Breaking Bitcoin Hardware Wallets

Glitches cause stitches!

Josh Datko
Chris Quartier
Kirill Belyayev

Updated: 2017/07/07

cryptotronix

Link Drop!

All updated references, notes, links, can be found here:

`https://www.cryptotronix.com/breakingbitcoin`

The bug that started it all

```
1 bool storage_is_pin_correct(const char *pin)
2 {
3     return strcmp(shadow_config.storage.pin,
4                   pin) == 0;
5 }
```

On the STM32F205, when the first pin character is wrong it returns in 100ns. When the fourth was wrong, it returned in about 1100ns.

Broken Window Theory for Bugs

If this was there, what else could we find?

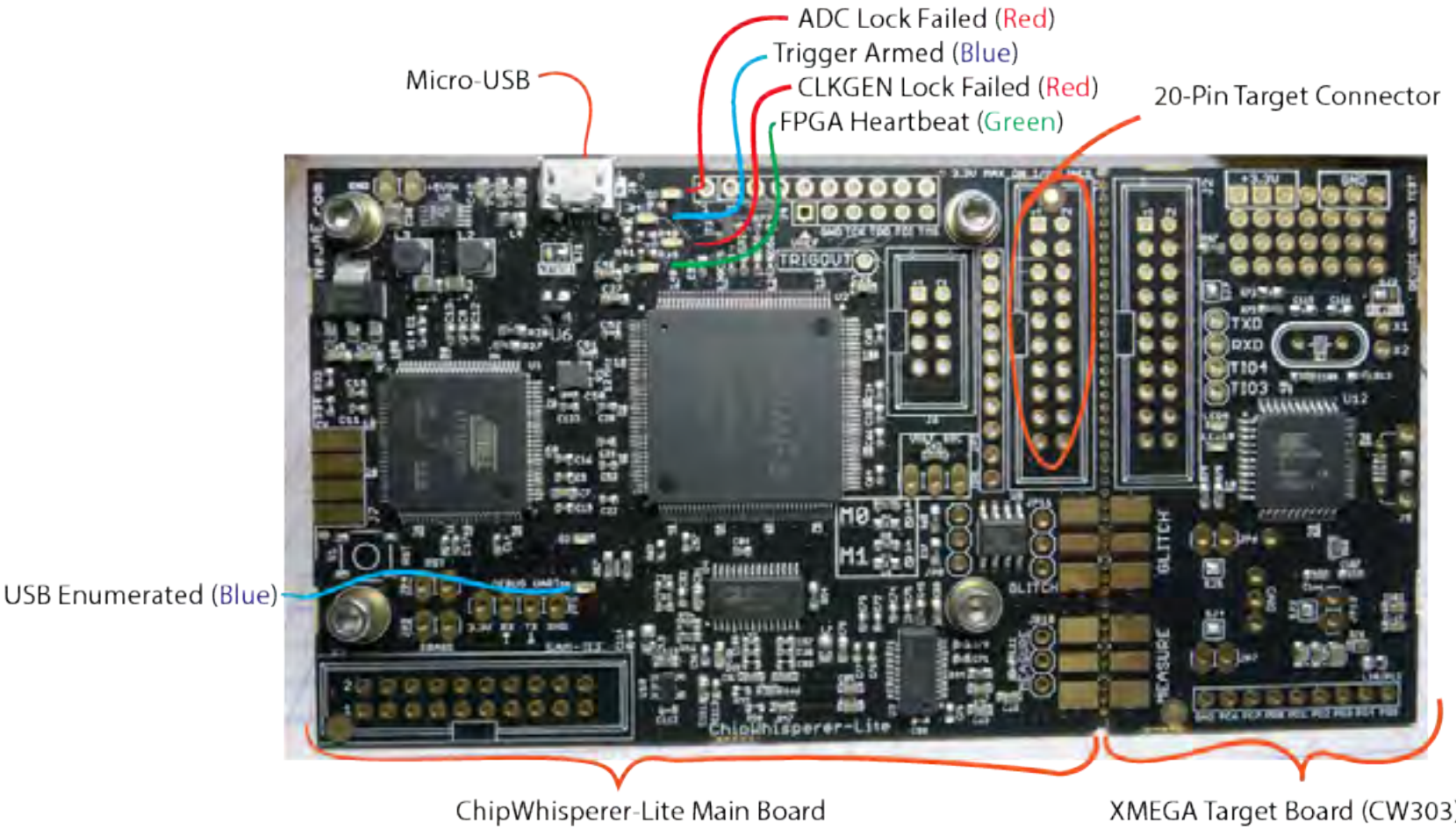
Initial Attack Plan

1. Send `change_pin` via Python.
2. Watch the return over USB–measure when the PIN failed.
3. Profit?!

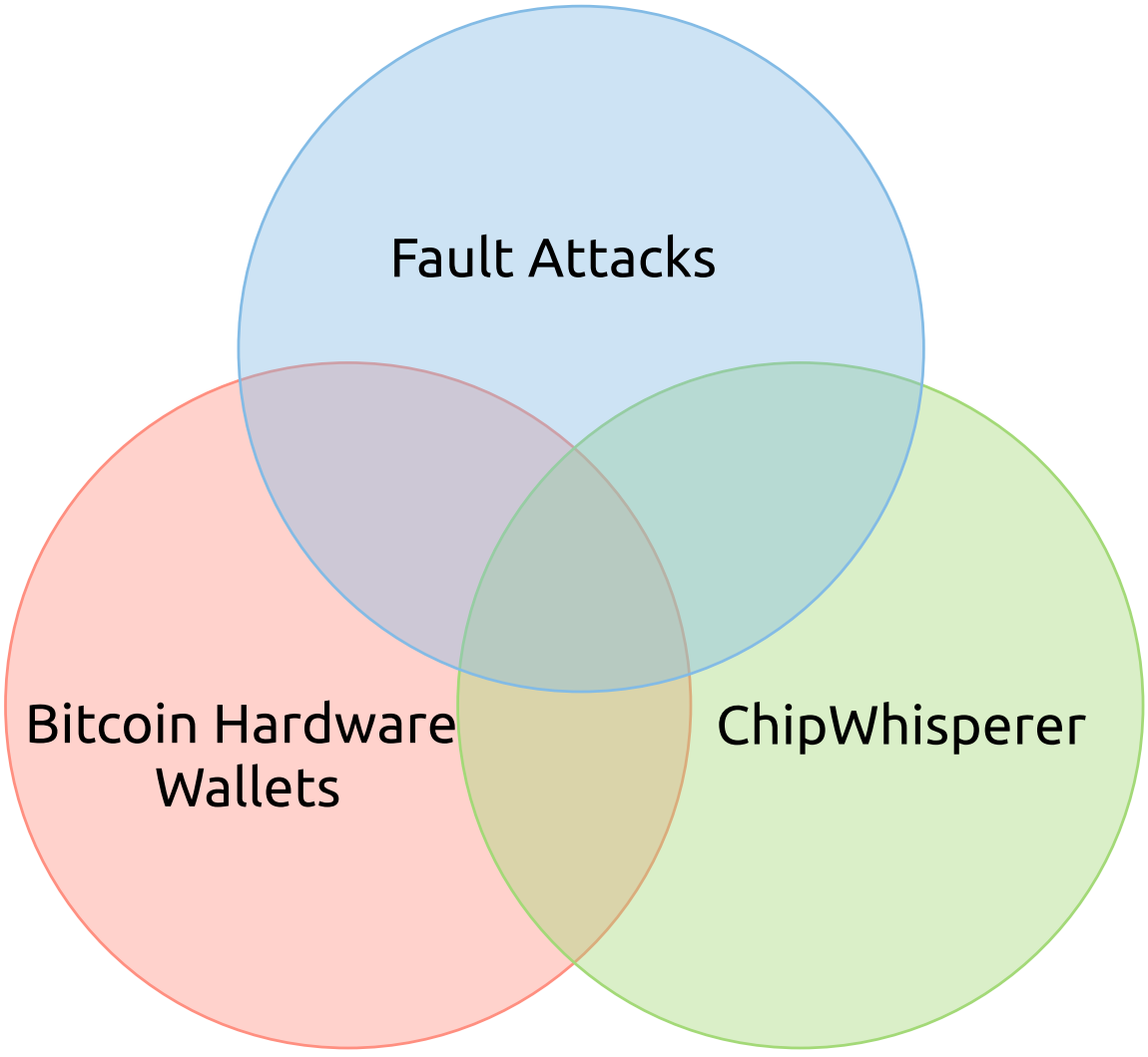
Back off timer

Prevents retries with a busy wait loop.

ChipWhisperer



This talk



One slide intro to Fault Attacks

Definition

An attack that applies an external stress on electronic system, which generates a security failure[2].

Two Parts:

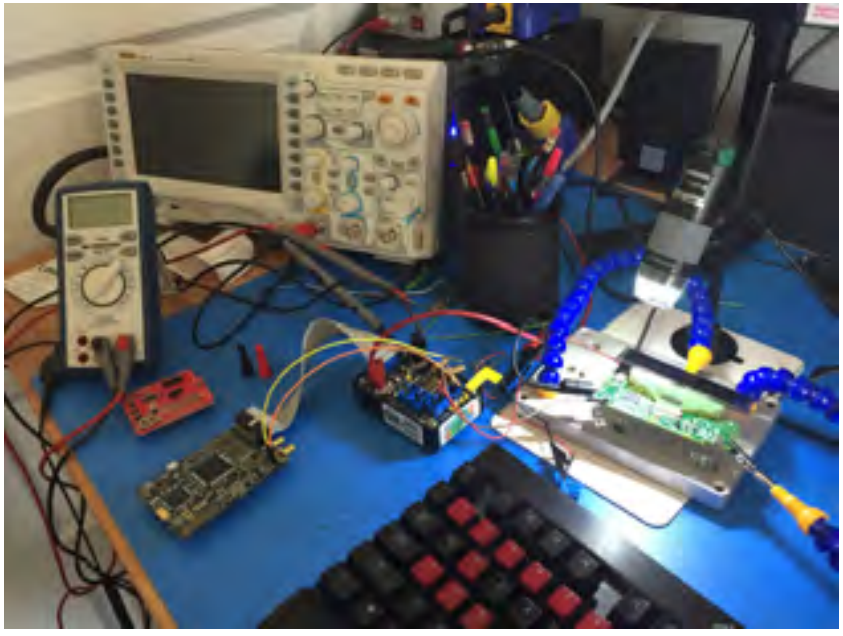
1. Fault Injection
   - Vcc glitching
