



Sense HAT (B)

用户手册

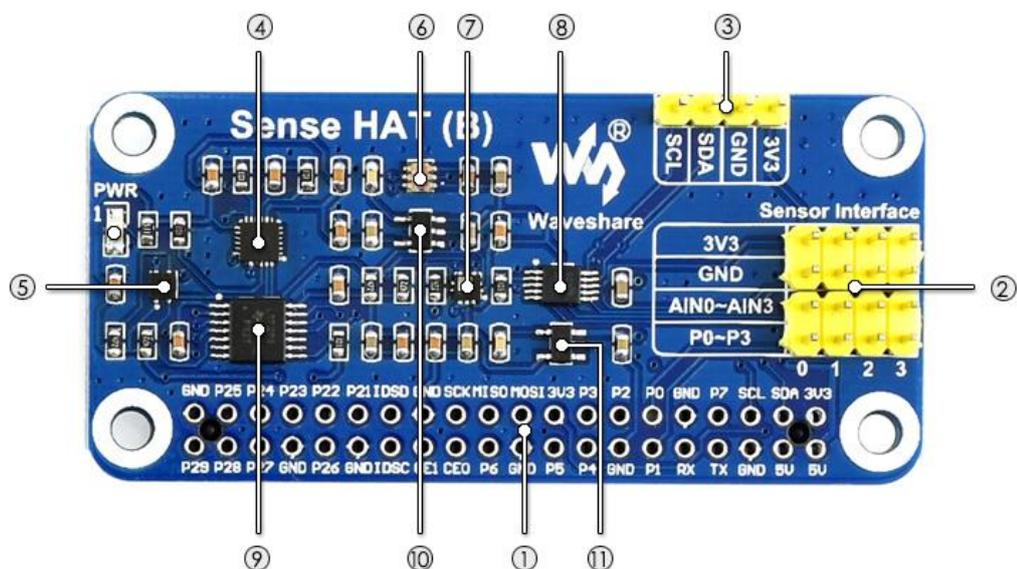
产品概述

我是专为树莓派设计的传感器扩展板，板载了陀螺仪、加速度计、磁力计、气压计和温湿度传感器等，I2C 接口通信，支持外接更多传感器。

如果你想把树莓派 DIY 成可以检测运动姿态、方位的机器人，或者想让树莓派采集周围环境的温湿度、大气压强等传感器数据，那就带上我吧。

产品特点

- 板载 Raspberry Pi 40pin GPIO 接口，适用于 Raspberry Pi 系列主板
- 板载 ICM20948(3 轴加速度、3 轴陀螺仪和 3 轴磁力计)，可检测运动姿态、方位和磁场
- 板载 SHTC3 数字温湿度传感器，可感知环境的温度和湿度
- 板载 LPS22HB 大气压强传感器，可感知环境的大气压强
- 板载 TCS34725 颜色识别传感器，可识别周围物体的颜色
- 板载 ADS1015 芯片，4 通道 12 位精度 ADC，可扩展 AD 功能以便接入更多传感器
- 引出 I2C 控制接口，方便接入 STM32 等主控板
- 提供完善的配套资料手册(Raspberry/STM32 等示例程序)



[接口简介]

- | | |
|--|--|
| <p>1. Raspberry Pi GPIO 接口
方便接入树莓派</p> <p>2. 传感器接口
方便接入各类传感器</p> <p>3. I2C 扩展接口
方便接入 Arduino/STM32 等主控板</p> | <p>6. TCS34725
颜色识别传感器</p> <p>7. LPS22HB
大气压强传感器</p> <p>8. ADS1015
12 位精度 AD 转换芯片</p> |
|--|--|

[器件简介]

- | | |
|---|---|
| <p>4. ICM-20948
9 轴运动传感器</p> <p>5. SHTC3
温湿度传感器</p> | <p>9. LSF0204PWR
4 路电平转换芯片</p> <p>10. RT9193-18
1.8V 线性稳压芯片</p> <p>11. RT9193-33
3.3V 线性稳压芯片</p> |
|---|---|

产品参数

工作电压:	3.3V
通信接口:	I2C
逻辑电压:	3.3V
产品尺寸:	65 x 30.5(mm)
加速度计特性:	分辨率: 16 位 量程(可选): ± 2 、 ± 4 、 ± 8 、 $\pm 16g$
陀螺仪特性:	分辨率: 16 位 量程(可选): ± 250 、 ± 500 、 ± 1000 、 $\pm 2000^\circ/\text{sec}$
磁力计特性:	分辨率: 16 位 量程: $\pm 4900\mu\text{T}$
气压计特性:	分辨率: 24 位压力数据, 16 位温度数据 测量精度 (常温下): $\pm 0.025\text{hPa}$ 测量范围: 260 ~ 1260 hPa 测量速率: 1 Hz - 75 Hz
温湿度传感器特性:	测量精度 (湿度): $\pm 2\% \text{ rH}$ 测量范围 (湿度): 0% ~ 100% rH 测量精度 (温度): $\pm 0.2^\circ\text{C}$ 测量范围 (湿度): $-30 \sim 100^\circ\text{C}$
颜色识别传感器:	分辨率: 4 通道 RGBC, 每个通道 16 位
AD 转换芯片:	分辨率: 12 位

