

**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](#_Toc821764458)

[1.1 实验目的 1](#_Toc1550538395)

[1.2 实验原理 1](#_Toc1906447234)

[1.3 实验步骤 17](#_Toc240858680)

[1.4 实验过程与结果分析 18](#_Toc1784274866)

[1.5 实验问题分析 25](#_Toc1281522974)

[1.6 实验总结 26](#_Toc1573436092)

[1.7 参考文献 26](#_Toc1659947385)

# 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

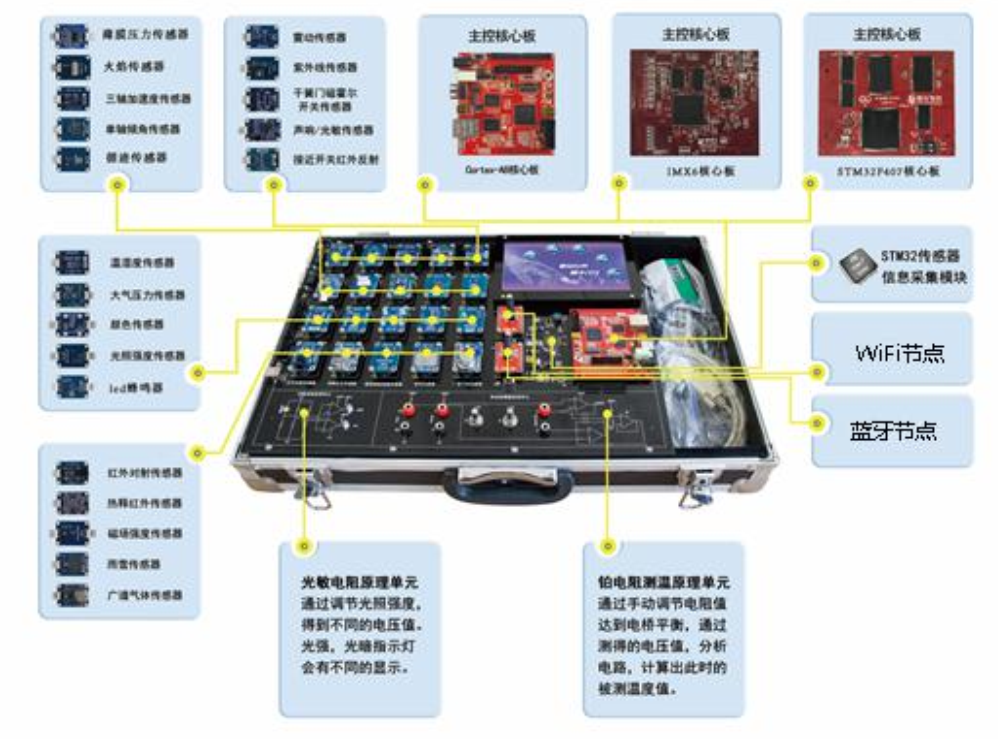


图1.1 试验箱组成图

### 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

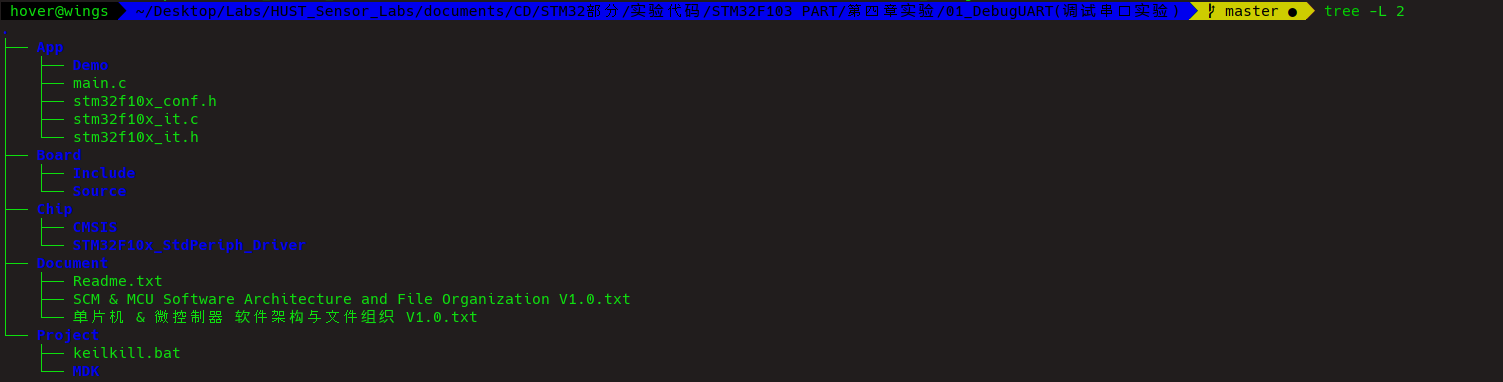


图1.2 实验代码架构图

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

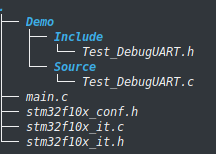


图1.3 APP代码架构图

.

