Instructions for UPS-Lite V1.3

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 CW2015, 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 ±2%,

the measurement errors of battery voltages ± 3mV

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.

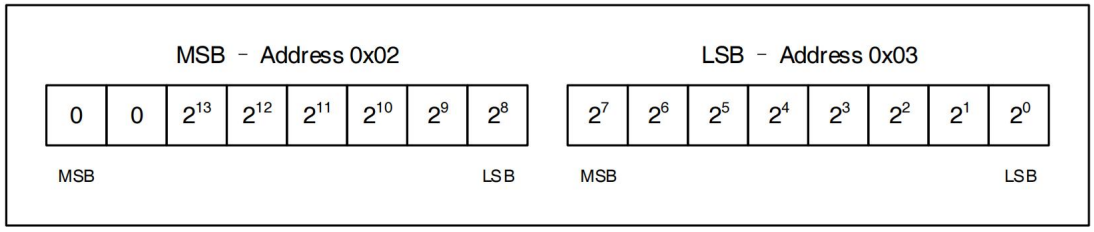
3.2 Battery power measurement functions

Create a script program named UPS_Lite_V1.3_CW2015.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 CW2015. The CW2015 device address is 0x62, the register VCELL is the 14bit battery voltage ADC measurement value, the address is 02h-03h, and the ADC measurement accuracy unit is 305uV. 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_V1.3_CW2015.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 CW2015 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

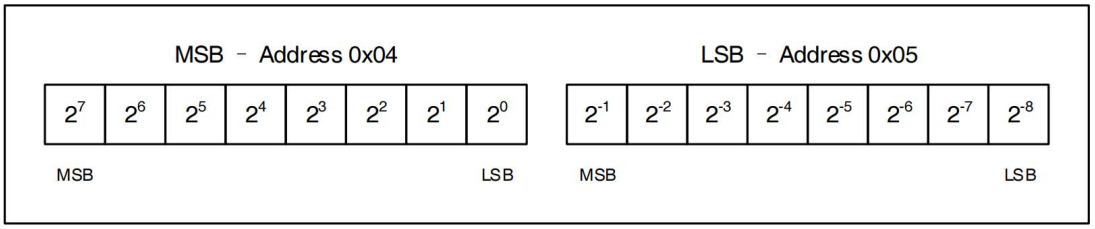
CW2015 register address and Features

REGISTER NAME	ADDRESS	DESCRIPTION	READ/WRITE	DEFAULT VALUE
VERSION	0x00	Returns IC version, software version	R	0x73
VCELL	0x02-0x03	Reports 14-bit A/D measurement of battery voltage	R	0x00
SOC	0x04-0x05	Reports 16-bit SOC result calculated	R	0x00
RRT_Alert	0x06-0x07	13 bits remaining run time and low SOC alert bit	W/R	0x00
CONFIG	0x08	Configure register, alert threshold set	W/R	0x18
MODE	0x0A	Special command for IC state	W/R	0xC0

