

UPS-Lite V1.2 使用说明

for Raspberry Pi Zero

-- by XiaoJ



1. 简介

UPS-Lite 是一款专门为树莓派 Zero (以下简称 pi) 所设计的 UPS 电源, 采用一块 1000mAh 的聚合物锂电池进行供电, 支持外部电源插入检测, 支持边充边放, 既插上外部电源时, pi 由外部电源供电, 拔掉外部电源时, pi 无缝切换为锂电池供电。UPS-Lite 通过 5 根弹性顶针与 pi 主板连接, pi 的供电以及电量读取功能都是通过顶针来完成。另外 UPS-Lite 还集成了专业电量计芯片 MAX17040G、双色充电状态指示灯。



参数：

充电电流：**max 400mA@5V**

输出电流：**max 1.3A@5V** (在没有插入外部电源，仅用电池供电的情况下)

max 2A@5V (在插入外部电源的情况下)

电量测量：输出电池电量百分比，误差 $\pm 2\%$ ，电压测量误差在 $\pm 3\text{mV}$

2.安装

将 pi 的四个螺丝孔对准四个螺丝放入，锁上配套的螺母即可。注意，pi 需要焊好排针才能使用 UPS-Lite。因为 pi 与 UPS-Lite 的连接是通过顶针与排针的接触来实现的。



3.功能使用

3.1 充电与电源插入检测功能

建议用功率在 5V2A 及以上电源适配器给 UPS-Lite 充电。因为当锂电池电量较低时，外部电源适配器不仅要给 pi 供电，还需要提供部分电流供锂电池充电，充电过程中，充电状态指示灯为红色，表示电池正在充电。当锂电池充满电时，充电状态指示灯会变成绿色，表示电池已充满电。另外 UPS-Lite 带有电源适配器插入检测功能，可以提高 GPIO 口的高低电平来判断外部电源是否插入，当插入电源时 pi 的 io4(BCM 编号)会检测到高电平，拔出时为低电平，使能该功能需要短接 UPS 背面的两个焊盘，详细见下图。



3.2 电池计量功能

用 nano 编辑器新建一个名字为 UPS_Lite.py 的脚本程序 ,代码如下(详细代码见附录)。这个脚本是利用 Python 的 smbus 库对 MAX17040G 进行 i2c 读操作。MAX17040G 设备地址为 0x36 , 寄存器 VCELL 为 12bit 电池电压 ADC 测量值 , 地址为 02h-03h , ADC 测量精度单位为 1.25mV。寄存器 SOC 为 16bit 电池容量百分比读数 , 地址为 04h-05h , SOC 的高 8bit 单位为电池容量的 1% , 低 8bit 单位为 1/256% , 提供电池容量百分比小数位的读数。保存 UPS_Lite.py 脚本程序到一个自己知道的目录下(如以下保存到 /home/pi/ 目录下) , 然后用 Python 运行该程序 , 可以看到程序每隔两秒就会输出当前电池的电压值以及电池容量百分比。另外由于 MAX17040G 电池容量的计算方式 , 当电池容量读数为 1% 时 , 此时电池电压的读数约为 3.5V。由于锂电池一般电压为 3.5V 时 , 对应的电池容量已经很低了 , 过度放电会损坏电池 , 所以用户后续编写低电量自动关机程序时 , 建议电池容量为 1% 时就自动关闭 pi , 如果继续运行的话 , 当电池电压下降到 2.7V 时 , UPS-Lite 会自动停止供电。具体操作步骤见下文

MAX17040G 寄存器地址以及功能介绍

ADDRESS (HEX)	REGISTER	DESCRIPTION	READ/ WRITE	DEFAULT (HEX)
02h–03h	VCELL	Reports 12-bit A/D measurement of battery voltage.	R	—
04h–05h	SOC	Reports 16-bit SOC result calculated by ModelGauge algorithm.	R	—
06h–07h	MODE	Sends special commands to the IC.	W	—
08h–09h	VERSION	Returns IC version.	R	—
0Ch–0Dh	RCOMP	Battery compensation. Adjusts IC performance based on application conditions.	R/W	9700h
FEh–FFh	COMMAND	Sends special commands to the IC.	W	—

