

ACCESSIBLE USB: DEVELOPING LOW-COST USB TOOLS

KATE TEMKIN • MIKAELA SZEKELY
TEARDOWN 2019



SO, WHO ARE YOU?



Katherine/Kate Temkin (@ktemkin):

- software lead, Great Scott Gadgets
- glitch witch & open-source-tool-builder
- educational (reverse) engineer
- lauded by the Daily Mail as a “cyber criminal”

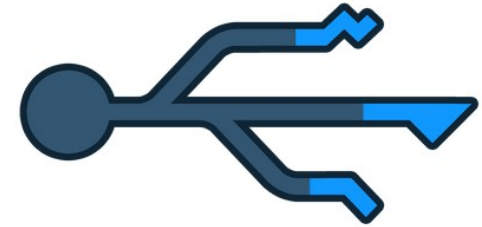
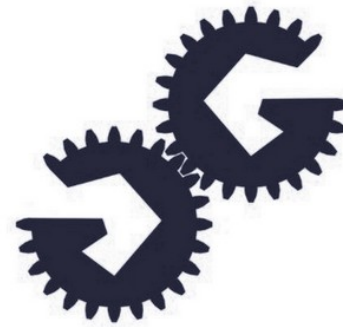
Mikaela Szekely (@Qyriad):

- student, and yet master*
- got a bit too deep in some open-source USB stuff
- apparently better at cybercrime (not caught by the Daily Mail)



The 'unpatchable' security flaw that puts EVERY Nintendo Switch at risk of being hacked by **cyber criminals**

- The bug was first reported by Denver-based security researcher Kate Temkin
- It exploits a bug in computer graphics specialist Nvidia's Tegra X1 chipsets
- Attackers force the system into USB recovery mode by short circuiting a wire
- This gives them access to the consoles most basic command level - its bootROM
- By overloading this they can then run any software or code that they wish
- This process is similar to 'jailbreaking' an iPhone or Android smartphone



Hacking the USB World with FaceDancer with [Kate Temkin](#), [Great Scott Gadgets](#), and Friends

Current iteration: [TROOPERS](#)

March 18th - 19th, 2019; Heidelberg, Germany

Older accounts: welcome to the slightly-newer CTF system!
To resume using your account, you'll need to [reset your password](#).

Getting Started & Materials

- [Approximate Schedule](#)
- [Setting up the hardware for the exercises](#)
- [Monitoring USB communications with USBMon](#)

<https://usbc.tf>

Monitoring USB Communications with USBMon

When working with USB devices, it's often helpful to have insight into the data that's being exchanged between the *host* and the *device*. There are several methods to inspect USB communications as they occur:

- A [USB protocol analyzer](#) is an expensive piece of equipment, but is the most flexible way of capturing USB communications.
- *USBProxy-Nouveau* provides a simple ability to analyze the USB data being proxied. Use of USBProxy will be covered in the training course.
- *Software analyzers* can be used on most operating systems. These analyzers are limited, and require control over the target-- but they're low cost and convenient.

In this training course, we'll use the Linux `usbmon` analyzer to capture traffic between your *host computer* and a target *USB Device*. As a software-only solution, `usbmon` is an excellent zero-cost starting point.

Setting up USBMon

USBMon performs its monitoring from inside of a *Linux kernel module*, which has full access to all USB packets processed by the system. Accordingly, we'll need to load the module before we can use it:

```
sudo modprobe usbmon
```

Starting Wireshark

The easiest way to view USBMon output is with *Wireshark*, a common suite used for protocol analysis. While it's not as full-featured as other USB analyzers, it has the significant benefit of being free.

To run Wireshark, run the following command:

```
wireshark
```

**LET'S LOOK AT
WIRESHARK**

SUPER-CHEAP USB ANALYSIS



ModuleLive

1Pcs EZ-USB FX2LP CY7C68013A USB Core Board Development Board Logic Analyzer With I2C Serial SPI Interface

★★★★★ 5.0 (3 votes) | 8 orders

Price: ~~US \$3.51~~ / piece

Discount Price: **US \$3.16** / piece -10% 23h:22m:36s

Shipping: US \$0.64 to United States via SunYou Economic Air Mail

Estimated Delivery Time: 45 days

Quantity: piece (9974 pieces available)

[Buy Now](#) [Add to Cart](#) ♥ 8

New User Coupon: **201,84 py6.** [GET IT NOW](#)



Comidox 1Set USB Logic Analyzer Channel UART IIC SPI Debug for A

by Comidox

★★★★☆ 4.0 (1 customer review)


Price: **\$6.99** ✓prime

- This item is an inexpensive logic analyzer designed to not an official Saleae product. This item is also supported.
- Sampling rate up to: 24 MHz, can be 24MHz, 16MHz, 100KHz, 50KHz, 25KHz.
- The logic for each channel sampling rate of 24M/s. On some occasions.
- A total of 8 digital channels, the voltage range is 0V considered low, above 1.5V is considered high.
- UART, SPI, IIC and other communication debugging, can automatically analyze UART, IIC, SPI and many other protocols.