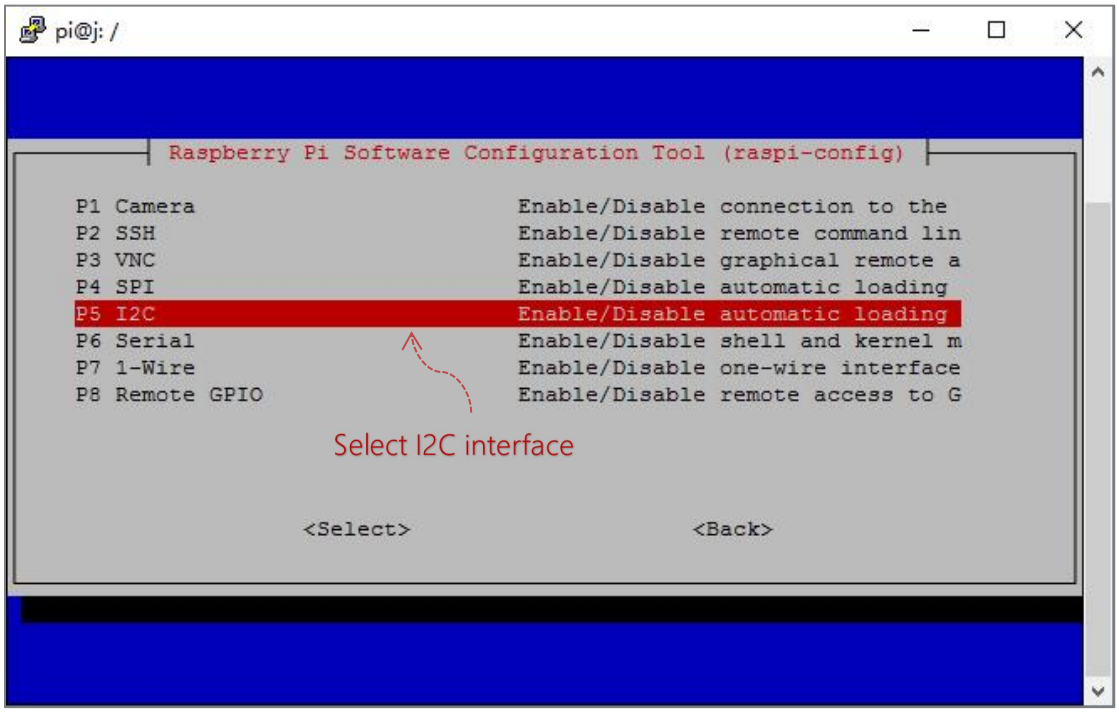
VCELL register

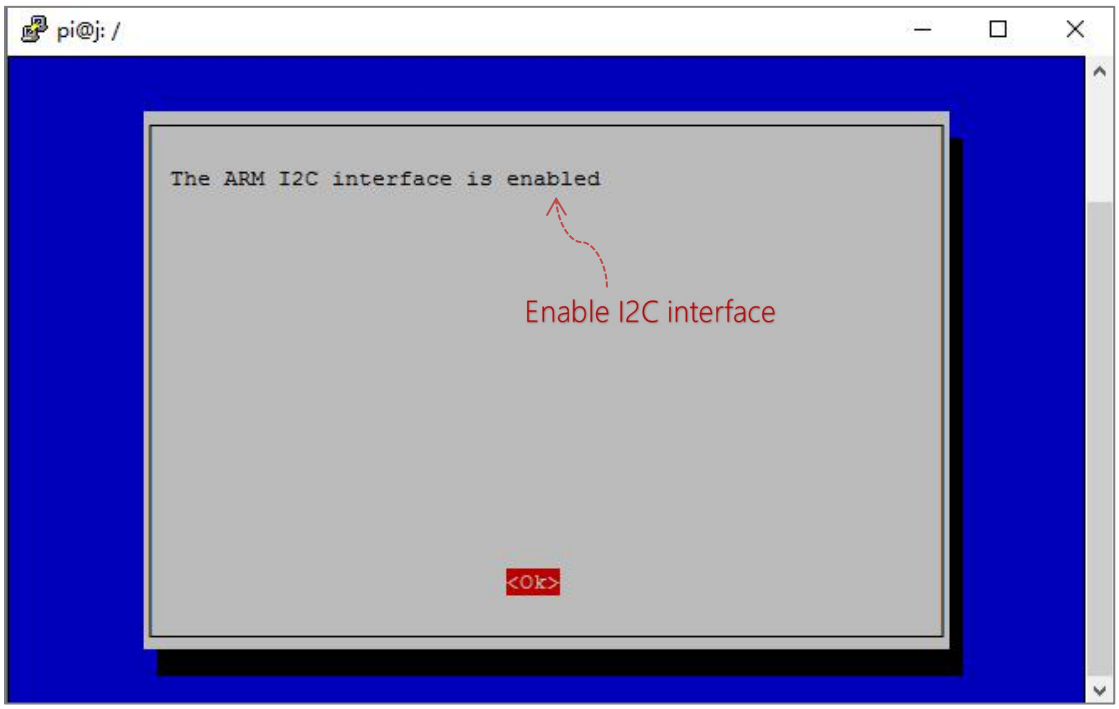


SOC register

