black hat ASIA 2020

OCTOBER 1-2, 2020
BRIEFINGS

Raiden Glitching Framework

by Grzegorz Wypych (h0rac) and Adam Laurie (M@jor Malfunction)

Bio

Grzegorz Wypych has worked in IT industry for more than 15 years as network engineer, architect, developer and security expert. He is a speaker on major local and international conferences like: Black Hat, hardwear.io, Security PWNing. He is also security tool inventor and developer. His experience is not only software security but also hardware and IoT. During his security career Grzegorz reported many major CVEs for different devices and firmware's. He holds master degree of computer science. Currently he is working as IBM X-Force Red Senior Security Consultant. He lives in Poland

```
{
    Name: "Grzegorz Wypych",
    nick: h0rac,
    age: 37
 }
```

Bio

```
{
    Name: "Adam Laurie",
    nick: Major Malfunction,
    age: Old Skool
}
```

Adam Laurie, a.k.a. "Major Malfunction" has four decades of information technology and security experience. He has been focused on cyber security for over 30 years and pioneered the concept of re-using military data centers housed in underground nuclear bunkers as secure hosting facilities.

For the last 15 years he has been focused on hardware security, and has found vulnerabilities in processors, embedded systems, smart meters and other critical infrastructure, building access controls and cryptographic key stores, to name a few.

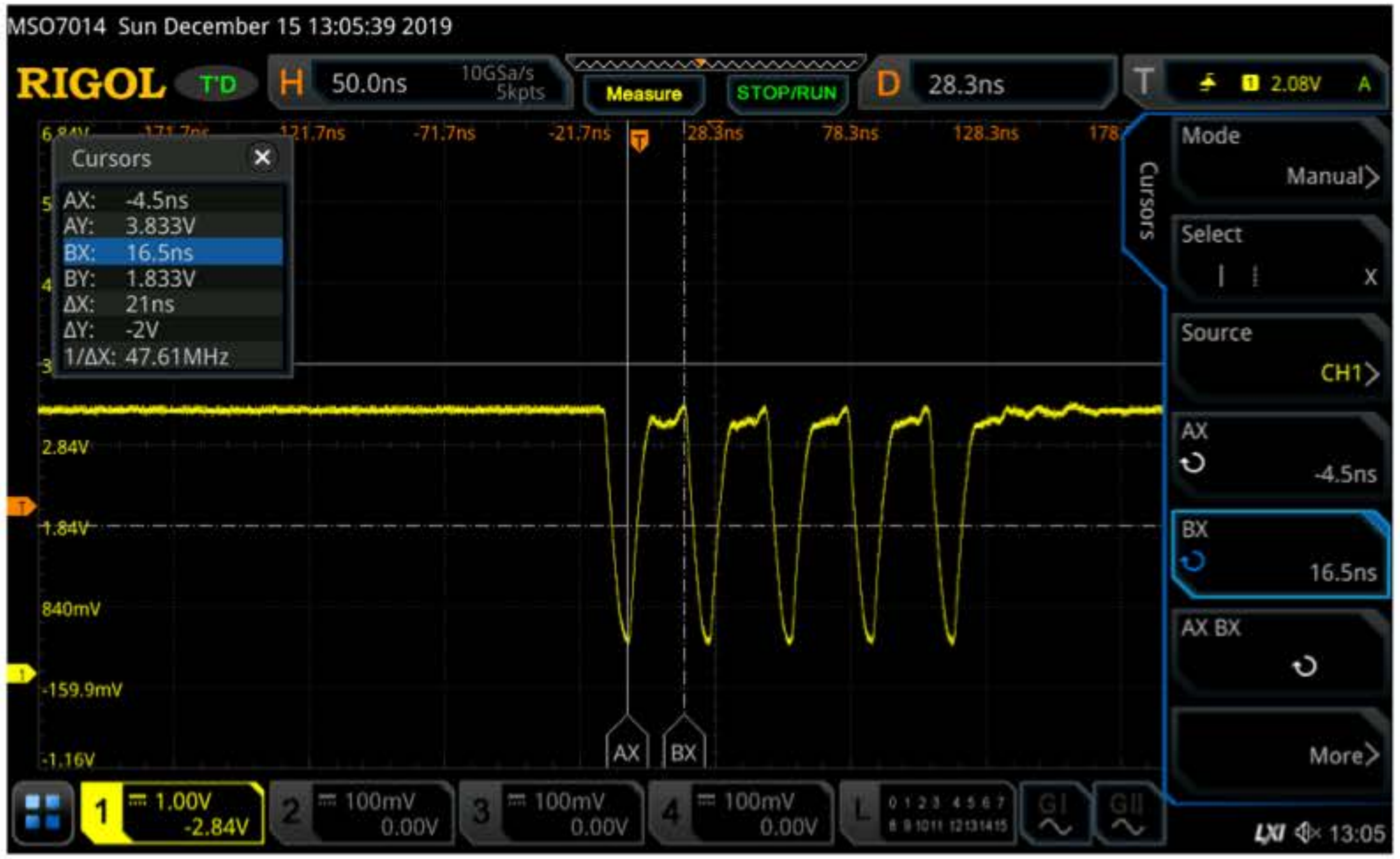
Adam has been a senior member of the DEF CON staff since 1997 and is the point-of-contact for the London DEF CON chapter DC4420. Adam has given presentations on forensics, magnetic stripe, EMV, InfraRed, RF, RFID, terrestrial and satellite TV hacking, and, of course, Magic Moonbeams.

Adam has been hacking since he was 16 years old.

What is Raiden ?