   - Clock glitching
2. Fault Exploitation
   - Nicolas Bacca suggested glitching flash ops[3], we wanted to bypass the PIN as it was closer to ChipWhisperer examples.

Our Motivation

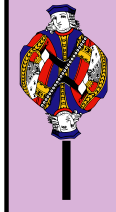
What happens when you apply the ChipWhisperer to the STM32F205 (F205)?

- Is the F205 vulnerable to fault injection?
- Is the TREZOR firmware exploitable via a fault?
- How do we raise awareness for these kinds of attacks?

We just press the glitch button right?



- Turns out, you can't just shake the wallet and have BTC fall out.
- Requires some RE to determine voltages, test points, how to modify the firmware, etc…
- HW Wallets went OOS :(

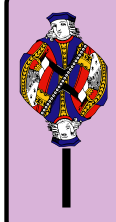


How to slow down attacks

Exhaust the supply chain

The Fail Train Cometh

- Clock glitching kinda worked? It made Windows USB very sad :(
- Rebooting unsigned firmware is teh suck (buttons to press).
- Timing analysis was working, but power analysis with CW was not.
- Logic level conversion is proof that the singularity is far away.
- Lots of scotch.



F-it dude, let's go bowling.

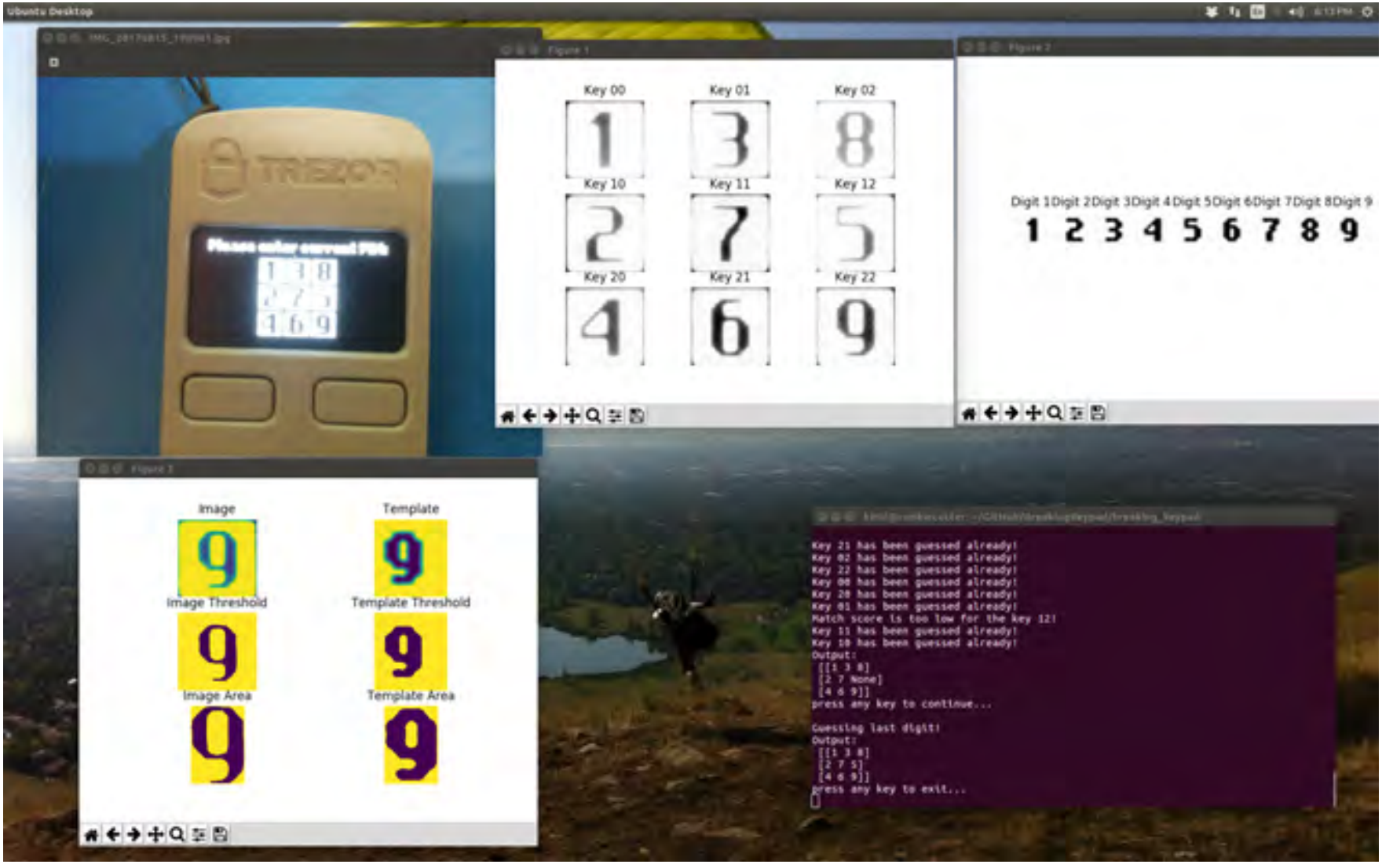
Or why don't we just make our own TREZOR?