产品大 PK

PK 项	Sense HAT (B)	树莓派 Sense HAT	备注
陀螺仪	测量范围: $\pm 250/500/1000/2000$ dps 分辨率: 16 位	测量范围: $\pm 245/500/2000$ dps 分辨率: 16 位	B 型角速度可选量程更多
加速度计	测量范围: $\pm 2/4/8/16$ g 分辨率: 16 位	测试范围: $\pm 2/4/8/16$ g 分辨率: 16 位	
磁力计	测量范围: ± 49 gauss 分辨率: 16 位	测量范围: $\pm 4/8/12/16$ gauss 分辨率: 16 位	B 型磁力测量范围更广
气压计	测量范围: 260 ~ 1260 hPa 测量精度 (常温下): ± 0.025 hPa 测量速率: 1 Hz - 75 Hz	测量范围: 260 ~ 1260 hPa 测量精度 (常温下): ± 0.1 hPa 测量速率: 1 Hz - 25 Hz	B 型大气压测量的精度更高, 速率更快
温湿度传感器	测量精度 (湿度): $\pm 2\%$ rH 测量范围 (湿度): 0% ~ 100% rH 测量精度 (温度): $\pm 0.2^\circ\text{C}$ 测量范围 (温度): $-30 \sim 100^\circ\text{C}$	测量精度: $\pm 4.5\%$ rH 测量范围: 20% ~ 80% rH 测量精度 (温度): $\pm 0.5^\circ\text{C}$ 测量范围 (温度): $15 \sim 40^\circ\text{C}$	B 型温湿度测量的精度更高, 范围更广
其他	颜色识别传感器 高精度 12 位 AD 转换芯片	8x8 RGB LED 矩阵 五向摇杆	B 型板载颜色传感器, 支持 AD 接入更多传感器

目录

产品概述	1
产品特点	1
产品参数	3
产品大 PK	4
使用指南	7
下载例程	7
树莓派例程使用	7
拷贝到树莓派	7
安装函数库	8
组装	10
1. ICM20948 示例程序 -9 轴传感器演示	11
1.1. bcm2835 程序	11
1.2. wiringPi 程序	11
1.3. python 程序	12
1.4. STM32 程序	12
2. LPS22HBTR 示例程序 -气压传感器演示	13
2.1. bcm2835 程序	13
2.2. wiringPi 程序	14
2.3. python 程序	15
2.4. STM32 程序	15
3. SHTC3 示例程序 -温湿度传感器演示	16

3.1.	bcm2835 程序	16
3.2.	wiringPi 程序.....	17
3.3.	STM32 程序	18
4.	TC34725 示例程序 -颜色识别传感器演示.....	19
4.1.	bcm2835 程序	19
4.2.	wiringPi 程序.....	20
4.3.	python 程序.....	20
4.4.	STM32 程序	21
5.	ADS1015 示例程序 -AD 转换演示.....	23
5.1.	bcm2835 程序	23
5.2.	wiringPi 程序.....	24
5.3.	STM32 程序	25
	常见问题	26

使用指南

下载例程

在微雪电子官网上找到对应产品，在产品资料打开下载路径，在 wiki 中下载示例程序：

文档

- [用户手册](#)
- [原理图](#)

程序

- [示例程序](#)

将下载下来的解压包解压，得到如下文件：

	ADS1015	2019/6/27 16:18	文件夹
	ICM-20948	2019/6/27 16:18	文件夹
	LPS22HB	2019/6/27 16:18	文件夹
	SHTC3	2019/6/27 16:18	文件夹
	TC34725	2019/6/27 16:18	文件夹

ADS1015: AD 转换例程 (STM32 , BCM2835, WringPi 和 Python 四种例程)

ICM-20948: 9 轴传感器例程 (STM32 , BCM2835, WringPi 和 Python 四种例程)

LPS22HB: 气压传感器例程(STM32 , BCM2835, WringPi 和 Python 四种例程)

SHTC3: 温湿度传感器例程(STM32 , BCM2835 和 WringPi 三种例程)

TC34725:颜色识别传感器例程(STM32 , BCM2835, WringPi 和 Python 四种例程)

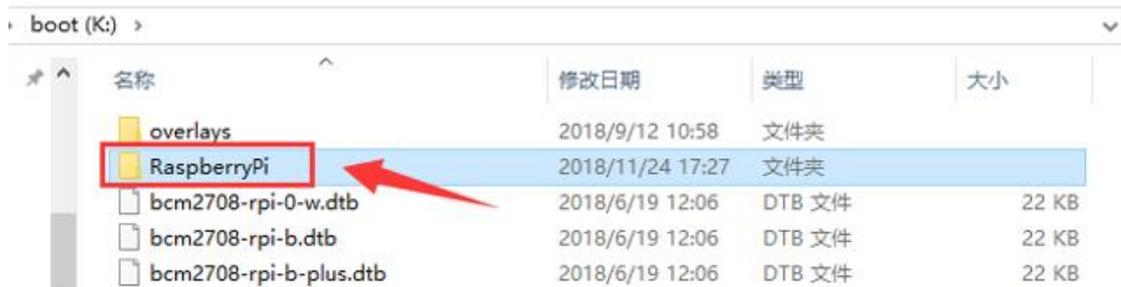
树莓派例程使用

拷贝到树莓派

1. 使用读卡器将 SD 卡插入电脑，将会显示一个名为 Boot 的可移动盘。



2. 将解压文件中 RaspberryPi 文件夹复制到 boot 根目录下



3. 然后弹出 U 盘，将 SD 卡插入树莓派中，插上 USB 上电，查看/boot 目录的文件

```
pi@raspberrypi:~$ ls /boot/
bcm2708-rpi-0-w.dtb  bcm2710-rpi-3-b.dtb  config.txt  fixup_x.dat  kernel.img  start_cd.elf
bcm2708-rpi-b.dtb  bcm2710-rpi-3-b-plus.dtb  COPYING.Linux  FSCK0000.REC  LICENCE.broadcom  start_db.elf
bcm2708-rpi-b-plus.dtb  bcm2710-rpi-cm3.dtb  fixup_cd.dat  FSCK0001.REC  LICENCE.oracle  start.elf
bcm2708-rpi-cm.dtb  bootcode.bin  fixup.dat  issue.txt  overlays  start_x.elf
bcm2709-rpi-2-b.dtb  cmdline.txt  fixup_db.dat  kernel7.img  RaspberryPi  System Volume Information
```

