02 November 2018

Writing Your Own Gadget

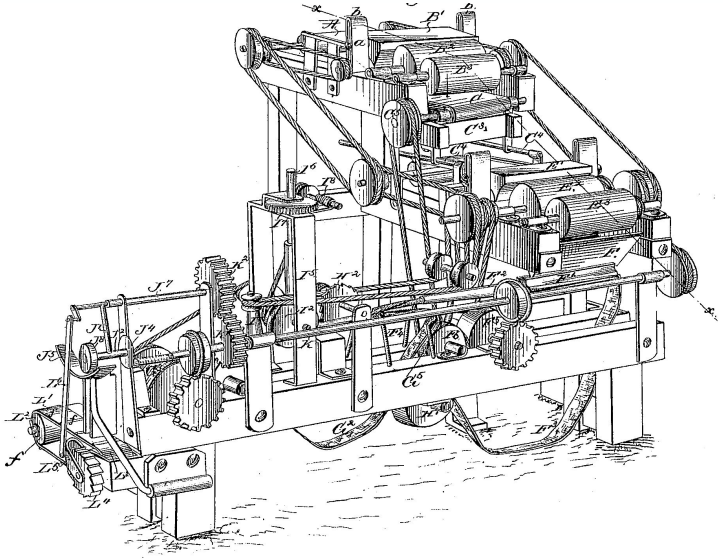
Andrei Emeltchenko

Contents

Introduction: Problem

- Devices around (sensors, switches, etc)
- Problem connecting to devices with non standard PC interfaces
  - SPI, I2C, etc

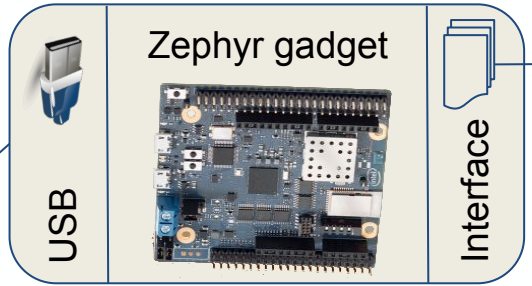
Access

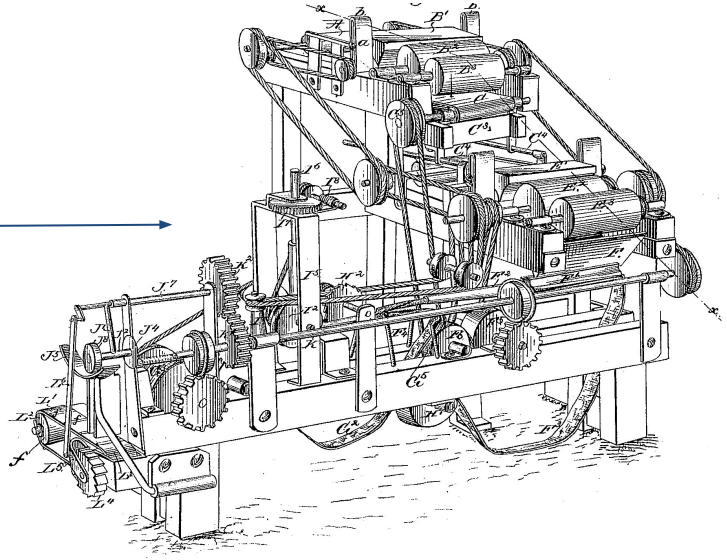


Complex Hardware

Introduction: Solution

- Use embedded board with special interfaces powered by Zephyr OS
- Zephyr board connects to Host via USB





Complex Hardware

Introduction: Zephyr

- Open Source RTOS for connected resource constrained devices https://www.zephyrproject.org/what-is-zephyr/
- Hosted by Linux Foundation
- Supported more then 100 boards: https://docs.zephyrproject.org/latest/boards/boards.html
- Zephyr Project Documentation https://docs.zephyrproject.org/latest/index.html
- License: Apache 2.0
- Source: https://github.com/zephyrproject-rtos/zephyr

Zephyr™ Project

Hello world in Zephyr

- Set up a development system
  https://docs.zephyrproject.org/latest/getting_started/getting_started.html#set-up-a-development-system
- Set up build environment

```
$ source zephyr-env.sh
```

- Build hello_world sample for Qemu

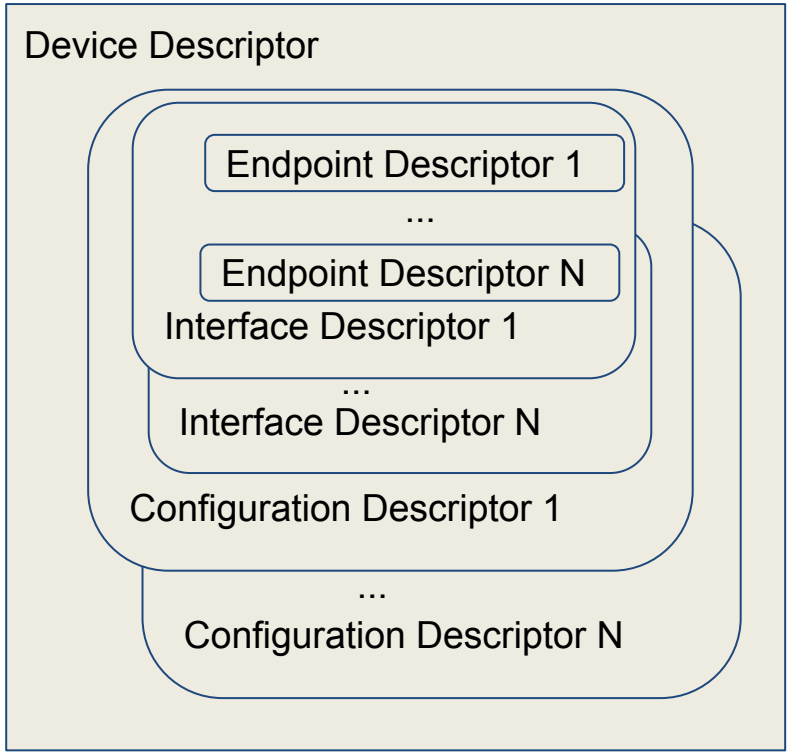
```
$ cd samples/hello_world/
$ mkdir build
$ cd build
# Use cmake to configure a Make-based build
$ cmake -DBOARD=qemu_x86 ..
# Now run make on the generated build system:
$ make
```

- Run hello_world in Qemu

```
$ make run

To exit from QEMU enter: 'CTRL+a, x'
[QEMU] CPU: qemu32,+nx,+pae

***** Booting Zephyr OS zephyr-v1.13.0 *****
Hello World! qemu_x86
```

USB: General overview

- One Host connected to many devices
- Device identifies itself through Descriptors
- Descriptors are binary data describing USB capabilities
  - Device Class
  - Product ID / Vendor ID
  - Configuration, Interfaces, Endpoints
- Endpoints are communication channels between Host and Device



