

**课 程 实 验 报 告**

**课程名称： 传感器原理及工程应用**

**专业班级： 物联网1601班**

**学 号： U201614898**

**姓 名： 潘翔**

**指导教师： 宋恩民**

**报告日期： 2019年 3月**

**计算机科学与技术学院**

目 录

[1 实验平台的熟悉和使用实验 1](#_Toc476223162)

[1.1 实验目的 1](#_Toc1510883798)

[1.2 实验原理 1](#_Toc61379855)

[1.3 实验步骤 17](#_Toc1580425772)

[1.4 实验过程与结果分析 18](#_Toc513861706)

[1.5 实验问题分析 25](#_Toc609571125)

[1.6 实验总结 26](#_Toc1365610958)

[1.7 参考文献 26](#_Toc2124085336)

[2 颜色传感器实验 27](#_Toc884703383)

[2.1 实验目的 27](#_Toc1903549318)

[2.2 实验原理 27](#_Toc1748827374)

[2.3 实验步骤 29](#_Toc609928509)

[2.4 实验过程与结果分析 29](#_Toc559231350)

[2.5 源码分析 33](#_Toc1302809893)

[2.6 应用场景 34](#_Toc1204037147)

[2.7 实验总结 35](#_Toc1195881663)

# 实验平台的熟悉和使用实验

## 实验目的

了解各种传感器

## 实验原理

### 实验箱构成

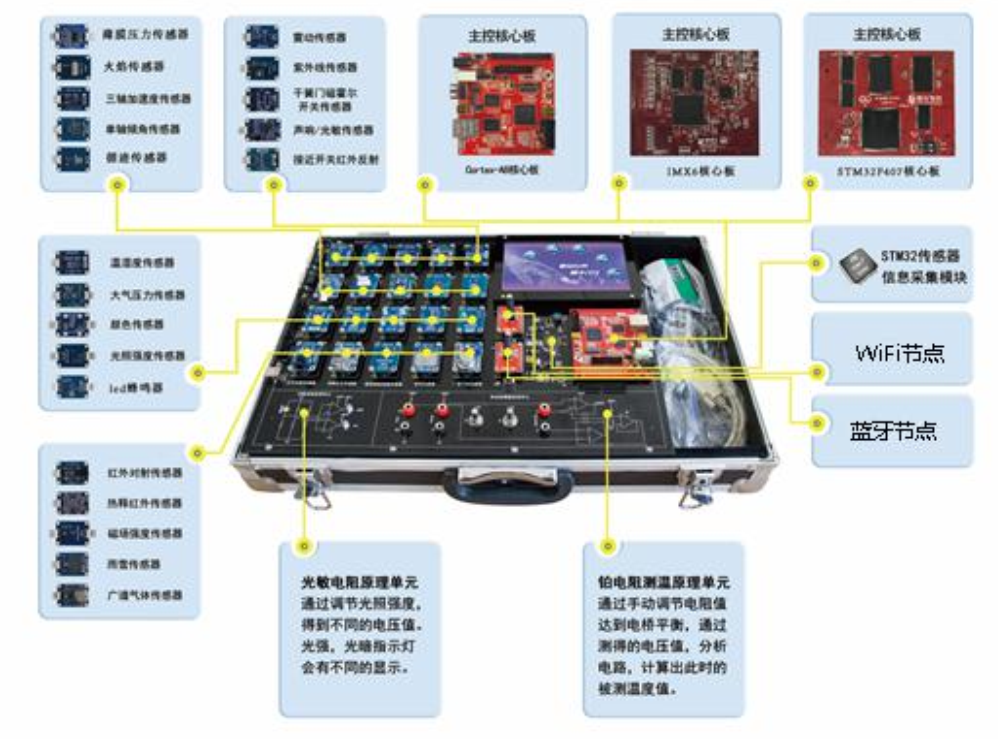


图1.1 试验箱组成图

### 实验箱原理

1. 整体架构

试验箱整体上使用一块IMX6核心板和一块STM32核心板进行控制，其中STM32传感器信息采集模块进行信息采集，将数据通过JLink送至宿主机，从而实现宿主机显示实时的反馈。

1. 连接方式

J-Link是德国SEGGER公司推出基于JTAG的仿真器。简单地说，是给一个JTAG协议转换盒，即一个小型USB到JTAG的转换盒，其连接到计算机用的是USB接口，而到目标板内部用的还是JTAG协议。它完成了从软件到硬件转换的工作。