Glitching tool:

Precisely controlled fault injection – usually to the clock or power lines, but can be induced via EMP, light, heat etc.
With precision can skip or corrupt specific instruction – e.g. access control decision – ’no’ becomes ‘yes’
Without precision will usually cause a crash – e.g. overclocking too fast or overheating



MSO7014 Sun December 15 13:05:39 2019

H 50.0ns 10GSa/s 5kpts
D 28.3ns
T 2.08V A

Cursors

AX:	-4.5ns
AY:	3.833V
BX:	16.5ns
BY:	1.833V
ΔX:	21ns
ΔY:	-2V
1/ΔX:	47.61MHz

Mode: Manual
Source: CH1
AX: -4.5ns
BX: 16.5ns

1 1.00V -2.84V
2 100mV 0.00V
3 100mV 0.00V
4 100mV 0.00V

13:05

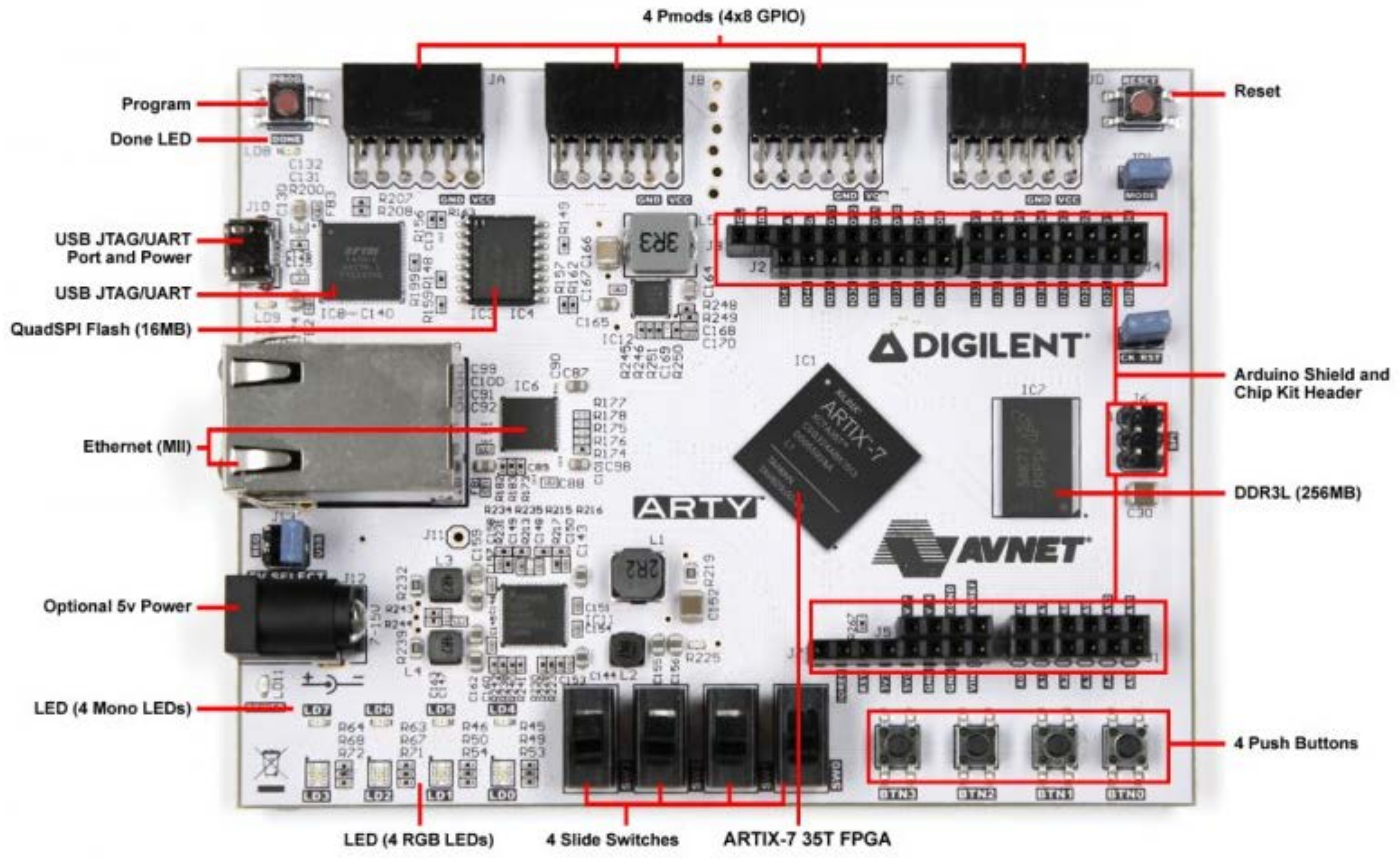
What we expected from Raiden when we started project:

1) Make a tool as much as possible independent from hardware.
2) Add features that existing solutions are missing.
3) Add flexibility allowing code portability between different FPGA.
4) Create easy and simple API to access Raiden functionality.
5) Allow it to work with voltage glitching frontend (MAX switches) and EMFI probes.
6) Find a way to scale hardware power if required.
7) Cost efficient.
8) Open source code for community.

Raiden hardware

Raiden tested on Xilinx Artix-7 35T

- Standard FPGA development board
- FPGA Xilinx Artix-7 XC7A35T-L1CSG324I
- Easy access to UART via USB
- Customisable
- Reach I/O access
- Cost around 200$ US



What Raiden is ?

- Written in Verilog using Xilinx Design Studio (1 year subscription in price with dev board)
- Portable code, only constraints file need to be updated to work with other FPGas
- Advanced pulse generator instead fixed hardware depended Glitcher
- Flexible solution to connect to any Glitching frontend (EMFI, MAX switches)
- Free to update/modify Verilog code
- Solution tested in the battle and proved to give CVEs :)