Specifications for this item

Brand Name	Comidox
EAN	0661083
Part Number	CP317
UPC	661083

Roll over image to zoom in



Condition: **New**

Quantity: More than 10 available 22 sold / See feedback

Was: ~~US \$4.30~~

You save: **\$0.22 (5% off)**

Price: **US \$4.08**

[Buy It Now](#)

[Add to cart](#)

[Add to watch list](#)

eBay Money Back Guarantee
We'll make sure you get this item or get your money back. [Learn more](#)

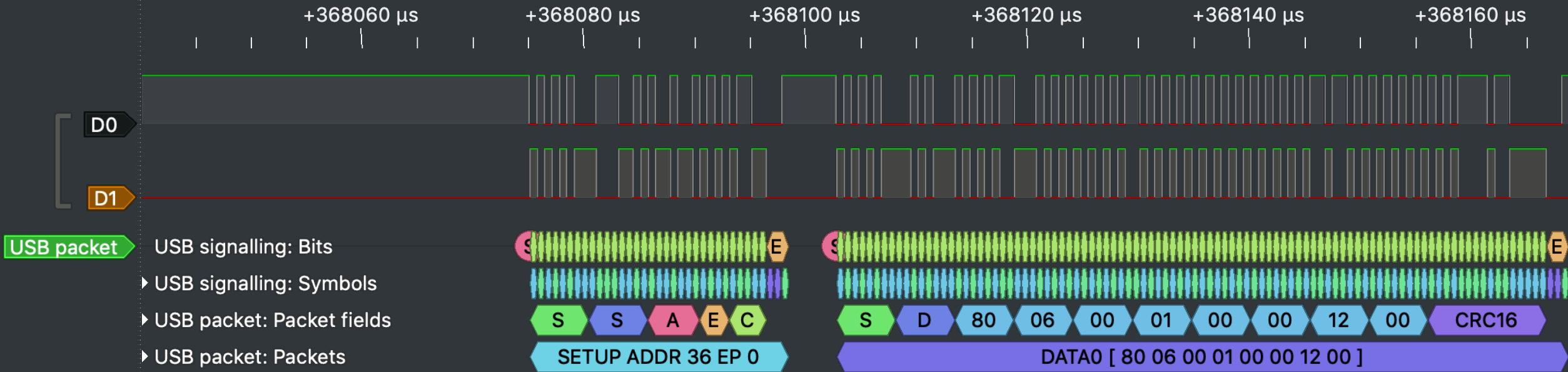
100% buyer satisfaction 22 sold Free shipping

Bucks You'll earn **\$0.04** in eBay Bucks. [See conditions](#)

Super-cheap FX2 “LA” boards are really freaking cheap, and can capture up to 24MHz via Sigrok.

That’s enough for LS/FS USB!

SIMPLE LS / FS ANALYSIS



REQUEST DECODE WITH SIGROK? **YEP**
DOES IT GO FAR ENOUGH? ...**NOT REALLY**

THE “STATE OF THE ART” IN HS ANALYSIS

Extremely useful tools...

Beagle USB 480 Protocol Analyzer

Part Number: TP320510

Distribution: Physical Shipment

Availability: In-Stock

Price: \$1,200.00

▶ Buy 5 for **\$1,100.00** each and **save 9%**

Receive 15% off any cable and 20% off any board with purchase of select devices. Discount applied at checkout.



...with extremely inaccessible price tags.

**WAIT,
ISN'T USING USB
SUPER COMPLICATED?**

– engineers who can't decode USB
because analyzers are too damned expensive

To validate that the vulnerability is present on a given device, one can try issuing an oversized request and watch as the device responds. Pictured below is the response generated when sending a oversized GET_STATUS control request with an ENDPOINT recipient to a T124:

[7 ORPHANED]	[Periodic Timeout]
Get Endpoint Status	Endpoint 00 OUT
SETUP txn	82 00 00 00 00 00 E8 03
IN txn [2 POLL]	00 00 00 00 00 00 00 00 E8 03 00 00 04 DD 00 40 01 00 00 00 00 80 00 40...
IN txn	00 00 00 00 40 25 00 40 14 02 00 00 00 40 00 40 C1 22 10 00 95 ED A0 42...
IN txn	F3 6E 4F E7 38 7F 6A 10 B7 91 7F AF 9D 5A 85 67 C0 A7 2A 25 6B 3D 10 50...
IN txn	FC FF E8 5C 00 6D 28 25 5B 7B CF 73 01 A4 22 30 79 FB B5 15 83 41 02 50...
IN txn	D3 86 BD 1A 30 40 40 15 EF FA BB FF 30 00 D3 0E D3 F1 7C 18 FC 04 10 2D...
IN txn	2A CA DC 77 CF A0 DD 1E CF 9D 7D 0E 22 87 D7 99 54 E7 9E B6 93 00 E8 70...
IN txn	57 94 64 87 B6 60 45 C0 D6 77 7D 69 46 66 B3 71 C0 88 B6 3D 3D 66 34 2B...
IN txn	A0 94 CF F3 61 46 C8 19 FE 23 DF B2 0A 40 00 00 BD 00 00 00 00 00 00 00...
IN txn	F5 72 E1 E0 75 96 D1 08 F7 E2 89 8F EE 68 07 4C EC BB F5 BB 86 48 02 29...
IN txn	19 EC CD B8 04 5F A4 1D 8E 66 DF 34 73 6A 9C A3 C3 64 BA 32 CC 00 C0 00...
IN txn	CC 00 C0 00 0C 00 00 00 C0 03 00 00 90 28 00 40 D0 21 00 40 08 00 00 00...
IN txn	20 00 00 40 04 00 00 00 51 14 10 00 00 00 00 00 00 00 00 00 01 00 00 00...
IN txn	60 C1 2A 13 D0 03 89 00 AB BE A2 F0 45 31 80 BE 98 23 EA AA 10 20 09 1D...
IN txn	F2 57 71 72 E6 C0 56 15 B2 B0 61 7B 64 44 23 20 EE C4 09 3C 97 02 00 52...
IN txn	BD FA 80 37 6B 42 E3 E8 84 EF A4 B9 95 8F 68 0E 33 7E 1F 63 41 10 65 63...
IN txn	8R R7 RF 81 78 0C 75 03 F4 RR C7 76 78 75 98 10 5D DF 4R FN CA 14 4A F1

