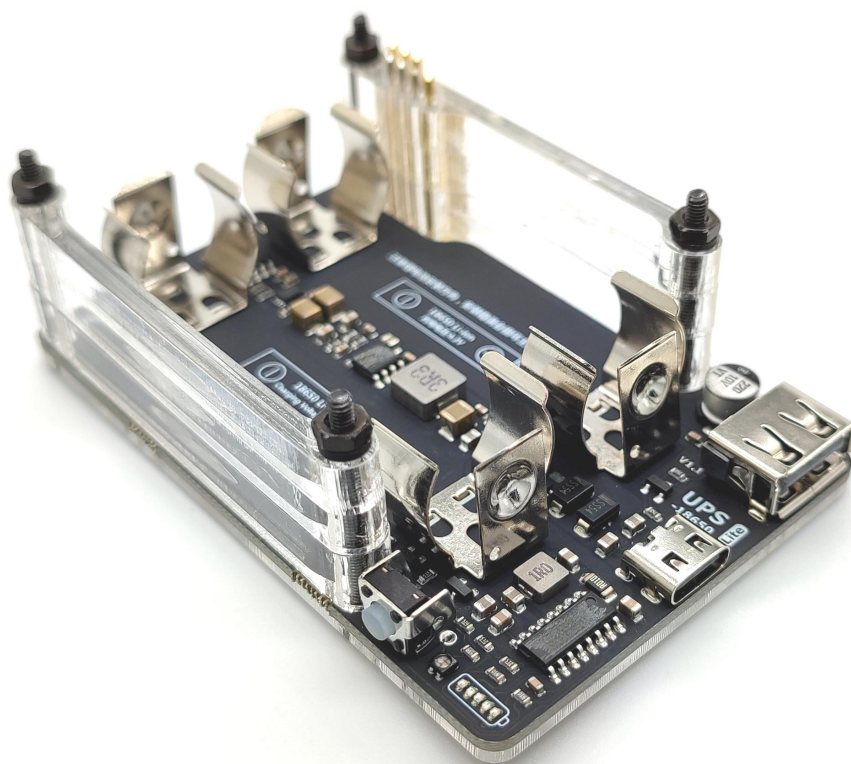


UPS-18650-Lite 使用说明

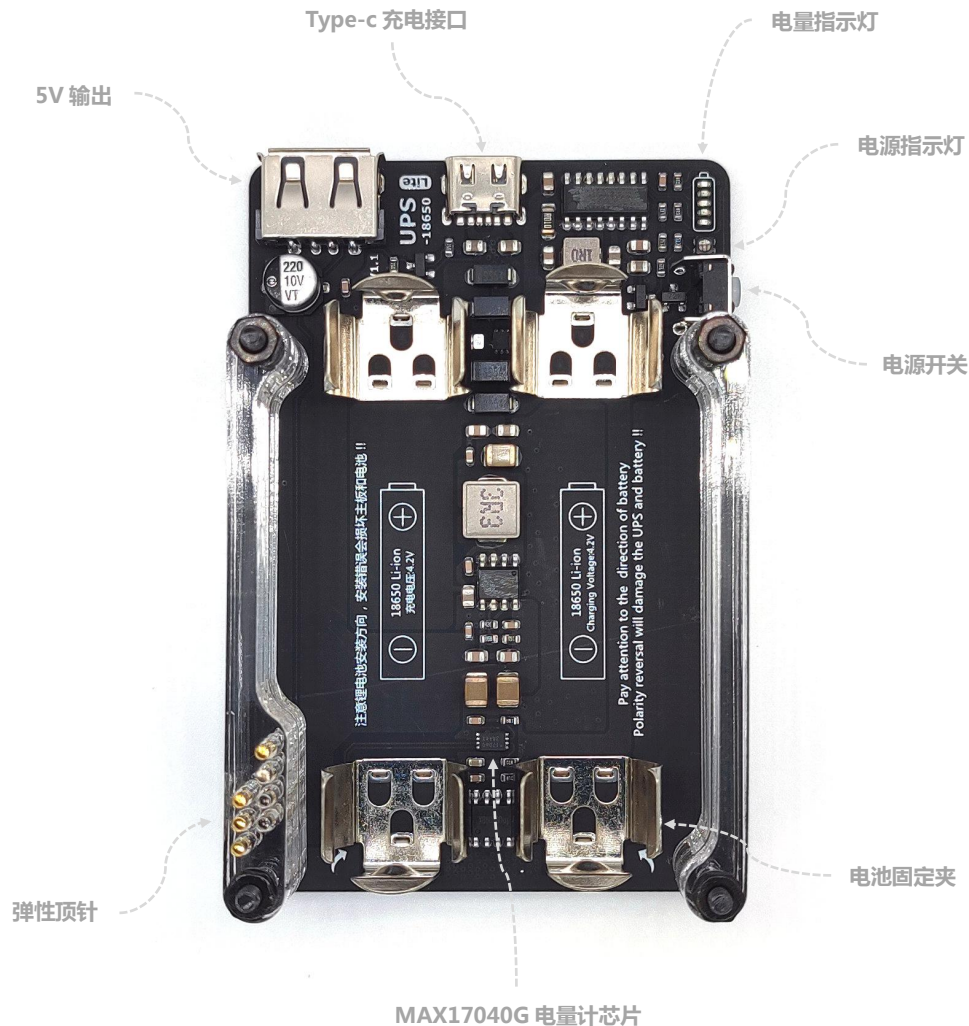
for Raspberry Pi 4B/3B+/3B

-- by XiaoJ



1. 简介

UPS-18650-Lite 是一款专门为树莓派 B 型系列主板(既 4B/3B+/3B 等 ,以下简称 pi) 所设计的 UPS 电源 ,采用两节标准的平头 18650 锂电池进行供电 ,支持过充、过放、过流、短路保护 ,支持外部电源插入检测 ,支持边充电边放 ,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-18650-Lite 