Raiden features

Common signal generator features

- Glitch delay
- Glitch width
- Glitch count

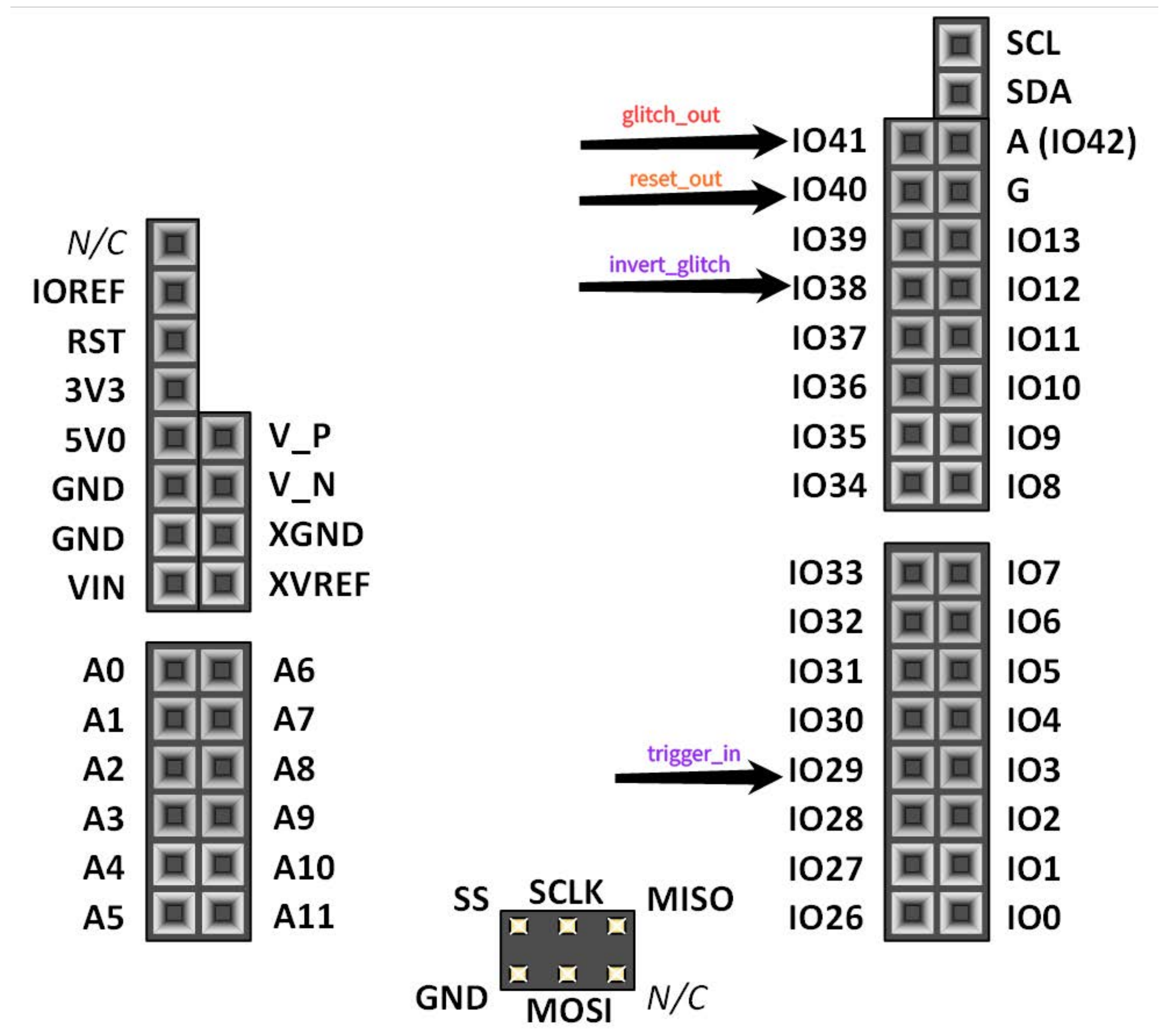
Raiden unique features

- Reset out (possibility to generate and send reset signal to target device and trigger on it)
- Vstart - Voltage start (Low or High, allow to power on/off target device)
- Glitch inversion (you can invert glitch signal)
- Flexible gap setting between glitch signals
- Glitch Max - maximum number of glitches (TBD explained @major)

