

**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](#_Toc1975886384)

[1.1 实验目的 1](#_Toc1560307071)

[1.2 实验原理 1](#_Toc2077837209)

[1.3 实验步骤 17](#_Toc1063641238)

[1.4 实验过程与结果分析 18](#_Toc607382700)

[1.5 实验问题分析 25](#_Toc644834684)

[1.6 实验总结 26](#_Toc220510539)

[1.7 参考文献 26](#_Toc620529490)

[2 颜色传感器实验 28](#_Toc752064835)

[2.1 实验目的 28](#_Toc1129707775)

[2.2 实验原理 28](#_Toc1741160450)

[2.3 实验步骤 30](#_Toc1943679781)

[2.4 实验过程与结果分析 30](#_Toc1106211926)

[2.5 源码分析 34](#_Toc327305250)

[2.6 应用场景 35](#_Toc435482362)

[2.7 实验总结 36](#_Toc1362243428)

[3 磁场强度和三轴加速度传感器实验 38](#_Toc1101709878)

[3.1 实验目的 38](#_Toc271826264)

[3.2 实验原理 38](#_Toc1196153579)

[3.3 实验步骤 39](#_Toc26416758)

[3.4 实验过程及结果 40](#_Toc1719506853)

[3.5 实验问题 50](#_Toc278608944)

[3.6 应用场景 51](#_Toc384696634)

[3.7 实验总结 53](#_Toc1254390769)

[4 传感器综合 55](#_Toc959112711)

[4.1 实验目的 55](#_Toc133286977)

[4.2 实验原理 55](#_Toc61204105)

[4.3 实验步骤 57](#_Toc219976257)

[4.4 实验过程及结果 57](#_Toc831661736)

[4.5 实验问题 67](#_Toc1656951142)

[4.6 应用场景 68](#_Toc1874412034)

[4.7 实验总结 68](#_Toc660064472)

[4.8 参考文献 69](#_Toc1069774565)

# 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

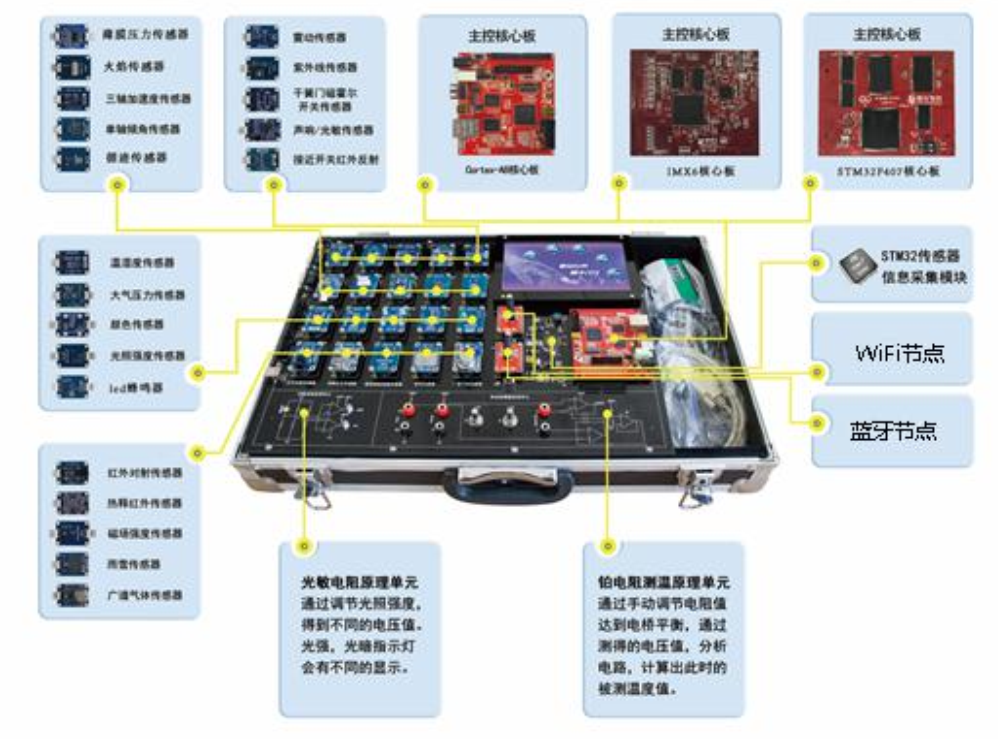


图1.1 试验箱组成图

### 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

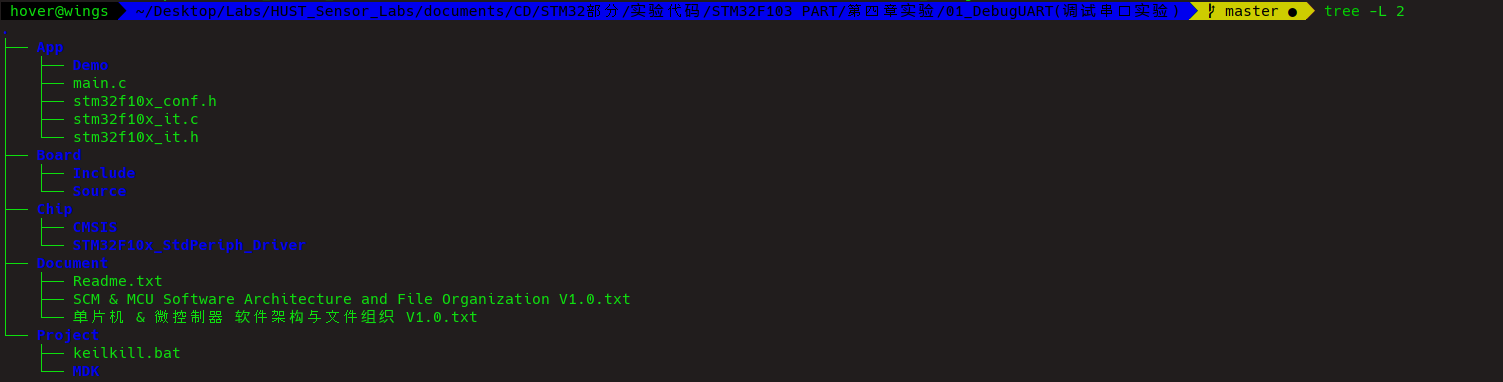


图1.2 实验代码架构图

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

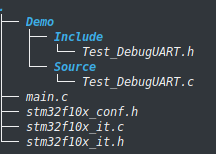


图1.3 APP代码架构图

.