A compliant device should generate a two-byte response to a GET_STATUS request-- but the affected Tegra responds with significantly longer response. This is a clear indication that we've run into the vulnerability described above.

"OpenVizsla" Open Source USB Protocol Analyzer



OpenVizsla is an open source, high-speed USB sniffer that will help decode, debug and hack proprietary USB hardware devices.

Created by
bushing

584 backers pledged \$81,025 to help bring this project to life.

All departments ▾

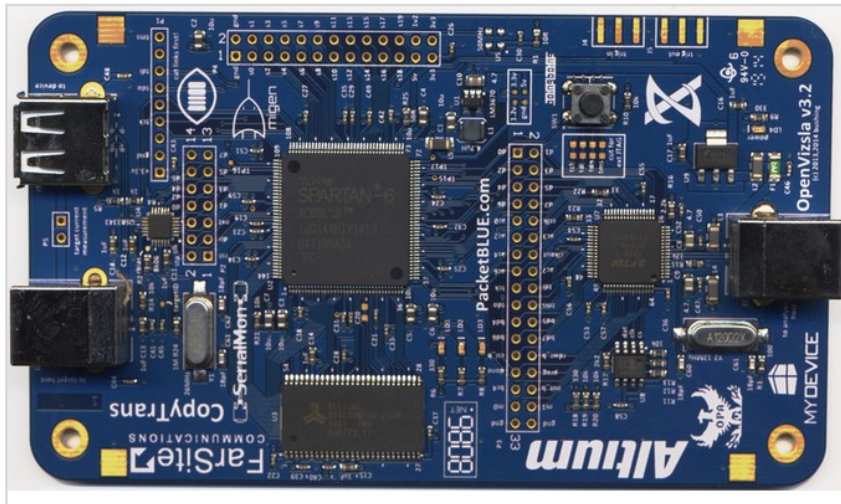
Search

Search

HOME

 CART: (EMPTY)

IMPORTANT NOTE: No orders will be handled from June 25, 2019 until July 7, 2019. Orders placed until 10:00am German local time (CEST) on June 24th will still be handled+shipped before this period of absence.



OpenVizsla v3.2 USB Protocol Analyzer PCBA

This is fully assembled and tested OpenVizsla v3.2 USB protocol analyzer.

OpenVizsla is a bus sniffer/analyzer for USB. It allows you to passively monitor the communication between a USB host and USB peripheral. It supports USB low-speed, full-speed and high-speed.

The product is shipped as a bare printed circuit board assembly, without any enclosure.

For more information about OpenVizsla, see <https://openvizsla.org/>

PRICE
119.00 € (inc. VAT)

