**Making a Pi 4/Zero W to a USB Gadget**

**May 7th 2021, Dennis**

# **Basic Idea**

Pi runs a script -> Send keystroke/mouse coordinates

General idea is to have the pi run a script on boot(either through a service or /etc/rc.local)  
so every time the device is plugged in to a host computer it gets recognized immediately as a USB Composite Device.

1. **Script** is /usr/bin/**usb\_emu**
2. At /etc/rc.local put /usr/bin/usb\_emu after fi to to run on boot

The **report descriptor** ON **the script**, tells the HOST computer, how to interpret the byte packages we send from the Pi.

The script I tried : <https://www.isticktoit.net/?p=1383> (scroll down a little)

In script /usr/bin/usb\_emu

*#!/bin/bash*

*#put this script in /usr/bin/*

*#make it run on boot by adding "/usr/bin/usb\_emu" before the line exit in /etc/rc.local*

cd /sys/kernel/config/usb\_gadget/

mkdir -p upHILkeyboard  
cd upHILkeyboard  
echo 0x1d6b > idVendor *# Linux Foundation*  
echo 0x0104 > idProduct *# Multifunction Composite Gadget*  
echo 0x0100 > bcdDevice *# v1.0.0*  
echo 0x0200 > bcdUSB *# USB2*  
mkdir -p strings/0x409  
echo "12052021" > strings/0x409/serialnumber  
echo "Brainlab" > strings/0x409/manufacturer  
echo "upHIL Keyboard USB Device" > strings/0x409/product  
mkdir -p configs/c.1/strings/0x409  
echo "Config 1: ECM network" > configs/c.1/strings/0x409/configuration  
echo 250 > configs/c.1/MaxPower

*# Add functions here*

mkdir -p functions/hid.usb0

echo 1 > functions/hid.usb0/protocol  
echo 1 > functions/hid.usb0/subclass  
echo 8 > functions/hid.usb0/report\_length  
echo -ne \\x05\\x01\\x09\\x06\\xa1\\x01\\x05\\x07\\x19\\xe0\\x29\\xe7\\x15\\x00\\x25\\x01\\x75\\x01\\x95\\x08\\x81\\x02\\x95\\x01\\x75\\x08\\x81\\x03\\x95\\x05\\x75\\x01\\x05\\x08\\x19\\x01\\x29\\x05\\x91\\x02\\x95\\x01\\x75\\x03\\x91\\x03\\x95\\x06\\x75\\x08\\x15\\x00\\x25\\x65\\x05\\x07\\x19\\x00\\x29\\x65\\x81\\x00\\xc0 > functions/hid.usb0/report\_desc

ln -s functions/hid.usb0 configs/c.1/

*# End functions*

*# see gadget configurations below*

*# End functions*

ls /sys/class/udc > UDC

By having /usr/bin/usb\_emu listed on /etc/rc.local it runs right after booting by default

**However this one only emulates keyboard, the next one emulates both.**

**BUT Currently on Pi 4**

stjeong at raspvusb managed to emulate both mouse and keyboard at the same time.

Device manager lists 3 new devices(2 mouse+1 keyboard) after implementing his code.  
2 Mouse is one absolute, one relative

Following stjeong’s tutorial step by step didn’t work.

Deploying PSCP from Windows to the Pi could cause some issues when trying stjeong’s repo , using dos2unix to convert the deployed files fixed it for me.

The issue was the service create-triple-usb.service deployed couldn’t run the .sh files, on stackoverflow someone mentioned converting from dos2unix might fix it which it did.

**Or** on a fresh install of raspbian, extract the scripts from his repo from the **script** folder to /share

Create the directory /share

sudo mkdir -m 1777 /share

Clone the repo and move his scripts to /share  
Move rasp\_vusb\_server.out to /share too, that file is in bin  
Then run install\_usb.sh

sudo /share/install\_usb.sh  
sudo reboot

stjeong created a service called create-triple-usb.service to run the script on boot **(triple\_usb\_device.sh** is the script**)**

systemctl stop/start xxx.service   
systemctl status xxx.service to check logs

By removing /sys/kernel/config/usb\_gadget/g1/configs/c.1/**hid.usb0** the USB emulator effectively stops.

The report descriptor on his script (triple\_usb\_device.sh) is on next page

*#!/bin/bash*

*# From the README at https://github.com/girst/hardpass*

cd /sys/kernel/config/usb\_gadget/

mkdir -p g1

cd g1

echo 0x1d6b > idVendor *# Linux Foundation*

echo 0x0104 > idProduct *# Multifunction Composite Gadget*

echo 0x0100 > bcdDevice *# v1.0.0*

echo 0x0200 > bcdUSB *# USB2*

mkdir -p strings/0x409

echo "fedcba9876543210" > strings/0x409/serialnumber

echo "girst" > strings/0x409/manufacturer

echo "Hardpass" > strings/0x409/product

N="usb0"

mkdir -p functions/hid.usb0

echo 1 > functions/hid.usb0/protocol

echo 1 > functions/hid.usb0/subclass

echo 8 > functions/hid.usb0/report\_length

*# the report descriptor*

echo -ne \\x05\\x01\\x09\\x06\\xa1\\x01\\x85\\x01\\x05\\x07\\x19\\xe0\\x29\\xe7\\x15\\x00\\x25\\x01\\x75\\x01\\x95\\x08\\x81\\x02\\x95\\x01\\x75\\x08\\x81\\x03\\x95\\x05\\x75\\x01\\x05\\x08\\x19\\x01\\x29\\x05\\x91\\x02\\x95\\x01\\x75\\x03\\x91\\x03\\x95\\x05\\x75\\x08\\x15\\x00\\cdx25\\x65\\x05\\x07\\x19\\x00\\x29\\x65\\x81\\x00\\xc0\\x05\\x01\\x09\\x02\\xa1\\x01\\x09\\x01\\xa1\\x00\\x85\\x02\\x05\\x09\\x19\\x01\\x29\\x03\\x15\\x00\\x25\\x01\\x95\\x03\\x75\\x01\\x81\\x02\\x95\\x01\\x75\\x05\\x81\\x03\\x05\\x01\\x09\\x30\\x09\\x31\\x15\\x81\\x25\\x7f\\x75\\x08\\x95\\x02\\x81\\x06\\xc0\\xc0\\x05\\x01\\x09\\x02\\xa1\\x01\\x09\\x01\\xa1\\x00\\x85\\x03\\x05\\x01\\x09\\x30\\x09\\x31\\x15\\x00\\x26\\xff\\x7f\\x75\\x10\\x95\\x02\\x81\\x02\\x09\\x38\\x15\\x81\\x25\\x7f\\x75\\x08\\x95\\x01\\x81\\x06\\xc0\\xc0   > functions/hid.usb0/report\_desc

C=1

mkdir -p configs/c.$C/strings/0x409

echo "Config $C: ECM network" > configs/c.$C/strings/0x409/configuration

echo 250 > configs/c.$C/MaxPower

ln -s functions/hid.usb0 configs/c.$C/

ls /sys/class/udc > UDC