1. 编译方式

采用宿主机交叉编译生成STM32源码，使用串口传输至STM32开发板

### 程序架构

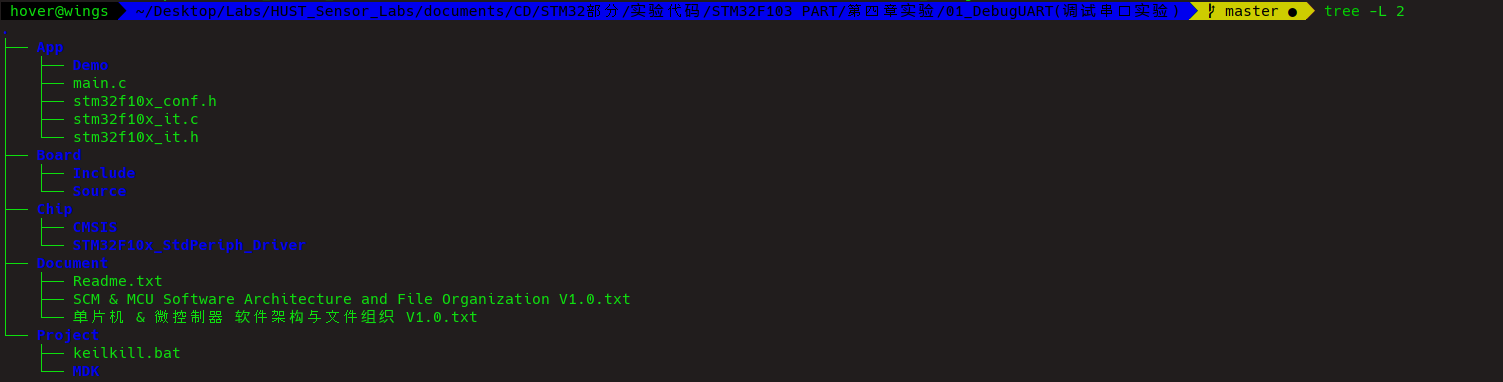


图1.2 实验代码架构图

以01实验为例，进行实验代码架构分析，整个拷贝代码主要由上层应用和下层驱动构成

使用TEST程序调用，使用BSP程序实现相应接口

.

├── App //上层应用程序

│   ├── Demo

│   ├── main.c //主程序入口

│   ├── stm32f10x\_conf.h

│   ├── stm32f10x\_it.c

│   └── stm32f10x\_it.h

├── Board

│   ├── Include

│   └── Source

├── Chip

│   ├── CMSIS

│   └── STM32F10x\_StdPeriph\_Driver

├── Document

│   ├── Readme.txt

│   ├── SCM & MCU Software Architecture and File Organization V1.0.txt

│   └── 单片机 & 微控制器 软件架构与文件组织 V1.0.txt

└── Project

├── keilkill.bat

└── MDK

1. APP

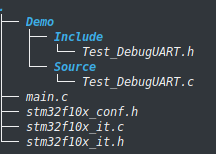


图1.3 APP代码架构图

.

├── Demo //样例代码文件夹

│   ├── Include //头文件

│   │   └── Test\_DebugUART.h

│   └── Source //源文件

│   └── Test\_DebugUART.c

├── main.c //主函数调用接口

├── stm32f10x\_conf.h //配置文件

├── stm32f10x\_it.c //中断配置源文件

└── stm32f10x\_it.h //中断配置头文件

1. Board

存放基础驱动文件

├── Board

│   ├── Include //驱动头文件

│   └── Source //驱动源文件

1. Chip

├── CMSIS //ARM Cortex微控制器软件接口

│ ├── core\_cm3.c

│ ├── core\_cm3.h

│ ├── startup //startup程序

│ ├── stm32f10x.h

│ ├── system\_stm32f10x.c

│ └── system\_stm32f10x.h

└── STM32F10x\_StdPeriph\_Driver//STM32驱动程序

### 串口实验

#### 串口原理

串行接口 (Serial Interface) 是指数据一位一位地顺序传送， 其特点是通信线路简单，只要一对传输线就可以实现双向通信（可以直接利用电话线作为传输线） ， 从而大大降低了成本，特别适用于远距离通信，但传送速度较慢。一条信息的各位数据被逐位按顺序传送的通讯方式称为串行通讯。

串行通讯的特点是：

数据位的传送，按位顺序进行，最少只需一根传输线即可完成；成本低但传送速度慢。串行通讯的距离可以从几米到几千米；

#### 源码注释

##### Main.c

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 模块：Test\_DebugUART

\* 描述：DebugUART 应用测试

\* 作者：Shao

\* 时间：2018.06.12

\* 版本：Version 1.0.0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#include "stm32f10x.h"

#include "BSP\_DebugUART.h"

#include "Test\_DebugUART.h"

/\* 主函数 \*/

int main(void)

{

/\* 优先级分组设置为 4，不使用默认分组方案 \*/

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_4);

/\* 初始化 \*/

BSP\_DebugUART\_Init(115200);

Test\_DebugUART();

}

##### BSP\_DebugUART.c

//板载 DebugUART GPIO 初始化函数

static void BSP\_DebugUART\_GPIO\_Init(void);

板载 DebugUART USART 初始化函数

static void BSP\_DebugUART\_USART\_Init(uint32\_t BaudRate);

板载 DebugUART NVIC 初始化函数

static void BSP\_DebugUART\_NVIC\_Init(void);

其中，GPIO为：General-purpose Input/Output 通过引脚的高低电频实现功能

### LED蜂鸣器模块实验

#### 蜂鸣器原理

LED 由三极管控制是否导通。 GPH\_1 连高电平， 三极管导通， LED 亮； GPH\_1 连低电平， 三极管截止， LED 熄灭。

蜂鸣器同样由三极管控制是否导通。 GPH\_0 连高电平， 三极管导通， 蜂鸣器发声； GPH\_0 连低电平，三极管截止， 蜂鸣器息声。

#### 源码注释

##### BSP\_LEDBuzzer.c

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名: BSP\_LEDBuzzer\_GPIO\_Init

\* 功能说明: 板级 LEDBuzzer GPIO 内部初始化函数

\* 形 参: 无

\* 返 回 值: 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

static void BSP\_LEDBuzzer\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_APB2PeriphClockCmd(RCC\_LED | RCC\_Buzzer, ENABLE); //使能对应时钟