Programming USB gadgets: Standard Classes

- Basic USB Device Classes supported by Zephyr Device Stack
  - Can be enabled in Zephyr application via menuconfig

```
(top menu)
    ↑↑↑↑↑↑↑↑↑↑↑↑↑↑        Zephyr Kernel Configuration
    Console  --->
    Debugging Options  --->
    Disk  --->
    File Systems  --->
    Logging Options  --->
    Management  --->
    Networking  --->
    Shell Options  --->
[*] USB device stack  --->
    DFU options  --->
[ ] Non-random number generator
    Random generator  ---> (empty)
    Storage  ---> (empty)
    General Kernel Options  --->
    Framebuffer  --->
    External Sources  --->
    Testing  --->
```

```
(top menu) → USB device stack
                          Zephyr Kernel Configuration
    Max compiled-in log level for usb device (Info)  --->
(0x2FE3) USB Vendor ID
(0x100) USB Product ID
(ZEPHYR) USB manufacturer name
(USB-DEV) USB product name
(0.01) USB serial number
[ ] Enable composite device driver
[ ] Enable USB Binary Device Object Store (BOS)
[ ] Enable MS OS Descriptors support
[ ] USB CDC ACM Device Class Driver
[ ] USB Mass Storage Device Class Driver
[*] USB Bluetooth Device Class Driver
[ ] USB Loopback Function Driver
    USB Device Networking support  --->
[ ] USB Human Interface Device support
```

Programming USB gadgets: Custom Device

Define Device Descriptors

```
static const struct dev_common_descriptor {
    struct usb_device_descriptor device_descriptor;
    struct usb_cfg_descriptor configuration_descr;
    struct usb_device_config {
        struct usb_if_descriptor if0;
        struct usb_ep_descriptor if0_in_ep;
    } __packed device_configuration;
} __packed desc = {
    .device_descriptor = {
        .bDeviceClass = CUSTOM_CLASS,
        .idVendor = sys_cpu_to_le16((u16_t)CONFIG_USB_DEVICE_VID),
        .idProduct = sys_cpu_to_le16((u16_t)CONFIG_USB_DEVICE_PID),
        .bcdDevice = sys_cpu_to_le16(BCDDEVICE_RELNUM),
    },
    .device_configuration = {
        .if0 = {
            .bInterfaceClass = CUSTOM_CLASS,
            .bInterfaceSubClass = WPANUSB_SUBCLASS,
            .bInterfaceProtocol = WPANUSB_PROTOCOL,
        },
        .if0_in_ep = {
            .bEndpointAddress = ENDP_BULK_IN,
            .bmAttributes = USB_DC_EP_BULK,
...
```

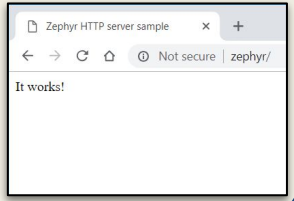
Define Endpoints and config data and enable

```
static struct usb_ep_cfg_data ep[] = {
    {
        .ep_cb = bulk_in,
        .ep_addr = ENDP_BULK_IN
    },
};
static struct usb_cfg_data config = {
    .usb_device_description = (u8_t *)&desc,
    .cb_usb_status = status_cb,
    .interface = {
        .vendor_handler = NULL,
        .class_handler = NULL,
        .custom_handler = NULL,
    },
    .num_endpoints = ARRAY_SIZE(ep),
    .endpoint = ep,
};
```

```
/* Initialize the USB driver with the right configuration */
ret = usb_set_config(&wpanusb_config);

/* Enable USB driver */
ret = usb_enable(&wpanusb_config);
```

Ethernet over USB: Use case

Browser



Host PC

Apps: DNS / LLMNR / MDNS / CoAP / MQTT / HTTP

Transport Layer: UDP / TCP

Network Layer: IPv4 / IPv6

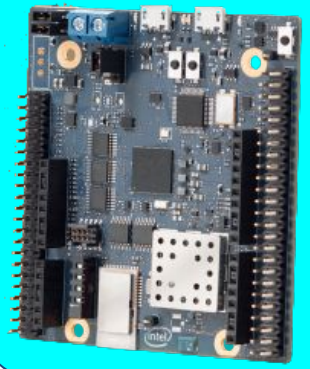
Physical Layer: ECM / EEM / RNDIS Ethernet frames

USB

Zephyr app

http server

Zephyr gadget



Ethernet over USB: Standards

- Motivation: Application independent data exchange
- Standards:
  - Microsoft Remote NDIS (RNDIS)
  - Communications Device Class (CDC) protocols
    - Ethernet Control Model (ECM)
    - Ethernet Emulation Model (EEM)
    - Network Control Model (NCM)
- macOS supports CDC ECM
- MS Windows supports RNDIS
- Linux support all protocols

Ethernet over USB: Enabling in Zephyr

- Zephyr supports RNDIS, ECM and EEM
- Simple configuration: protocols are implemented in USB Device Stack
  - User only need to select Ethernet over USB checkboxes

```
(top menu)
    Bluetooth  --->
    Console  --->
    Debugging Options  --->
    Disk  --->
    File Systems  --->
    Logging Options  --->
    Management  --->
    Networking  --->
    Shell Options  --->
[*] USB device stack  --->
    DFU options  --->
[*] Non-random number gener
    Random generator (x86 t
    Storage  ---> (empty)
    General Kernel Options
    External Sources  --->
    Testing  --->
```

```
(top menu) → USB device stack
                    Zephyr Kernel
    Max compiled-in log level for usb de
(0x2FE3) USB Vendor ID
(0x100) USB Product ID
(ZEPHYR) USB manufacturer name
(USB-DEV) USB product name
(0.01) USB serial number
[ ] Enable composite device driver
[ ] Enable USB Binary Device Object Stor
[ ] Enable MS OS Descriptors support
[ ] USB CDC ACM Device Class Driver
[ ] USB Mass Storage Device Class Driver
[ ] USB Bluetooth Device Class Driver
[ ] USB Loopback Function Driver
    USB Device Networking support  --->
[ ] USB Human Interface Device support
```

```
(top menu) → USB device stack → USB Device Networking support
                    Zephyr Kernel Configuration
[*] USB Ethernet Control Model (ECM) Networking device
[ ] USB Ethernet Emulation Model (EEM) Networking device
[ ] USB Remote NDIS (RNDIS) Networking device
    USB Device Network log level (Warning)  --->
```

Ethernet over USB: Zero configuration

- IPv4 address autoconfiguration for Host and Zephyr

```
IcannIan_00:53:01          Broadcast          |
|                                |             |
|   Who has 169.254.81.1         |             | ARP: Who has 169.254.81.199? Tell 0.0.0.0
|------------------------------>|             |
|   Gratuitous ARP               |             | ARP: Gratuitous ARP for 169.254.81.199 (Request)
|------------------------------>|             |
```