├── 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。

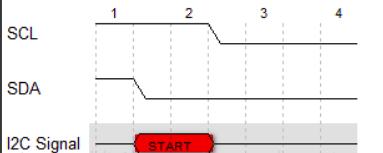


图1.4 START对应信号图

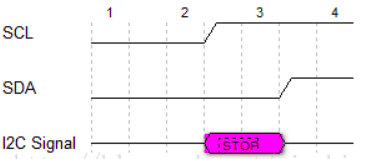


图1.5 STOP对应信号图

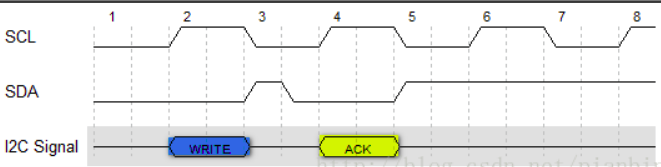


图1.6 WRITE和ACK对应信号图

##### 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

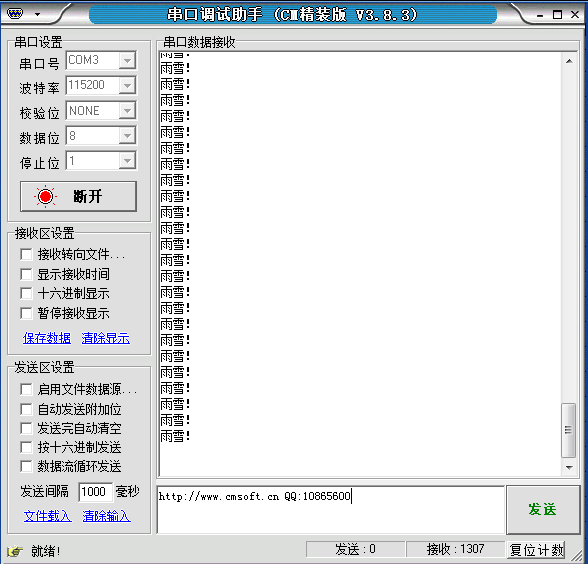


图1.7 雨雪传感器测试图

##### 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。

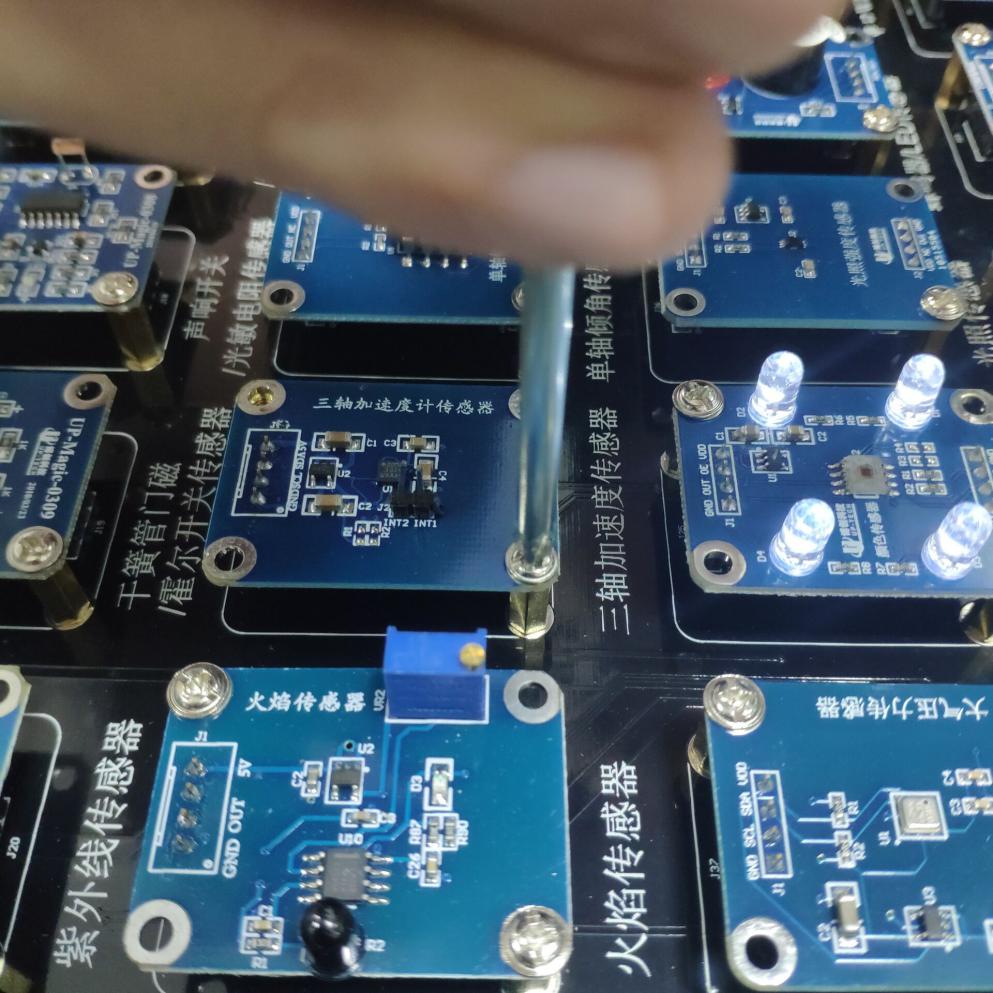


图1.10 拆卸三轴加速度传感器



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

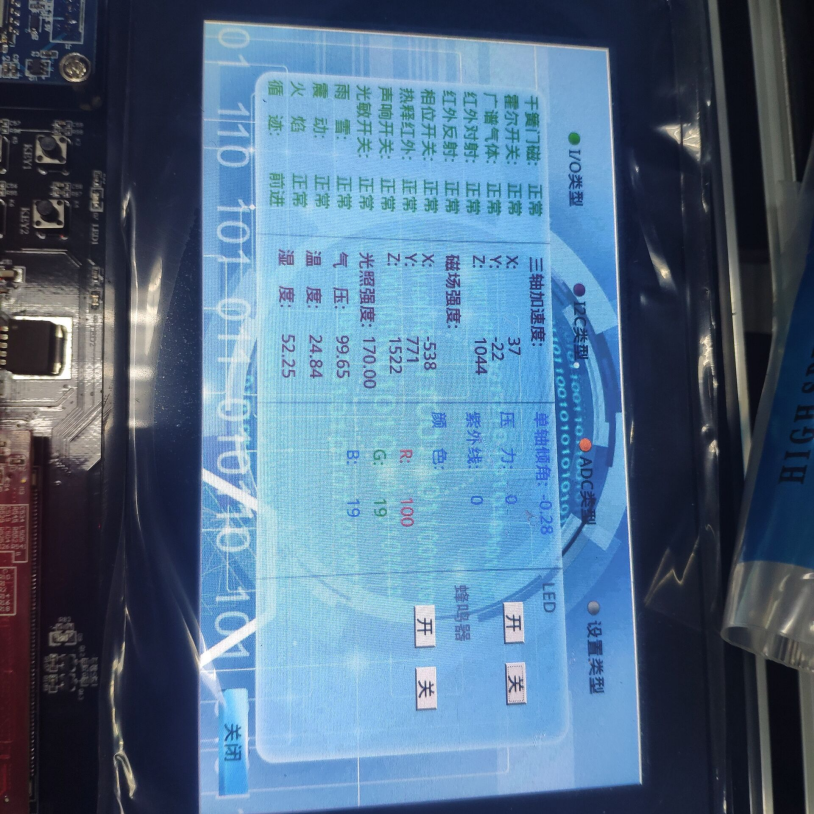


图1.13 传感器板载测试

## 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



**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**

# 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。

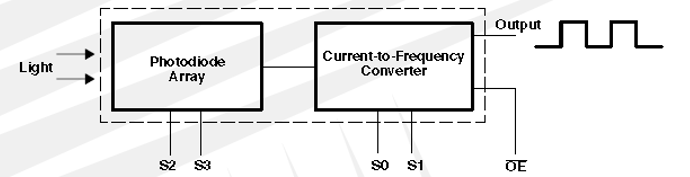


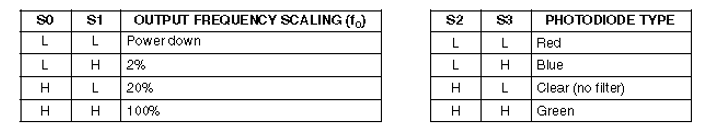
图2.1 TCS3200的引脚和功能框图

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\_))

