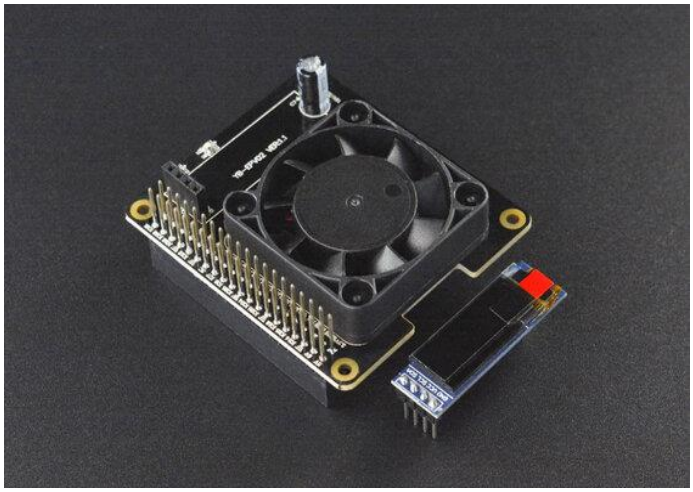
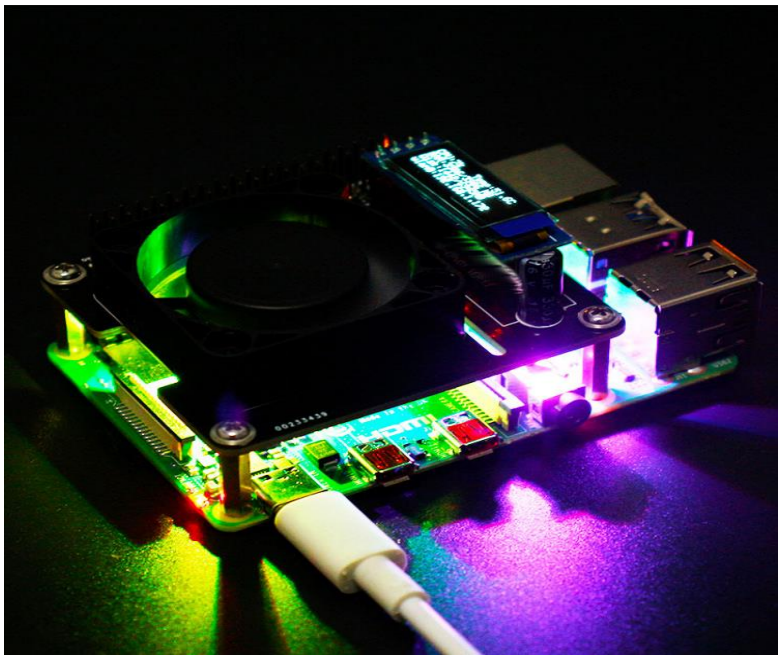
DFROBOT
DRIVE THE FUTURE



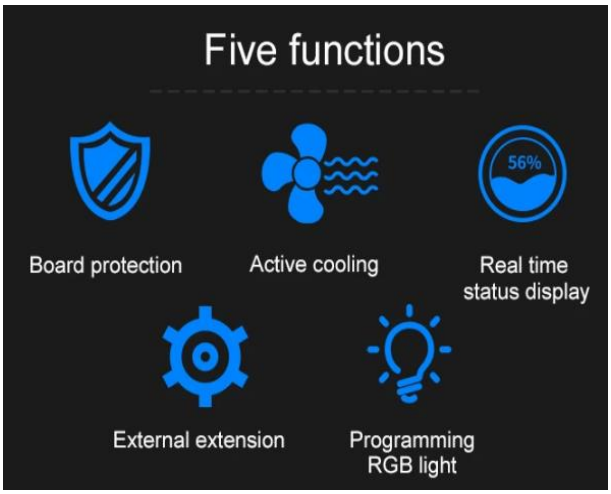
SKU:DFR0672

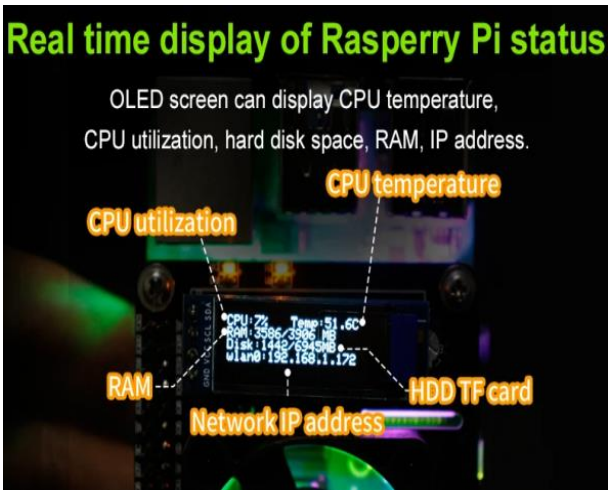
Introduction

This is a multi-function cooling expansion board specifically designed for Raspberry Pi. It provides full compatibility with Pi 4B/3B+/3B to protect Raspberry Pi board and extends its lifespan.

