

华中科技大学

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

华中科技大学课程实验报告

目 录

1 实验平台的熟悉和使用实验.....	1
1.1 实验目的.....	1
1.2 实验原理.....	1
1.3 实验步骤.....	17
1.4 实验过程与结果分析.....	18
1.5 实验问题分析.....	25
1.6 实验总结.....	26
1.7 参考文献.....	26
2 颜色传感器实验.....	27
2.1 实验目的.....	27
2.2 实验原理.....	27
2.3 实验步骤.....	29
2.4 实验过程与结果分析.....	29
2.5 源码分析.....	33
2.6 应用场景.....	34
2.7 实验总结.....	35

1 实验平台的熟悉和使用实验

1.1 实验目的

了解各种传感器

1.2 实验原理

1.2.1 实验箱构成



图 1.1 试验箱组成图

华中科技大学课程实验报告

1.2.2 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。

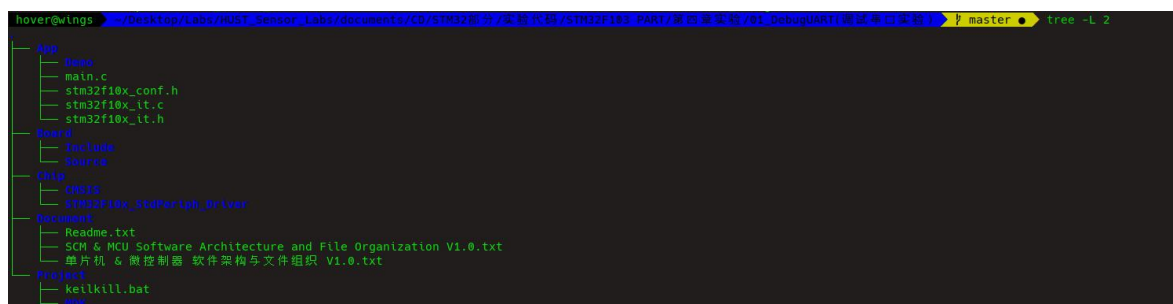
2) 连接方式

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。

3) 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

1.2.3 程序架构



```
hover@wings: ~/Desktop/Labs/IMST_Sensor_Labs/documents/CD/STM32F10x/实验代码/STM32F10x PART1/实验代码/vcl_DebugUART/实验代码/1.1
tree -L 2
.
├── App
│   ├── Bsp
│   ├── main.c
│   ├── 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
│   └── bsp
```

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

华中科技大学课程实验报告

```
.
├── 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
```

华中科技大学课程实验报告

1) APP

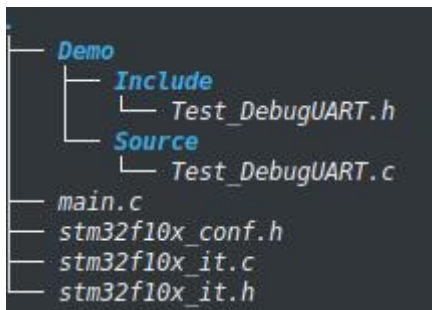
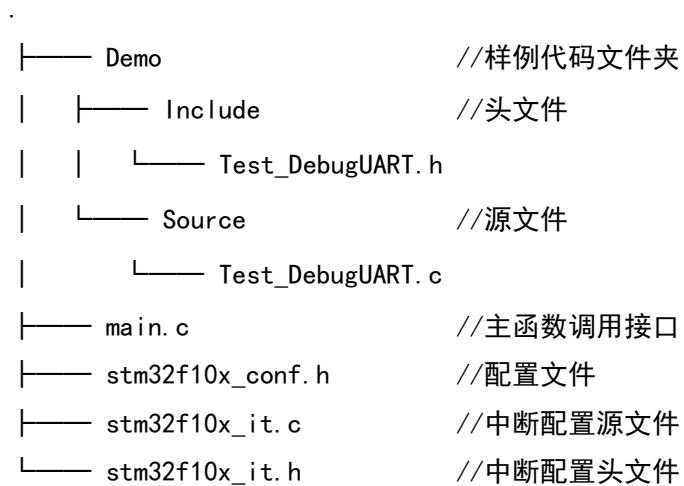
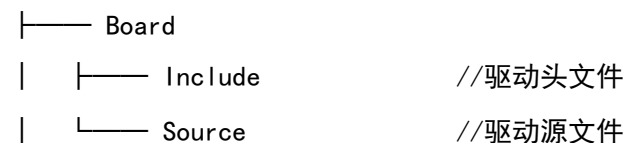