GPIO\_InitStructure.GPIO\_Pin = GPIO\_PIN\_LED; //绑定GPIO引脚

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz; //设定GPIO扫描频率

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP; //推挽输出

GPIO\_Init(GPIO\_PORT\_LED, &GPIO\_InitStructure); //init GPIO

GPIO\_InitStructure.GPIO\_Pin = GPIO\_PIN\_Buzzer;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;

GPIO\_Init(GPIO\_PORT\_Buzzer, &GPIO\_InitStructure); //init GPIO

BSP\_LEDBuzzer\_Off(LED | Buzzer);

}

### 震动传感器终端实验

#### 震动传感器原理

震动传感器采用中断，检测到震动阈值，产生中断输出

#### 源码注释

##### BSP\_ExtInt.c

void BSP\_ExtInt\_Init(void)

{

BSP\_ExtInt\_GPIO\_Init();

BSP\_ExtInt\_NVIC\_Init(); //中断序列初始化

BSP\_ExtInt\_EXTI\_Init(); //外部中断程序初始化

}

### 红外对射传感器实验

#### 红外对射传感器原理

红外对射传感器采用定时器中断，检测到定时检测状态，若检测到遮挡，产生产生中断输出

#### 源码注释

##### BSP\_Timer.c

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名: BSP\_Timer\_Init

\* 功能说明: 板载 Timer 初始化函数

\* 形 参: usCount 定时 us 时常计数

\* 返 回 值: 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void BSP\_Timer\_Init(uint16\_t usCount)

{

BSP\_Timer\_NVIC\_Init(); //中断程序设定

BSP\_Timer\_TIM2\_Init(usCount); //定时器设定

}

### 热释红外传感器实验

同红外对射传感器，采用定时器中断

### 光谱气体传感器实验

同红外对射传感器，采用定时器中断

### 雨雪传感器实验

同红外对射传感器，采用定时器中断

### 干簧门磁霍尔开关模块实验

##### 干簧门磁霍尔开关模块实验原理

同红外对射传感器，采用定时器中断

##### 源码注释

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名: TIM2\_IRQHandler

\* 功能说明: STM32 TIM2 中断服务函数

\* 形 参: 无

\* 返 回 值: 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void TIM2\_IRQHandler(void)

{

static uint8\_t count1,count2;

if (TIM\_GetITStatus(TIM2, TIM\_IT\_Update) != RESET) //检查 TIM2 更新中断

{

if(GPIO\_ReadInputDataBit(GPIO\_PORT\_ExtInt1,GPIO\_PIN\_ExtInt1) == 0)

{

count1++;

if(count1 > 10) //累计时序检测

{

printf("干簧管\r\n");

}

}

}

else

count1 = 0;

if(GPIO\_ReadInputDataBit(GPIO\_PORT\_ExtInt2,GPIO\_PIN\_ExtInt2) == 0)

{

count2++;

if(count2 > 10)

{

printf("霍尔开关\r\n"); //累计时序检测

}

}

else

count2 = 0;

}

TIM\_ClearITPendingBit(TIM2, TIM\_IT\_Update); //清除 TIMx 更新中断标志

}

### 声响开关光敏传感器实验

同红外对射传感器，采用定时器中断

### 接近开关红外反射模块实验

同红外对射传感器，采用定时器中断

### 循迹传感器实验

##### 循迹传感器原理

同红外对射传感器，采用定时器中断

##### 源码注释

中断处理函数

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名: TIM2\_IRQHandler

\* 功能说明: STM32 TIM2中断服务函数

\* 形 参: 无

\* 返 回 值: 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void TIM2\_IRQHandler(void)

{

uint8\_t state;

if (TIM\_GetITStatus(TIM2, TIM\_IT\_Update) != RESET) //检查TIM2更新中断

{

state = GPIO\_ReadInputDataBit(GPIO\_PORT\_ExtInt1,GPIO\_PIN\_ExtInt1);

if(GPIO\_ReadInputDataBit(GPIO\_PORT\_ExtInt2,GPIO\_PIN\_ExtInt2))

state += 0x10;// 判定状态

switch (state)

{

case 0x00:

{

printf("停止！\n");

break;

}

case 0x01:

{

printf("向左！\n");

break;

}

case 0x10:

{

printf("向右！\n");

break;

}

case 0x11:

{

printf("直行！\n");

break;

}

}

}

TIM\_ClearITPendingBit(TIM2, TIM\_IT\_Update ); //清除TIMx更新中断标志

}

### 三轴加速度计传感器实验

##### 三轴加速度计传感器原理

GPIO 模拟 IIC 时序，达到软件模拟实现 IIC 通信的目的。

对于I2C信号，需要有START，STOP，ACK，NACK，以及接收DATA。接收DATA是在SCL的低电平可能发生跳变，START和STOP是在高电平跳变。当SCL保持高电平的时候，SDA从H跳变到L，即为START；当SCL保持高电平的时候，SDA从L跳变到H，即为STOP。