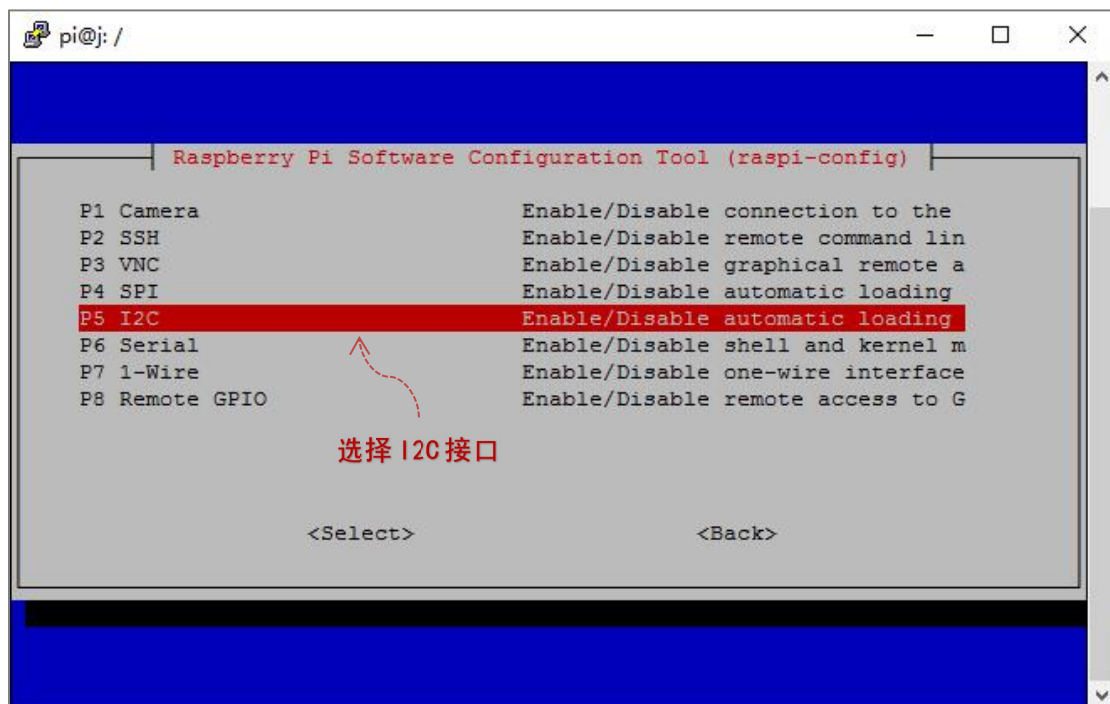
VCELL 寄存器

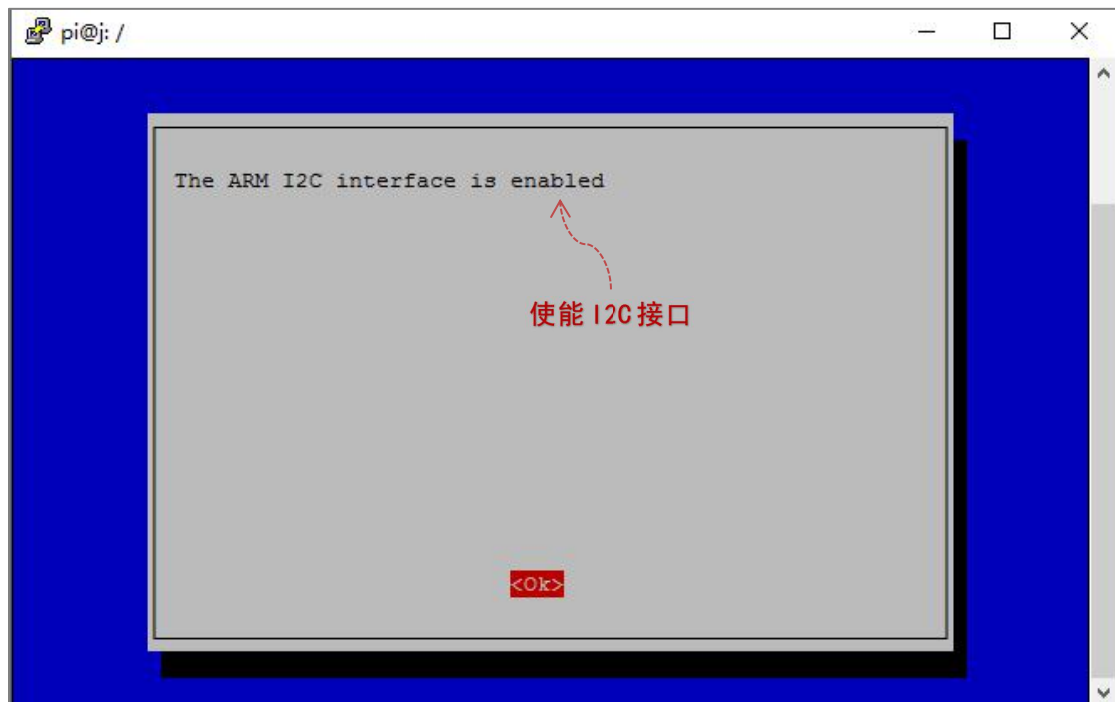
MSB—ADDRESS 02h								LSB—ADDRESS 03h							
2 ¹¹	2 ¹⁰	2 ⁹	2 ⁸	2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	0	0	0	0
MSB								LSB							
0: BITS ALWAYS READ LOGIC 0								UNITS: 1.25mV FOR MAX17040 2.50mV FOR MAX17041							

SOC 寄存器

MSB—ADDRESS 04h								LSB—ADDRESS 05h							
2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	2 ⁻¹	2 ⁻²	2 ⁻³	2 ⁻⁴	2 ⁻⁵	2 ⁻⁶	2 ⁻⁷	2 ⁻⁸
MSB								LSB							
								UNITS: 1.0%							

a. 打开 pi 配置工具 raspi-config , 使能 I2C 接口





b. 安装 i2c-tools 和 python-smbus , 安装完成后重启一下 pi

```
pi@j: /  
pi@j:~$ sudo raspi-config  
pi@j:~$ sudo apt-get install i2c-tools python-smbus  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
i2c-tools is already the newest version.  
python-smbus is already the newest version.  
0 upgraded, 0 newly installed, 0 to remove and 57 not upgraded.  
pi@j:~$ sudo reboot
```

安装软件

重启 pi

c. 运行 `sudo i2cdetect -l` 查看当前 pi 是采用哪个 i2c 总线。

```
pi@j: /  
pi@j:~$ sudo i2cdetect -l  
i2c-1  i2c          bcm2835 I2C adapter          I2C adapter  
pi@j:~$
```

