MuntsOS Embedded Linux

Application Note #13: Raspberry Pi USB Gadget Modes

Revision 6
4 March 2020

by Philip Munts
President, Munts Technologies
http://tech.munts.com

Introduction

This application note describes how to install and configure ***MuntsOS Embedded Linux*** for a Raspberry Pi Zero microcomputer, with *USB Gadget* support enabled. Three different USB Gadget modes are supported: *USB Network Gadget*, *USB Serial Port Gadget*, and *USB Raw HID Gadget*.

Note: The procedures in this application note will also work with the Raspberry Pi 1 or 3 Model A or A+, using a USB type A to type A crossover cable. These procedures will ***not*** *work with the Raspberry Pi 1, 2, or 3 Model B or B+.*

In *USB Network Gadget* mode, a microcomputer like the Raspberry Pi Zero configures its USB port for device (slave) operation instead of the usual host (master) operation. When plugged into a host computer, the microcomputer will present itself as a USB Ethernet device that appears to be connected to a local area network with its own DHCP server. Most host computer systems will then autoconfigure this USB network interface using DHCP. At this point the host computer can communicate with software running on the microcomputer by normal network protocols such as **`ssh`** or **`http`**.

In *USB Serial Port Gadget* mode, the Raspberry Pi Zero will present itself as a USB serial port, with the **`getty`** programming optionally running on the Raspberry Pi Zero to provide a serial port login. If a terminal program on the host computer connects to the USB serial port, the user will be presented with a login prompt for MuntsOS on the Raspberry Pi Zero.

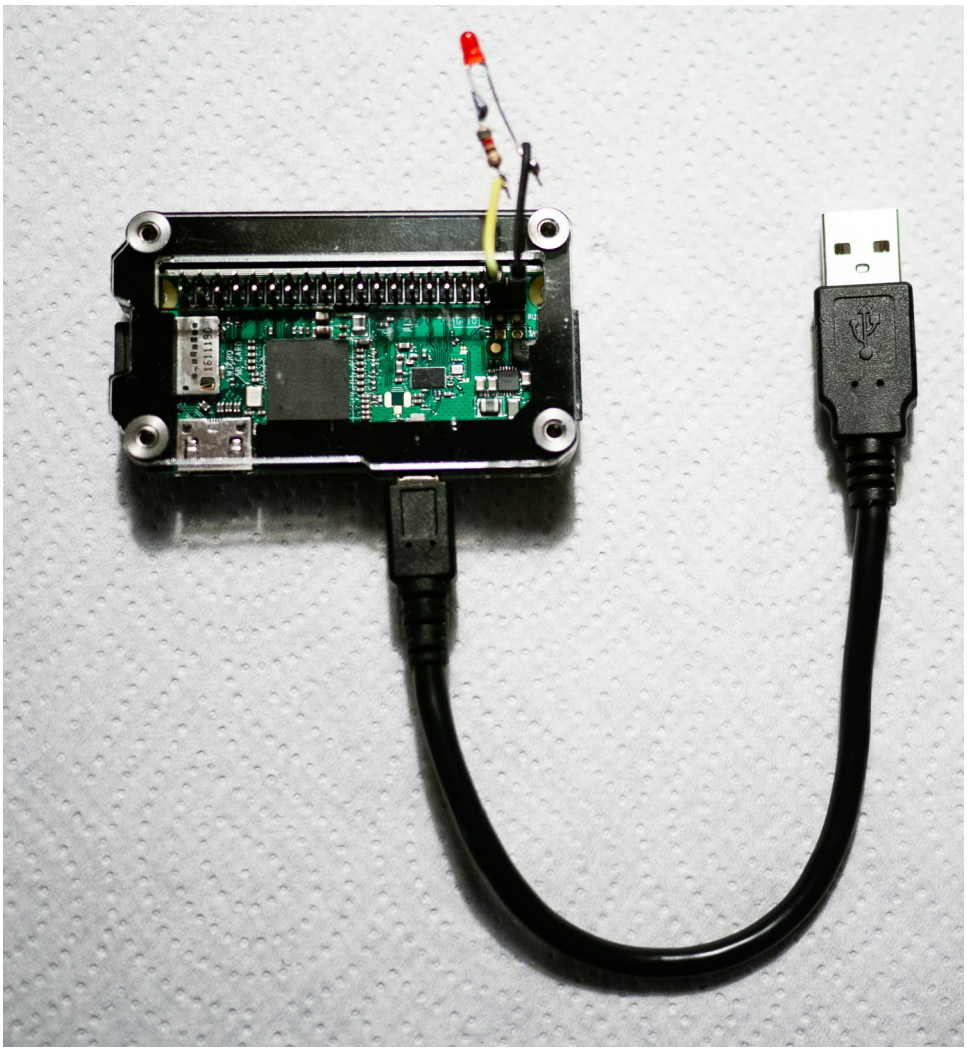
In *USB Raw HID Gadget* mode, the Raspberry Pi Zero will present itself as a raw HID device. Linux, MacOS, and Windows all automatically recognize and install device drivers for raw HID devices. Software on the host computer connects to its end of the raw HID device with libraries like **`libsimpleio`** or **`hidapi`** or **`HidSharp`**. Software on the Raspberry Pi Zero connects to its end of the raw HID device by opening **`/dev/hidg0`** for read and write access, to receive and send 64-byte reports (binary messages) from and to the host computer.

