

Instructions for UPS-Lite V1.2

for Raspberry Pi Zero

-- by XiaoJ



1. Introduction

UPS-Lite is a UPS power supply specially designed for Raspberry Pi Zero (hereinafter referred to as pi). It uses a 1000mAh polymer lithium battery for power supply. It supports external power supply insertion detection, supports charging and discharging while plugging in external. When power is supplied, pi is powered by an external power supply. When the external power supply is unplugged, pi seamlessly switches to the lithium battery. UPS-Lite is connected to the pi through 5 Pogopin (the pi must be soldered with a 40-pin header). The power supply and battery power measurement functions of the pi are completed by the Pogopin. In addition, UPS-Lite also integrates a professional fuel gauge chip MAX17040G, dual-color charging status indicator.



Parameter :

Charging current: Max 400ma@5V

Output current: Max 1.3A@5V (only powered by battery without external power supply)
Max 2A@5V (with external power supply plugged in)

Battery measurement: percentage of battery SOC, an error $\pm 2\%$,
the measurement errors of battery voltages $\pm 3\text{mV}$

2.Install

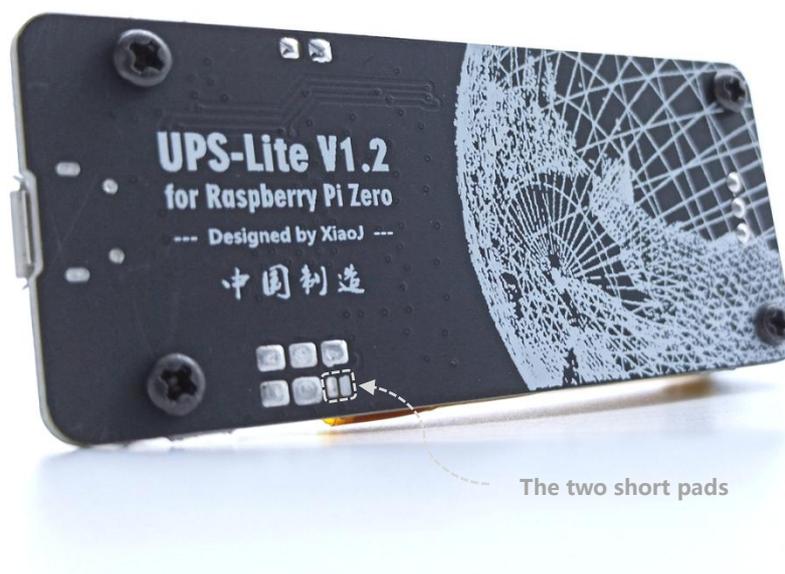
Align the four fixing holes of pi with the four screws of the UPS, and lock the matching nuts. Note that pi needs to be soldered to use UPS-Lite. Because the connection between pi and UPS-Lite is achieved through the contact of Pogopin and pin.



3.Function and using

3.1 Charging and power insertion detection function

It is recommended to use a power adapter with a power of 5V2A or above to charge the UPS-Lite. Because when the lithium battery is low, the external power adapter not only needs to supply power to the pi, but also needs to provide part of the current for the lithium battery to charge. During the charging process, the charging status indicator is red, indicating that the battery is charging. When the lithium battery is fully charged, the charging status indicator will turn green, indicating that the battery is fully charged. In addition, UPS-Lite has an external power adapter insertion detection function. You can judge whether the external power supply is inserted by the high and low levels of the GPIO port. When the power supply is plugged in, the io4 (BCM mode) of the pi will detect a high level. When unplugging ,the pi will detect a low level, **to enable the detection function, the two pads on the back of the UPS must be shorted, and the GPIO is set to the floating input state. If the pad is not shorted, the state of the GPIO detection is unstable.** See the following figure for details.



