TCS34725 Color Sensor

用户手册

产品概述

本模块是 ams AG 的 TCS34725FN 彩色光数字转换器为核心的颜色传感器，传感器提供红色，绿色，蓝色（RGB）和清晰光感应值的数字输出。集成红外阻挡滤光片可最大限度地减少入射光的红外光谱成分，并可精确地进行颜色测量。具有高灵敏度，宽动态范围和红外阻隔滤波器。最小化 IR 和 UV 光谱分量效应，以产生准确的颜色测量。并且带有环境光强检测和可屏蔽中断。通过 I2C 接口通信。

产品参数

工作电压:	3.3V/5V
控制芯片	TCS34725FN
逻辑电压:	3.3V/5V
通信接口:	I2C
产品尺寸:	27X20(mm)

接口说明

功能引脚	描述
VCC	3.3V/5V 电源正
GND	电源地
SDA	I2C 数据输入
SCL	I2C 时钟输入
INT	中断输出（开漏输出）
LED	发光二极管

目录

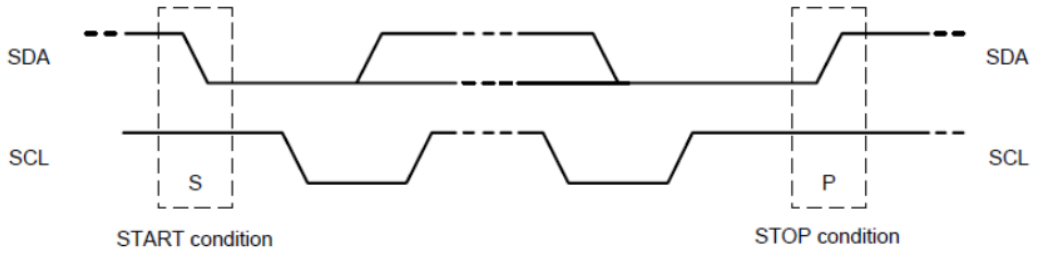
硬件说明

芯片

本产品采用 TCS34725 为核心，是一款基于 IIC 总线通信的彩色光数字转换器。传感器提供红色，绿色，蓝色（RGB）和清晰光感应值的数字输出，带有集成红外阻隔滤波器。具有高灵敏度，宽动态范围。可在不同的光照条件下都可以实现准确的色彩和环境光线感应，带有可屏蔽中断。

通信协议

从上的得知使用的是 I2C 通信，I2C 通信，一条数据线，一条时钟线。 I2C 总线在传送数据过程中共有三种类型信号：开始信号、结束信号和应答信号。

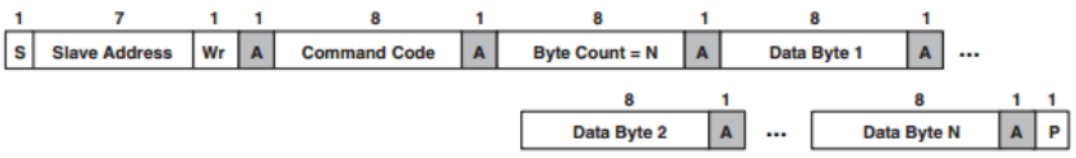


开始信号：SCL 为高电平时， SDA 由高电平向低电平跳变，开始传送数据。

结束信号：SCL 为高电平时， SDA 由低电平向高电平跳变，结束传送数据。

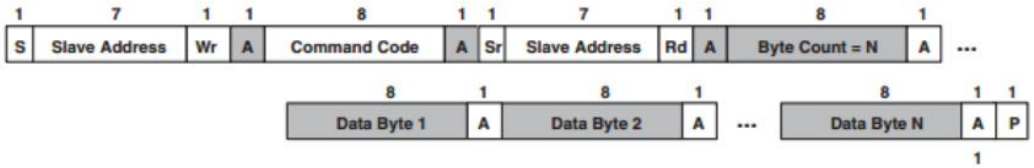
应答信号：接收数据的 IC 在接收到 8bit 数据后，向发送数据的 IC 发出特定的低电平脉冲，表示已收到数据

I2C 写时序



首先主机（即树莓派，后面统称为主机）会发送一个开始信号，然后将其 I2C 的 7 位地址与写操作位组合成 8 位的数据发送给从机（即 TSL2581 传感器模块,后面统称为从机），从机接收到后会响应一个应答信号，主机此时将命令寄存器地址发送给从机，从机接收到发送响应信号，此时主机发送命令寄存的值，从机回应一个响应信号，直到主机发送一个停止信号，此次 I2C 写数据操作结束