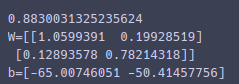


图2.3 最小二乘法拟合结果

### 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。

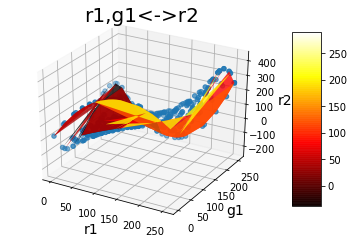


图2.4 曲面拟合结果1

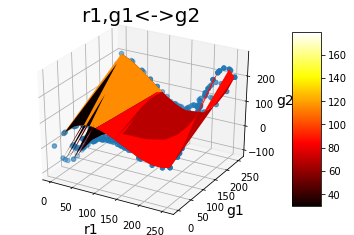


图2.5 曲面拟合结果2

### 实验四，五

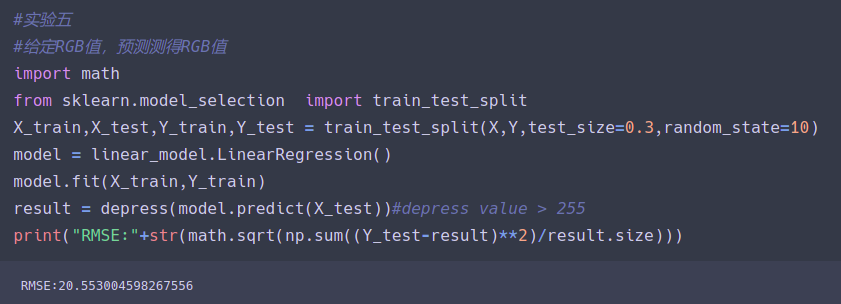


图2.6 模型预测结果

## 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。



**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 4月**

**计算机科学与技术学院**

# 磁场强度和三轴加速度传感器实验

## 实验目的

通过实验，更多地了解磁场强度和三轴加速度传感器的特性、测量方法、性能特点及可能的应用

## 实验原理

### 磁场强度传感器

磁场强度传感器是可以将各种磁场及其变化的量转变成电信号输出的装置。自然界和人类社会生活的许多地方都存在磁场或与磁场相关的信息。在当今的信息社会中，磁场传感器已成为信息技术和信息产业中不可缺少的基础元件。



图3.1 磁场强度传感器

### 三轴加速度计传感器

三轴加速度计传感器是基于加速度的基本原理去实现工作的，本质上属于惯性力传感器，使用的是牛顿第二运动定律：F=ma（物体所受到的合力等于其质量与其加速度的乘积，矢量运算）。



图3.2 三轴加速度传感器

## 实验步骤

1. 用磁铁改变磁场强度传感器周边的磁场强度、方向，探索测量结果与磁场情况之间的规律，判别传感器的X、Y、Z轴的走向，判断两个磁铁产生的磁场叠加在一起对测量结果的影响。
2. 用三轴加速度传感器
   1. 测量斜放着的书的坡度
   2. 测量传感器作圆周运动时的运动半径。

