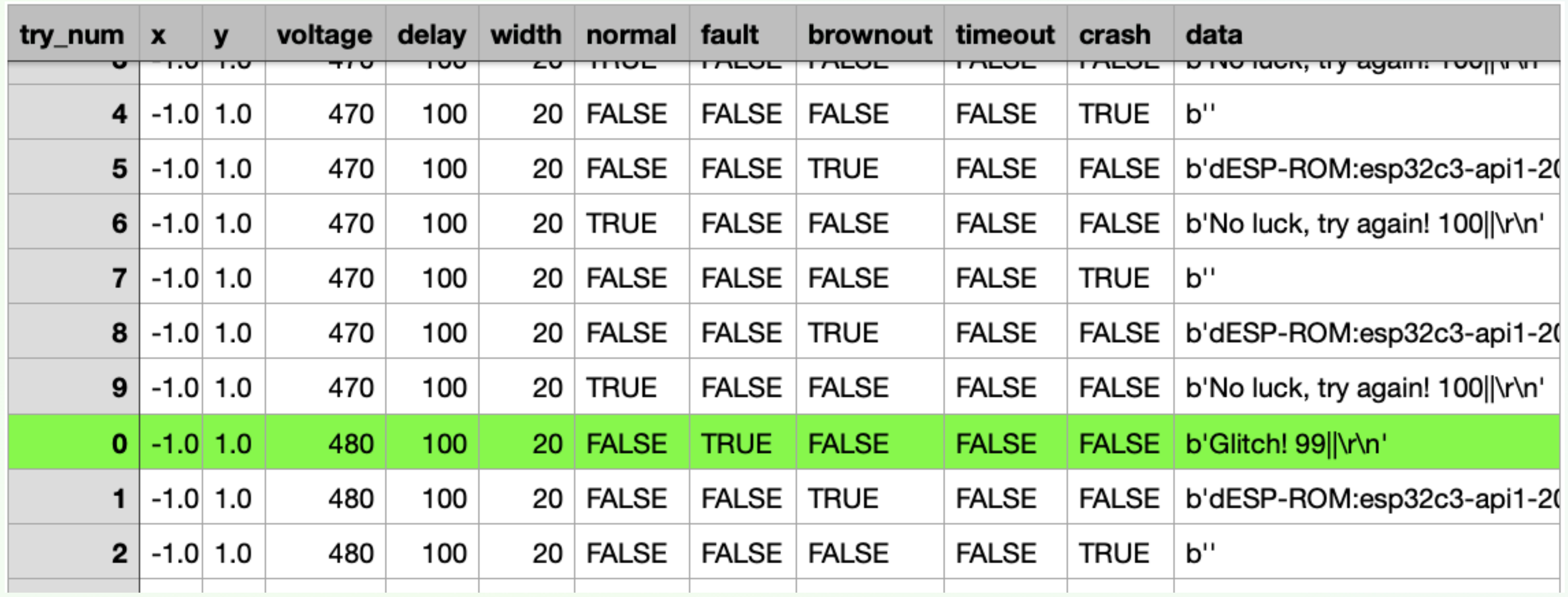
```

void loop(void)

{
  count = 0;
  __digitalWrite(7,1);
  __digitalWrite(0,1);
  count = count + 500;
  __digitalWrite(7,0);
  __digitalWrite(0,'\0')
  if (count == 500) {
    Print::print((Print
    Print::print((Print
    Print::print((Print
    Print::print((Print
    Print::print((Print
    Print::print((Print
    Print::print((Print
    Print::println((Prin
```