- Link Local Multicast Name Resolution (LLMNR)

```
Host                  224.0.0.252            Zephyr          |
|                          |                     |            |
|  Standard query A zephyr |                     |            | LLMNR: Standard query A zephyr
|------------------------->|                     |            |
|  Standard query resp A zephyr 169.254.9.70     |            | LLMNR: Standard query response
|<-----------------------------------------------|            | A zephyr A 169.254.9.70
```

- **zephyr** can now be used as a hostname

Ethernet over USB: HTTP server app

Zephyr app + extra configuration

```
# USB Device settings
CONFIG_USB=y
CONFIG_USB_DEVICE_STACK=y

# Select USB Configurations
CONFIG_USB_DEVICE_NETWORK_ECM=y
```

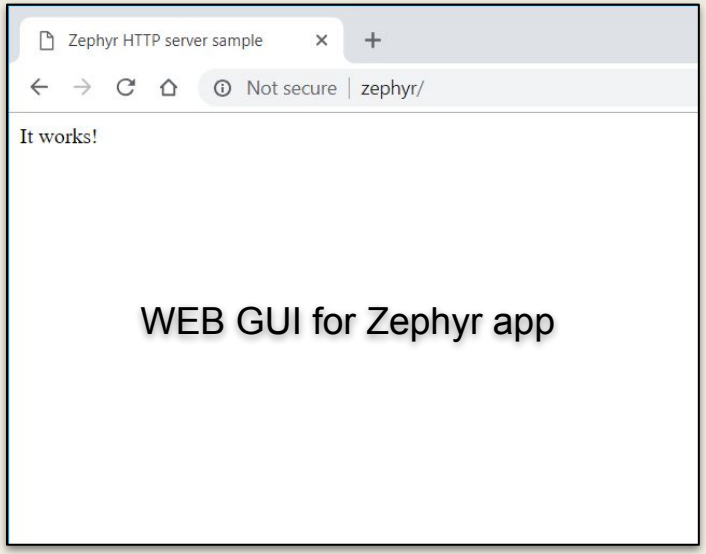
Ethernet over USB overlay config

```
# Zero Configuration
CONFIG_NET_IPV4_AUTO=y
CONFIG_NET_HOSTNAME_ENABLE=y
CONFIG_DNS_RESOLVER=y
CONFIG_LLMNR_RESOLVER=y
CONFIG_LLMNR_RESPONDER=y
```

Zero Configuration overlay config

```
$ cmake ... -DOVERLAY_CONFIG=overlay-netusb.conf
```

Host PC

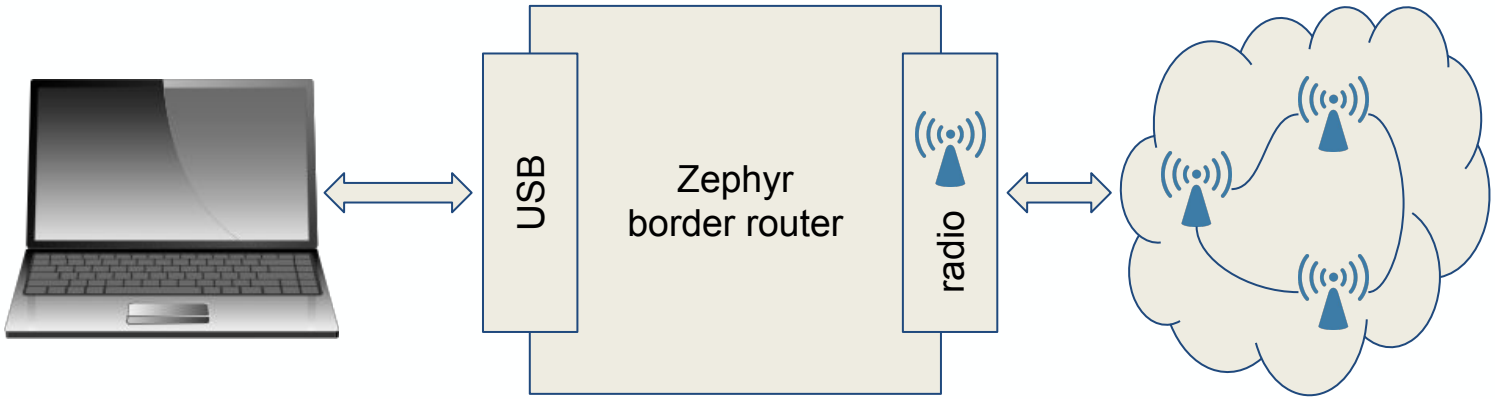


WEB GUI for Zephyr app

Sample README https://docs.zephyrproject.org/latest/samples/net/http_server/README.html

Ethernet over USB: border router app

- Border Router connects networks with different routing domains
- Zephyr capabilities:
  - Multiple interfaces
  - Routing



