

学校代码：10022



北京林业大学
Beijing Forestry University

《电子信息实践与创新》实验报告

作者姓名 李雨梦

同组人姓名 刘昕盈

指导教师 张立

学号 231101128

班级 电子 231

成绩

日期 2025 年 7 月 20 日

目录

1.实验一：基于 STM32G474RE 开发板的测试实验.....	3
1.1 实验题目	3
1.2 实验目的	3
1.3 实验要求	3
1.4 实验原理	3
1.5 实验内容	3
1.6 程序代码	4
1.7 实验结果及讨论	13
1.7.1 实验结果:	13
1.7.2 讨论:	13
2 实验二：基于 STM32G474 及霍尔元件的磁场测量.....	14
2.1 实验目的:	14
2.2 实验要求:	14
2.3 实验器材:	14
2.4 基本原理:	14
2.5 软硬件环境及设计:	16
2.6 测试结果:	36
2.7 总结与讨论:	37
3 实验三 激光测距与雷达成像.....	37
3.1 实验目的:	37
3.2 程序代码	37
3.3 讨论	44
3.4 结果展示	45
4 实验四 MPU9250 使用及姿态可视化成像.....	45
4.1 实验目的:	45
4.2 程序代码	45
4.3 成果展示	56

1.实验一：基于 STM32G474RE 开发板的测试实验

1.1 实验题目

基于 STM32G474RE 开发板的按键控制与 ADC 采样串口通信实验。

1.2 实验目的

1. 掌握 STM32G474RE 开发板的管脚配置与基本外设使用方法。
2. 学会利用 STM32CubeMX 进行项目配置和代码生成。
3. 理解按键长按与短按的控制逻辑以及 PWM 信号在 LED 亮度控制中的应用。
4. 掌握 ADC 采样与串口通信的基本原理和编程实现。

1.3 实验要求

1. 按照给定的管脚配置完成硬件连接。

实现按键短按控制 LED 开关，长按实现 LED 的 PWM 渐亮渐灭功能。

采样外部电压值并通过串口发送至上位机，确保数据的准确性和完整性。

编写代码时遵循良好的编程规范，注释清晰，便于理解和维护。

1.4 实验原理

按键控制原理：通过检测按键的按下时间来区分短按和长按。短按时，直接改变 LED 的开关状态；长按时，利用定时器产生 PWM 信号，改变 LED 的亮度，实现渐亮渐灭的效果。

ADC 采样原理：STM32G474RE 的 ADC 模块将输入的模拟电压信号转换为数字量，通过读取转换结果寄存器获取采样值。

串口通信原理：利用 USART2 模块，按照约定的波特率和数据格式，将采样数据发送至上位机，实现数据的传输和显示。

1.5 实验内容

硬件连接：

将外部信号源连接到 PA0 引脚。（设置输出 0~3V，频率 10Hz）

将 PA2 和 PA3 分别连接到 PC 的串口发送和接收引脚。（板子已连接）

将 PA5 引脚连接到 LED 的一端，LED 的另一端接地。（板子已连接）

将 PC13 引脚连接到按键的一端，按键的另一端接地，并在 PC13 引脚和 3.3V 之间连接一个上拉电阻。（板子已连接）

将 24MHz 的外部晶振连接到 STM32G474RE 的相应引脚。（板子已连接）

软件设计：

使用 STM32CubeMX 配置管脚功能，生成初始化代码。

STM32CubeMX 管脚配置：

PA0-->ADC1_IN1;

PA2-->USART2_TX;

PA3-->USART2_RX;

PA5-->TIM2_CH1 (输出高电平点亮 LED，配置为：内部时钟-->PWM Generation CH1) ;