I2C 读时序



首先主机会发送一个开始信号，然后将其 I2C 的 7 位地址与写操作位组合成 8 位的数据发送给从机，从机接收到后会响应一个应答信号，主机此时将命令寄存器地址发送给从机，从机接收到发送响应信号，此时主机重新发送一个开始信号，并且将其 7 位地址和读操作位组合成 8 位的数据发送给从机，从机接收到信号后发送响应信号，再将其寄存器中的值发送给主机，主机端给予响应信号，直到主机端发送停止信号，此次通信结束

I2C 地址

TCS34725 的 I2C 设备地址为 0X29

Device	Address	Package-Leads	Interface Description	Ordering Number
TCS34725	0x29	FN–6	I^2C V_{BUS} = V_{DD} Interface	TCS34725FN
TCS34727	0x29	FN–6	I^2C V_{BUS} = 1.8 V Interface	TCS34727FN

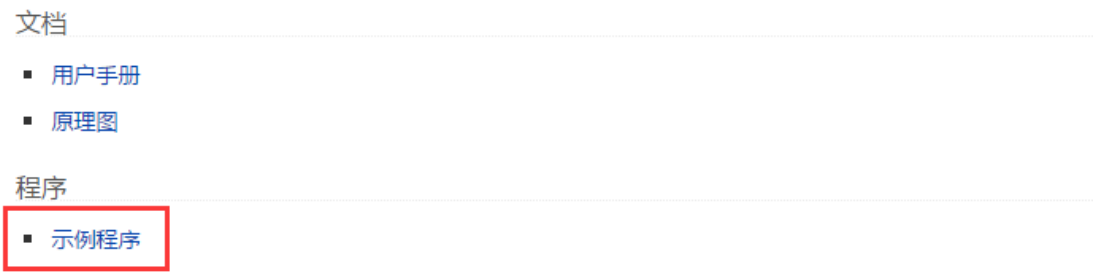
TCS34725 数据手册第 34 页

注意：0X29 这个设备地址是 7 位的，8 位设备地址需要向高位移一位变成 0X52

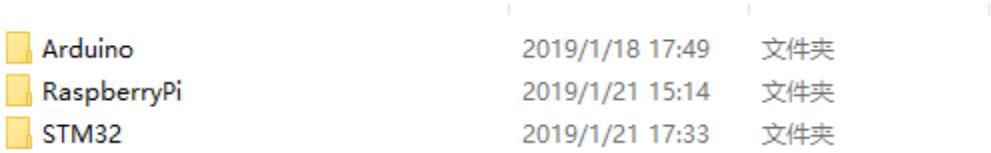
使用说明

下载例程

在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：



得到解压包并解压，得到如下文件：



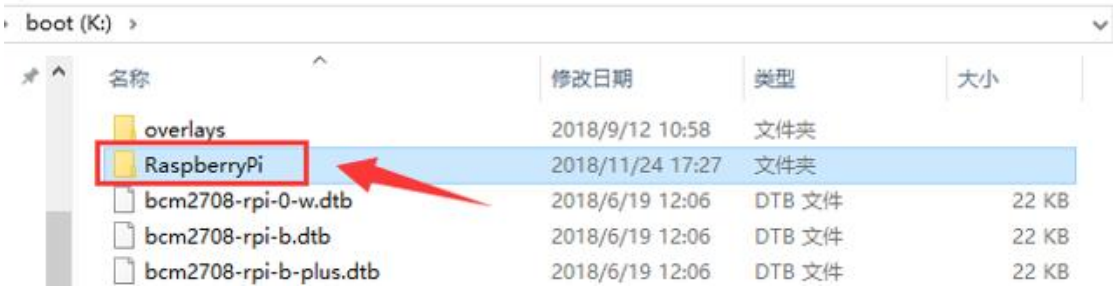
例程：树莓派 BCM2835、WiringPi、python 例程、STM32 例程、Arduino 例程。

使用

树莓派

使用读卡器将 SD 卡插入电脑，将会显示一个 40M 左右的 U 盘，盘名叫：boot.