1 

Add To Cart

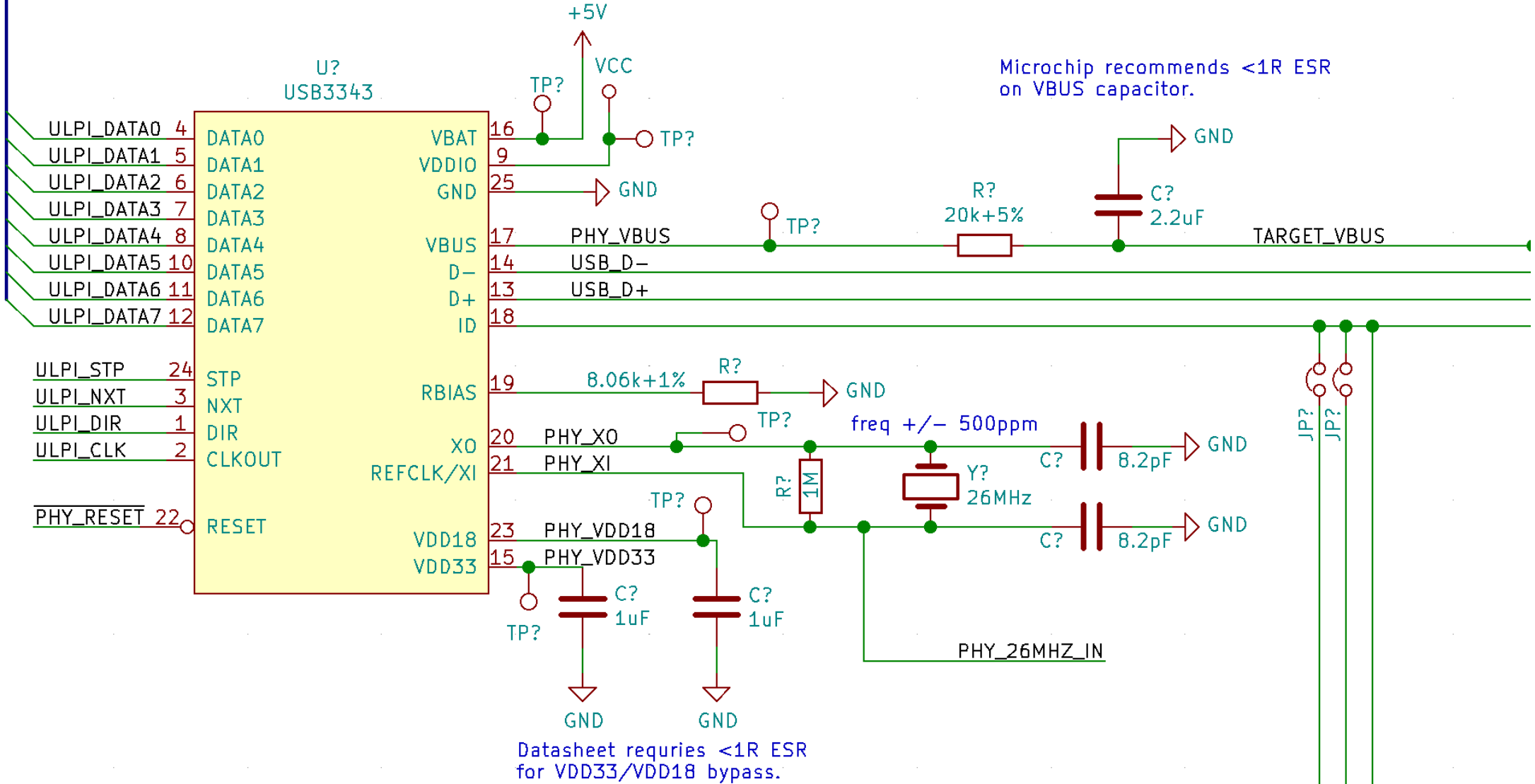
LOOK FOR SIMILAR ITEMS

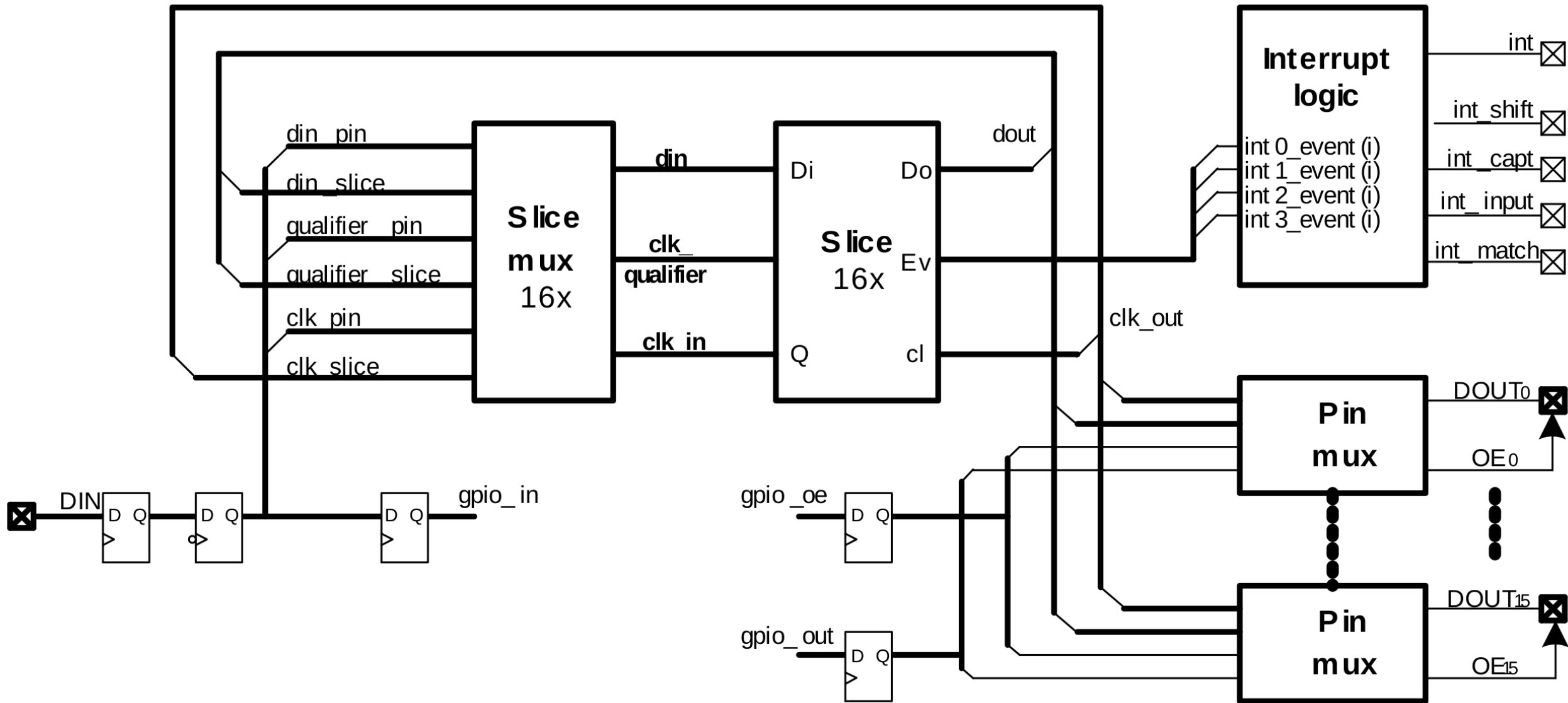
[Development Boards](#)



