**2022年盟升杯电子设计竞赛（低年级组）**

**设计报告**



**B题： 色彩提取器**

**制作人：顾屹恒**

**2022年10月**

## 色彩提取器

【摘 要】 一个基于STM32的色彩提取器。使用色彩传感器TCS34725获取24bit RGB数值及8bit亮度数值，并在一块由ST7735S驱动的0.96英寸LCD屏幕上复现该颜色并显示相关信息，同时在一盏RGB LED灯上使用PWM调光复现该颜色。

【关键词】 色彩提取器；STM32；模拟SPI；I2C；ST7735；TCS34725；RGB LED灯泡；PWM控制。

目 录

[色彩提取器 1](#_Toc117351333)

[1 设计任务与要求 3](#_Toc117351334)

[1.1 基本要求 3](#_Toc117351335)

[1.2 发挥部分 3](#_Toc117351336)

[2 方案设计与论证 4](#_Toc117351337)

[2.1 待测颜色获取方案 4](#_Toc117351338)

[2.2 RGB LED灯控制方案 4](#_Toc117351339)

[2.3 屏幕显示方案 4](#_Toc117351340)

[3 理论分析与计算 5](#_Toc117351341)

[3.1 TCS34725原始数据获取 5](#_Toc117351342)

[3.2 RGB LED灯控制 5](#_Toc117351343)

[3.3 LCD屏幕控制 5](#_Toc117351344)

[4 系统软、硬件设计 5](#_Toc117351345)

[4.1 系统结构设计 5](#_Toc117351346)

[4.2 硬件电路设计 6](#_Toc117351347)

[4.3 系统软件及交互 7](#_Toc117351348)

[5 系统测试及结果 8](#_Toc117351349)

[5.1 测试仪器 8](#_Toc117351350)

[5.2 测试方法 8](#_Toc117351351)

[5.3 性能指标 8](#_Toc117351352)

[6 总结 9](#_Toc117351353)

## 1 设计任务与要求

### 1.1 基本要求

1.1.1使用7805设计供电电路，使用学生电源供电。

1.1.2按下按键即测量物体颜色RGB值，测量时间不超过3s。

1.1.3可通过屏幕显示测得的R,G,B相应数值。

1.1.4将测得的颜色用RGB灯复现，要求肉眼看不出明显色差。

1.1.5可将识别到的色彩用彩色TFT屏幕显示，无明显色差。

1.1.6显示出从测量开始到结束所需时间

### 1.2 发挥部分

1.2.1实现自动测量，即靠近物体自动测量，无需人工干预。

1.2.2可保存并复现出开机以来的所有测量结果。

1.2.3可标记并记录一种颜色，当再次测量相同颜色时可以给出提示。（如LED闪烁，蜂鸣器报警，屏幕显示提示文字等等）

1.2.4测量时间不超过1s。

1.2.5其他（即自由发挥，优秀的地方）。

## 2 方案设计与论证

设计并制作一个基于STM32和TCS34725的色彩传提取器。使用7805芯片配合滤波电容进行降压供电，驱动整个系统。通过TCS34725获取RGB值，通过改变三路PWM波占空比调整RGB灯颜色，同时将原24bit RGB值转为16bit输出给屏幕进行复现。系统整体框图如下页图1所示。

图示

描述已自动生成

图1 色彩提取器系统框图

### 2.1 待测颜色获取方案

采用TCS34725传感器获取颜色每项8bit的R、G、B、C数值，并通过I2C协议传输给微控制器（本次使用STM32F411CEU）。

### 2.2 RGB LED灯控制方案

使用STM32F411CEU的定时器分别生成三路PWM波以调整输出的RGB颜色。

相关重要代码见附件。

### 2.3 屏幕显示方案

方案一：使用ST7735S驱动的一块0.96’IPS屏幕。

方案二：使用ST7735S驱动的一块1.8’TFT屏幕。

因驱动及显示效果问题，选择方案一。

## 3 理论分析与计算

### 3.1 TCS34725原始数据获取

读取四次RGB通道数值并求平均值。为了数据操作方便，同时防止溢出，未使用驱动提供的结构体。

发现由于传感器准确度不佳，在测量白色物体时偏蓝绿，原因是此时读取到的G>B>R，故进行白平衡处理。又由于输入数据因亮度过高有大于255的可能，因此，若某组数据中RGB三者任一值大于255，则进行整组数据的缩小，直至该数据小于255。

