

3.Display_usage_and_CPU_temperature

After CPU Info LCD screen is correctly inserted into the Raspberry Pi, you need to compile and run the program to display it normally. This experiment is used to display the current CPU usage and CPU temperature of the Raspberry Pi.

1. Install the wiringPi library

CPU Info LCD screen is used for data communication through the GPIO port of the Raspberry Pi, so we must install the wiringPi library file.

Enter the following command to install the wiringPi library. Users who have already installed the wiringPi library can ignore this step.

```
cd ~
```

```
git clone git://git.drogon.net/wiringPi
```

```
cd wiringPi
```

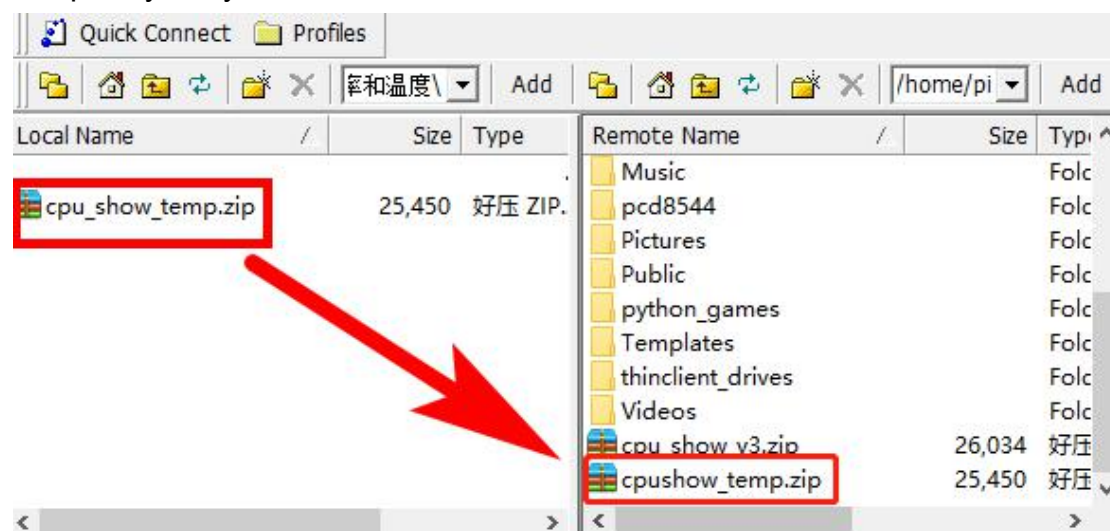
```
./build
```

2.Install Drive

2.1 Transfer the driver file to the Raspberry Pi

You need to install the SSH Secure Shell Client tool on your computer. After connecting to the Raspberry Pi, transfer the [cpu_show_temp.zip](#) package from this folder to the pi directory of the Raspberry Pi.

As shown below, drag and drop [cpu_show_temp.zip](#) directly into the Raspberry Pi system.



2.2 Extract file

Open the Raspberry Pi terminal and find the [cpu_show_temp.zip](#) file.

Enter command:

```
ls
```

```
pi@raspberrypi:~ $ ls
cpu_show_temp.zip  Documents      Music          Templates
cpu_show_uptime.zip Downloads      pcd8544        thinclient_drives
cpu_show_v3        LCD-show      Pictures       Videos
cpu_show_v3.zip    MagPi         Public
Desktop            matchbox-keyboard python_games
pi@raspberrypi:~ $
```

Enter command:

unzip cpu_show_temp.zip

```
pi@raspberrypi:~ $ unzip cpu_show_temp.zip
Archive:  cpu_show_temp.zip
  creating: cpu_show_temp/
  inflating: cpu_show_temp/PCD8544.h
  inflating: cpu_show_temp/cpushow_temp
  inflating: cpu_show_temp/README.txt
  creating: cpu_show_temp/BL/
  inflating: cpu_show_temp/BL/bl
  inflating: cpu_show_temp/BL/test.c
  inflating: cpu_show_temp/pcd8544_rpi.c
  inflating: cpu_show_temp/PCD8544.c
  creating: cpu_show_temp/cputemp/
  inflating: cpu_show_temp/cputemp/cputemp.c
  inflating: cpu_show_temp/cputemp/temp
pi@raspberrypi:~ $
```

