Introduction to USB hacking

Andrey Konovalov <andreyknvl@gmail.com>

PHDays 2018, Hardware Village
May 15th 2018

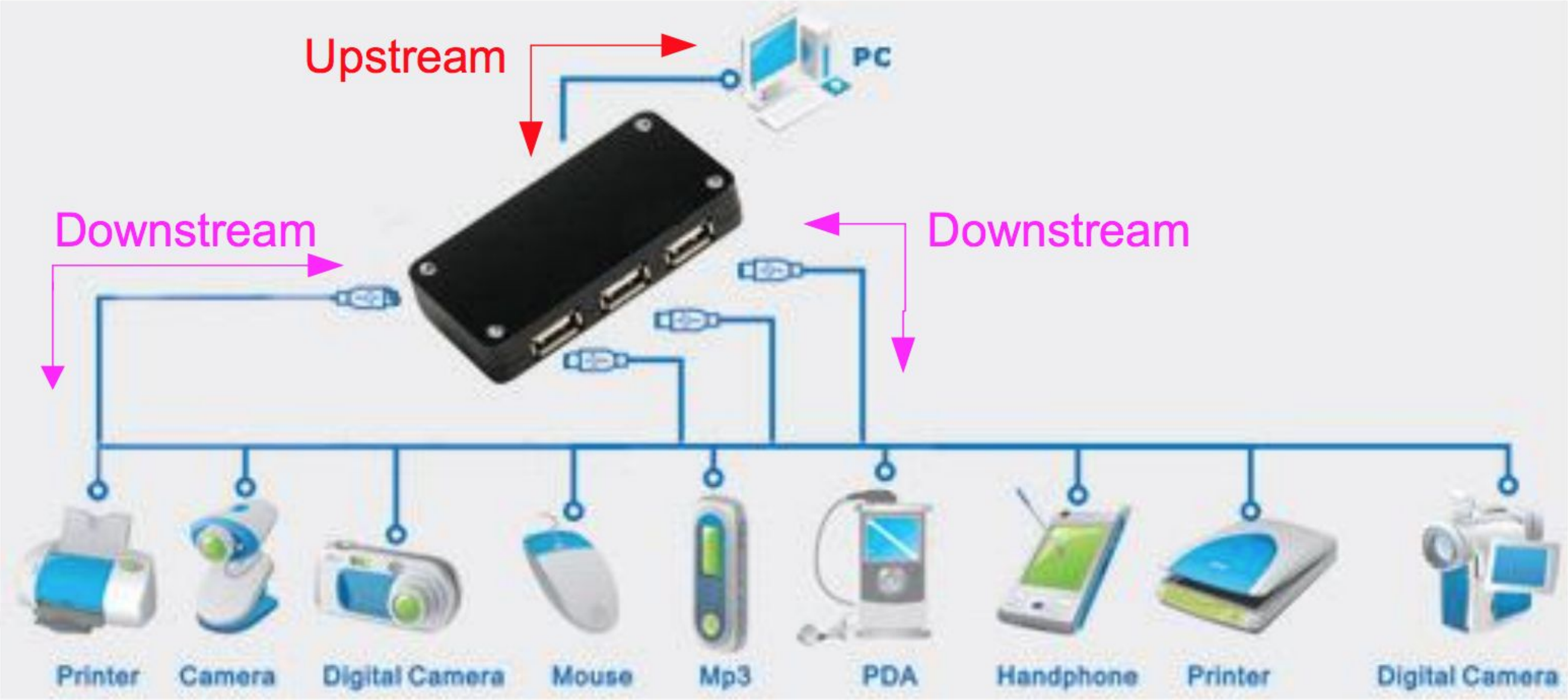
Agenda

- Part 1: USB 101
- Part 2: USB attack surface
- Part 3: Consumer ready BadUSB
- Part 4: Microcontroller based BadUSB
- Part 5: Linux gadget subsystem based
- Part 6: Facedancer21
- Part 7: USB fuzzing
- Part 8: USB sniffing

(USB 2.0 only, Linux only)

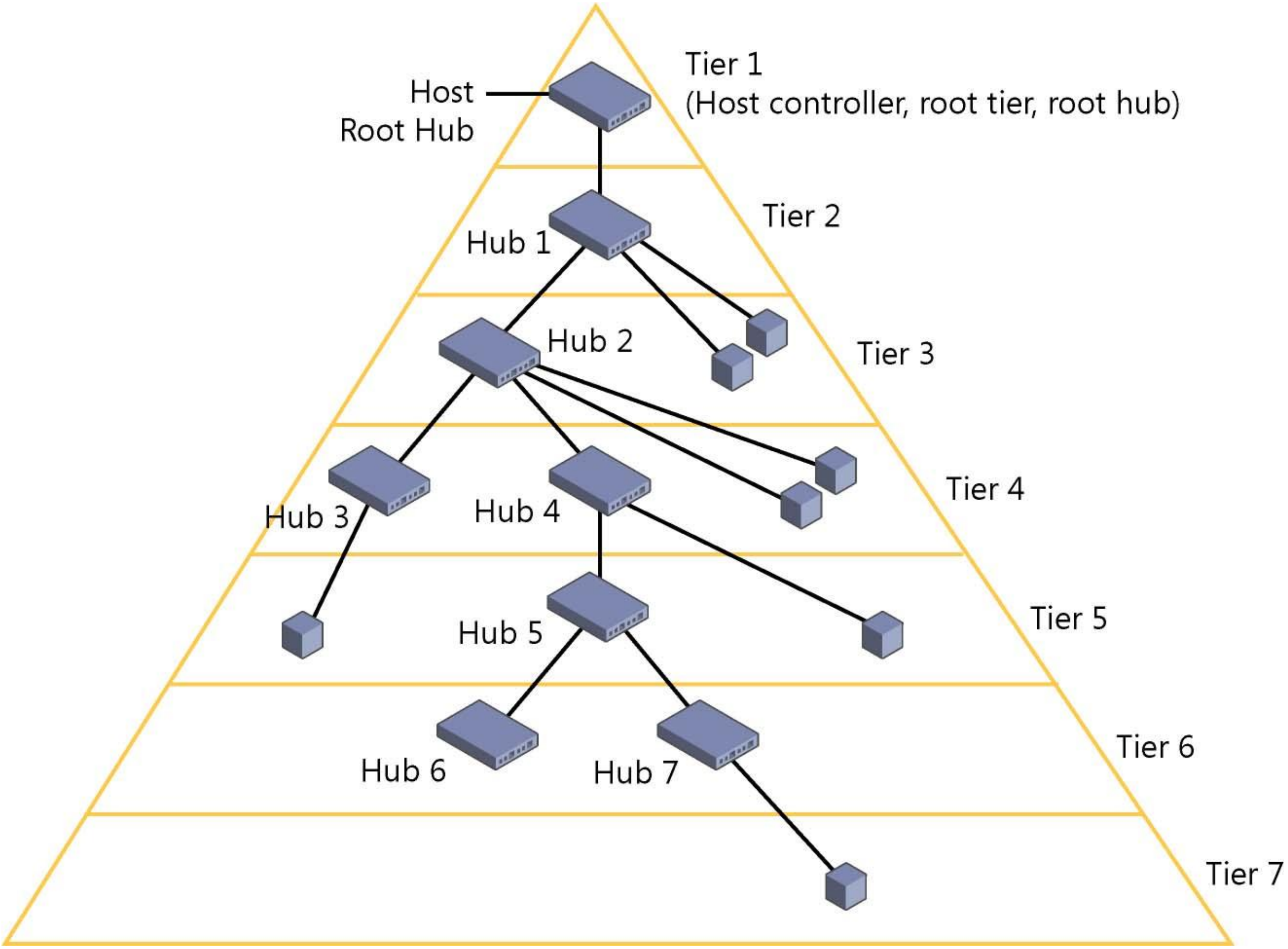
Part 1: USB 101

USB topology



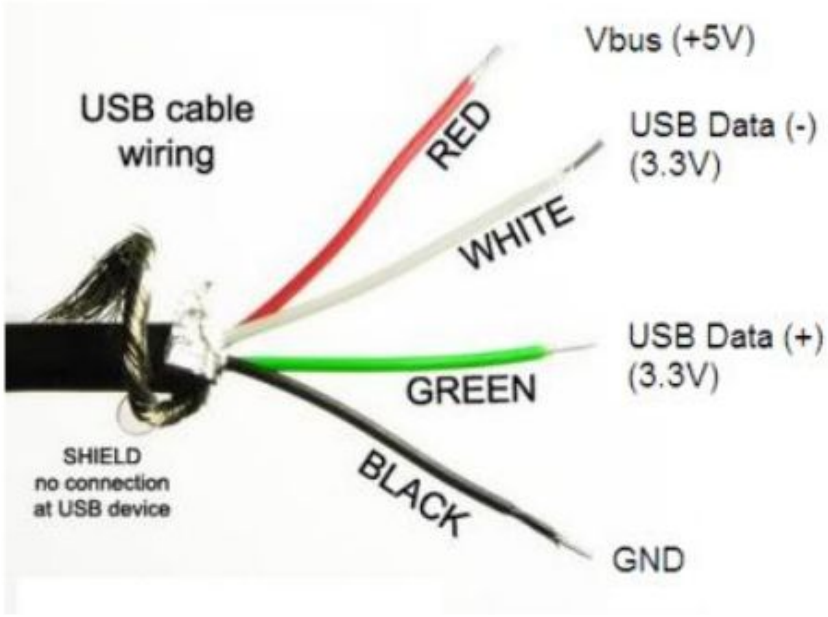
"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