利用GPIO引脚电平高低组合生成IIC时序所需的控制信号，此处已经提供底层的模拟接口实现。

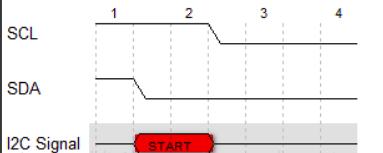


图1.4 START对应信号图

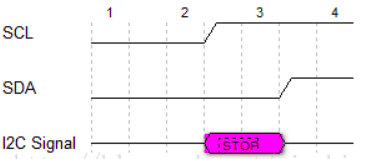


图1.5 STOP对应信号图

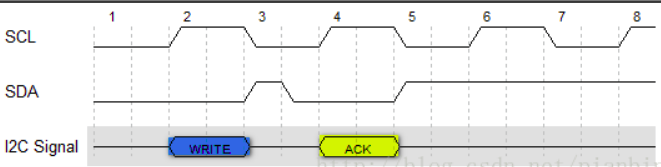


图1.6 WRITE和ACK对应信号图

##### 源码注释

u8\_t LIS331DLH\_ReadReg(u8\_t deviceAddr, u8\_t Reg, u8\_t\* Data);//实现读数据

u8\_t LIS331DLH\_WriteReg(u8\_t deviceAddress, u8\_t WriteAddr, u8\_t Data);//实现写数据

void BSP\_MyIIC\_Start(void)

{

/\* 当 SCL 高电平时， SDA 出现一个下跳沿表示 I2C 总线启动信号 \*/

I2C\_SDA\_1();

I2C\_SCL\_1();

BSP\_MyIIC\_Delay();

I2C\_SDA\_0();

BSP\_MyIIC\_Delay(); //SDA 和SDL相继跳变

I2C\_SCL\_0();

BSP\_MyIIC\_Delay();

}

### 大气压力传感器实验

使用 GPIO 模拟 IIC 通信，同三轴加速度传感器

### 磁场强度传感器实验

使用 GPIO 模拟 IIC 通信，同三轴加速度传感器

### 光照强度传感器实验

使用I2C总线轮询

### 温湿度传感器实验

同震动传感器，采用外部中断和GPIO

### 颜色传感器实验

同震动传感器，采用外部中断和GPIO，检测到颜色值改变时产生中断，同时在对RGB分量论询，在采样一个分量的时候，关闭另外两个分量的使能关

### 薄膜压力传感器实验

采用ADC，进行周期采样和转换

### 单轴倾角传感器实验

采用ADC，进行周期采样和转换

### 铂电阻传感器实验

采用ADC，进行周期采样和转换

## 实验步骤

### 硬件环境准备

1. 插上电源想
2. 连接JLINK
3. 打开电源开关

### 软件环境配置

1. 安装USB 转串口芯片 CP2102 的驱动
2. 打开串口调试助手并选定串口号
3. 打开光盘对应核心板的工程，编译无误后，可以通过 MDK 的 Download 按钮下载程序写到核心板。
4. 利用串口调试助手观察实验现象：串口打印菜单信息，根据菜单提示信息在串口输入信息控制传感器

## 实验过程与结果分析

实验过程中对于综合测试程序和所有能够使用的传感器进行了测试，此处只列举具有明显现象和描述价值的实验过程

### 雨雪传感器

##### 实验过程

* 1. 将平台通电，通过J-link链接到PC端，同时连接PC和平台之间的串口数据线（实现双工通信）。
  2. 打开对应实验工程，编译连接通过后，烧写程序到MCU，打开串口调试助手并配置（默认配置）。
  3. 实验结果：串口打印菜单信息，用手指轻按传感器上的长条金属片，串口打印报警信息。

##### 实验结果

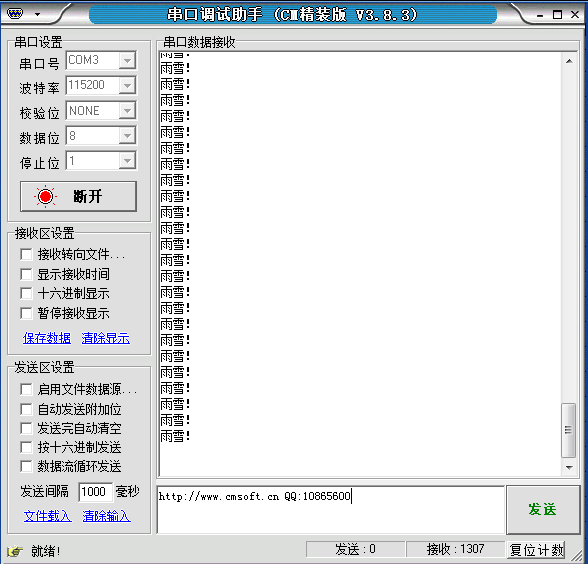


图1.7 雨雪传感器测试图

##### 实验分析

实际过程中，需要使用较大的力量才能达到传感器阈值

##### 应用场景

用于[气象](https://baike.baidu.com/item/%E6%B0%94%E8%B1%A1" \t "/home/hover/Documents\\x/_blank)、海洋、环境、机场、港口、实验室、工农业及交通等领域的雨雪有无定性测量。

### 光照传感器

##### 实验过程

1. 将平台通电，通过J-link链接到PC端，同时连接PC和平台之间的串口数据线（实现双工通信）。
2. 打开对应实验工程，编译连接通过后，烧写程序到MCU，打开串口调试助手并配置（默认配置）。
3. 实验结果：串口打印菜单信息（截图中未显示），不断输出当前模式下光照强度传感器测得 的结果。