通过 7 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 , pi 不需要再插任何电源线 ,只要把外部电源线插在 UPS 的 type-c 口即可完成供电和充电。另外 UPS-18650-Lite 集成了专业电量计芯片 MAX17040G ,可用于对 UPS 电池电压及电池剩余电量进行测量。



参数：

充电电流：**1A@5V**

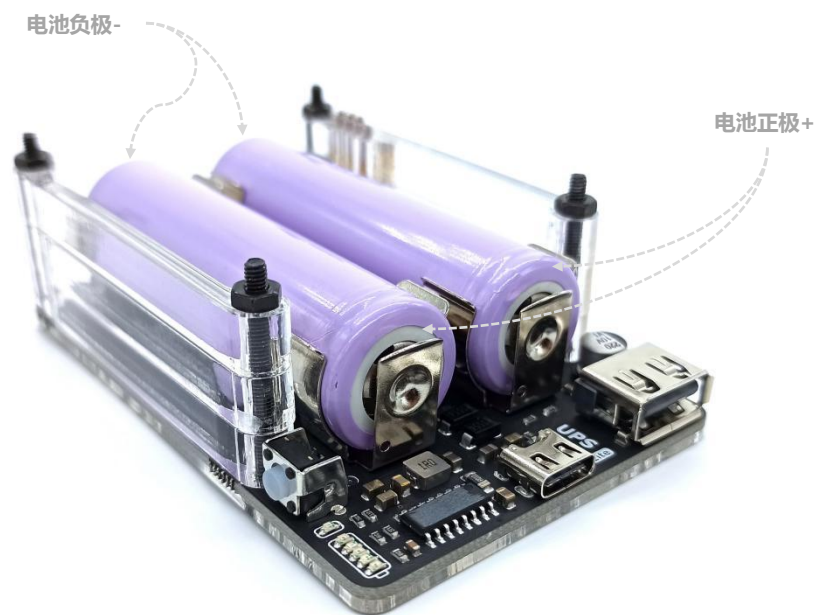
输出电流：瞬间放电 **3.5A@5.1V**，持续放电 **2.5A@5.1V**

(PS:请使用质量好的电池以及电源适配器，以确保 UPS 正常工作)