## 实验过程及结果

### 磁场传感器

其中，浅色方块代表传感器，而深色边框的每一个面进行放置磁铁，从而获得六组数据，在此基础上，给出任意角度的倾斜角度预测。

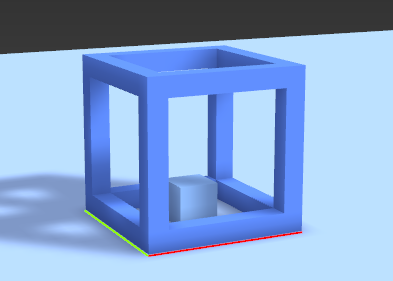


图3.3 磁场传感演示图

分别在不放置磁铁，和磁铁不同朝向的情况下进行放置，测得七组数据如下

表3.1 不放置磁铁数据记录表

X=-659,Y=-2583,Z=-702

X=-655,Y=-2582,Z=435

X=-660,Y=-2578,Z=443

X=-656,Y=-2581,Z=442

X=-659,Y=-2580,Z=438

X=-660,Y=-2571,Z=450

X=-658,Y=-2576,Z=442

X=-656,Y=-2576,Z=449

X=-666,Y=-2575,Z=443

X=-659,Y=-2573,Z=434

X=-661,Y=-2574,Z=441

表3.2 磁铁向下放置数据记录表

X=232,Y=99,Z=1862

X=232,Y=106,Z=1852

X=231,Y=103,Z=1871

X=236,Y=104,Z=1863

X=233,Y=107,Z=1851

X=232,Y=98,Z=1846

X=232,Y=104,Z=1863

X=229,Y=105,Z=1843

X=226,Y=103,Z=1870

X=231,Y=102,Z=1865

X=229,Y=101,Z=1878

表3.3 磁铁向上放置数据记录表

X=28812,Y=-6896,Z=-32768

X=28812,Y=-7079,Z=-32768

X=28812,Y=-7034,Z=-32768

X=28812,Y=-6993,Z=-32768

X=28812,Y=-6978,Z=-32768

X=28818,Y=-7066,Z=-32768

X=28818,Y=-7131,Z=-32768

X=28818,Y=-7095,Z=-32768

X=28818,Y=-7191,Z=-32768

X=28818,Y=-7251,Z=-32768

X=28812,Y=-7190,Z=-32768

表3.4 磁铁向前放置数据记录表

X=6098,Y=2410,Z=-1490

X=6091,Y=2407,Z=-1491

X=6127,Y=2376,Z=-1427

X=6096,Y=2404,Z=-1478

X=6095,Y=2401,Z=-1507

X=6096,Y=2406,Z=-1493

X=6115,Y=2387,Z=-1443

X=6100,Y=2398,Z=-1472

X=6134,Y=2367,Z=-1401

X=6087,Y=2443,Z=-1572

X=6088,Y=2450,Z=-1564

表3.5 磁铁向后放置数据记录表

X=3235,Y=-2998,Z=-16183

X=3236,Y=-2992,Z=-16189

X=3227,Y=-2979,Z=-16201

X=3239,Y=-3001,Z=-16181

X=3258,Y=-3017,Z=-16139

X=3268,Y=-3032,Z=-16131

X=3291,Y=-3068,Z=-16073

X=3303,Y=-3085,Z=-16067

X=3317,Y=-3102,Z=-16068

X=3323,Y=-3111,Z=-16065

X=3344,Y=-3134,Z=-16004

表3.6 磁铁向左放置数据记录表

X=302,Y=6375,Z=28633

X=283,Y=6318,Z=28236

X=295,Y=6334,Z=28416

X=284,Y=6305,Z=28225

X=298,Y=6349,Z=28541

X=303,Y=6381,Z=28652

X=294,Y=6365,Z=28623

X=286,Y=6323,Z=28354

X=286,Y=6322,Z=28347

X=286,Y=6332,Z=28375

X=268,Y=6264,Z=27974

表3.7 磁铁向右放置数据记录表

X=-4181,Y=3,Z=631

X=-4173,Y=-2,Z=609

X=-4180,Y=6,Z=627

X=-4147,Y=-3,Z=546

X=-4122,Y=0,Z=531

X=-4149,Y=-4,Z=574

X=-4091,Y=-7,Z=467

X=-4075,Y=-7,Z=431

X=-4127,Y=-7,Z=536

X=-4101,Y=-6,Z=466

X=-4129,Y=-4,Z=520

基于上述数据，整理得表格如下：

表3.8 磁铁放置均值表

|  |  |  |  |
| --- | --- | --- | --- |
|  | X | Y | Z |
| 无磁铁NULL | -659 | -2583 | -702 |
| 上U | 232 | 102 | 1861 |
| 下D | 28812 | -7066 | -32768 |
| 左L | 6100 | 2403 | -1430 |
| 右R | 3235 | -3017 | -16183 |
| 前F | 296 | 6334 | 28623 |
| 后B | -4140 | 1 | 537 |

表3.9 左上45°

|  |  |  |
| --- | --- | --- |
| 23204 | -701 | 1660 |
| 23285 | -707 | 1683 |
| 23275 | -700 | 1676 |
| 23256 | -694 | 1698 |
| 23329 | -704 | 1673 |
| 23247 | -709 | 1682 |
| 23260 | -711 | 1677 |
| 23297 | -714 | 1672 |
| 23170 | -734 | 1698 |
| 23269 | -726 | 1705 |
| 23259.2 | -710 | 1682.4 |

表3.10 右上45°

|  |  |  |
| --- | --- | --- |
| -29085 | 52 | -32768 |
| -29090 | -4 | -32768 |
| -29090 | -40 | -32768 |
| -29101 | -53 | -32768 |
| -29090 | -57 | -32768 |
| -29101 | -33 | -32768 |
| -29090 | -75 | -32768 |
| -29101 | -109 | -32768 |
| -29096 | -95 | -32768 |
| -29090 | -125 | -32768 |
| -29093.4 | -53.9 | -32768 |

由上述可得，XYZ近乎关于某一中线对称。

-2921，-350，-15544

由于存在电路板的极化作用，磁铁存在的情况下，磁感线并非理想下均匀分布，加之采用的磁铁并非均匀磁铁，故测量数据并非严格对称，而是呈现趋势分布。在磁铁进行原理的过程中，磁场强度先增大后减小。

由于磁场强度为矢量，在进行运算的时候需要使用矢量运算，进而算出总的标量矢量长度。

### 三轴加速度传感器

1. 半径测定

采用杜邦线将三轴加速度传感器引出，利用不同长度(10cm,20cm,30cm,40cm,50cm)进行数据分析，从而获得传感器数据到半径的预测模型。

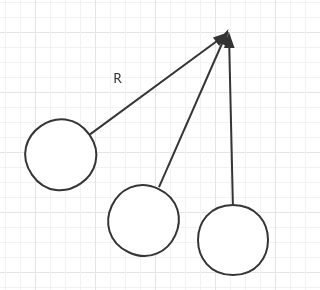


图3.4 三轴加速度半径测量图

对表格进行做图，得数据如下

图3.4 三轴加速度半径测量图

可以判定根据https://gss2.bdstatic.com/-fo3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D81/sign=e3add1e7dbc8a786ba2a470f6609108d/8ad4b31c8701a18b41646352942f07082838fe24.jpg计算得



表3.11 三轴加速度半径估算表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| l | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
| T | 0.8 | 1 | 1.1 | 1.2 | NAN |
| I(估计) | 0.15 | 0.24 | 0.3 | 0.357 | NAN |
| 绝对误差 | 0.05 | 0.04 | 0 | -0.04 | NAN |
| 相对误差 | 0.5 | 0.2 | 0 | 0.106 | NAN |

由于采用杜邦线作为半径，并非柔性材料，存在一定阻尼，在摆长度较大的时候较为明显，故仅仅采用较短部分进行计算，可见误差仍然较大。

1. 倾斜角测定

正反放置传感器，进行倾斜角判定。采用30°，60°，90°进行不同倾斜角测定

(此处进行三轴传感器进行测量时由于三轴传感器与平面平行困难，采用倾角传感器进行尝试)

|  |  |  |  |
| --- | --- | --- | --- |
|  | 15 | 45 | 90 |
| + | 0.596 | -1.146 | -3.027 |
| - | -0.183 | 2.201 | -2.935 |
| SUB | 0.779 | -3.347 | -0.092 |

