7.Install OLED driver and display information

RGB cooling HAT needs to be correctly inserted into the GPIO port of the Raspberry Pi and the I2C function of the Raspberry Pi must be turned on.

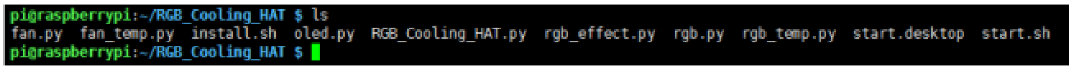
The phenomenon of this experiment is that OLED displays the CPU usage, CPU temperature, running memory usage, disk usage and IP address of Raspberry Pi.

1.Compile and run the program

1.1 Enter the folder and view the files in the current folder

```
cd RGB_Cooling_HAT/
```

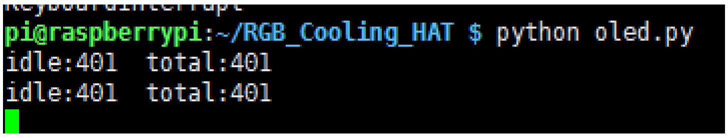
```
ls
```



```
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 $ 
```

1.2 Run program

```
python oled.py
```

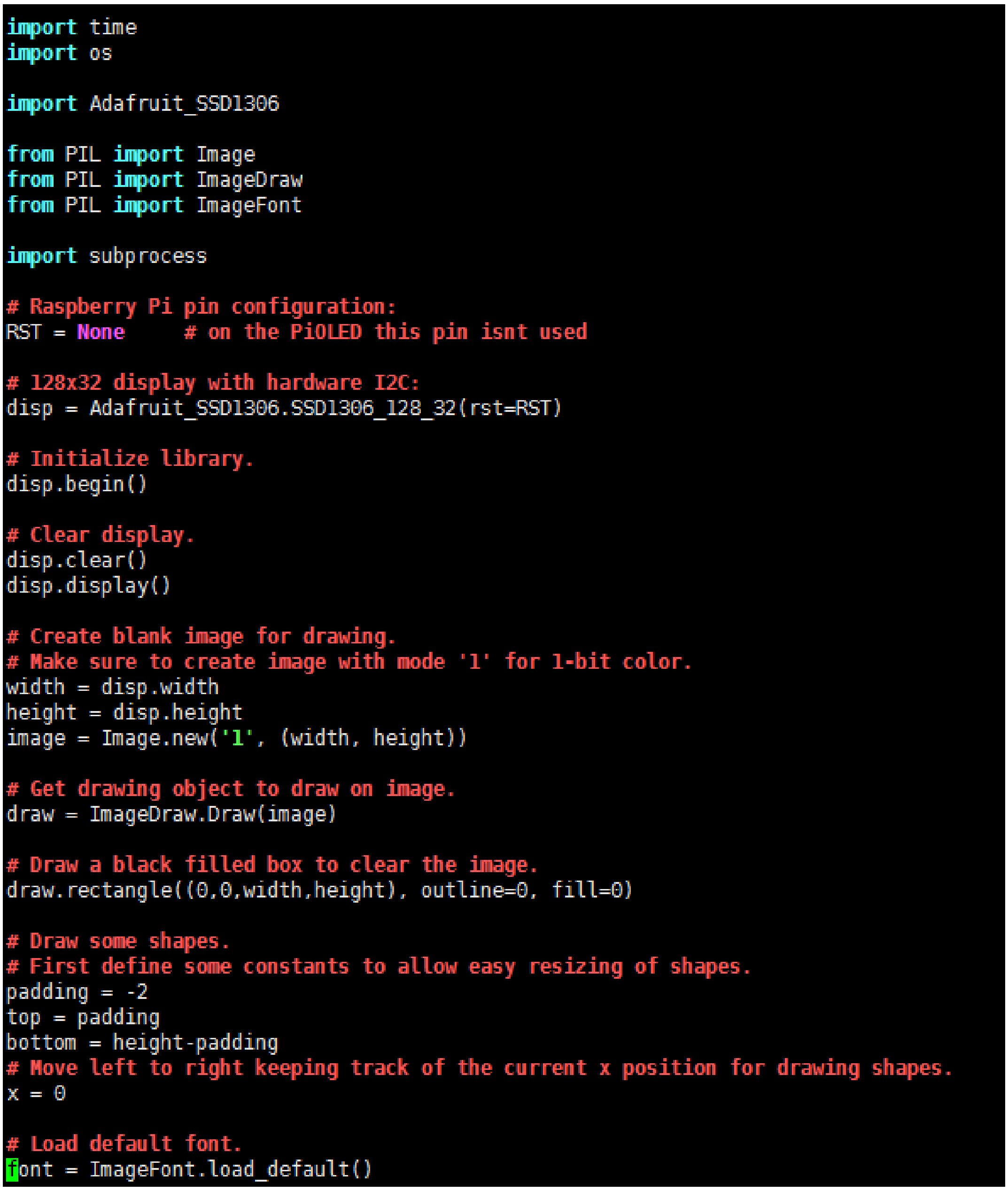


```
KeyboardInterrupt
pi@raspberrypi:~/RGB_Cooling_HAT $ python oled.py
idle:401  total:401
idle:401  total:401
```

At this time, we can see the Raspberry Pi's CPU usage, CPU temperature, running memory usage, disk usage, IP address and other information displayed on the OLED screen.