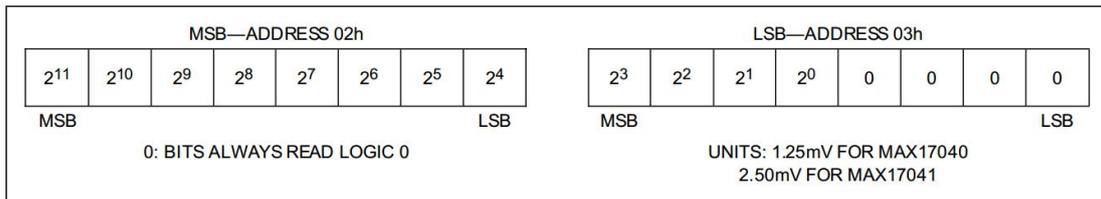
3.2 Battery power measurement functions

Create a script program named UPS_Lite.py with the nano editor. The code is as follows (see appendix for detailed code). This script uses Python's smbus library to perform i2c read operations on the MAX17040G. The MAX17040G device address is 0x36, the register VCELL is the 12bit battery voltage ADC measurement value, the address is 02h-03h, and the ADC measurement accuracy unit is 1.25mV. The register SOC is a 16-bit battery capacity percentage reading, The address is 04h-05h. The upper 8-bit unit of the SOC is 1% of the battery capacity, and the lower 8-bit unit is 1/256%. It provides the reading of the decimal point of the battery capacity percentage. Save the UPS_Lite.py script program to a directory you know (such as the following to the /home/pi/ directory), then run the program in Python, you can see that the program will output the current battery voltage value and the percentage of battery capacity every two seconds. In addition, due to the calculation method of the MAX17040G battery capacity, when the battery capacity reading is 1%, the battery voltage reading is about 3.5V. When the voltage of the lithium battery is 3.5V, the corresponding battery capacity is already very low, and excessive discharge will damage the battery. Therefore, when the user subsequently writes a low-power automatic shutdown program, it is recommended to automatically turn off the pi when the battery capacity is 1%. When running, when the battery voltage drops to 2.7V, the UPS-Lite protection circuit will automatically stop supplying power. See below for specific steps