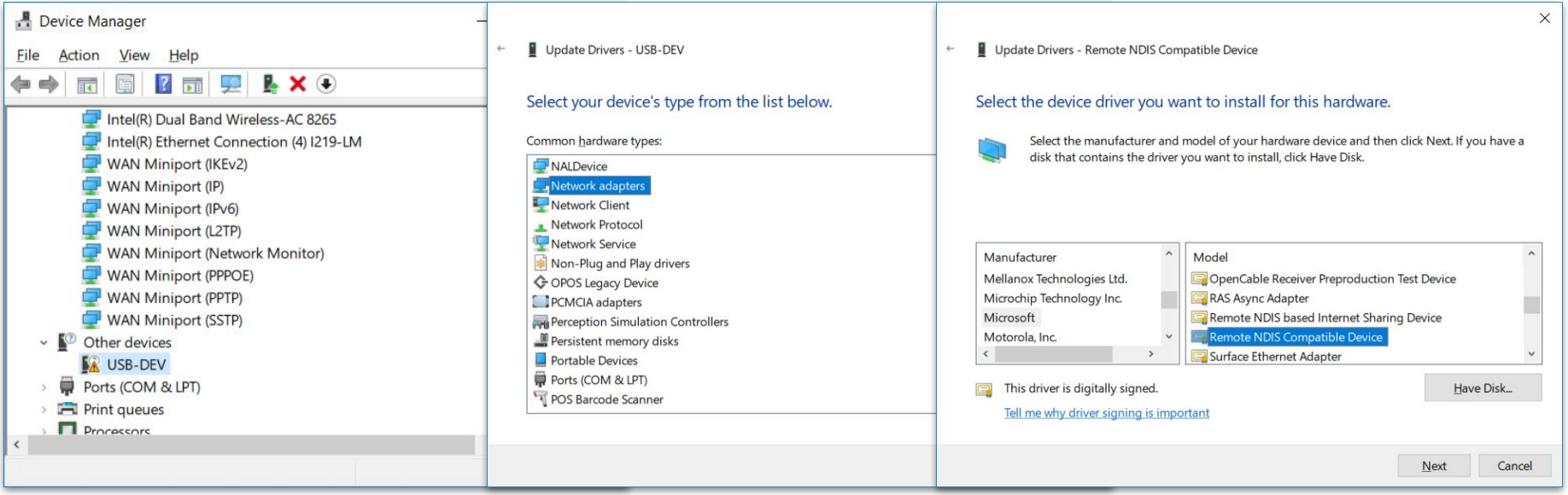
some custom radio network

Ethernet over USB Drivers: Linux

- ECM, EEM, RNDIS protocols are all supported by Linux

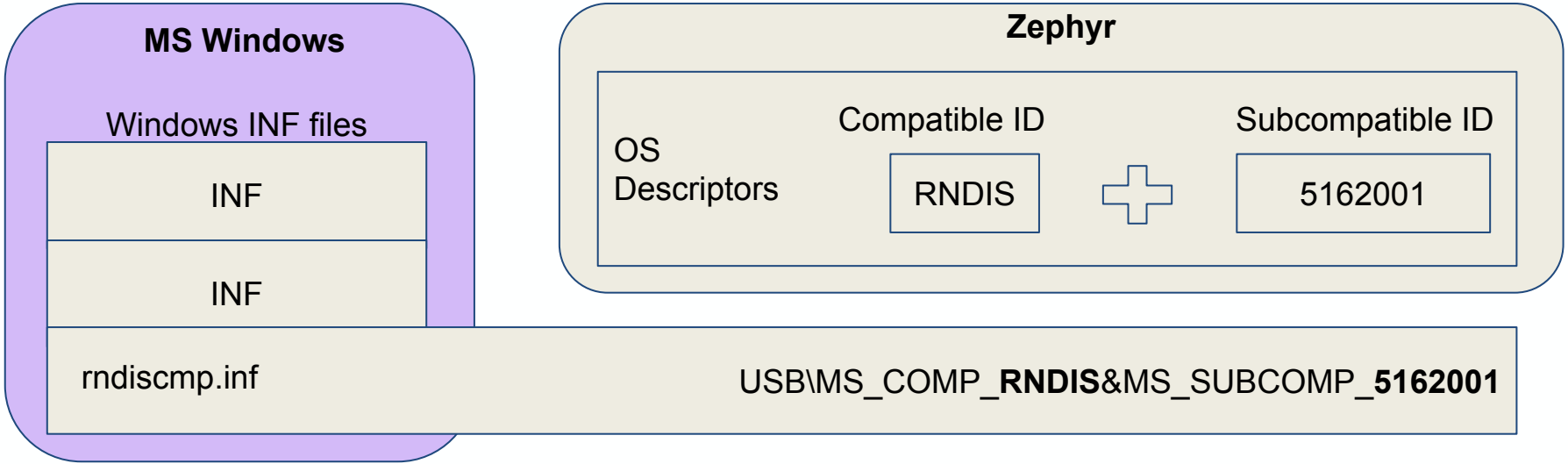
Ethernet over USB Drivers: Windows

- Only MS RNDIS is supported
- RNDIS gadget may be recognized as Serial / COM
  - Device Class / Subclass matches usbser.inf



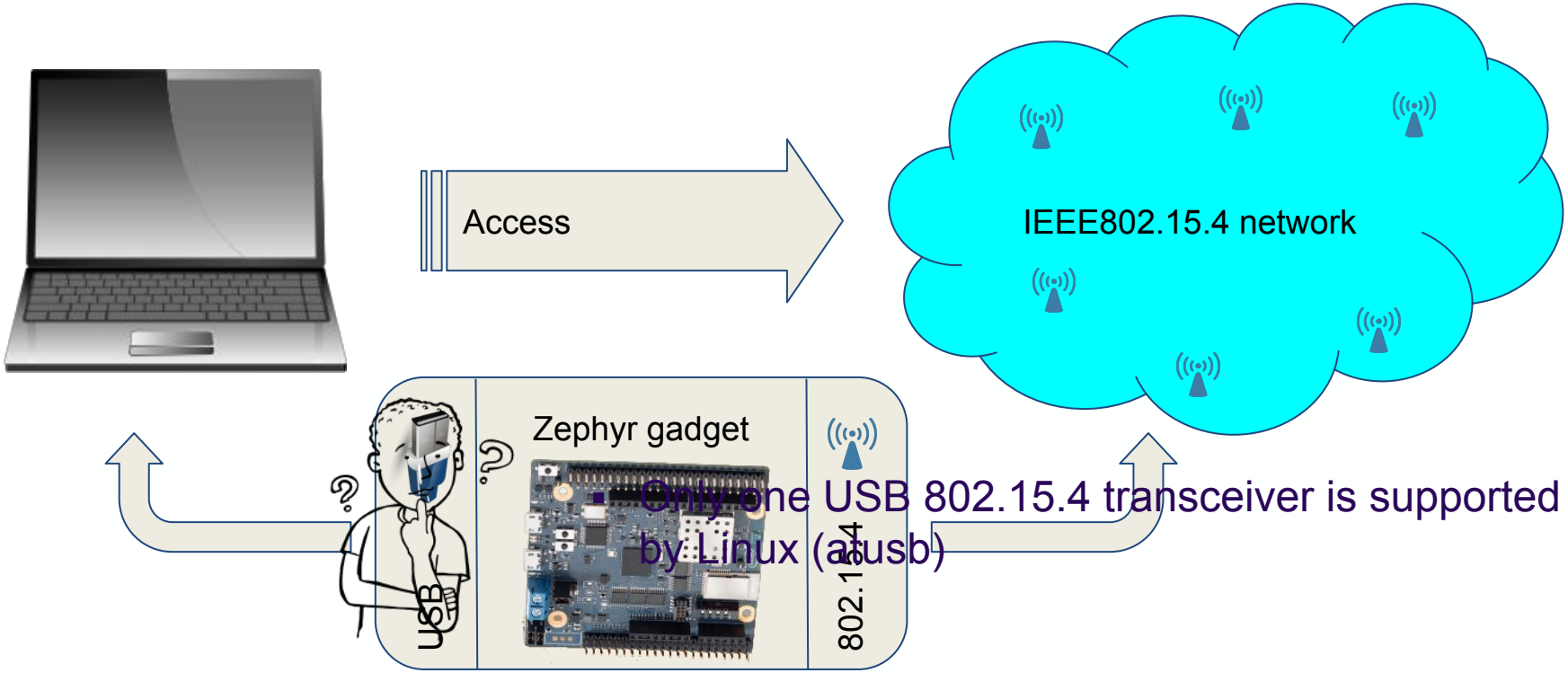
OS Drivers for Zephyr USB: OS Descriptors

- Microsoft OS Descriptors help to autoconfigure MS Windows driver
  - Implemented for RNDIS at the moment
  - Tested with MS Windows 8.1 / 10