2.3 Enter the program folder

Enter command:

cd ~/cpu_show_temp

ls

```
pi@raspberrypi:~/cpu_show_temp $ ls
BL  cpushow_temp  cputemp  PCD8544.c  PCD8544.h  pcd8544_rpi.c  README.txt
pi@raspberrypi:~/cpu_show_temp $
```

2.4 Compiler file

Enter command:

cc -o cpushow_temp pcd8544_rpi.c PCD8544.c -L/usr/local/lib -lwiringPi

```
pi@raspberrypi:~/cpu_show_temp $ cc -o cpushow_temp pcd8544_rpi.c PCD8544.c -L/usr/local/lib -lwiringPi
pcd8544_rpi.c: In function 'main':
pcd8544_rpi.c:71:2: warning: implicit declaration of function 'getifaddrs' [-Wimplicit-function-declaration]
  getifaddrs(&ifAddrStruct);
  ^~~~~~
pcd8544_rpi.c:131:6: warning: implicit declaration of function 'read' [-Wimplicit-function-declaration]
  if (read(fd, buf, MAX_SIZE) < 0)
  ^~~~~
pcd8544_rpi.c:145:2: warning: implicit declaration of function 'close' [-Wimplicit-function-declaration]
  close(fd);
  ^~~~~
pi@raspberrypi:~/cpu_show_temp $
```

cc is the compile command, -o is the compile parameter, cpushow_temp is the generated program name, pcd8544_rpi.c and PCD8544.c are the source files in the current directory, -L/usr/local/lib and -lwiringPi are referenced libraries file.

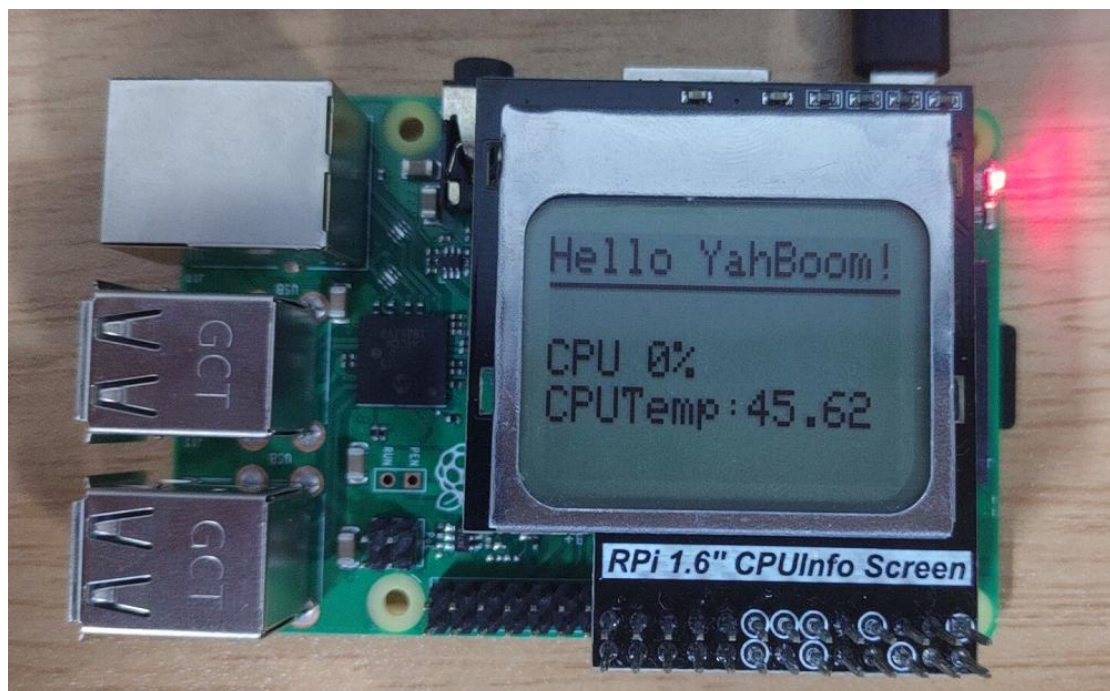
2.5 Running procedure

Enter command:

sudo ./cpushow_temp

```
pi@raspberrypi:~/cpu_show_temp $ sudo ./cpushow_temp
Raspberry Pi PCD8544 sysinfo display
=====
temp: 46.16
temp: 46.16
temp: 46.16
temp: 46.16
temp: 46.16
temp: 46.16
temp: 46.16
temp: 45.62
temp: 46.70
temp: 45.62
```

The system will print the current CPU temperature value and display the following on the CPU Info screen.