##### 实验结果

串口打印菜单信息（截图中未显示），不断输出当前模式下光照强度传感器测得 的结果。

##### 实验分析

实际过程中，需要遮挡附近的光照传感器光源

##### 应用场景



图1.8 光照传感器应用场景

### 单轴倾角传感器实验

##### 实验过程

* 1. 将平台通电，通过J-link链接到PC端，同时连接PC和平台之间的串口数据线（实现双工通信）。
  2. 打开对应实验工程，编译连接通过后，烧写程序到MCU，打开串口调试助手并配置（默认配置）。
  3. 实验结果：串口打印菜单信息，不断输出单轴倾角传感器采集到的数据。倾斜传感器，可看到数据改变。

##### 实验结果

串口打印菜单信息，不断输出单轴倾角传感器采集到的数据。倾斜传感器，可看到数据改变。

##### 实验分析

倾斜箱体，需注意串口通信的连接处，更好的方法是将其进行茶歇

##### 应用场景

倾角传感器用于各种测量角度的应用中。例如,高精度激光仪器水平、工程机械设备调零。



图1.9 单轴倾角传感器应用场景-相机姿态确定

### 三轴加速度传感器实验

##### 实验过程

* 1. 连接 J-Link， 使用 mini USB 连接线连接电脑与底板 J2 串口， 连接 5V 电源给平台供电。
  2. 打开平台开关。
  3. 打开串口调试助手
  4. 打开光盘对应核心板的工程， 编译无误后， 可以通过 MDK 的 Download 按钮下载程序写到核心板。
  5. 利用串口调试助手观察实验现象： 串口首先打印菜单和初始化信息， 然后不断输出三轴加速度计
  6. 传感器采集到的数据； 更改传感器的姿态， 可以看到 X/Y/Z 三个轴测量数据的变化。