4. 执行如下命令将其复制到用户目录下，并修改其用户权限

```
sudo cp -r /boot/RaspberryPi/ ./
```

```
sudo chmod 777 -R RaspberryPi/
```

```
pi@raspberrypi:~$ sudo cp -r /boot/RaspberryPi/ ./
pi@raspberrypi:~$ ls
code  libcode  RaspberryPi  RPiLib  ubuntu  usbdisk
pi@raspberrypi:~$ sudo chmod 777 -R RaspberryPi/
pi@raspberrypi:~$ ls
code  libcode  RaspberryPi  RPiLib  ubuntu  usbdisk
```

5. 进入目录，查看文件：

```
pi@raspberrypi:~$ cd RaspberryPi
pi@raspberrypi:~/RaspberryPi$ ls
Light Sensor  Servo Driver  test  web_Python
pi@raspberrypi:~/RaspberryPi$
```

安装函数库

需要安装必要的函数库 (wiringPi、bcm2835、python 库)，否则以下的示例程序可能无法

正常工作。安装方法详见：

安装 BCM2835 库:

<http://www.airspayce.com/mikem/bcm2835/>

进入 BCM2835 的官网下载并把安装包复制到树莓派上, 运行如下:

```
sudo tar zxvf bcm2835-1.xx.tar.gz

cd bcm2835-1.xx

sudo ./configure

make

sudo make check

sudo make install
```

其中 xx 代表的是下载的版本号, 例如我下载的 bcm2835-1.52

那么就应该执行: `sudo tar zxvf bcm2835-1.52.tar.gz`

安装 wiringPi 库:

```
sudo apt-get install git

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

cd wiringPi

sudo ./build
```

安装 python 库:

```
sudo apt-get install python-pip

sudo pip install RPi.GPIO

sudo pip install spidev

sudo apt-get install python-imaging

sudo apt-get install python-smbus
```

开启 I2C 接口:

```
sudo raspi-config
```

```

Raspberry Pi Software Configuration Tool (raspi-config)

1 Change User Password Change password for the current user
2 Network Options      Configure network settings
3 Boot Options         Configure options for start-up
4 Localisation Options Set up language and regional settings to match your location
5 Interfacing Options  Configure connections to peripherals
6 Overclock            Configure overclocking for your Pi
7 Advanced Options    Configure advanced settings
8 Update               Update this tool to the latest version
9 About raspi-config   Information about this configuration tool

<Select>                                <Finish>

```

```

Raspberry Pi Software Configuration Tool (raspi-config)

P1 Camera      Enable/Disable connection to the Raspberry Pi Camera
P2 SSH         Enable/Disable remote command line access to your Pi using SSH
P3 VNC         Enable/Disable graphical remote access to your Pi using RealVNC
P4 SPI         Enable/Disable automatic loading of SPI kernel module
P5 I2C         Enable/Disable automatic loading of I2C kernel module
P6 Serial      Enable/Disable shell and kernel messages on the serial connection
P7 1-Wire      Enable/Disable one-wire interface
P8 Remote GPIO Enable/Disable remote access to GPIO pins

```

组装

注意：组装前进行调试，舵机初始角度不是在起始位置，舵机旋转时可能会卡死，所以建议在第一次使用时先不要组装云台，先单独测试舵机转的角度，防止舵机意外损坏。

硬件连接:

Sense HAT (B)	Raspberry Pi (Board)	Raspberry Pi (BCM)
VCC	3.3V	3.3V
GND	GND	GND

SDA	3	P2
SCL	5	P3

1. ICM20948 示例程序 - 9 轴传感器演示

1.1. bcm2835 程序

进入 Linux 终端，在终端执行以下命令。

编译程序：

```
pi@raspberrypi ~/Sense HAT (B)/ICM-20948/Raspberry Pi/bcm2835 $ make
```

执行程序：

```
pi@raspberrypi ~/Sense HAT (B)/ICM-20948/Raspberry Pi/bcm2835  
$ sudo ./ICM20948_D
```

预期结果：

```
-----/  
Roll: 18.22    Pitch: -10.73    Yaw: -162.27  
Acceleration: X: 64    Y: 1935    Z: 3248  
Gyroscope: X: -2    Y: -1    Z: -2  
Magnetic: X: 3    Y: -55    Z: -22  
-----/
```

按下 **Ctrl+C** 结束程序。

1.2. wiringPi 程序

进入 Linux 终端，在终端执行以下命令。

编译程序：

```
pi@raspberrypi ~/Sense HAT (B)/ICM-20948/Raspberry Pi/wiringPi $ make
```

执行程序：

```
pi@raspberrypi ~/Sense HAT (B)/ICM-20948/Raspberry Pi/wiringPi
$ sudo ./ICM20948_D
```