其中，垂直和水平状态下，正反差值应该较为接近0，而45°存在较大差异。

## 实验问题

### 磁场强度

在进行磁场强度测量的时候，手持磁铁会有较大的抖动误差，且方向不一定准确，造成数据处理麻烦。

由于周围磁性介质的极化作用，磁场也会有较大的偏离。

### 测量传感器作圆周运动时的运动半径

由于不同的测量方式有不同的要求，在测量时需要采用不同的初始角度。如果采用矢量三角形进行计算，那需要进行尽可能的圆周运动。

如果采用周期进行近似计算，单摆周期有如下假设

1. 摆线由质量不计、不可伸缩的细线提供；
2. 摆球密度较大，而且球的半径比摆线的长度小得多，这样才可以将摆球看做[质点](https://baike.baidu.com/item/%E8%B4%A8%E7%82%B9" \t "/home/hover/Documents\\x/_blank)
3. 偏角小于10°的条件

实验过程中，由于传感器较轻，采用附上重物(磁铁)，但仍然密度较小。

采用杜邦线作为摆线，存在较大阻尼，造成实验误差会先减小后增大，在0.3m左右误差最小。

### 测量斜面坡度

由于三轴加速度传感器无法保持完全水平，所以在实验中采用坡度传感器测量的方式，进行正负的线性相减，获得减得结果与最终角度的线性关系。

## 应用场景

### 磁场传感器

1. 工业
   1. 电机

在无刷电动机中，用磁传感器来作转子磁极位置传感和定子电枢电流换向器，磁传感器中，霍尔器件、威根德器件、磁阻器件等都可以使用，但主要还是以霍尔传感器为主。另外磁传感器还可以对电机进行过载保护及转矩检测；

* 1. 大功率原件

以磁传感器为基础的各种电流传感器被用来监测控制和保护这些大功率器件

1. 信息记录磁信息记录装置

磁信息记录装置：磁带，磁盘，磁卡，磁墨水记录帐册，钞票的磁记录

1. 交通控制

目前，在加强行车支持道路系统(AHS)、智能运输系统(ITS)和道路交通信息系统(VICS)等的开发与建设中，高灵敏度、高速响应的微型DMR磁传感器大有用武之地。例如，用分辨率可达1nT的SI传感器，可构成ITS传感器（作高速路上的道路标志，测车轮角度，货车接近距离），汽车通过记录仪（测通行方向、速度、车身长度、车种识别）、停车场成批车辆传感器、加速度传感器（测车辆通过时路桥的振动等）等。

1. 汽车

磁传感器在汽车上的应用尤其普遍，例如包括汽车安全、汽车舒适性、汽车节能降耗等。它在汽车中主要被用于车速、倾角、角度、距离、接近、位置等参数检测以及导航、定位等方面的应用，比如车速测量、踏板位置、变速箱位置、电机旋转、助力扭矩测量、曲轴位置、倾角测量、电子导航、防抱死检测、泊车定位、安全气囊与太阳能板中的缺陷检测、座椅位置记忆、改善导航系统的航向分辨率。

1. 医疗

使用换向传感器在医疗设备之中用于电机控制，比如呼吸机、输液、胰岛素和肾脏透析机等方面的应用；电磁编码器霍尔传感器用于注射泵中检测流速以确定注射器是否为空和注射器是否堵塞的（应用于监测血液再造系统、自动血样分析系统）。

### 三轴加速度传感器

IMG_256在实际过程中，由于不同的方式和位置，在同一种运动规律下，三轴加速度传感器会出现不同的周期模式。

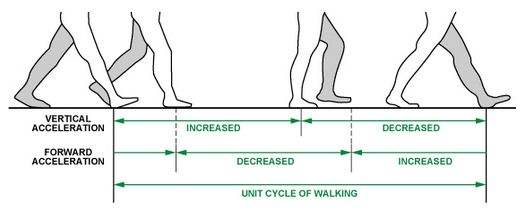


图3.5 三轴加速度传感计步图

1. 获取放置方向

因为用户在运动中可能用手平持设备，或者将设备置于口袋中。所以，设备的放置方向不定。为此，通过计算三个加速度的矢量长度，可以获得一条步行运动的正弦曲线轨迹。

1. 峰值检测

记录了上次矢量长度和运动方向，通过矢量长度的变化，可以判断目前加速度的方向，并和上一次保存的加速度方向进行比较。如果是相反的，即是刚过峰值状态，则进入计步逻辑进行计步，否则舍弃。通过对峰值的次数累加，可得到用户步行的步伐。

1. 去干扰

　　手持设备会有一些低幅度和快速的抽动状态，或是我们俗称的手抖，或者某个恶作剧用户想通过短时快速反复摇动设备来模拟人走路，这些干扰数据如果不剔除，会影响记步的准确值，对于这种干扰，我们可以通过给检测加上阀值和步频判断来过滤。

## 实验总结

本次实验采用python进行之后的数据清洗，生成相应的csv表格，利用excel进行数据统计。

实验过程中，由于传感器方面的限制，造成了一定的误差，但是在实验过程中尽可能的采取措施减少系统误差，并通过多次测量减少偶然误差。

实验熟悉了磁场和三轴传感器的相应使用和数据处理，并了解了相应传感器数据处理的相应算法。



**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 5月**

**计算机科学与技术学院**

# 传感器综合

## 实验目的

通过本次实验，熟悉智能车平台的硬件及软件设施，掌握代码烧写入智能车的方法；

仔细阅读几个样例代码，烧写进智能车，观察智能车展现出来的功能，建立代码—行为之间的联系；

自行修改代码，使智能车实现3种基本功能和至少2种进阶功能。

## 实验原理

### 红外传感器

红外探测器是红外系统的核心。它是利用红外辐射与物质相互作用所呈现出来的物理效应探测红外辐射的传感器，多数情况下是利用这种相互作用所呈现出的电学效应。此类探测器可分为光子探测器和热敏感探测器两大类型。实验中的红外传感器显然属于前者。

智能车的诸多功能中，循迹、避障、追踪、避开悬崖和定点停车等功能都依赖于多个红外传感器的协同工作。具体实现方法将在之后提到。

### 单轴倾角传感器

单轴倾角传感器理论基础是[牛顿第二定律](http://baike.baidu.com/edit/%E5%80%BE%E8%A7%92%E4%BC%A0%E6%84%9F%E5%99%A8/969912?dl=1)：根据基本的物理原理，在一个系统内部，速度是无法测量的，但却可以测量其加速度。如果初速度已知，就可以通过积分算出[线速度](http://baike.baidu.com/edit/%E5%80%BE%E8%A7%92%E4%BC%A0%E6%84%9F%E5%99%A8/969912?dl=1)，进而可以计算出[直线位移](http://baike.baidu.com/edit/%E5%80%BE%E8%A7%92%E4%BC%A0%E6%84%9F%E5%99%A8/969912?dl=1)，所以它其实是运用[惯性原理](http://baike.baidu.com/edit/%E5%80%BE%E8%A7%92%E4%BC%A0%E6%84%9F%E5%99%A8/969912?dl=1)的一种[加速度传感器](http://baike.baidu.com/edit/%E5%80%BE%E8%A7%92%E4%BC%A0%E6%84%9F%E5%99%A8/969912?dl=1)。当倾角传感器静止时也就是侧面和垂直方向没有加速度作用，那么作用在它上面的只有重力加速度。重力垂直轴与加速度传感器灵敏轴之间的夹角就是倾斜角了。