And now for something completely different

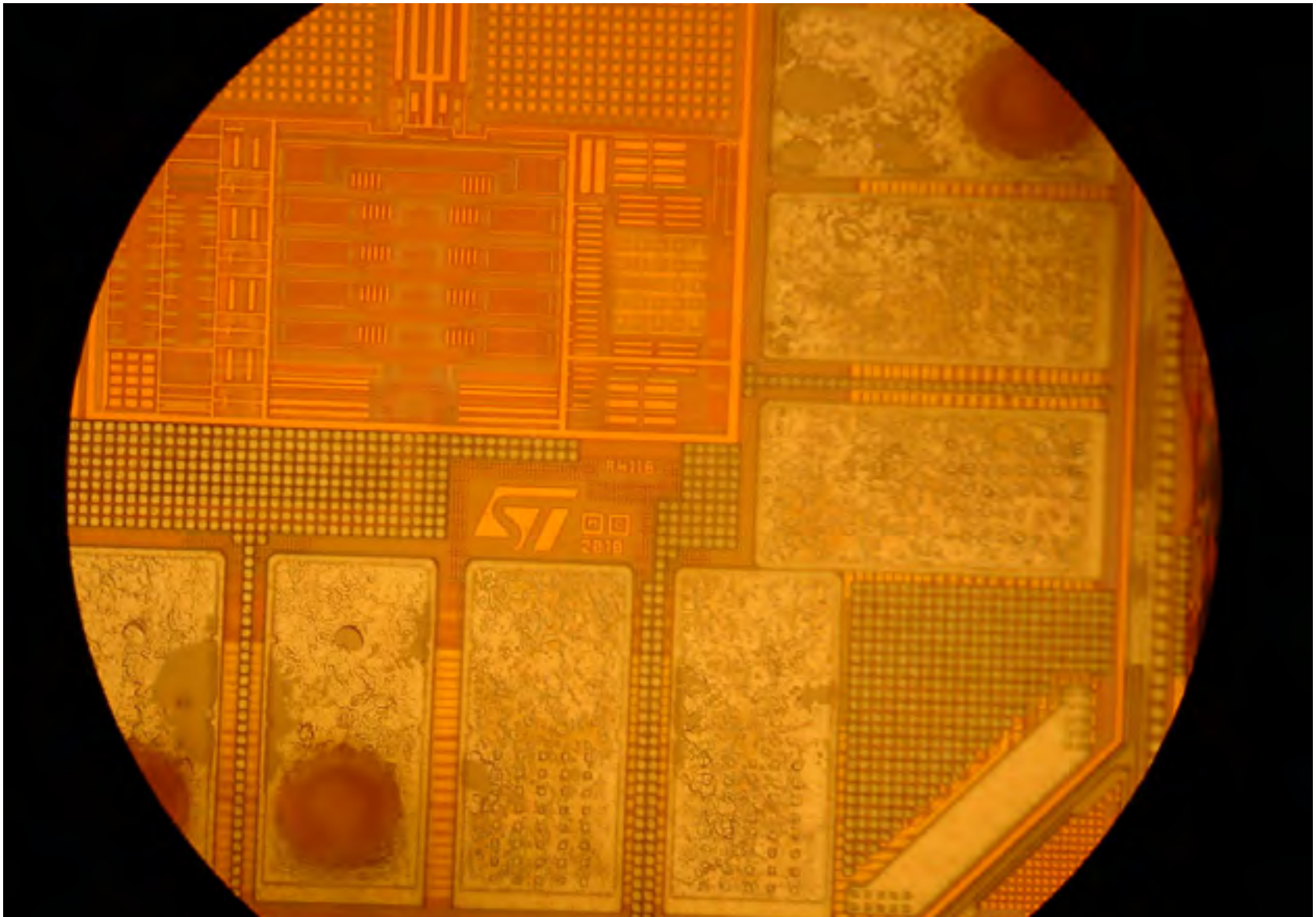
Before we get to the new hardware, we tried two other paths.

- De-scrambling the pin via OpenCV to automate testing.
- Decapping the STM32F205

I spy with my little eye



Decap all the things!