3. Code analysis

Enter command:

nano pcd8544_rpi.c

This command is to open pcd8544_rpi.c

1. The following sections are pin settings. The corresponding relationship of the GPIO ports has been indicated on the back of the LCD.

```
// pin setup
int _din = 1;
int _sclk = 0;
int _dc = 2;
int _rst = 4;
int _cs = 3;
```

2.Main function

```
int main (void)
{
    struct ifaddrs * ifAddrStruct=NULL;
    void * tmpAddrPtr=NULL;

    getifaddrs (&ifAddrStruct);

    // print infos
    printf("Raspberry Pi PCD8544 sysinfo display\n");
    printf("=====\n");

    // check wiringPi setup
    if (wiringPiSetup() == -1)
    {
        printf("wiringPi-Error\n");
        exit(1);
    }

    // init and clear lcd
    LCDInit(_sclk, _din, _dc, _cs, _rst, contrast);
    LCDclear();
```

3.The front part is the initialize program and the prompt information; the latter part is a for loop, which is display CPU usage.

```
for (;;)
{
    // clear lcd
    LCDclear();

    // get system usage / info
    struct sysinfo sys_info;
    if(sysinfo(&sys_info) != 0)
    {
        printf("sysinfo-Error\n");
    }

    // cpu info
    char cpuInfo[10];
    unsigned long avgCpuLoad = sys_info.loads[0] / 1000;
    sprintf(cpuInfo, "CPU %ld%%", avgCpuLoad);
```

Get information about the Raspberry Pi, prompt the user if an error occurs

Get the current usage of the Raspberry Pi CPU, set its format and store it in the cpuInfo array

`sprintf (cpuInfo, "CPU %ld%%", avgCpuLoad)` is a splicing function, replace

the value of avgCpuLoad to the %ld position, and then save it to cpuInfo. If the value of avgCpuLoad is 2, the result of output cpuInfo is CPU 2%.

4. Get CPU temperature

```
char CPUTemp[15];
{
    int i;
    for(i=0;i<15;i++)
    {
        CPUTemp[i]=0;
    }
}

int fd;
double temp = 0;
char buf[MAX_SIZE];

fd = open(TEMP_PATH, O_RDONLY);
if (fd < 0)
{
    printf("failed to open thermal_zone0/temp\n");
}

// read data
if (read(fd, buf, MAX_SIZE) < 0)
{
    printf("failed to read temp\n");
}

temp = atoi(buf) / 1000.0;
printf("temp: %.2f\n", temp);

sprintf(CPUTemp, "CPUTemp: %.2f", temp);
```

Define an array of strings to hold temperature data

Get CPU temperature, and save it to CPUTemp array

5. LCD display

```
// build screen

LCDdrawstring(0, 1, "Hello YahBoom!");
LCDdrawline(0, 10, 83, 10, BLACK);
LCDdrawstring(0, 21, cpuInfo);
LCDdrawstring(0, 30, CPUTemp);

LCDdisplay();

delay(1000);
```

LCDdrawstring(0, 1, "Hello YahBoom!") meaning is first row, second line display 'Hello YahBoom!'. First parameter is 0, which meaning is starting from the first column on the left. Second parameter is 1, which meaning is starting from the second column count from above. Third parameter is "hello YahBoom!", which meaning is data we need to display.

Note: If you have added the boot-up display program, please edit the **rc.local file comment or delete the code related to the display. After restarting, close the running driver and then operate.**

If the program that has already been run is not closed, the new program will run and the screen will always change due to the conflict.

The method of modification is as follows:

sudo nano /etc/rc.local

You need to shield the program, which related to the cpu Info LCD (add a # in front of the code to shield the code)

```
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
    printf "My IP address is %s\n" "$_IP"
fi

#sudo /home/pi/cpu_show_v3/cpu_show/cpushow
exit 0
```

Enter command:

sudo reboot

This command is to restart the Raspberry Pi.