├── 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。

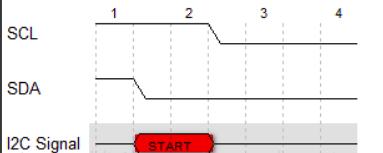


图1.4 START对应信号图

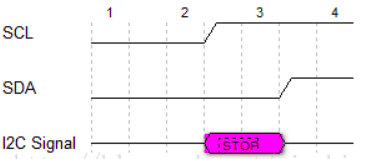


图1.5 STOP对应信号图

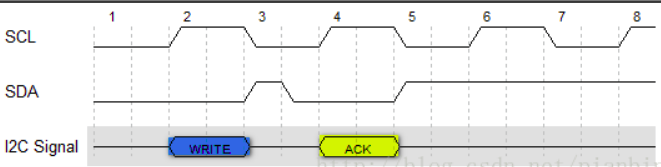


图1.6 WRITE和ACK对应信号图

##### 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

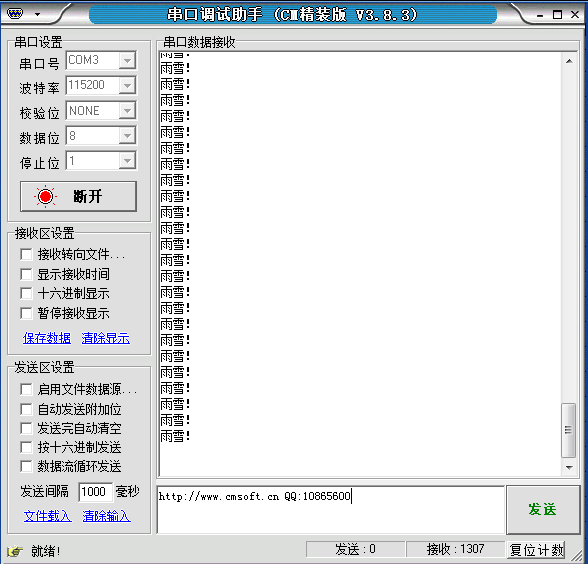