将解压文件中 RaspberryPi 文件夹复制到 boot 根目录下



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：

ls /boot

```
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  LICENSE.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
```

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/ ./

```
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
```

sudo chmod 777 -R RaspberryPi/

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

运行程序前需要安装对应的函数库（wiringpi, bcm2835, python），否则程序无法正常使用

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:

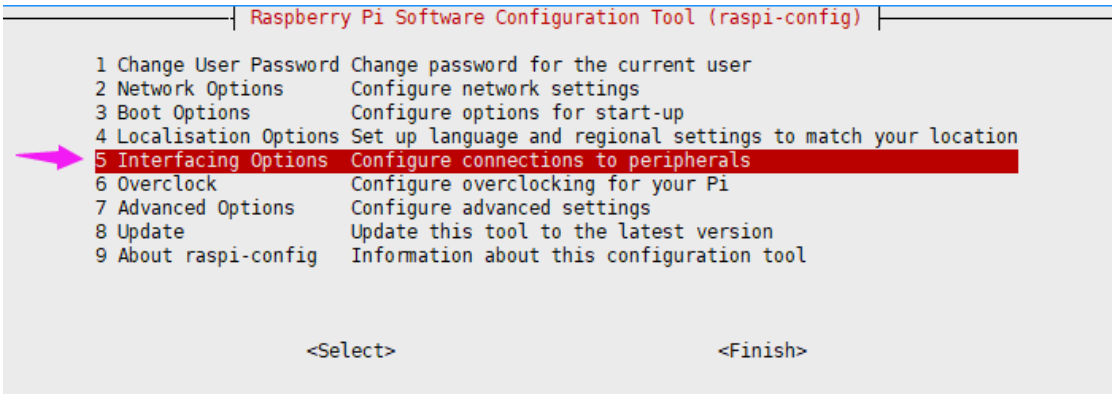
```
sudo apt-get install git

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
```

打开树莓派 I2C 接口

sudo raspi-config



```
┤ 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
```

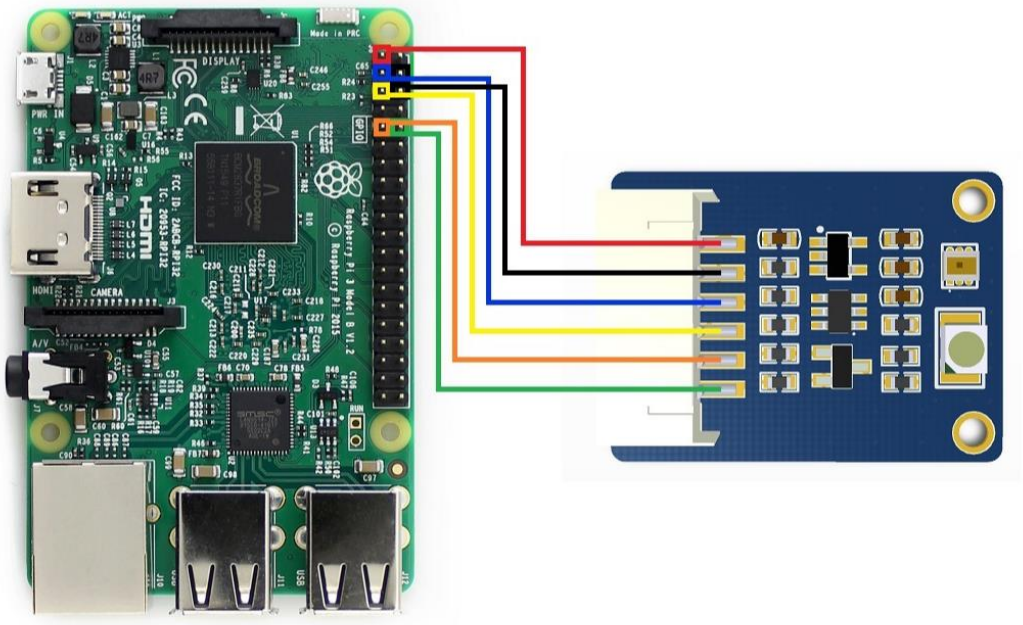
打开 I2C 功能后重启树莓派 sudo reboot

运行 i2cdetect -y 1

最后面的是数字 1 不是字母 l,运行后可以看到当前已经正确连接的 I2C 设备。

```
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: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
pi@raspberrypi:~ $
```