电量测量：电池剩余电量 SOC 百分比，误差 $\pm 2\%$ ，电压测量误差在 $\pm 3\text{mV}$ ，测量误差因电池个体、环境温度而异。

2.安装

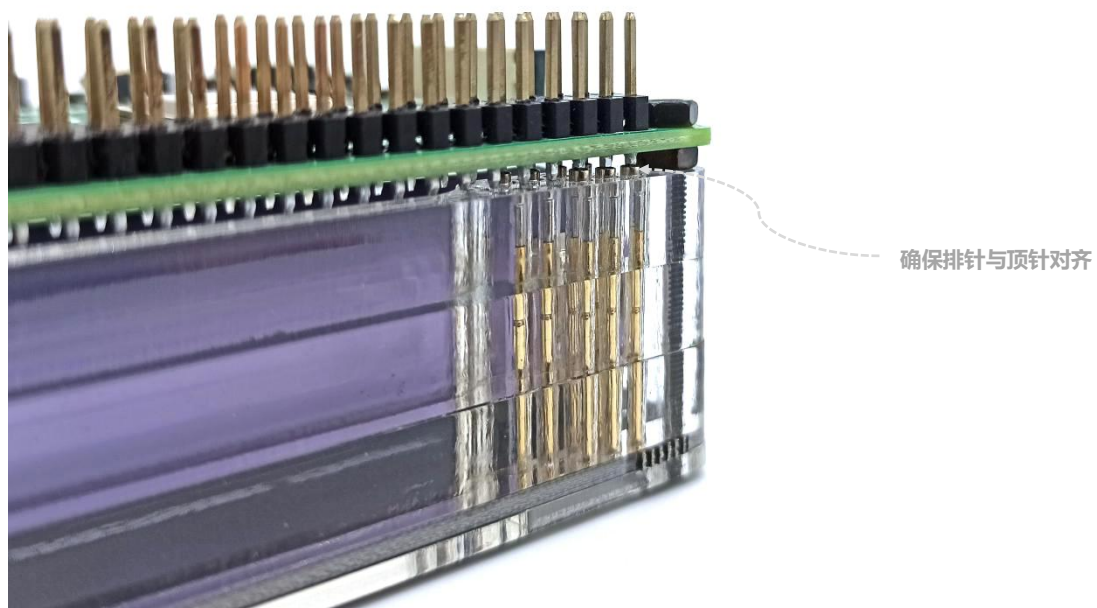
a. 请按照板子上的电池标识方向装好两节 18650 锂电池,安装时要注意电池方向 ,安装错误会烧毁板子以及电池！！！（备注：两节锂电池必须是同个型号且内阻及容量相差不多的标准平头 18650 电池，不可混用两节不同参数的锂电池，更不能使用带保护板的锂电池）。



