3 Get CPU temperature,Adjust fan speed

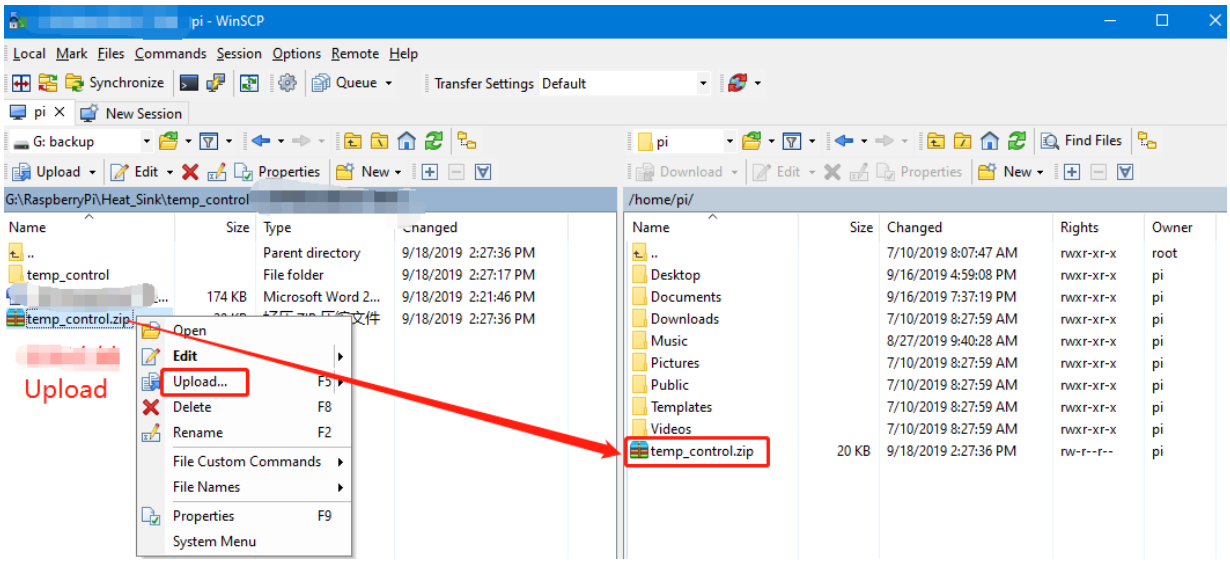
The Raspberry Pi RGB_Cooling_HAT needs to be properly plugged into the GPIO port of the Raspberry Pi and open the Raspberry Pi system **I2C** function.

The experimental phenomenon is to read and print the Raspberry Pi CPU temperature, and adjust the speed of the fan rotation according to the temperature. The higher the temperature, the faster the rotation speed.

1. File transfer

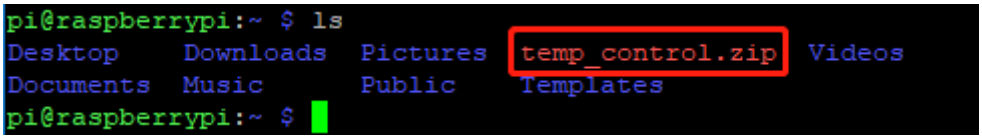
1.1 Install **WinSCP** tool on the computer side, connect the Raspberry Pi and transfer the **temp_control.zip** package to the pi directory of the Raspberry Pi.

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:



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/ssd1306_i2c.c
  inflating: temp_control/ssd1306_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         oled          rgb.c          ssd1306_i2c.h  temp_control.c
fan.c       oled.c        rgb_effect     start.desktop
fan_temp    oled_fonts.h  rgb_effect.c   start.sh
fan_temp.c  rgb           ssd1306_i2c.c  temp_control
pi@raspberrypi:~/temp_control $
```

2.2 Input command to compile:

gcc -o fan_temp fan_temp.c -lwiringPi

```
pi@raspberrypi:~/temp_control $ gcc -o fan_temp fan_temp.c -lwiringPi
fan_temp.c: In function 'main':
fan_temp.c:44:13: warning: implicit declaration of function 'read'; did you mean
 'fread'? [-Wimplicit-function-declaration]
         if (read(fd_temp, buf, MAX_SIZE) < 0)
             ^~~~
             fread
fan_temp.c:53:9: 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         oled          rgb.c          ssd1306_i2c.h  temp_control.c
fan.c       oled.c        rgb_effect     start.desktop
fan_temp    oled_fonts.h  rgb_effect.c   start.sh
fan_temp.c  rgb           ssd1306_i2c.c  temp_control
pi@raspberrypi:~/temp_control $
```

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

2.3 Input command to run the program

./fan_temp

```
pi@raspberrypi:~/temp_control $ ./fan_temp
temp: 44.8C
temp: 44.8C
temp: 44.3C
temp: 44.8C
temp: 45.8C
temp: 45.8C
temp: 46.3C
temp: 45.3C
temp: 46.7C
temp: 45.8C
temp: 46.7C
```

We can see the Raspberry Pi CPU temperature will be printed, and adjust the speed of the fan rotation according to the temperature. The higher the temperature, the faster the rotation speed.

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 CPU temperature related parameters and I2C related parameters in the main function

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

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

3.3 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);
```

3.4 Get the temperature. Next, judge the temperature value and modify the fan speed. Can be modified according to actual needs.

```
if (abs(temp - level_temp) >= 1)
{
    if (temp <= 45)
    {
        level_temp = 45;
        wiringPiI2CWriteReg8(fd_i2c, 0x08, 0x00);
    }
    else if (temp <= 47)
    {
        level_temp = 47;
        wiringPiI2CWriteReg8(fd_i2c, 0x08, 0x04);
    }
    else if (temp <= 49)
    {
        level_temp = 49;
        wiringPiI2CWriteReg8(fd_i2c, 0x08, 0x06);
    }
    else if (temp <= 51)
    {
        level_temp = 51;
        wiringPiI2CWriteReg8(fd_i2c, 0x08, 0x08);
    }
    else if (temp <= 53)
    {
        level_temp = 53;
        wiringPiI2CWriteReg8(fd_i2c, 0x08, 0x09);
    }
    else
    {
        level_temp = 55;
        wiringPiI2CWriteReg8(fd_i2c, 0x08, 0x01);
    }
```