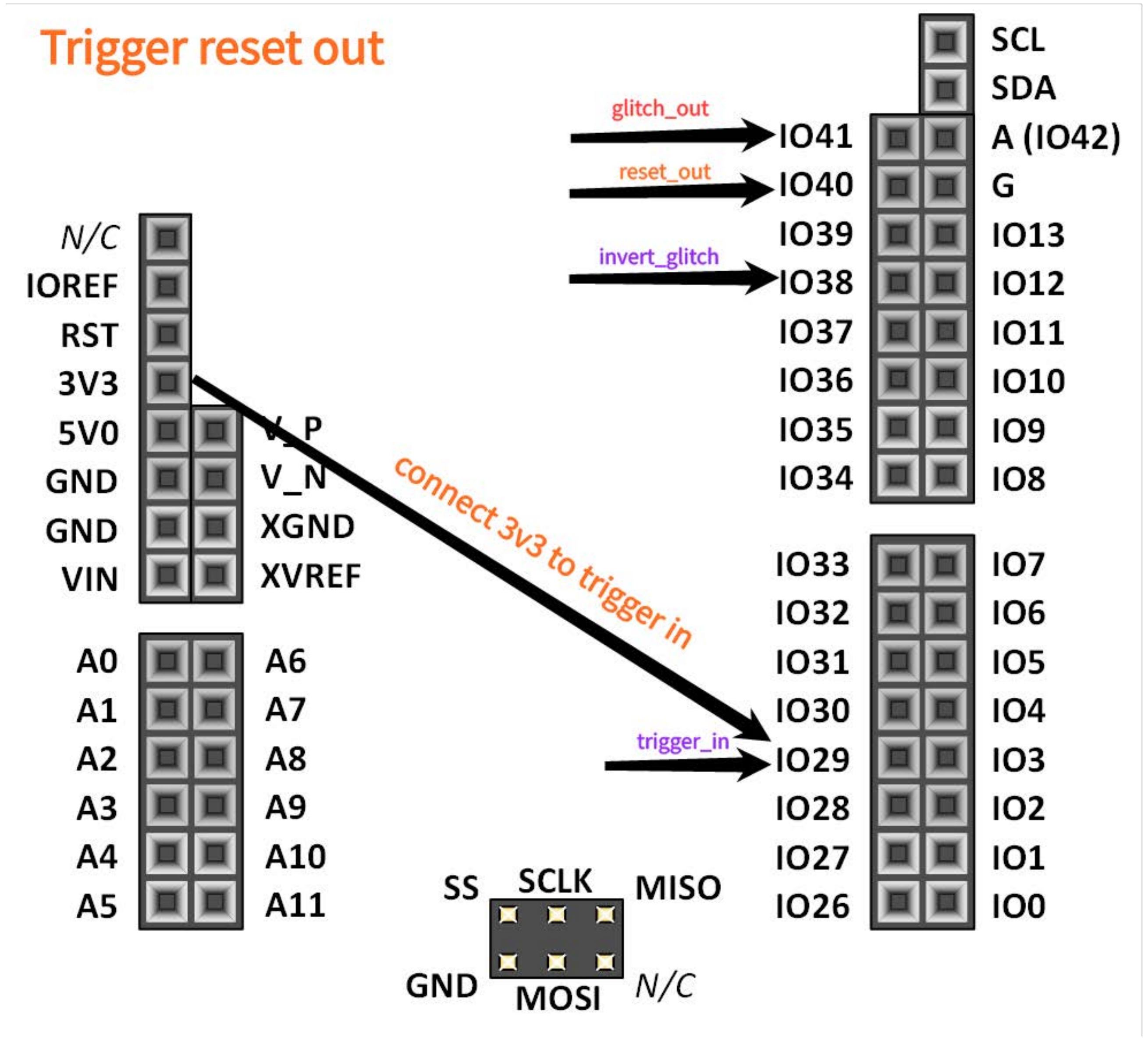
Inside Raiden

Raiden Pinout

Raiden connection with external trigger



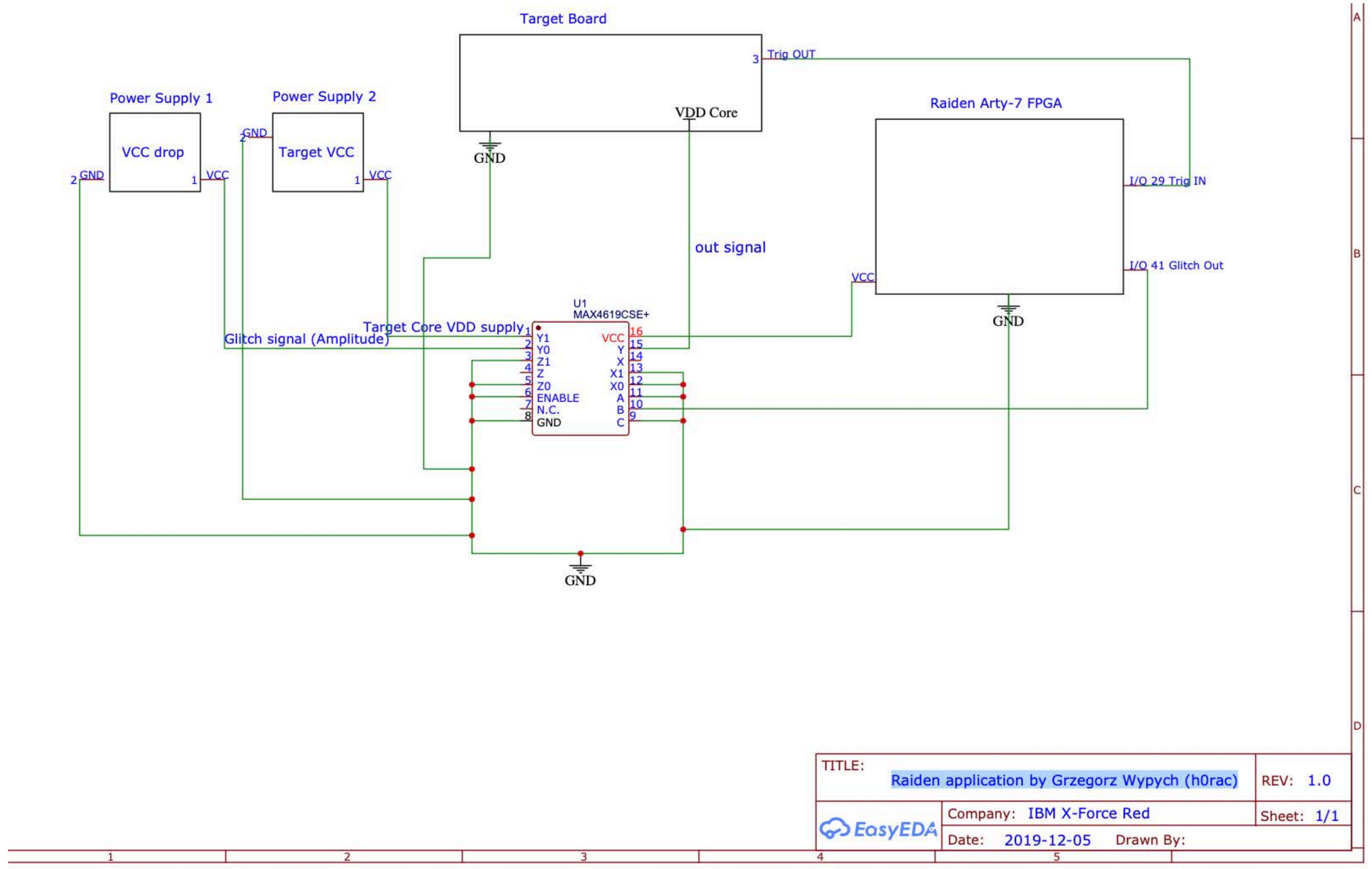
Raiden connection with internal trigger



Raiden Frontend

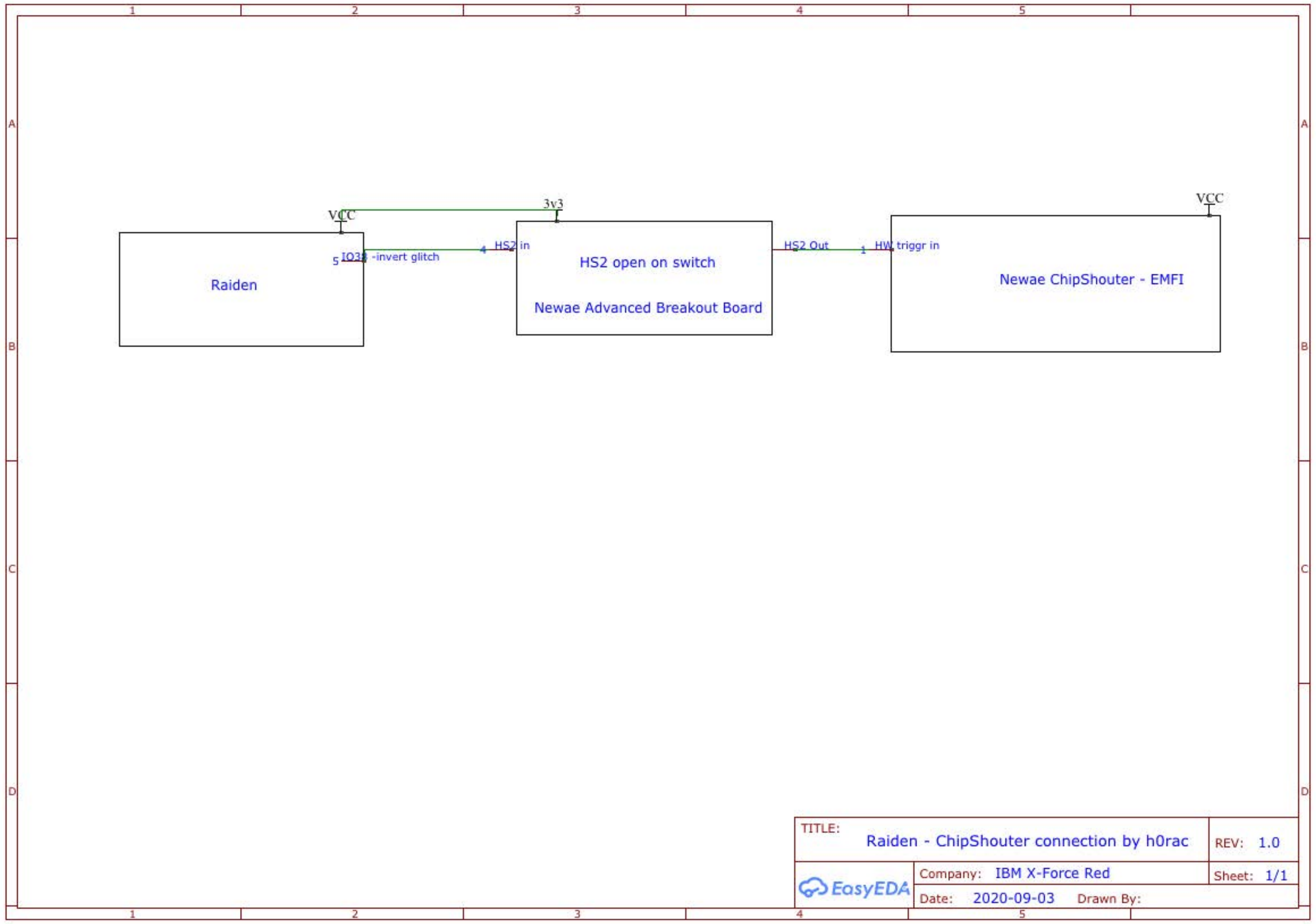
Raiden Applications

Raiden connection with MAX4619CPE

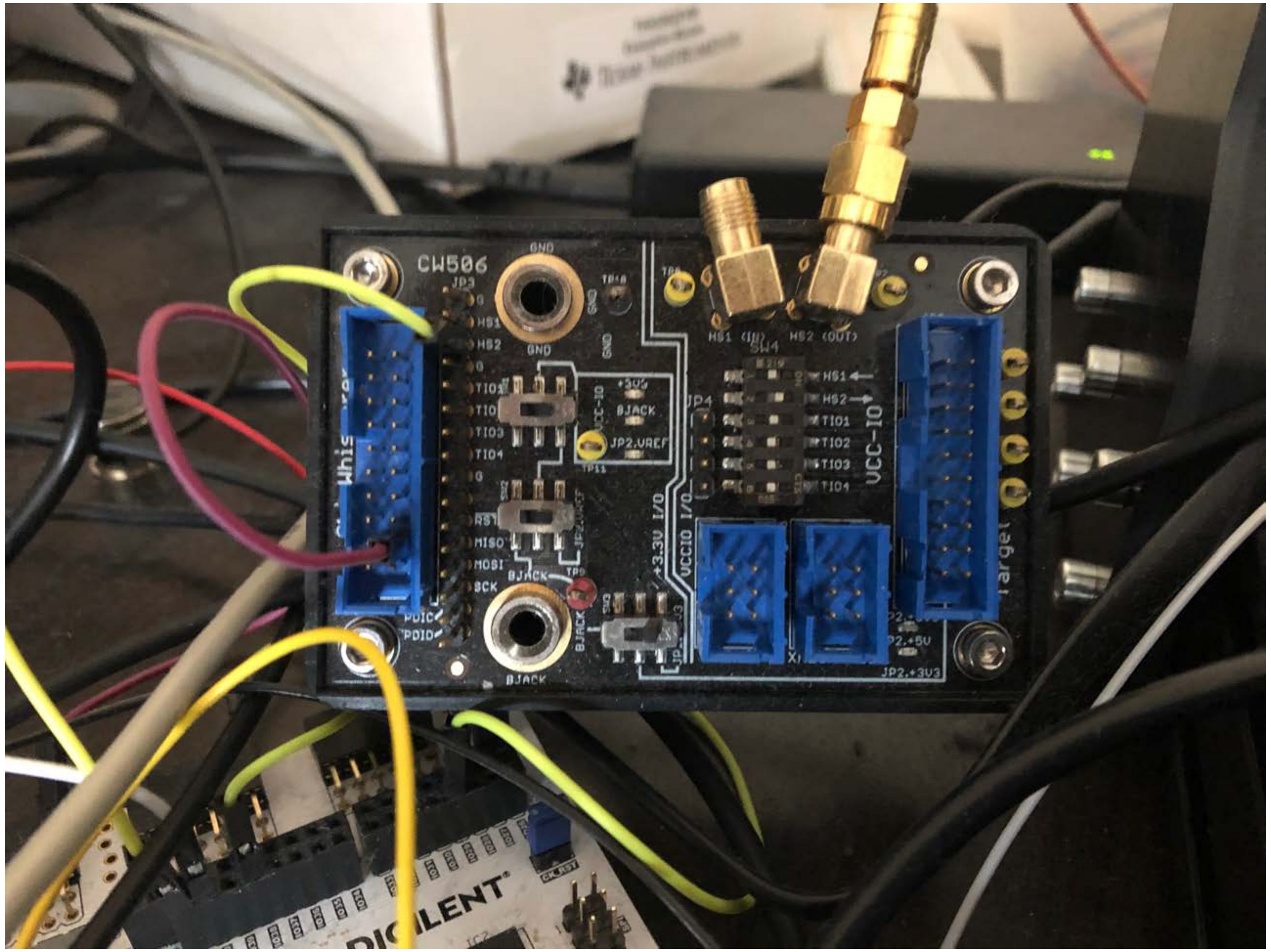


#BHASIA @BLACKHATEVENTS

Raiden connection with ChipShouter