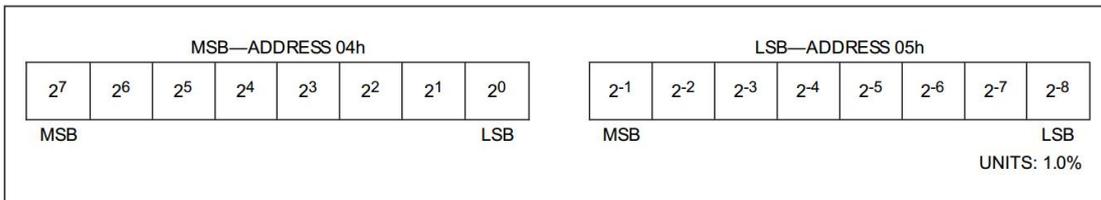
MAX17040G register address and Features

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	—

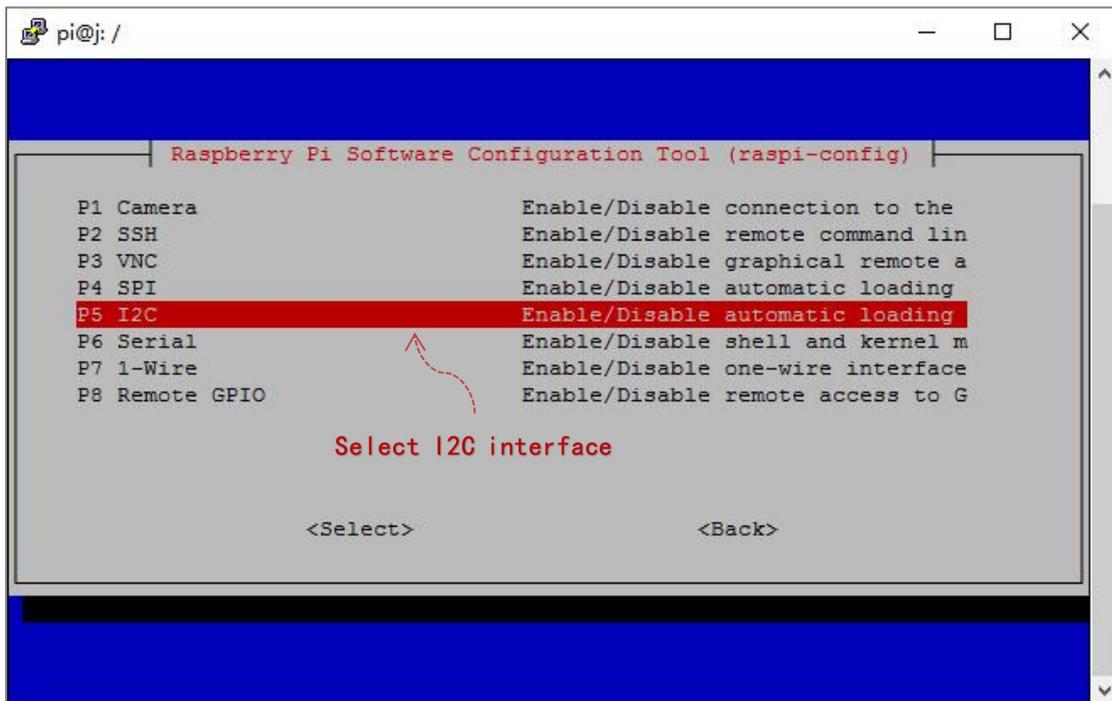
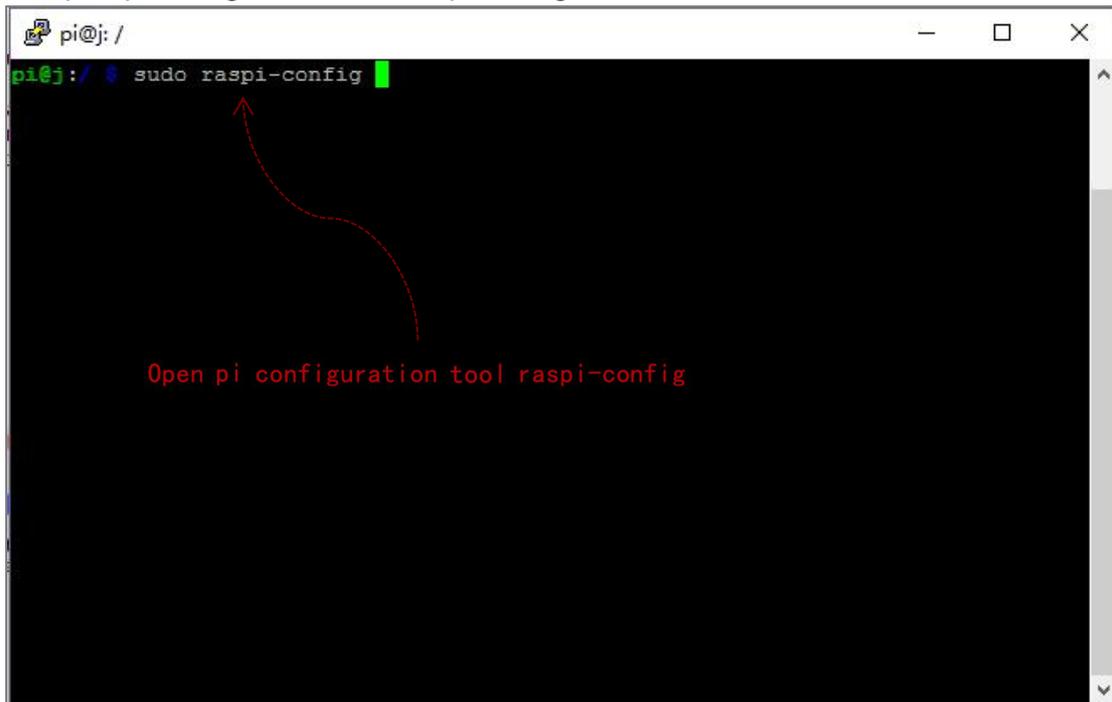
VCELL register