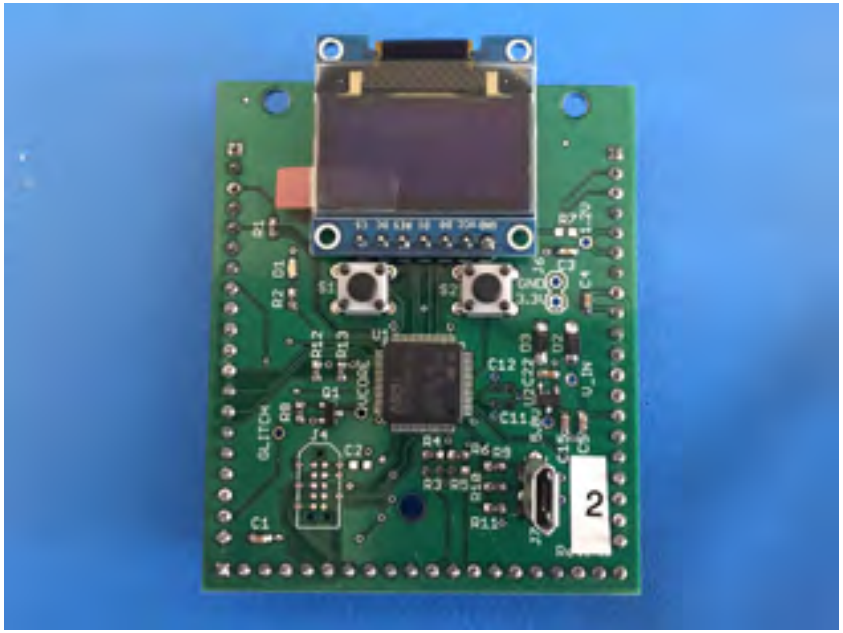
We are silicon n00bs

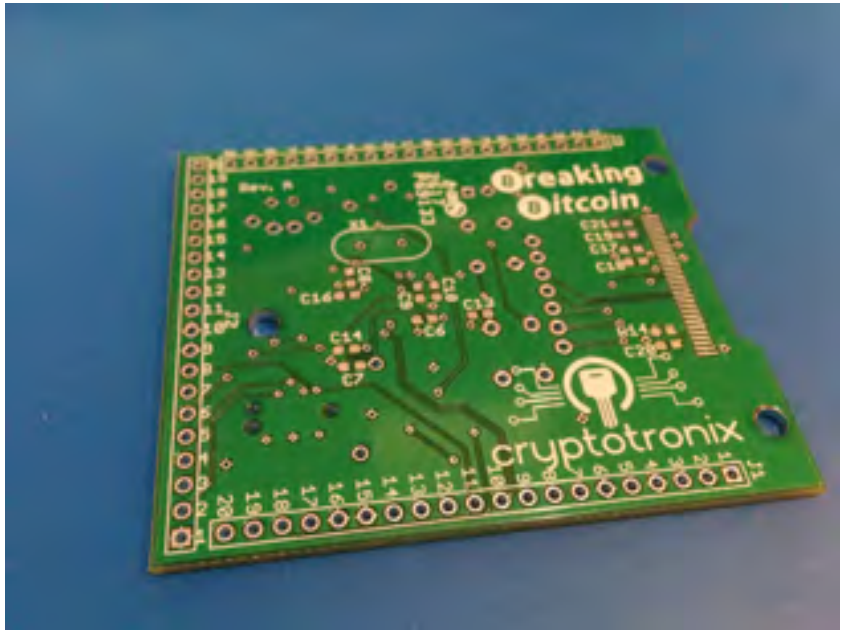
- TBH, I just wanted to a cool silicon pic for DEF CON :)
- Decapping-as-a-Service exists though (Dangerous Prototypes)
- I asked smarter people about this:
  - Cheap images don't tell you much.
  - Some interconnects are exposed.
  - Maybe flip bits during runtime?

Want more pics?

All the decap pics are on the website.