图 1.3 APP 代码架构图

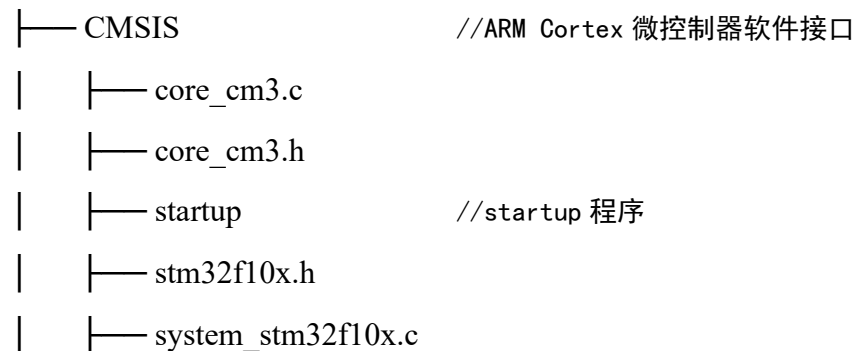


2) Board

存放基础驱动文件



3) Chip



华中科技大学课程实验报告

```
|  └─ system_stm32f10x.h
└─ STM32F10x_StdPeriph_Driver//STM32 驱动程序
```

1.2.4 串口实验

1) 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；

2) 源码注释

a) 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"
```

华中科技大学课程实验报告

```
#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();
```

```
}
```

b) 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 通过引脚的高低电平实现功能

华中科技大学课程实验报告

1.2.5 LED 蜂鸣器模块实验

1) 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。

2) 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;
}
```

华中科技大学课程实验报告

```
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);
}
```

1.2.6 震动传感器终端实验

1) 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

2) 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
}
```

1.2.7 红外对射传感器实验

1) 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

华中科技大学课程实验报告

2) 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
}
```

1.2.8 热释红外传感器实验

同红外对射传感器，采用定时器中断

1.2.9 光谱气体传感器实验

同红外对射传感器，采用定时器中断

1.2.10 雨雪传感器实验

同红外对射传感器，采用定时器中断

华中科技大学课程实验报告

1.2.11 干簧门磁霍尔开关模块实验

1) 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

2) 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
```

华中科技大学课程实验报告

```
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
}
```

1.2.12 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

1.2.13 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

1.2.14 循迹传感器实验

1) 循迹传感器原理

同红外对射传感器，采用定时器中断

2) 源码注释

华中科技大学课程实验报告

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;
            }
        }
    }
}
```

华中科技大学课程实验报告

```
        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
}
```