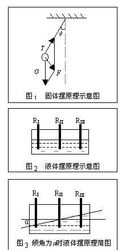


图4.1 单轴传感器图

将单轴倾角传感器固定在智能车上（角度正确），即可得到车辆行驶路径的坡度。

### 三轴加速度传感器

三轴加速度计传感器是基于加速度的基本原理去实现工作的，本质上属于惯性力传感器，使用的是牛顿第二运动定律：F=ma（物体所受到的合力等于其质量与其加速度的乘积，矢量运算）。

若将三轴加速度传感器固定在车轮侧面，那么在车轮转动的过程中，三轴加速度传感器的输出会呈周期性变化，这个时候我们只要知道串口读取数的时间间隔，就能算出轮胎转一圈的时间，而车轮的直径是可测的，由C=πd即可得到车轮周长。车速V=C/T。



图4.2 三轴及速度传感器

## 实验步骤

1. 熟悉将代码烧入智能车的方法，并将Qt文件APP3烧入智能车，测试其功能；
2. 通过功能测试发现其未完善之处并加以修改，使智能车完美实现其基本功能——循迹、避障、追踪、遥控、躲避悬崖和定点停车等等；
3. 通过外挂合适的传感器，测得车速和行车路径的坡度。
4. 使用中断，采用接口输出发动机转速信息

## 实验过程及结果

### 小车功能模式设置

利用volatile标志位进行通信，判断小车当前的状态位置。采用Time2进行定时器论询，同时由于volatile可以进行多线程操作。

代码4.1 小车状态位设置

**volatile** u8 MODE\_Select=1;                    //模式选择模式

**volatile** u8 MODE\_Wifi=0;                      //WiFi控制模式

**volatile** u8 MODE\_Remote=0;                    //遥控控制模式

**volatile** u8 MODE\_Trackline=0;             //循迹模式

**volatile** u8 MODE\_Avoid=0;                     //避障避险模式

**volatile** u8 MODE\_Stalk=0;                     //追踪模式

### 循迹

1. 功能

能够追寻黑色跑道进行定向行驶

1. 原理

小车前置五个朝向行驶面的传感器进行颜色检测，判断反射的红外线强度进行判断与黑色跑带的相对位置，当出现传感器非黑色的情况，则设置相应的转向。

当右边的传感器没有检测到反射的红外线（检测到黑线）而左侧的检测到反射的红外线（检测到白色区域）时就右拐，同理，当左边的传感器检测到黑线而右侧的没检测到就左拐，否则就直行。

### 避障

1. 功能描述：

小车在平面上能够自主躲避障碍物。

1. 原理

利用三个前向传感器进行检测，判定传感器前方是否有物体，如果发现物体距离小于某一个阈值，则进行后退且转向操作。

相关代码如下（3个以IR\_为前缀命名的信号来自3个前向红外传感器，KEY1和KEY5信号来自智能车的5个底向红外传感器的两端，因为两端的传感器就足以反应所有的应用场景）

代码4.2 避障模式设定

/\* Avoid 避障避险模式信息处理 \*/

**void** Avoid\_msgHandler(**void**)

{

**if**((IR\_LEFT==IR\_ON && IR\_MIDDLE==IR\_ON && IR\_RIGHT==IR\_ON) || (key(KEY1)==KEY\_ON && key(KEY5)==KEY\_ON)) //遇到障碍，前3个灯全亮

    {//底灯全亮表示到悬崖

        back\_off(200);//退后

        LED\_ALL\_OFF;

        led(LED5,LED\_ON);

        led(LED3,LED\_ON);

        led(LED1,LED\_ON);

    }

**else** **if**((IR\_LEFT==IR\_OFF && IR\_MIDDLE==IR\_ON && IR\_RIGHT==IR\_OFF) || (key(KEY1)==KEY\_ON && key(KEY5)==KEY\_ON)) //遇到较小障碍前灯仅中间一盏亮起

    {//底灯全亮表示遇到悬崖

        turn\_left(200); //前灯没有全亮说明还有行进余地，所以左拐避开

        LED\_ALL\_OFF;

        led(LED5,LED\_ON);

        led(LED4,LED\_ON);

    }

**else** **if**((IR\_LEFT==IR\_ON && IR\_RIGHT==IR\_OFF) || (key(KEY1)==KEY\_ON && key(KEY5)==KEY\_OFF)) //底灯低亮高不亮表示车辆右前方有悬崖

    { //不论中间的前灯亮与否，只要左亮右不亮就说明车辆右前方有障碍物

        turn\_left(200);//车辆左拐避开障碍物或悬崖

        LED\_ALL\_OFF;

        led(LED1,LED\_ON);

        led(LED2,LED\_ON);

    }

**else** **if**((IR\_LEFT==IR\_OFF && IR\_RIGHT==IR\_ON) || (key(KEY1)==KEY\_OFF && key(KEY5)==KEY\_ON))

    { //不论中间的前灯亮与否，只要右亮左不亮就说明车辆左前方有障碍物

        turn\_right(200);//车辆右拐避开障碍物或悬崖

        LED\_ALL\_OFF;

        led(LED5,LED\_ON);

        led(LED4,LED\_ON);

    }

**else** //否则车辆直行

    {

        go\_forward(200);

        LED\_ALL\_OFF;

        led(LED4,LED\_ON);

        led(LED3,LED\_ON);

        led(LED2,LED\_ON);

    }

}

追踪相关的问题，只需增加遇到较小追踪物体的情况即可。根据不同的前灯状态设置不同的转角。

代码4.3 追踪小物体设定

/\* Stalk 跟踪信息处理 \*/

**void** Stalk\_msgHandler(**void**)

{

**if**((IR\_LEFT==IR\_ON && IR\_MIDDLE==IR\_ON && IR\_RIGHT==IR\_ON))

    { //3个前灯都亮表示前方有追踪物体

        go\_forward(200); //直走

        LED\_ALL\_OFF;

        led(LED5,LED\_ON);

        led(LED3,LED\_ON);

        led(LED1,LED\_ON);

    }

**else** **if**((IR\_LEFT==IR\_OFF && IR\_MIDDLE==IR\_ON && IR\_RIGHT==IR\_OFF))

    { //中间1个亮表示前方有较小追踪物体

        go\_forward(200); //直走

        LED\_ALL\_OFF;

        led(LED5,LED\_ON);

        led(LED3,LED\_ON);

        led(LED1,LED\_ON);