The Raspberry Pi Zero, A, and A+ can generally obtain enough power from the host computer via the USB cable. The Raspberry Pi 3 Model A+ requires significantly more power and may require a separate power supply, especially if the wireless network interface will be used.

Prerequisites

Linux, MacOS, or Windows host computer with a micro-SD card slot, reader and/or adapter.

Test Platform Hardware



The test platform for the purposes of this application node consists of a Raspberry Pi Zero with a micro-USB cable plugged into the socket labeled **`USB`**, not **`PWR IN`**.

Installation Procedure for USB Network Gadget Mode

Step 1: Partition and format the micro-SD card in the host computer.

The exact procedure for this varies widely among Linux, MacOS, and Windows systems. Consult your system documentation for details.

The micro-SD card must be formatted with a single FAT32 partition that is marked bootable. The FAT32 partition must be formatted as a DOS FAT32 file system. It is worthwhile to label the DOS file system partition with the volume label **`USBGADGET`**.

Step 2: Mount the micro-SD card file system on the host computer.

The exact procedure for this varies widely among Linux, MacOS, and Windows systems. Consult your system documentation for details. MacOS, Windows, and most modern Linux distributions will auto-mount a formatted SD card.

Step 3: Download the ***MuntsOS*** Thin Server from the following URL:

`http://repo.munts.com/muntsos/thinservers/muntsos-RaspberryPiGadget.zip`

Step 4: Unpack the **`.zip`** file to the micro-SD card:

Linux and MacOS -- Use a command similar to the following:

```
unzip muntsos-RaspberryPiGadget.zip -d /media/pmunts/USBGADGET
```

Windows -- If a command-line **`unzip`** program has been installed, a command like this:

```
unzip muntsos-gpio-server-RaspberryPiGadget.zip -d E:\
```

Or, from the Windows FIle Explorer, double click on the **`.zip`** file in one window, and double click on the SD card drive in another, and just drag everything from the **`.zip`** file to the SD card.

Step 5: Umount and/or eject the SD card, remove it from the host computer and insert it into the Raspberry Pi Zero.

Step 6: Connect the Raspberry Pi Zero to the host computer with a micro-USB cable.

After a short time the host computer operating system should configure a new Ethernet network interface. The IP address of the new network interface should be **`10.254.254.253`** and the IP address of the Raspberry Pi Zero should be **`10.254.254.252`**. The domain name **`usbgadget.munts.net`** resolves to **`10.254.254.252`** and can be used to reference the Raspberry Pi Zero if desired.

Step 7: Log in to the Raspberry Pi Zero using SSH (via the **`ssh`** command or Putty application).

From the command line:

```
ssh root@usbgadget.munts.net
```

The default user and password are **`root`** and **`default`**.

Installation Procedure for USB Serial Port Gadget Mode

Steps 1 through 4: Same as the previous section.

Step 5: Edit **`cmdline.txt`** to enable USB Serial Port Gadget mode:

With your favorite text editor, open **`cmdline.txt`** on the SD card and change the value at the end of the line to **`0x72D`**. The value **`0x72D`** enables USB Serial Port Gadget mode and runs the **`getty`** login server on **`/dev/ttyGS0`**, the USB Serial Port Gadget device created on the Raspberry Pi Zero. Save the file and exit the editor.

Step 6: Umount and/or eject the SD card, remove it from the host computer and insert it into the Raspberry Pi Zero.

Step 7: Connect the Raspberry Pi Zero to the host computer with a micro-USB cable.

After a short time the host computer operating system should configure a new USB serial port device.

Step 9: After you have figured out what serial port was created (it will be **`/dev/ttyACMx`** on a Linux host computer), run your favorite terminal emulator and log in to the Raspberry Pi Zero. The default user and password are **`root`** and **`default`**.