预期结果：

```
-----/
Roll: 18.22    Pitch: -10.73    Yaw: -162.27
Acceleration: X: 64    Y: 1935    Z: 3248
Gyroscope: X: -2    Y: -1    Z: -2
Magnetic: X: 3    Y: -55    Z: -22
-----/
```

按下 **Ctrl+C** 结束程序。

1.3. python 程序

在终端输入以下命令执行程序：

```
pi@raspberrypi ~/Sense HAT (B)/ICM-20948/Raspberry Pi/python $ sudo python
ICM20948.py
```

预期结果：

```
-----/
Roll: 18.22    Pitch: -10.73    Yaw: -162.27
Acceleration: X: 64    Y: 1935    Z: 3248
Gyroscope: X: -2    Y: -1    Z: -2
Magnetic: X: 3    Y: -55    Z: -22
-----/
```

按下 **Ctrl+C** 结束程序。

1.4. STM32 程序

该例程基于 XNUCLEO-F103RB 开发板，通过串口 2 输出数据。

连线如下：

+5V/+3.3V-----VCC (注意: 跳线帽要跳到相应的位置)

GND-----GND

PB9-----SDA

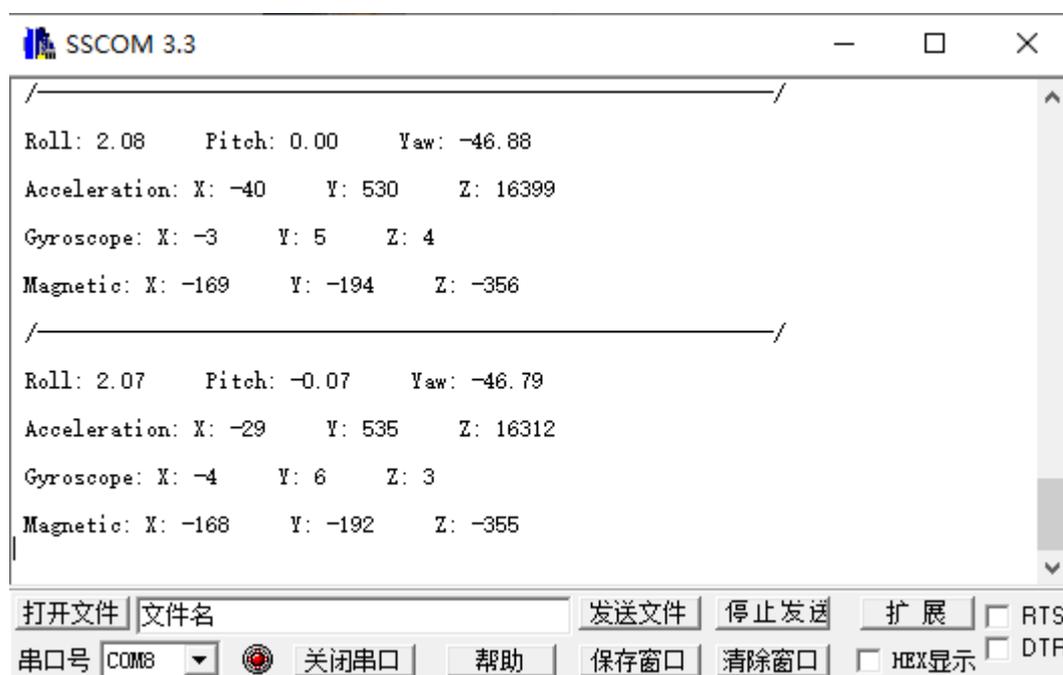
PB8-----SCL

编译并下载程序:



打开串口助手, 设置波特率为 115200

预期结果:



2. LPS22HBTR 示例程序 -气压传感器演示

2.1. bcm2835 程序

进入 Linux 终端，在终端执行以下命令。

编译程序：

```
pi@raspberrypi ~/Sense HAT (B)/LPS22HB/Raspberry Pi/bcm2835 $ make
```

执行程序：

```
pi@raspberrypi ~/Sense HAT (B)/LPS22HB/Raspberry Pi/bcm2835 $ sudo ./LPS22HB
```

预期结果：

```
Pressure Sensor Test Program ...
Pressure Sensor OK
Pressure = 997.15 hPa , Temperature = 31.86 °C
Pressure = 997.15 hPa , Temperature = 31.86 °C
Pressure = 997.17 hPa , Temperature = 31.89 °C
Pressure = 997.17 hPa , Temperature = 31.89 °C
Pressure = 997.11 hPa , Temperature = 31.90 °C
Pressure = 997.11 hPa , Temperature = 31.90 °C
Pressure = 997.13 hPa , Temperature = 31.92 °C
Pressure = 997.13 hPa , Temperature = 31.92 °C
Pressure = 997.10 hPa , Temperature = 31.94 °C
Pressure = 997.10 hPa , Temperature = 31.94 °C
Pressure = 997.11 hPa , Temperature = 31.94 °C
```

按下 **Ctrl+C** 结束程序。

2.2. wiringPi 程序

进入 Linux 终端，在终端执行以下命令。

编译程序：

```
pi@raspberrypi ~/Sense HAT (B)/LPS22HB/Raspberry Pi/wiringPi $ make
```

执行程序：

```
pi@raspberrypi ~/Sense HAT (B)/LPS22HB/Raspberry Pi/wiringPi $ sudo ./LPS22HB
```