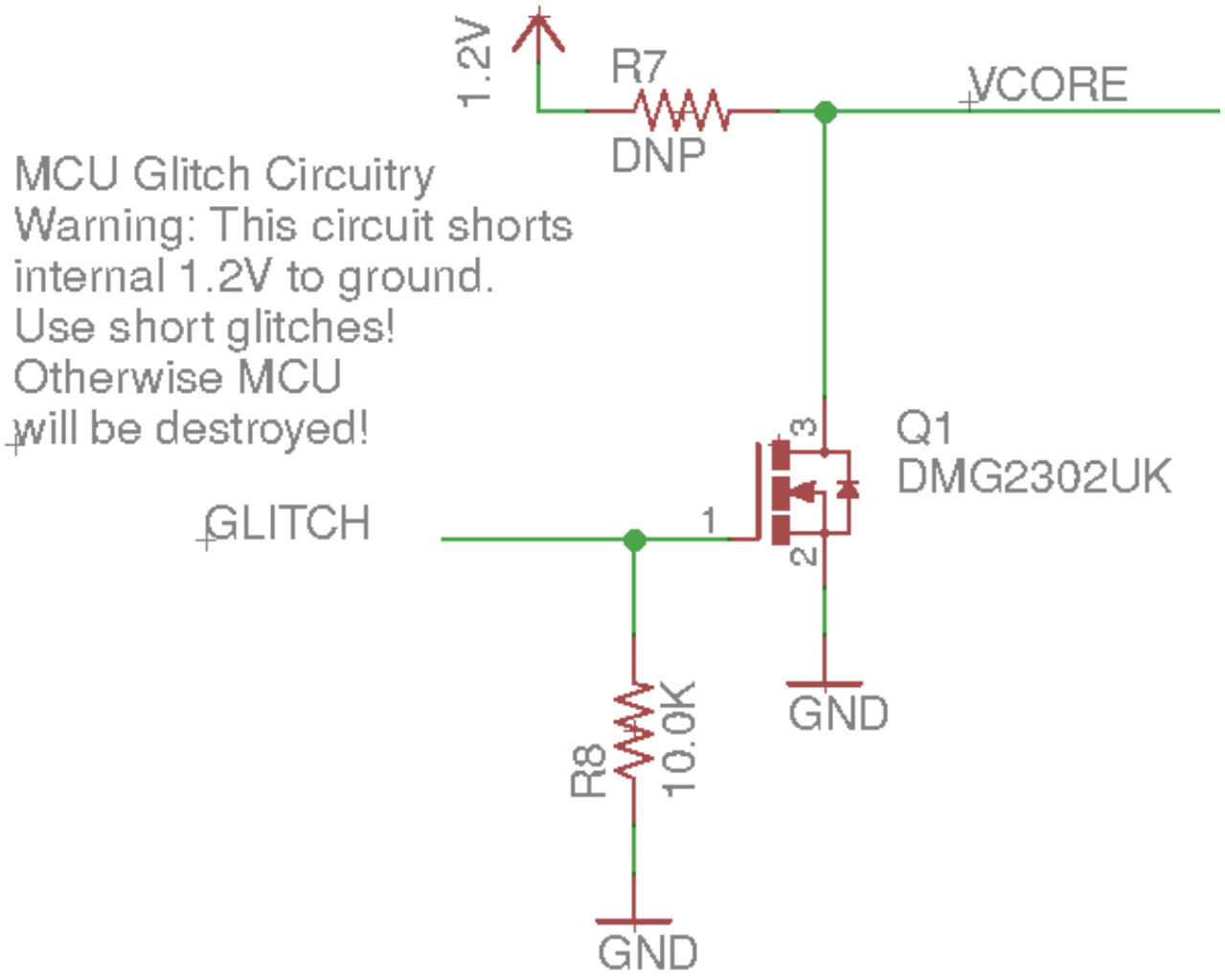
Breaking Bitcoin Board



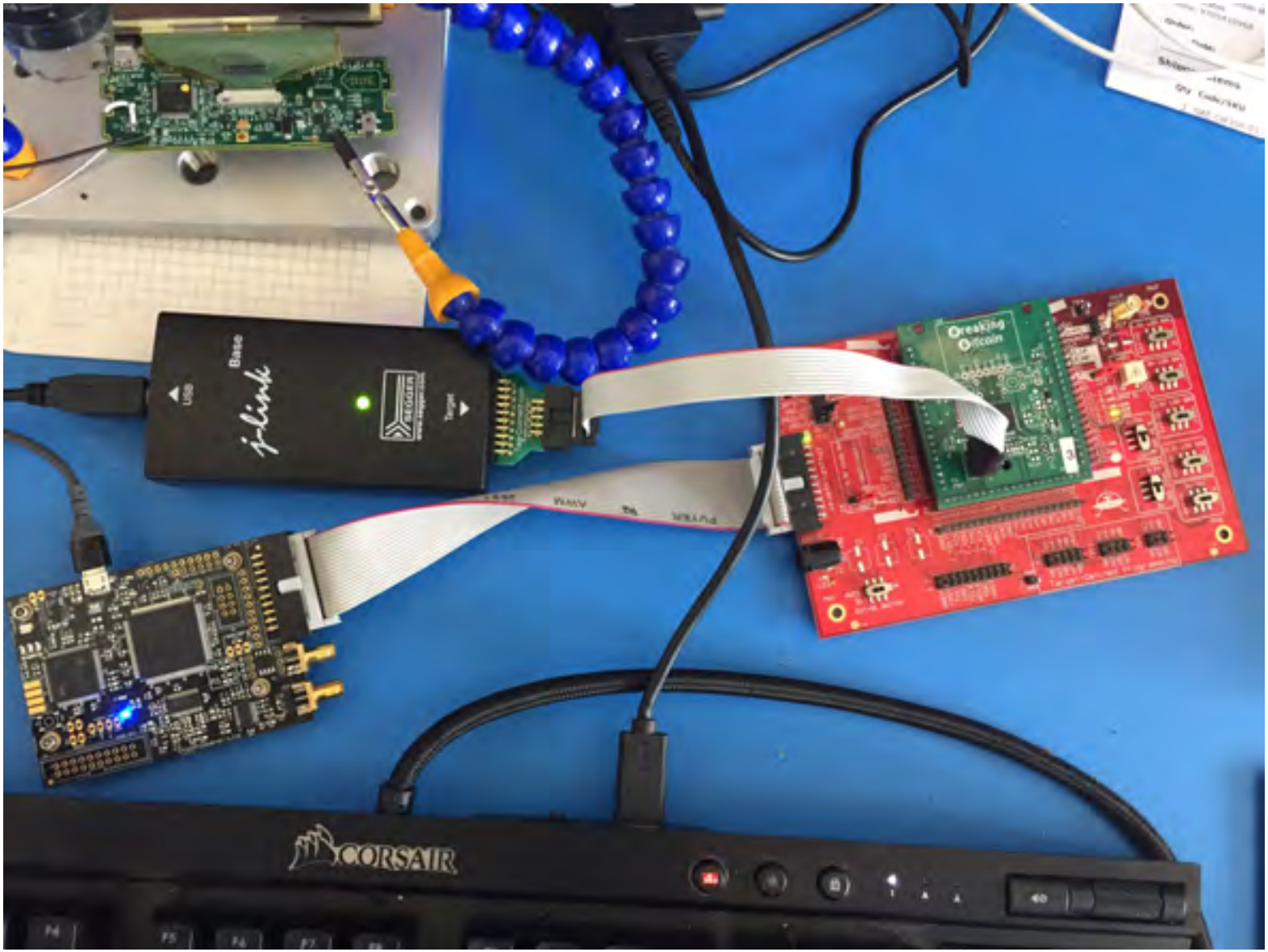


- Fits the ChipWhisperer UFO format
- It is *also* a TREZOR clone.
- Through-hole XTAL for more fun :)
- On board glitch hardware to attack *without* a ChipWhisperer