OKAY, SHOW ME THE
SOFTWARE

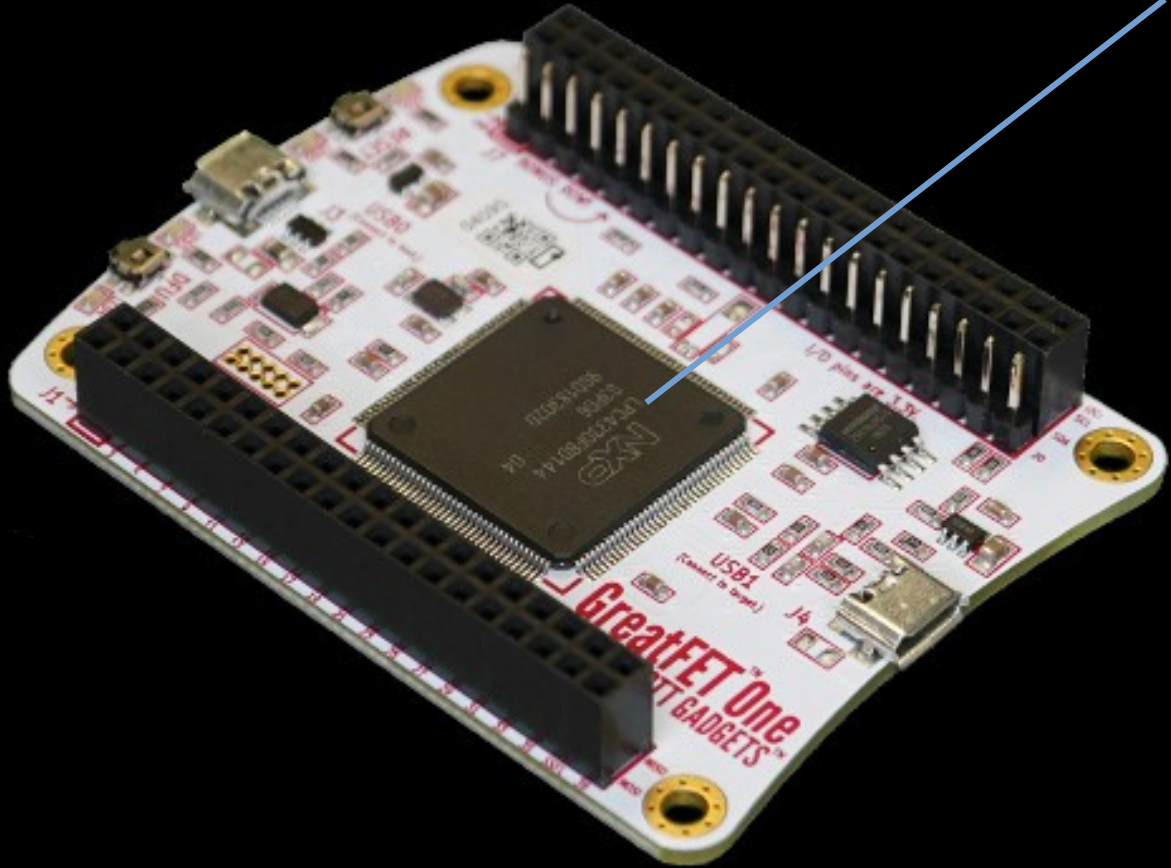
ULPI_DATA





LPC43XX

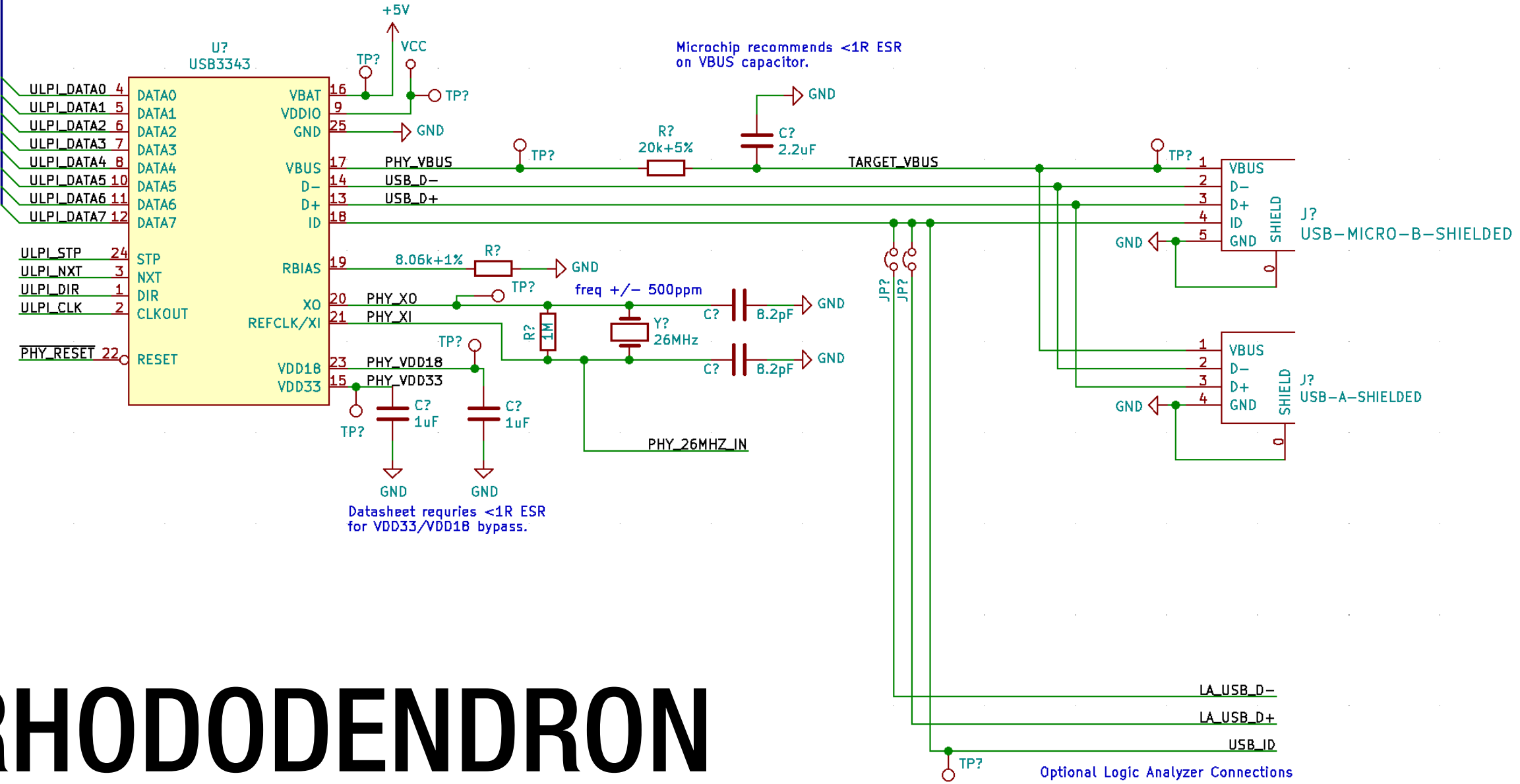
Fig 36. SGPIO block diagram



GreatFET One (codename Azalea)

- Multi-tool for hardware hacking – including lots of USB functionality.
- Centered around the LPC4330, so we have access to that 204MHz **SGPIO** fanciness.
- **Super-fast USB stack**, so we can saturate the host's USB bus with captured data.
- Build them yourselves! Design files at <https://github.com/greatfet-hardware>

ULPI_DATA



RHODODENDRON

RECIPE FOR A HIGH SPEED ANALYZER

Components:

- LPC43xx; or similar
- SDRAM for packet buffering*
- ULPI PHY
- SPI Flash

What this will get you:

- a super-cheap analyzer, especially if you omit the SDRAM
 - an SDRAM-less design can still capture a lot of stuff

RECIPE FOR A HIGH SPEED ANALYZER

Components:

- LPC43xx; or similar
- SDRAM for packet buffering*
- ULPI PHY
- SPI Flash

What this won't get you:

- working analysis software
 - *wait*

commit c4f0b67bed514349a4a78c3c8c2bee4e6f3b9daa

Author: Mikaela Szekely <qyriad@gmail.com>

Date: Wed Jun 19 20:01:54 2019 -0600

usbproxy backend: always convert to `bytes`

Data passed to our FaceDancer filter varies; it's better to make sure we know what we're working with.