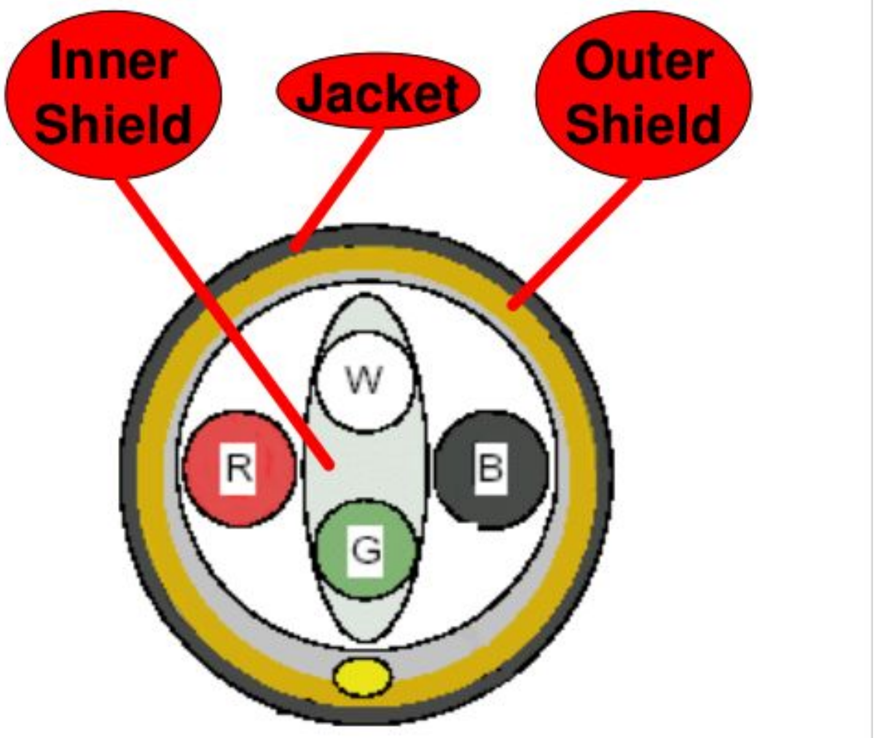
USB hubs

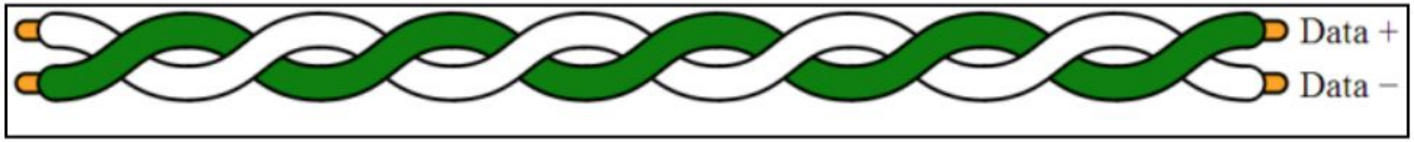


"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

USB cable

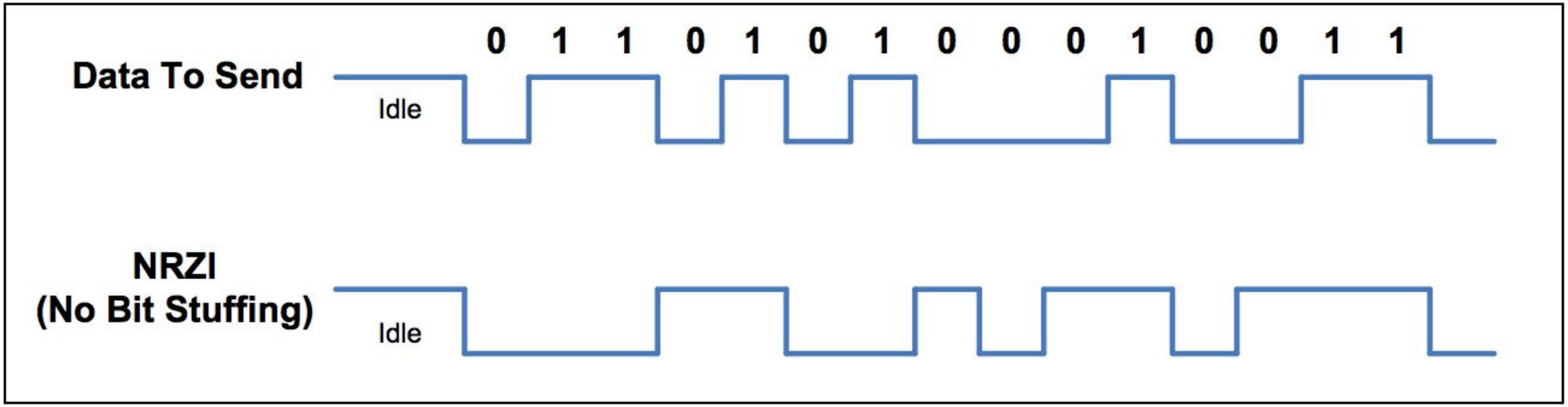


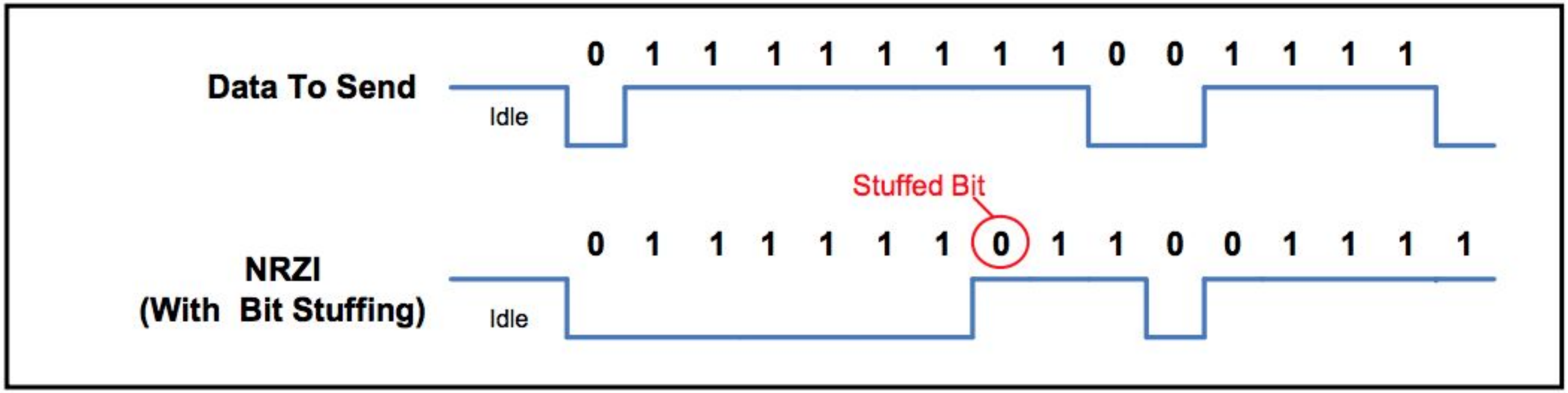




"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

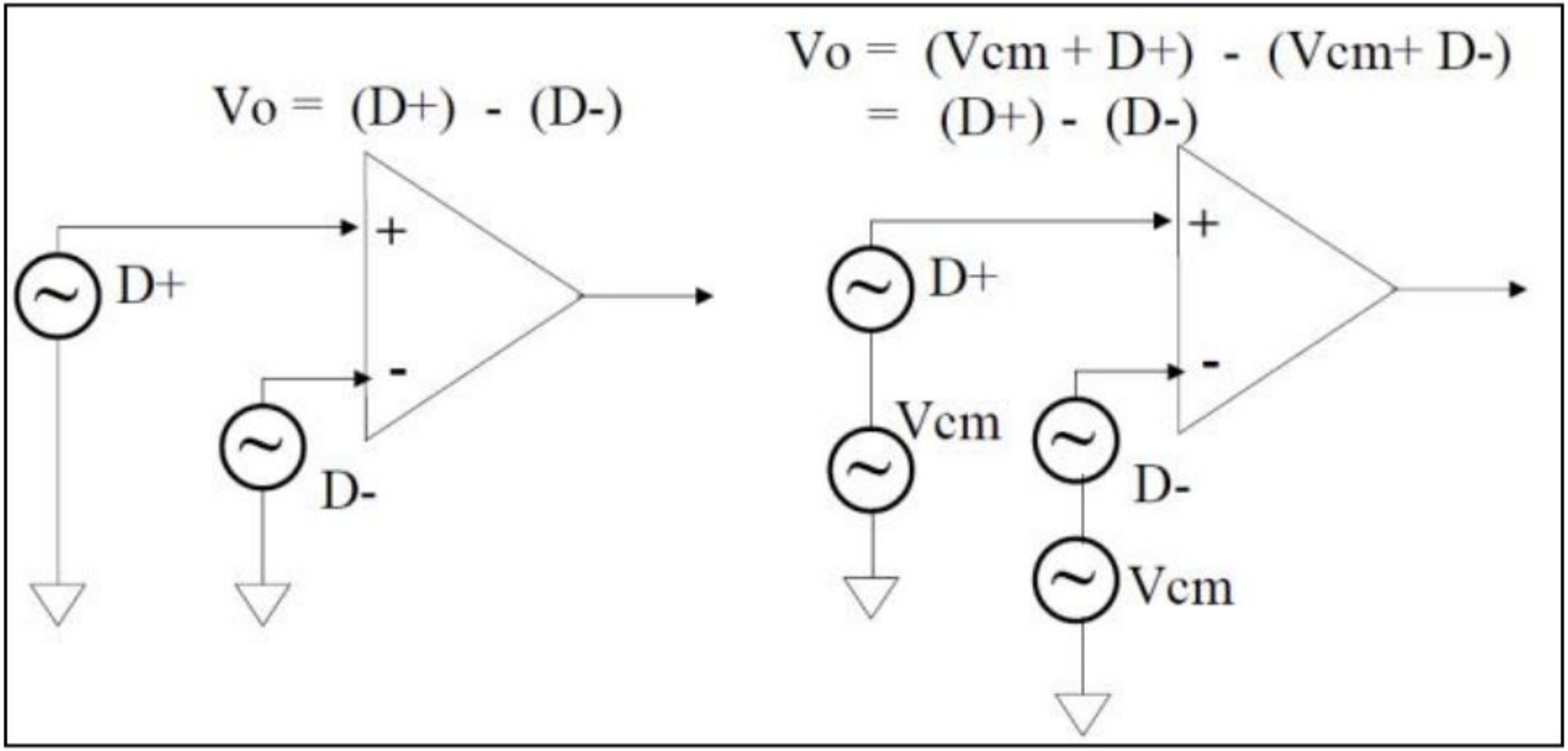
NRZI encoding





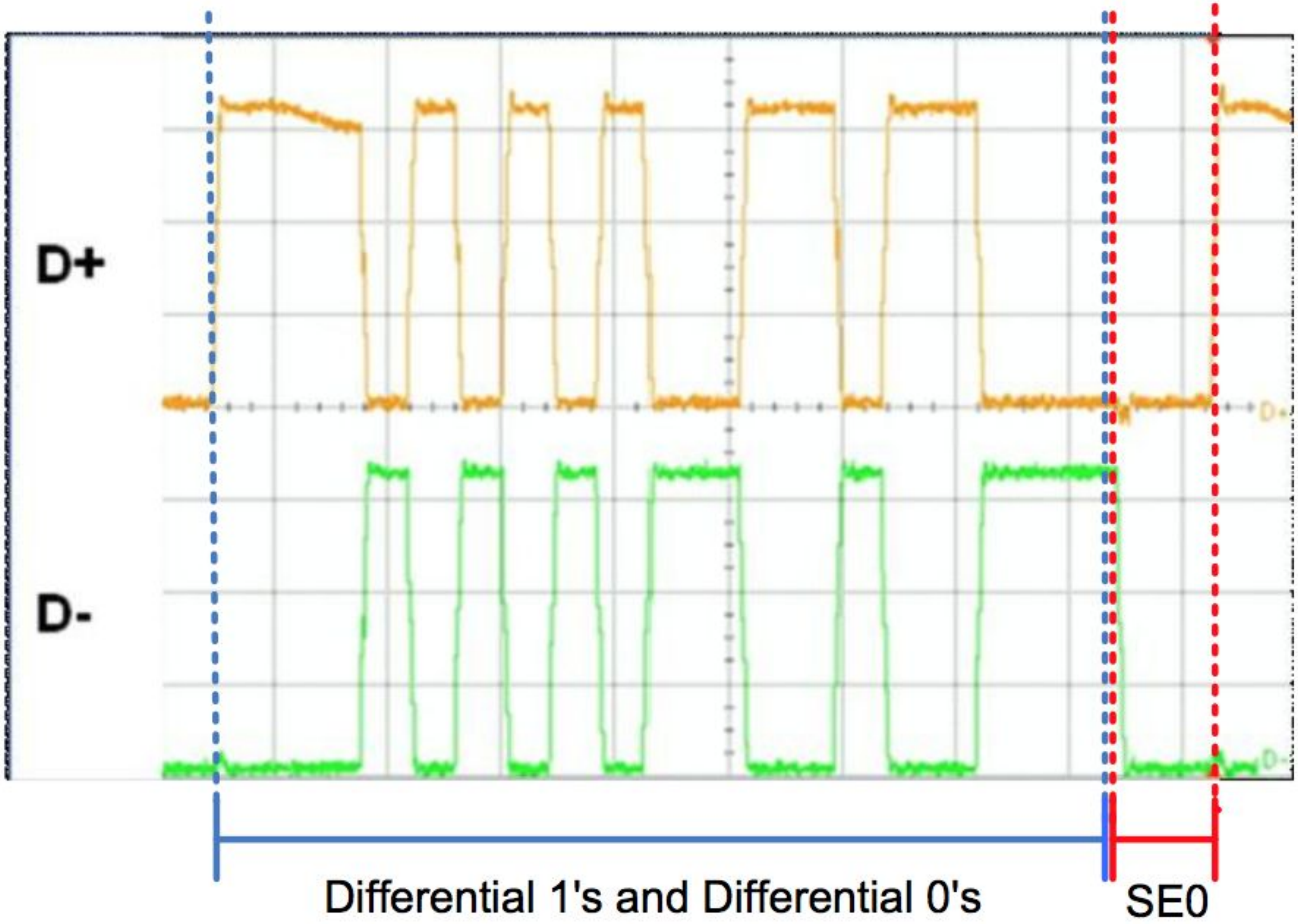
"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