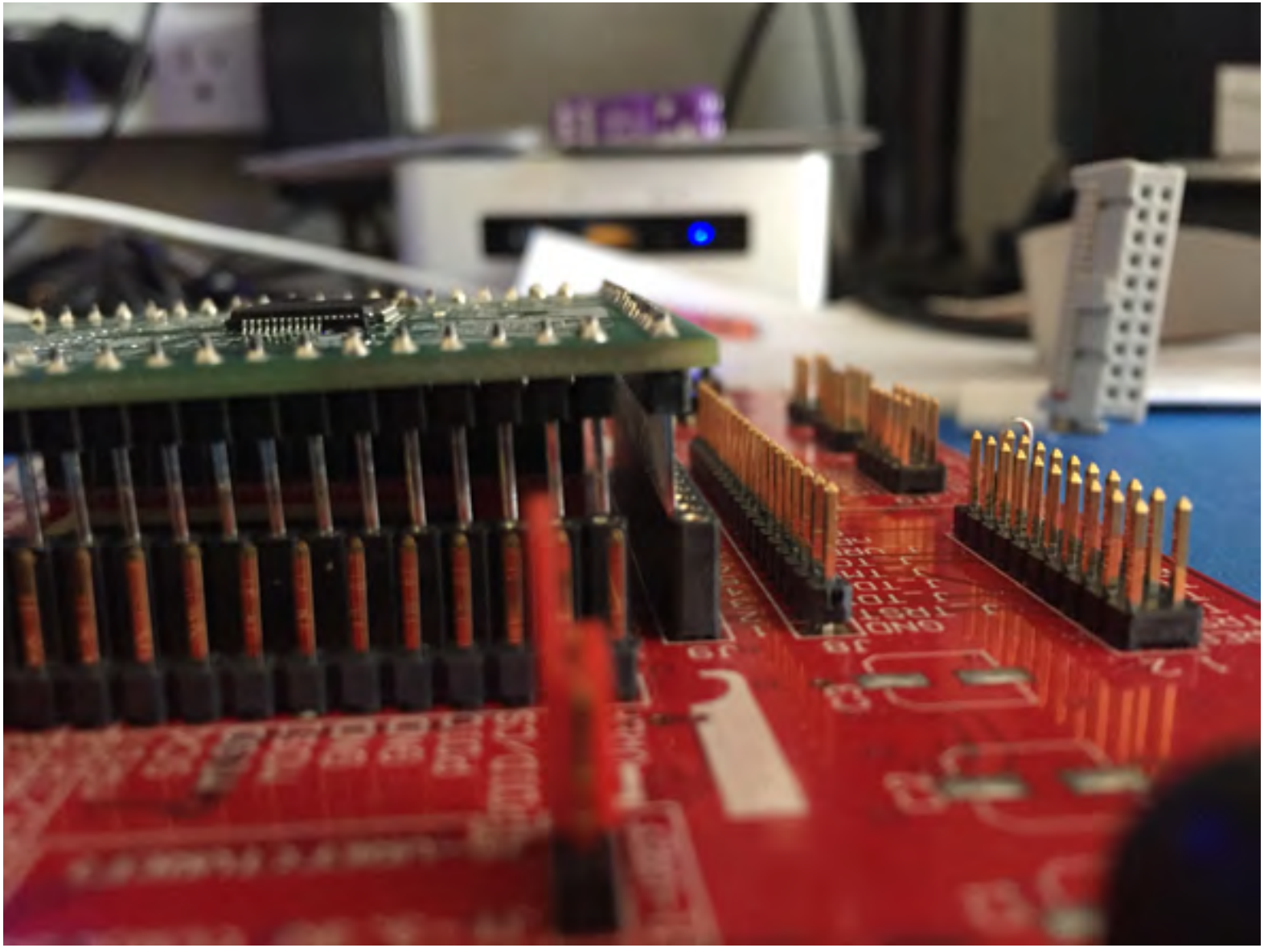
Glitch on the cheap



A better setup



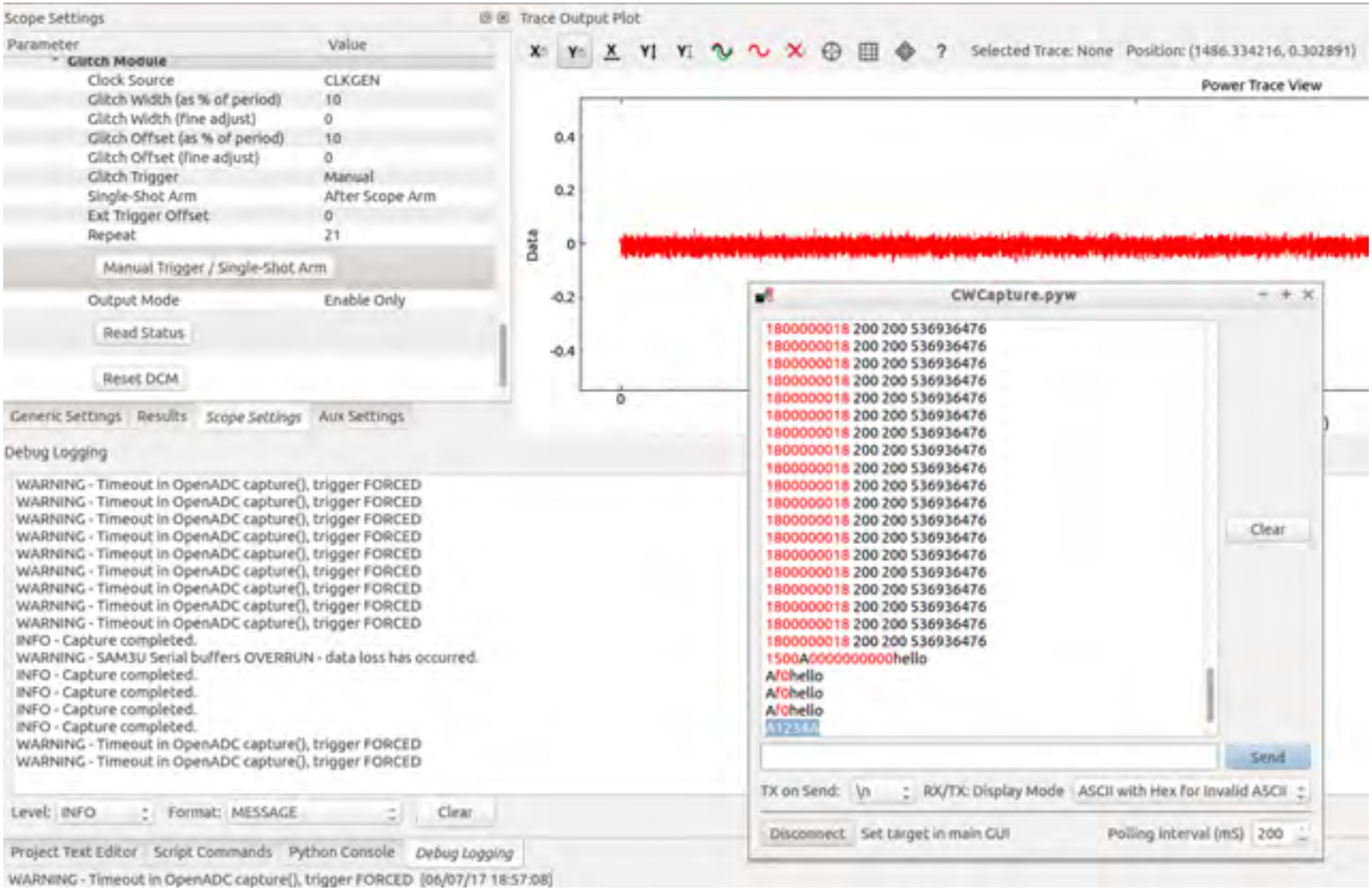
There's always a Rev B



Loop, what loop?

```
1  void glitch1(void)
2  {
3      //Some fake variable
4      volatile uint8_t a = 0;
5      putch('A');
6      //Should be an infinite loop
7
8      while(a != 2){;}
9
10     uart_puts("1234");
11     while(1){;}
12 }
```

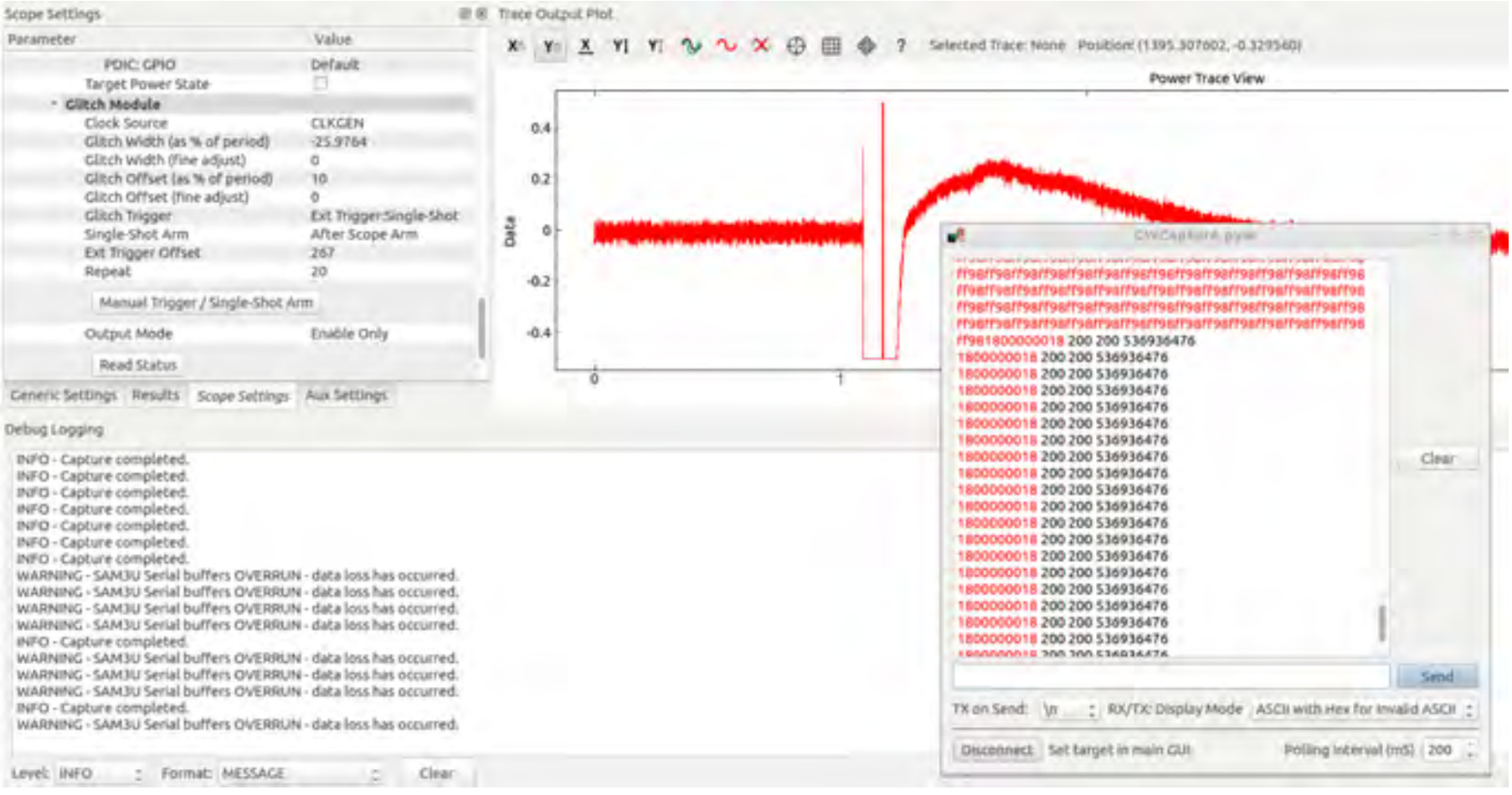
Loop, what loop?