FAULT CHARACTERIZATION

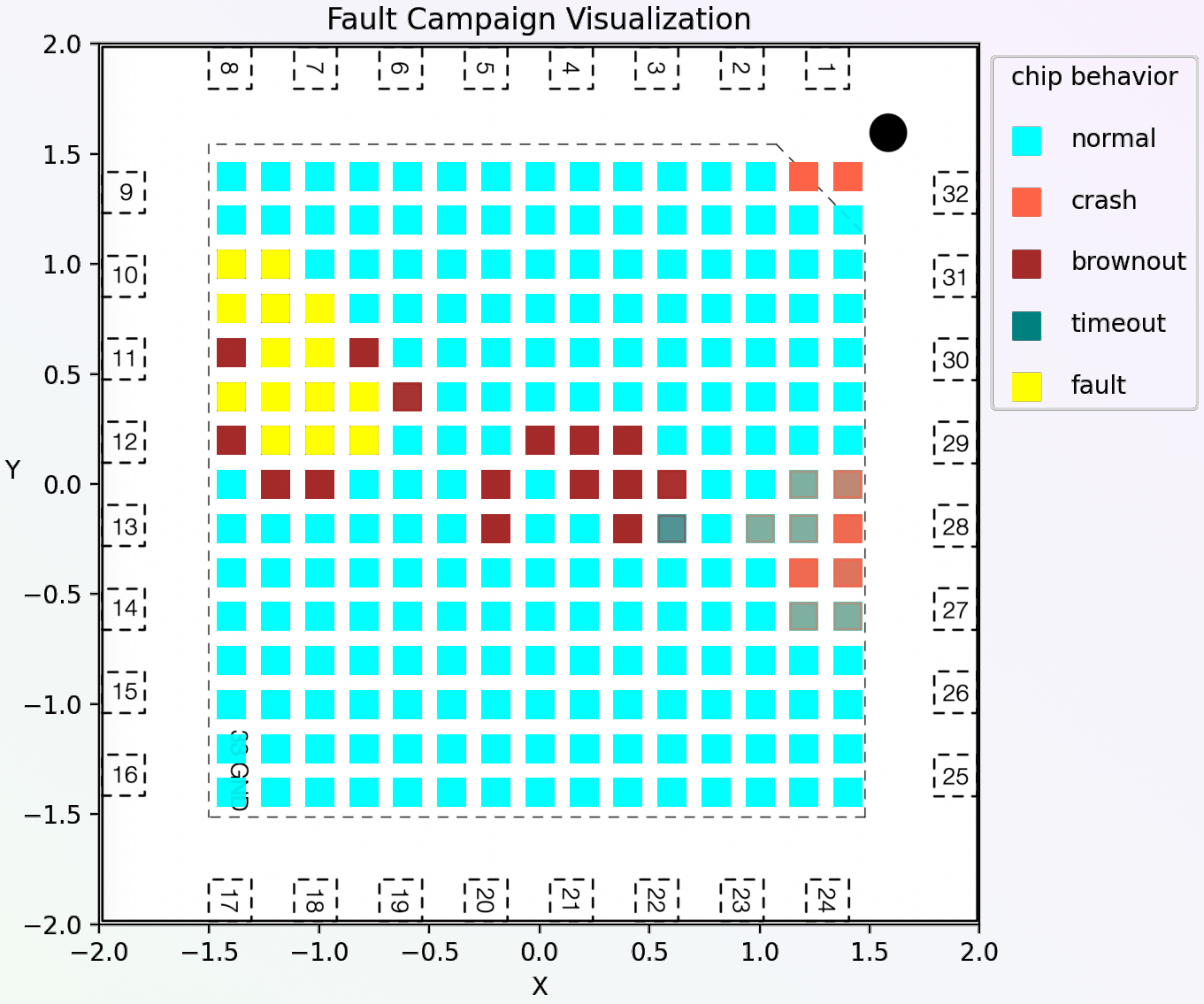


FAULT CAMPAIGN

- **Successful Instruction Skip!! 🎉🥳**
- **No die-shots available, but CPU maybe in the top left?**
- **Complete top left corner is “protected” by brownout detectors.**