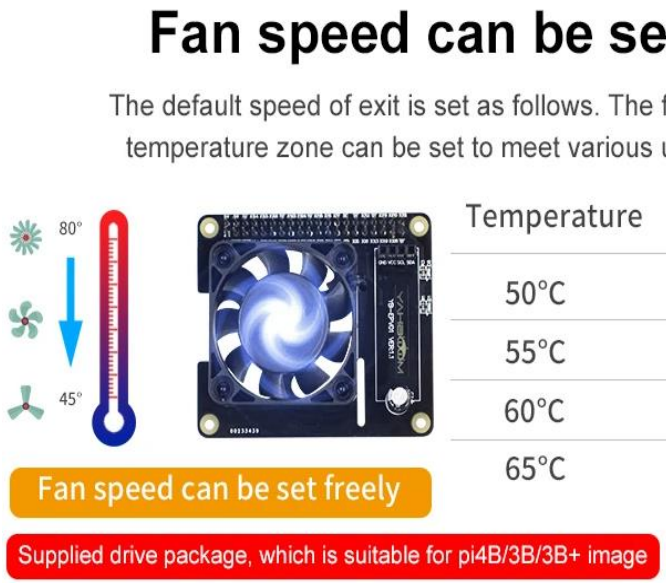


There is a large-size cooling fan on the board with strong wind power that can make Raspberry Pi run more stably by automatically adjusting its speed according to the CPU temperature. Three high-brightness RGB programming LEDs on the bottom of the expansion board can be used to display various lighting effects: waterfall lighting, breathing lights, rainbow lamp and more. This board can be directly plugged onto Raspberry Pi board via its 40 pin interfaces, and it has expanded 40 pin header for connecting to other devices without affecting the use of GPIO ports on Pi board. Furthermore, the place for OLED screen is reserved on the board to display CPU temperature, CPU usage, hard disk space, memory and IP address in real-time.





This fan can effectively reduce the board temperature by about 15℃ in actual tests.



Note: Raspberry Pi board is not included.

Specification

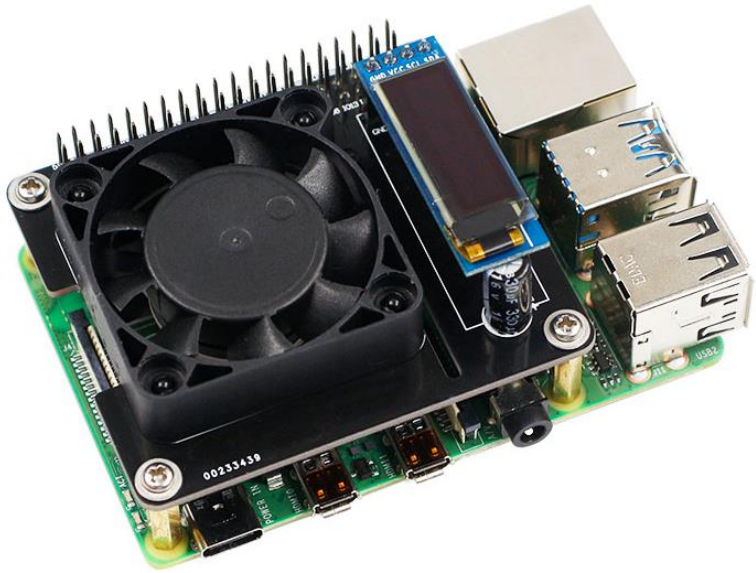
- Support OLED display, can display the running status of Raspberry Pi in real-time.
- The fan on the board can automatically adjust its speed according to the CPU temperature.
- Perfectly compatible with Raspberry Pi 4B/3B+/3B, not occupy any Raspberry Pi pin or heat sink space.
- Interface: directly plug into Raspberry Pi board IO ports
- Dimension: 655*25mm/2.562.170.98”
- Weight: 34.1g

Tutorial

Requirements

- **Hardware**
  - Raspberry Pi x1
  - Network Cable x1
  - Micro SD Card x1
  - Raspberry Pi Adapter x1
  - HDMI Display x1
  - HDMI Connector x1
  - Smart Cooling Hat For Raspberry Pi 4B x1

Plug the fan hat into the Raspberry Pi board's GPIO and install the screws before we are ready to install and configure Raspberry Pi system.