相关重要代码见附件。

### 3.2 RGB LED灯控制

使用STM32F411CEU的TIM3定时器的CH2、CH3、CH4分别生成PWM波，将PCS设为99（100-1，APB1、APB2主频都为100MHz，也即PWM波频率为1MHz），将ARR设为255，方便后续直接使用RGB值调整占空比。

后通过输入的8bit RGB值（0-255）调整输出颜色。

相关重要代码见附件。

### 3.3 LCD屏幕控制

在基础测量页面，左边显示当前测得颜色色块，右边显示RGBC数值以及测量耗时（按下按钮直至数据处理完成，由于需同步显示耗时，故该时间不包括显示耗时）。

因源数据是RGB24bit（RGB888）格式，而屏幕采用RGB16bit（RGB565，高位在前）故通过二进制位操作将RGB888转为RGB565。

相关重要代码见附件。

## 4 系统软、硬件设计

### 4.1 系统结构设计

使用PCB板配合排母、排针、杜邦线进行电路连接，为提升TCS34725模块的可移动性，故使用杜邦线将其与PCB板连接。

### 4.2 硬件电路设计

4.2.1 电源降压电路

使用7805芯片配合滤波电容进行降压供电。电路原理图如下：

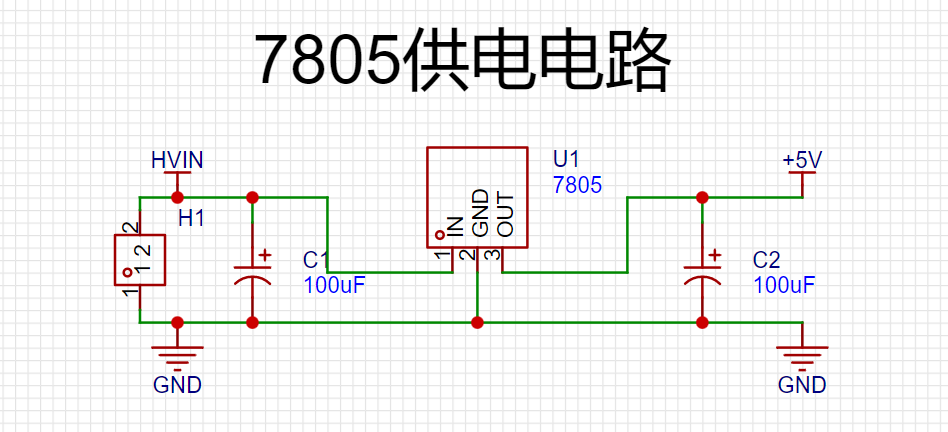


图2 7805降压供电电路原理图

4.2.2 主模块

由于使用开发板，故使用排针连接各模块，原理图如下：