USB differential amplifier



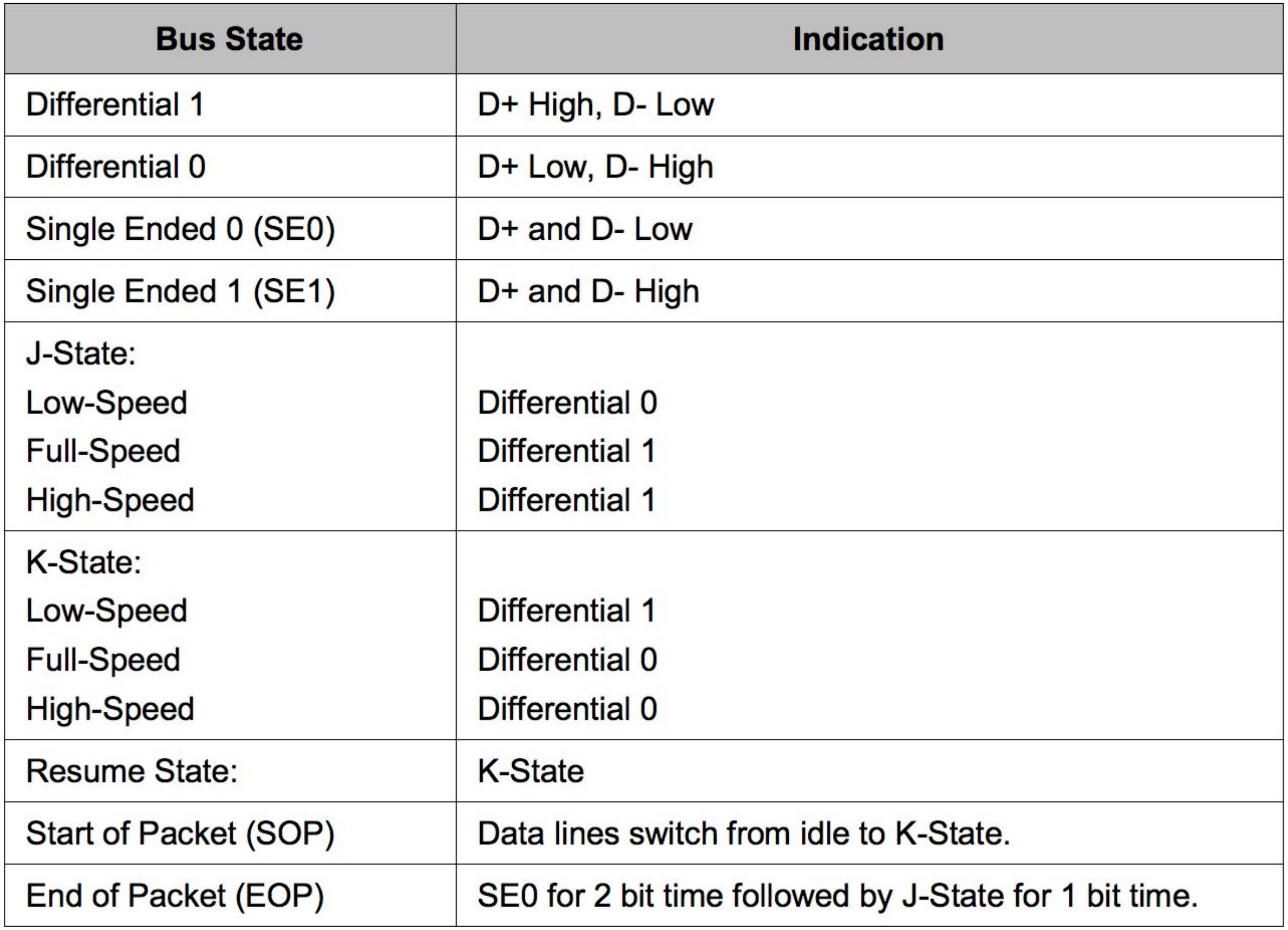
"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

USB D+ and D- communication



"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

USB communication states

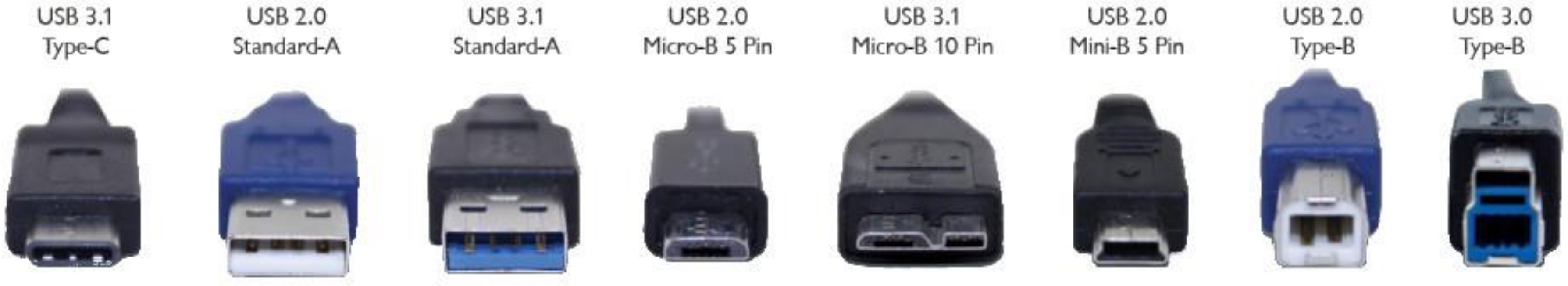


Bus State	Indication
Differential 1	D+ High, D- Low
Differential 0	D+ Low, D- High
Single Ended 0 (SE0)	D+ and D- Low
Single Ended 1 (SE1)	D+ and D- High
J-State: Low-Speed Full-Speed High-Speed	 Differential 0 Differential 1 Differential 1
K-State: Low-Speed Full-Speed High-Speed	 Differential 1 Differential 0 Differential 0
Resume State:	K-State
Start of Packet (SOP)	Data lines switch from idle to K-State.
End of Packet (EOP)	SE0 for 2 bit time followed by J-State for 1 bit time.

"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

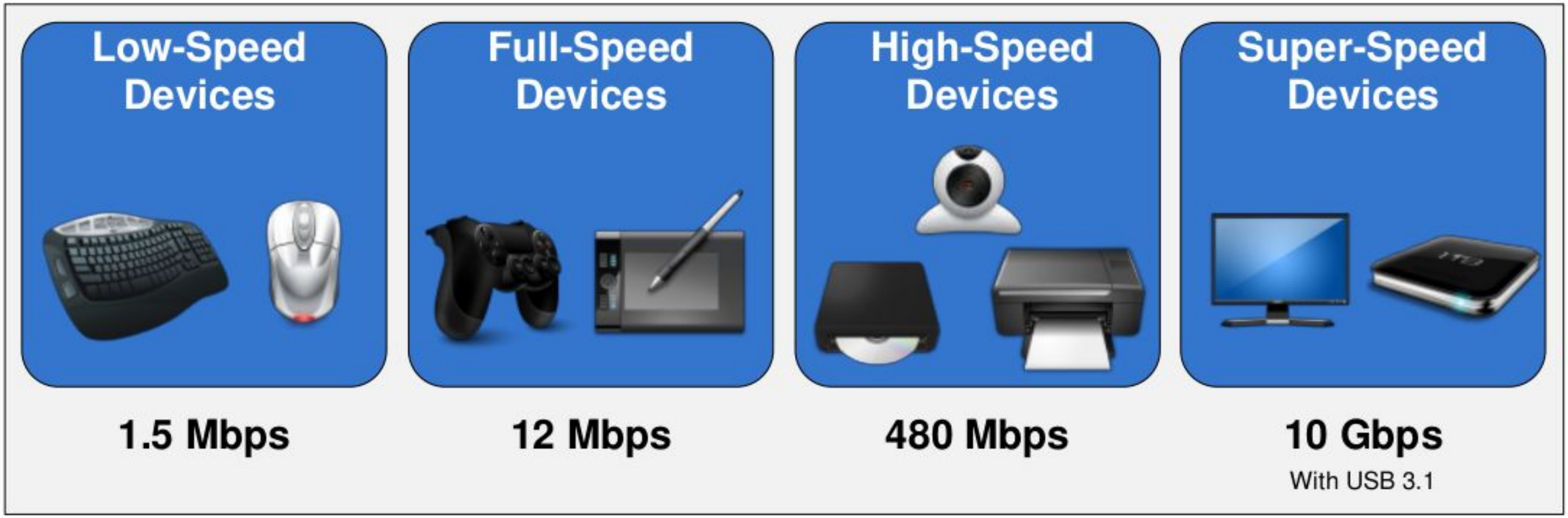
Demo: sniffing USB with a Logic Analyzer

USB connectors



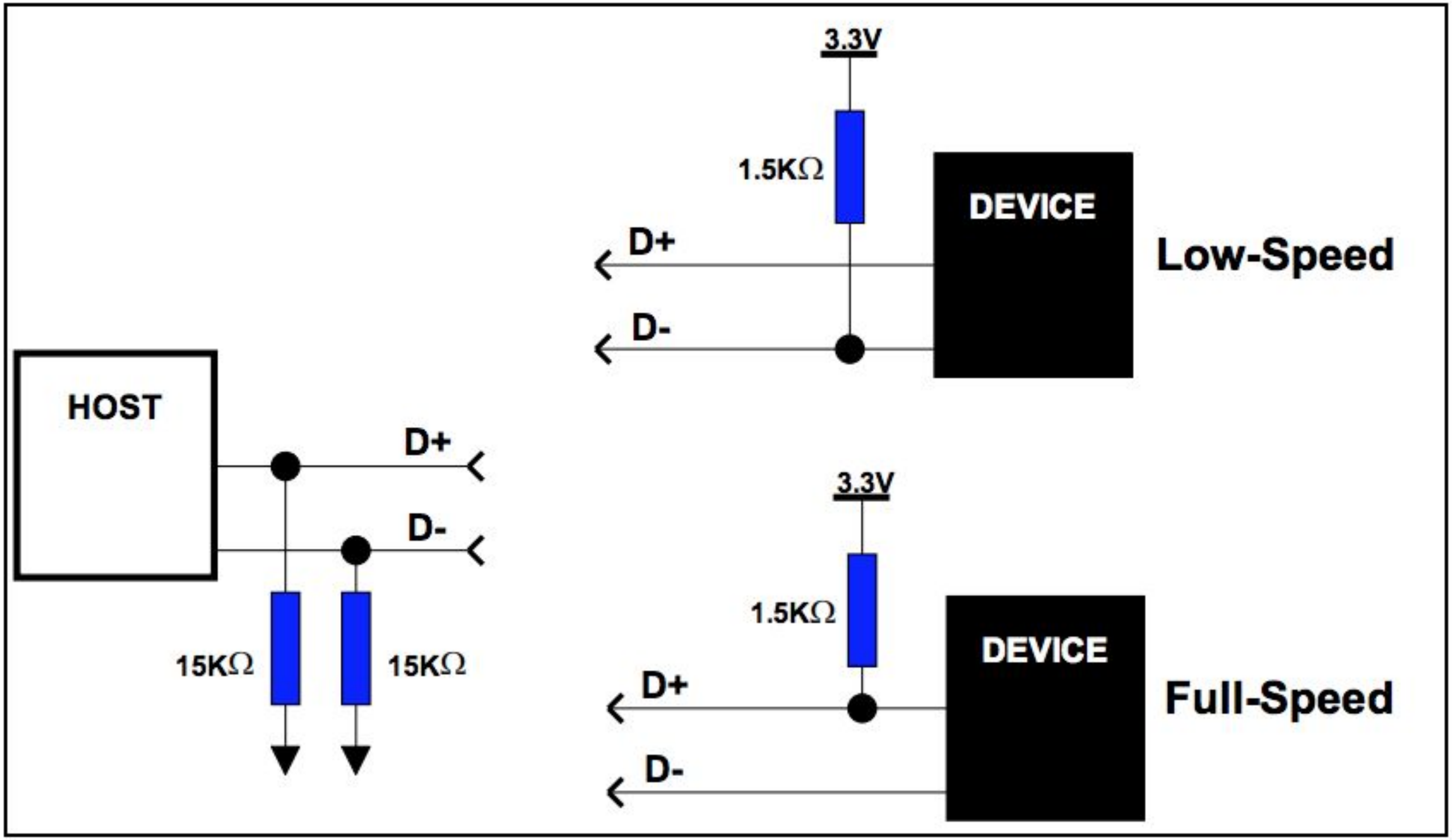
https://stroom.live/blog/index.php/tag/micro-usb/