硬件连接



具体连接如下表所示：

TCS34725 Color Sensor	树莓派
VCC	3.3V
GND	GND
SDA	SDA
SCL	SCL
INT	17
LED	18

运行程序

BCM2835 例程

```
cd bcm2835

sudo ./main
```

wiringPi 例程

```
cd wiringpi

sudo ./main
```

python 例程

```
cd python

sudo python main.py
```

注：BCM2835、wiringpi 程序运行如果提示找不到文件，执行 make 即可

预期效果

执行 BCM2835、wiringpi 或者 python 程序得到的效果类似，下面是 python 测试红色运行结果:

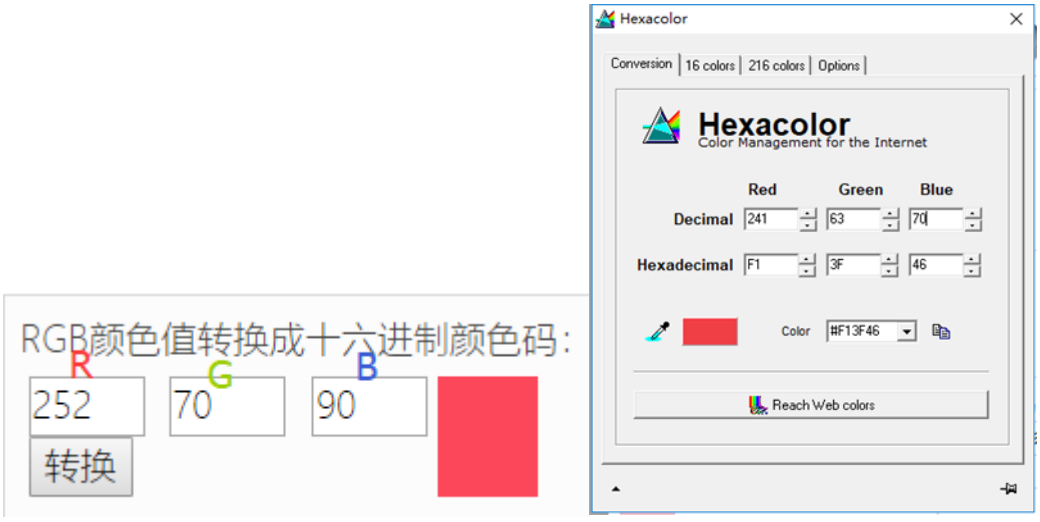
其中 R、G、B 是 RGB888 格式以十进制分开输出，C 是没有做任何处理的环境光强数值，RGB565、RGB888 分别是对应格式的十六进制输出，LUX 是已经处理过的环境光强数字，CT 是色温（http://www.360doc.com/content/17/0629/07/44859260_667365022.shtml），用户如果想测量色温偏差小，建议关闭 LED 灯。INT 是中断，1 标识光强超过设定值。

```
pi@raspberrypi:                          $ sudo python main.py
TCS34725 initialization success!!
R: 252  G: 70  B: 90  C: 0x7d16  RGB565: 0xfa2b  RGB888: 0xfc465a  LUX: 112  CT: 2339K  INT: 1
R: 252  G: 71  B: 90  C: 0x7d89  RGB565: 0xfa2b  RGB888: 0xfc475a  LUX: 113  CT: 2340K  INT: 0
R: 251  G: 70  B: 90  C: 0x7d23  RGB565: 0xfa2b  RGB888: 0xfb465a  LUX: 113  CT: 2342K  INT: 0
R: 252  G: 71  B: 90  C: 0x7d7b  RGB565: 0xfa2b  RGB888: 0xfc475a  LUX: 113  CT: 2339K  INT: 0
R: 251  G: 70  B: 90  C: 0x7d2a  RGB565: 0xfa2b  RGB888: 0xfb465a  LUX: 113  CT: 2342K  INT: 0
R: 251  G: 70  B: 90  C: 0x7cf6  RGB565: 0xfa2b  RGB888: 0xfb465a  LUX: 113  CT: 2341K  INT: 0
R: 252  G: 70  B: 90  C: 0x7d3c  RGB565: 0xfa2b  RGB888: 0xfc465a  LUX: 113  CT: 2339K  INT: 0
R: 252  G: 70  B: 90  C: 0x7d34  RGB565: 0xfa2b  RGB888: 0xfc465a  LUX: 113  CT: 2341K  INT: 0
R: 252  G: 70  B: 90  C: 0x7d50  RGB565: 0xfa2b  RGB888: 0xfc465a  LUX: 113  CT: 2340K  INT: 0
R: 251  G: 70  B: 90  C: 0x7d5a  RGB565: 0xfa2b  RGB888: 0xfb465a  LUX: 114  CT: 2349K  INT: 0
R: 253  G: 71  B: 91  C: 0x7d23  RGB565: 0xfa2b  RGB888: 0xfd475b  LUX: 113  CT: 2349K  INT: 0
R: 252  G: 71  B: 90  C: 0x7f99  RGB565: 0xfa2b  RGB888: 0xfc475a  LUX: 114  CT: 2332K  INT: 0
R: 253  G: 71  B: 90  C: 0x7d81  RGB565: 0xfa2b  RGB888: 0xfd475a  LUX: 112  CT: 2338K  INT: 0
R: 252  G: 70  B: 90  C: 0x7d01  RGB565: 0xfa2b  RGB888: 0xfc465a  LUX: 112  CT: 2339K  INT: 0
R: 252  G: 70  B: 90  C: 0x7d2d  RGB565: 0xfa2b  RGB888: 0xfc465a  LUX: 113  CT: 2342K  INT: 0
R: 252  G: 71  B: 90  C: 0x7d9c  RGB565: 0xfa2b  RGB888: 0xfc475a  LUX: 113  CT: 2341K  INT: 0
R: 252  G: 70  B: 90  C: 0x7d3d  RGB565: 0xfa2b  RGB888: 0xfc465a  LUX: 112  CT: 2339K  INT: 0
R: 251  G: 70  B: 90  C: 0x7cea  RGB565: 0xfa2b  RGB888: 0xfb465a  LUX: 112  CT: 2339K  INT: 0
R: 252  G: 70  B: 90  C: 0x7d28  RGB565: 0xfa2b  RGB888: 0xfc465a  LUX: 112  CT: 2342K  INT: 0
R: 253  G: 71  B: 90  C: 0x7d91  RGB565: 0xfa2b  RGB888: 0xfd475a  LUX: 112  CT: 2340K  INT: 0
R: 252  G: 71  B: 90  C: 0x7d71  RGB565: 0xfa2b  RGB888: 0xfc475a  LUX: 112  CT: 2339K  INT: 0
R: 251  G: 70  B: 90  C: 0x7cfb  RGB565: 0xfa2b  RGB888: 0xfb465a  LUX: 112  CT: 2340K  INT: 0
```