commit 09889f5e75ae208e00d7664a4e468f17b3b93be2

Author: Mikaela Szekely <qyriad@gmail.com>

Date: Wed Jun 19 14:36:58 2019 -0600

usbproxy backend: fix _get_device_address

commit 5c5fdfa7bed7e828a99aaba584e8f46b1fdea2a5

Author: Kate J. Temkin <k@ktemkin.com>

Date: Tue Jun 18 01:07:14 2019 -0600

implement most of the basic packet types, and the USBProxy backend

commit 4c7cf111b4a875ea51463eedc095a8907c007128

Author: Kate J. Temkin <k@ktemkin.com>

Date: Mon Jun 17 06:44:30 2019 -0600

initial commit: teensy start at our WIP analyzer solution

commit c4f0b67bed514349a4a78c3c8c2bee4e6f3b9daa

Author: Mikaela Szekely <qyriad@gmail.com>

Date: Wed Jun 19 20:01:54 2019 -0600

usbproxy backend: always convert to `bytes`

Data passed to our FaceDancer filter varies; it's better to make sure we know what we're working with.

commit 09889f5e75ae208e00d7664a4e468f17b3b93be2

Author: Mikaela Szekely <qyriad@gmail.com>

Date: Wed Jun 19 14:36:58 2019 -0600

usbproxy backend: fix _get_device_address

commit 5c5fdfa7bed7e828a99aaba584e8f46b1fdea2a5

Author: Kate J. Temkin <k@ktemkin.com>

Date: Tue Jun 18 01:07:14 2019 -0600

implement most of the basic packet types, and the USBProxy backend

commit 4c7cf111b4a875ea51463eedc095a8907c007128

Author: Kate J. Temkin <k@ktemkin.com>

Date: Mon Jun 17 06:44:30 2019 -0600

initial commit: teensy start at our WIP analyzer solution



Backend

Decoders

Frontend

ViewSB Architecture

- Three component types run in their own processes; so they can **start** and **stop** independently.
- All three components are designed to be easily swappable using **simple, modular python**.
- All completely open-source, and contributor-friendly!
<https://github.com/usb-tools/viewsb>

standalone python host-side module to talk to OpenVizsla devices; based on LibOV

Edit

Manage topics

7 commits 1 branch 0 releases 1 contributor BSD-2-Clause

Branch: master

New pull request

Create new file

Upload files

Find File

Clone or download



ktemkin python: device: fix type confusion re: firmware package names

Latest commit 4360e67 3 days ago



libov

initial commit: basic implementation of python-package'd OV

5 days ago



openvizsla

python: device: fix type confusion re: firmware package names

3 days ago



.gitignore

initial commit: basic implementation of python-package'd OV

5 days ago



LICENSE

initial commit: basic implementation of python-package'd OV

5 days ago



MANIFEST.in

initial commit: basic implementation of python-package'd OV

5 days ago



setup.py

cleanup: split the libOV monolith and clean up our structure

4 days ago

Help people interested in this repository understand your project by adding a README.

Add a README

OKAY, SHOW ME THE
SOFTWARE

Backend

Decoders

Frontend

```
def handle_usb_packet(self, timestamp, raw_packet, flags):  
    """ Called whenever the OpenVizsla device detects a new USB packet. """  
  
    # For now, ignore any populated USB packets as noise.  
    if not len(raw_packet):  
        return  
  
    # TODO: convert flags to status?  
    packet = USBPacket.from_raw_packet(raw_packet, timestamp=timestamp)  
  
    # Assume the packet isn't one we're suppressing, emit it to our stack.  
    if not self._should_be_suppressed(packet):  
        self._emit_packet(packet)
```

`[capture-cli-ov.py]`

Backend

Decoders

Frontend

```
class SetAddressRequest(StandardControlRequest):  
    ···· REQUEST_NUMBER = 5  
    ···· REQUEST_NAME = "SET ADDESS"  
    ···· FIELDS = { "new_address" }  
  
    ···· def validate(self):  
    ···· |···· self.new_address = self.value  
  
    ···· def summarize(self):  
    ···· |···· return "requesting device use address {}".format(self.new_address)
```