USB transfer speeds



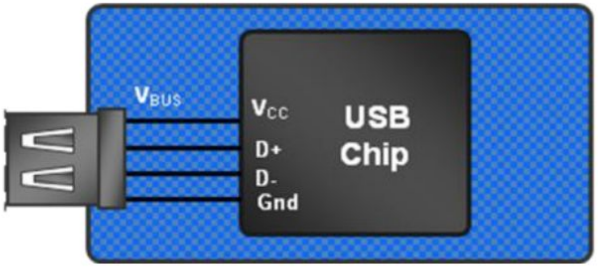
"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

USB speed detection

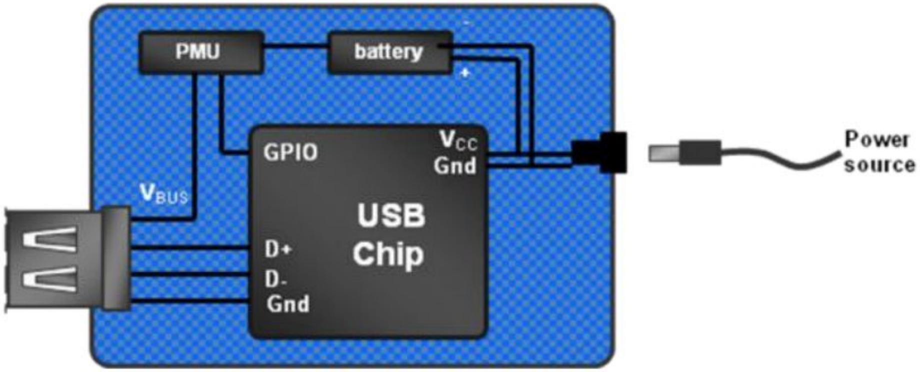


"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

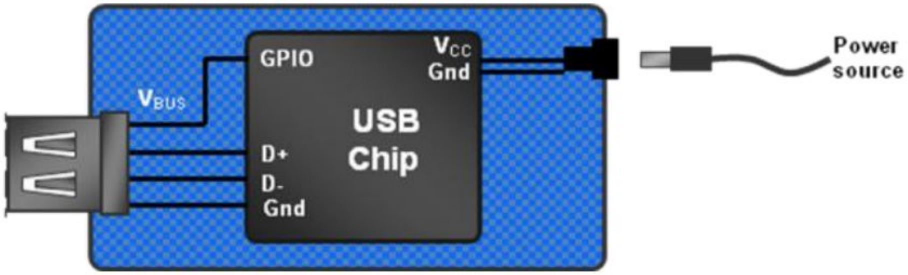
USB power



bus-powered



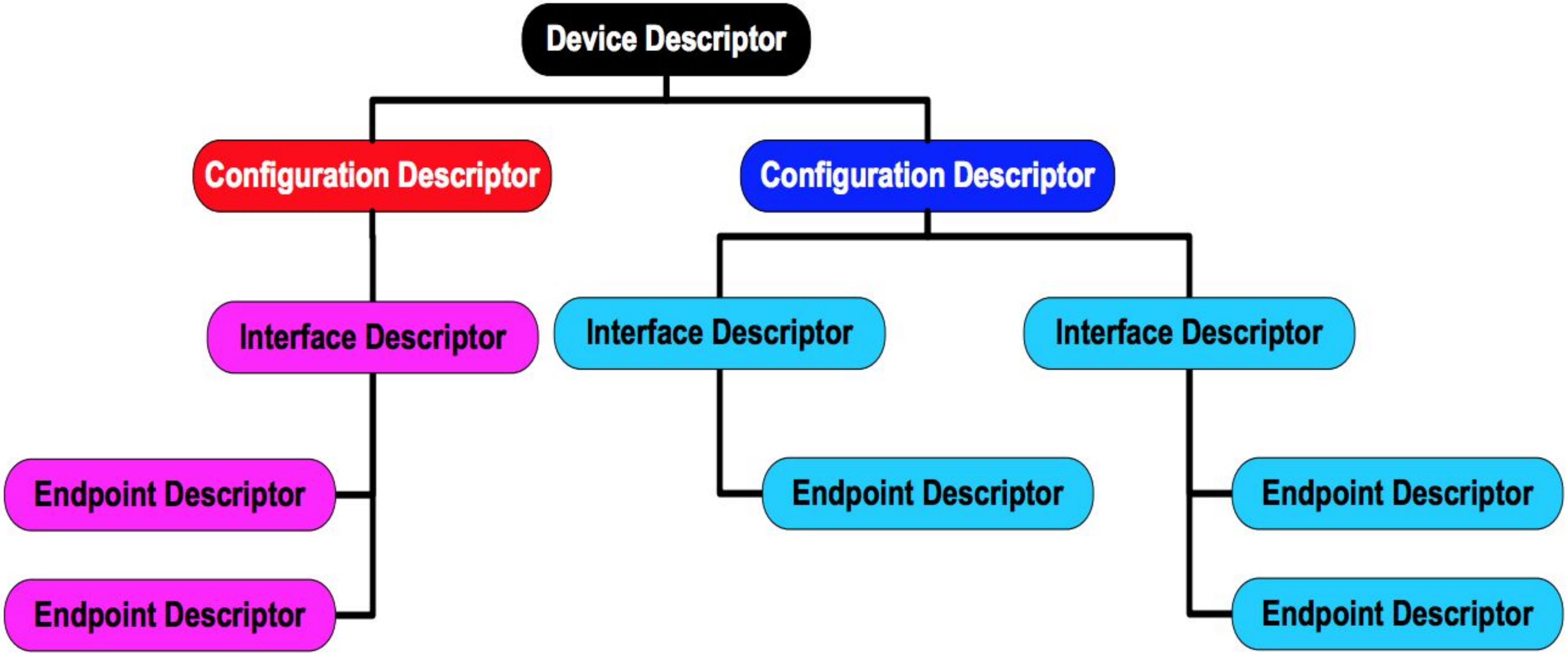
hybrid powered



self-powered

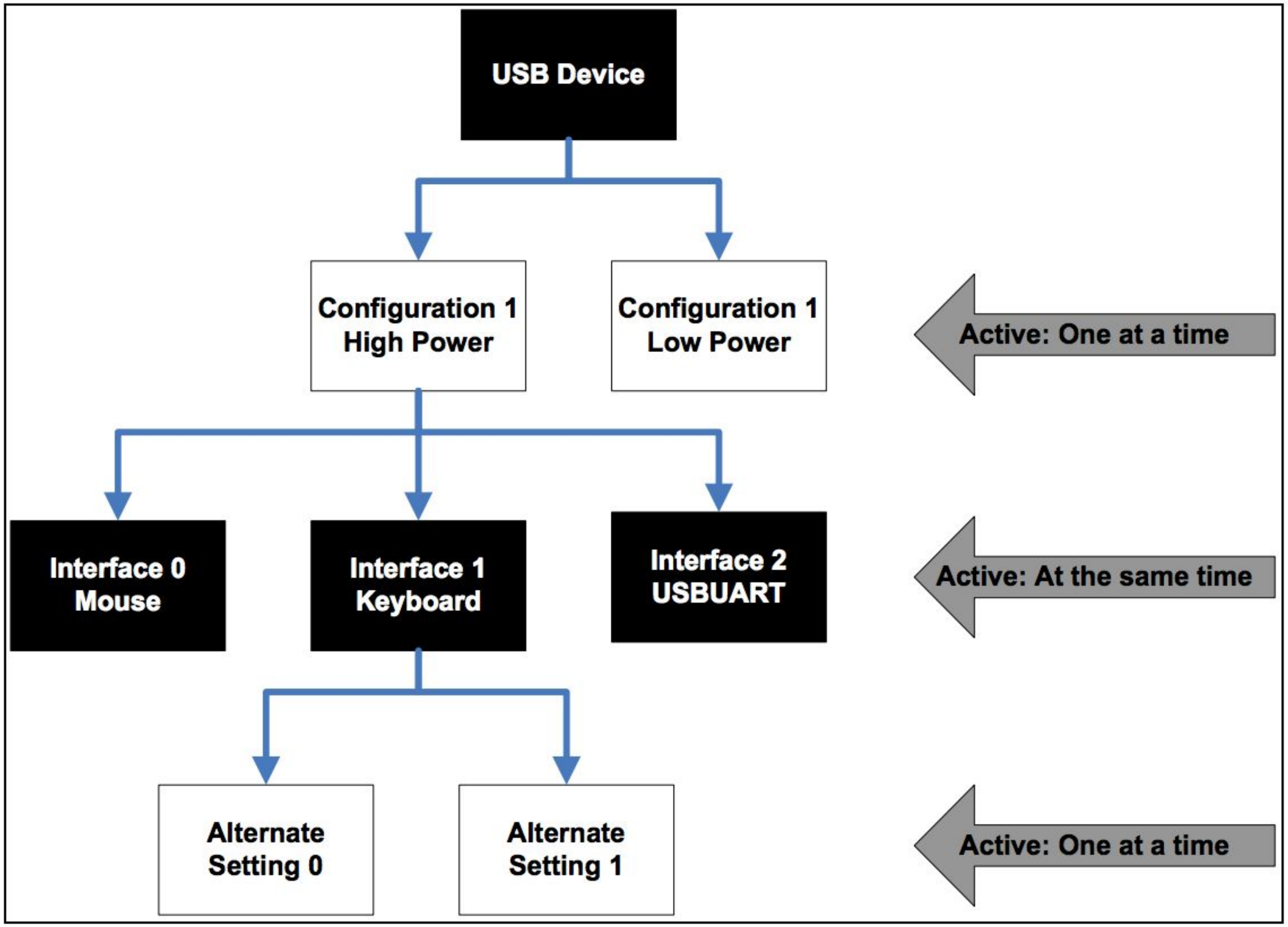
"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

USB device descriptor



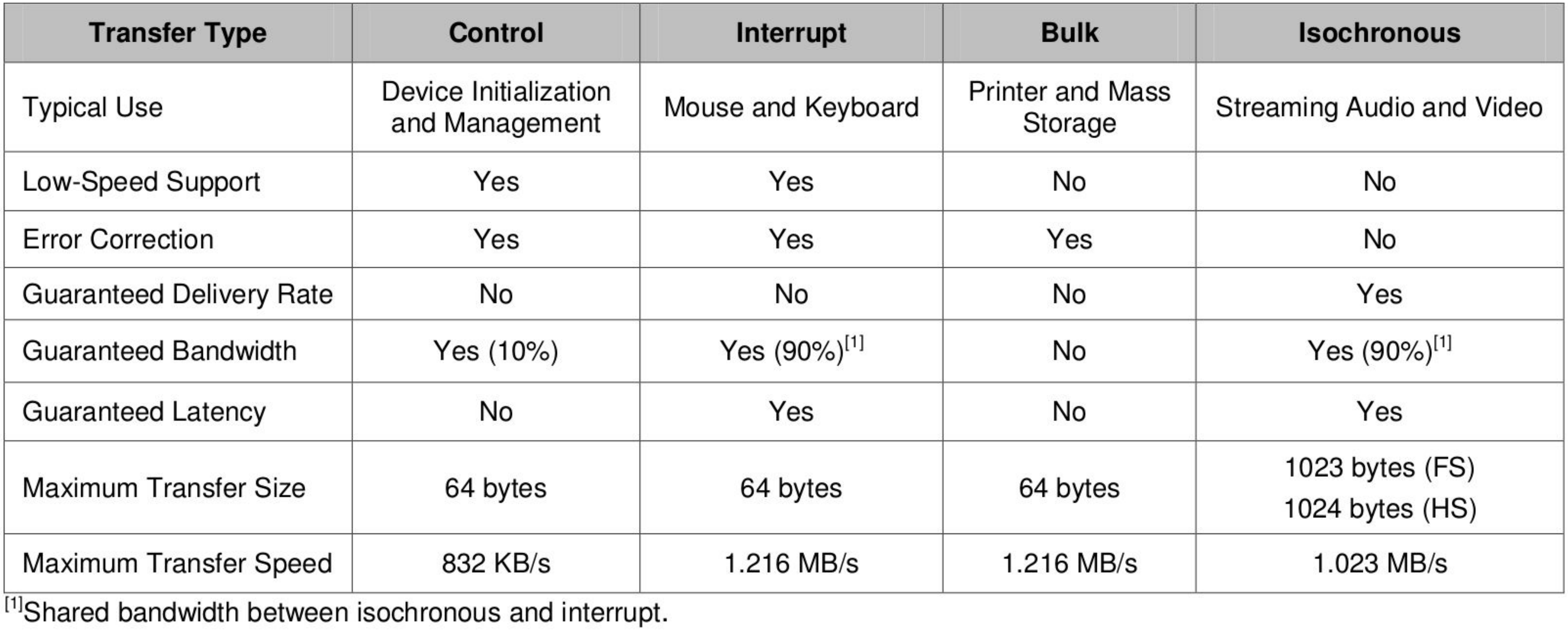
"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