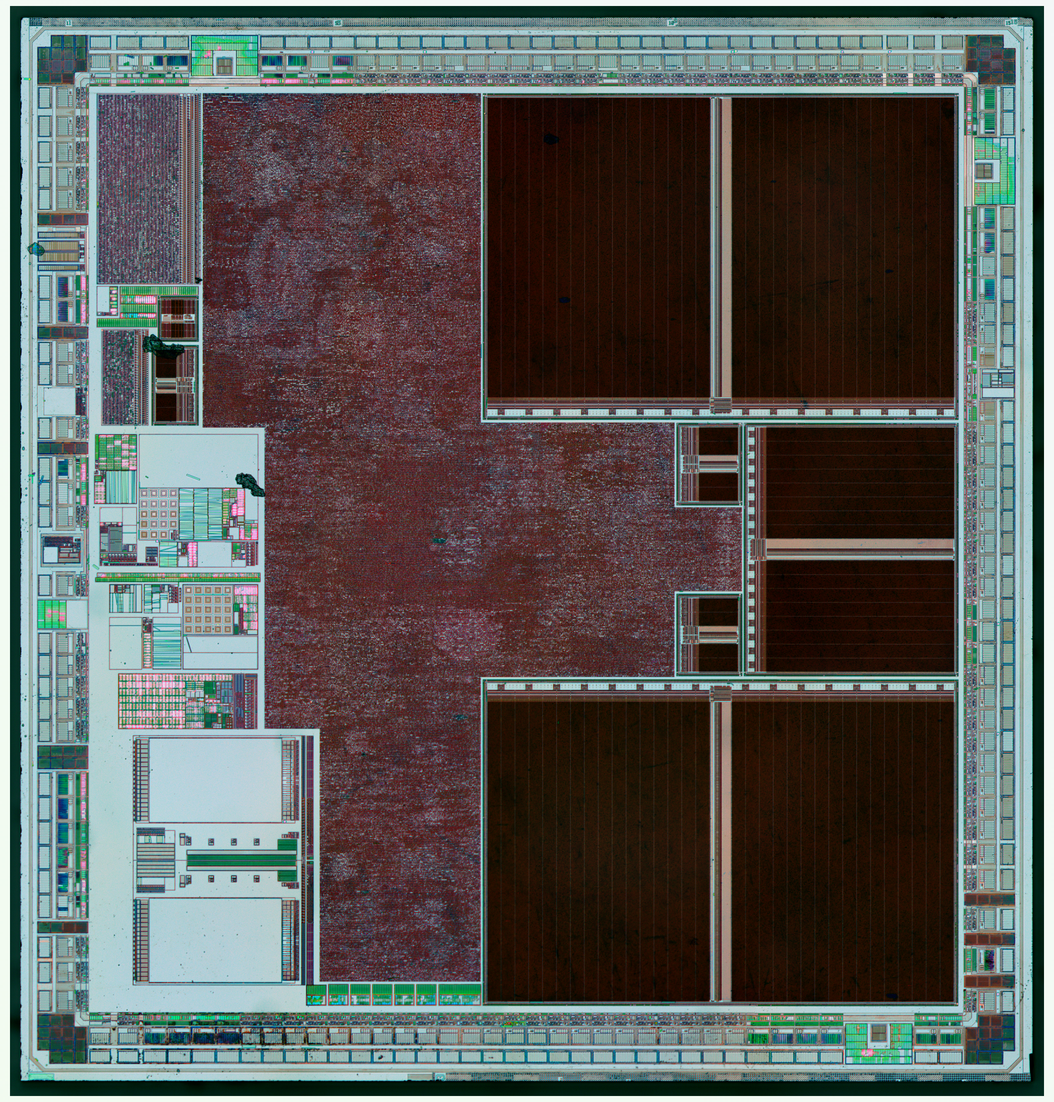
try_num	x	y	voltage	delay	width	normal	fault	brownout	timeout	crash	data
4	-1.0	1.0	470	100	20	FALSE	FALSE	FALSE	FALSE	TRUE	b''
5	-1.0	1.0	470	100	20	FALSE	FALSE	TRUE	FALSE	FALSE	b'dESP-ROM:esp32c3-api1-2(
6	-1.0	1.0	470	100	20	TRUE	FALSE	FALSE	FALSE	FALSE	b'No luck, try again! 100||\r\n'
7	-1.0	1.0	470	100	20	FALSE	FALSE	FALSE	FALSE	TRUE	b''
8	-1.0	1.0	470	100	20	FALSE	FALSE	TRUE	FALSE	FALSE	b'dESP-ROM:esp32c3-api1-2(
9	-1.0	1.0	470	100	20	TRUE	FALSE	FALSE	FALSE	FALSE	b'No luck, try again! 100||\r\n'
0	-1.0	1.0	480	100	20	FALSE	TRUE	FALSE	FALSE	FALSE	b'Glitch! 99||\r\n'
1	-1.0	1.0	480	100	20	FALSE	FALSE	TRUE	FALSE	FALSE	b'dESP-ROM:esp32c3-api1-2(
2	-1.0	1.0	480	100	20	FALSE	FALSE	FALSE	FALSE	TRUE	b''



