Kernel Graphics Development on Remote Machines

2018-02-03, Joonas Lahtinen
Intel Corporation

About the author

Joonas Lahtinen <joonas.lahtinen@linux.intel.com>

- IRC: dolphin@freenode (Registered : Aug 20 15:42:25 2002 (15y 24w 2d ago))
  - That's actually when irc.openprojects.net turned into freenode.net
- Working at the Intel Open Source Technology Center since 2013
- Is a co-maintainer of the i915 driver in the DRM subsystem
  - i915 is the Linux kernel driver for Intel integrated graphics
- Has a long history as a home-based employee
- Worked with a lot with unstable prototype hardware in the past
  - Has been feeling the pain of remotely accessing machines!
- When not staring at more or less frozen screens, probably goes diving

Agenda

- Problem Statement
- Equipment and Setup
- Viewing the Machine
- Controlling the Machine
- Demo
- Questions

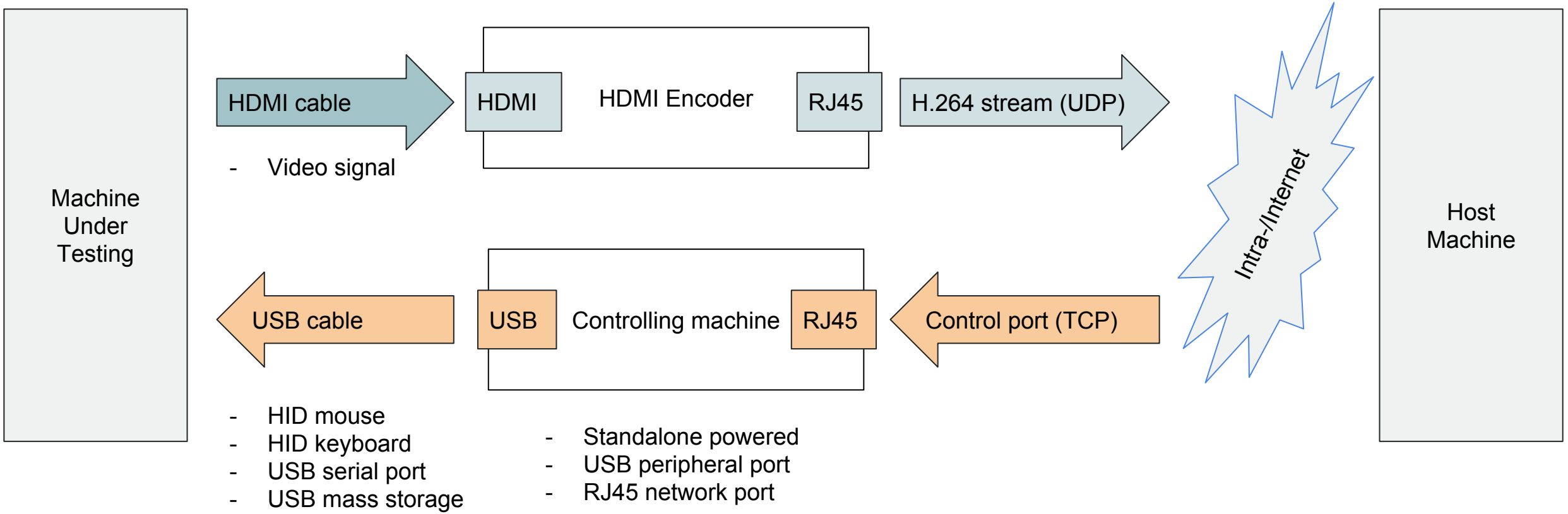
How to remotely access a machine for graphics development?

And remain sane...

What do we want to Interface for Development?

- Serial port or USB debug port
  - For early boot debugging (Documentation/x86/earlyprintk.txt)
- Network connection
  - For command line access and file transfer (SSH / SFTP)
  - netconsole (Documentation/networking/netconsole.txt)
- Display (at all times, not just remote desktop viewer)
  - Is Kernel Mode-Setting working?
  - For updating that broken BIOS
- Keyboard and mouse
- USB sticks
  - Recovering the system when the filesystem is corrupt
- Power button
  - For booting and for when other interfaces fail

System diagram



Equipment and Setup

Equipment Used in Demo

- Lenkeng LKV373A HDMI Extender V3.0
  - http://www.lenkeng.net/Index/detail/id/149
  - ~29 EUR (only Sender/TX is needed)
- BeagleBone Black (BBB)
  - https://beagleboard.org/black
  - ~55 EUR
- Generic 5V 2A power supply for BBB
  - ~10 EUR
- Generic USB stick
  - ~5 EUR