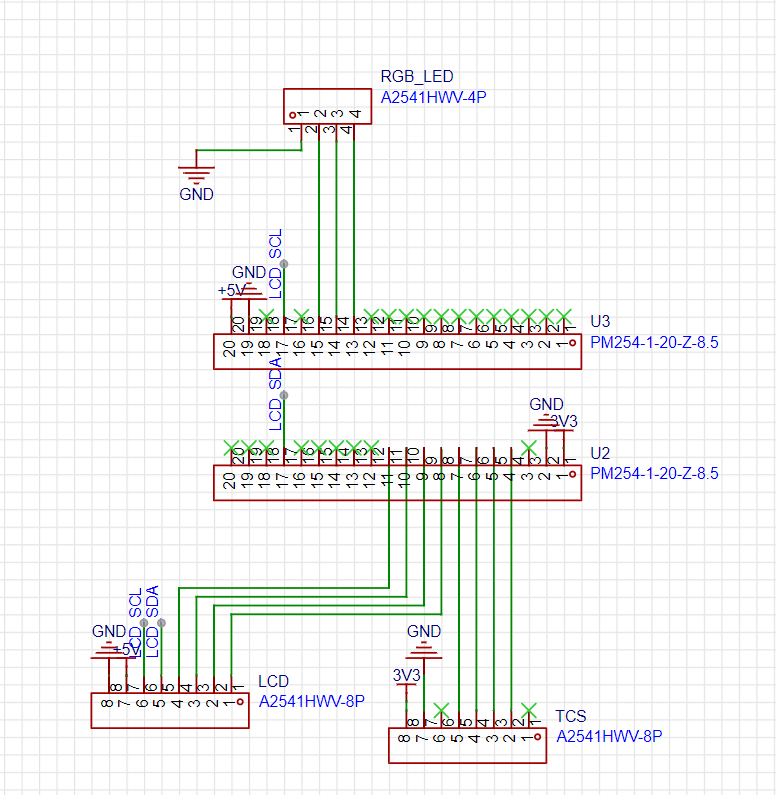


图3 主模块原理图

4.2.3 PCB绘制

PCB绘制如图4、5：

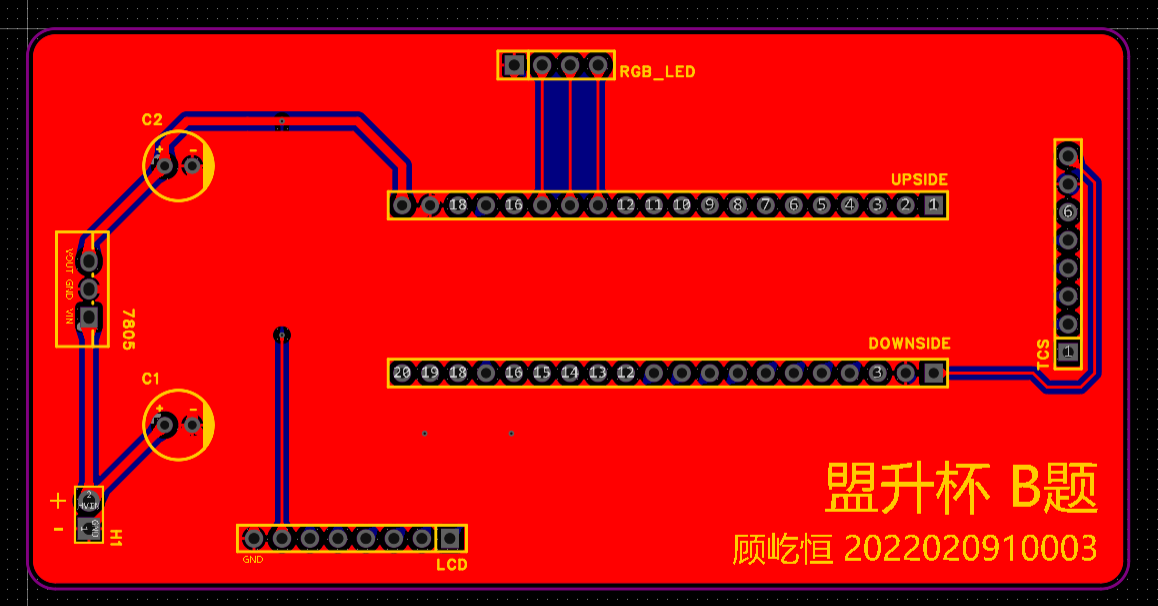


图4 PCB顶层

电脑萤幕画面

描述已自动生成

图5 PCB底层

### 4.3 系统软件及交互

4.3.1 主控程序流程

使用STM32CUBEMX+KEIL MDK + VScode(使用EIDE扩展)，基于HAL库编程，开发软件开发STM32单片机作为主控，作为系统控制核心完成数据输入、屏幕驱动、灯光输出的操作。

4.3.2 用户输入

使用开发板自带的按钮（KEY,PA0）识别用户的短按、长按实现不同功能。

4.3.3 初始化

上电开机即进行初始化，RGB分别显示50ms，后进行屏幕、TCS34725的初始化，若TCS34725初始化失败，则红屏显示 E TCS，初始化成功则显示电子科技大学校徽。

4.4.4 单次测量

单击KEY，获取数据并处理后输出（具体方法及现象见上文），随后等待下次操作输入。通过定时器溢出中断次数计算耗时。

4.4.5 自动测量

在单次测量界面长按KEY（大于500ms），屏幕输出“AUTO”字样，500ms后开始自动识别，靠近待测物体，使传感器获得的亮度大于预设数值，后自动测量。此时单击KEY退回至手动单次测量模式，显示“MANUAL”字样。