PC13-->B1 按键（输入下拉电阻）

外部晶振 24MHz；

编写按键检测函数，判断按键的短按和长按操作。（参考 PP T4.4 内容及其他）

编写 ADC 采样函数，读取外部电压值。（参考 PP T4.8 内容及其他）

编写 PWM 输出函数，控制 LED 的亮度变化。（参考 PP T4.5 内容及其他）

编写串口通信函数，实现数据的发送。（参考 PP T4.6 内容及其他）

在主函数及中断中实现各功能的调用和逻辑控制。

1.6 程序代码

```
/* USER CODE BEGIN Header */
/**


*****



***



* @file          : main.c
* @brief         : Main program body



*****



***



* @attention
*
* Copyright (c) 2025 STMicroelectronics.
* All rights reserved.
*
* This software is licensed under terms that can be found in the LICENSE file
* in the root directory of this software component.
* If no LICENSE file comes with this software, it is provided AS-IS.
*



*****



***



*/
/* USER CODE END Header */
/* Includes
-----*/
#include "main.h"
#include "adc.h"
#include "tim.h"
#include "usart.h"
#include "gpio.h"

/* Private includes
-----*/
/* USER CODE BEGIN Includes */
```

```
#include "headfile.h"
/* USER CODE END Includes */

/* Private typedef
-----*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define
-----*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro
-----*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables
-----*/
/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes
-----*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code
-----*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/***
 * @brief The application entry point.
 * @retval int
 */
```

```

*/
int main(void)
{
    /* USER CODE BEGIN 1 */

    /* USER CODE END 1 */

    /* MCU
    Configuration-----*/
    /* Reset of all peripherals, Initializes the Flash interface and the Systick.
*/
    HAL_Init();

    /* USER CODE BEGIN Init */

    /* USER CODE END Init */

    /* Configure the system clock */
    SystemClock_Config();

    /* USER CODE BEGIN SysInit */

    /* USER CODE END SysInit */

    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_ADC1_Init();
    MX_USART2_UART_Init();
    MX_TIM2_Init();
    MX_TIM4_Init();
    MX_TIM3_Init();
    /* USER CODE BEGIN 2 */
    HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
    HAL_TIM_Base_Start_IT(&htim4);
    HAL_UART_Receive_IT(&huart2, &rec_data, 1);
    HAL_TIM_Base_Start(&htim3);
    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {

```

```

        key_scan();
        uart_data_rec();
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
}

/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

    /** Configure the main internal regulator output voltage
    */
    HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1);

    /**
     * Initializes the RCC Oscillators according to the specified parameters
     * in the RCC_OscInitTypeDef structure.
     */
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
    RCC_OscInitStruct.HSEState = RCC_HSE_ON;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
    RCC_OscInitStruct.PLL.PLLM = RCC_PLLM_DIV2;
    RCC_OscInitStruct.PLL.PLLN = 12;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
    RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
    RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
    if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
    {
        Error_Handler();
    }

    /**
     * Initializes the CPU, AHB and APB buses clocks
     */
    RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                                |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
    RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;

```

```

RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
{
    Error_Handler();
}
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
 * @brief This function is executed in case of error occurrence.
 * @retval None
 */
void Error_Handler(void)
{
    /* USER CODE BEGIN Error_Handler_Debug */
    /* User can add his own implementation to report the HAL error return state
 */
    __disable_irq();
    while (1)
    {
    }
    /* USER CODE END Error_Handler_Debug */
}

#ifndef USE_FULL_ASSERT
/**
 * @brief Reports the name of the source file and the source line number
 *        where the assert_param error has occurred.
 * @param file: pointer to the source file name
 * @param line: assert_param error line source number
 * @retval None
 */
void assert_failed(uint8_t *file, uint32_t line)
{
    /* USER CODE BEGIN 6 */
    /* User can add his own implementation to report the file name and line number,
       ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line)
 */

```

```

/* USER CODE END 6 */
}

#endif /* USE_FULL_ASSERT */

#include "headfile.h"

uint16_t pwmVal=0;
uint8_t B1_press,B1_press_duration;
char send_buff[20];
uint8_t rec_buff[20];
uint8_t rec_data,rec_flag,count;
uint32_t adc_value;

const uint32_t tD[]=
{
    3178, //t=0
    3139, 3099, 3059, 3017, 2975, 2932, 2888, 2844, 2799, 2754, //t=1~10
    2708, 2662, 2615, 2569, 2521, 2474, 2427, 2379, 2331, 2284, //t=11~20
    2236, 2189, 2141, 2094, 2048, 2001, 1955, 1909, 1864, 1819, //t=21~30
    1775, 1731, 1688, 1645, 1603, 1562, 1522, 1482, 1442, 1404 //t=31~40
};