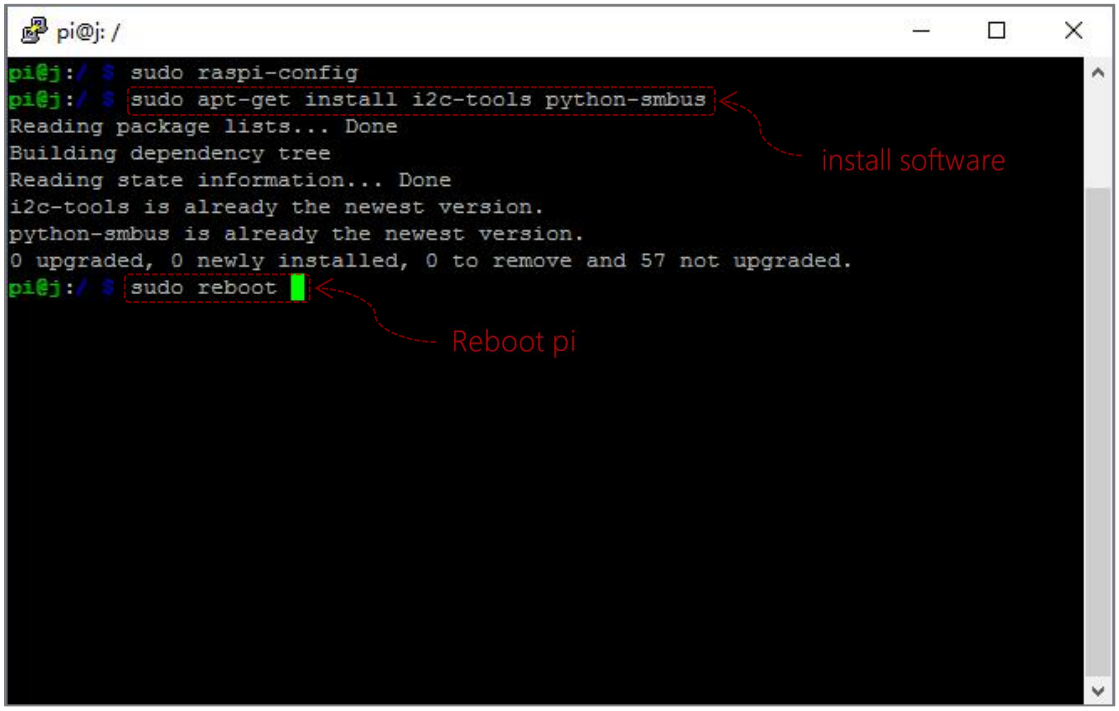


a. Open pi configuration tool raspi-config, enable I2C interface

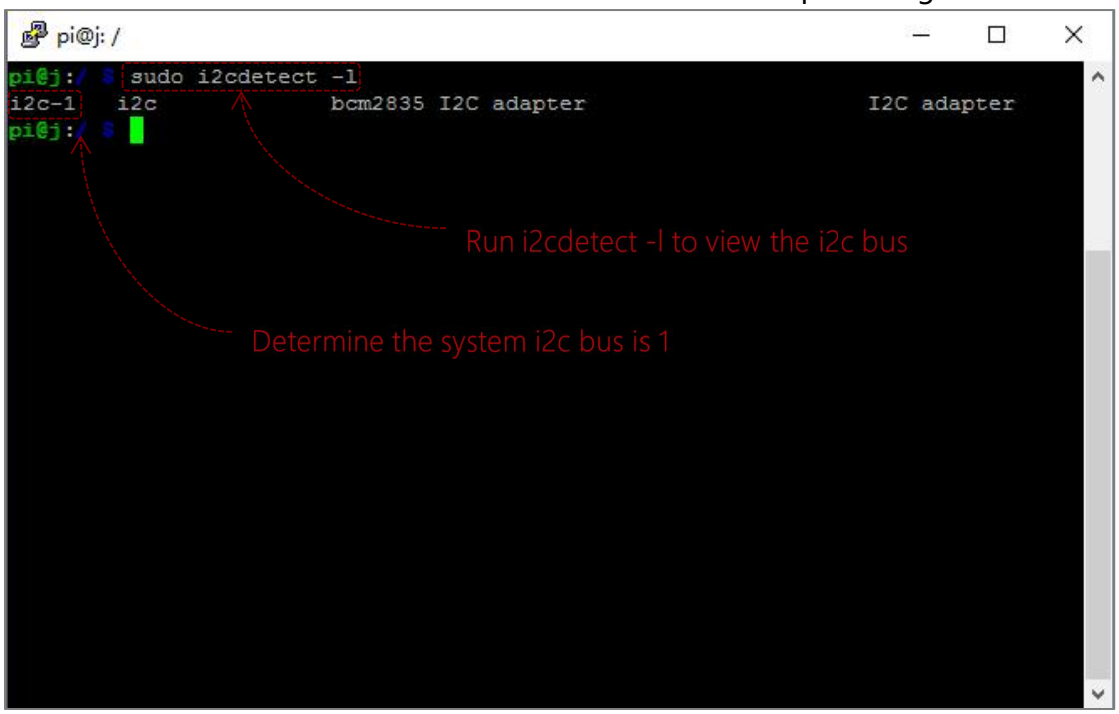