将数值转换成颜色，可以通过下面连接的工具实现：

https://www.sioe.cn/yingyong/yanse-rgb-16/

或者直接下载：http://www.waveshare.net/w/upload/0/05/Hexacolor3.7z



STM32

下载例程解压，使用 Keil uVision5 打开。例程使用的是 HAL 库。测试使用的开发板为微雪 XNUCLEO-F103RB，芯片为 STM32F103RBT6。例程使用串口 2（PA2,PA3）输出数据。串口波特率为 115200，其他默认：数据位 8 位，停止位 1 位，没有校验。

硬件连接

连接如下表所示

TCS34725 Color Sensor	STM32
VCC	3.3V
GND	GND
SDA	SDA/D14/PB9
SCL	SCL/D15/PB8
INT	D8/PA9
LED	PWM1/D9/PC7

预期效果

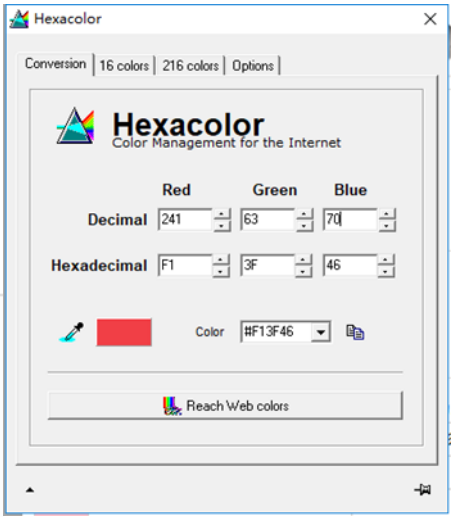
下图为测试红色输出数据

```
RGB888 :R=242   G=63  B=71
RGB888=0XF23F47  RGB565=0X97E7
Lux_Interrupt = 0

RGB888 :R=241   G=62  B=70
RGB888=0XF13E46  RGB565=0X8FC6
Lux_Interrupt = 0

RGB888 :R=243   G=63  B=71
RGB888=0XF33F47  RGB565=0X9FE7
Lux_Interrupt = 0

RGB888 :R=243   G=63  B=71
RGB888=0XF33F47  RGB565=0X9FE7
Lux_Interrupt = 0
```