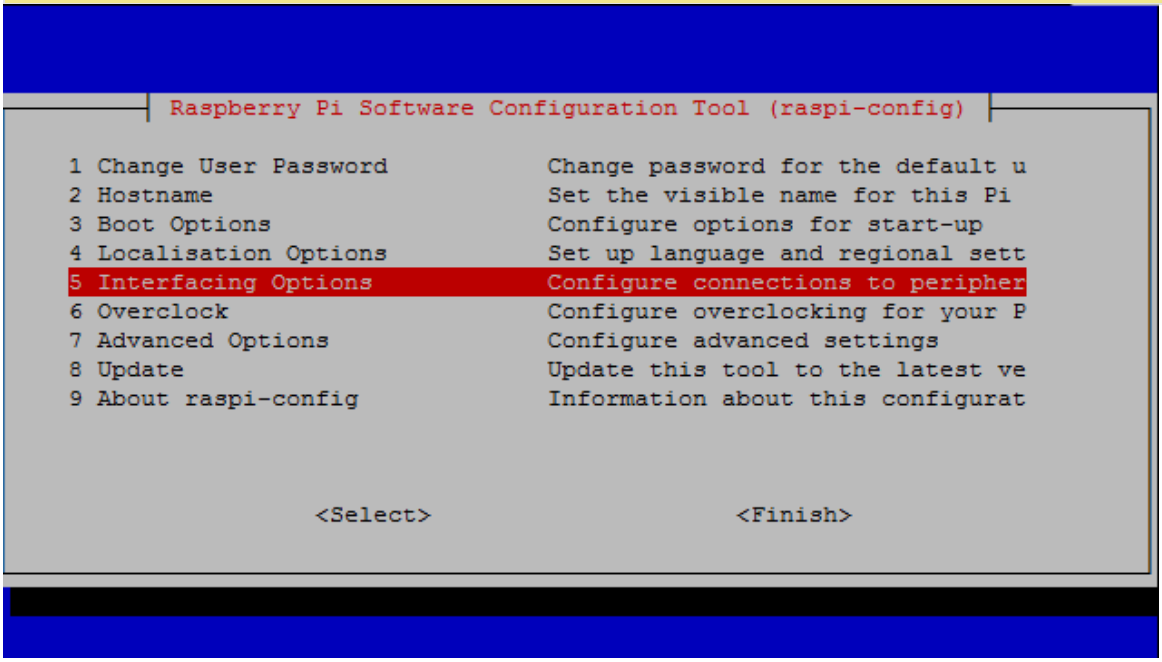


Raspberry Pi System Installation and Configuration

For detailed information about how to install Raspberry Pi system, please refer to the official document: Install raspberrypi guide

Power on to start Raspberry Pi system. The Camera and I2C are all in disabled state for the initial Pi system so we have to enable them mannually. Run the command:

```
sudo raspi-config
```



Select option 5, enable I2C and then restart.

Software Download

There are two ways to download and transfer software packages to Raspberry Pi

- Run the command:

```
git clone https://github.com/OldWang-666/temp_control.git
```

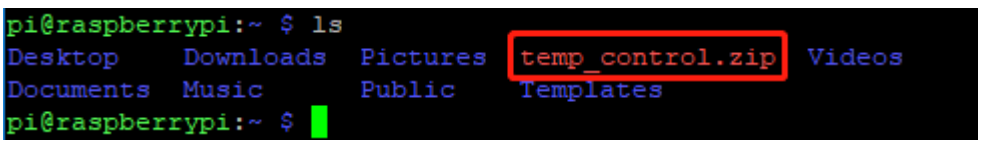
Copy the files into Raspberry Pi system directly.

- Click the link to download software package

The software package can be tansferred into Raspberry Pi system via samba or copied into the system using U-disk.

Unzip the software package

Open Raspberry Pi terminal, find the temp_control.zip file we tranferred into just now.



Input the following command to unzip the 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:~ $
```

How to use software package

1. Control a fan

- Enter a file folder and check the files inside it.

```
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 $
```

- Compile program files

```
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 $
```

Invoke gcc compiler. -o means to generate files, and will be followed by the name of generated file. fan.c is source program, and -lwiringPi means to use the wiringPi library in Raspberry Pi.

- Run program

```
./fan
```

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

The fan will start to rotate in 2s, and its speed will increase every second. When reaching to the highest speed, it will run for two seconds and then stop, and so on.

- Codes Analysis

1. Init Raspberry Pi I2C configuration.