Raiden to ChipShouter EMFI connection

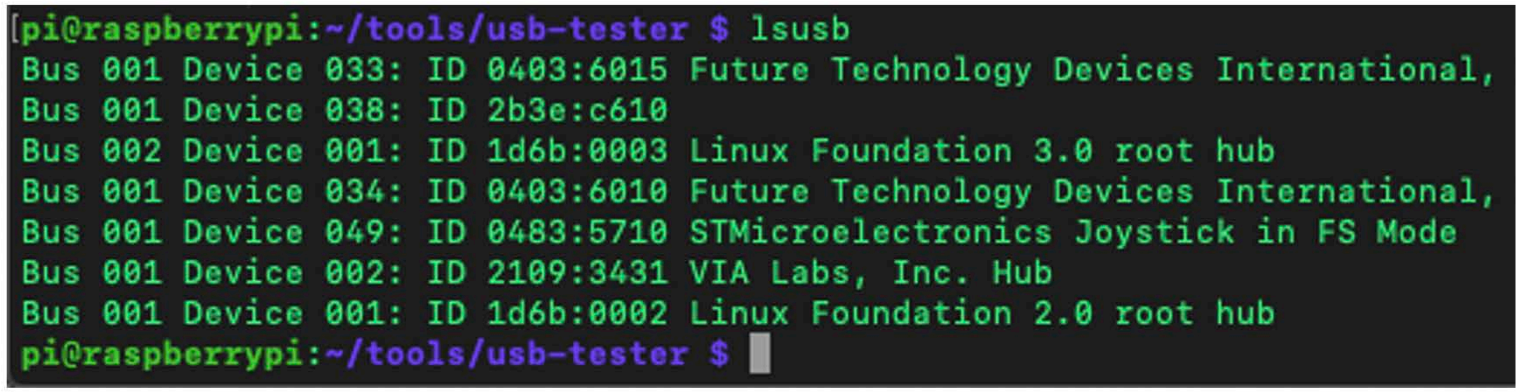


Raiden in the battle (Demo 1)

Raiden in the battle (Demo 2)

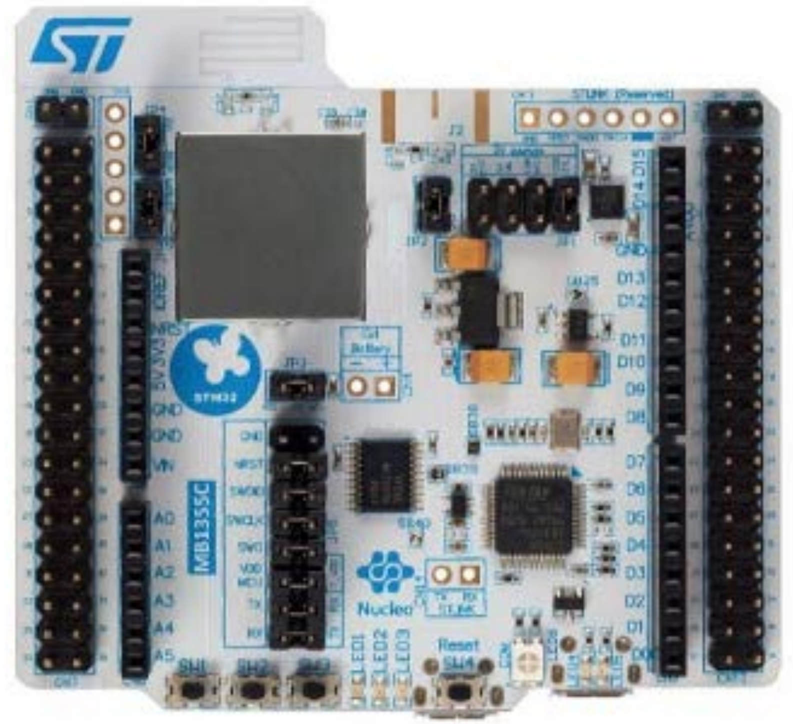
Target device STM32WB55, testing USB library

- Board is supported by latest version of USB middleware library

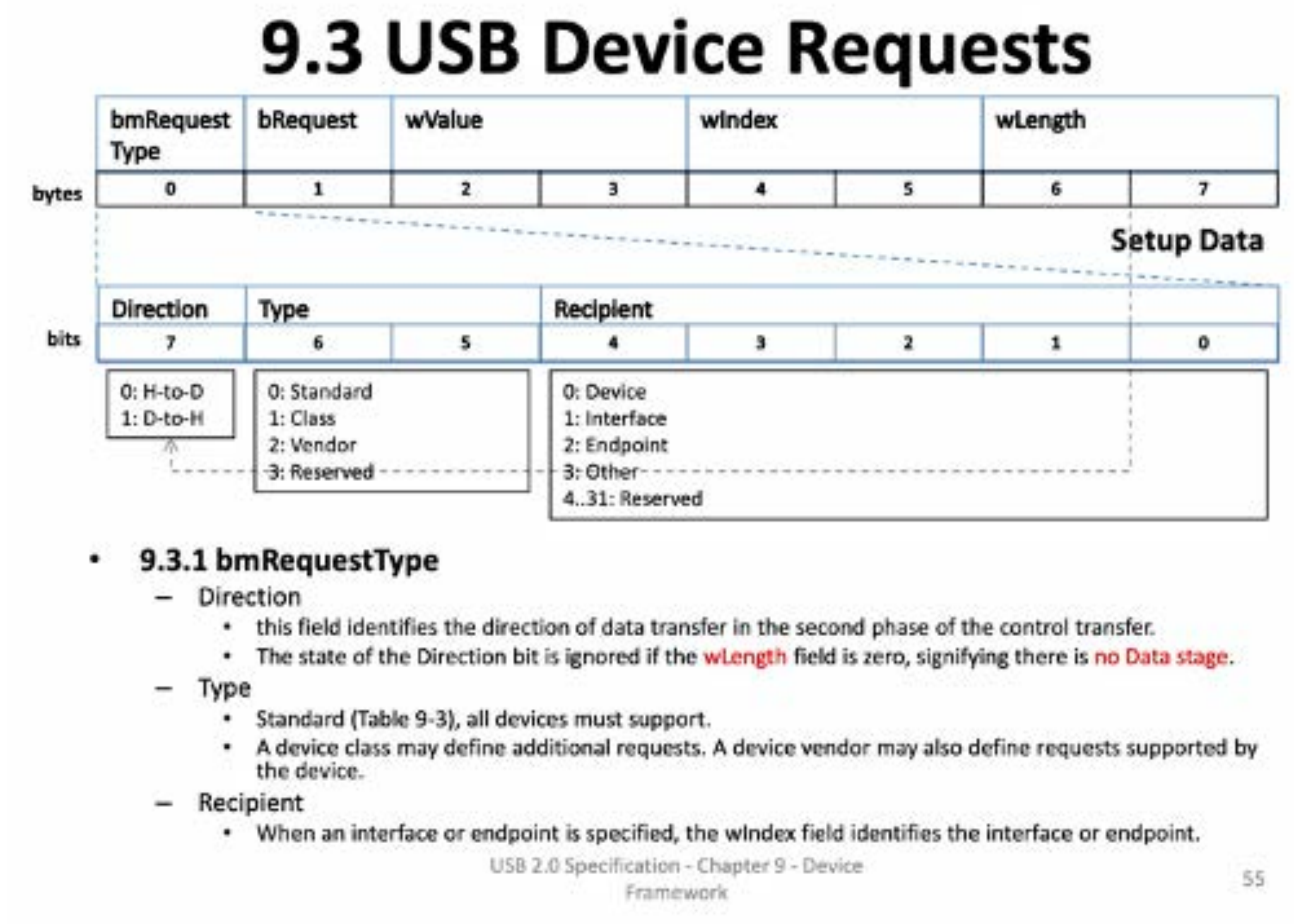


```
pi@raspberrypi:~/tools/usb-tester $ lsusb
Bus 001 Device 033: ID 0403:6015 Future Technology Devices International,
Bus 001 Device 038: ID 2b3e:c610
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 034: ID 0403:6010 Future Technology Devices International,
Bus 001 Device 049: ID 0483:5710 STMicroelectronics Joystick in FS Mode
Bus 001 Device 002: ID 2109:3431 VIA Labs, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
pi@raspberrypi:~/tools/usb-tester $ 
```