将数值转换成颜色，可以通过下面连接的工具实现：

https://www.sioe.cn/yingyong/yanse-rgb-16/

或者直接下载：http://www.waveshare.net/w/upload/0/05/Hexacolor3.7z



ARDUINO

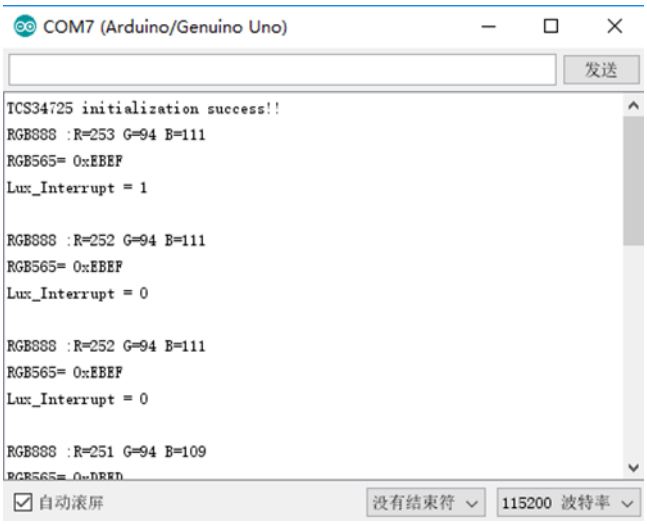
下载例程解压，测试使用的是 ARDUION UNO 开发板，波特率为 115200

硬件连接

TCS34725 Color Sensor	Arduino
VCC	3.3V/5V
GND	GND
SDA	SDA
SCL	SCL
INT	D8
LED	D9

预期效果

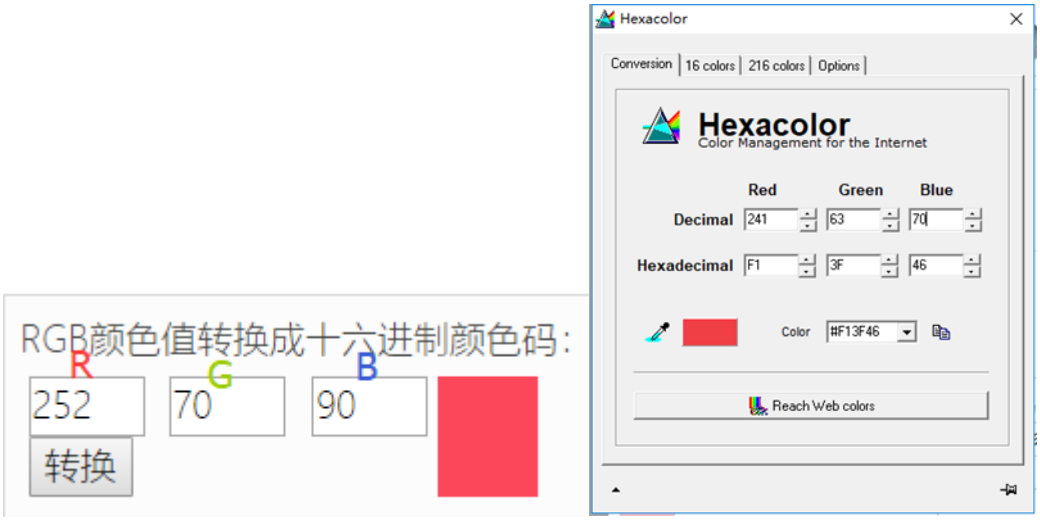
下图为测试红色输出数据



将数值转换成颜色，可以通过下面连接的工具实现：

https://www.sioe.cn/yingyong/yanse-rgb-16/

或者直接下载：http://www.waveshare.net/w/upload/0/05/Hexacolor3.7z



常见问题

1. 树莓派例程初始化失败?