```c
#include <stdio.h>
//Import wiringPi / I2C library
#include <wiringPi.h>
#include <wiringPiI2C.h>

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

2. Control fan speed in loop. The fan speed level can be known from the protocol: 0x00: close; 0x01: full speed; 0x02: 20%speed; 0x03: 30%speed;....0x09: 90%speed.

```
// Let state increase by 1 in loop, meanwhile,  send a command to adjust fan
while (1)                                                          speed.
{
    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. Limit "state" within 9. When it is more than 9, set it to 0 to realize the loop.

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

state++;

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

delay(1000);
```

2. Read CPU temperature, and adjust fan speed.

- Enter a file folder and check the files inside it.

```
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 $
```

- Compile program files

```
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); //Close the file
        ^~~~~
        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 $
```

Invoke gcc compiler. -o means to generate files, and will be followed by the name of generated file. fan.c is source program, and -lwiringPi means to use the wiringPi library in Raspberry Pi.

- 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
```

The current CPU tempterature will be printed on the Raspberry Pi terminal, and the fan speed increases as the temperature goes up.

- Codes Analysis

1. First, import file control library and I2C library. The path to check Raspberry Pi's CPU temperature is defined as TEMP_PATH.

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

// Import file control function library
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

// Import wiringPi / I2C library
#include <wiringPi.h>
#include <wiringPiI2C.h>

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

2. Define parameters related to CPU temperature and I2C.

```c
// Define parameters related to CPU temperature
 int fd_temp;
 double temp = 0, level_temp = 0;
 char buf[MAX_SIZE];

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

3. Open CPU temperature file circularly and save fd_Temp (In Linux, everything is a file). If it fails to open, return - 1. Next, read the temperature and save it in buf. also returns - 1 when failed. After the temperature reading is saved to buf successfully, it need to be divided by 1000 because the value is relatively large, then the current temperature can be obtained in centigrade, and save it to temp. Run the close() function to close the file manually after reading the file.

```c
while (1)
{
    fd_temp = open(TEMP_PATH, O_RDONLY);
    // Judge whether the  file has been opened properly
    if (fd_temp < 0)
    {
        fprintf(stderr, "fail to open thermal_zone0/temp\n");
        return -1;
    }

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

    // Convert format, output  number with 2 digits after decimal point
    temp = atoi(buf) / 1000.0;
    printf("temp: %.1fC\n", temp);
    close(fd_temp); // Close the file
```

4. Get temperature, judge CPU's temperature value, and revise fan speed(can be changed 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);
    }
```

3. Turn on RGB light

- Enter a file folder and check the files inside it.

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 $
```

- Compile program files

gcc -o rgb rgb.c -lwiringPi

```
pi@raspberrypi:~/temp_control $ gcc -o rgb rgb.c -lwiringPi
pi@raspberrypi:~/temp_control $ ls
fan       fan_temp.c  oled_fonts.h  rgb_effect     ssd1306_i2c.h  temp_control
fan.c     oled        rgb           rgb_effect.c   start.desktop  temp_control.c
fan_temp  oled.c      rgb.c         ssd1306_i2c.c  start.sh
pi@raspberrypi:~/temp_control $
```

Invoke gcc compiler. -o means to generate files, and will be followed by the name of generated file. rgb.c is source program, and -lwiringPi means to use the wiringPi library in Raspberry Pi.

- Run the program

```
./rgb
```

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

Three RGB LEDs light up in green at the same time.

- Codes Analysis

1. There are three RGB LEDs on the board, so we define the number of light as 3, declare setRGB and closeRGB function.

```
#define Max_LED 3

int fd_i2c;
void setRGB(int num, int R, int G, int B);
void closeRGB();
```

2. void setRGB(int num, int R, int G, int B) function:

Set RGB LED color. num refers to the LED number: o is the first LED, 1 for the second LED, 2 for the third LED. When num is over 3, it means to set all the LEDs. The range of R, G, B is 0~255.

```
// num=(0~3),R=(0~255),G=(0~255),B=(0~255)
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);
    }
}
```

3. Turn off RGB LED. According to the protocol, the register to turn off RGB is 0x07, data 0x00.

```
// Turn off RGB LED
void closeRGB()
{
    wiringPiI2CWriteReg8(fd_i2c, 0x07, 0x00);
}
```

4. Init I2C configuration in main function.

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

5. Turn off the RGB light before setting the RGB light, otherwise, the display effect may be influenced. The effect of setRGB can be set by yourself. Here, for example, all the LEDs are set to be ON in green.

```
closeRGB();
delay(1);
setRGB(Max_LED, 0, 255, 0);
```

4. Program RGB lights to change color with CPU temperature

- Enter a file folder and check the files inside it.

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 $
```

- Compile program files

```
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); // Close the files
         ^~~~~
         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    ssd1306_i2c.c  temp_control
fan_temp.c  oled_fonts.h  rgb_effect.c  ssd1306_i2c.h  temp_control.c
pi@raspberrypi:~/temp_control $
```

Invoke gcc compiler. -o means to generate files, and will be followed by the name of generated file. rgb_temp.c is source program, and -lwiringPi means to use the wiringPi library in Raspberry Pi.

- Run the progam

```
./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 prints out the current CPU temperature, and the RGB LEDs changes its color with the CPU temperature.