USB device descriptor: example



"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

USB endpoint types

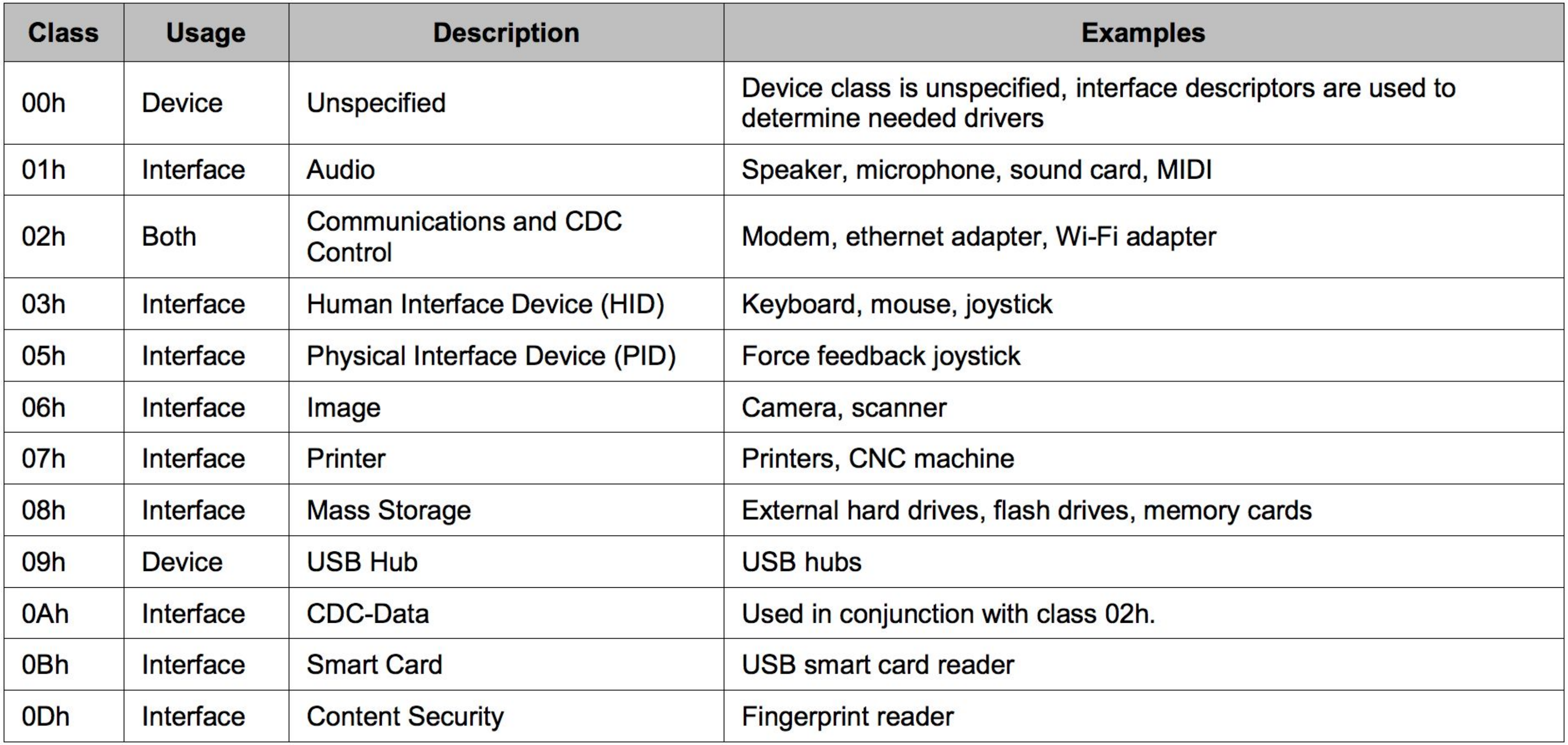


Transfer Type	Control	Interrupt	Bulk	Isochronous
Typical Use	Device Initialization and Management	Mouse and Keyboard	Printer and Mass Storage	Streaming Audio and Video
Low-Speed Support	Yes	Yes	No	No
Error Correction	Yes	Yes	Yes	No
Guaranteed Delivery Rate	No	No	No	Yes
Guaranteed Bandwidth	Yes (10%)	Yes (90%)[1]	No	Yes (90%)[1]
Guaranteed Latency	No	Yes	No	Yes
Maximum Transfer Size	64 bytes	64 bytes	64 bytes	1023 bytes (FS) 1024 bytes (HS)
Maximum Transfer Speed	832 KB/s	1.216 MB/s	1.216 MB/s	1.023 MB/s

[1] Shared bandwidth between isochronous and interrupt.

"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

USB class codes



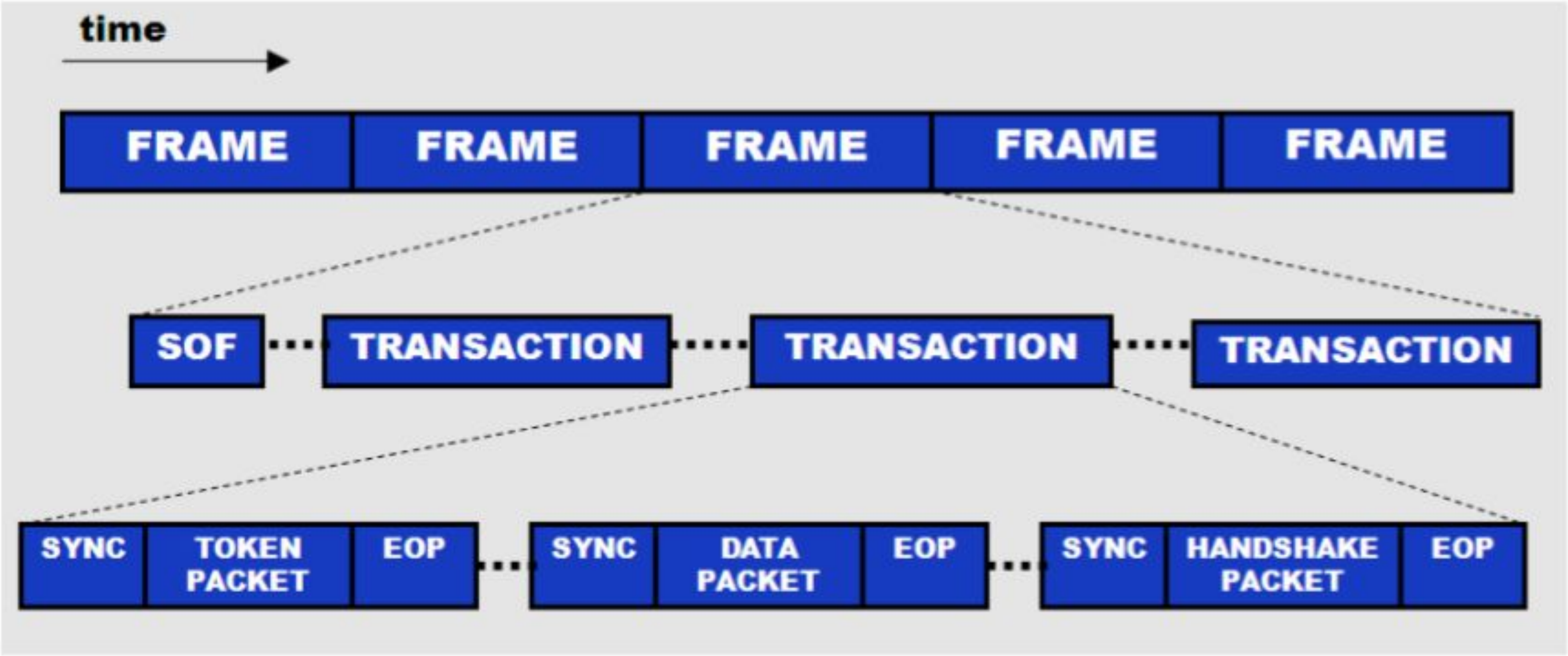
Class	Usage	Description	Examples
00h	Device	Unspecified	Device class is unspecified, interface descriptors are used to determine needed drivers
01h	Interface	Audio	Speaker, microphone, sound card, MIDI
02h	Both	Communications and CDC Control	Modem, ethernet adapter, Wi-Fi adapter
03h	Interface	Human Interface Device (HID)	Keyboard, mouse, joystick
05h	Interface	Physical Interface Device (PID)	Force feedback joystick
06h	Interface	Image	Camera, scanner
07h	Interface	Printer	Printers, CNC machine
08h	Interface	Mass Storage	External hard drives, flash drives, memory cards
09h	Device	USB Hub	USB hubs
0Ah	Interface	CDC-Data	Used in conjunction with class 02h.
0Bh	Interface	Smart Card	USB smart card reader
0Dh	Interface	Content Security	Fingerprint reader

and even more ...

"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

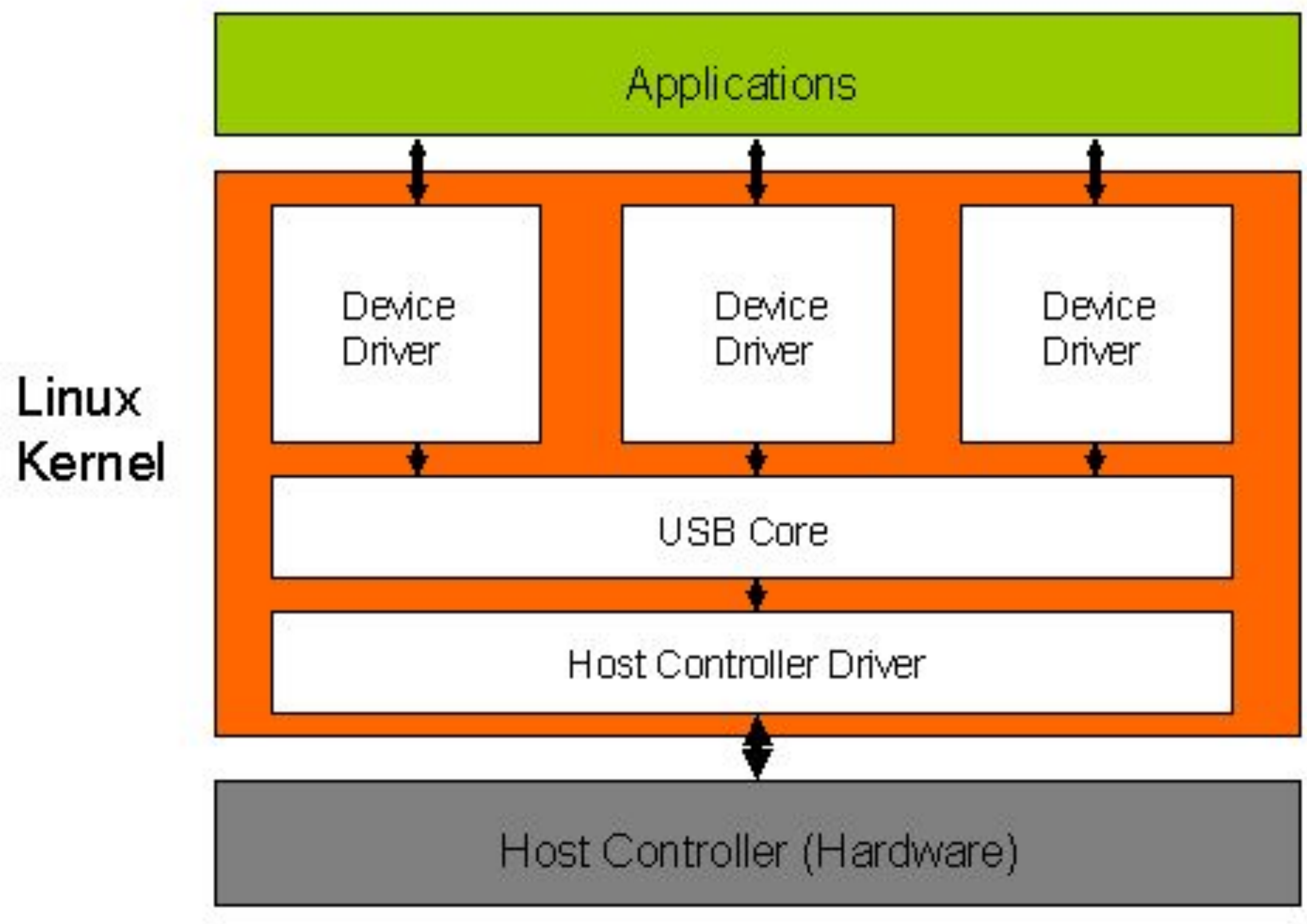
Demo: lsusb and syslog

USB communication



"USB 101: An Introduction to Universal Serial Bus 2.0" by Robert Murphy

USB host



http://www.linux-usb.org/USBMon/dissertation/USB-dissertation.htm

USB enumeration (simplified)