备注：电池正负极识别方法



b. 将 pi 的四个固定孔对准 UPS 的四个螺丝放入，使 pi 的排针与弹性顶针对准后，锁上配套的螺母即可！！





c. 电池拆卸更换，请用镊子按照板上箭头指示的方向插入后往上撬出电池

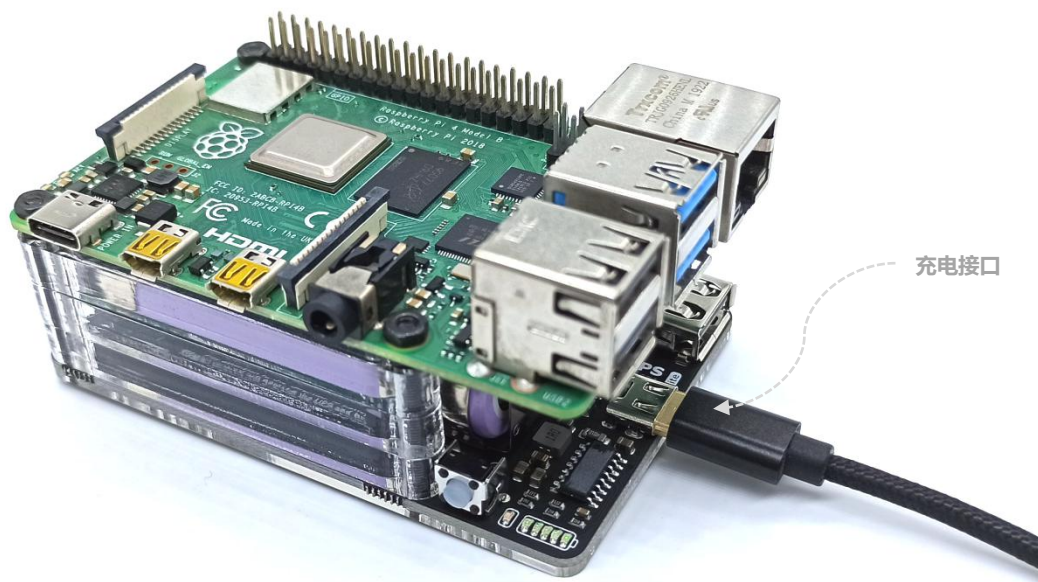


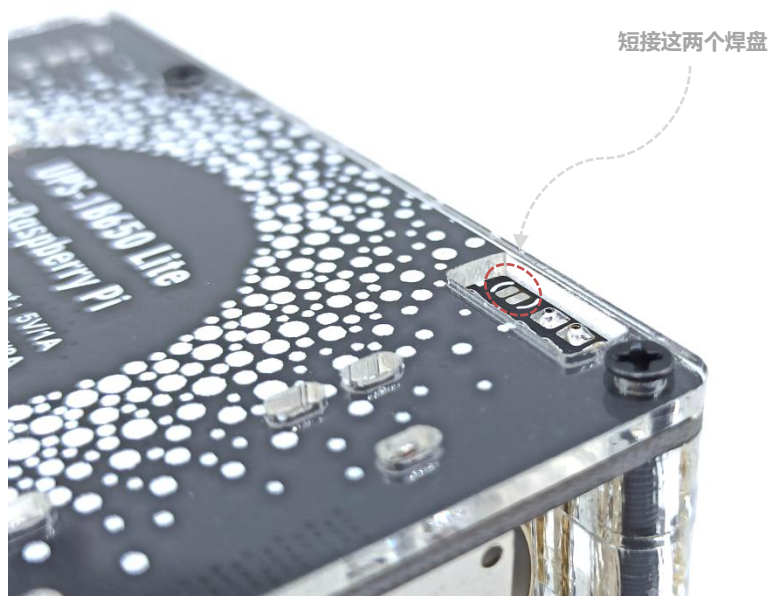
3.功能使用

3.1 充电与电源插入检测功能

建议使用功率在 5V3A 及以上电源适配器给 UPS-18650-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，电量指示灯会相继闪烁，当锂电池充满电时，5 个绿电量指示灯会全亮，表示电池已经充满电。5 个电量指示灯表示的电池容量依次为 20%、40%、60%、80%、100%。

UPS-18650-Lite 带有外部电源适配器插入检测功能，外部电源插入时，蓝色指示灯亮，可以通过 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)会检测到低电平，拔出时为高电平，使能该检测功能必须短接 UPS 背面的两个焊盘，且 GPIO 设置为输入状态，如果未短接焊盘则 GPIO 检测的状态为不稳定状态，详细见下图。





