7 OLED display raspberry pi status

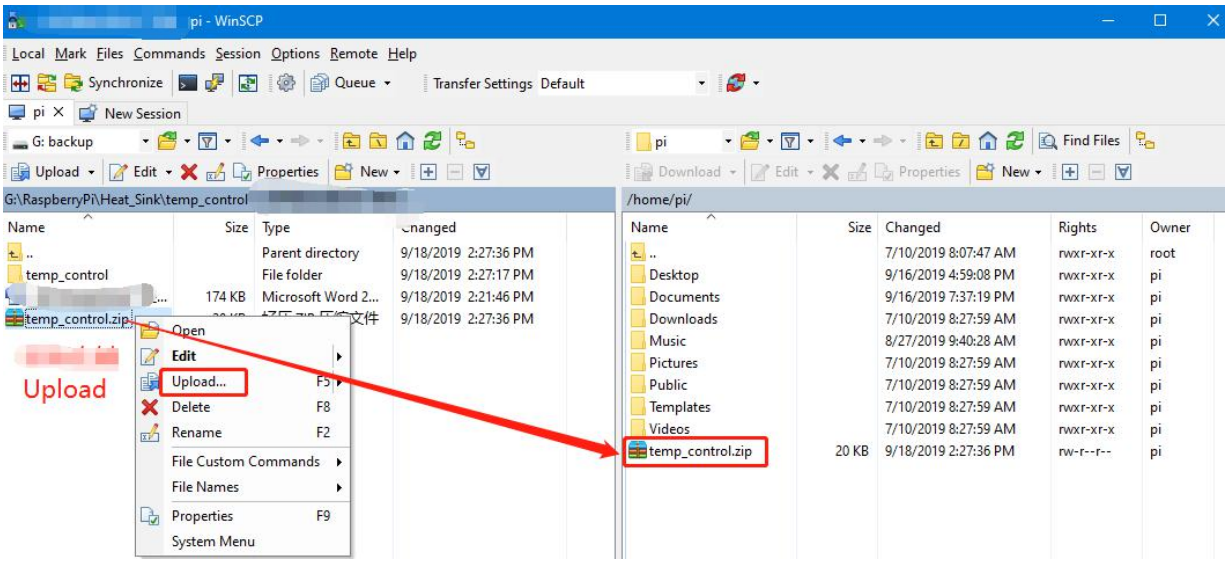
The Raspberry Pi RGB_Cooling_HAT needs to be properly plugged into the GPIO port of the Raspberry Pi and open the Raspberry Pi system **I2C** function.

This experimental phenomenon shows that OLED display CPU usage, CPU temperature, running memory usage, disk usage and IP address of the Raspberry Pi

1. File transfer

1.1 Install **WinSCP** tool on the computer side, connect the Raspberry Pi and transfer the **temp_control.zip** package to the pi directory of the Raspberry Pi.

Path of WinSCP:[Raspberry Pi RGB_Cooling_HAT]---[Tools]---[winscp556_setup.1416364912.exe]



1.2 Extract file

Open the Raspberry Pi terminal and input command ls to find the RGB_Cooling_HAT.zip file. As shown below:

Input command to extract file:

unzip RGB_Cooling_HAT.zip

```
pi@raspberrypi:~ $ unzip RGB_Cooling_HAT.zip
Archive:  RGB_Cooling_HAT.zip
   creating: RGB_Cooling_HAT/
  inflating: RGB_Cooling_HAT/RGB_Cooling_HAT.py
  inflating: RGB_Cooling_HAT/fan.py
  inflating: RGB_Cooling_HAT/fan_temp.py
  inflating: RGB_Cooling_HAT/install.sh
  inflating: RGB_Cooling_HAT/oled.py
  inflating: RGB_Cooling_HAT/rgb.py
  inflating: RGB_Cooling_HAT/rgb_effect.py
  inflating: RGB_Cooling_HAT/rgb_temp.py
 extracting: RGB_Cooling_HAT/start.desktop
  inflating: RGB_Cooling_HAT/start.sh
```

2. Compiling and running program

2.1 Input command to enter RGB_Cooling_HAT find file.

```
cd RGB_Cooling_HAT/
ls
```

```
pi@raspberrypi:~/RGB_Cooling_HAT $ ls
fan.py  fan_temp.py  install.sh  oled.py  RGB_Cooling_HAT.py  rgb_effect.py  rgb.py  rgb_temp.py  start.desktop  start.sh
pi@raspberrypi:~/RGB_Cooling_HAT $
```

2.2 Input command to run program.

```
python oled.py
```

```
pi@raspberrypi:~/RGB_Cooling_HAT $ python oled.py
idle:401  total:401
idle:401  total:401
```

We can see that OLED display CPU usage, CPU temperature, running memory usage, disk usage and IP address of the Raspberry Pi.