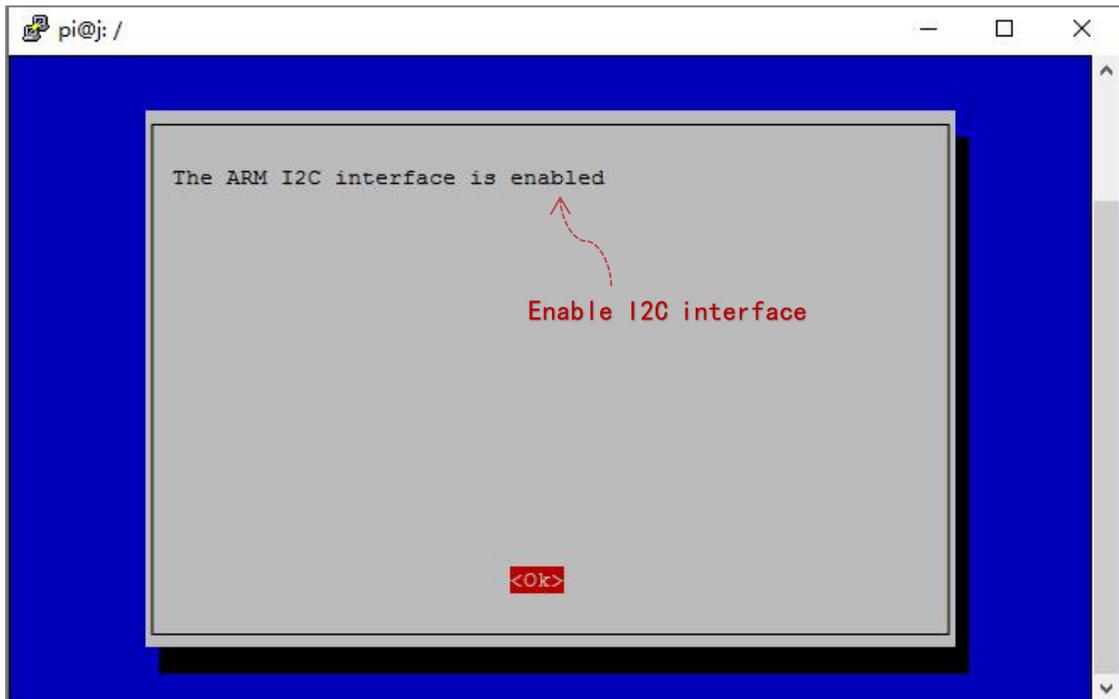


SOC register

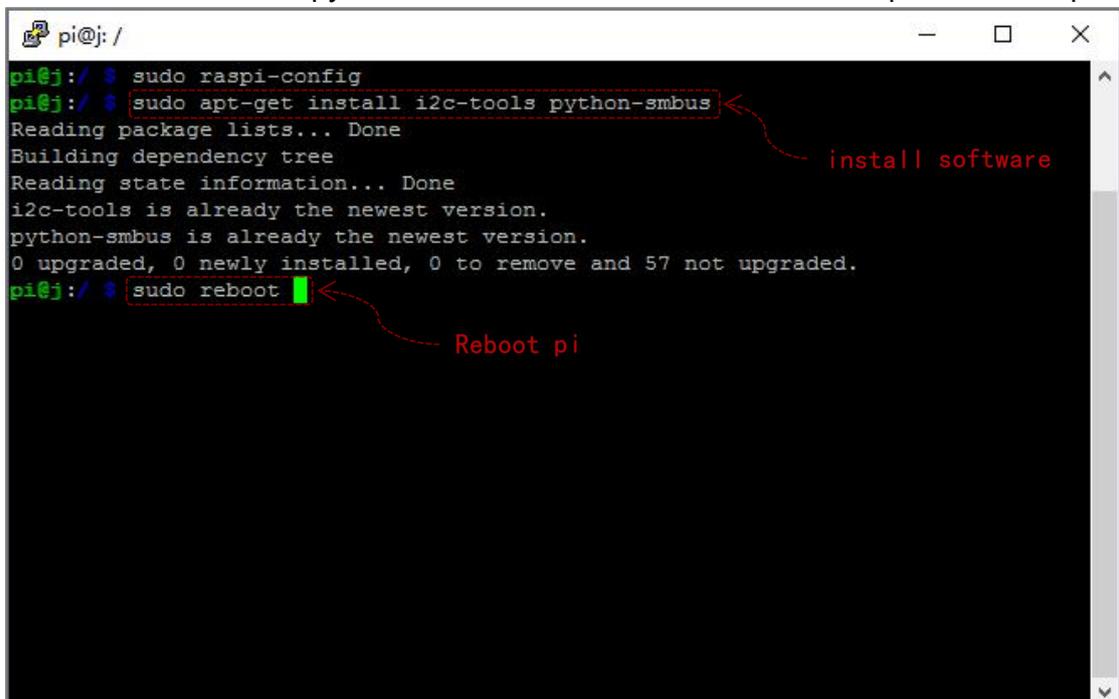


a. Open pi configuration tool raspi-config, enable I2C interface





b. Install i2c-tools and python-smbus, after the installation is complete. reboot pi



c. Run `sudo i2cdetect -l` to see which i2c bus the current pi is using.