Makefile for USB Raw HID Gadget Test Program

Available for download at: http://git.munts.com/muntsos/doc/.rawhid/Makefile

```
CC          = $(CROSS_COMPILE)gcc
CFLAGS            += -Wall $(DEBUGFLAGS) $(EXTRAFLAGS)
STRIP       = $(CROSS_COMPILE)strip

# Cygwin

ifeq ($(findstring CYGWIN, $(shell uname)), CYGWIN)
LDFLAGS           += -lhidapi
endif

# Linux

ifeq ($(shell uname), Linux)
LDFLAGS           += -lhidapi-hidraw
endif

# MacOS X

ifeq ($(shell uname), Darwin)
CFLAGS            += -I/usr/local/include
LDFLAGS           += -lhidapi
endif

# Define a pattern rule to compile a C program

%: %.c
      $(CC) $(CFLAGS) -o $@ $< $(LDFLAGS)
      $(STRIP) $@

# Compile the test program

default: test_version

# Remove working files

clean:
      rm -rf test_version
```

Source Code for USB Raw HID Gadget Mode Test Program

Available for download at: http://git.munts.com/muntsos/doc/.rawhid/test_version.c

```
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <hidapi/hidapi.h>

int main(void)
{
  hid_device *handle;
  int len;
  uint8_t outbuf[65];
  uint8_t inbuf[65];

  // Initialize HID API library

  if (hid_init())
  {
    fprintf(stderr, "ERROR: hid_init() failed\n");
    exit(0);
  }

  // Open raw HID device

  handle = hid_open(0x16D0, 0x0AFA, NULL);

  if (handle == NULL)
  {
    fprintf(stderr, "ERROR: hid_open() failed\n");
    exit(1);
  }

  // Send version string request

  memset(outbuf, 0, sizeof(outbuf));

  // Byte 0 is the Report ID so the actual request begins at byte 1

  outbuf[1] = 2;

  len = hid_write(handle, outbuf, sizeof(outbuf));

  if (len != sizeof(outbuf))
  {
    fprintf(stderr, "ERROR: hid_write() failed\n");
    exit(1);
  }

  // Receive version string response

  memset(inbuf, 0, sizeof(inbuf));
```

```
  len = hid_read_timeout(handle, inbuf, sizeof(inbuf), 1000);

  if (len == 64)       // No Report ID byte
    printf("\nRemote I/O Device Info: %s\n\n", inbuf+3);
  else if (len == 65)  // Skip Report ID byte
    printf("\nRemote I/O Device Info: %s\n\n", inbuf+4);
  else
  {
    fprintf(stderr, "ERROR: hid_read_timeout() failed\n\n");
    exit(1);
  }

  // Close the raw HID device

  hid_close(handle);
}
```

Installation Procedure for USB Raw HID Gadget Mode

Steps 1 and 2: Same as the previous section.

Step 3: Download the ***MuntsOS*** GPIO Thin Server for Raspberry P i boards from the following URL:

http://repo.munts.com/muntsos/thinservers/muntsos-gpio-server-RaspberryPiGadget.zip

Step 4: Same as the previous section.

Step 5: Edit **cmdline.txt** to enable USB Raw HID Gadget mode:

With your favorite text editor, open **cmdline.txt** on the SD card and change the value at the end of the line to **0x172C**. The value **0x172C** enables USB Raw HID Gadget mode. Save the file and exit the editor.

Step 6: Unmount and/or eject the SD card, remove it from the host computer and insert it into the Raspberry Pi Zero.

Step 7: Connect the Raspberry Pi Zero to the host computer with a micro-USB cable.

After a short time the host computer operating system should configure a new USB raw HID device. On the Raspberry Pi Zero, the device **/dev/hidg0** will be created. It is the Raspberry Pi Zero's end of the raw HID device. On a Linux host computer, a device **/dev/hidrawN** will be created. Other kinds of host computers will vary widely.

One of the server programs included in the ***MuntsOS*** GPIO Thin Server provides Remote I/O Protocol (http://git.munts.com/libsimpleio/doc/RemoteIOProtocol.pdf) services over USB raw HID. We will now build and run a test program on the host computer that will issue a Remote I/O Protocol command to and receive a response from the Raspberry Pi Zero.

Step 8: Download the test program and its **Makefile** from http://git.munts.com/muntsos/doc/.rawhid/test_version.c and http://git.munts.com/muntsos/doc/.rawhid/Makefile.

Step 9: Compile the test program with **make**. This should work on Linux, MacOS and Cygwin, provided that the **hidapi** library has been installed.

```
cd <download directory>
make
```

Step 10: Run the test program:

```
./test_version
```

You should see output similar to the following:

```
Remote I/O Device Info: MuntsOS Remote I/O Server
```