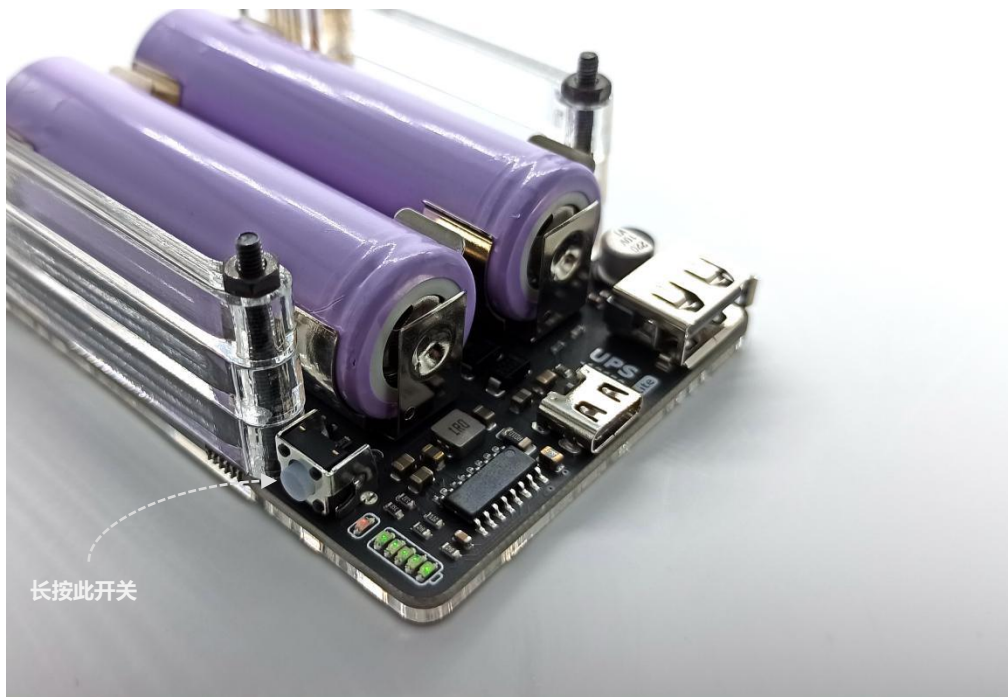
3.2 电源输出操作

在有接外部电源适配器的情况下，长按电源开关 3 秒以上，红色电源指示灯亮，表示输出 5V 电压。再一次长按 3 秒以上，红色电源指示灯灭，表示关闭 5V 电压输出。

在没有接外部电源适配器的情况下，长按电源开关 3 秒以上，红色电源指示灯亮，表示输出 5V 电压。再一次长按 3 秒以上，红色电源指示灯灭，表示关闭 5V 电压输出，也可快速短按电源开关两次关闭 5V 电压输出。短按电源开关一次，电量指示灯显示当前电池电量，稍后会熄灭。

UPS-18650-Lite 的升压电路输出功率受电池电压、电池放电电流、外部电源功率、以及电源线压降等因素影响。所以建议用质量好、放电电流大的 18650 电池进行供电，另外为了提高电路升压效率，建议电源线选择线压降小的线，这样可以尽量减小线压降所带来的功率消耗。

UPS-18650-Lite 持续输出电流为 2.5A，瞬间最大输出电流为 3.5A，在长时间持续大电流输出的情况下请做好散热措施，尽量在通风处使用，以免高温导致热关断保护及影响电池寿命。



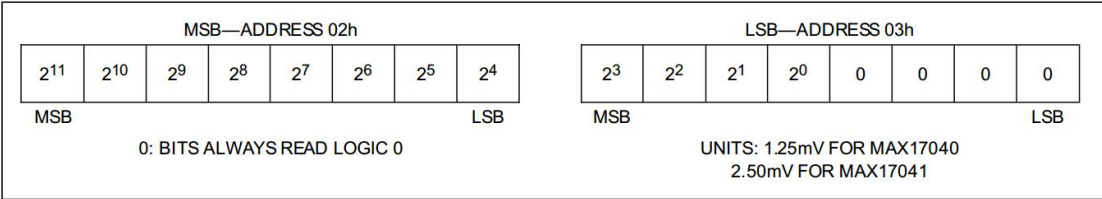
3.3 电池计量功能

用 nano 编辑器新建一个名字为 UPS_18650_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 3V 时，UPS 的保护电路会自动停止供电。具体操作步骤见下文