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

Run `i2cdetect -l` to view the i2c bus

Determine the system i2c bus is 1

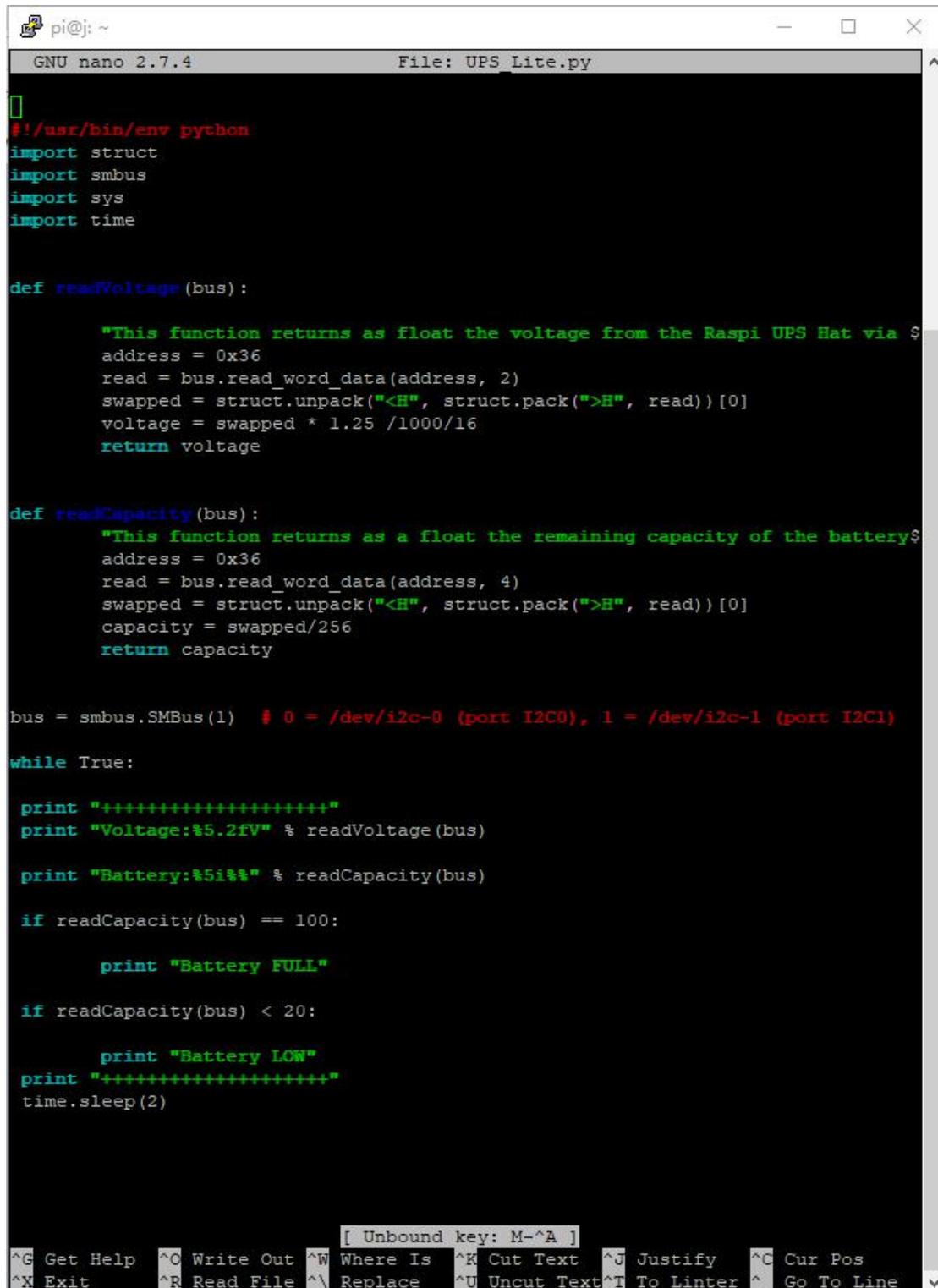
d. Run `sudo i2cdetect -y 1` to view the devices mounted on the i2c bus of the current pi

```
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: / $
```

