Pi Zero W als OTG-Gerät (Festplatte/Netzwerkkarte/Serielle Schnittstelle, …)

Links:

https://gist.github.com/gbaman/50b6cca61dd1c3f88f41

Schritt 1: „Normales“ Raspbian SD-Karten Image (Lite-Variante) auf SD-Karte installieren

```
unzip -p 2017-11-29-raspbian-stretch-lite.zip 2017-11-29-raspbian-stretch-lite.img | dd of=/dev/sdd bs=1M
```

Achtung: Auch der Raspi Zero W braucht die Option „enable_uart=1“ in der config.txt auf der 1. Partition, sonst ist die serielle Schnittstelle nicht verfügbar!

Schritt 2: Kernel-Funktion durch Overlay erweitern

In die **`config.txt`** auf der 1. Partition der SD-Karten eintragen:

`dtoverlay=dwc2`

Optional: Per

`watch lsusb`

im Terminal auf dem PC, an dem der Raspi über die aufgelöteten USB-Stecker angeschlossen ist, mitverfolgen, was am USB-Bus passiert. Abbrechen mit Steuerung-C möglich.

Test af dem Raspberry Pi nach Reboot:

`dmesg|grep dwc2`

[9.563333] dwc2 20980000.usb: EPs: 8, dedicated fifos, 4080 entries in SPRAM

[9.564273] dwc2 20980000.usb: DWC OTG Controller

[9.564348] dwc2 20980000.usb: new USB bus registered, assigned bus number 1

[9.564422] dwc2 20980000.usb: irq 33, io mem 0x00000000

[9.564838] usb usb1: Manufacturer: Linux 4.9.59+ dwc2_hsotg

→ OTG-Unterstützung ist jetzt aktiv!

Schritt 3: Gewünschte Module für OTG-Funktionen laden, diese beginnen alle mit g_ .

z.B. Laden des Ethernet-Moduls (Raspi wird zur Netzwerkkarte):

```
sudo modprobe g_ether
```

Resultat: Der PC, an dem der Raspi hängt, erhält eine Schnittstelle „usb0“, die per DHCP versucht, eine Adresse zu erhalten. Setzen von statischen IP-Adressen auf beiden Seiten erlaubt Datenübertragung.

```
Bus 002 Device 102: ID 0525:a4a2 Netchip Technology, Inc. Linux-USB Ethernet/RNDIS Gadget
```

Multifunktions-Module

Es gibt einige Gadget-Module, die zwei Funktionen auf die gleiche Schnittstelle legen, z.B. Ethernet und Serielle Schnittstelle.

```
sudo modprobe g_cdc
```

```
sudo systemctl enable getty@ttyGS0.service
```

```
sudo systemctl start getty@ttyGS0.service
```

Nun kann auf dem PC über das Gerät /dev/ttyACM0 oder putty eine Verbindung zum Raspberry Pi aufgebaut werden.

Es ist auch möglich, über das libcomposite-Modul und Dateien, die im /sys-Verzeichnis angelegt werden, weitere Geräte parallel zur Verfügung zu stellen.

Permanent eintragen

Mit Eintrag in /etc/modules können die Module schon beim Booten geladen werden.

```
echo g_cdc | sudo tee -a /etc/modules
```

Raspberry Pi Zero als USB-Festplatte nutzen

Beispiel: Image-Datei anlegen, die als „Festplatte“ exportiert wird.

```
dd if=/dev/zero of=/data.img bs=1M count=2000
```

Modul laden:

```
sudo modprobe g_mass_storage file=/data.img stall=0
```

Nun erkennt der PC den angesteckten Raspberry Pi als „unformatierte“ Festplatte, die mit `fdisk` und Formatierungsprogrammen eingerichtet werden kann.

Achtung: BEVOR ein `g_` Modul wieder mit `rmmod` entladen wird, müssen die Zugriffe auf das Gerät beendet werden, sonst hängt sich `rmmod` auf.

Mehr als 2 Geräte per Composite-Modul

Nach Laden des Kernel-Moduls „libcomposite“ lassen sich über die `/sys`-Schnittstelle Geräte „on Demand“ konfigurieren → Hier Beispiel: EIN Gerät, MEHRERE Funktionen

Das folgende Skript kann beispielsweise in `/etc/rc.local` eingetragen werden, damit es beim Systemstart ausgeführt wird.

```
# Multi-USB-Device OTG configuration
modprobe libcomposite >/dev/null 2>&1
udevadm settle -t 5

cd /sys/kernel/config/usb_gadget/
mkdir -p multigadget
cd multigadget
echo 0x04e8 > idVendor
echo 0x6848 > idProduct
echo 0x0100 > bcdDevice # v1.0.0
echo 0x0200 > bcdUSB # USB2
mkdir -p strings/0x409
echo "badc0deddeadbeef" > strings/0x409/serialnumber
echo "Knoppix Industries" > strings/0x409/manufacturer
echo "USB Everything" > strings/0x409/product
mkdir -p configs/c.1/strings/0x409
echo "0x80" > configs/c.1/bmAttributes
echo 250    > configs/c.1/MaxPower
echo "Config 1: RNDIS network, Mass Storage and Serial Device" >
configs/c.1/strings/0x409/configuration
echo "1"       > os_desc/use
echo "0xcd"    > os_desc/b_vendor_code
echo "MSFT100" > os_desc/qw_sign

# Network function (RNDIS)
mkdir -p functions/rndis.usb0
echo "42:61:64:55:53:42" > functions/rndis.usb0/dev_addr
echo "48:6f:73:74:50:42" > functions/rndis.usb0/host_addr
echo RNDIS   > functions/rndis.usb0/os_desc/interface.rndis/compatible_id
echo 5162001 > functions/rndis.usb0/os_desc/interface.rndis/sub_compatible_id
ln -s functions/rndis.usb0 configs/c.1/

# add more functions, same device
```

```
# Alternate network card (ECM, unsupported by Windows?)
mkdir -p functions/ecm.usb0
echo "42:61:64:55:53:43" > functions/ecm.usb0/dev_addr
echo "48:6f:73:74:50:43" > functions/ecm.usb0/host_addr
ln -s functions/ecm.usb0 configs/c.1/

# Serial device
mkdir -p functions/acm.gs0
# Auf dem Pi eintragen/ausführen, um das Login zu aktivieren
# sudo systemctl enable getty@ttyGS0.service
# sudo systemctl start getty@ttyGS0.service
# Device unter Linux (PC): /dev/ttyACM0
# Device auf dem Pi: /dev/ttyGS0
ln -s functions/acm.gs0 configs/c.1/

# Hard Disk
mkdir -p functions/mass_storage.usb0/lun.0
echo 0 > functions/mass_storage.usb0/stall
echo 0 > functions/mass_storage.usb0/lun.0/cdrom
echo 0 > functions/mass_storage.usb0/lun.0/ro
echo 0 > functions/mass_storage.usb0/lun.0/nofua
echo /data.img > functions/mass_storage.usb0/lun.0/file
ln -s functions/mass_storage.usb0 configs/c.1/

# *** End functions
ln -s configs/c.1 os_desc/
ls /sys/class/udc/ > UDC

# Network masquerading etc.
iptables -t nat -I POSTROUTING -j MASQUERADE
echo 1 > /proc/sys/net/ipv4/ip_forward

# Hinweis: auch dnsmasq konfigurieren, damit die USB-NW-Karten eine IP-Adresse
verteilen
```