1.2.15 三轴加速度计传感器实验

1) 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。

华中科技大学课程实验报告

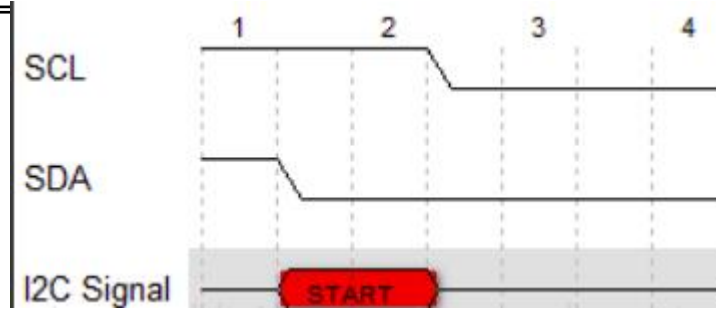


图 1.4 START 对应信号图

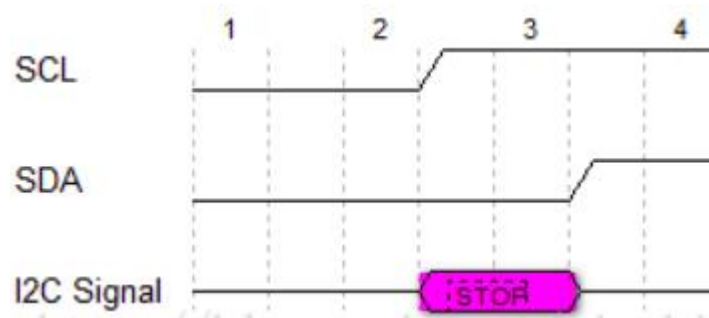


图 1.5 STOP 对应信号图

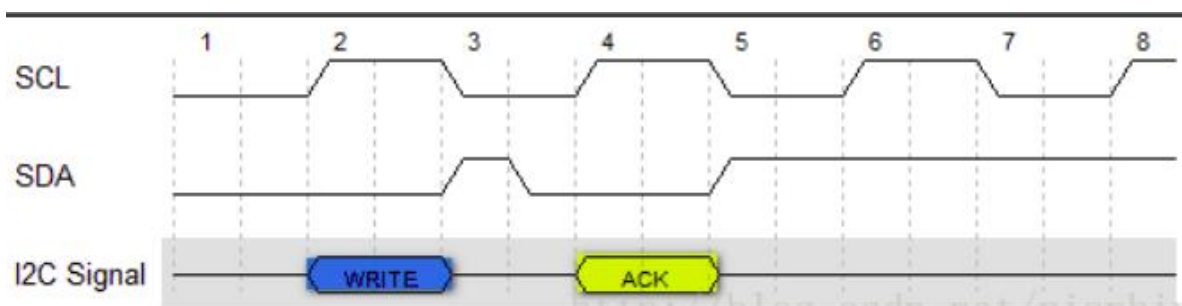


图 1.6 WRITE 和 ACK 对应信号图

2) 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
```


华中科技大学课程实验报告

```
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();
}
```

1.2.16 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

1.2.17 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

1.2.18 光照强度传感器实验

使用 I2C 总线轮询

1.2.19 温湿度传感器实验

同震动传感器，采用外部中断和 GPIO

1.2.20 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 分量轮询，在采样一个分量的时候，关闭另外两个分量的使能关

1.2.21 薄膜压力传感器实验

采用 ADC，进行周期采样和转换

1.2.22 单轴倾角传感器实验

采用 ADC，进行周期采样和转换

1.2.23 铂电阻传感器实验

采用 ADC，进行周期采样和转换

1.3 实验步骤

1.3.1 硬件环境准备

- 1) 插上电源想
- 2) 连接 JLINK
- 3) 打开电源开关

1.3.2 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

1.4 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

1.4.1 雨雪传感器

1) 实验过程

- 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）。
- 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）。
- 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。

2) 实验结果

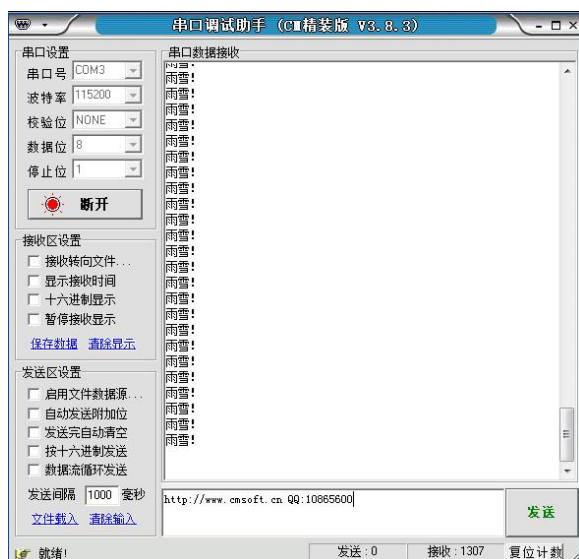


图 1.7 雨雪传感器测试图

华中科技大学课程实验报告

3) 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

4) 应用场景

用于气象、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。

1.4.2 光照传感器

1) 实验过程

a) 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）。

b) 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）。

c) 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。

2) 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。

3) 实验分析

实际过程中，需要遮挡附近的光照传感器光源

4) 应用场景



图 1.8 光照传感器应用场景

1.4.3 单轴倾角传感器实验

1) 实验过程

- a) 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）。
- b) 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）。
- c) 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。

2) 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。

华中科技大学课程实验报告

3) 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

4) 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

1.4.4 三轴加速度传感器实验

1) 实验过程

- 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。