```

```

void pwm_control_led()
{
    while (pwmVal< 500)
    {
        pwmVal++;
        __HAL_TIM_SetCompare(&htim2, TIM_CHANNEL_1, pwmVal);
        HAL_Delay(1);
    }
    while (pwmVal)
    {
        pwmVal--;
        __HAL_TIM_SetCompare(&htim2, TIM_CHANNEL_1, pwmVal);
        HAL_Delay(1);
    }
}

```

```

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if(htim->Instance==TIM4)
    {
        if(B1_press)
            B1_press_duration++;
    }
}

```

```

        }
    }

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    if(huart->Instance==USART2)
    {
        TIM3->CNT=0;
        rec_flag=1;
        rec_buff[count]=rec_data;
        count++;
        HAL_UART_Receive_IT(huart, &rec_data, 1);
    }
}

double get_volt()
{
    HAL_ADC_Start(&hadc1);
    adc_value=HAL_ADC_GetValue(&hadc1);
    return 3.3*adc_value/4096;
}

void uart_data_rec()
{
    if(rec_flag)
    {
        if(TIM3->CNT>15)
        {
            if(count==3)
            {
                if(rec_buff[0]=='a' && rec_buff[1]=='d' && rec_buff[2]=='c')
                {
                    sprintf(send_buff, "v=%f\r\n", get_volt());

                    HAL_UART_Transmit(&huart2, (uint8_t*)send_buff, sizeof(send_buff), 50);
                }
                else
                {
                    sprintf(send_buff, "error!\r\n");
                    HAL_UART_Transmit(&huart2, (uint8_t*)send_buff, sizeof(send_buff), 50);
                }
            }
            else
        }
    }
}

```

```

    {
        sprintf(send_buff, "error!\r\n");

        HAL_UART_Transmit(&huart2, (uint8_t*)send_buff, sizeof(send_buff), 50);
    }

    rec_flag=0;
    for(int i=0;i<count;i++)
        rec_buff[i]=0;
    count=0;
}
}

#endif
#define _FUN_H

void pwm_control_led(void);
double get_vol(void);
void uart_data_rec(void);

#endif
#ifndef _HEADFILE_H
#define _HEADFILE_H

#include "stm32g4xx.h" // Device header
#include "string.h"
#include "stdint.h"
#include "stdio.h"

#include "main.h"
#include "adc.h"
#include "tim.h"
#include "usart.h"
#include "gpio.h"

#include "fun.h"
#include "key.h"

extern uint8_t B1_press,B1_press_duration;
extern uint8_t long_or_short;
extern uint8_t rec_data;

#endif
#include "headfile.h"