3. About code

3.1 Import the time module for delay. Import the os module for access to operating system services. And import the library used by the oled display.

SSD1306_128_32 represents the initialization method of the 128×32 resolution screen.

```
import time
import os

import Adafruit_SSD1306

from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont

import subprocess

# Raspberry Pi pin configuration:
RST = None     # on the PiOLED this pin isnt used

# 128x32 display with hardware I2C:
disp = Adafruit_SSD1306.SSD1306_128_32(rst=RST)

# Initialize library.
disp.begin()

# Clear display.
disp.clear()
disp.display()

# Create blank image for drawing.
# Make sure to create image with mode '1' for 1-bit color.
width = disp.width
height = disp.height
image = Image.new('1', (width, height))

# Get drawing object to draw on image.
draw = ImageDraw.Draw(image)

# Draw a black filled box to clear the image.
draw.rectangle((0,0,width,height), outline=0, fill=0)

# Draw some shapes.
# First define some constants to allow easy resizing of shapes.
padding = -2
top = padding
bottom = height-padding
# Move left to right keeping track of the current x position for drawing shapes.
x = 0

# Load default font.
font = ImageFont.load_default()
```

3.2 To read the CPU usage.

First, open /proc/stat file. This file save the CPU activity information.

All the values of this file are accumulated from the system startup to the current moment.

Enter cat /proc/stat in the terminal to check CPU activity data:

```
pi@raspberrypi:~ $ cat /proc/stat
cpu  969 0 1557 20390 623 0 10 0 0 0
cpu0 237 0 350 5170 142 0 9 0 0 0
cpu1 234 0 291 5199 155 0 0 0 0 0
cpu2 241 0 620 4779 192 0 1 0 0 0
cpu3 257 0 296 5241 132 0 0 0 0 0
intr 65172 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8063 0 0 0 0 365 0 0 0 0 0 0 0 85 0 0 6646 0 0 1331 0 0 0 0 22
002 0 0 0 0 0 249 0 0 0 0 0 36 0 0 39
ctxt 84864
btime 1576578059
processes 930
procs_running 1
procs_blocked 0
softirq 43173 2 6540 34 563 0 0 14422 5751 0 15861
pi@raspberrypi:~ $
```

To calculate the CPU usage, we only need to use the top row of data. In this course, I will only explain the meaning of this row of data.

(Jiffies is a global variable in the kernel, used to record the number of beats generated since the system started up. In Linux, a beat represents the minimum time slice of operating system process scheduling.
Different Linux kernels may have different values. 1 jiffies = 10ms)

Parameter	Analysis (unit: jiffies)
user(969)	From the system start up to the current moment, the running time in user mode, which not include the process with a nice value.
nice(0)	From the system start up to the current moment, nice value is CPU time occupied by processes negative.
system(1557)	From the system start up to the current moment, running time in the core state
idle(20390)	From the system start up to the current moment, waiting time other than IO waiting time
iowait(623)	From the system start up to the current moment, io waiting time
irq(0)	From the system start up to the current moment, hard interrupts time
softirq(10)	From the system start up to the current moment, soft interrupts time
stealstolen(0)	The time spent in other operating systems when running in a virtual environment
guest(0)	The Time to run the virtual CPU of the guest operating system under the control of the Linux kernel

The calculation formula of the total CPU time (cumulative value):
Total Time = user+nice+system+idle+iowait+irq+softirq+stealstolen+guest
We need to calculate the current CPU occupancy rate by reading the above parameters.
Since it is accumulated from the system start up to the current moment, the difference between the two parameters collected through a short time (1 second) interval can be calculate the total amount of CPU time total.
Then, calculate the idle time idle in the same way.
Finally, the CPU usage rate=100*(total-idle)/total.