Compatibe ID: https://docs.microsoft.com/en-us/windows-hardware/drivers/install/compatible-ids

IEEE802.15.4 USB: Use case



- Only one USB 802.15.4 transceiver is supported by Linux (atusb)

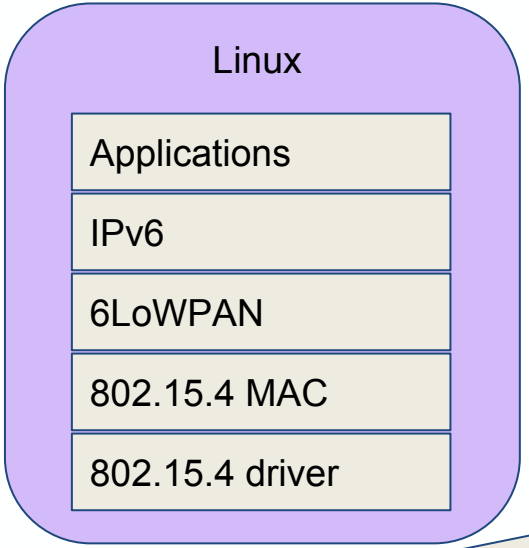
IEEE802.15.4 USB gadget

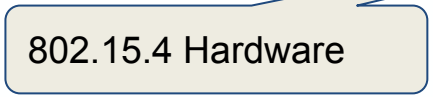
- A board with 802.15.4 radio and USB might be used as a USB adapter
  - Board, radio and USB shall be supported by Zephyr

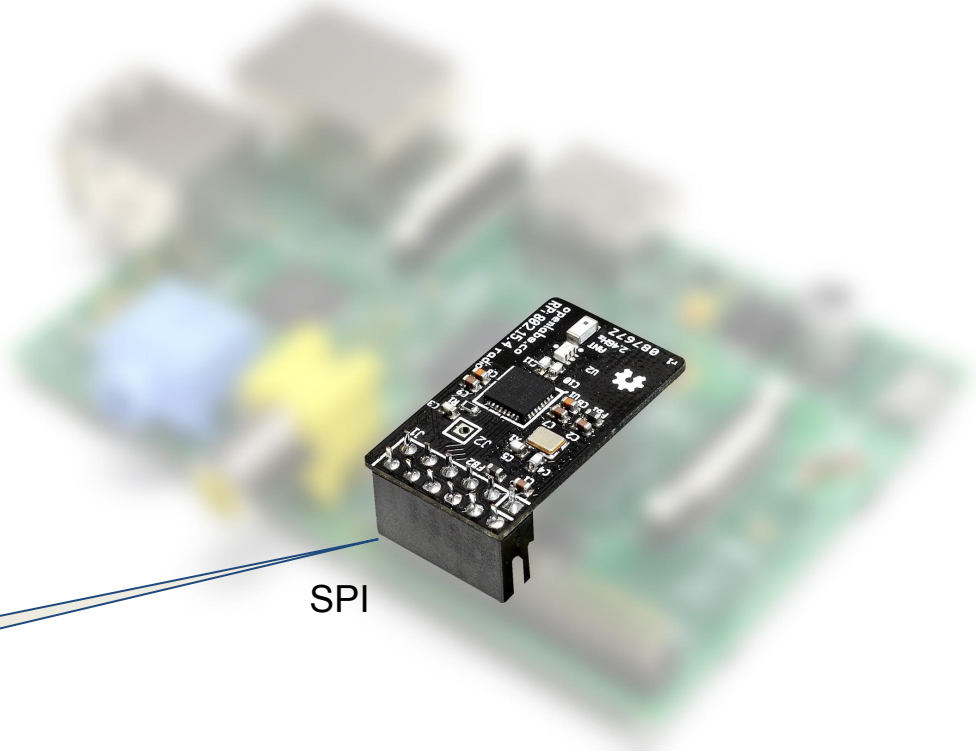
Board used:
Intel C1000 Eval Kit
- TI CC2520 802.15.4 radio module connected over SPI
- DW USB controller

Source code: https://github.com/zephyrproject-rtos/zephyr/tree/master/samples/net/wpanusb

Low Power Wireless Linux stack

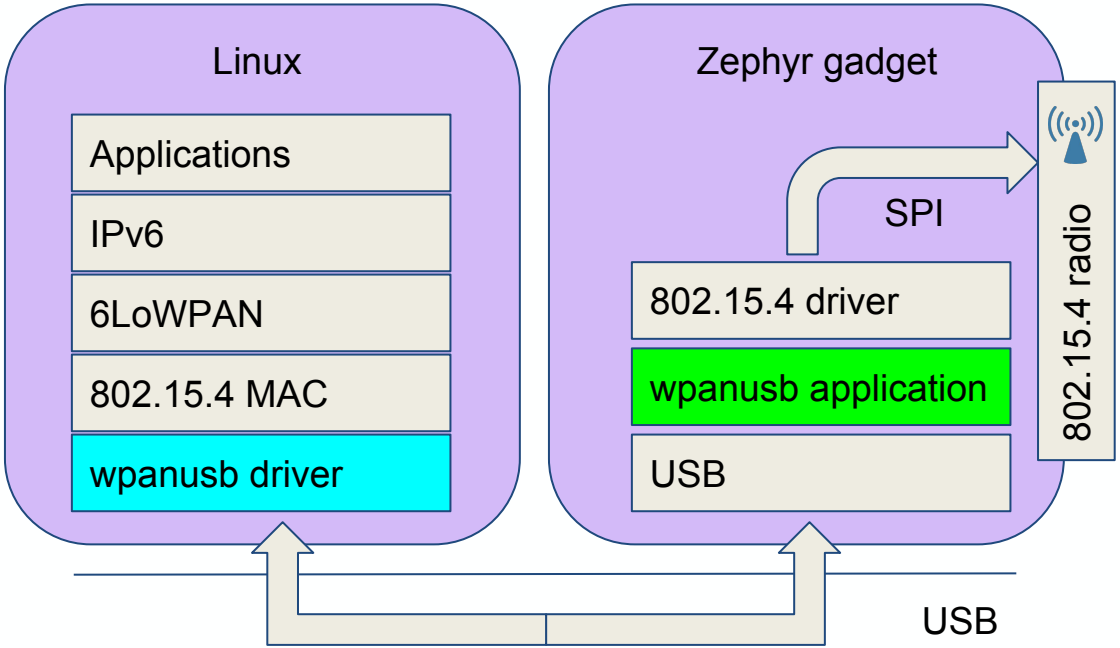






Raspberry Pi + OpenLabs Atmel AT86RF233 radio

IEEE802.15.4 USB gadget



IEEE802.15.4 USB gadget: API

Zephyr 802154 Driver API

```
struct ieee802154_radio_api {
    ...
    int (*start)(struct device *dev);
    int (*stop)(struct device *dev);
    int (*tx)(struct device *dev,
              struct net_pkt *pkt,
              struct net_buf *frag);
    int (*set_channel)(struct device *dev,
                       u16_t channel);
    int (*set_txpower)(struct device *dev, s16_t dbm);
    ...
} __packed;
```