4.4.6 显示历史

在单次测量界面长按KEY，跳过“AUTO”界面，出现“HISTORY”字样后松开KEY，即显示开机以来的测量记录（RGB值及对应颜色），每页六项，右侧显示页码及总记录数，单击KEY翻页，在尾页双击KEY退出。

4.4.6 显示历史

在单次测量界面长按KEY，跳过“AUTO”、“HISTORY”界面，当出现“MARK”字样后松开按键，即标记当前测得颜色，再次单击返回手动测量，当测得相同颜色（允许因亮度等原因存在的误差）时屏幕上出现提示并同时展示两种颜色。

## 5 系统测试及结果

### 5.1 测试仪器

常见不同颜色的物体（不反光）。

### 5.2 测试方法

按照设计操作颜色提取器提取颜色。

## 5.3 性能指标

表1 基本要求 测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 测试项目 | 题目指标 | 完成情况 | 是否达标 |
| 供电设计 | 完成 | 完成 | 是 |
| 测量时间 | 小于3s | 约2.4ms | 是 |
| 屏幕显示RGB数值 | 显示 | 显示 | 是 |
| RGB灯复现颜色 | 无明显色差 | 完成 | 是 |
| 彩色TFT屏幕显示 | 无明显色差 | 完成 | 是 |
| 显示测量耗时 | 显示 | 完成 | 是 |

表2 发挥要求 测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 测试项目 | 题目指标 | 完成情况 | 是否达标 |
| 自动测量 | 自动测量 | 完成 | 是 |
| 保存并复现开机以来测量的所有结果 | 保存并复现 | 保存并复现 | 是 |
| 可标记并记录一种颜色，当再次测量相同颜色时可以给出提示 | 完成 | 完成 | 是 |
| 测量时间不超过1s | 小于1s | 约2.4ms（自动测量大于200ms  小于1s） | 是 |

## 6 总结

通过这次制作，获得了使用STM32的基本知识，能够在制作过程中发现错误，解决错误。最终成品较为完善，但仍存在如交互方面的不足，此次竞赛经历是人生中难得的机会与体验。

注释：

参考文献：

附件：

1.相关重要及代码：

  //  pwm初始化

  HAL\_TIM\_PWM\_Start(&htim3, TIM\_CHANNEL\_2); // b LED pwm START

  HAL\_TIM\_PWM\_Start(&htim3, TIM\_CHANNEL\_3); // g LED pwm START

  HAL\_TIM\_PWM\_Start(&htim3, TIM\_CHANNEL\_4); // r LED pwm START

  HAL\_TIM\_PWM\_Start(&htim4, TIM\_CHANNEL\_4); // TCS\_LED pwm START

// rgb led灯pwm输出

void rgb(uint16\_t r00, uint16\_t g00, uint16\_t b00)

{

  \_\_HAL\_TIM\_SET\_COMPARE(&htim3, TIM\_CHANNEL\_2, b00 \* 0.8);

  \_\_HAL\_TIM\_SET\_COMPARE(&htim3, TIM\_CHANNEL\_3, g00 \* 0.9);

  \_\_HAL\_TIM\_SET\_COMPARE(&htim3, TIM\_CHANNEL\_4, r00);

}

// TCS LED亮度pwm输出

void T\_LED\_OUTPUT(uint8\_t T\_LED\_LEVEL)

{

  \_\_HAL\_TIM\_SET\_COMPARE(&htim4, TIM\_CHANNEL\_4, T\_LED\_LEVEL);

}

// 24bit rgb数据转16bit

uint16\_t rgb16(uint8\_t r00, uint8\_t g00, uint8\_t b00)

{

  r00 >>= 3;

  g00 >>= 2;

  b00 >>= 3;

  return ((r00 << 11) | (g00 << 5) | b00);

}

//单次手动测量(部分代码)

  uint32\_t color0r = 0, color0g = 0, color0b = 0, color0c = 0; //各通道颜色初始化

  float getcount;                                              //计时终数值

  timcount = 0;                                            //计时溢出计数归零，开始计时

  for (int i = 0; i <= 3; i++)