    }

**else** **if**((IR\_LEFT==IR\_ON && IR\_RIGHT==IR\_OFF)) //左亮右不亮说明右前方有追踪物

    {

        turn\_right(200);

        LED\_ALL\_OFF;

        led(LED1,LED\_ON);

        led(LED2,LED\_ON);

    }

**else** **if**((IR\_LEFT==IR\_OFF && IR\_RIGHT==IR\_ON) )//右亮左不亮说明左前方有追踪物

    {

        turn\_left(200);

        LED\_ALL\_OFF;

        led(LED5,LED\_ON);

        led(LED4,LED\_ON);

    }

**else** //否则停止、灭灯

    {

        brake();

        LED\_ALL\_OFF;

        led(LED4,LED\_ON);

        led(LED3,LED\_ON);

        led(LED2,LED\_ON);

    }

}

### 避开悬崖

1. 功能

将小车放在带有“悬崖”的平台上，当小车检测到悬崖时，就后退一段距离再转向（避开悬崖），然后再继续前进。

1. 原理

更改循迹模式相应代码当5个传感器都没有接收到反射的红外线时(悬崖将红外线进行了不规则的反射)，小车就认为前方是悬崖，进而进行相应的后续操作。

### 标记停车

1. 功能

到达标记处进行停车操作

1. 原理

更改循迹模式相应代码，当5个传感器同时检测到前方为黑线时，就停车，否则按照原有模式运行。

### 行走倾角测量-外挂传感器(倾角传感器)

将单轴倾角传感器以如所示方式固定在智能车上，即可在行车过程中通过串口读取车辆当前行驶路径的坡度情况。

单轴倾角传感器具有方向性。实验中要注意单轴倾角传感器固定的方向，总方向出错将影响到测量结果，传感器与车是否水平则关系到测量结果的准确性。

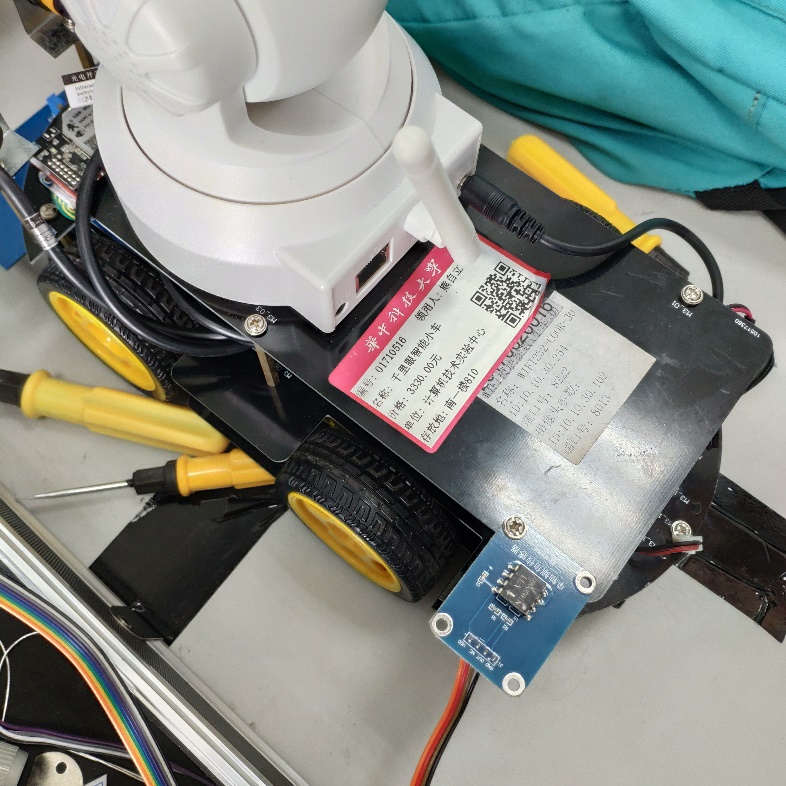


图4. 1 单轴倾角传感器测坡度



图4. 2 单轴倾角传感器测试结果

### 行走距离测量-外挂传感器(三轴加速度)

1. 功能

计算小车的行走距离

1. 原理

此处采用了两种方式，可使用外置三轴传感器进行周期测量，同时使用轮径计算最终行驶距离。

1. 三轴倾角传感器

如图所示，将三周加速度传感器测方向固定在车轮上，在行车过程中，从串口读取数据。

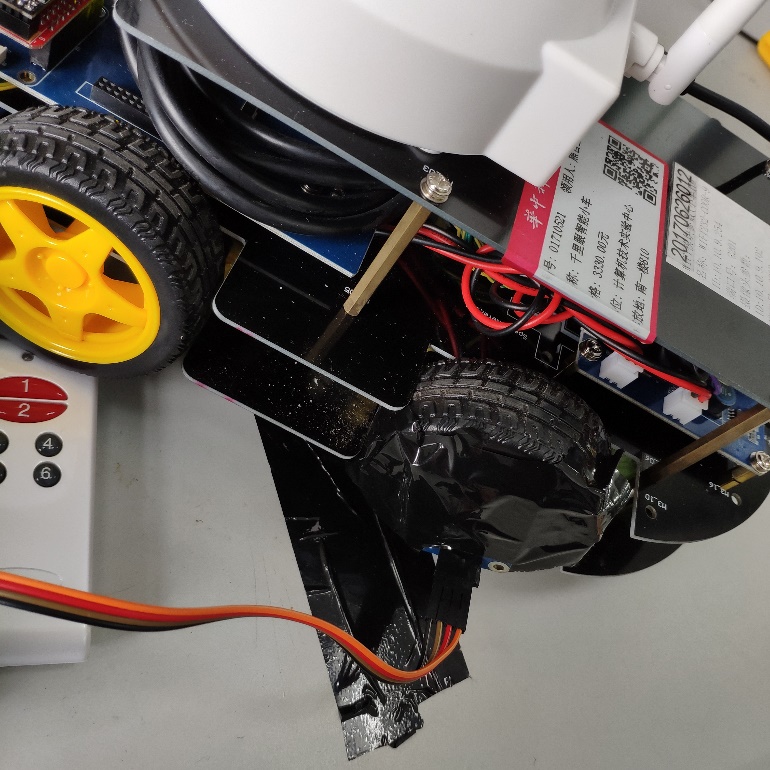


图4. 3 三轴加速度传感器测车速



图4. 4三周加速度传感器测试结果

可以看出，以Y轴为例，从第三行的-925开始，每隔三行就会有一个较大的负值，所以不难理解为串口每输出三个数据，车轮大约转1圈。

已知串口输出数据的速率a行/秒，即车轮转速为 rad/秒，又已知车轮的周长C米（车轮转一圈走过的距离），不难由得到车速，单位是米/秒。

### 行走距离测量-车载参数

估算车行走距离

在调用驱动小车前进、左转或右转的函数时都会传入一个参数，默认为200。而这个参数会在驱动小车的函数会与Tim内的参数进行比较，从而在更底层来决定小车轮子转动的方向。