##### 实验结果

传感器采集到的数据； 更改传感器的姿态， 可以看到 X/Y/Z 三个轴测量数据的变化。

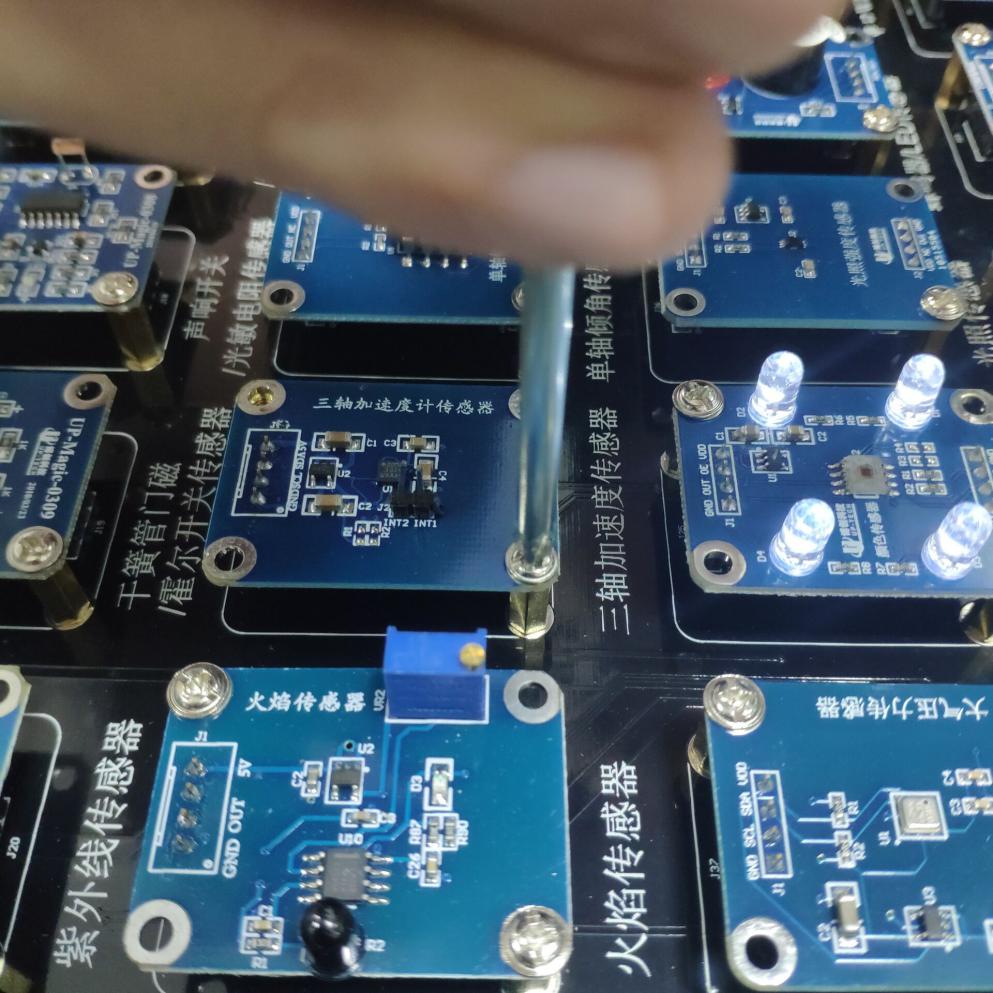


图1.10 拆卸三轴加速度传感器



图1.11 三轴传感器测试图

##### 实验分析

敏感度十分高，对于需要测量量需短时间平均值处理。

##### 应用场景



图1.12 三轴加速度传感器引用场景-手环运动检测

### 传感器板载测试

拷贝上板载默认测试程序，对所有传感器进行板载显示

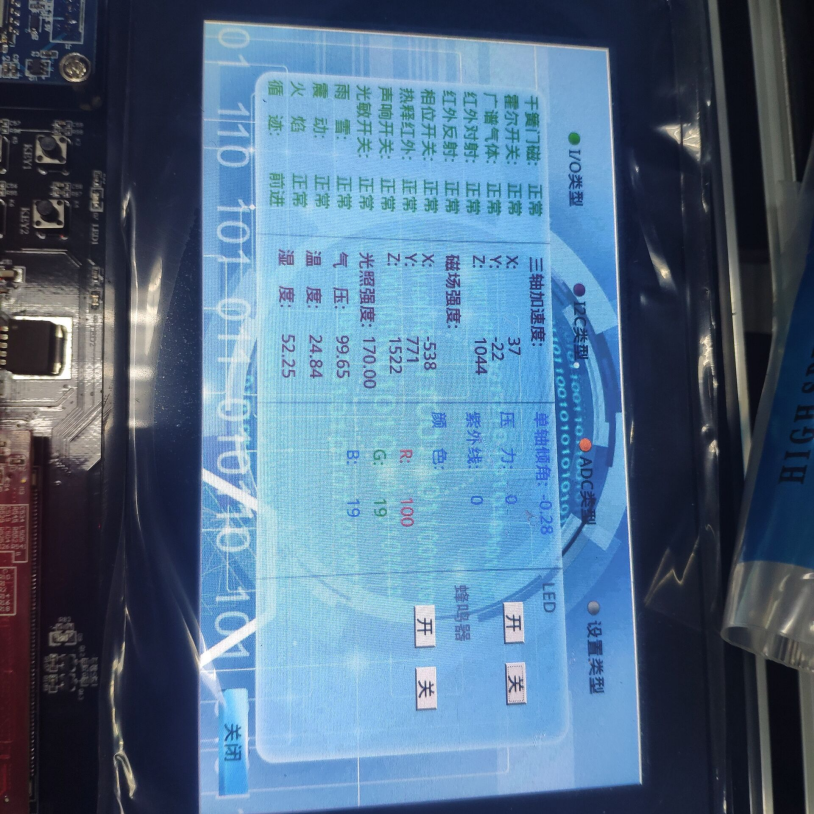


图1.13 传感器板载测试

## 实验问题分析

### 串口助手输入

1. 问题

存在输入框锁死，被广告信息占用

1. 分析

程序BUG

1. 解决

多发送几次

### 传感器过于敏感

1. 问题

部分传感器（如三轴传感器）过于敏感

1. 分析

程序轮询时间过短

1. 解决
   1. 软件方法：改变总线问询频率
   2. 统计方法：短时间内多次取平均值

### 传感器过于迟钝

1. 问题

部分传感器（如雨雪传感器）过于迟钝

1. 分析
   1. 可能传感器本身老化
   2. 采用标准强度进行测试

## 实验总结

实验过程，基于实验平台的硬件设置和软件代码对于平台进行了综合分析，对于不同的传感器，由于硬件的不同，其软件的交互方式也不同，有采用GPIO直接控制引脚高低电平，有采用总线轮询，也有采用外部中断的。

其中比较特殊的是不同通讯协议之间的转换如利用GPIO 模拟 IIC 时序，在不同的硬件之间通过软件实现了模拟。

尝试了不同的传感器及其敏感度，并对试验箱的相关原理做了分析，为以后的实验打下了基础。

## 参考文献

1. 传感器原理与应用教学平台实验指导书
2. JTAG调试原理.URL:

<https://blog.csdn.net/sinat_24088685/article/details/50980501>

1. STM32 中断优先级相关概念与使用笔记

# 颜色传感器实验

## 实验目的

通过实验，更多地了解颜色传感器的特性、测量方法、性能特点及可能的应用

## 实验原理

TCS3200 是 TAOS 公司推出的可编程彩色光到频率的转换器，它把可配置的硅光电二极管与电流频率转换器集成在一个单一的 CMOS 电路上，同时在单一芯片上集成了红绿蓝（RGB）三种滤光器，是业界第一个有数字兼容接口的 RGB 彩色传感器，TCS3200 的输出信号是数字量，可以驱动标准的 TTL 或 CMOS逻辑输入，因此可直接与微处理器或其他逻辑电路相连接，由于输出的是数字量，并且能够实现每个彩色信道 10 位以上的转换精度，因而不再需要 A/D 转换电路，使电路变得更简单，图2.1 是 TCS3200的引脚和功能框图。