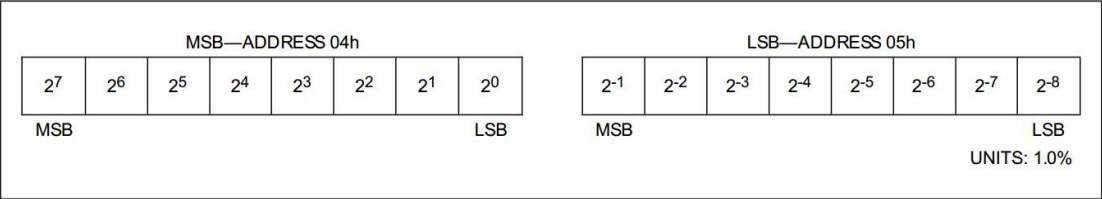
MAX17040G 寄存器地址以及功能介绍

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	—

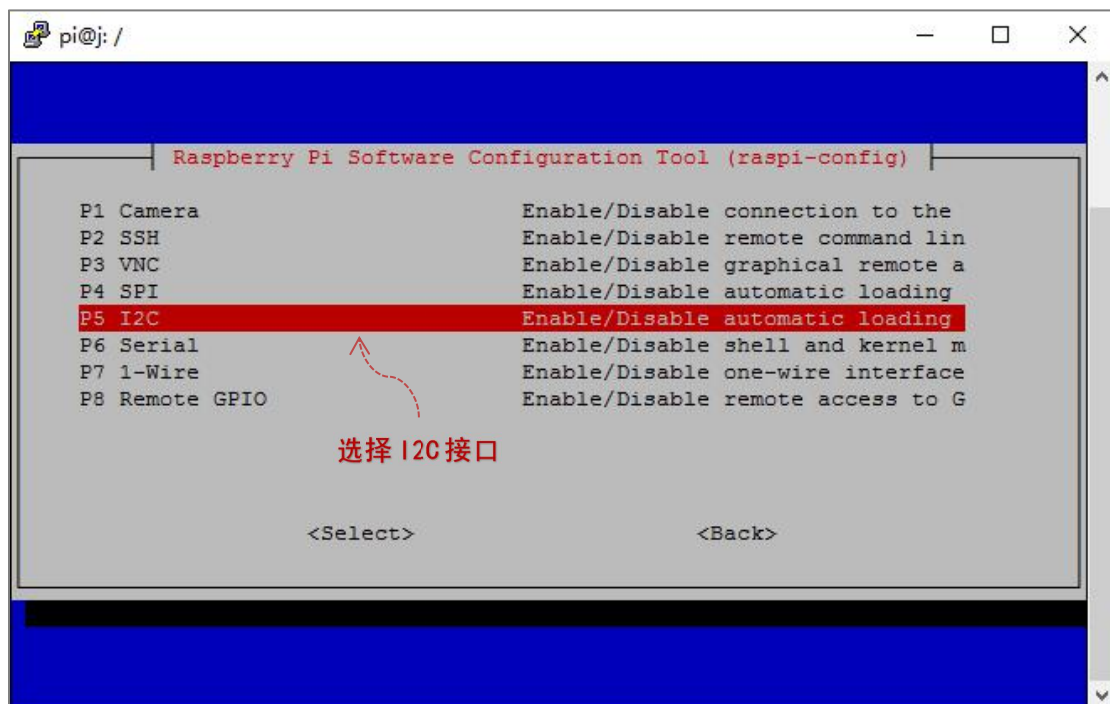
VCELL 寄存器



SOC 寄存器



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 总线

确定系统 i2c 总线为 1

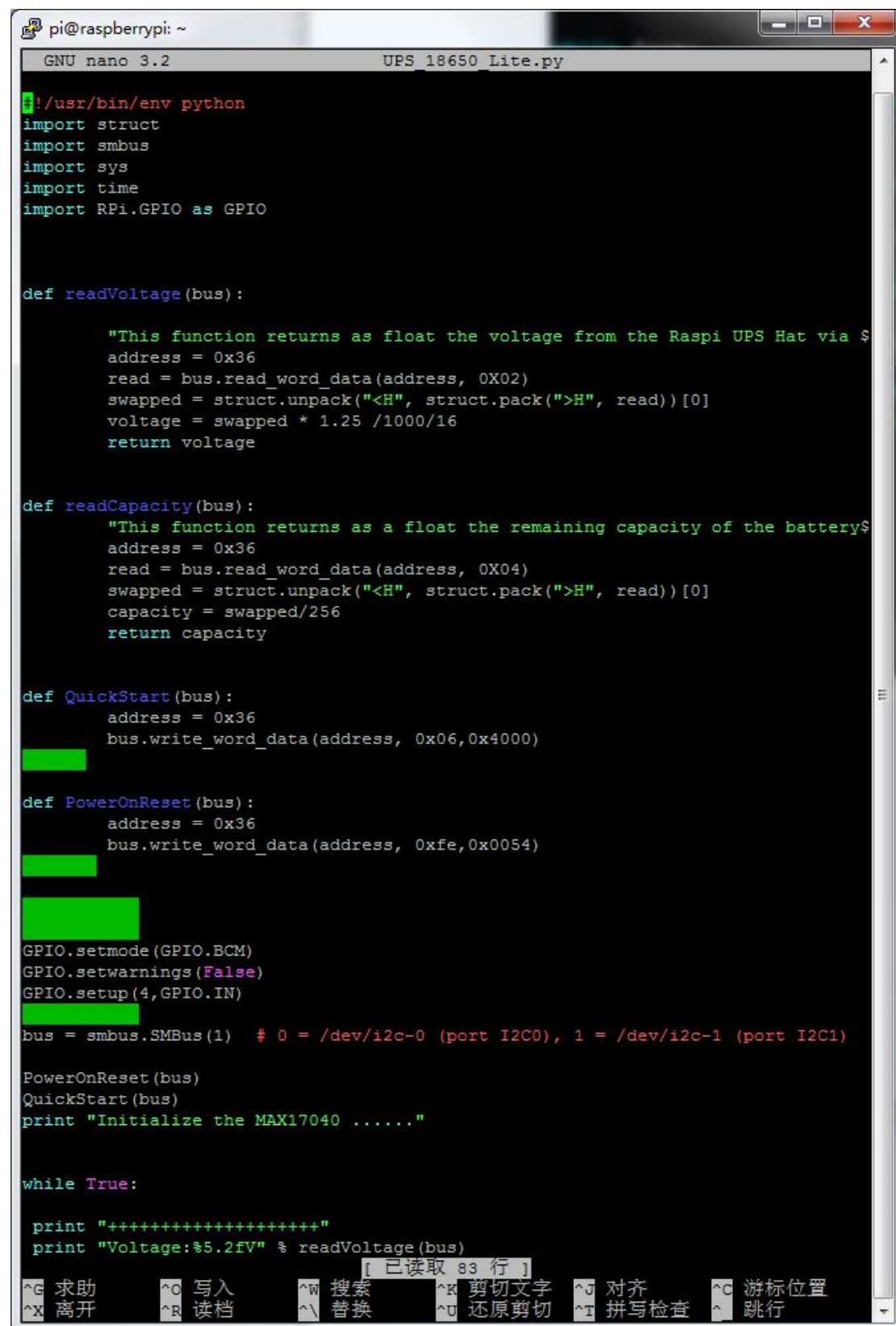
d. 运行 `sudo i2cdetect -y 1` 查看当前 pi 的 i2c 总线上挂载的设备。

```
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 总线

地址 0x36 为 MAX17040G

e. 用 nano 编辑器新建以下 UPS_18650_Lite.py 脚本程序 (详细代码见附录)



```
pi@raspberrypi: ~
GNU nano 3.2 UPS_18650_Lite.py

#!/usr/bin/env python
import struct
import smbus
import sys
import time
import RPi.GPIO as GPIO

def readVoltage(bus):

    "This function returns as float the voltage from the Raspi UPS Hat via $
    address = 0x36
    read = bus.read_word_data(address, 0x02)
    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, 0x04)
    swapped = struct.unpack("<H", struct.pack(">H", read))[0]
    capacity = swapped / 256
    return capacity

def QuickStart(bus):
    address = 0x36
    bus.write_word_data(address, 0x06, 0x4000)

def PowerOnReset(bus):
    address = 0x36
    bus.write_word_data(address, 0xfe, 0x0054)

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(4, GPIO.IN)

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

PowerOnReset(bus)
QuickStart(bus)
print "Initialize the MAX17040 ....."

while True:

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

[已读取 83 行]

^G 求助	^O 写入	^W 搜索	^K 剪切文字	^J 对齐	^C 光标位置
^X 离开	^R 读档	^\ 替换	^U 还原剪切	^T 拼写检查	^_ 跳行

f.用 Python 运行该脚本程序

```
pi@raspberrypi: ~  
pi@raspberrypi:~$ python UPS_18650_Lite.py  
Initialize the MAX17040 .....  
+++++  
Voltage: 0.00V  
Battery: 0%  
Battery LOW  
Power Adapter Plug In  
+++++  
+++++  
Voltage: 4.16V  
Battery: 96%  
Power Adapter Plug In  
+++++  
+++++  
Voltage: 4.16V  
Battery: 96%  
Power Adapter Plug In  
+++++  
+++++  
Voltage: 4.16V  
Battery: 96%  
Power Adapter Plug In  
+++++  
█
```