`[capture-cli.py]`



Backend

Decoders

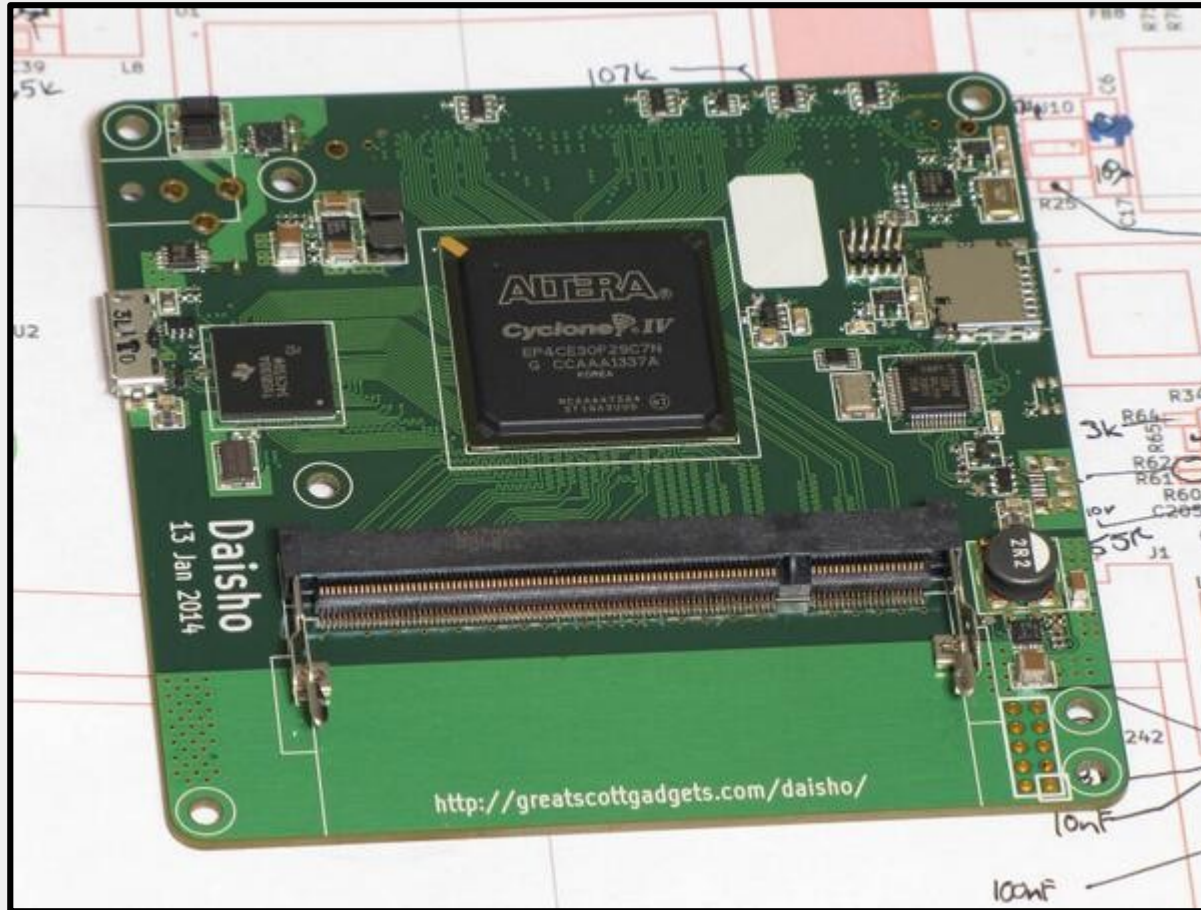
Frontend


```
class CLIFrontend(ViewSBFFrontend):  
    """ Simplest possible frontend: print our packets. """  
  
    def __init__(self):  
        """ Creates a new CLI display frontend. """  
        pass  
  
    def handle_incoming_packet(self, packet):  
        """ Render any incoming packets to our UI. """  
        # just print; no fancy frontend  
        print(repr(packet))
```

[capture-tui.py]

[oh, and we support `usbmon`]

**BONUS
TIME**



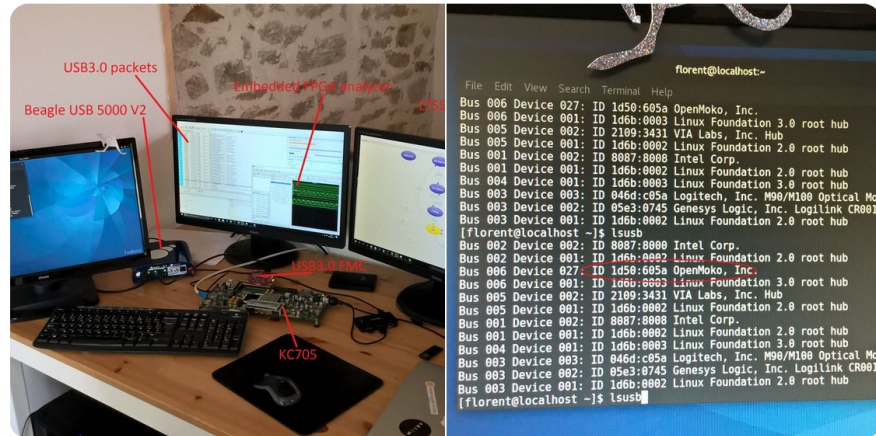
DAISHO

us, able to see rx data :)	2 years ago
tion	2 years ago
add precision for kc705	2 years ago

Enjoy Digital @enjoy_digital

Follow

Just finished porting Daisho's open source USB3 core to Xilinx FPGAs :) github.com/enjoy-digital/ ...



7:43 AM - 3 Apr 2017

- load.py add kc705 base design
- nexys_video.py replace litex.gen imports with migen

README.md

Test of the USB3 IP Core from Daisho on a Xilinx device

In this repository we are testing the USB3 IP Core from

- USB2 / ULPI working :) (vendor agnostic)



Luke Valenty @TinyFPGA

Follow

Looks like the Daisho USB3 core easily fits in the ECP5 85K part. It uses about 10% of the logic resources. It also meets timing with plenty of margin. These are all good signs. Next I need to wrap up the 5G SERDES in a PIPE interface.

BONUS
QUESTIONS?