[illegible]



Quick USB descriptors look



9.3 USB Device Requests

	bmRequest Type	bRequest	wValue		wIndex		wLength	
bytes	0	1	2	3	4	5	6	7

Setup Data

	Direction	Type		Recipient				
bits	7	6	5	4	3	2	1	0

0: H-to-D
1: D-to-H

0: Standard
1: Class
2: Vendor
3: Reserved

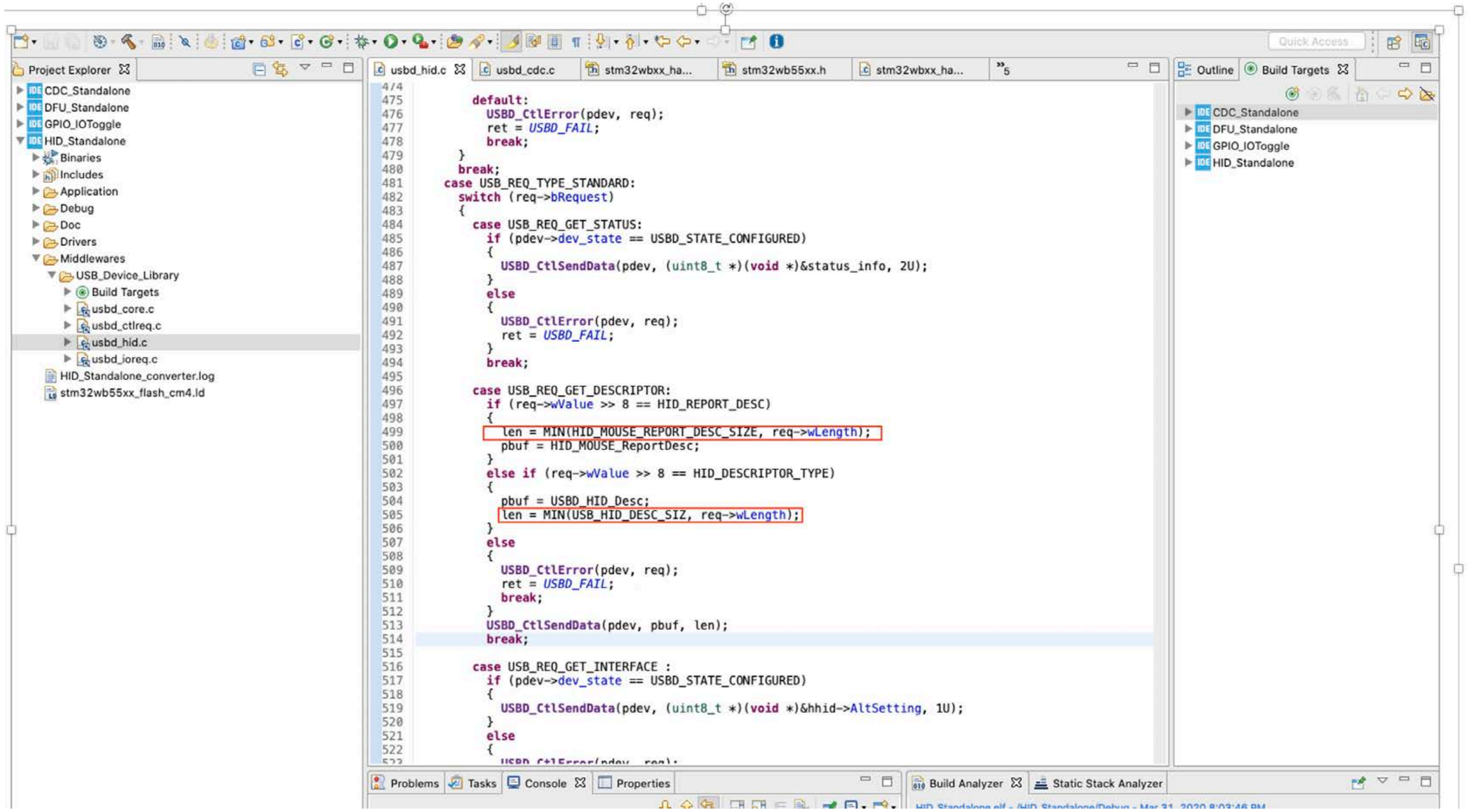
0: Device
1: Interface
2: Endpoint
3: Other
4..31: Reserved

- **9.3.1 bmRequestType**
  - Direction
    - this field identifies the direction of data transfer in the second phase of the control transfer.
    - The state of the Direction bit is ignored if the wLength field is zero, signifying there is no Data stage.
  - Type
    - Standard (Table 9-3), all devices must support.
    - A device class may define additional requests. A device vendor may also define requests supported by the device.
  - Recipient
    - When an interface or endpoint is specified, the wIndex field identifies the interface or endpoint.