运行 `i2cdetect -l` 查看 i2c 总

确定系统 i2c 总线为

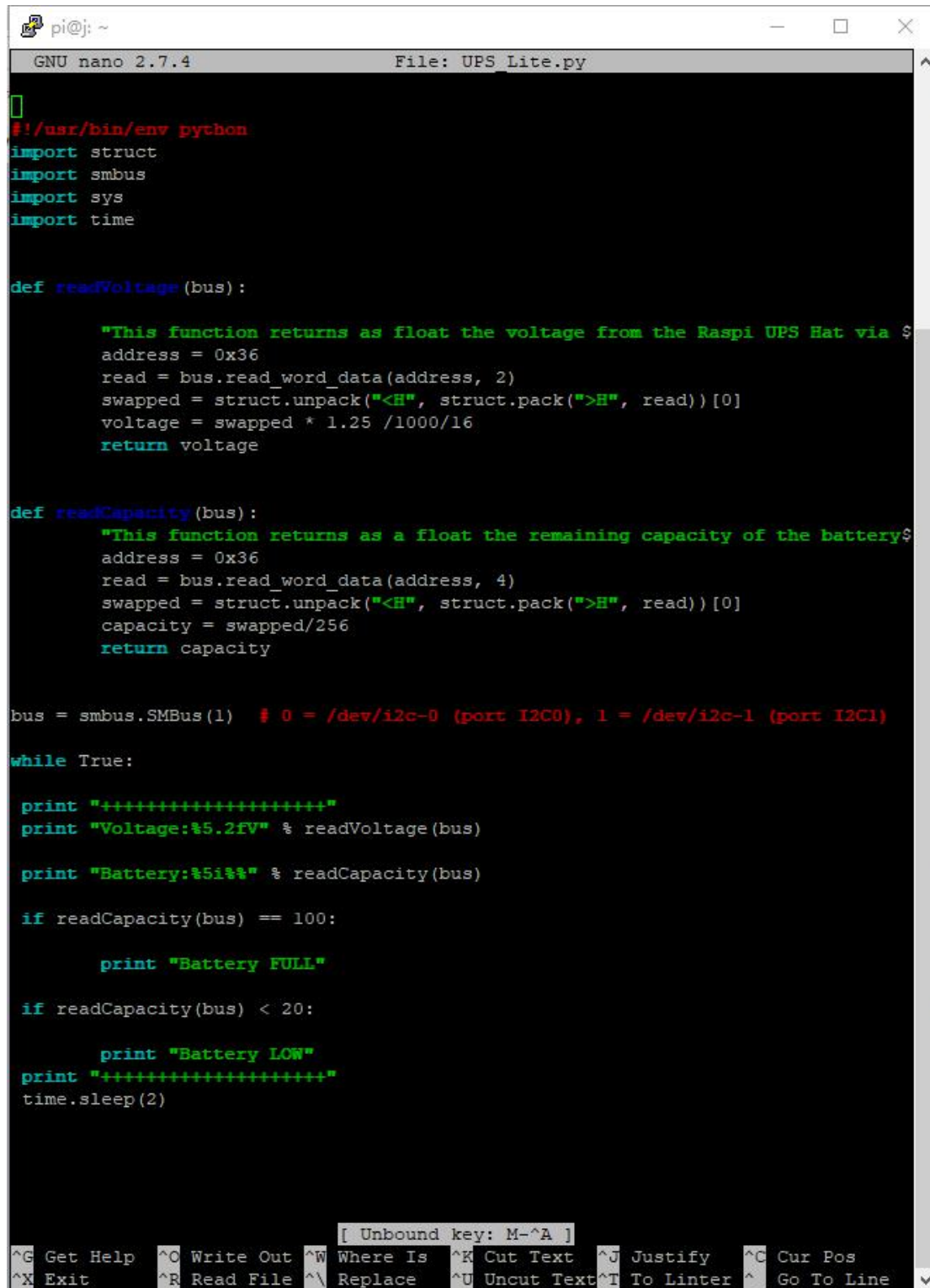
d. 运行 `sudo i2cdetect -y 1` 查看当前 pi 的 i2c 总线上挂载的设备。

```
pi@j: /  
pi@j:~$ sudo i2cdetect -l  
i2c-1  i2c          bcm2835 I2C adapter          I2C adapter  
pi@j:~$ sudo i2cdetect -y 1  
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  
00: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
30: -- -- -- -- -- 36 -- -- -- -- -- -- -- -- --  
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
60: -- -- -- -- -- -- 68 -- -- -- -- -- -- -- -- --  
70: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  
pi@j:~$
```