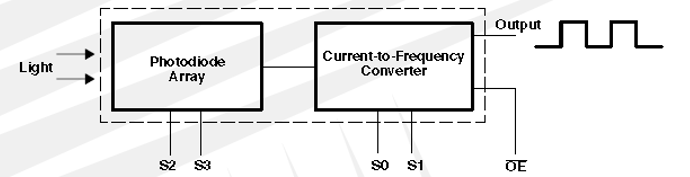


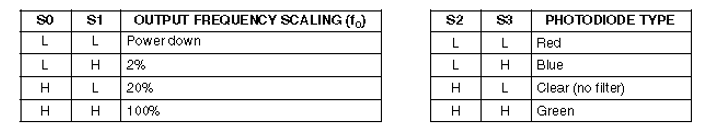
图2.1 TCS3200的引脚和功能框图

图2.1 中，TCS3200采用8引脚的SOIC表面贴装式封装，在单一芯片上集成有64个光电二极管，这些二极管分为四种类型，其16个光电二极管带有红色滤波器；16个光电二极管带有绿色滤波器；16个光电二极管带有蓝色滤波器，其余16个不带有任何滤波器，可以透过全部的光信息，这些光电二极管在芯片内是交叉排列的，能够最大限度地减少入射光辐射的不均匀性，从而增加颜色识别的精确度；另一方面，相同颜色的16个光电二极管是并联连接的，均匀分布在二极管阵列中，可以消除颜色的位置误差。工作时，通过两个可编程的引脚来动态选择所需要的滤波器，该传感器的典型输出频率范围从2Hz－500kHz，用户还可以通过两个可编程引脚来选择100％、20％或2％的输出比例因子，或电源关断模式。输出比例因子使传感器的输出能够适应不同的测量范围，提高了它的适应能力。例如，当使用低速的频率计数器时，就可以选择小的定标值，使TCS3200 的输出频率和计数器相匹配。

从图可知：当入射光投射到TCS3200上时，通过光电二极管控制引脚S2、S3的不同组合，可以选择不同的滤波器；经过电流到频率转换器后输出不同频率的方波（占空比是50％），不同的颜色和光强对应不同频率的方波；还可以通过输出定标控制引脚S0、S1，选择不同的输出比例因子，对输出频率范围进行调整，以适应不同的需求。

下面简要介绍TCS3200芯片各个引脚的功能及它的一些组合选项。 S0、S1 用于选择输出比例因子或电源关断模式；S2、S3 用于选择滤波器的类型；OE 反是频率输出使能引脚，可以控制输出的状态，当有多个芯片引脚共用微处理器的输出引脚时，也可以作为片选信号，OUT 是频率输出引脚，GND 是芯片的接地引脚，VCC 为芯片提供工作电压，表 1 是 S0、S1 及 S2、S3 的可用组合。

表2.1 传感器引脚组合表



三原色的感应原理 ，通常所看到的物体颜色，实际上是物体表面吸收了照射到它上面的白光（日光）中的一部分有色成分，而反射出的另一部分有色光在人眼中的反应。白色是由各种频率的可见光混合在一起构成的，也就是说白光中包含着各种颜色的色光（如红 R、黄 Y、绿 G、青 V、蓝 B、紫 P）。根据德国物理学家赫姆霍兹（Helinholtz）的三原色理论可知，各种颜色是由不同比例的三原色（红、绿、蓝）混合而成的。

TCS3200 识别颜色的原理 ，由三原色感应原理可知，如果知道构成各种颜色的三原色的值，就能够知道所测试物体的颜色。对于 TCS3200 来说，当选定一个颜色滤波器时，它只允许某种特定的原色通过，阻止其他原色的通过。例如：当选择红色滤波器时，入射光中只有红色可以通过，蓝色和绿色都被阻止，这样就可以得到红色光的光强；同时，选择其他的滤波器，就可以得到蓝色光和绿色光的光强。通过这三个值，就可以分析投射到 TCS3200 传感器上的光的颜色。 白平衡和颜色识别原理 ，白平衡就是告诉系统什么是白色。从理论上讲，白色是由等量的红色、绿色和蓝色混合而成的；但实际上，白色中的三原色并不完全相等，并且对于 TCS3200 的光传感器来说，它对这三种基本色的敏感性是不相同的，导致TCS3200的RGB输出并不相等，因此在测试前必须进行白平衡调整，使得 TCS3200 对所检测的"白色"中的三原色是相等的。进行白平衡调整是为后续的颜色识别作准备。

## 实验步骤

1. 连接 J-Link， 连接 USB 线至底板 J2 串口处， 连接 5V 电源给开发板供电。
2. 打开电源开关。
3. 打开串口调试助手
4. 打开光盘对应核心板的工程， 编译无误后， 可以通过 MDK 的 Download 按钮下载程序写到核心板。
5. 利用串口调试助手观察实验现象： 串口打印菜单信息， 不断输出颜色传感器采集到的 RGB 值及经过程序处理出的 HSV 值。

## 实验过程与结果分析

### 实验一

建立颜色显示空间与颜色测试空间的关系。即，当显示颜色值为（r1,g1,b1）时，测得值为（r2,g2,b2），测 20 组不同颜色值。



图2.2 颜色设定图

表2.2 颜色传感器结果纪录表

|  |  |  |  |
| --- | --- | --- | --- |
| R1 | G1 | R2 | G2 |
| 0 | 255 | 45 | 195 |
| 23 | 230 | 36 | 160 |
| 26 | 28 | 1 | 3 |
| 47 | 207 | 32 | 122 |
| 48 | 45 | 5 | 5 |
| 52 | 182 | 25 | 90 |
| 67 | 163 | 26 | 68 |
| 67 | 66 | 11 | 9 |
| 83 | 136 | 28 | 45 |
| 83 | 80 | 19 | 14 |
| 95 | 97 | 28 | 22 |
| 106 | 106 | 36 | 25 |
| 125 | 115 | 56 | 32 |
| 134 | 112 | 73 | 35 |
| 141 | 132 | 81 | 45 |
| 150 | 144 | 94 | 55 |
| 188 | 65 | 154 | 20 |
| 192 | 189 | 162 | 96 |
| 217 | 36 | 215 | 22 |
| 239 | 236 | 255 | 183 |
| 255 | 0 | 255 | 33 |
| 255 | 255 | 255 | 198 |