声明一个全局变量，在小车前进的函数中对传入的参数进行累加，无论小车在哪个模式下前进都会加上传入的参数，而前进的距离与传入的参数呈线性关系，只有经过测量算出其比值即可根据累加的参数值求得所走距离。然后可再通过串口将该值输出出来。

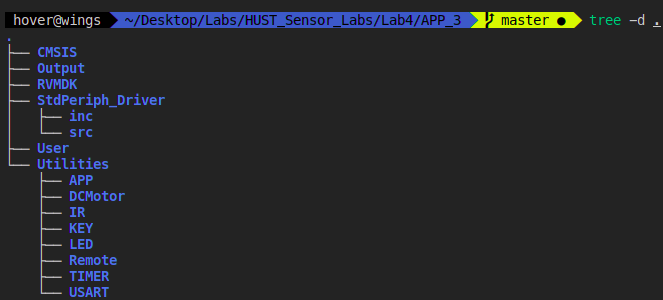


图4.3 目录结构

其中DCMotor为发动机相关函数，利用原试验箱STM32代码部分printf重定向进行中断输出。

代码4.4 printf中断输出

/\*

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

\*   函 数 名: ISRprintf

\*   功能说明: 板载 中断printf 功能函数

\*   形    参: string 字符串，num 字符个数（当为字符串时为 0）

\*   返 回 值: 无

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

\*/

**void** ISRprintf(**char**\* string,uint8\_t num)

{

    uint8\_t count = 0;

**if**(num)

**while**(count < num)

        {

**while**((USART1->SR & USART\_FLAG\_TXE)==0) {};   //检测是否发送完毕

            USART\_ClearFlag(USART1, USART\_FLAG\_TXE);

//          while((USART1->SR & USART\_SR\_TC)==0) {};  //检测是否发送完毕

            USART1->DR = (\*string++ & (uint16\_t)0x01FF);//发送数据

//          USART\_SendData(USART1,(uint16\_t)\*string++);

            count++;

        }

**else**

**while**(\*string)

        {

**while**((USART1->SR & USART\_FLAG\_TXE)==0) {};   //检测是否发送完毕

            USART\_ClearFlag(USART1, USART\_FLAG\_TXE);

//          while((USART1->SR & USART\_SR\_TC)==0) {};  //检测是否发送完毕

            USART1->DR = (\*string++ & (uint16\_t)0x01FF);//发送数据

//          USART\_SendData(USART1,(uint16\_t)\*string++);

        }

}



图4.4 目录结构

进行测量的过程中，由于关系线性，先选定一个特定的距离(1m)进行标定量的测量，从而计算相应的标定量值，然后中断时进行标定量转换。

串口不断输出标定量，在存在输入中断的时候，进行标定量转换，计算得到最终的距离。

## 实验问题

### 小车Download代码失败

由于GPIO通信转换的时候可能存在不同的线序，在使用转换器和不使用转换的时候需要进行标志位操作，此时进行开关电源即可。

### 串口中断通信

1. 由于在printf过程中，踩了不少坑，线存在有USB旁路输出和无旁路输出两种，仅仅使用旁路输出的USB线才可以进行中断通信，而普通线仅仅能够Download代码。
2. 中断过程中，需要主动发送消息

## 应用场景

智能小车为综合实验平台，可以理解为较为大型的小车的模拟或者仅仅为单体小车，智能小车可以分为三部分——传感器部分、控制器部分、执行器部分。

其中，利用传感器进行负反馈维持小车的正常状态为常规的工业控制思路，对于不同朝向的传感器进行不同程度的判定，同时利用串口通信进行输出。

实际中，可以利用小车进行灾害现场的检测，特殊地形的探测，物流的自动化等实际应用。

而对于更大型车的模拟，小车可以检测传感器模型的控制精度，进行控制模型的矫正和功能测试。

## 实验总结

实验的小车为封装较好的小车，主要由三部分构成：传感器部分、控制器部分、执行器部分。

其中，传感器进行数据采集，控制器进行状态判断，执行器进行决策执行，为典型的工业控制流程。

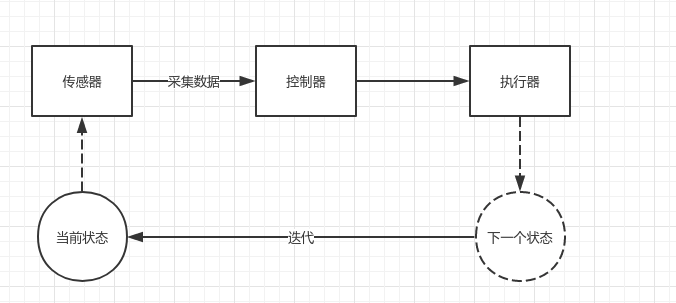


图4.5 小车工作结构

其实可以注意到，典型的工业流程控制与强化学习的工作流程类似，其中控制器部分此处由人工设定规则进行学习，此处可以直接利用强化学习进行控制器规则学习，可以适应不同的工作环境和工作流程，让小车更智能。这也是目前自动驾驶技术的发展思路之一。

实验过程中，对于整体代码架构进行了一定程度的理解，在有了基本工作模式上，只需要分清各部分的功能和模块通信即可。

实验最复杂的部分莫过于通信，由于线序等底层不熟悉，导致自制线不成功，阻碍了实验进度，在有了线之后能够进行实时检测能够较快完成。

串口通信部分，利用中断机制，进行printf绑定，其中有不同的实现思路，一种是进行printf输出重定向，将std重定向到GPIO，另一种是直接封装底层函数进行中断输出。

传感器应用部分，采用了车载传感器和外挂传感器的测试，并对精度进行了一定程度的计算，由于外挂传感器的外挂方式和传感器本身的原因引入了误差，导致精度远不如车载传感器。

实验过程中锻炼了软件和硬件联调的能力，同时对串口通信机制有了更深的理解，对于复杂项目的理解能力有了锻炼，收获颇丰。

最后，感谢相关参阅文献的提供者和在实验过程中提供帮助的老师，助教和同学。

## 参考文献

1. UP-WebCar千里眼智能车光盘V1.0@20160815实验指导书
2. STM32 Pinrtf

<https://blog.csdn.net/qq_26904271/article/details/80113740>