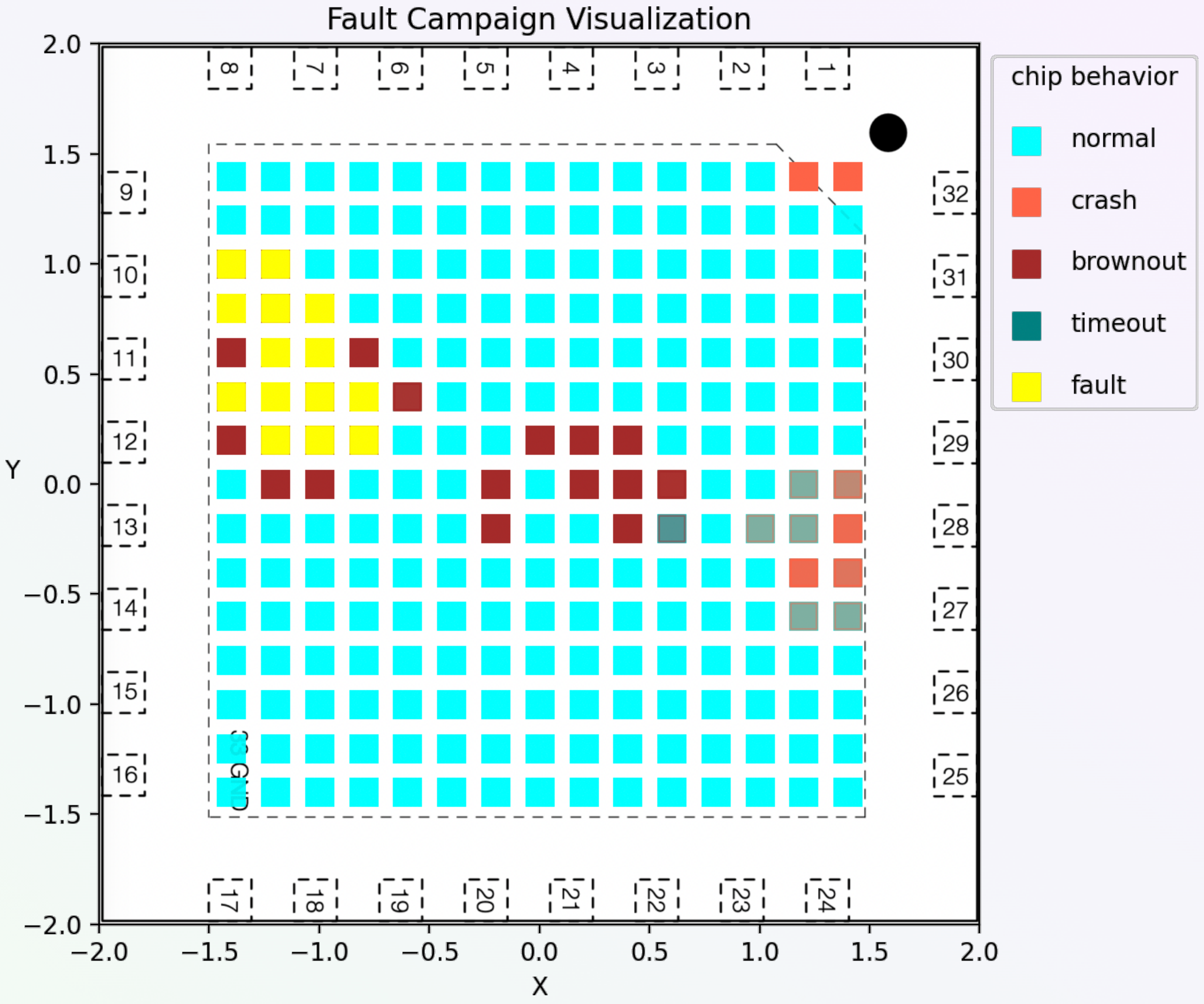
DAVIDE TOLDO | MAY 2023 | HWIO USA

FAULT CAMPAIGN

- 🕰️ Clock - related crashes in the center-right red area; proximity to oscillator.
- Die-shots 📸 could help more accurately map the SoC (requires decapping).