- 打开平台开关。
- 打开串口调试助手
- 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。
- 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

华中科技大学课程实验报告

f) 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。

2) 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。

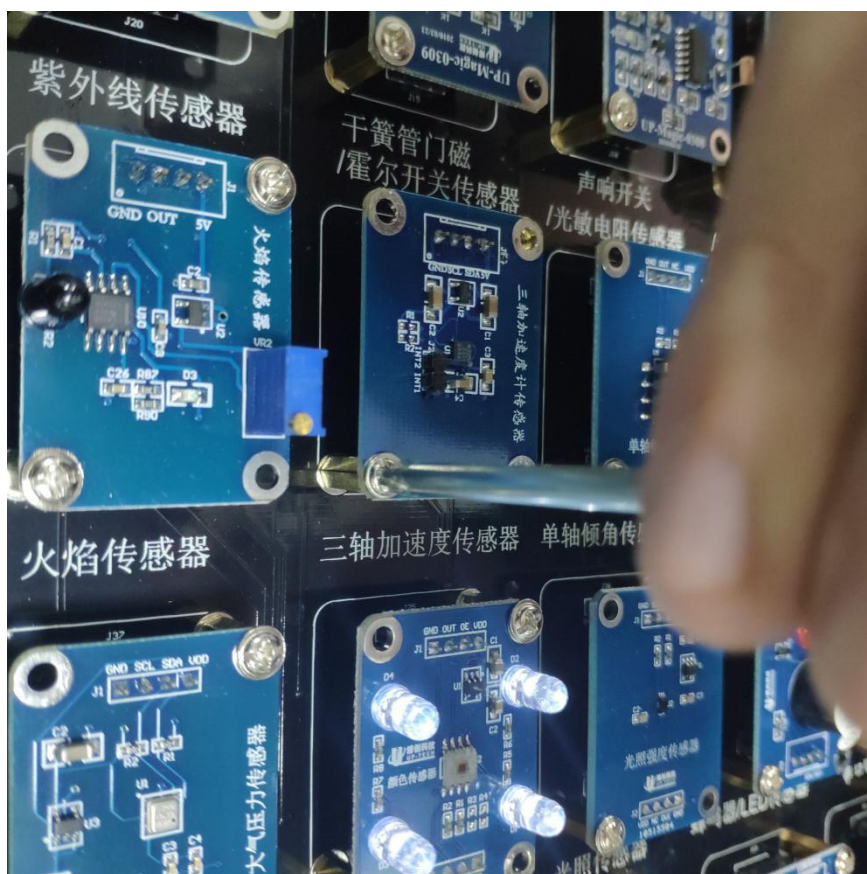


图 1.10 拆卸三轴加速度传感器

华中科技大学课程实验报告



图 1.11 三轴传感器测试图

3) 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。

华中科技大学课程实验报告

4) 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

1.4.5 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

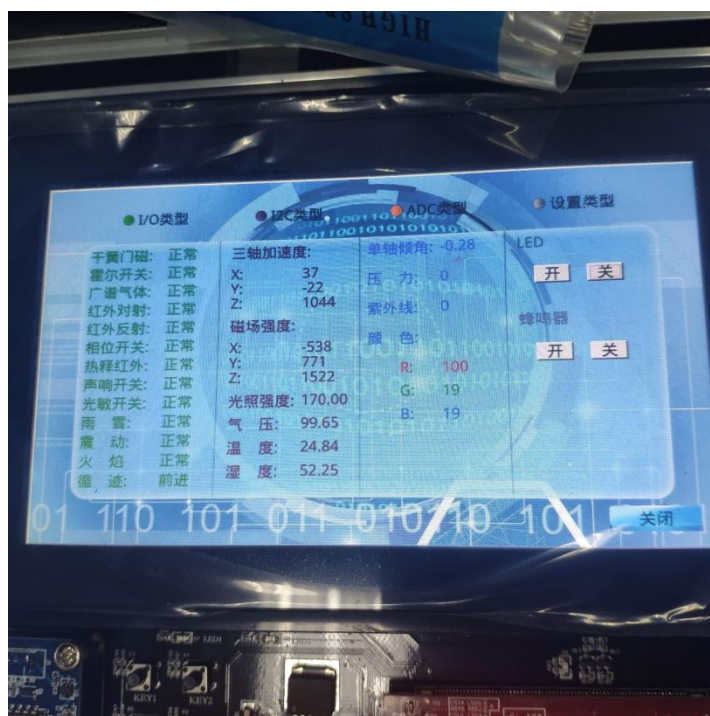


图 1.13 传感器板载测试

1.5 实验问题分析

1.5.1 串口助手输入

1) 问题

存在输入框锁死，被广告信息占用

2) 分析

程序 BUG

3) 解决

多发送几次

1.5.2 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

2) 分析

程序轮询时间过短

3) 解决

a) 软件方法：改变总线问询频率

b) 统计方法：短时间内多次取平均值

1.5.3 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

2) 分析

a) 可能传感器本身老化

b) 采用标准强度进行测试

1.6 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。

1.7 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

[3] STM32 中断优先级相关概念与使用笔记

2 颜色传感器实验

2.1 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

2.2 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。

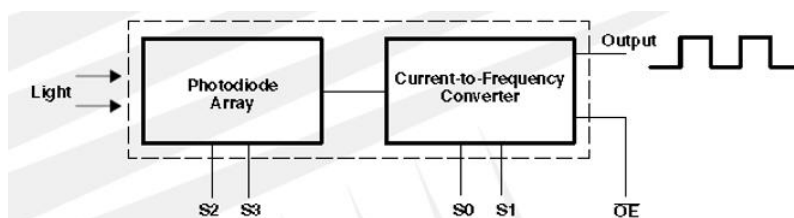


图 2.1 TCS3200 的引脚和功能框图

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

华中科技大学课程实验报告

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

S0	S1	OUTPUT FREQUENCY SCALING (f_o)	S2	S3	PHOTODIODE TYPE
L	L	Power down	L	L	Red
L	H	2%	L	H	Blue
H	L	20%	H	L	Clear (no filter)
H	H	100%	H	H	Green

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（Helmholtz）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

华中科技大学课程实验报告

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。

2.3 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。

2.4 实验过程与结果分析

2.4.1 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

华中科技大学课程实验报告

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

华中科技大学课程实验报告

2.4.2 实验二

当固定 $r_1=100$ 、 $g_1=50$ 时，画出 b_1 与 b_2 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_))
```

```
0.8830031325235624  
W=[[1.0599391  0.19928519]  
    [0.12893578 0.78214318]]  
b=[-65.00746051 -50.41457756]
```