2. Code analysis

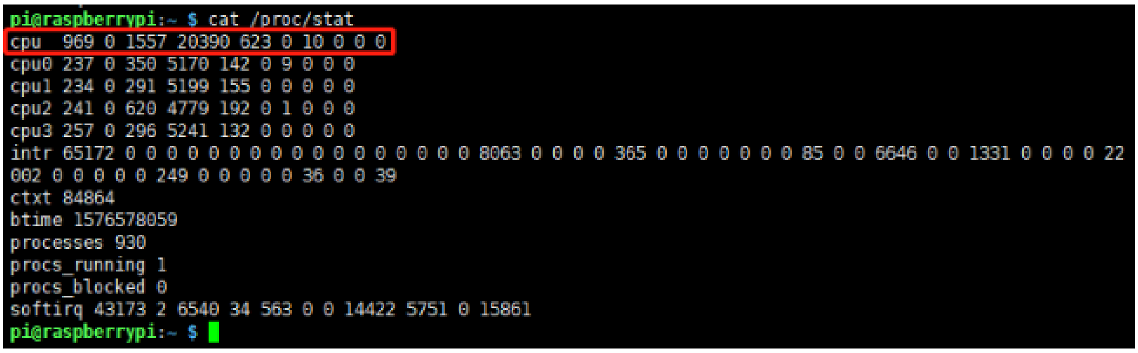
2.1 Import the time module for delay, the os module for accessing operating system service functions, and the libraries used for OLED displays.
SSD1306_128_32 represents the initialization method for 128×32 resolution screen.



```python
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()
```

2.2 Read the CPU usage, the first file you need to open is /proc/stat. This file saves the CPU activity information. All values in this file are accumulated from the system startup to the current moment.

Enter cat /proc/stat in the terminal to check the 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 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:~ $
```

If we want to calculate the CPU usage, we only need to use the top row of data. Below 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. In Linux, a beat can be roughly understood as the minimum time slice for operating system process scheduling. Different Linux kernels may have different values. We It can be thought of as: 1 jiffies = 10ms)

Parameter	Analysis (unit: jiffies)
user(969)	The running time in user mode accumulated from system startup to the current moment, excluding processes with negative nice values.
nice(0)	The CPU time occupied by processes with negative nice values accumulated from system startup to the current moment.
system(1557)	The running time in the core state accumulated from the system startup to the current moment
idle(20390)	Waiting time other than IO waiting time accumulated from system startup to the current moment
iowait(623)	IO waiting time accumulated from system startup to the current moment
irq(0)	Accumulated from system startup to the current moment, the hard interrupt time
softirq(10)	The soft interrupt time accumulated from system startup to the current moment
stealstolen(0)	How much time is spent in other operating systems when running in a virtual environment
guest(0)	How much time does it take to run a virtual CPU of a guest operating system under the control of the Linux kernel

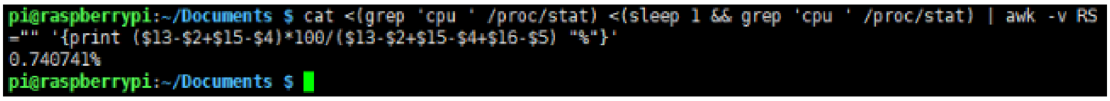
Calculation formula of total CPU time (accumulated value):

totalTime = user+nice+system+idle+iowait+irq+softirq+stealstolen+guest

We need to calculate the current CPU occupancy by reading the parameters of the above content. Since it is accumulated from the system startup to the current moment, the difference between the two parameters collected at short intervals (1 second) can be Calculate the total CPU time, then calculate the idle time idle in the same way, and finally the CPU usage rate=100*(total-idle)/total.

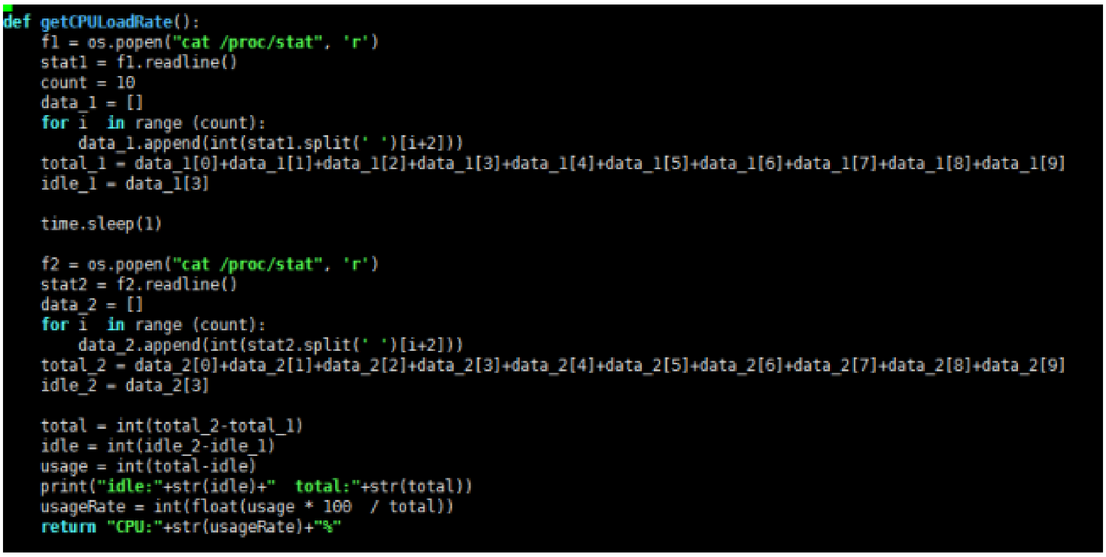
In addition: In fact, you can also directly enter commands on the Raspberry Pi terminal to view the current CPU usage. You can mainly enter the following code to display the CPU usage:

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 $ 
```