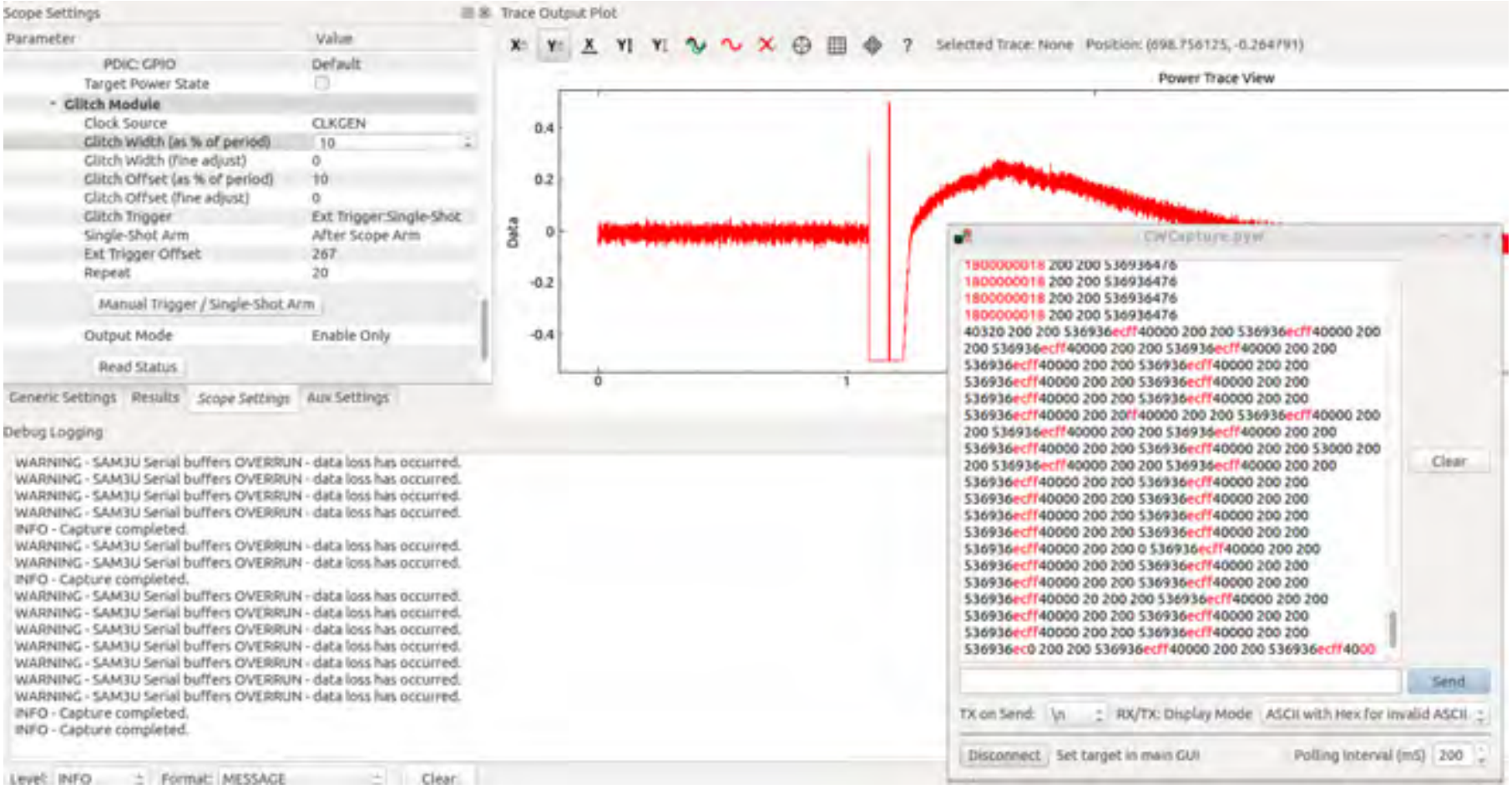
Ooof, that hurts

```
1  void glitch_infinite(void)
2  {
3      char str[64]; unsigned int k = 0;
4      //This also adds lots of SRAM access
5      volatile uint16_t i, j;
6      volatile uint32_t cnt;
7      while(1){
8          cnt = 0;trigger_high();trigger_low();
9          for(i=0; i<200; i++){
10             for(j=0; j<200; j++){cnt++;}}
11         sprintf(str, "%lu %d %d %d\n",
12                 cnt, i, j, k++);
13         uart_puts(str);}}
```