- Codes Analysis

1. Import file control library and I2C library, the path to check CPU temperature of Raspberry Pi is defined as TEMP_PATH.

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

// Import file control function library
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
// Import wiringPi /I2C library
#include <wiringPi.h>
#include <wiringPiI2C.h>

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

2. Define RGB LED and I2C parameters

```
#define Max_LED  3

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

int fd_i2c = -1;
```

3. Define temperature related parameters

```
// Define CPU related parameters
int fd_temp;
double temp = 0, level_temp = 0;
char buf[MAX_SIZE];

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

4. Open CPU temperature file circularly and save fd_Temp (In Linux, everything is a file). If it fails to open, return - 1. Next, read the temperature and save it in buf. also returns - 1 when failed. After the temperature reading is saved to buf successfully, it need to be divided by 1000 because the value is relatively large, then the current temperature can be obtained in centigrade, and save it to temp. Run the close() function to close the file manually after reading the file.

```c
while (1)
{
    fd_temp = open(TEMP_PATH, O_RDONLY);
    // Judge whether the  file has been opened properly
    if (fd_temp < 0)
    {
        fprintf(stderr, "fail to open thermal_zone0/temp\n");
        return -1;
    }

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

    // Convert format, output  number with 2 digits after decimal point
    temp = atoi(buf) / 1000.0;
    printf("temp: %.1fC\n", temp);
    close(fd_temp); // Close the file
```

5. After the current temperature is obtained, determine the temperature value and then revise the fan speed (can be changed according to actual use). Search the corresponding color table online for the RGB LEDs.

```c
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, 0xbf, 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);
```

6. Set the state of RGB LEDs.

```
// num=(0~3),R=(0~255),G=(0~255),B=(0~255)
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);
    }
}
```

5. RGB LED Display Effects

- Enter a file folder and check the files inside it.

```
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 $
```

- Compile program files

```
gcc -o rgb_effect rgb_effect.c -lwiringPi
```

```
pi@raspberrypi:~/temp_control $ gcc -o rgb rgb.c -lwiringPi
pi@raspberrypi:~/temp_control $ ls
fan         fan_temp.c  oled_fonts.h  rgb_effect     ssd1306_i2c.h  temp_control
fan.c       oled        rgb           rgb_effect.c   start.desktop  temp_control.c
fan_temp    oled.c      rgb.c         ssd1306_i2c.c  start.sh
pi@raspberrypi:~/temp_control $
```

Invoke gcc compiler. -o means to generate files, and will be followed by the name of generated file. rgb_effect.c is source program, and -lwiringPi means to use the wiringPi library in Raspberry Pi.

- Run the programs

```
./rgb_effect
```

Three RGB LEDs display breathing effect in purple at the same time.

- Code Analysis

1. There are three RGB LEDs on the board, so define the number of LEDs as 3. Define the register address: RGB_Effect is 0x04, RGB_Speed is 0x05, RGB_Color is 0x06. Declare the necessary functions.

```
#define Max_LED    3
#define RGB_Effect 0x04
#define RGB_Speed  0x05
#define RGB_Color  0x06

int fd_i2c;
void setRGB(int num, int R, int G, int B);
void closeRGB();

void setRGBEffect(int effect);
void setRGBSpeed(int speed);
void setRGBColor(int color);
```

2. void setRGB(int num, int R, int G, int B) function:

Set RGB LED color. num refers to the LED number: o is the first LED, 1 for the second LED, 2 for the third one. When num is over 3, it means to set all the LEDs. The range of R, G, B is 0~255.

```
// num=(0~3),R=(0~255),G=(0~255),B=(0~255)
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);
    }
}
```

3. Turn off RGB LED. According to the protocol, the register to turn off RGB is 0x07, data 0x00.

```
// Turn off RGB LED
void closeRGB()
{
    wiringPiI2CWriteReg8(fd_i2c, 0x07, 0x00);
}
```

4. void setRGBEffect(int effect) function:

Determine the input value. Corresponding to the protocol, there are 5 display effects to choose: 0->waterfall, 1->breathing, 2->chasing, 3->rainbow, 4->colorful

```
void setRGBEffect(int effect)
{
    if (effect >= 0 && effect <= 4)
    {
        wiringPiI2CWriteReg8(fd_i2c, RGB_Effect, effect);
    }
}
```

5. void setRGBSpeed(int speed) function:

Revise the switching speed of the LEDs display effects. 1->slow, 2->Middle(default), 3->fast.