运行脚本程序

打印出电池电压值、电池容量百分比，外部电源插入情况

附录：

UPS_18650_Lite.py 的脚本程序代码：

```
#!/usr/bin/env python
import struct
import smbus
import sys
import time
import RPi.GPIO as GPIO

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, 0x02)
    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, 0x04)
    swapped = struct.unpack("<H", struct.pack(">H", read))[0]
    capacity = swapped / 256
    return capacity

def QuickStart(bus):
    address = 0x36
    bus.write_word_data(address, 0x06, 0x4000)

def PowerOnReset(bus):
    address = 0x36
    bus.write_word_data(address, 0xfe, 0x0054)
```

```

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(4, GPIO.IN)

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

print " "
print "Initialize the MAX17040 ....."

# The following is the power plug detection judgment program of V1.0 version. GPIO is
high when power is plugged in
#if (GPIO.input(4) == GPIO.LOW):
#     PowerOnReset(bus)
#-----

#The following is the power plug detection judgment program of V1.1 version. GPIO is
low when power is plugged in
if (GPIO.input(4) == GPIO.HIGH):
    PowerOnReset(bus)
#-----

while True:

    print "+++++++"
    print "Voltage:%5.2fV" % readVoltage(bus)
    print "Battery:%5i%" % readCapacity(bus)

    if readCapacity(bus) == 100:
        print "Battery FULL"
    if readCapacity(bus) < 5:
        print "Battery LOW"

#The following is the power plug detection judgment program of V1.0 version. GPIO is
high when power is plugged in
# if (GPIO.input(4) == GPIO.HIGH):
#     print "Power Adapter Plug In "
# if (GPIO.input(4) == GPIO.LOW):
#     print "Power Adapter Unplug"
#-----

```

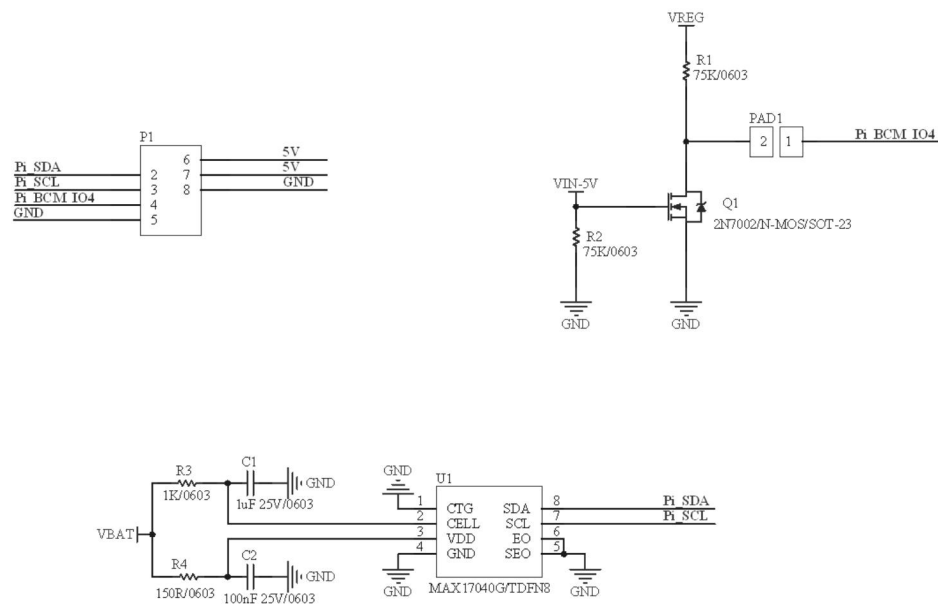
#The following is the power plug detection judgment program of V1.1 version. GPIO is low when power is plugged in

```
if (GPIO.input(4) == GPIO.LOW):  
    print "Power Adapter Plug In "  
if (GPIO.input(4) == GPIO.HIGH):  
    print "Power Adapter Unplug"
```

#-----

```
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>

如果使用上有其他问题的话可以联系我：416386001@qq.com