  {

    color0r += TCS34725\_GetChannelData(TCS34725\_RDATAL);

    color0g += TCS34725\_GetChannelData(TCS34725\_GDATAL);

    color0b += TCS34725\_GetChannelData(TCS34725\_BDATAL);

    color0c += TCS34725\_GetChannelData(TCS34725\_CDATAL);

  } //读取4次各通道数据并求平均

  color0r = color0r / 3.1;

  color0g = color0g / 4.85;

  color0b = color0b / 4;

  float color1r = color0r, color1g = color0g, color1b = color0b, i = 0;

  while (color1r >= 255 || color1g >= 255 || color1b >= 255)

  {

    color1r = color0r / i;

    color1g = color0g / i;

    color1b = color0b / i;

    i += 0.01;

  }

  color0b = color1b;

  color0r = color1r;

  color0g = color1g;

  getcount = (float)timcount / 10.0;                     //计算用时，每0.1ms溢出一次

//自动测量（部分代码）

while (1)

  {

    if (TCS34725\_GetChannelData(TCS34725\_CDATAL) >= 350)

    {

      float getcount; //计时终数值

      timcount = 0;   //计时溢出计数归零，开始计时

      uint8\_t i = 9;

      TCS34725\_SetIntegrationTime(TCS34725\_INTEGRATIONTIME\_24MS);

      HAL\_Delay(25);

      while (TCS34725\_GetChannelData(TCS34725\_CDATAL) >= 300 && i >= 1)

      {

        T\_LED\_OUTPUT(i);

        HAL\_Delay(25);

        i--;

      }

      TCS34725\_SetIntegrationTime(TCS34725\_INTEGRATIONTIME\_101MS);

      HAL\_Delay(110);

      uint32\_t color0r = 0, color0g = 0, color0b = 0, color0c = 0; //各通道数据

      for (int i = 0; i <= 3; i++)

      {

        color0r += TCS34725\_GetChannelData(TCS34725\_RDATAL);

        color0g += TCS34725\_GetChannelData(TCS34725\_GDATAL);

        color0b += TCS34725\_GetChannelData(TCS34725\_BDATAL);

        color0c += TCS34725\_GetChannelData(TCS34725\_CDATAL);

      } //读取4次各通道数据并求平均

      color0r = color0r / 2.4;

      color0g = color0g / 3.78;

      color0b = color0b / 3.15;                                              //调色

      float color1r = color0r, color1g = color0g, color1b = color0b, i1 = 0; // i1用于计数

      while (color1r >= 255 || color1g >= 255 || color1b >= 255)

      {

        color1r = color0r / i1;

        color1g = color0g / i1;

        color1b = color0b / i1;

        i1 += 0.01;

      } //防止溢出

      color0b = color1b;

      color0r = color1r;

      color0g = color1g;

      getcount = (float)timcount / 10.0;                        //计算用时，0.1ms溢出一次

…………

//显示历史

while (p <= pmax)

  {

    char pgstr[10];

    sprintf(pgstr, "%d/%d", p, pmax);

    LCD\_Fill(0, 0, 160, 80, BLACK);

    LCD\_ShowString(120, 5, rcdstr, WHITE, BLACK, 16, 1);

    LCD\_ShowString(105, 25, "records", WHITE, BLACK, 16, 1);

    LCD\_ShowString(115, 50, pgstr, WHITE, BLACK, 24, 1);

    while (i <= 17 && j >= 2)

    {

      sprintf(rgbstr, "(%3d,%3d,%3d)", mem[j - 2], mem[j - 1], mem[j]);

      LCD\_ShowString(0, 16 \* i / 3, rgbstr, rgb16(255 - mem[j - 2], 255 - mem[j - 1], 255 - mem[j]), rgb16(mem[j - 2], mem[j - 1], mem[j]), 16, 0);

      i += 3;

      j -= 3;

    }

//记录（mem[]、mem\_i为全局数组、变量）

void MEM(uint8\_t r, uint8\_t g, uint8\_t b)

{

  mem[mem\_i] = r;

  mem[mem\_i + 1] = g;

  mem[mem\_i + 2] = b;

  mem\_i += 3;

}