```
void setRGBSpeed(int speed)
{
    if (speed >= 1 && speed <= 3)
    {
        wiringPiI2CWriteReg8(fd_i2c, RGB_Speed, speed);
    }
}
```

6. void setRGBColor(int color) function:

Set the color of the LEDs effect in waterfall and breathing mode. 0 for red, 1 for green(default), 2 for blue, 3 for yellow, 4 for purple, 5 for cyan, 6 for white.

```
void setRGBColor(int color)
{
    if (color >= 0 && color <= 6)
    {
        wiringPiI2CWriteReg8(fd_i2c, RGB_Color, color);
    }
}
```

7. Init I2C configuration

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

8. Display fast breathing effect in purple.

```
closeRGB();
delay(1);

setRGBEffect(1);
setRGBSpeed(3);
setRGBColor(4);
```

6. Present Raspberry Pi state on OLED

- Enter a file folder and check the files inside it.

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 $
```

- Compile program files

gcc -o oled oled.c ssd1306_i2c.c -lwiringPi

```
pi@raspberrypi:~/temp_control $ gcc -o oled oled.c ssd1306_i2c.c -lwiringPi
ssd1306_i2c.c: In function 'ssd1306_fillRect':
ssd1306_i2c.c:724:3: warning: implicit declaration of function 'swap_values' [-W
implicit-function-declaration]
   swap_values(x, y);
   ^~~~~~~~~~~
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 $ ./oled
init ok!
```

Invoke gcc compiler. -o means to generate files, and will be followed by the name of generated file. oled.c and ssd1306_i2c.c are the source programs, and -lwiringPi means to use the wiringPi library in Raspberry Pi.

- Run the program

```
./oled
```

```
pi@raspberrypi:~/temp_control $ ./oled
init ok!
```

Now, you can check the CPU utilization, CPU temperature, operatiing memory utilization, disk utilization, IP address and other information of your Pi on the OLED screen.

- Codes Analysis

1. Import wiringPi/I2C library, oled display library, file control library, IP reading library, and disk reading library.

```
// Import wiringPi/I2C library
#include <wiringPi.h>
#include <wiringPiI2C.h>

 // Import oled display library
#include "ssd1306_i2c.h"

 // Import file control library
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/sysinfo.h>
 // IP reading library
#include <ifaddrs.h>
#include <netinet/in.h>
#include <string.h>
#include <arpa/inet.h>
 // Disk reading library
#include <sys/vfs.h>
#include <unistd.h>

#define TEMP_PATH "/sys/class/thermal/thermal_zone0/temp"
#define MAX_SIZE 32
```

2. Define parameters about temperature, system information, disk information, and IP address.

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

// get system usage / info
struct sysinfo sys_info;
struct statfs disk_info;

struct ifaddrs *ifAddrStruct = NULL;
void *tmpAddrPtr = NULL;
getifaddrs(&ifAddrStruct);
```

3. Init oled screen, and output init success information from the terminal.

```
ssd1306_begin(SSD1306_SWITCHCAPVCC, SSD1306_I2C_ADDRESS);
// ssd1306_display();      //show logo
// ssd1306_clearDisplay();
// delay(500);
printf("init ok!\n");
```

4. Read system information. Display sysinfo-Error on the OLED if it fails, and re-read 0.5s later.

```
if (sysinfo(&sys_info) != 0) // sysinfo(&sys_info) != 0
{
    printf("sysinfo-Error\n");
    ssd1306_clearDisplay();
    char *text = "sysinfo-Error";
    ssd1306_drawString(text);
    ssd1306_display();
    delay(500);
    continue;
}
```

5. Read CPU utilization. There is a sprintf function inside for concatenating strings.

```
char CPUInfo[MAX_SIZE];
unsigned long avgCpuLoad = sys_info.loads[0] / 1000;
sprintf(CPUInfo, "CPU:%ld%%", avgCpuLoad);
```

6. Read RAM usage (unit: b). For easy display, it needs to be converted to Mb. Shift the value 20 bits to the right, or written as follows: unsigned long totalRam = sys_info.totalram / 1024 / 1024;

```
char RamInfo[MAX_SIZE];
unsigned long totalRam = sys_info.totalram >> 20;
unsigned long freeRam = sys_info.freeram >> 20;
sprintf(RamInfo, "RAM:%ld/%ld MB", freeRam, totalRam);
```

7. Read IP address of cable network and WiFi, prior to display WiFi IP address.