```

```

uint8_t B1_state;
uint8_t B1_last_state;
uint8_t long_or_short;

void key_scan()
{
    B1_state=HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13);
    if(B1_state==1 && B1_last_state==0)
    {
        __HAL_TIM_SetCompare(&htim2, TIM_CHANNEL_1, 250);
        B1_press_duration=0;
        B1_press=1;
    }
    else if(B1_state==0 && B1_last_state==1)
    {
        if(B1_press_duration>=100)
        {
            B1_press=0;
            B1_press_duration=0;
            __HAL_TIM_SetCompare(&htim2, TIM_CHANNEL_1, 0);
            HAL_Delay(500);

            pwm_control_led();
        }
    }
    else
    {
        __HAL_TIM_SetCompare(&htim2, TIM_CHANNEL_1, 0);
    }
}
B1_last_state=B1_state;
}

#ifndef _KEY_H
#define _KEY_H

void key_scan(void);

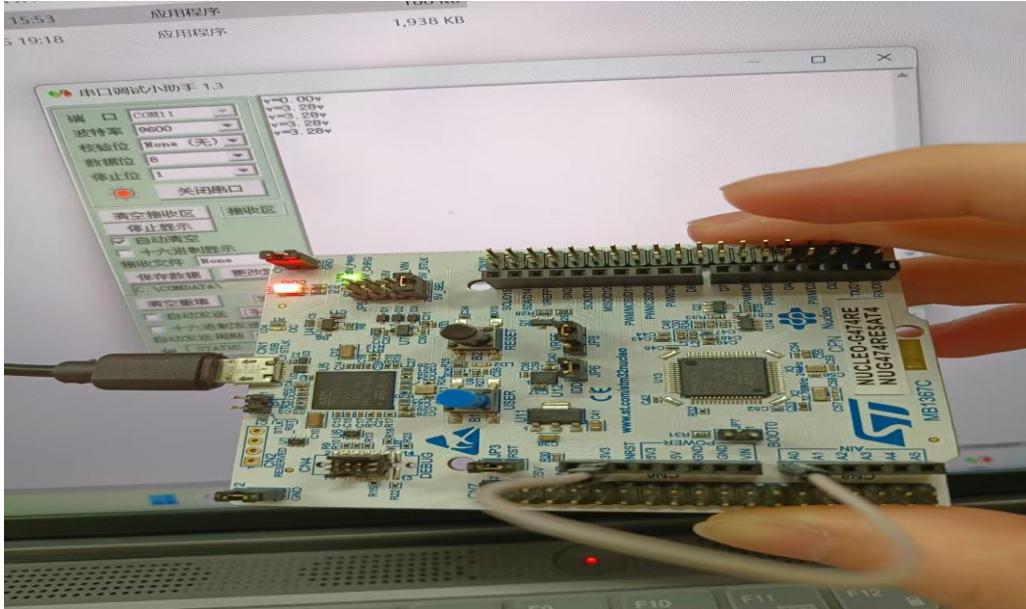
#endif

```

1.7 实验结果及讨论

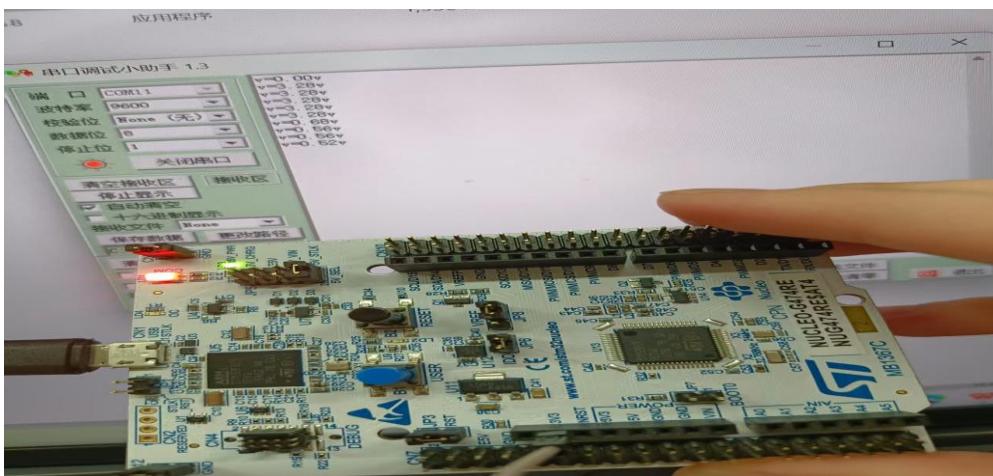
1.7.1 实验结果：

1. 按键短按时，LED 能够正常点亮和熄灭。



图表 1 按键短按

2. 按键长按时，LED 实现 PWM 渐亮渐灭的效果，周期为 1 秒。



图表 2 按键长按

串口调试助手中能够接收到采样到的外部电压值，数据准确无误。

1.7.2 讨论：

分析实验中可能出现的问题，如按键抖动、ADC 采样精度、PWM 信号的稳定性等，并探讨解决方案。

1. 讨论如何优化代码结构，提高程序的可读性和可维护性。

可以首先采用模块化设计，将不同功能（如按键处理、PWM 控制、UART 通信）分离到独立文件，通过清晰的接口函数交互，减少全局变量的使用，改用结构体封装相关数据；其次遵循一致的命名规范（如驼峰命名法），添加详细的函数注释和关键逻辑说明；