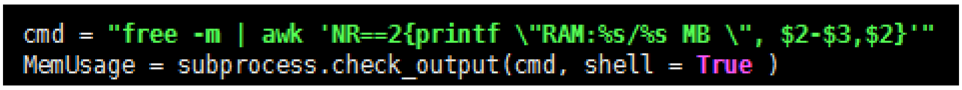
Code:



```
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)+"%"
```

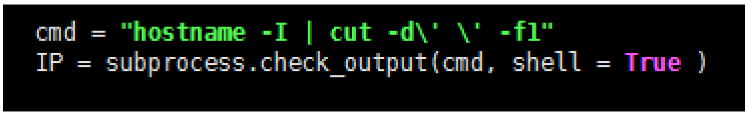
Get the CPU file information through os.popen, and only read the top line of information to stat1. Read the stat1 information into the date_1 list, and then calculate the total_1 and idle_1 read for the first time; then delay it by 1 second. It has been tested that if the time is less than 1 second, the read data will fail; the second read The data is saved to total_2 and idle_2; finally, the value of CPU usage usageRate is calculated.

2.3 Read the running memory usage and obtain the result of the specified command through subprocess.check_output. cmd defines a command to obtain the memory usage of the specified format;



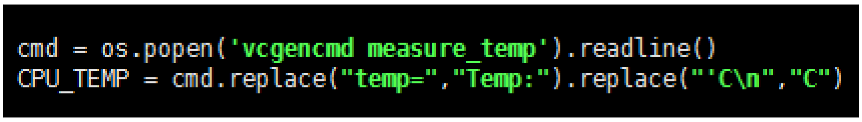
```
cmd = "free -m | awk 'NR==2{printf \"RAM:%s/%s MB \", $2-$3,$2}'"
MemUsage = subprocess.check_output(cmd, shell = True )
```

2.4 Read the IP address, which can display the IP addresses of the network cable and WiFi network, and the IP address of the network cable will be displayed first.



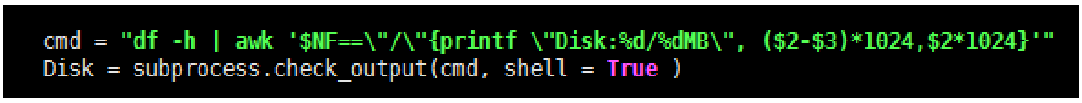
```
cmd = "hostname -I | cut -d\' \' -f1"
IP = subprocess.check_output(cmd, shell = True )
```

2.5 Read the temperature.



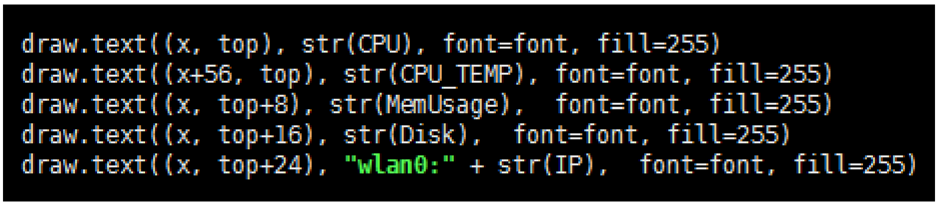
```
cmd = os.popen('vcgencmd measure_temp').readline()
CPU_TEMP = cmd.replace("temp=","Temp:").replace("'C\n","C")
```

2.6 Read disk space



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

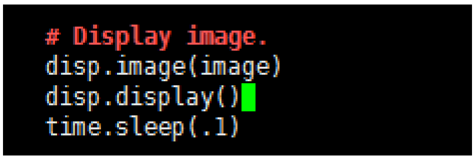
2.7 Display content on oled



```
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 the 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 a string, that is Content to display.

Finally, the disp.display()) function must be run to refresh the display.



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