USB 2.0 Specification - Chapter 9 - Device Framework

55

STM32CubeX Microcontroller middleware 0day

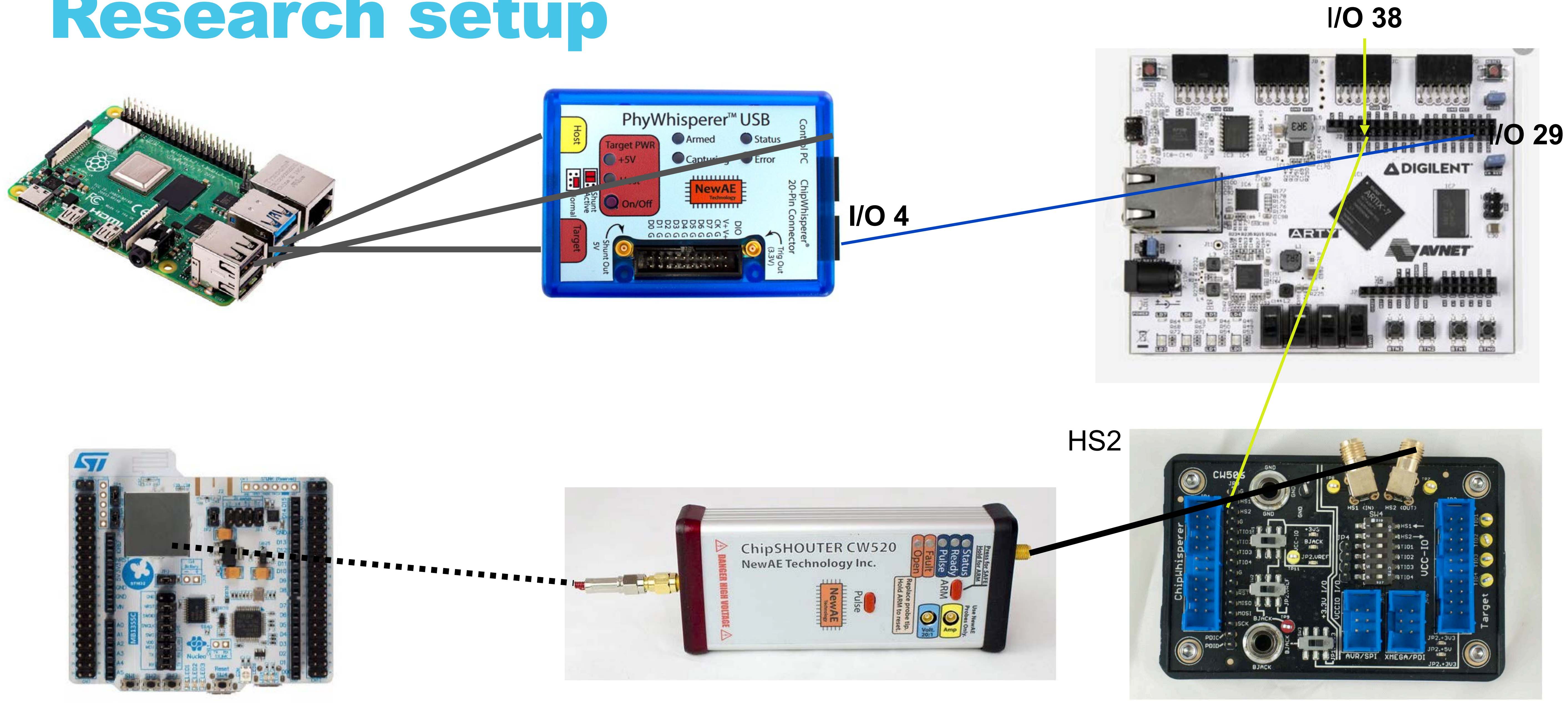


```
        default:
          USBD_CtlError(pdev, req);
          ret = USBD_FAIL;
          break;
      }
      break;
    case USB_REQ_TYPE_STANDARD:
      switch (req->bRequest)
      {
        case USB_REQ_GET_STATUS:
          if (pdev->dev_state == USBD_STATE_CONFIGURED)
          {
            USBD_CtlSendData(pdev, (uint8_t *)(void *)&status_info, 2U);
          }
          else
          {
            USBD_CtlError(pdev, req);
            ret = USBD_FAIL;
          }
          break;

        case USB_REQ_GET_DESCRIPTOR:
          if (req->wValue >> 8 == HID_REPORT_DESC)
          {
            len = MIN(HID_MOUSE_REPORT_DESC_SIZE, req->wLength);
            pbuf = HID_MOUSE_ReportDesc;
          }
          else if (req->wValue >> 8 == HID_DESCRIPTOR_TYPE)
          {
            pbuf = USBD_HID_Desc;
            len = MIN(USB_HID_DESC_SIZ, req->wLength);
          }
          else
          {
            USBD_CtlError(pdev, req);
            ret = USBD_FAIL;
            break;
          }
          USBD_CtlSendData(pdev, pbuf, len);
          break;

        case USB_REQ_GET_INTERFACE :
          if (pdev->dev_state == USBD_STATE_CONFIGURED)
          {
            USBD_CtlSendData(pdev, (uint8_t *)(void *)&hhid->AltSetting, 1U);
          }
          else
          {
            USBD_CtlError(pdev, req);
```

Research setup



Thank you for watching !

CVE-2020-15808

STMicroelectronics could allow a remote attacker to obtain sensitive information, caused by a buffer overread in the code for the CDC Communication interface in the STM32 USB device library. By requesting a larger length of response data than the device is programmed to expect, an attacker could exploit this vulnerability using a specially crafted USB packet to obtain the device firmware, private keys, and other sensitive information.

Blog will be published here:

https://securityintelligence.com

h0rac & Major