Linux 802154 SoftMAC API

```
struct ieee802154_ops {
    ...
    int (*start)(struct ieee802154_hw *hw);
    void (*stop)(struct ieee802154_hw *hw);
    int (*xmit_async)(struct ieee802154_hw *hw,
                      struct sk_buff *skb);
    int (*set_channel)(struct ieee802154_hw *hw,
                       u8 page, u8 channel);
    int (*set_txpower)(struct ieee802154_hw *hw,
                       s32 mbm);
    ...
};
```

opcode: **SET_CHANNEL**
parameters: channel, page

setup.bRequest	data
opcode	parameters

USB

IEEE802.15.4 USB: Linux driver

- SoftMAC driver similar to atusb Linux kernel driver
- Protocol defined: https://github.com/zephyrproject-rtos/zephyr/blob/master/samples/net/wpanusb/wpan-radio-spec.txt
- Vendor specific class - driver loads on specific Product ID / Vendor ID pair
- Zephyr has own Vendor ID (0x2FE3) https://docs.zephyrproject.org/latest/subsystems/usb/usb.html#usb-vendor-and-product-identifiers
- Source code: https://github.com/finikorg/wpanusb

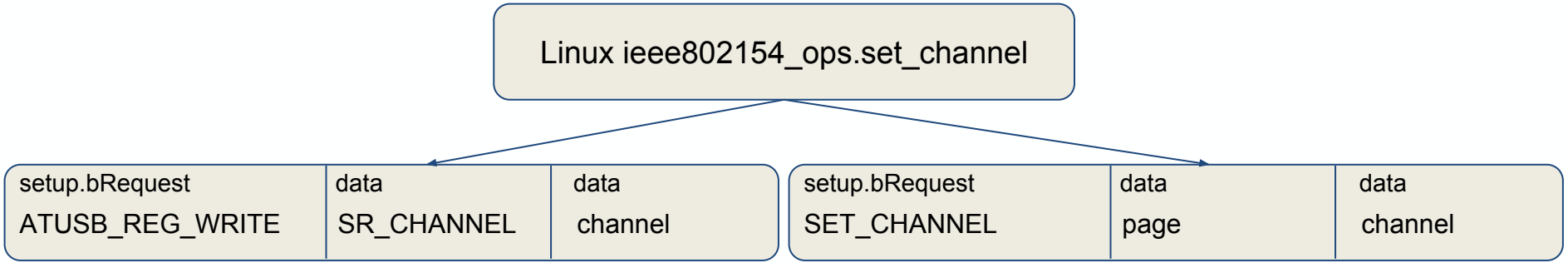
IEEE802.15.4 atusb vs wpanusb protocol

atusb

- Vendor - dependent protocol

wpanusb

- Vendor - independent GENERIC protocol



IEEE802.15.4 USB: Linux driver

usb-devices output for the gadget:

```
T:  Bus=01 Lev=03 Prnt=03 Port=01 Cnt=01 Dev#= 23 Spd=12  MxCh= 0
D:  Ver= 1.10 Cls=ff(vend.) Sub=00 Prot=00 MxPS=64 #Cfgs=  1
P:  Vendor=2fe3 ProdID=0101 Rev=00.11
C:  #Ifs= 1 Cfg#= 1 Atr=c0 MxPwr=100mA
I:  If#= 0 Alt= 0 #EPs= 1 Cls=ff(vend.) Sub=00 Prot=00 Driver=wpanusb
```

```
#!/bin/sh
PHY=`iwpan phy | grep wpan_phy | cut -d' ' -f2`
CHAN=${1:-20}
echo 'Using phy' $PHY 'channel' $CHAN
iwpan dev wpan0 set pan_id 0xabcd
iwpan dev wpan0 set short_addr 0xbeef
iwpan phy $PHY set channel 0 $CHAN
ip link add link wpan0 name lowpan0 type lowpan
ip link set wpan0 up
ip link set lowpan0 up
```

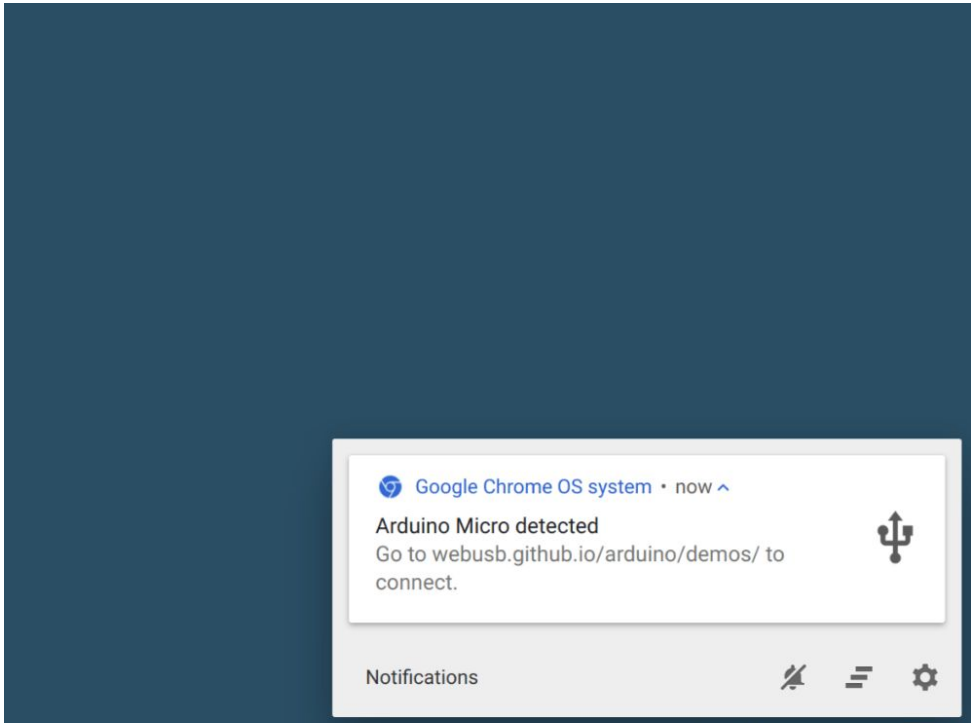
interfaces needed for IPv6 communication

```
20: wpan0: <BROADCAST,NOARP,UP,LOWER_UP> mtu 123 qdisc pfifo_fast state UNKNOWN group default qlen 300
      link/ieee802.15.4 6e:18:24:17:8c:e4:ef:88 brd ff:ff:ff:ff:ff:ff:ff:ff
21: lowpan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1280 qdisc noqueue state UNKNOWN group default qlen 1000
      link/[825] 6e:18:24:17:8c:e4:ef:88 brd ff:ff:ff:ff:ff:ff:ff:ff
      inet6 fe80::6c18:2417:8ce4:ef88/64 scope link
      valid_lft forever preferred_lft forever
```