Off-the-shelf components, no soldering, easy to scale to a lab environment

Equipment Setup

BeagleBone Black

- ➔ Pre-built and tuned Linux distributions available at rcn-ee.net by Robert Nelson
  - ◆ http://rcn-ee.net/rootfs/2018-01-05/flasher/
  - ◆ CAUTION! Flasher images will overwrite eMMC contents, you've been warned!
- ➔ BBB-eMMC-flasher-debian-9.3-console-armhf-2018-01-05-2gb.img.xz
  - ◆ Debian found to be working best at the moment, reasonable performance
- ➔ Mainline kernel runs, Fedora ARM image can be installed
  - ◆ Performance is not so great, probably due to some unmerged driver patches :(

Equipment Setup

HDMI Sender

- ➔ Nothing for the device, plug and play!
  - ◆ Stock firmware downscales 1080p to 720p (GPU draws same amount, less bandwidth!)
  - ◆ Will multicast UDP stream even when there is no video signal
- ➔ Specialty consideration to remember when testing
  - ◆ Sends an occasional zero UDP packet, need to have latest FFmpeg or run:
  - ◆ `$ iptables -t raw -A PREROUTING -p udp -m length --length 28 -j DROP`
- ➔ For the curious, firmware can be exchanged to get Full HD resolution and unicast
  - ◆ https://blog.danman.eu/new-version-of-lenkeng-hdmi-over-ip-extender-lkv373a/
  - ◆ Lots of research on the device done by Daniel “danman” Kucera!

Viewing the machine

FFmpeg (built from source) or GStreamer

```
# The more robust solution based on the limited testing, zero packets are handled
host$ ffplay -fflags nobuffer uri=udp://239.255.42.42:5004

# Trips over timestamps when encoder resets them in a modeset, handles zero packets
host$ gst-launch-1.0 udpsrc uri=udp://239.255.42.42:5004 ! \
      tsdemux ! decodebin ! xvimagesink sync=false

# Split pipeline to squeeze last bits out of a VPN link
beaglebone$ socat TCP4-LISTEN:5004,reuseaddr,fork \
      EXEC:"gst-launch-1.0 udpsrc uri=udp\://239.255.42.42\:5004 caps='video/mpegts' ! \
      tsdemux ! 'video/x-h264' ! fdsink fd=1"
host$ gst-launch-1.0 tcpclientsrc host=192.168.1.243 port=5004 ! h264parse ! \
      openh264dec ! xvimagesink sync=false
```

Connecting from a real world network through proxy

"You shall not pass!" and other IT support quotes

➔ The quick'n dirty version, leading to ~50% CPU load

```
beaglebone$ socat TCP4-LISTEN:5004,reuseaddr,fork \
   UDP4-RECV:5004,bind=239.255.42.42,ip-add-membership=239.255.42.42:eth0
host$ ffplay http://beaglebone.lan:5004
```

➔ A more optimized solution with http://udpxy.com (~20% CPU load), OpenWrt has it too

```
beaglebone or router$ udpxy -p 4022
host$ ffplay http://beaglebone.lan:4022/udp/239.255.42.42:5004
```

Controlling the machine

libcomposite module (Documentation/usb/gadget_configfs.txt)

```
# First, add “exit” to the start of /opt/scripts/boot/am335x_evm.h and reboot
$ modprobe libcomposite && cd /sys/kernel/config/usb_gadget
$ mkdir my_gadget && cd my_gadget
… setup steps, write to idProduct, idVendor and other files ...
$ mkdir functions/hid.mouse
$ echo 3 > functions/hid.mouse/report_length
$ cat mouse.report_desc.bin > functions/hid.mouse/report_desc
...
$ ln -s functions/hid.mouse configs/c.1/
$ ls /sys/class/udc > UDC # to activate
```

Move the mouse or press keyboard keys

- For mouse, it's quite easy, just like moving a turtle on the screen

```
bbb$ echo -en "\x00\x10\x00" > /dev/hidg0 # To move slightly to right
bbb$ echo -en "\x00\x00\x10" > /dev/hidg0 # To move slightly to up
bbb$ echo -en "\x01\x00\x00" > /dev/hidg0 # bit0 = left, bit1 = right, bit2 = mid
```

- For keyboard, to avoid key repeat you need to work harder

```
#!/bin/sh
echo -en "\x00\xFF\x04\x00\x00\x00\x00\x00" # Key "a" (SDL2/SDL_scancode.h)
usleep 200000 # Sleep some 200 milliseconds
echo -en "\x00\xFF\x00\x00\x00\x00\x00\x00" > /dev/hidg0 # Reset state
```