   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。

```
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: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

   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。

2. 不正确的使用树莓派控制可能会导致?

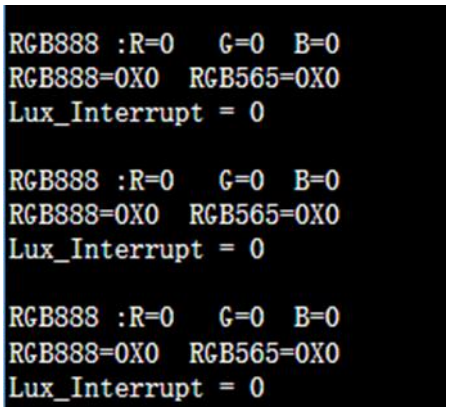
   答：如果运行 wiringPi 例程正常，再运行 python 或者 BCM2835 可能会屏幕无法正常刷新，因为 bcm2835 库是树莓派 cpu 芯片的库函数，底层是直接操作寄存器，而 wiringPi 库和 python 的底层都是通过读写 linux 系统的设备文件操作设备，可能导致 GPIO 口异常，重启树莓派可完美解决。

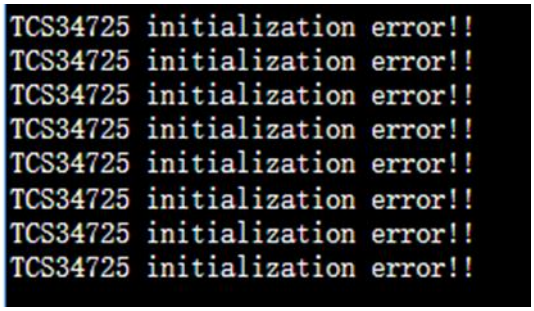
3. STM32 和 Arduino 例程串口输出没有数据或者数据输出乱码?

   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 和 Arduino 例程串口输出 RGB 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键。





5. 输出的 RGB 数据全为 253 并且中断引脚产生中断等等如图

   答：这种情况是光强超出检查范围，减小增益可以完美解决（在初始里面修改，或者在初

始化后面重新写入一个增益设置 TCS34725_Set_Gain(TCS34725_GAIN_16X))

```
R: 253  G: 253  B: 253  C: 0xffff  RGB565: 0xffff  RGB888: 0xfdfdfd  LUX: 0  CT: 5201K  INT: 1
R: 253  G: 253  B: 253  C: 0xffff  RGB565: 0xffff  RGB888: 0xfdfdfd  LUX: 0  CT: 5201K  INT: 0
R: 253  G: 253  B: 253  C: 0xffff  RGB565: 0xffff  RGB888: 0xfdfdfd  LUX: 0  CT: 5201K  INT: 0
R: 253  G: 253  B: 253  C: 0xffff  RGB565: 0xffff  RGB888: 0xfdfdfd  LUX: 0  CT: 5201K  INT: 1
R: 253  G: 253  B: 253  C: 0xffff  RGB565: 0xffff  RGB888: 0xfdfdfd  LUX: 0  CT: 5201K  INT: 0
R: 253  G: 253  B: 253  C: 0xffff  RGB565: 0xffff  RGB888: 0xfdfdfd  LUX: 0  CT: 5201K  INT: 1
R: 253  G: 253  B: 253  C: 0xffff  RGB565: 0xffff  RGB888: 0xfdfdfd  LUX: 0  CT: 5201K  INT: 0
R: 253  G: 253  B: 253  C: 0xffff  RGB565: 0xffff  RGB888: 0xfdfdfd  LUX: 0  CT: 5201K  INT: 0
R: 253  G: 253  B: 253  C: 0xffff  RGB565: 0xffff  RGB888: 0xfdfdfd  LUX: 0  CT: 5201K  INT: 1
```

6. 修改积分时间后导致颜色不正常？

答：因为积分时间决定了 RGBC 通道数据最大值，修改积分时间会导致颜色偏暗或者偏白。只需要增加或减少 LED 亮度即可。

7. 修改积分时间无法触发中断或者一直重复中断？

答：中断是和 Clear 通道里面的数据进行比较，Clear 通道里面的数据和积分时间有关系，经过实际测量在增益为 60 倍情况下

积分时间	通道最大值
2.4ms	1024
24ms	10240
50ms	5400
101ms	21504
154ms	65535
700ms	65535

所以用户如果需要速度比较快的采集数据时，要注意重新设置中断数值。另外在积分时间为 2.4ms 时 RGB 数据比较低所以转换出来的颜色与实际颜色有偏差，需要加大 LED 灯亮度即可。