图1.7 雨雪传感器测试图

##### 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。

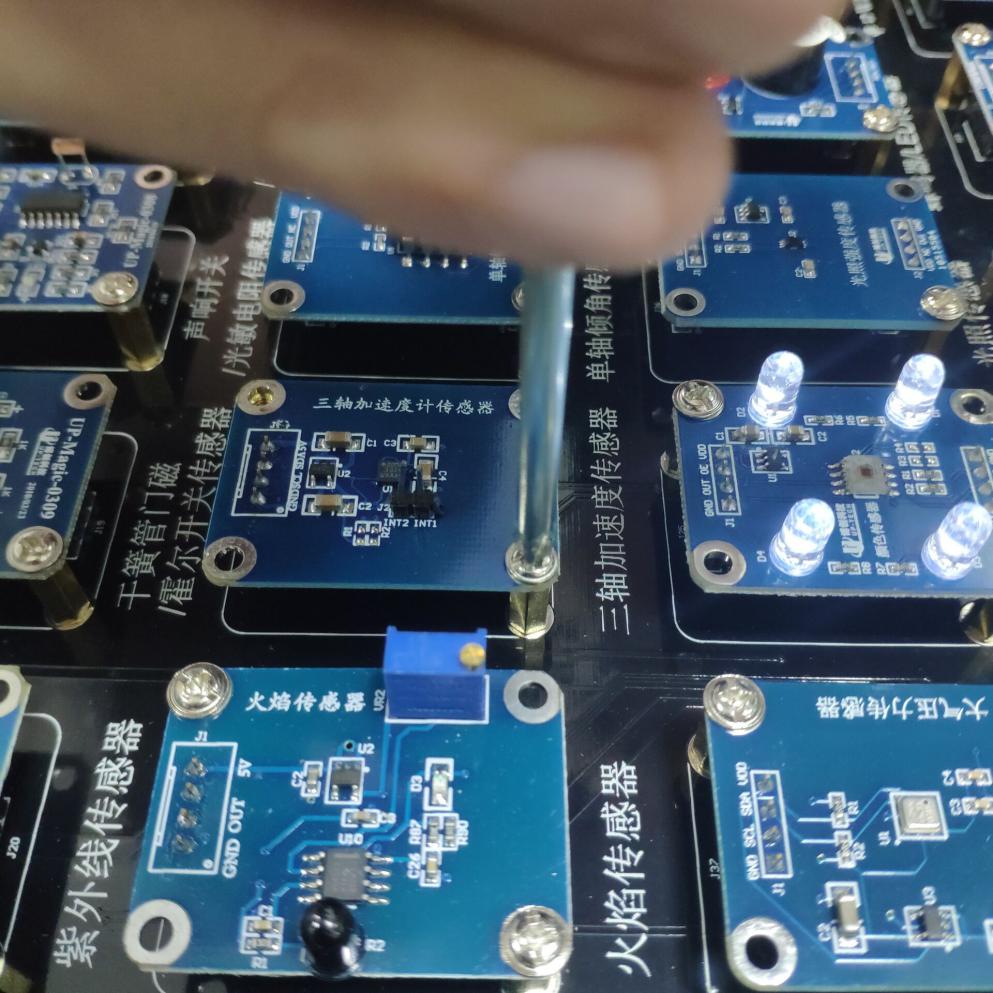


图1.10 拆卸三轴加速度传感器



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

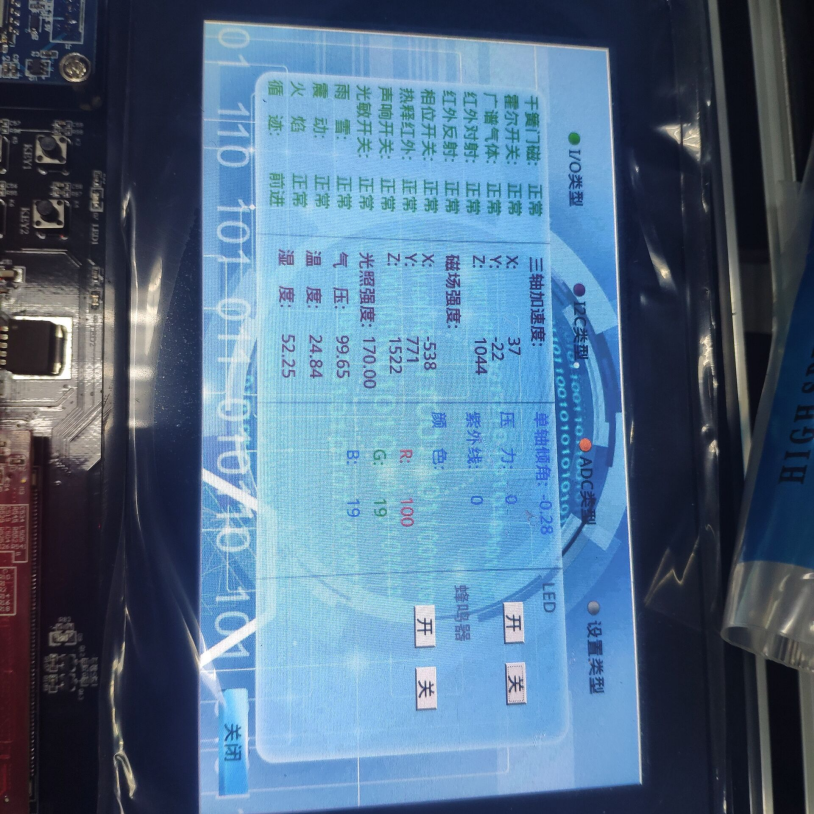


图1.13 传感器板载测试

## 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