Ooof, that hurts



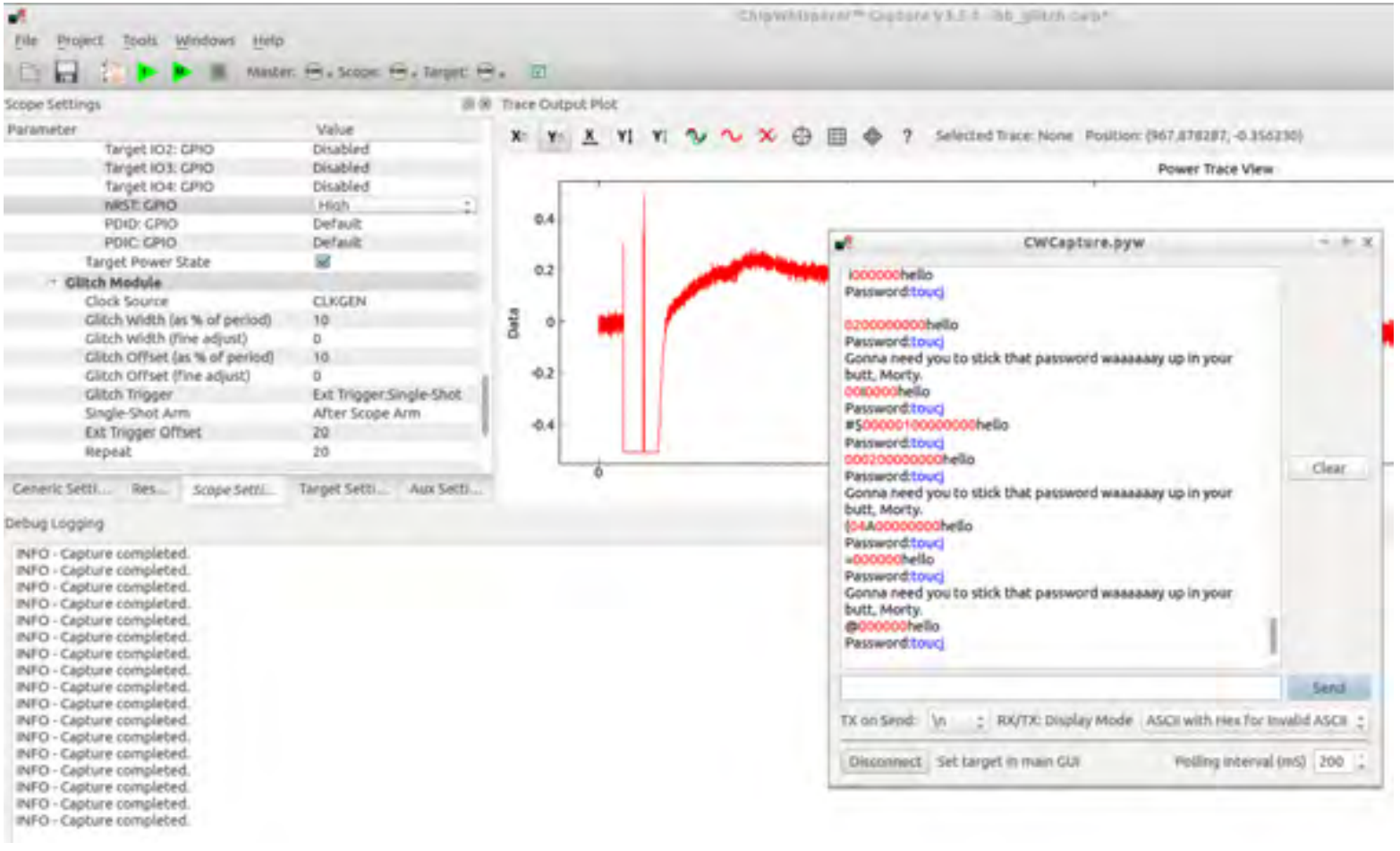
Ooof, that hurts



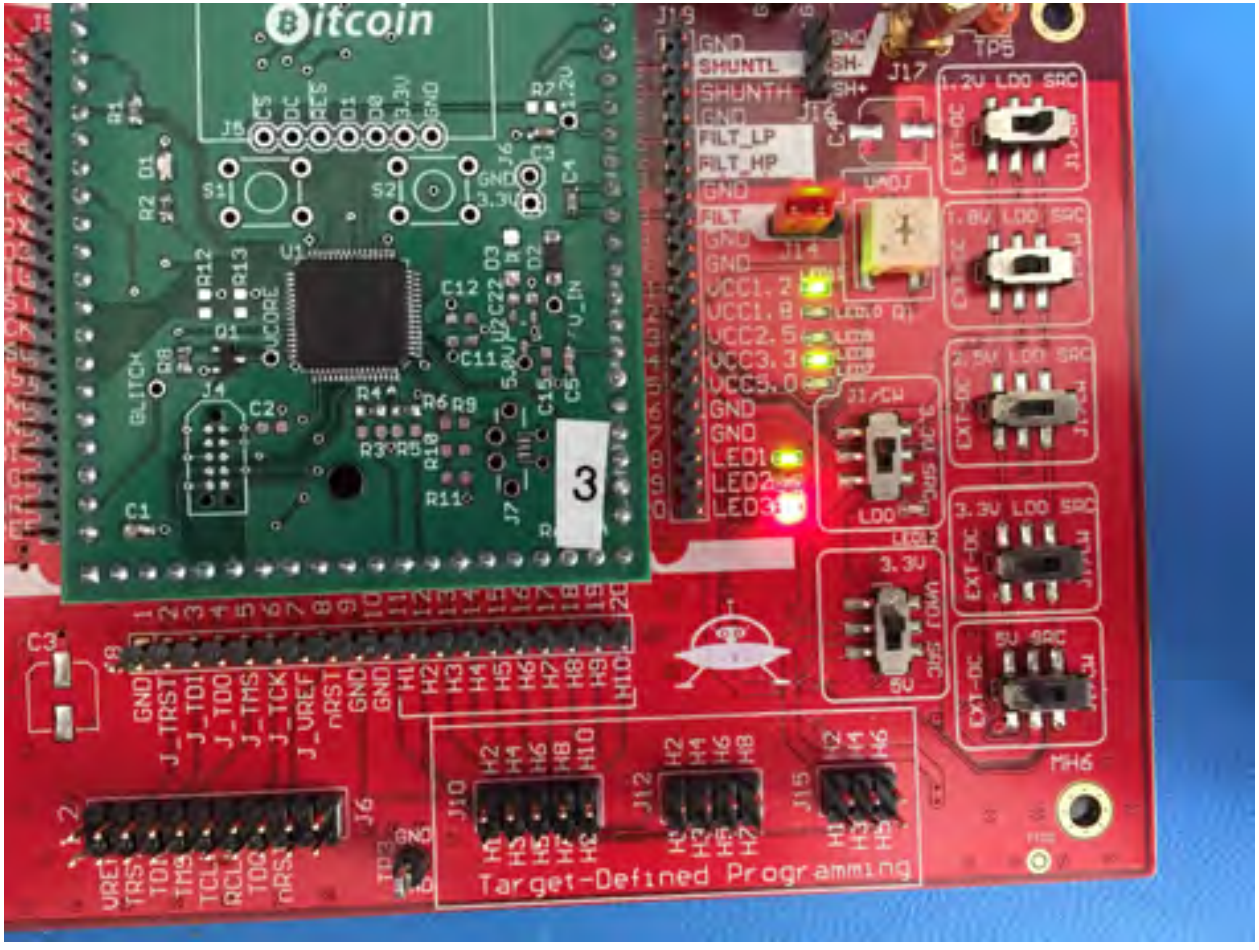
O Password, My Password

```
1  void glitch3(void)
2  {
3      char passwd[] = "touch";char passok = 1;
4      for(cnt = 0; cnt < 5; cnt++){
5          if (inp[cnt] != passwd[cnt]){
6              passok = 0;}}
7      if (!passok){
8          uart_puts("Denied\n"); while (1);
9      } else {
10         uart_puts("Welcome\n");
11     }
12
13     led_error(1);led_error(1);led_error(1);
14 }
```

O Password, My Password



O Password, My Password

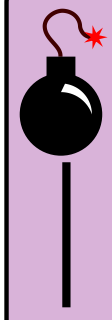


Ok, how'd we do

- Is the F205 vulnerable to fault injection?
  - Absolutely, yes.
- Is the TREZOR firmware exploitable via a fault?
  - Maybe? We have thoughts on how to trigger but going from example to exploit takes some work still.
  - We talked to TREZOR and KeepKey about some issues.
- How do we raise awareness for these kinds of attacks?
  - While not quite an *unlooper device*, our PCB will help you find the BORE (Break Once Run Everywhere) attack.

Summary of Vulnerabilities

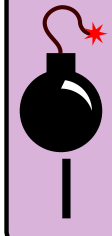
- STM32F205 is susceptible to fault attacks.
- KeepKey had a timing analysis bug on PIN verfication.
- TREZOR (and all clones) did not enable Clock Security System in the MCU, allowing injection of clock faults.
- A few pieces of code that could be made to more resilient.



Takeaway for wallet users

Don't loose physical control of your wallet.

You really want to set PIN plus password.



Takeaway for wallet designers

You will be glitched–can you trust your clock and VCC?

Defenses from Fault Attacks

Write code assuming you will be glitched! (Riscure RSA 2008)[4] and The Sorcerer's Apprentice Guide to Fault Attacks.

- Don't use 0 and not 0, using Hamming distance.
- Count your functions!
- Check for complete loop completion.
- Add Random delay–makes triggering a bit harder.
- Check sensitive operations multiple times and compare results.
- Use multiple MCUs and check results?!

Live Demo!

Chipwhisperer vs. STM32F205

Let's see some glitches!!!

Endnotes

[1] https://wiki.newae.com/File:Cwlite_basic.png

[2] *Encyclopedia of Cryptography and Security, 2nd Edition.*

[3] https://www.slideshare.net/EricLarcheveque/bitcoin-hardware-wallets-security

[4] https://cryptotronix.files.wordpress.com/2017/07/paper_side_channel_patterns.pdf