运行 `i2cdetect -l` 查看 i2c 总

地址 0x36 为 MAX17040G

e. 用 nano 编辑器新建以下 UPS_Lite.py 脚本程序（详细代码见附录）



```
pi@j: ~
GNU nano 2.7.4 File: UPS_Lite.py

#!/usr/bin/env python
import struct
import smbus
import sys
import time

def readVoltage(bus):

    "This function returns as float the voltage from the Raspi UPS Hat via $
    address = 0x36
    read = bus.read_word_data(address, 2)
    swapped = struct.unpack("<H", struct.pack(">H", read))[0]
    voltage = swapped * 1.25 / 1000 / 16
    return voltage

def readCapacity(bus):

    "This function returns as a float the remaining capacity of the battery$
    address = 0x36
    read = bus.read_word_data(address, 4)
    swapped = struct.unpack("<H", struct.pack(">H", read))[0]
    capacity = swapped / 256
    return capacity

bus = smbus.SMBus(1) # 0 = /dev/i2c-0 (port I2C0), 1 = /dev/i2c-1 (port I2C1)

while True:

    print "+++++"
    print "Voltage:%5.2fV" % readVoltage(bus)

    print "Battery:%5i%%" % readCapacity(bus)

    if readCapacity(bus) == 100:

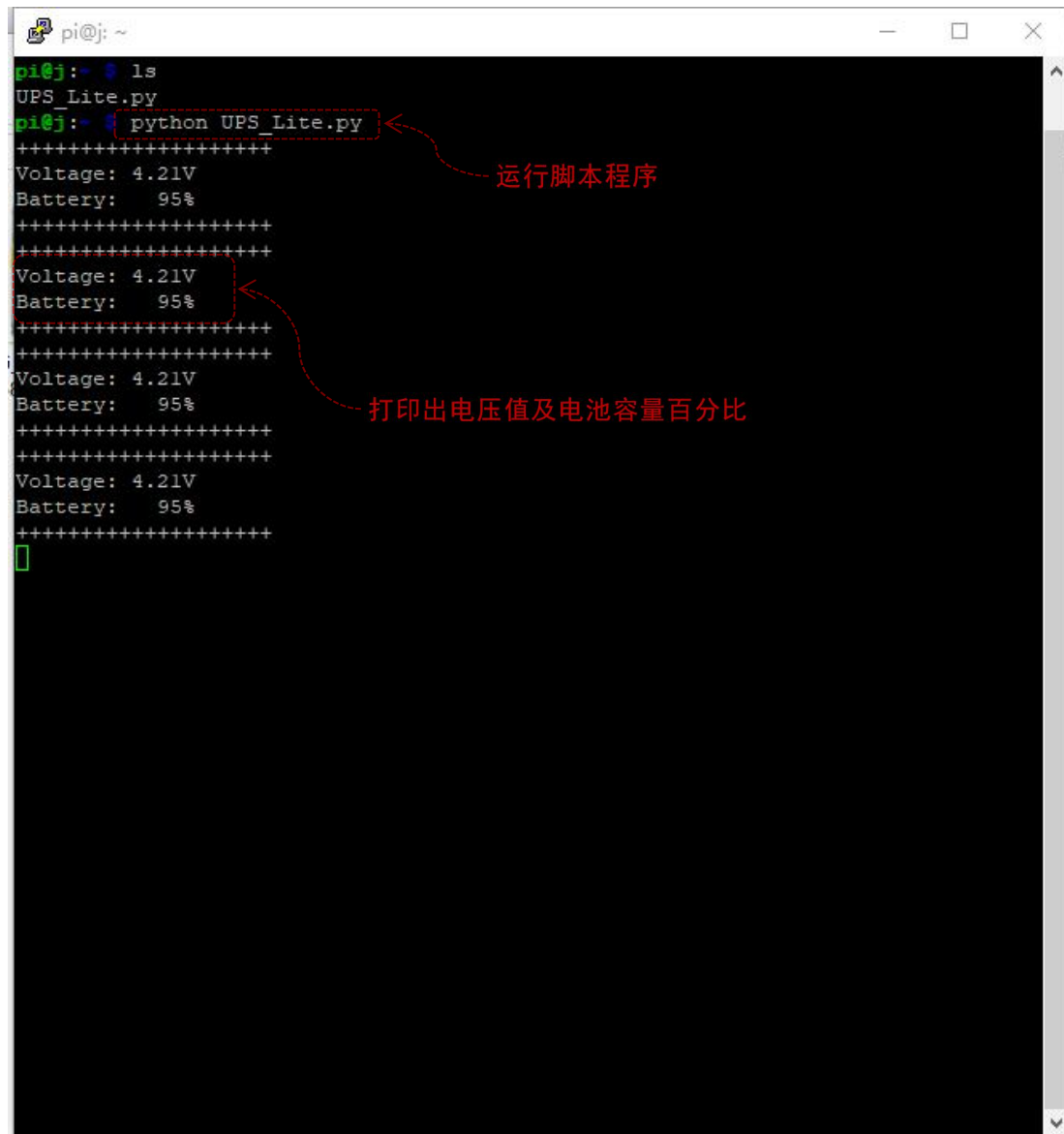
        print "Battery FULL"

    if readCapacity(bus) < 20:

        print "Battery LOW"
    print "+++++"
    time.sleep(2)

[ Unbound key: M-^A ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Linter ^_ Go To Line
```