In addition, you can also directly enter the command to view the current CPU usage on the Raspberry Pi terminal.
You can display the CPU usage by entering the following command.
cat <(grep 'cpu ' /proc/stat) <(sleep 1 && grep 'cpu ' /proc/stat) | awk -v RS="" '{print ($13-$2+$15-$4)*100/($13-$2+$15-$4+$16-$5) "%"}'

```
pi@raspberrypi:~/Documents $ cat <(grep 'cpu ' /proc/stat) <(sleep 1 && grep 'cpu ' /proc/stat) | awk -v RS
="" '{print ($13-$2+$15-$4)*100/($13-$2+$15-$4+$16-$5) "%"}'
0.740741%
pi@raspberrypi:~/Documents $
```

Code:

```
def getCPULoadRate():
    f1 = os.popen("cat /proc/stat", 'r')
    stat1 = f1.readline()
    count = 10
    data_1 = []
    for i  in range (count):
        data_1.append(int(stat1.split(' ')[i+2]))
    total_1 = data_1[0]+data_1[1]+data_1[2]+data_1[3]+data_1[4]+data_1[5]+data_1[6]+data_1[7]+data_1[8]+data_1[9]
    idle_1 = data_1[3]

    time.sleep(1)

    f2 = os.popen("cat /proc/stat", 'r')
    stat2 = f2.readline()
    data_2 = []
    for i  in range (count):
        data_2.append(int(stat2.split(' ')[i+2]))
    total_2 = data_2[0]+data_2[1]+data_2[2]+data_2[3]+data_2[4]+data_2[5]+data_2[6]+data_2[7]+data_2[8]+data_2[9]
    idle_2 = data_2[3]

    total = int(total_2-total_1)
    idle = int(idle_2-idle_1)
    usage = int(total-idle)
    print("idle:"+str(idle)+"  total:"+str(total))
    usageRate = int(float(usage * 100  / total))
    return "CPU:"+str(usageRate)+"%"
```

Obtain the CPU file information through os.popen, and only read the information in the top row to stat1.
Read the information of stat1 to the date_1 list, and then calculate the total_1 and idle_1 of the first reading.
Then, delay by 1 second(the tested data will be invalid if the time is less than 1 second).
Next, save the second reading data in total_2 and idle_2.
Finally, calculate the value of CPU usageRate.

3.3 Read the running memory occupancy rate, get the result of the specified command through subprocess.check_output, cmd defines the command to get the memory ratio of the specified format;

```
cmd = "free -m | awk 'NR==2{printf \"RAM:%s/%s MB \", $2-$3,$2}'"
MemUsage = subprocess.check_output(cmd, shell = True )
```

3.4 Read the IP address, you can display the IP address of the network cable and WiFi network. Display network cable IP address has higher priority

```
cmd = "hostname -I | cut -d\' \' -f1"
IP = subprocess.check_output(cmd, shell = True )
```

5. Get temperature.

```
cmd = os.popen('vcgencmd measure_temp').readline()
CPU_TEMP = cmd.replace("temp=","Temp:").replace("'C\n","C")
```

6. Read disk space.

```
cmd = "df -h | awk '$NF==\"/\"{printf \"Disk:%d/%dMB\", ($2-$3)*1024,$2*1024}'"
Disk = subprocess.check_output(cmd, shell = True )
```

7. Display this message on OLED.

```
draw.text((x, top), str(CPU), font=font, fill=255)
draw.text((x+56, top), str(CPU_TEMP), font=font, fill=255)
draw.text((x, top+8), str(MemUsage),  font=font, fill=255)
draw.text((x, top+16), str(Disk),  font=font, fill=255)
draw.text((x, top+24), "wlan0:" + str(IP),  font=font, fill=255)
```

The draw.text() function is to set the content displayed on OLED.
The first parameter is x, which controls the left and right offsets;
the second parameter is y, which controls the upper and lower offsets;
the third parameter is the string, which is what to display.

Finally, you need to run the disp.display() function to refresh the display.

```
# Display image.
disp.image(image)
disp.display()
time.sleep(.1)
```