Run `i2cdetect -l` to view the i2c bus

Address 0x36 is MAX17040G

e. Use nano editor to create the following UPS_Lite.py script program (see appendix for detailed code)



```
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:

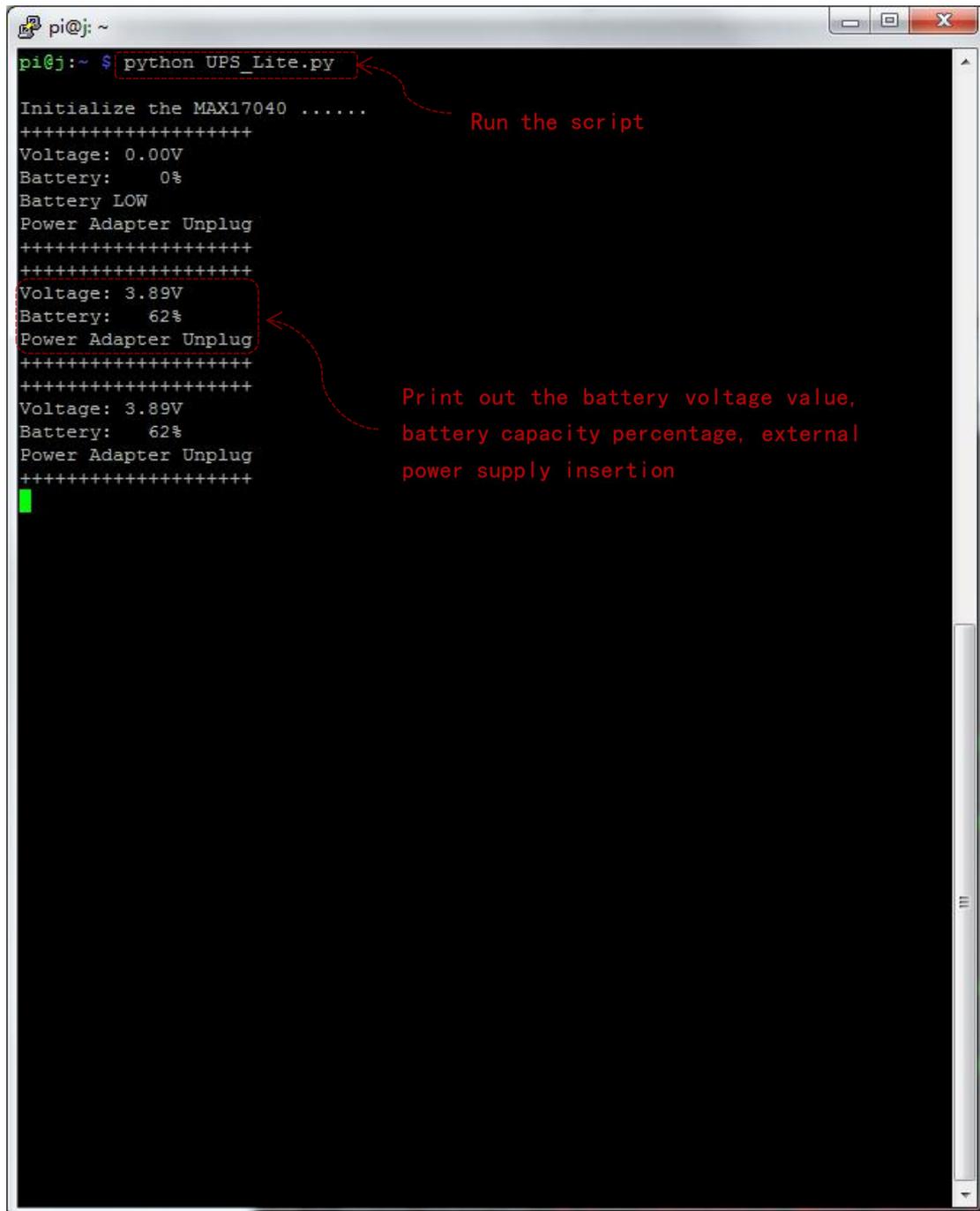
        print "Battery FULL"

    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.Run the script with Python