### 实验二

当固定r1=100、g1=50时，画出b1与b2的对应关系 曲线，并写出二次多项式拟合的关系式。

最小二乘法线性回归

**from sklearn import** linear\_model

model = linear\_model.LinearRegression()

model.fit(X,Y)**print**(model.score(X,Y))

**print**('W='+str(model.coef\_))**print**('b='+str(model.intercept\_))

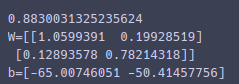


图2.3 最小二乘法拟合结果

### 实验三

当b1固定为180时，分别拟合出r1,g1与（r2,g2）的二元二次曲面的关系。

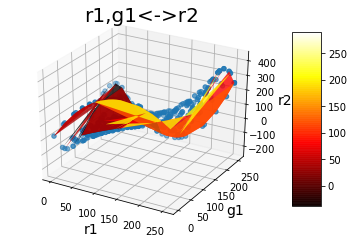


图2.4 曲面拟合结果1

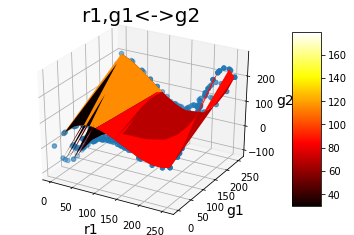


图2.5 曲面拟合结果2

### 实验四，五

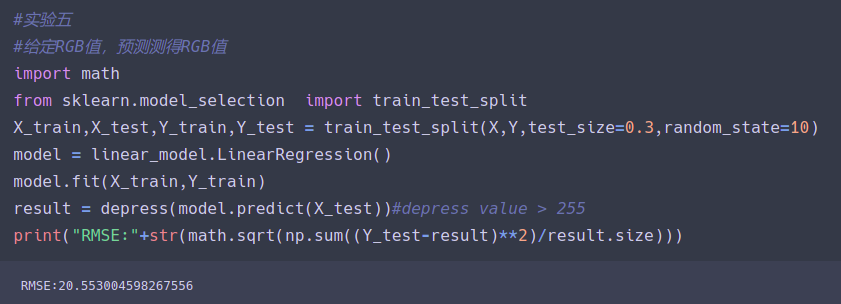


图2.6 模型预测结果

## 源码分析

结构体存储

**typedef struct**

{

uint16\_t Rgena;

uint16\_t Ggena;

uint16\_t Bgena;

}\_RGB;

引脚绑定

#define OUT PCin(13)

#define S2 PEout(5)

#define S3 PEout(6)

GPIO通信

static void BSP\_TCS3200\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOC | RCC\_APB2Periph\_GPIOE, ENABLE);

//S0--->3.3V S1--->3.3V nOE-->PF9 S2--->PE5 S3--->PE6

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_13;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOC, &GPIO\_InitStructure);

GPIO\_ResetBits(GPIOC,GPIO\_Pin\_13); // nOE 使能

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_5;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOE, &GPIO\_InitStructure);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_6;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOE, &GPIO\_InitStructure);

GPIO\_SetBits(GPIOE,GPIO\_Pin\_5); //S2=1 S3=0

GPIO\_ResetBits(GPIOE,GPIO\_Pin\_6);

}

## 应用场景

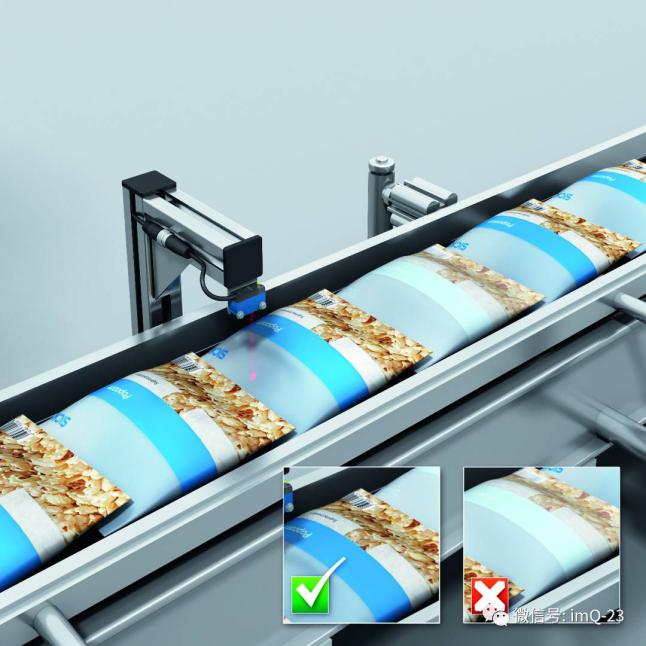


图2.7 颜色传感器做流水线检测

颜色实际上可以是多种反馈量的转换，如检测包装是否正常，食品是否变质。

## 实验总结

本次实验熟悉了颜色传感器的相应使用及模型的相应训练方法，由于引脚数目的不匹配，进行GPIO通信的时候需要进行引脚的对应绑定。实验过程中，因为传感器的抖动造成数值滚动，需均匀采样。需要注意传感器与屏幕之间的夹脚可能会对实验结果有一定的影响。

数值处理上，使用机器学习和线性拟合的不同方式对数据进行了处理，并进行了相应的可视化工作。