f.用 Python 运行该脚本程序



```
pi@j: ~  
pi@j:~$ ls  
UPS_Lite.py  
pi@j:~$ python UPS_Lite.py  
+++++  
Voltage: 4.21V  
Battery: 95%  
+++++  
+++++  
Voltage: 4.21V  
Battery: 95%  
+++++  
+++++  
Voltage: 4.21V  
Battery: 95%  
+++++  
+++++  
Voltage: 4.21V  
Battery: 95%  
+++++  
[
```

运行脚本程序

打印出电压值及电池容量百分比

附录：

UPS_Lite.py 的脚本程序代码：

```
#!/usr/bin/env python
import struct
import smbus
import sys
import time

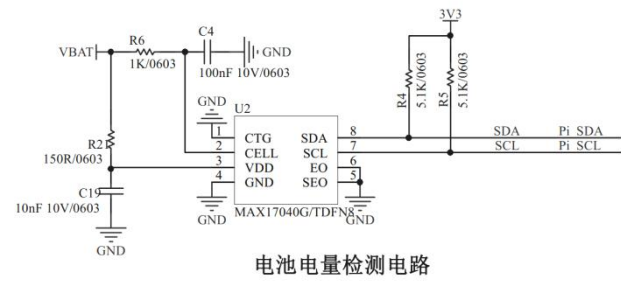
def readVoltage(bus):
    """This function returns as float the voltage from the Raspi UPS Hat via the provided SMBus object"""
    address = 0x36
    read = bus.read_word_data(address, 2)
    swapped = struct.unpack("<H", struct.pack(">H", read))[0]
    voltage = swapped * 1.25 / 1000 / 16
    return voltage

def readCapacity(bus):
    """This function returns as a float the remaining capacity of the battery connected to the Raspi UPS
    Hat via the provided SMBus object"""
    address = 0x36
    read = bus.read_word_data(address, 4)
    swapped = struct.unpack("<H", struct.pack(">H", read))[0]
    capacity = swapped / 256
    return capacity

bus = smbus.SMBus(1) # 0 = /dev/i2c-0 (port I2C0), 1 = /dev/i2c-1 (port I2C1)

while True:
    print "+++++"
    print "Voltage:%5.2fV" % readVoltage(bus)
    print "Battery:%5i%" % readCapacity(bus)
    if readCapacity(bus) == 100:
        print "Battery FULL"
    if readCapacity(bus) < 20:
        print "Battery LOW"
    print "+++++"
    time.sleep(2)
```

部分参考原理图：



参考资料：

树莓派学习笔记——I2C Tools 学习笔记 - CSDN 博客

<http://blog.csdn.net/xukai871105/article/details/15029843>

MAX17040 结构紧凑的低成本 1S/2S 电量计 - Maxim 美信

<https://www.maximintegrated.com/cn/products/power/battery-management/MAX17040.html>

CP2104 驱动下载 USB to UART Bridge VCP Drivers | Silicon Labs

<https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers>

联系我：



微信扫一扫

感谢