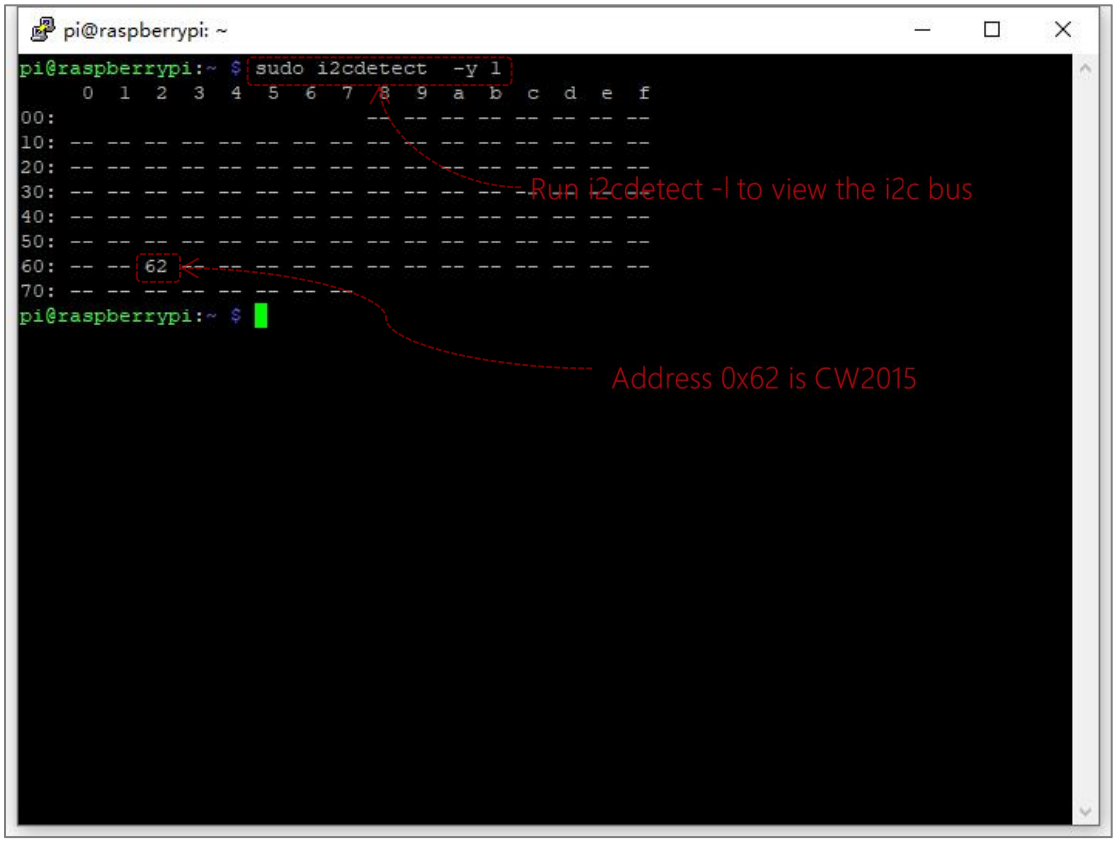
b. Install i2c-tools and python-smbus, after the installation is complete. reboot pi