```
pi@j: ~  
pi@j:~ $ python UPS_Lite.py  
Initialize the MAX17040 .....  
+++++  
Voltage: 0.00V  
Battery: 0%  
Battery LOW  
Power Adapter Unplug  
+++++  
+++++  
Voltage: 3.89V  
Battery: 62%  
Power Adapter Unplug  
+++++  
+++++  
Voltage: 3.89V  
Battery: 62%  
Power Adapter Unplug  
+++++  
█
```

Run the script

Print out the battery voltage value, battery capacity percentage, external power supply insertion

appendix :

Script code of UPS_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 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)

PowerOnReset(bus)
QuickStart(bus)

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

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"

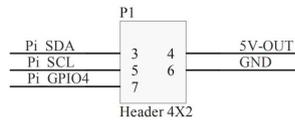
    if (GPIO.input(4) == GPIO.HIGH):
        print "Power Adapter Plug In "

    if (GPIO.input(4) == GPIO.LOW):
        print "Power Adapter Unplug"

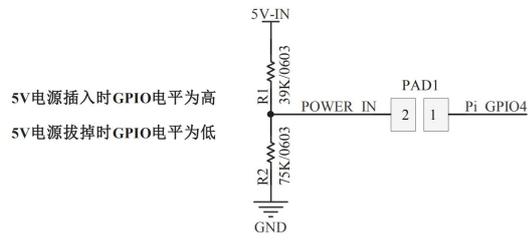
    print "+++++"
    time.sleep(2)

```

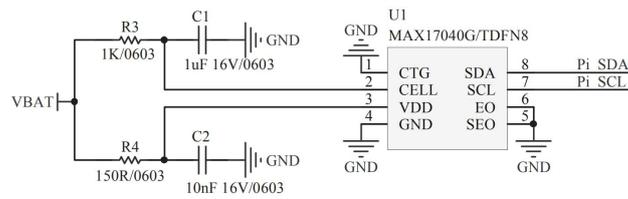
Schematic Reference Part :



树莓派接口



5V电源插入检测



电池电量检测电路

References :

MAX17040 Compact, low-cost 1S/2S fuel gauge - Maxim

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

If you have other questions, please contact me: 416386001@qq.com