```
char IPInfo[MAX_SIZE];
while (ifAddrStruct != NULL)
{
    if (ifAddrStruct->ifa_addr->sa_family == AF_INET)
    { // check it is IP4 is a valid IP4 Address
        tmpAddrPtr = &((struct sockaddr_in *)ifAddrStruct->ifa_addr)->sin_addr;
        char addressBuffer[INET_ADDRSTRLEN];
        inet_ntop(AF_INET, tmpAddrPtr, addressBuffer, INET_ADDRSTRLEN);

        if (strcmp(ifAddrStruct->ifa_name, "eth0") == 0)
        {
            sprintf(IPInfo, "eth0:IP:%s", addressBuffer);
            break;
        }
        else if (strcmp(ifAddrStruct->ifa_name, "wlan0") == 0)
        {
            sprintf(IPInfo, "wlan0:%s", addressBuffer);
            break;
        }
    }
    ifAddrStruct = ifAddrStruct->ifa_next;
}
```

8. Read temperature.

```
// Read CPU temperature
char CPUTemp[MAX_SIZE];
fd_temp = open(TEMP_PATH, O_RDONLY);
if (fd_temp < 0)
{
    temp = 0;
    printf("failed to open thermal_zone0/temp\n");
}
else
{
     // Read temperature and determine
    if (read(fd_temp, buf, MAX_SIZE) < 0)
    {
        temp = 0;
        printf("fail to read temp\n");
    }
    else
    {
         // Convert to floating number and print
        temp = atoi(buf) / 1000.0;
        // printf("temp: %.1f\n", temp);
        sprintf(CPUTemp, "Temp:%.1fC", temp);
    }
}
close(fd_temp);
```

9. Read disk space.

```
char DiskInfo[MAX_SIZE];
statfs("/", &disk_info);
unsigned long long totalBlocks = disk_info.f_bsize;
unsigned long long totalSize = totalBlocks * disk_info.f_blocks;
size_t mbTotalsize = totalSize >> 20;
unsigned long long freeDisk = disk_info.f_bfree * totalBlocks;
size_t mbFreedisk = freeDisk >> 20;
sprintf(DiskInfo, "Disk:%ld/%ldMB", mbFreedisk, mbTotalsize);
```

10. Display information on OLED.

The function ssd1306_drawText(int x, int y, char *str) is used to set the content displayed on OLED. The first parameter is x, which controls the left and right offset. The second one is y, which controls the up and down offset. The third is string pointer, also the content to be displayed.

Run the function ssd1306_display() to refresh and display.

```
ssd1306_drawText(0, 0, CPUInfo);
ssd1306_drawText(56, 0, CPUTemp);
ssd1306_drawText(0, 8, RamInfo);
ssd1306_drawText(0, 16, DiskInfo);
ssd1306_drawText(0, 24, IPInfo);


ssd1306_display();
```

7. Raspberry Pi Auto-start

1. Create script start.sh

```
nano /home/pi/temp_control/start.sh
```

Input the following content:

```
#!/bin/sh
sleep 5s
cd /home/pi/temp_control/
./temp_control
```

Press ctrl+X+Y to save, and tap button "Enter".

2. Create auto-start program

Input the following command to open .config file:

```
cd /home/pi/.config
```

Create autostart file, skip this step if it already exists:

```
mkdir autostart
```

Enter autostart file:

```
cd autostart
```

Create autostart shortcut

```
nano start.desktop
```

Then, input the following content.

[Desktop Entry]

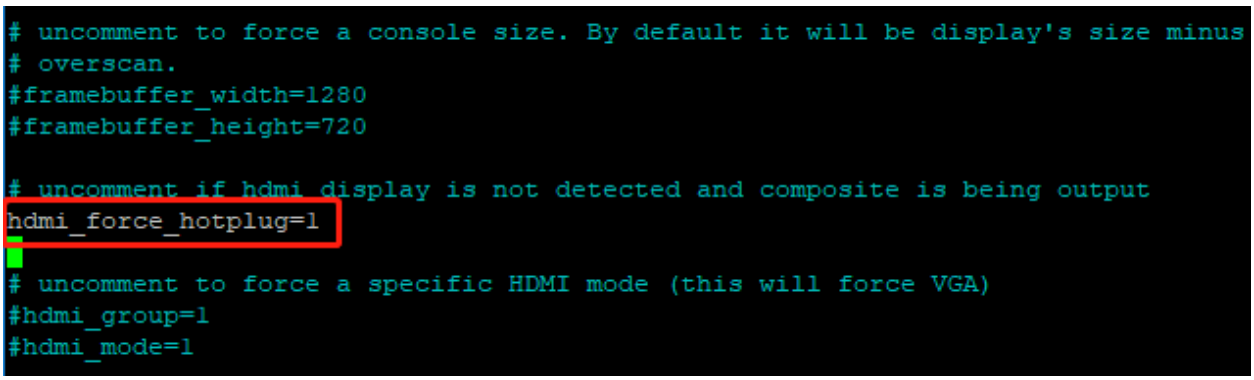
Type=Application

Exec=sh /home/pi/cpu_show_v3/start.sh

Press ctrl+X+Y to save, and tap button "Enter".

Exec=startup command among them.

This autostart method can only be started after the desktop is started, so it would take some time. If it cannot be started automatically after adding library, pleas check whether there is a "#" in front of hdmi_force_hotplug=1 in /boot/config.txt file, if so, delete it, shown as below:



2. Restart Raspberry Pi

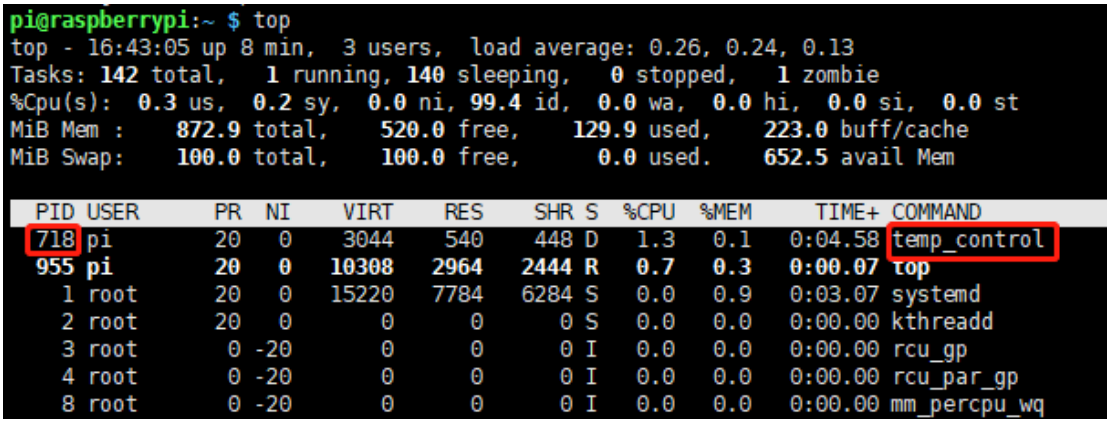
Input the following command into the terminal to restart Raspberry Pi. The temp_control program will be auto started after that. Then, the fan, RGB LEDs, and OLED will response accordingly.

```
sudo reboot
```

3. Exit the program

Since the autostart program is running in the background, we cannot directly exit the program at the opened terminal. If we need to revise the program, but worry that the background process would interfere with the debug results, what should we do? Well, it is easy, just check the process number(PID) of the background program, and then end the process.

1. Input top into the terminal to open process table.

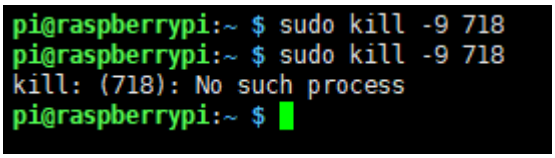


The system will list out the processes that take up more CPU resources, and sometimes it may be necessary to wait for a while. From the figure above, we can see that the PID of the autostart program temp_control is718. Now, just kill it, and press Ctrl+C to exit top.

2. End the process

```
sudo kill -9 PID
```

For example, run the command sudo kill -9 718 to end the temp_control process, then it will prompt the process does not exist when re-running.



3. Restart and Run in background

If we have ended a process running in the background, but don't want to restart the background program, we can restart the Raspberry Pi first of course. However, for saving our time, we can directly add a symbol "&" behind the running program.

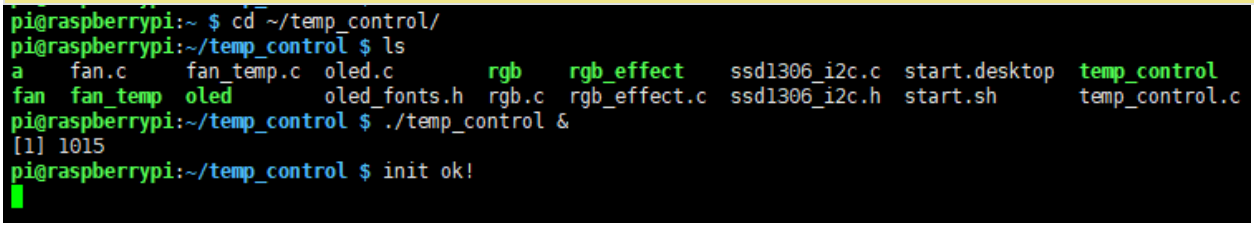
For instance, we need to run the temp_control program in the background:

First of all, enter the target file holder.

```
cd ~/temp_control
```

Add a & at the end of the command to indicate it runs in the background.

```
./temp_control &
```



Then the system will prompt the PID(1015) of the process.

Press Ctrl+C again. Now we can input other commands in the terminal with the program running in the background.

FAQ

For any questions, advice or cool ideas to share, please visit the **DFRobot Forum**