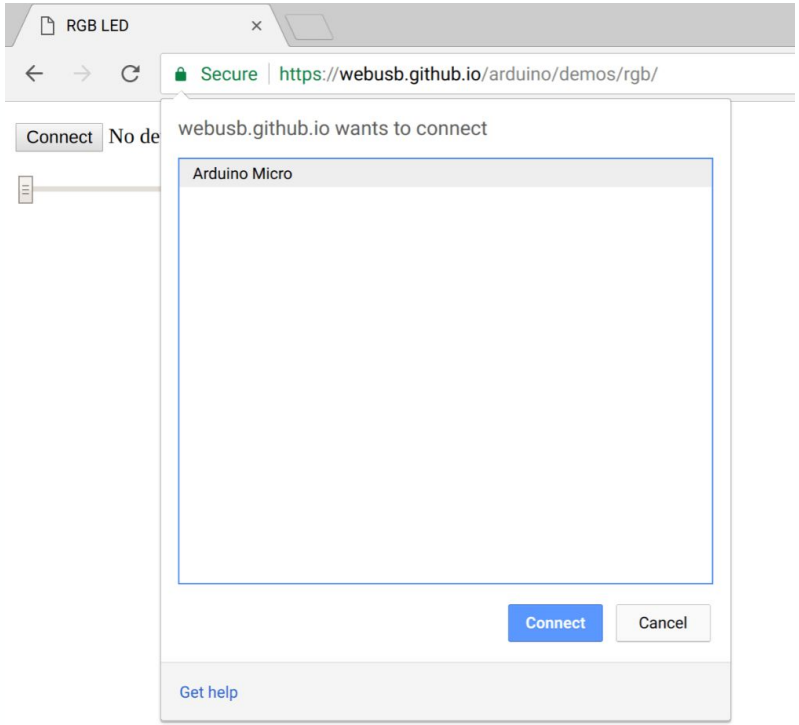
Other USB usages

- Human Interface Device (HID)
- Device Firmware Upgrade (DFU)
- Mass storage over USB
- Serial Port over USB
- 802.15.4 “serial radio” protocol
  - 802.15.4 frames over Serial USB
  - Works with Contiki native border router app https://github.com/contiki-os/contiki/tree/master/examples/ipv6/native-border-router
- Bluetooth USB adapter
  - Bluetooth HCI RAW access is used to send HCI over USB
  - Bluetooth USB Transport Layer Spec. (bluetooth.org)

WebUSB: Use case



WebUSB Notification when device is connected



WebUSB Device Chooser

WebUSB: API & Zephyr support

- Simple WebUSB API https://wicg.github.io/webusb/
- Works with recent Chrome
- USB Device direct access
- USB Device announce support by including special Descriptor
- Vendor - Specific Request specified
- Zephyr supports WebUSB

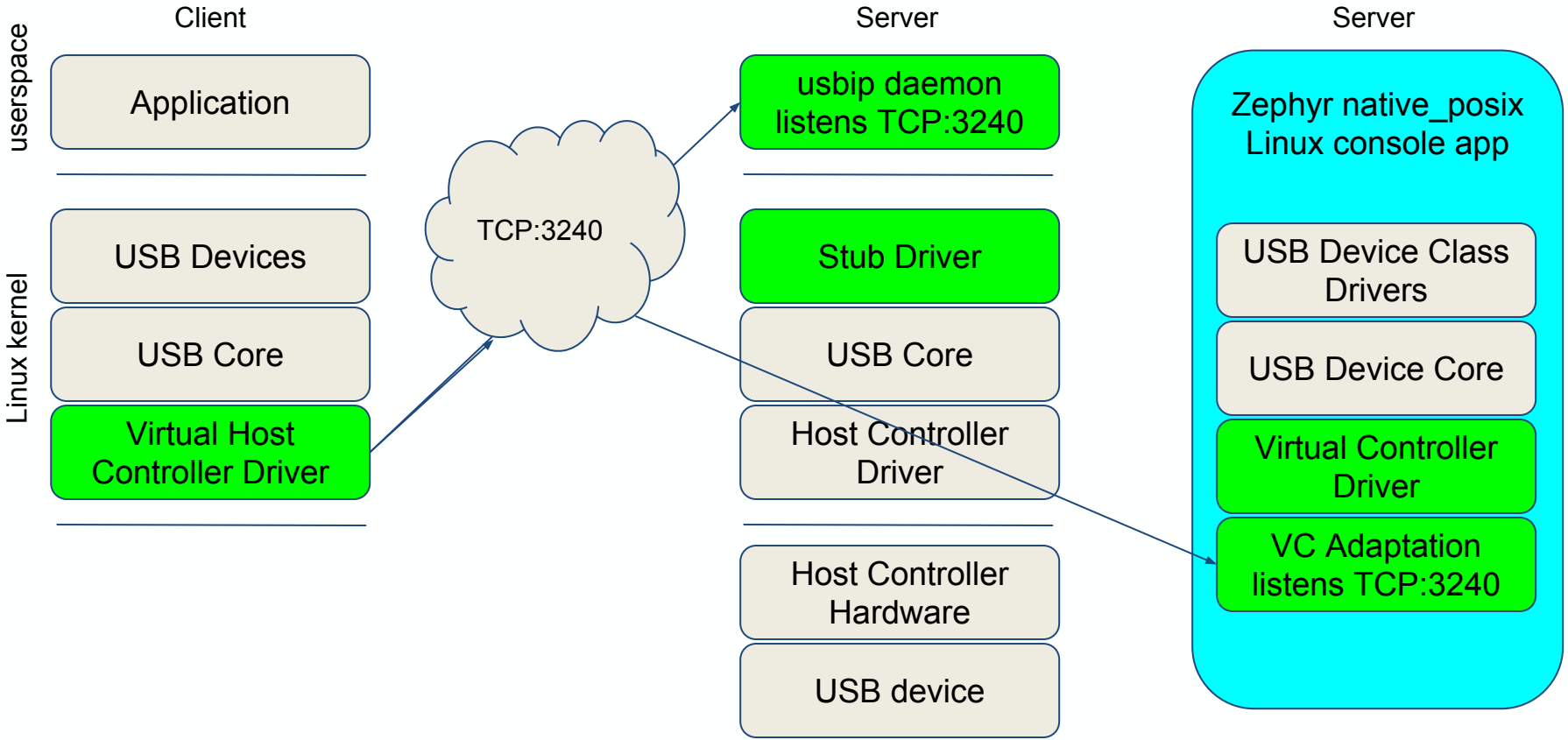
```
// Select configuration #1 for the device
device.selectConfiguration(1)

// Request exclusive control over interface #2
device.claimInterface(2)

// Waiting for 64 bytes of data from endpoint #3
device.transferIn(3, 64)

// Send Control Transfer
device.controlTransferOut({
    requestType: 'class',
    recipient: 'interface',
    request: 0x22,
    value: 0x01,
    index: 0x02})
```