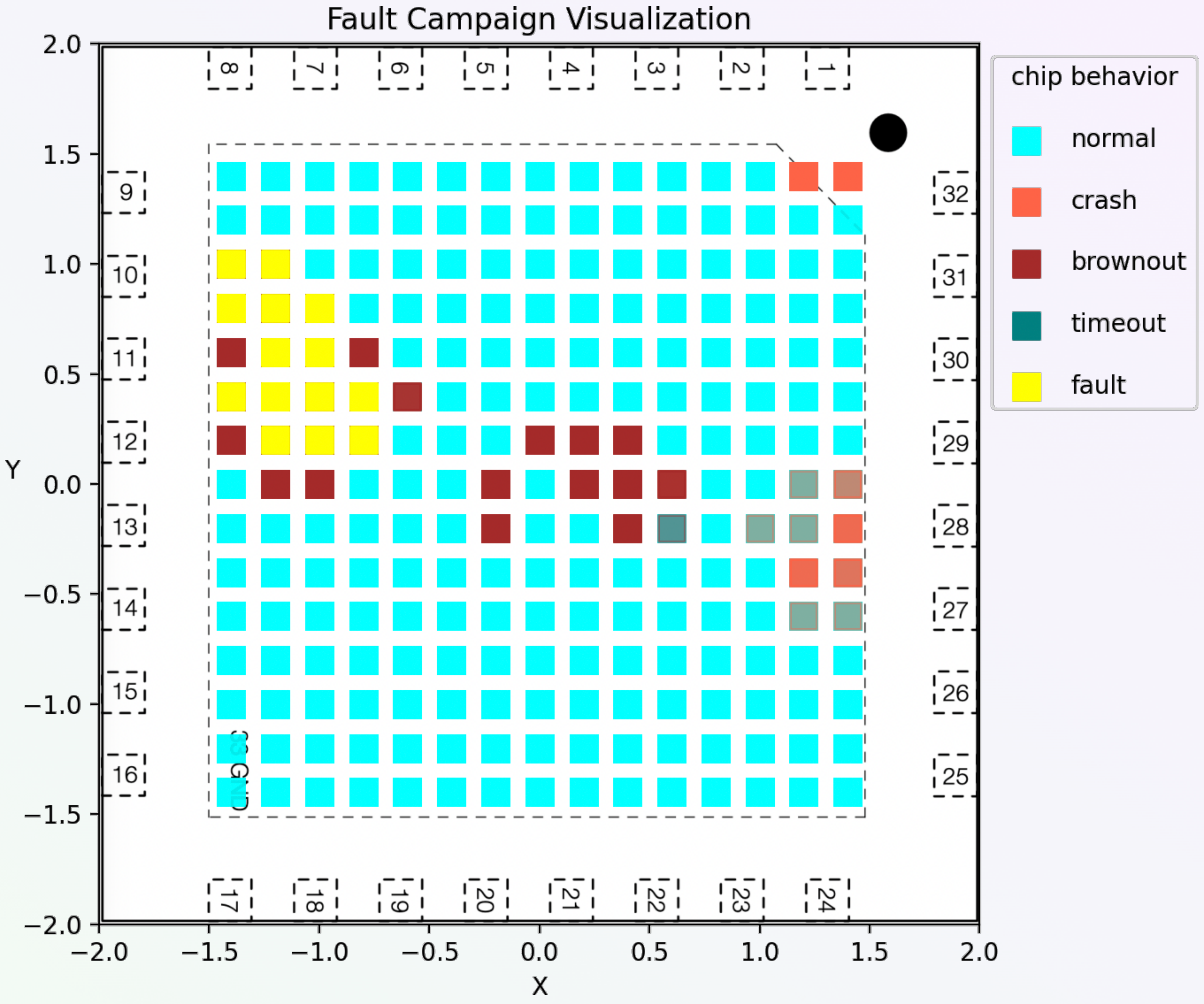


https://commons.wikimedia.org/wiki/File:GD32F103CBT6-Si-HD.jpg



IMPACT

- "No way to fix, but... you can buy the next version 😉" - Limited Results, 2020
- Generally advanced attack, right now not reliable enough to do "in the field".
- Be aware that security features / code paths that check them can be skipped.
- Look into other chips (different architecture, maybe also more secure?)



CONCLUSION

- Low-cost, mostly FOSS / OSHW setup
  - ~$200 X-Y stage
  - ~$2000 EMFI pulser
  - ~$100 delay generator
- Can be improved with little extra cost
  - 3D printer (belts)
  - Hardware reset
  - Higher voltage pulser
- Secure against VFI, side channels, … but not against serious attackers 😈

SPECIAL THANKS

- Quentin Clement 🕵️‍♂️
- Philippe Teuwen 🧙‍♂️

Q&A

github.com/unixb0y/EMFI-Resources

@unixb0y@chaos.social

@unixb0y

dtoldo@seemoo.de