2- Control Fan

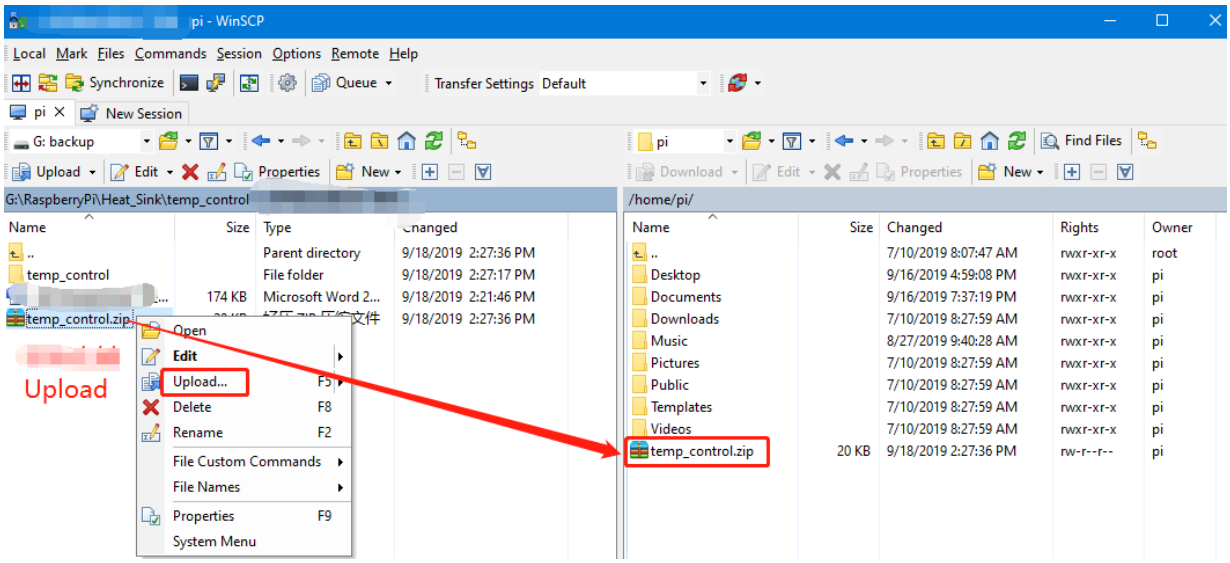
The Raspberry Pi RGB_Cooling_HAT needs to be properly plugged into the GPIO port of the Raspberry Pi and open the Raspberry Pi system **I2C** function.

This experimental phenomenon shows that after 2s, the fan speed is increased every second, next, it will run for 2 seconds with the highest speed, finally, it stops again and keep looping in this state.

1. File transfer

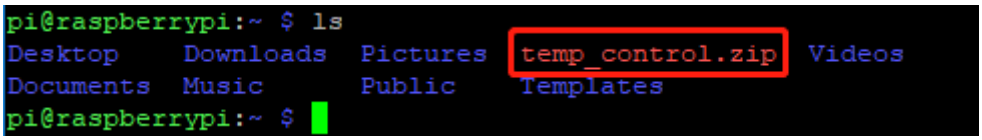
1.1 Install **WinSCP** tool on the computer side, connect the Raspberry Pi and transfer the **temp_control.zip** package to the pi directory of the Raspberry Pi.

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 fan.c -lwiringPi

```
pi@raspberrypi:~/temp_control $ gcc -o fan fan.c -lwiringPi
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** is the generated file name, **fan.c** is the source program, and **-lwiringPi** is the wiringPi library that references the Raspberry Pi.

2.3 Input command to run the program

./fan

```
pi@raspberrypi:~/temp_control $ ./fan
```

After 2s, the fan speed is increased every second, next, it will run for 2 seconds with the highest speed, finally, it stops again and keep looping in this state.

3. About code

3.1 Initialize the Raspberry Pi I2C configuration

```c
#include <stdio.h>
#include <wiringPi.h>
#include <wiringPiI2C.h>

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

3.2 Cycle control fan speed, according to the protocol, we can know that grade of fan speed:
0x00: Close fan,
0x01: full speed,
0x02: 20% speed,
0x03: 30% speed,
...,
0x09: 90% speed

```
while (1)
{
    switch (state)
    {
    case 0:
        wiringPiI2CWriteReg8(fd_i2c, 0x08, 0x00);
        break;
    case 1:
        wiringPiI2CWriteReg8(fd_i2c, 0x08, 0x02);
        break;
    case 2:
        wiringPiI2CWriteReg8(fd_i2c, 0x08, 0x03);
        break;
    case 3:
        wiringPiI2CWriteReg8(fd_i2c, 0x08, 0x04);
        break;
    case 4:
        wiringPiI2CWriteReg8(fd_i2c, 0x08, 0x05);
        break;
    case 5:
        wiringPiI2CWriteReg8(fd_i2c, 0x08, 0x06);
        break;
    case 6:
        wiringPiI2CWriteReg8(fd_i2c, 0x08, 0x07);
        break;
    case 7:
        wiringPiI2CWriteReg8(fd_i2c, 0x08, 0x08);
        break;
    case 8:
        wiringPiI2CWriteReg8(fd_i2c, 0x08, 0x09);
        break;
    case 9:
        wiringPiI2CWriteReg8(fd_i2c, 0x08, 0x01);
        break;
    default:
        break;
    }
```

3.3Limit the state size, set to 0 when greater than 9 to achieve loop effect

```
if (state == 0)
{
    delay(1000);
}

state++;

if (state > 9)
{
    delay(1000);
    state = 0;
}

delay(1000);
```