图 2.3 最小二乘法拟合结果

2.4.3 实验三

当 b_1 固定为 180 时，分别拟合出 r_1, g_1 与 (r_2, g_2) 的二元二次曲面的关系。

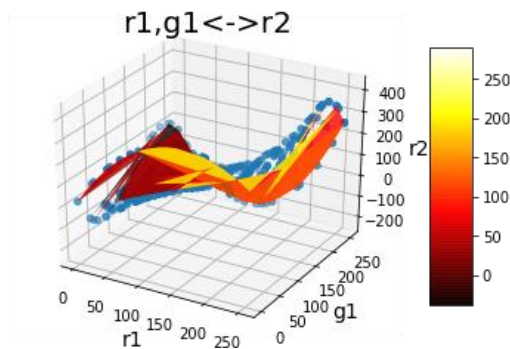


图 2.4 曲面拟合结果 1

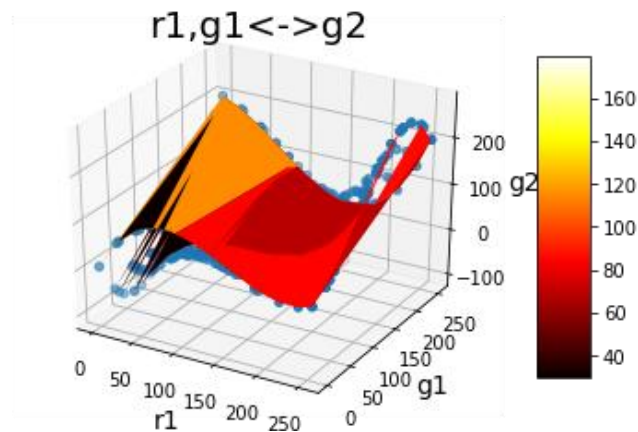


图 2.5 曲面拟合结果 2

2.4.4 实验四，五

```
#实验五
#给定RGB值，预测测得RGB值
import math
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.3,random_state=10)
model = linear_model.LinearRegression()
model.fit(X_train,Y_train)
result = depress(model.predict(X_test))#depress value > 255
print("RMSE:"+str(math.sqrt(np.sum((Y_test-result)**2)/result.size)))

RMSE:20.553004598267556
```

图 2.6 模型预测结果

2.5 源码分析

结构体存储

```
typedef struct
{
    uint16_t Rgen;
    uint16_t Ggen;
    uint16_t Bgen;
}_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);
```

华中科技大学课程实验报告

```
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);

}
```

2.6 应用场景



图 2.7 颜色传感器做流水线检测

华中科技大学课程实验报告

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。

2.7 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。