Zephyr WebUSB app https://github.com/zephyrproject-rtos/zephyr/tree/master/samples/subsys/usb/webusb

Debugging Zephyr USB app without a hardware

- Zephyr app and kernel run as a native Linux console application
  - native_posix board: https://docs.zephyrproject.org/latest/boards/posix/
- Virtual USB controller over USBIP protocol
  - Zephyr is a server which decapsulates USBIP packets into USB requests handled by Zephyr
  - Linux is a client which uses VHCI driver which encapsulate USB requests to USBIP packets

Virtual USB Controller over USB/IP



Zephyr Virtual USB Controller: Attaching

```
$ sudo modprobe vhci-hcd
```

Load USB/IP Virtual Host Controller driver

```
$ lsof -i -P | grep zephyr
zephyr.ex 13633 niko 4u IPv4 32089735  0t0  TCP *:3240 (LISTEN)
```

Run Zephyr app and verify USBIP listening socket

```
$ usbip list -r localhost
Exportable USB devices
======================
 - 127.0.0.1
   1-1: unknown vendor : unknown product (2fe3:0100)
     : /sys/devices/pci0000:00/0000:00:01.2/usb1/1-1
     : (Defined at Interface level) (00/00/00)
     : 0 - Human Interface Device / No Subclass / None(03/00/00)
```

List exportable USB devices with usbip list

```
$ sudo usbip attach -r localhost -b 1-1
```

Attach exportable device to the Host

```
$ lsusb -d 2fe3:0100
Bus 007 Device 013: ID 2fe3:0100
```

Verify USB device attached to the Host

Zephyr Virtual USB Controller: Linux logs

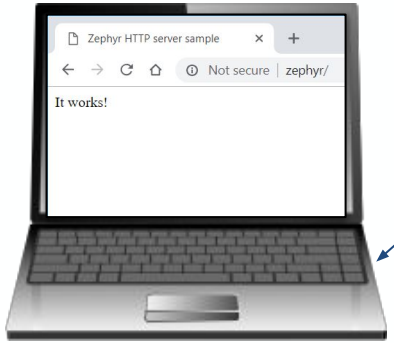
```
vhci_hcd vhci_hcd.0: USB/IP Virtual Host Controller
vhci_hcd vhci_hcd.0: new USB bus registered, assigned bus number 7
vhci_hcd: created sysfs vhci_hcd.0
usb usb7: New USB device found, idVendor=1d6b, idProduct=0002
usb usb7: New USB device strings: Mfr=3, Product=2, SerialNumber=1
usb usb7: Product: USB/IP Virtual Host Controller
usb usb7: Manufacturer: Linux 4.15.0-31-generic vhci_hcd
usb usb7: SerialNumber: vhci_hcd.0
hub 7-0:1.0: USB hub found
hub 7-0:1.0: 8 ports detected
```

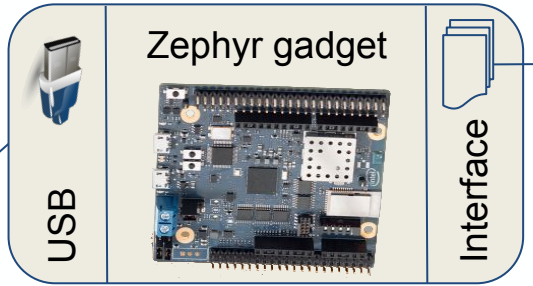
Loading Virtual Host Controller driver

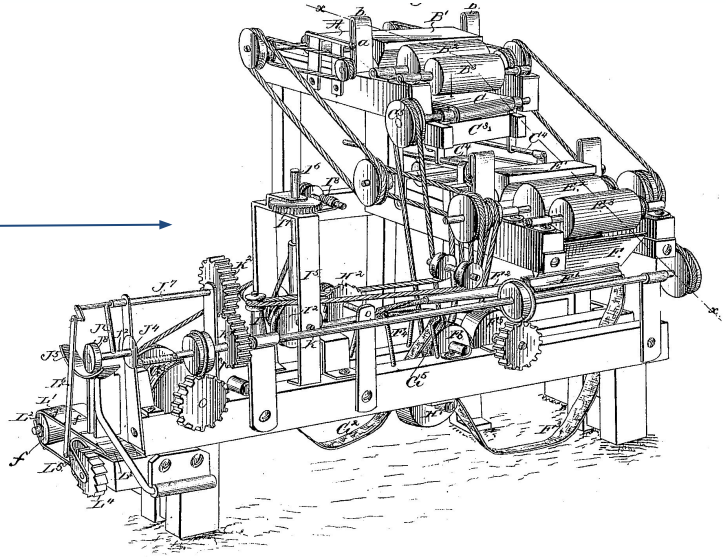
Attaching exportable device over USB/IP

```
usb 7-1: New USB device found, idVendor=2fe3, idProduct=0100
usb 7-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
usb 7-1: Product: Zephyr HID sample
usb 7-1: Manufacturer: ZEPHYR
usb 7-1: SerialNumber: 0.01
input: ZEPHYR Zephyr HID sample as
/devices/platform/vhci_hcd.0/usb7/7-1/7-1:1.0/0003:2FE3:0100.004E/input/input89
hid-generic 0003:2FE3:0100.004E: input,hidraw2: USB HID v1.10 Device [ZEPHYR Zephyr HID sample] on
usb-vhci_hcd.0-1/input0
```

Summary







Complex Hardware

References

Zephyr

- Zephyr OS https://www.zephyrproject.org/
- Native Posix board https://docs.zephyrproject.org/latest/boards/posix/
- Zephyr HTTP Server app https://docs.zephyrproject.org/latest/samples/net/http_server/README.html
- Zephyr wpanusb app https://github.com/zephyrproject-rtos/zephyr/tree/master/samples/net/wpanusb
- Zephyr webusb app https://docs.microsoft.com/en-us/windows-hardware/drivers/usbcon/microsoft-defined-usb-descriptors

Contiki

- Contiki OS http://www.contiki-os.org/
- Contiki native border router https://github.com/contiki-os/contiki/tree/master/examples/ipv6/native-border-router

USB

- How to Create and Program USB Devices
- Microsoft OS Descriptors https://docs.microsoft.com/en-us/windows-hardware/drivers/usbcon/microsoft-defined-usb-descriptors
- WebUSB API https://wicg.github.io/webusb/
- WebUSB Privacy and Security https://developers.google.com/web/updates/2016/03/access-usb-devices-on-the-web#privacy_and_security

Linux

- USB/IP project http://usbip.sourceforge.net/
- Linux IEEE 802.15.4 implementation

INTEL OPEN SOURCE TECHNOLOGY CENTER | 01.org