1. The device is plugged into a USB port
2. The root hub detects the device (by monitoring voltages on the ports)
3. The host detects that the device is plugged in
4. The host resets the devices
5. The host assigns address to the device
6. The host requests device descriptors
7. The host loads the appropriate driver
8. The host sets a specific device configuration
9. Done

Demo: sniffing USB with usbmon and wireshark

Demo: turning off LED
on a Logitech web camera

Part 2: USB attack surface

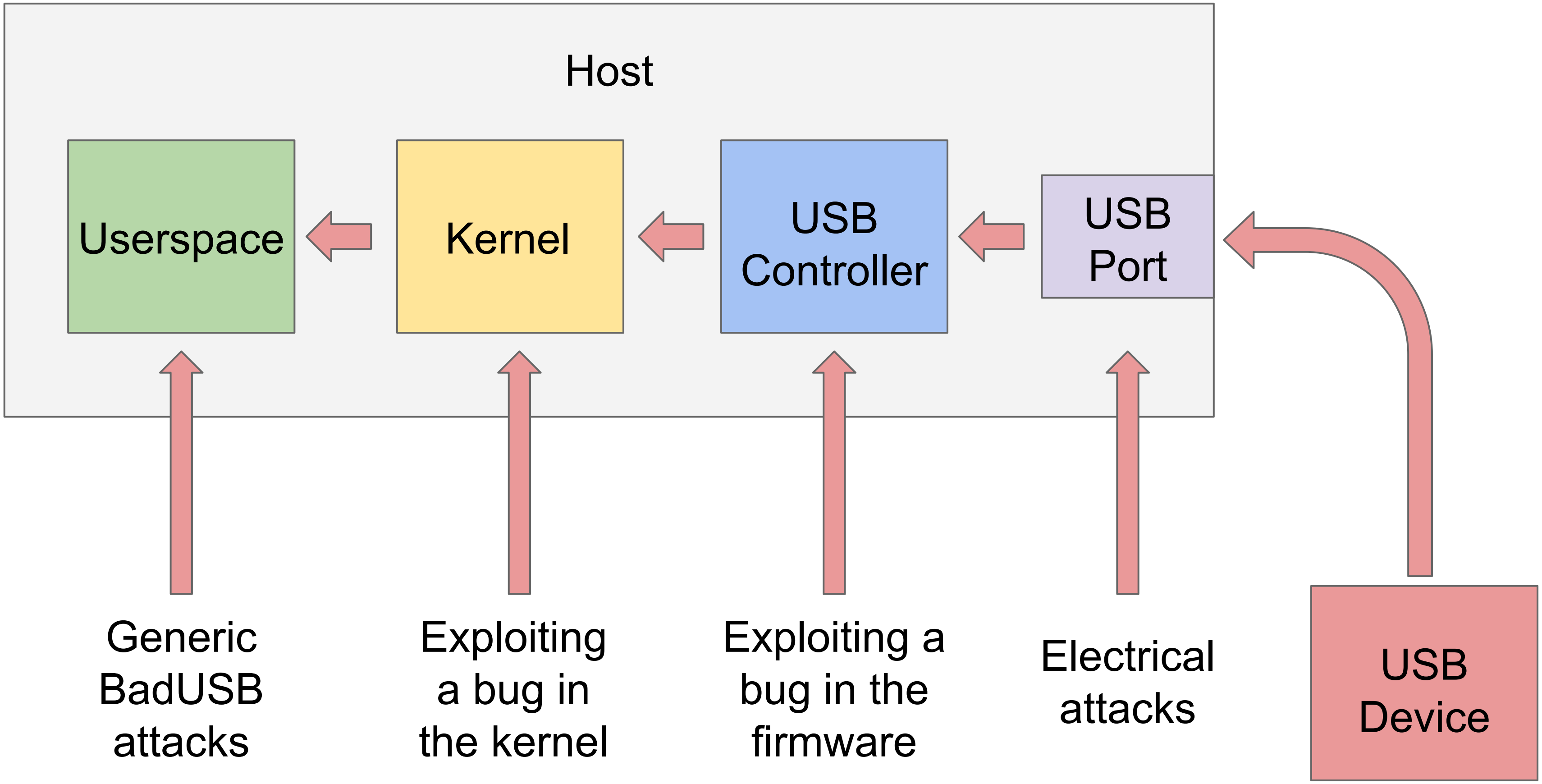
Host ⇔ Device

- The goal is to attack the USB host

- A couple of ways to do that:
  1. Insert a crafted malicious USB device
  2. Attack an already inserted USB device from the host side, hijack control, and turn the device malicious

Device => Host

- Attack surface scheme:



Host

Userspace

Kernel

USB Controller

USB Port

USB Device

Generic BadUSB attacks

Exploiting a bug in the kernel

Exploiting a bug in the firmware

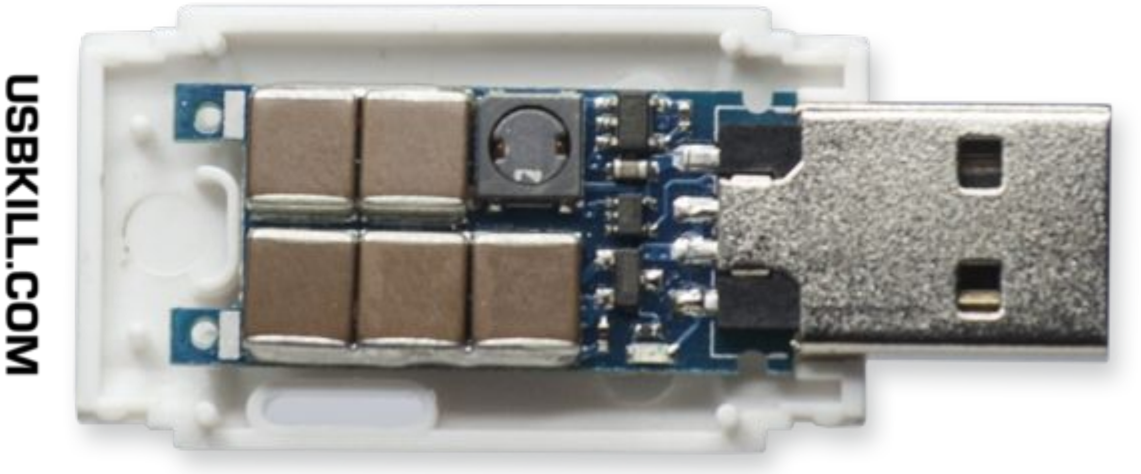
Electrical attacks

BadUSB

- The original talk by Karsten Nohl and Jakob Lell
- Today the term BadUSB is used to refer to any kind of a malicious USB device
- Typical examples:
    - BadUSB keyboard that looks like flash drive
    - BadUSB ethernet adapter that looks like flash drive
- Less typical: BadUSB that exploits a bug in the OS kernel
- Lots of consumer BadUSB devices, lots of way to make your own

USB Killer

- "When plugged into a device, the USB Killer rapidly charges its capacitors from the USB power lines. When the device is charged, -200VDC is discharged over the data lines of the host device. This charge/discharge cycle is repeated many times per second, until the USB Killer is removed."
- Available for 65$ at https://usbkill.com/products/usb-killer-v3



https://www.computerworld.com.sg/print-article/102264/

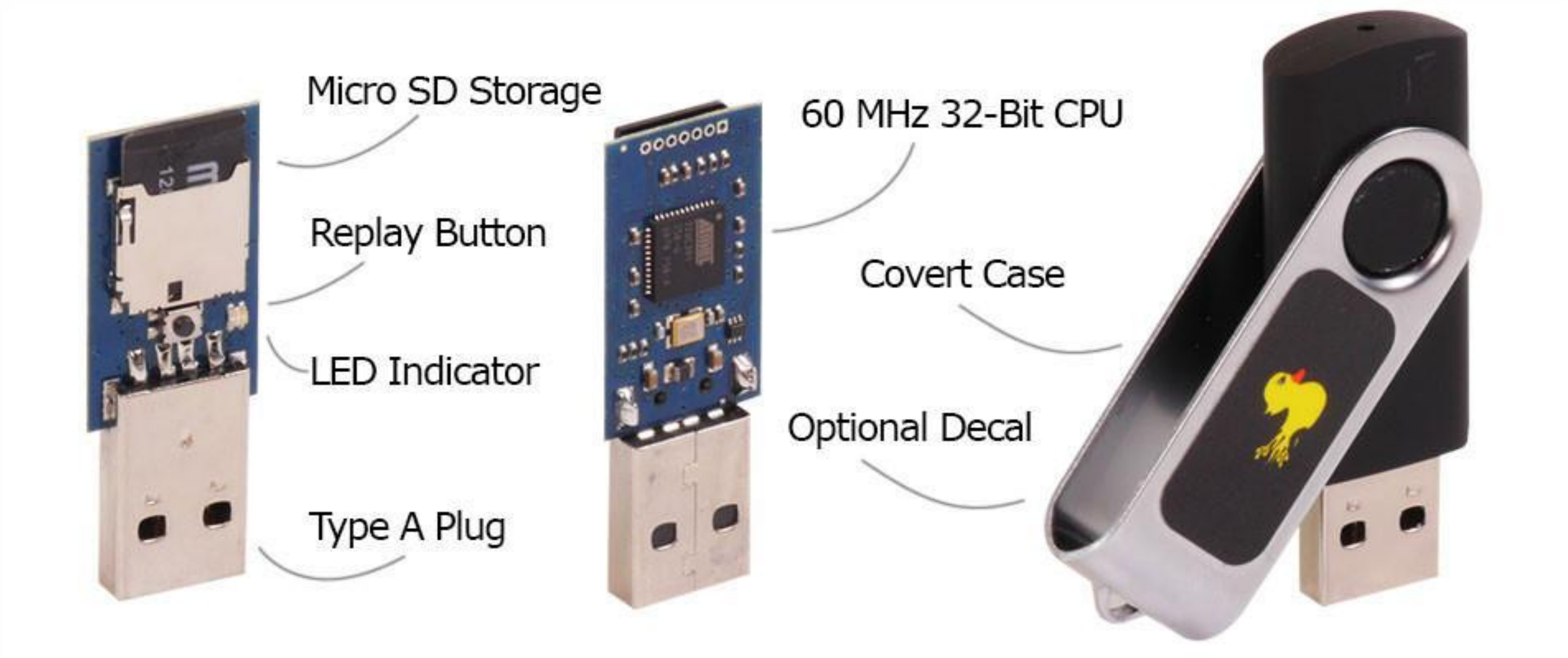
Host => Device => Host

- Interesting attack vector: exploit a legitimate USB device to hijack control and turn it into a BadUSB
- Lots of USB devices allow to reprogram the firmware by sending specific USB messages (example: iSeeYou)
- Can be exploited remotely, through WebUSB
- Can be used to break out of virtual machine environments

Part 3: Consumer ready BadUSB

Rubber Ducky

- "The USB Rubber Ducky is a keystroke injection tool disguised as a generic flash drive"