预期结果：

```
Pressure Sensor Test Program ...

Pressure Sensor OK
Pressure = 997.15 hPa , Temperature = 31.86 °C
Pressure = 997.15 hPa , Temperature = 31.86 °C
Pressure = 997.17 hPa , Temperature = 31.89 °C
Pressure = 997.17 hPa , Temperature = 31.89 °C
Pressure = 997.11 hPa , Temperature = 31.90 °C
Pressure = 997.11 hPa , Temperature = 31.90 °C
Pressure = 997.13 hPa , Temperature = 31.92 °C
Pressure = 997.13 hPa , Temperature = 31.92 °C
Pressure = 997.10 hPa , Temperature = 31.94 °C
Pressure = 997.10 hPa , Temperature = 31.94 °C
Pressure = 997.11 hPa , Temperature = 31.94 °C
```

按下 **Ctrl+C** 结束程序。

2.3. python 程序

在终端输入以下命令执行程序：

```
pi@raspberrypi ~/ Sense HAT (B)/LPS22HB/Raspberry Pi/python $ sudo python
LPS22HB.py
```

预期结果：

```
pi@raspberrypi:~/Sense/LPS22HB/python $ sudo python LPS22HB.py
Pressure Sensor Test Program ...

Pressure = 997.12 hPa , Temperature = 32.06 °C
Pressure = 997.12 hPa , Temperature = 31.83 °C
Pressure = 997.17 hPa , Temperature = 31.84 °C
Pressure = 997.17 hPa , Temperature = 31.85 °C
Pressure = 997.21 hPa , Temperature = 31.85 °C
Pressure = 997.21 hPa , Temperature = 31.85 °C
Pressure = 997.19 hPa , Temperature = 31.86 °C
Pressure = 997.20 hPa , Temperature = 31.86 °C
Pressure = 997.19 hPa , Temperature = 31.86 °C
Pressure = 997.21 hPa , Temperature = 31.86 °C
```

按下 **Ctrl+C** 结束程序。

2.4. STM32 程序

该例程基于 XNUCLEO-F103RB 开发板，通过串口 2 输出数据。

连线如下：

+5V/+3.3V-----VCC (注意：跳线帽要跳到相应的位置)

GND-----GND

PB9-----SDA

PB8-----SCL

编译并下载程序：



打开串口助手，设置波特率为 115200

预期结果：



3. SHTC3 示例程序 - 温湿度传感器演示

3.1. bcm2835 程序

进入 Linux 终端，在终端执行以下命令。

编译程序：

```
pi@raspberrypi ~/Sense HAT (B)/SHTC3/Raspberry Pi/bcm2835 $ make
```

执行程序：

```
pi@raspberrypi ~/Sense HAT (B)/SHTC3/Raspberry Pi/bcm2835 $ sudo ./SHTC3
```

预期结果：

```
SHTC3 Sensor Test Program ...
Temperature = 31.12°C , Humidity = 70.77%
Temperature = 31.14°C , Humidity = 70.76%
Temperature = 31.12°C , Humidity = 70.74%
Temperature = 31.15°C , Humidity = 70.75%
Temperature = 31.12°C , Humidity = 70.75%
Temperature = 31.16°C , Humidity = 70.76%
Temperature = 31.14°C , Humidity = 70.75%
Temperature = 31.15°C , Humidity = 70.75%
Temperature = 31.12°C , Humidity = 70.75%
Temperature = 31.14°C , Humidity = 70.74%
Temperature = 31.16°C , Humidity = 70.75%
Temperature = 31.17°C , Humidity = 70.76%
Temperature = 31.15°C , Humidity = 70.73%
Temperature = 31.15°C , Humidity = 70.76%
Temperature = 31.13°C , Humidity = 70.74%
Temperature = 31.12°C , Humidity = 70.77%
Temperature = 31.13°C , Humidity = 70.75%
Temperature = 31.13°C , Humidity = 70.75%
```

按下 **Ctrl+C** 结束程序。

3.2. wiringPi 程序

进入 Linux 终端，在终端执行以下命令。

编译程序：

```
pi@raspberrypi ~/Sense HAT (B)/SHTC3/Raspberry Pi/wiringPi $ make
```

执行程序：

```
pi@raspberrypi ~/Sense HAT (B)/SHTC3/Raspberry Pi/wiringPi $ sudo ./SHTC3
```

预期结果:

```

SHTC3 Sensor Test Program ...
Temperature = 31.12°C , Humidity = 70.77%
Temperature = 31.14°C , Humidity = 70.76%
Temperature = 31.12°C , Humidity = 70.74%
Temperature = 31.15°C , Humidity = 70.75%
Temperature = 31.12°C , Humidity = 70.75%
Temperature = 31.16°C , Humidity = 70.76%
Temperature = 31.14°C , Humidity = 70.75%
Temperature = 31.15°C , Humidity = 70.75%
Temperature = 31.12°C , Humidity = 70.75%
Temperature = 31.14°C , Humidity = 70.74%
Temperature = 31.16°C , Humidity = 70.75%
Temperature = 31.17°C , Humidity = 70.76%
Temperature = 31.15°C , Humidity = 70.73%
Temperature = 31.15°C , Humidity = 70.76%
Temperature = 31.13°C , Humidity = 70.74%
Temperature = 31.12°C , Humidity = 70.77%
Temperature = 31.13°C , Humidity = 70.75%
Temperature = 31.13°C , Humidity = 70.75%

```

按下 **Ctrl+C** 结束程序。

3.3. STM32 程序

该例程基于 XNUCLEO-F103RB 开发板，通过串口 2 输出数据。

连线如下:

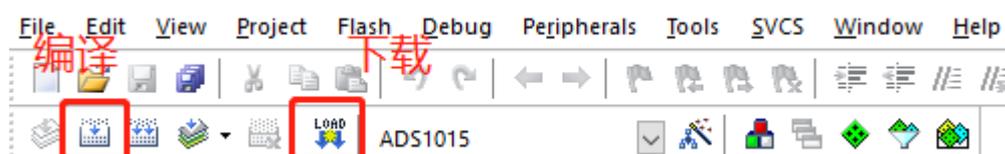
+5V/+3.3V-----VCC (注意: 跳线帽要跳到相应的位置)

GND-----GND

PB9-----SDA

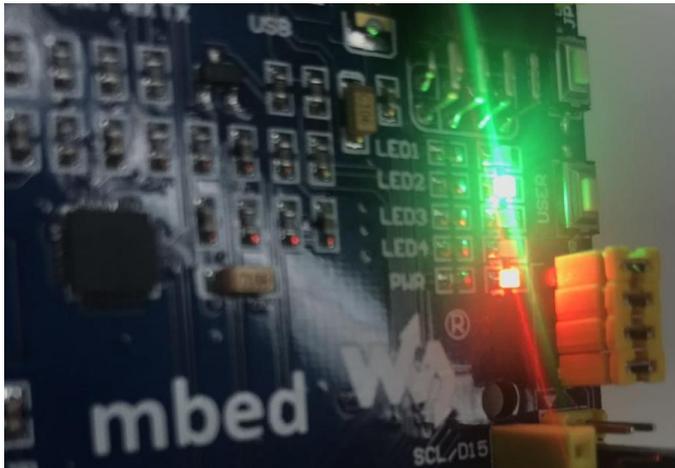
PB8-----SCL

编译并下载程序:



预期结果:

传感器正常无错误，LED2 亮起：



当空气湿度低于 80%时 LED3 不亮。当空气湿度大于或等于 80%时 LED3 亮起。



4. TC34725 示例程序 - 颜色识别传感器演示

4.1. bcm2835 程序

进入 Linux 终端，在终端执行以下命令。

编译程序：

```
pi@raspberrypi ~/Sense HAT (B)/TC34725/Raspberry Pi/bcm2835 $ make
```

执行程序：

```
pi@raspberrypi ~/Sense HAT (B)/TC34725/Raspberry Pi/bcm2835  
$ sudo ./TC34725
```

预期结果：

```
TCS34725 initialization success!!
RGB888 :R=0 G=0 B=0  RGB888=0X0  RGB565=0X0  Lux_Interrupt = 0
RGB888 :R=0 G=0 B=0  RGB888=0X0  RGB565=0X0  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=201  RGB888=0XF1F9C9  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=249 B=201  RGB888=0XF1F9C9  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=201  RGB888=0XF1F8C9  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=201  RGB888=0XF1F8C9  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=201  RGB888=0XF1F8C9  RGB565=0XF7D9  Lux_Interrupt = 0
```

按下 **Ctrl+C** 结束程序。

4.2. wiringPi 程序

进入 Linux 终端，在终端执行以下命令。

编译程序：

```
pi@raspberrypi ~/Sense HAT (B)/TC34725/Raspberry Pi/wiringPi $ make
```

执行程序：

```
pi@raspberrypi ~/Sense HAT (B)/TC34725/Raspberry Pi/wiringPi $ sudo ./
TC34725
```

预期结果：

```
TCS34725 initialization success!!
RGB888 :R=0 G=0 B=0  RGB888=0X0  RGB565=0X0  Lux_Interrupt = 0
RGB888 :R=0 G=0 B=0  RGB888=0X0  RGB565=0X0  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=249 B=201  RGB888=0XF1F9C9  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=249 B=201  RGB888=0XF1F9C9  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=201  RGB888=0XF1F8C9  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=201  RGB888=0XF1F8C9  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200  RGB888=0XF1F8C8  RGB565=0XF7D9  Lux_Interrupt = 0
RGB888 :R=241 G=248 B=201  RGB888=0XF1F8C9  RGB565=0XF7D9  Lux_Interrupt = 0
```

按下 **Ctrl+C** 结束程序。

4.3. python 程序

在终端输入以下命令执行程序：

```
pi@raspberrypi ~/ Sense HAT (B)/TC34725/Raspberry Pi/python $ sudo python
TC34725.py
```

预期结果：

```
TC34725 initialization success!!
RGB888 :R=0 G=0 B=0 RGB888=0X0 RGB565=0X0 Lux_Interrupt = 0
RGB888 :R=0 G=0 B=0 RGB888=0X0 RGB565=0X0 Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200 RGB888=0XF1F8C8 RGB565=0XF7D9 Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200 RGB888=0XF1F8C8 RGB565=0XF7D9 Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200 RGB888=0XF1F8C8 RGB565=0XF7D9 Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200 RGB888=0XF1F8C8 RGB565=0XF7D9 Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200 RGB888=0XF1F8C8 RGB565=0XF7D9 Lux_Interrupt = 0
RGB888 :R=241 G=248 B=201 RGB888=0XF1F8C9 RGB565=0XF7D9 Lux_Interrupt = 0
RGB888 :R=241 G=249 B=201 RGB888=0XF1F9C9 RGB565=0XF7D9 Lux_Interrupt = 0
RGB888 :R=241 G=249 B=201 RGB888=0XF1F9C9 RGB565=0XF7D9 Lux_Interrupt = 0
RGB888 :R=241 G=248 B=201 RGB888=0XF1F8C9 RGB565=0XF7D9 Lux_Interrupt = 0
RGB888 :R=241 G=248 B=201 RGB888=0XF1F8C9 RGB565=0XF7D9 Lux_Interrupt = 0
RGB888 :R=241 G=248 B=200 RGB888=0XF1F8C8 RGB565=0XF7D9 Lux_Interrupt = 0
RGB888 :R=241 G=248 B=201 RGB888=0XF1F8C9 RGB565=0XF7D9 Lux_Interrupt = 0
```