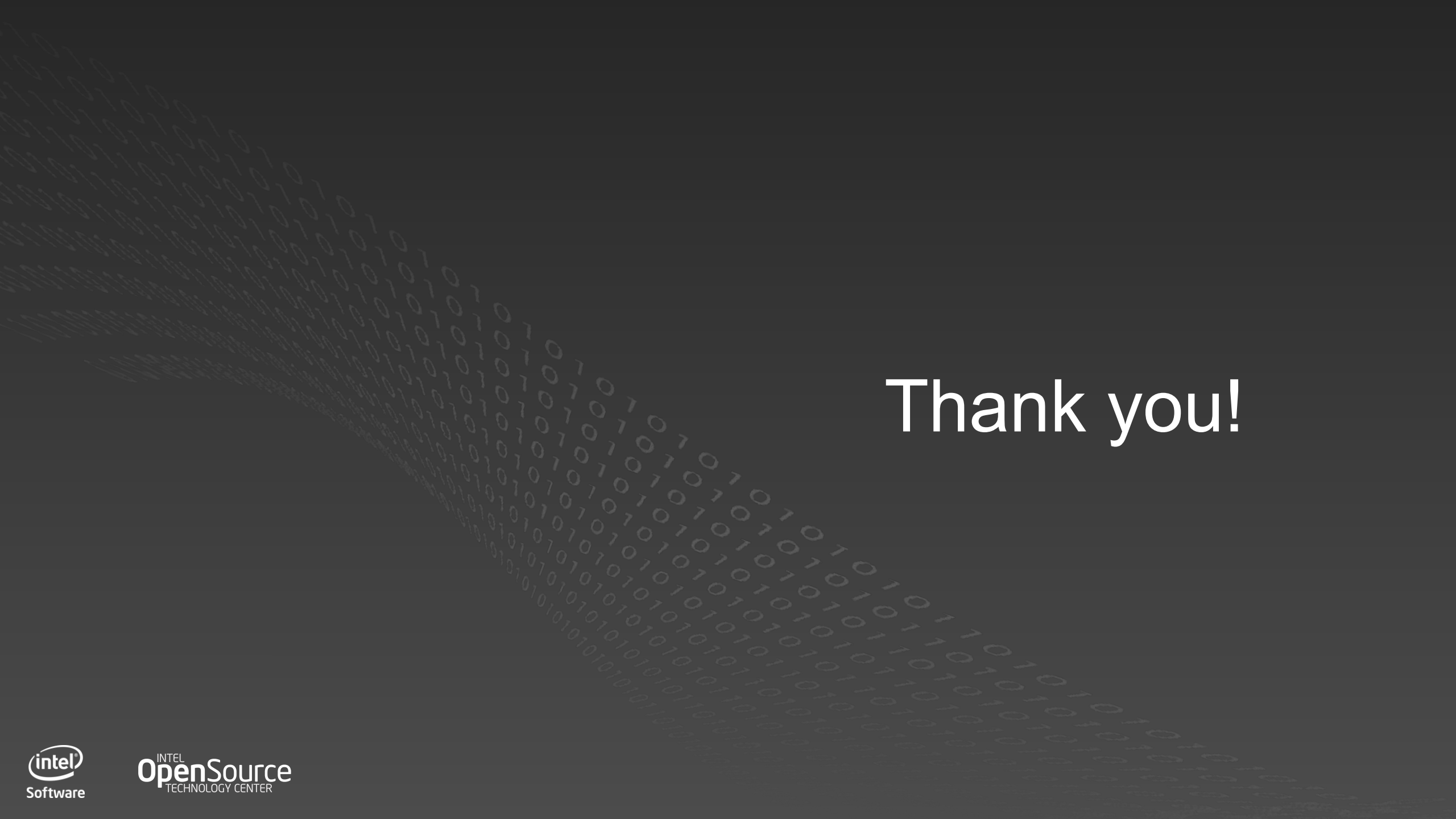
CAUTION! Note -n argument, the report length must match exactly!


Demo

Low-hanging fruits

- Simulate keypresses of certain length on the controlling machine
  - Avoid network induced sssssssssssstuck kkkkkkkkkkkkkkeys
- “mut-guest-addons” for clipboard handling through the emulated serial port
- Chamelium as video receiver (suggestion from Martin Peres)
  - Just 1 FPS decode for now, more versatile in connectors

Questions / discussion



Backup

Keyboard and mouse

- Standard: http://www.usb.org/developers/hidpage/HID1_11.pdf
- `$ hidrd-convert /sys/devices/.../usbN/.../report_descriptor -o xml`
- Standard mouse report is 3 bytes

0x01	0x43	0x10
<buttons>	<x-motion>	<y-motion>
bits 0-2	-127 – 127)	-127 – 127

- Standard keyboard report is 8 bytes

0x00	0xFF	0x04	0x05	0x00 0x00 0x00 x00
<modifiers>	<reserved>	<scancode 0>	<scancode 1>	...
bits 0-8	<FF>	4 – 101	4 – 101	...