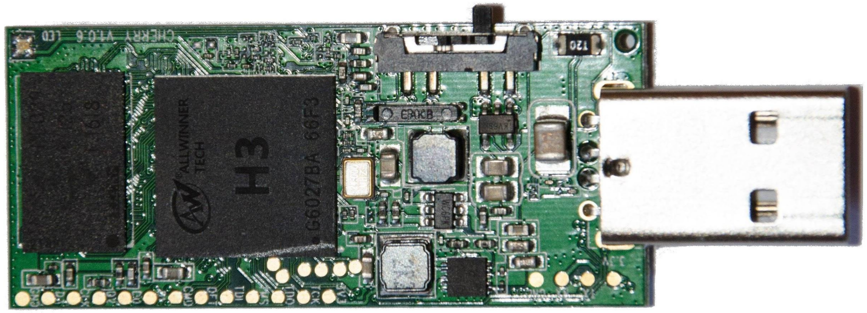
https://hakshop.com/products/usb-rubber-ducky-deluxe

Rubber Ducky

- Based on Atmel 32 bit AVR Microcontroller (AT32UC3B1256, other hardware listed here)
- Uses it's own language to describe keystroke payloads called Duckyscript
- A lot of available payloads
- **Horribly overpriced**: 45$ (vs 1.5$ for ATtiny45 or 10$ for CJMCU BadUSB)

Bash Bunny

- "The Bash Bunny by Hak5 is ... USB attack platform. It delivers ... by emulating ... gigabit Ethernet, serial, flash storage and keyboards"



https://hakshop.com/products/bash-bunny

https://forums.hak5.org/topic/40208-bash-bunny-specs/?tab=comments#comment-286703

Bash Bunny

- Quad-core ARM Cortex A7 (other hardware listed here)
- Allows to emulate a variety of USB devices: HID, Ethernet, Serial, Mass Storage
- **Quite overpriced**: 100$ (5$ Raspberry Pi Zero is a close alternative, but needs to be set up properly)

Lan Turtle

- "USB Ethernet adapter with covert backdoors"

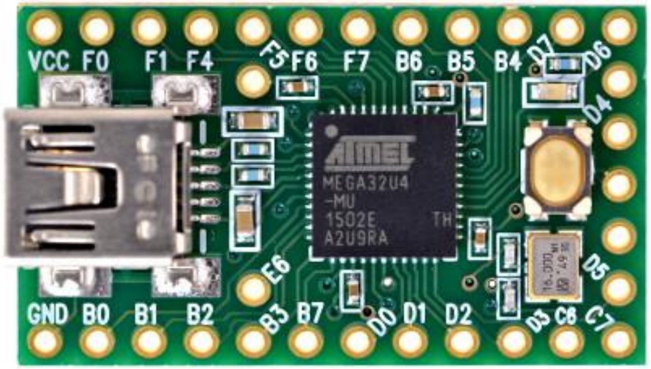
https://www.pantomath.xyz/hak5-lan-turtle-just-plug-hack-within-5-seconds/

Part 4: Microcontroller based BadUSB

Teensy

- "The Teensy USB Development Board is a complete USB-based microcontroller development system"

Teensy 3.2


VCC F0 F1 F4 F5 F6 F7 B6 B5 B4 D7 D6
D4
ATMEL
MEGA32U4
-MU
1502E
A2U9RA
E6
D5
GND B0 B1 B2 B3 B7 D0 D1 D2 D3 C6 C7

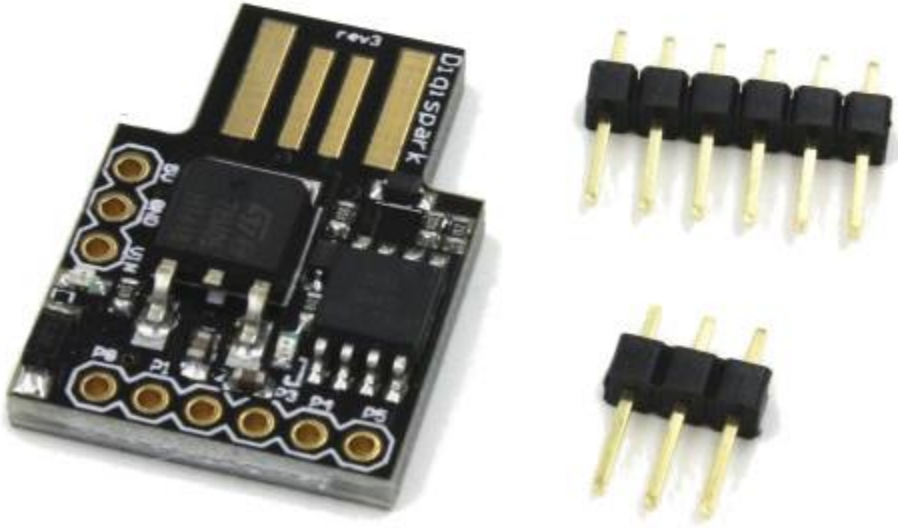
Teensy 2.0

https://www.pjrc.com/store/teensy32.html https://www.pjrc.com/store/teensy.html

Teensy

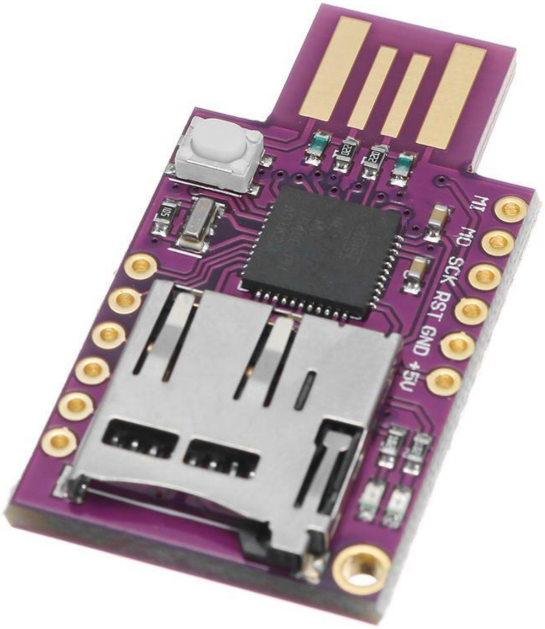
- Teensy 2.0 based on 8 bit AVR 16 MHz microcontroller (ATMEGA32U4), price: 16$
- Teensy 3.2 based on 32 bit ARM Cortex-M4 72 MHz (MK20DX256), price: 20$
- Can be programmed in C with Arduino Studio
- Has out-of-the-box support for emulating Serial, Keyboard, Mouse, Joystick, MIDI and Flight Sim USB devices
- Core libraries are open source

ATTiny85 board



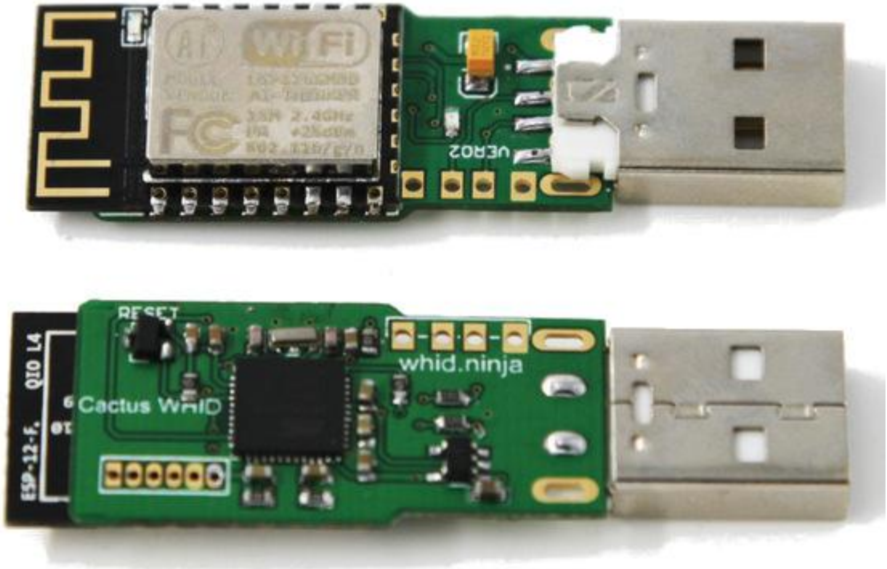
http://artofcircuits.com/product/digispark-attiny85-miniature-5v-usb-arduino-board

CJMCU BadUSB



https://www.pinterest.com/pin/751749362772345833/

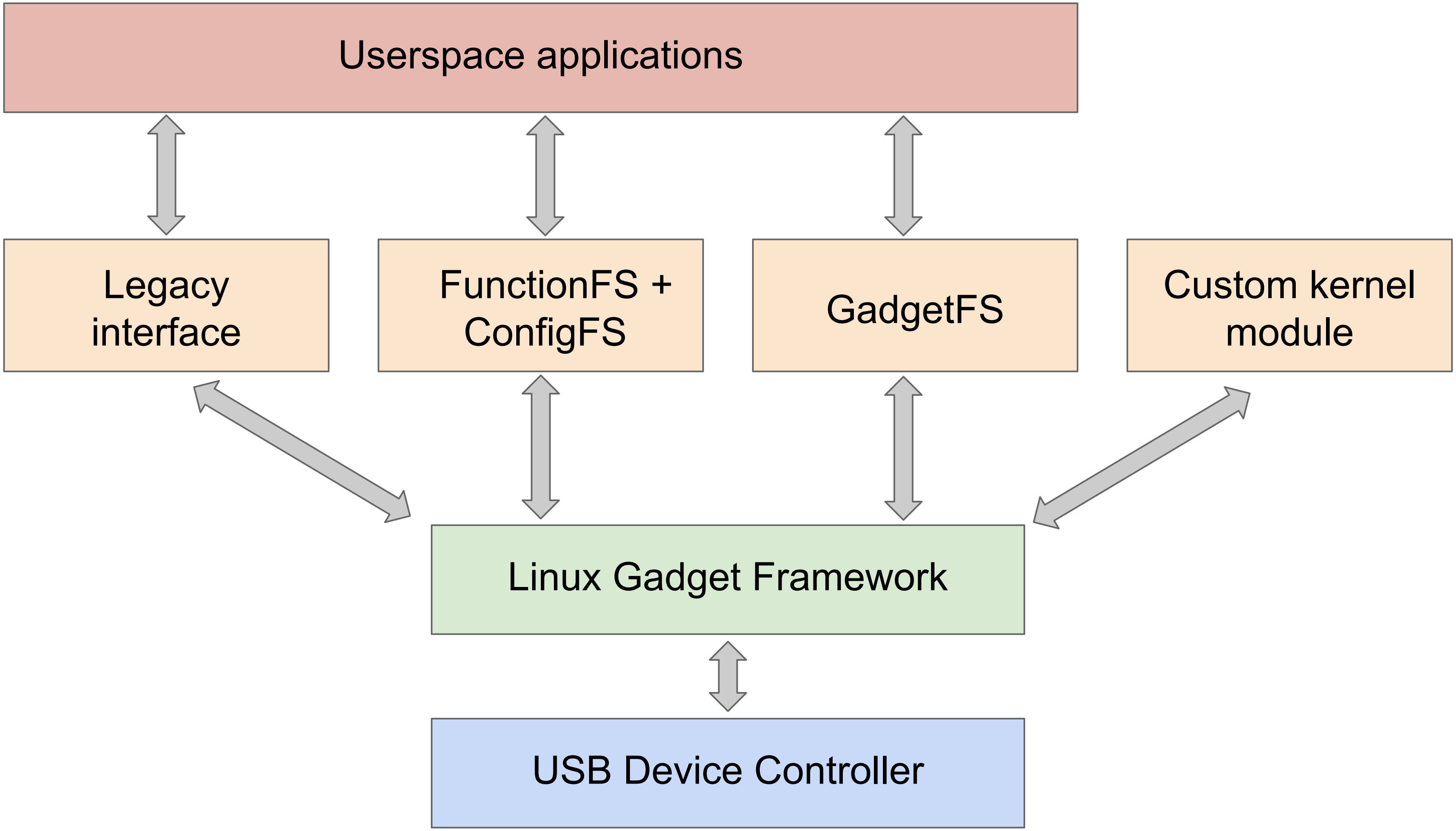
Cactus WHID



https://www.aliexpress.com/item/Cactus-Micro-compatible-board-plus-WIFI-chip-esp8266-for-atmega32u4/32318391529.html

Part 5: Linux gadget subsystem

Linux Gadget Subsystem



Legacy interface

- Available in the form of loadable kernel modules that emulate particular USB class devices
- Examples:
  - g_hid.ko - emulates HID devices
  - g_mass_storage.ko - Mass Storage
  - g_ether.ko - Ethernet
  - ...