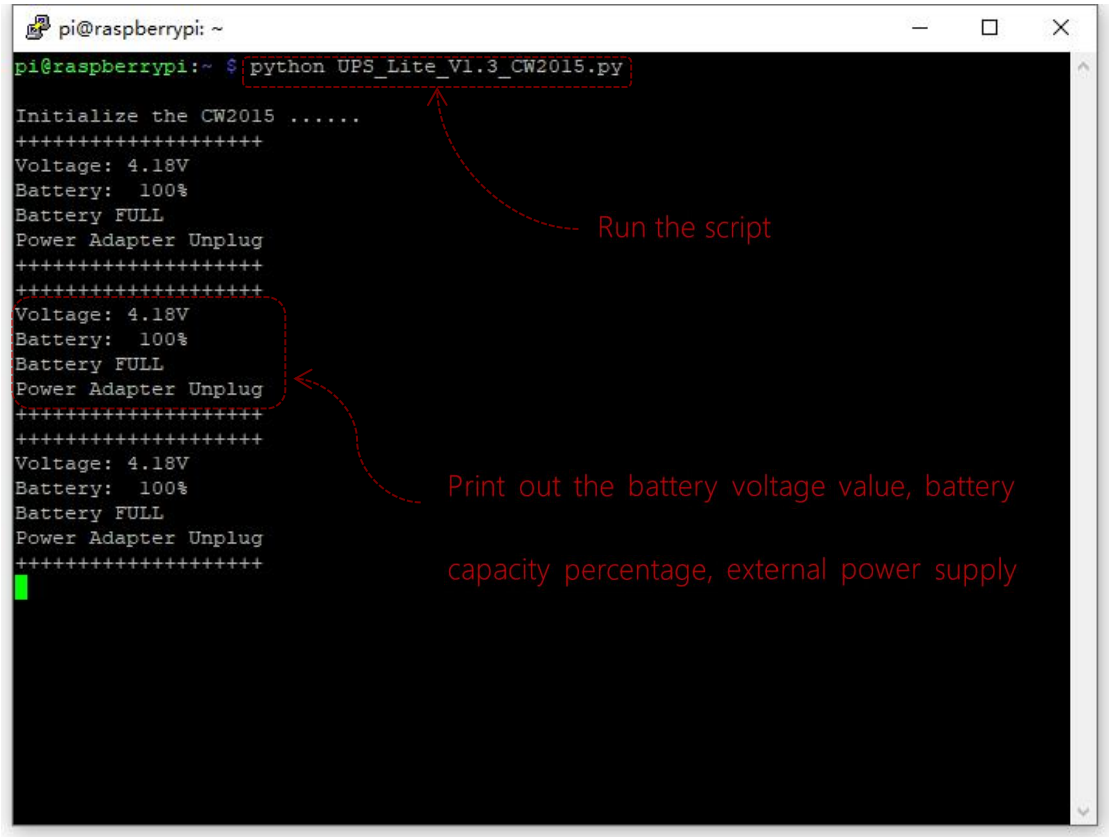
c. Run sudo i2cdetect -l to see which i2c bus the current pi is using.



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



e.Run the script with Python



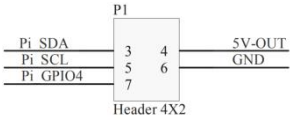
appendix:

Script code of UPS_Lite_V1.3_CW2015.py:

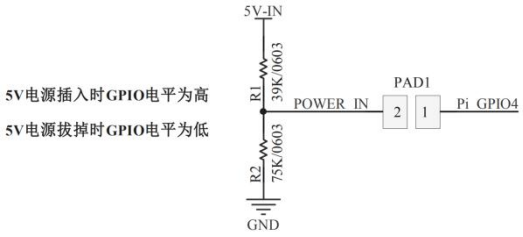
```
#!/usr/bin/env python
import struct
import smbus
import sys
import time
import RPi.GPIO as GPIO

CW2015_ADDRESS   = 0X62
CW2015_REG_VCELL = 0X02
CW2015_REG_SOC   = 0X04
CW2015_REG_MODE  = 0X0A



def readVoltage(bus):
        "This function returns as float the voltage from the Raspi UPS Hat
via the provided SMBus object"
        read = bus.read_word_data(CW2015_ADDRESS, CW2015_REG_VCELL)
        swapped = struct.unpack("<H", struct.pack(">H", read))[0]
        voltage = swapped * 0.305 /1000
        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"
        read = bus.read_word_data(CW2015_ADDRESS, CW2015_REG_SOC)
        swapped = struct.unpack("<H", struct.pack(">H", read))[0]
        capacity = swapped/256
        return capacity


def QuickStart(bus):
        "This function wake up the CW2015 and make a quick-start fuel-gauge
calculations "
        bus.write_word_data(CW2015_ADDRESS, CW2015_REG_MODE, 0x30)
```

```
#----------------------------------------------------------------------

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(4,GPIO.IN)  # GPIO4 is used to detect whether an external power
supply is inserted

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

QuickStart(bus)

print ("  ")
print ("Initialize the CW2015 ......")


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")



#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")



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

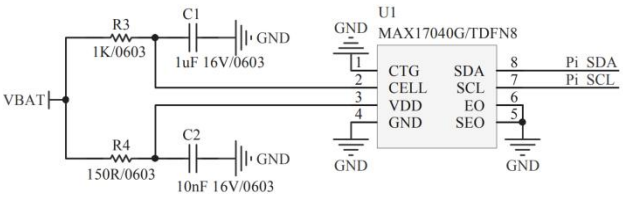
Schematic Reference Part:



树莓派接口



5V电源插入检测



电池电量检测电路

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