按下 **Ctrl+C** 结束程序。

4.4. STM32 程序

该例程基于 XNUCLEO-F103RB 开发板，通过串口 2 输出数据。

连线如下：

+5V/+3.3V-----VCC (注意：跳线帽要跳到相应的位置)

GND-----GND

PB9-----SDA

PB8-----SCL

编译并下载程序：



打开串口助手，设置波特率为 115200

预期结果：



这数据怎么转换成颜色呢？下面介绍一个工具，复制到浏览器打开即可

<https://www.sioe.cn/yinyong/yanse-rgb-16/>

也可以下载

<http://www.waveshare.net/w/upload/0/05/Hexacolor3.7z>





5. ADS1015 示例程序 -AD 转换演示

5.1. bcm2835 程序

进入 Linux 终端，在终端执行以下命令。

编译程序：

```
pi@raspberrypi ~/Sense HAT (B)/ADS1015/Raspberry Pi/bcm2835 $ make
```

执行程序：

```
pi@raspberrypi ~/Sense HAT (B)/ADS1015/Raspberry Pi/bcm2835 $ sudo ./AD
```

预期结果：

```
AIN0 = 1186(2372mv) ,AIN1 = 0(0mv) ,AIN2 = 261(522mv) AIN3 = 276(552mv)
AIN0 = 1462(2924mv) ,AIN1 = 0(0mv) ,AIN2 = 263(526mv) AIN3 = 280(560mv)
AIN0 = 1462(2924mv) ,AIN1 = 0(0mv) ,AIN2 = 266(532mv) AIN3 = 262(524mv)
AIN0 = 1655(3310mv) ,AIN1 = 1(2mv) ,AIN2 = 264(528mv) AIN3 = 261(522mv)
AIN0 = 1654(3308mv) ,AIN1 = 0(0mv) ,AIN2 = 260(520mv) AIN3 = 266(532mv)
AIN0 = 1654(3308mv) ,AIN1 = 0(0mv) ,AIN2 = 259(518mv) AIN3 = 278(556mv)
AIN0 = 1655(3310mv) ,AIN1 = 0(0mv) ,AIN2 = 260(520mv) AIN3 = 276(552mv)
AIN0 = 1655(3310mv) ,AIN1 = 0(0mv) ,AIN2 = 264(528mv) AIN3 = 260(520mv)
AIN0 = 1655(3310mv) ,AIN1 = 0(0mv) ,AIN2 = 267(534mv) AIN3 = 261(522mv)
AIN0 = 1654(3308mv) ,AIN1 = 0(0mv) ,AIN2 = 262(524mv) AIN3 = 272(544mv)
AIN0 = 1654(3308mv) ,AIN1 = 0(0mv) ,AIN2 = 260(520mv) AIN3 = 282(564mv)
```

按下 **Ctrl+C** 结束程序。

5.2. wiringPi 程序

进入 Linux 终端，在终端执行以下命令。

编译程序：

```
pi@raspberrypi ~/Sense HAT (B)/ADS1015/Raspberry Pi/wiringPi $ make
```

执行程序：

```
pi@raspberrypi ~/Sense HAT (B)/ADS1015/Raspberry Pi/wiringPi $ sudo ./AD
```

预期结果:

```
AIN0 = 1186(2372mv) ,AIN1 = 0(0mv) ,AIN2 = 261(522mv) AIN3 = 276(552mv)
AIN0 = 1462(2924mv) ,AIN1 = 0(0mv) ,AIN2 = 263(526mv) AIN3 = 280(560mv)
AIN0 = 1462(2924mv) ,AIN1 = 0(0mv) ,AIN2 = 266(532mv) AIN3 = 262(524mv)
AIN0 = 1655(3310mv) ,AIN1 = 1(2mv) ,AIN2 = 264(528mv) AIN3 = 261(522mv)
AIN0 = 1654(3308mv) ,AIN1 = 0(0mv) ,AIN2 = 260(520mv) AIN3 = 266(532mv)
AIN0 = 1654(3308mv) ,AIN1 = 0(0mv) ,AIN2 = 259(518mv) AIN3 = 278(556mv)
AIN0 = 1655(3310mv) ,AIN1 = 0(0mv) ,AIN2 = 260(520mv) AIN3 = 276(552mv)
AIN0 = 1655(3310mv) ,AIN1 = 0(0mv) ,AIN2 = 264(528mv) AIN3 = 260(520mv)
AIN0 = 1655(3310mv) ,AIN1 = 0(0mv) ,AIN2 = 267(534mv) AIN3 = 261(522mv)
AIN0 = 1654(3308mv) ,AIN1 = 0(0mv) ,AIN2 = 262(524mv) AIN3 = 272(544mv)
AIN0 = 1654(3308mv) ,AIN1 = 0(0mv) ,AIN2 = 260(520mv) AIN3 = 282(564mv)
```