FunctionFS + ConfigFS

- Allows to compose USB functions (each emulates a particular USB classes) into USB devices
- Basically a more convenient replacement for the legacy gadget interface
- Filesystem based interface

GadgetFS

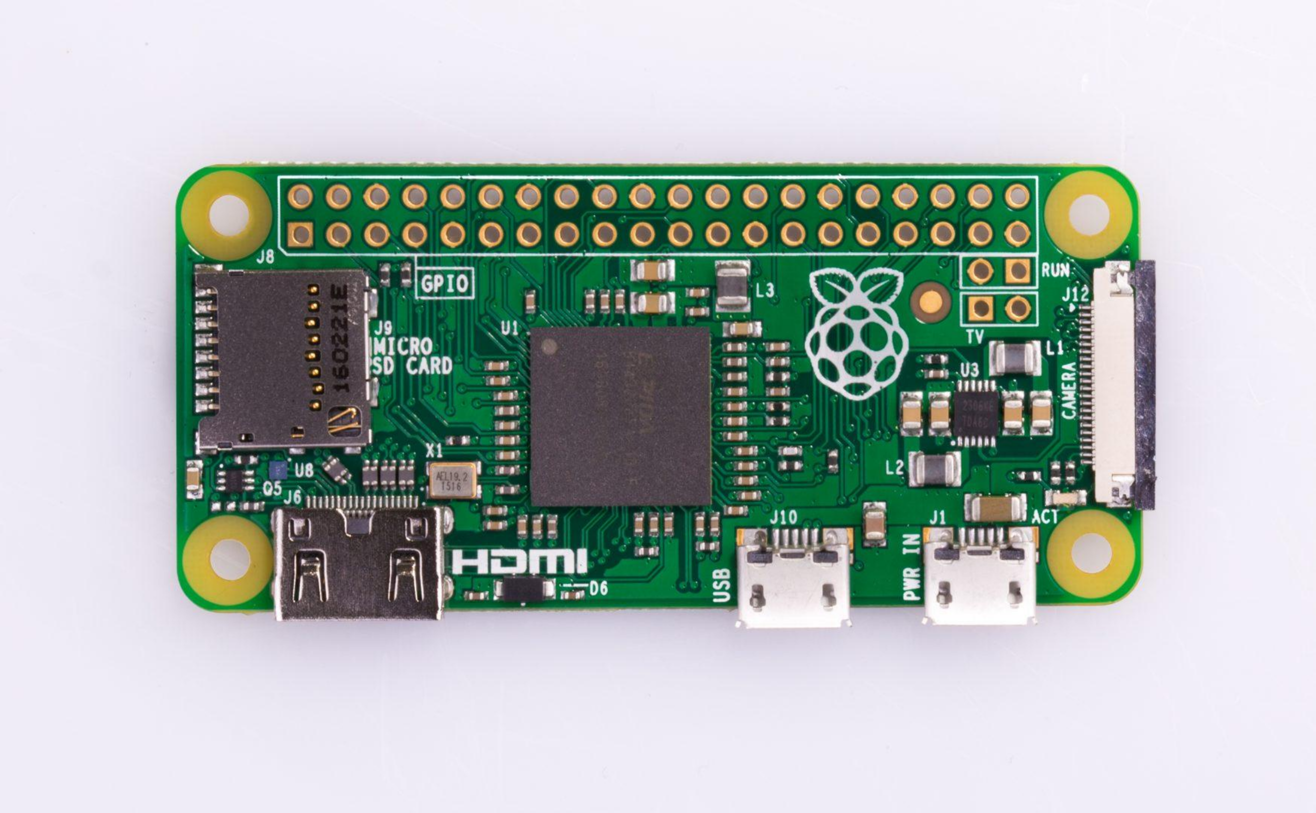
- Allows to emulate arbitrary (almost) USB devices
- Filesystem based interface
- The interface allows to receive USB messages sent to the device and reply when necessary

Custom kernel module

- The kernel provides internal API for creating USB gadgets
- Instead of using some pass-through interface from the userspace (GadgetFS, FunctionFS, …) we can implement a custom kernel modules that uses this API
- Allows a very low level control of the content of USB messages (invalid descriptors, etc.)

Raspberry Pi Zero

- A 5$ ARM based single board computer



https://www.raspberrypi.org/products/raspberry-pi-zero/

Raspberry Pi Zero

- 1GHz single-core CPU
- 512MB RAM
- Micro USB OTG port
- Micro USB power

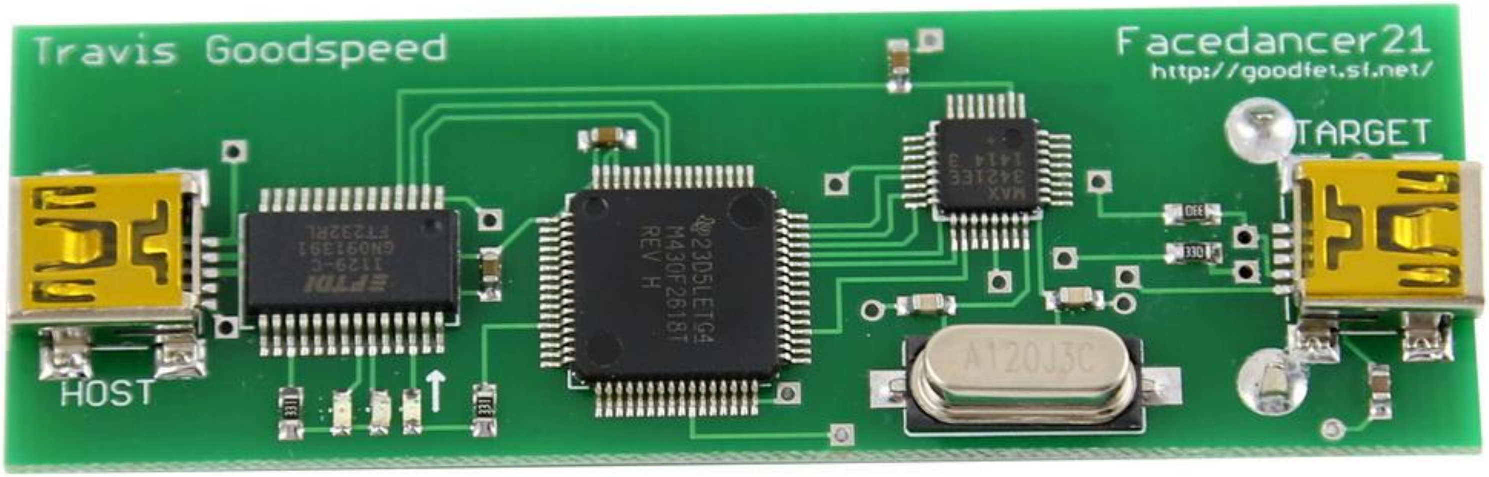
Raspberry Pi Zero W (10$):

- 802.11 b/g/n wireless LAN
- Bluetooth 4.1
- Bluetooth Low Energy (BLE)

Part 6: Facedancer21

FaceDancer21

- “The purpose of this board is to allow USB devices to be written in host-side Python, so that one workstation can fuzz-test the USB device drivers of another host”
- http://goodfet.sourceforge.net/hardware/facedancer21/



FaceDancer21

- https://github.com/travisgoodspeed/goodfet

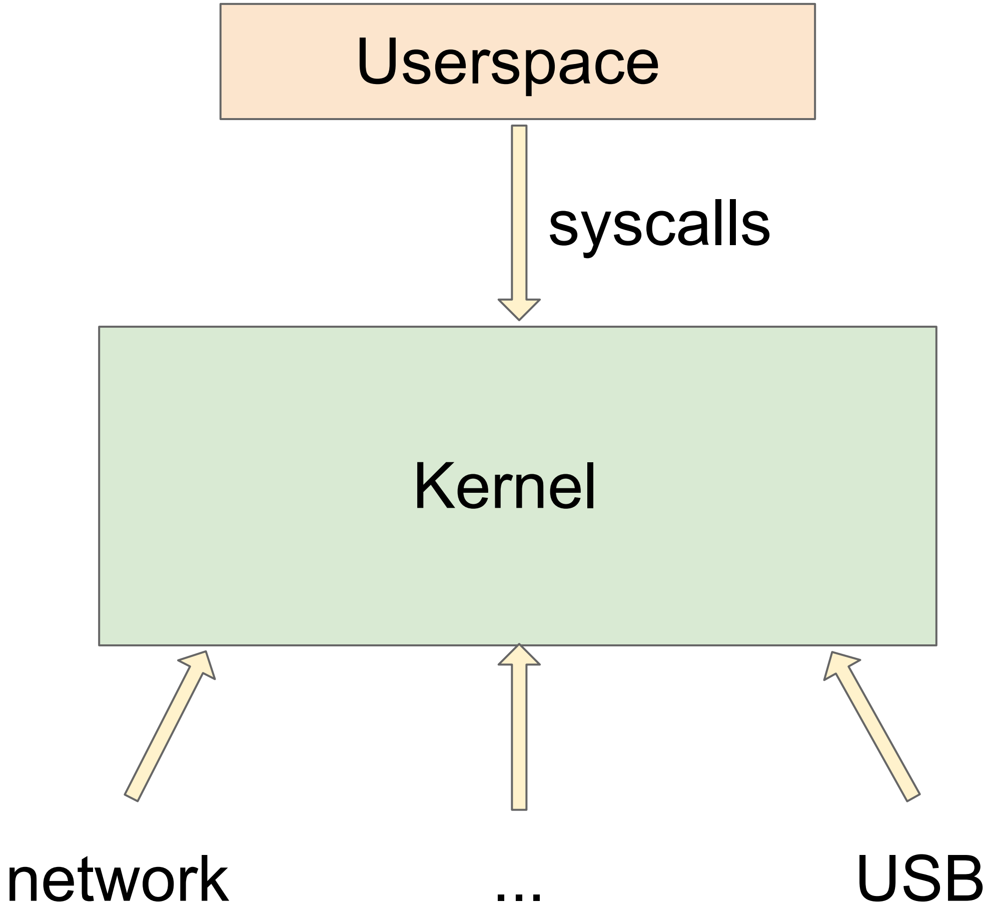
- https://github.com/ktemkin/Facedancer

- https://github.com/nccgroup/umap

- https://github.com/nccgroup/umap2

Part 7: USB fuzzing

Kernel inputs



syzkaller

- Coverage-guided syscall fuzzer for the Linux kernel
- As of now found over 1000 bugs ([1], [2]) (https://github.com/google/syzkaller/wiki/Found-Bugs)
- Fuzzes the kernel in a VM (e.g. QEMU) by using Linux Gadget API via a custom kernel module (+ dummy_hcd)
- Can perform external USB fuzzing
- Found over 80 bugs in USB core and drivers
- https://github.com/google/syzkaller

vUSBf

- Virtual USB fuzzer - a dedicated cross-OS fuzzer for USB
- Fuzzes the kernel by connecting random USB devices to QEMU running the kernel of interest via usbredir
- https://github.com/schumilo/vUSBf

CVE-2016-2384

- Double-free in USB-MIDI Linux kernel driver

- Found with vUSBf

- Confirmed and exploited with FaceDancer21

- https://xairy.github.io/blog/2016/cve-2016-2384

Part 8: USB sniffing

USB sniffing

- Commercial USB sniffers
  - Beagle (475$ for FS, 1400$ for HS, 6000$ for SS)
- OpenVizsla - open source FPGA based USB sniffer (the project is dead unfortunately)
- USBProxy - GadgetFS based USB sniffer
- USBProxy 'Nouveau' - Facedancer21 based USB sniffer

Questions?

Andrey Konovalov <andreyknvl@gmail.com>

https://github.com/xairy/hardware-village/tree/master/usb