

5.Change RGB color according to CPU temperature

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.

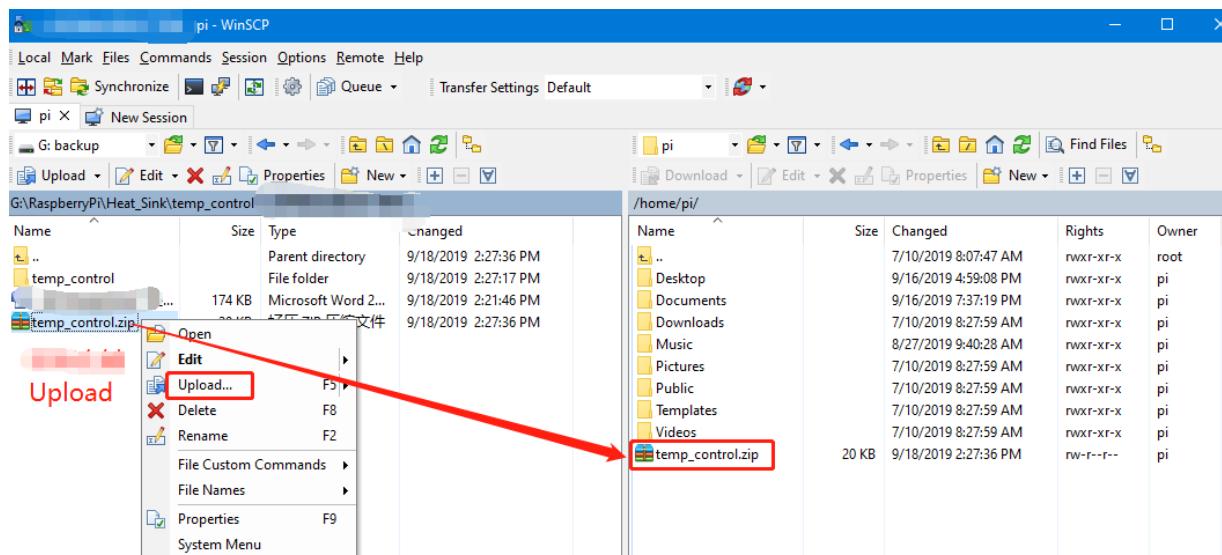
The experimental phenomenon is to read and print the Raspberry Pi CPU temperature, adjust the color change of the RGB lamp according to the temperature.

The color distribution is roughly as follows: the low temperature turns blue, the medium temperature turns yellow, and the high temperature turns red.

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 **temp_control.zip** file.

As shown below:

```
pi@raspberrypi:~ $ ls
Desktop  Downloads  Pictures  temp_control.zip  Videos
Documents  Music    Public    Templates
pi@raspberrypi:~ $
```

Input command to extract file:

unzip temp_control.zip

```
pi@raspberrypi:~ $ unzip temp_control.zip
Archive: temp_control.zip
  creating: temp_control/
  inflating: temp_control/fan
  inflating: temp_control/fan.c
  inflating: temp_control/fan_temp
  inflating: temp_control/fan_temp.c
  inflating: temp_control/oled
  inflating: temp_control/oled.c
  inflating: temp_control/oled_fonts.h
  inflating: temp_control/rgb
  inflating: temp_control/rgb.c
  inflating: temp_control/rgb_effect
  inflating: temp_control/rgb_effect.c
  inflating: temp_control/ssdl306_i2c.c
  inflating: temp_control/ssdl306_i2c.h
  inflating: temp_control/start.desktop
  inflating: temp_control/start.sh
  inflating: temp_control/temp_control
  inflating: temp_control/temp_control.c
pi@raspberrypi:~ $
```

2. Compiling and running program

2.1 Input command to enter temp_control find file:

```
cd temp_control/
ls
```

```
pi@raspberrypi:~ $ cd temp_control
pi@raspberrypi:~/temp_control $ ls
fan           install.sh    rgb          rgb_temp      start.desktop
fan.c         oled          rgb.c        rgb_temp.c   start.sh
fan_temp     oled.c        rgb_effect  ssdl306_i2c.c temp_control
fan_temp.c   oled_fonts.h  rgb_effect.c ssdl306_i2c.h  temp_control.c
pi@raspberrypi:~/temp_control $
```

2.2 Input command to compile:

```
gcc -o rgb_temp rgb_temp.c -lwiringPi
pi@raspberrypi:~/temp_control $ gcc -o rgb_temp `rgb_temp.c` -lwiringPi
rgb_temp.c: In function 'main':
rgb_temp.c:50:13: warning: implicit declaration of function 'read'; did you mean
  'fread'? [-Wimplicit-function-declaration]
    if (read(fd_temp, buf, MAX_SIZE) < 0)
    ^~~~~
      fread
rgb_temp.c:59:9: warning: implicit declaration of function 'close'; did you mean
  'pclose'? [-Wimplicit-function-declaration]
    close(fd_temp); //关闭文件
    ^~~~~
    pclose
pi@raspberrypi:~/temp_control $ ls
fan           install.sh    rgb          rgb_temp      start.desktop
fan.c         oled          rgb.c        rgb_temp.c   start.sh
fan_temp     oled.c        rgb_effect  ssdl306_i2c.c temp_control
fan_temp.c   oled_fonts.h  rgb_effect.c ssdl306_i2c.h  temp_control.c
pi@raspberrypi:~/temp_control $
```