按下 **Ctrl+C** 结束程序。

5.3. STM32 程序

该例程基于 XNUCLEO-F103RB 开发板, 通过串口 2 输出数据。

连线如下:

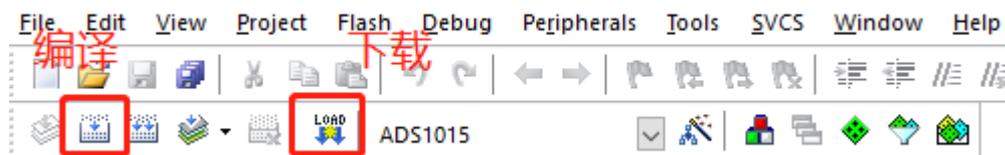
+5V/+3.3V-----VCC (注意: 跳线帽要跳到相应的位置)

GND-----GND

PB9-----SDA

PB8-----SCL

编译并下载程序：



打开串口助手，设置波特率为 115200

预期结果：



常见问题

1. 树莓派例程初始化失败？（以下以 TC34725 颜色识别传感器为例）

答：对于 BCM2835 和 wiringPi 例程出现这样的提示，

```
bcm2835 init success !!!
TCS34725 initialization error!!
```

Python 例程

```
Traceback (most recent call last):
  File "main.py", line 28, in <module>
    GPIO.cleanup()
NameError: name 'GPIO' is not defined
```

如果出现以上问题这是设备数据 I2C 数据传输错误。大多数是硬件连接错误，请检查硬件连接是否正确，检查硬件连接是否有问题，运行 `i2cdetect -y 1` 如果有显示 IIC 地址就表示硬件连接无问题。

如果硬件连接正确那么是不正确的使用树莓派控制可能会导致（详情看下面），重启树莓派即可。

```
pi@raspberrypi:~$ i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  29  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  48  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  5c  --  --  --
60:  --  --  --  --  --  --  68  --  --  --  --  --  --  --  --
70: 70  --  --  --  --  --  --  --  --  --  --  --  --  --  --
```

2. 不正确的使用树莓派控制可能会导致？

答：如果运行 `wiringPi` 例程正常，再运行 `python` 或者 `BCM2835` 可能会屏幕无法正常刷新，因为 `bcm2835` 库是树莓派 `cpu` 芯片的库函数，底层是直接操作寄存器，而 `wiringPi` 库和 `python` 的底层都是通过读写 `linux` 系统的设备文件操作设备，可能导致 `GPIO` 口异常，重启树莓派可完美解决。

3. STM32 例程串口输出没有数据或者数据输出乱码？

答：确认波特率是否设置为 115200，对于 STM32 例程请确认电脑正确连接开发板 `USART2` (`PA2,PA3`)，`PA2` 为 `TXD`，并且选择正确的 `COM` 端口。控制面板->硬件->设备管理器。



4. STM32 例程串口输出数据全部为 0 或者初始化失败？如图。

答：请确认器件连接没有问题，如果没问题请按下复位按钮。

```
RGB888 :R=0 G=0 B=0
RGB888=0X0 RGB565=0X0
Lux_Interrupt = 0

RGB888 :R=0 G=0 B=0
RGB888=0X0 RGB565=0X0
Lux_Interrupt = 0

RGB888 :R=0 G=0 B=0
RGB888=0X0 RGB565=0X0
Lux_Interrupt = 0

TCS34725 initialization error!!
```

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for [Development Boards & Kits - ARM category](#):

Click to view products by [Waveshare manufacturer](#):

Other Similar products are found below :

[CY4541](#) [OM13090UL](#) [Raspberry Pi 4 Model B,8GB](#) [YR0K77210B000BE](#) [B-U585I-IOT02A](#) [NUCLEO-C031C6](#) [NUCLEO-U5A5ZJ-Q](#)
[NUCLEO-WL55JC1](#) [STM32MP135F-DK](#) [ZDSD-Pinboard](#) [081ZYKFB](#) [LKS32MC034DOF6Q8-k](#) [LKS32MC077MBS8-K](#)
[LKS32MC038Y6P8B-K](#) [LKS32MC071DOC8T8-K](#) [LKS32MC074DOF8Q8-K](#) [LKS32MC038Y6P8-k](#) [Ai-WB2-32S-Kit](#) [GD32E103T-START](#)
[RTK7F124FPC01000BJ](#) [XDS601](#) [RP2040-Tiny](#) [LKS32MC033H6P8B-K](#) [VC-02-Kit_EN](#) [Ra-08H-Kit](#) [Hi-12FL-Kit](#) [PB-03M-Kit](#) [Ai-WB2-](#)
[13-Kit](#) [PB-03F-Kit](#) [Ra-08-Kit](#) [Hi-07SL-Kit](#) [Hi-07S-Kit](#) [PB-03-Kit](#) [Hi-12F-Kit](#) [AT-START-F407](#) [FT8132Q-3HALL-FOC+EMF](#) [FU6832L-](#)
[TGB-DEMO](#) [APM32F072VBT6](#) [APM32F091VC MINI](#) [APM32F003F6P6-MINIBOARD](#) [APM32F051R8 MINI](#) [GD32EPRTV-START](#)
[GD32E507R-START](#) [GD32E103R-START](#) [EPC1EVK-ECGPPG\(FS\)](#) [NS4EVKA-LC](#) [ENS1EVKD](#) [.ENS1EVKB](#) [ENS1EVKE](#) [HLK-7621-](#)
[ALL-SUIT](#)