Among them, the gcc compiler is called, -o means to generate the file, **rgb_temp** is the generated file name, **rgb_temp.c** is the source program, **-lwiringPi** is the wiringPi library that references the Raspberry Pi.

2.3 Input command to run the program

```
./rgb_temp
```

```
pi@raspberrypi:~/temp_control $ ./rgb_temp
temp: 45.8C
temp: 46.7C
temp: 46.7C
temp: 46.3C
temp: 46.3C
temp: 47.7C
temp: 47.7C
temp: 47.2C
temp: 47.2C
temp: 47.7C
```

The terminal will print the current CPU temperature value, and the RGB light color will also change with the color change.

3. About code

3.1 First, import the file control library and the I2C library. The path of the Raspberry Pi to view the CPU temperature is defined as TEMP_PATH.

```
#include <stdio.h>
#include <stdlib.h>

#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

#include <wiringPi.h>
#include <wiringPiI2C.h>

#define TEMP_PATH "/sys/class/thermal/thermal_zone0/temp"
#define MAX_SIZE 20
```

3.2 Define RGB related parameters and I2C related parameters

```
#define Max_LED 3

void setRGB(int num, int R, int G, int B);
void closeRGB();

int fd_i2c = -1;
```

3.3 Define CPU temperature related parameters

```
int fd_temp;
double temp = 0, level_temp = 0;
char buf[MAX_SIZE];

wiringPiSetup();
fd_i2c = wiringPiI2CSetup(0x0d);
if (fd_i2c < 0)
{
    fprintf(stderr, "fail to init I2C\n");
    return -1;
}
closeRGB();
```

3.4 Cycle through the CPU temperature file and save **fd_temp**, if the open fails it will return -1 . Then, read the temperature, save it to buf, and return -1 if it fails.

When the temperature reading is successfully saved to the buf, since the value is relatively large, divide by 1000 to get the current temperature, the unit is Celsius, and save to temp. Run the close() function to close the file each time the file is read.

```
while (1)
{
    fd_temp = open(TEMP_PATH, O_RDONLY);
    if (fd_temp < 0)
    {
        fprintf(stderr, "fail to open thermal_zone0/temp\n");
        return -1;
    }

    if (read(fd_temp, buf, MAX_SIZE) < 0)
    {
        fprintf(stderr, "fail to read temp\n");
        return -1;
    }

    temp = atoi(buf) / 1000.0;
    printf("temp: %.1fC\n", temp);
    close(fd_temp);
```

3.5 Get the temperature, then judge the temperature value and modify the fan speed. It can be modified according to actual needs. The RGB light color can be searched online to view the RGB color comparison table.

```

if (abs(temp - level_temp) >= 1)
{
    if (temp <= 45)
    {
        level_temp = 45;
        setRGB(Max_LED, 0x00, 0x00, 0xff);
    }
    else if (temp <= 47)
    {
        level_temp = 47;
        setRGB(Max_LED, 0x1e, 0x90, 0xff);
    }
    else if (temp <= 49)
    {
        level_temp = 49;
        setRGB(Max_LED, 0x00, 0xb7, 0xff);
    }
    else if (temp <= 51)
    {
        level_temp = 51;
        setRGB(Max_LED, 0x5f, 0x9e, 0xa0);
    }

    else if (temp <= 53)
    {
        level_temp = 53;
        setRGB(Max_LED, 0xff, 0xff, 0x00);
    }
}

```

3.6 Set RGB and close RGB

```

void setRGB(int num, int R, int G, int B)
{
    if (num >= Max_LED)
    {
        wiringPiI2CWriteReg8(fd_i2c, 0x00, 0xff);
        wiringPiI2CWriteReg8(fd_i2c, 0x01, R);
        wiringPiI2CWriteReg8(fd_i2c, 0x02, G);
        wiringPiI2CWriteReg8(fd_i2c, 0x03, B);
    }
    else if (num >= 0)
    {
        wiringPiI2CWriteReg8(fd_i2c, 0x00, num);
        wiringPiI2CWriteReg8(fd_i2c, 0x01, R);
        wiringPiI2CWriteReg8(fd_i2c, 0x02, G);
        wiringPiI2CWriteReg8(fd_i2c, 0x03, B);
    }
}

void closeRGB()
{
    wiringPiI2CWriteReg8(fd_i2c, 0x07, 0x00);
    delay(10);
}

```