最后增强错误处理机制，检查 HAL 函数返回值，避免直接操作硬件寄存器，改用 HAL 库标准 API，并合理使用静态函数限制作用域，这样既能降低模块间耦合度，也便于后续功能扩展和维护。

2. 探索如何进一步扩展实验功能，如增加多个按键控制不同的 LED，或者实现更多的串口通信交互功能。

在现有实验基础上，可以通过以下方式扩展功能：首先，增加多个按键控制不同 LED 时，可以在 key.c 中扩展按键扫描逻辑，为每个按键定义独立的状态变量和回调函数，通过 GPIO 外部中断或定时扫描方式检测按键动作，并关联不同的 PWM 通道控制多路 LED，实现诸如呼吸灯、闪烁模式切换等效果；其次，增强串口通信功能可设计简单的命令行交互协议，在 uart_data_rec() 函数中实现命令解析器，支持“led on/off”、“pwm 50”等指令，添加命令帮助提示和参数校验，同时采用环形缓冲区管理接收数据，结合状态机实现可靠的数据帧解析，还可增加 ADC 数据定时上报、系统状态查询等实用功能，使整个系统具备更丰富的交互能力。这些扩展既保持了原有架构，又提升了系统的实用性和可玩性。

3. 总结所用到的 HAL 函数。

本实验主要使用了 STM32 HAL 库的关键函数实现核心功能：GPIO 方面通过 `HAL_GPIO_ReadPin()` 检测按键状态；定时器方面使用 `__HAL_TIM_SetCompare()` 调节 PWM 占空比控制 LED 亮度；UART 通信采用 `HAL_UART_Receive_IT()` 和 `HAL_UART_Transmit()` 实现中断式数据收发；ADC 采样通过 `HAL_ADC_Start()` 和 `HAL_ADC_GetValue()` 获取电压值；系统控制方面则运用 `HAL_Delay()` 进行延时处理。这些 HAL 函数相互配合，在底层硬件驱动的基础上构建了按键检测、LED 控制、数据通信和模拟量采集等完整功能，体现了 STM32 HAL 库模块化、标准化的设计特点，为后续功能扩展提供了良好的基础框架。

2 实验二：基于 STM32G474 及霍尔元件的磁场测量

2.1 实验目的：

学习 Arduino 环境下安装 STM32 库、液晶 SS1306 库及编程，研究磁场测量传感器原理，用 2 种编程方法分别设计磁场测量装置。

2.2 实验要求：

设计的磁场检测装置测试的电压值和相应的磁场结果显示在液晶上，注意磁场有正负，磁铁反过来测量为负值，单位为 (mT)；同时，将测得的磁场值通过串口发送至上位机。

2.3 实验器材：

1、STM32G474 开发板；2、圆形小磁铁；3、线性霍尔传感器 AH3503；4、0.96” OLED 的 I2C 接口 SS1306 液晶；5、面包板；6、导线若干；7、下载线。

2.4 基本原理：

如图所示的霍尔传感器，根据所给线性霍尔传感器 AH3503 元件说明，利用霍尔原理，所施加磁场和霍尔电压的线性关系（根据表格相关数据推出公式），通过测量霍尔电压



间接测量出磁场。

AH3503 线性霍尔电路

AH3503 线性霍尔电路由电压调整器，霍尔电压发生器，线性放大器和射极跟随器组成，其输入是磁感应强度，输出是和输入量成正比的电压。

产品特点

- 线性好
- 功耗低
- 灵敏度高
- 输出电阻小
- 温度稳定性好
- 寿命长

典型应用

- 磁场测量
- 非接触测距
- 速度检测
- 黑色金属检测
- 缺口检测
- 无刷直流电机
- 远传仪表

极限参数

参 数	符 号	型 号 和 量 值	单 位
电源电压	V _{cc}	6	V
磁感应强度	B	不限	mT
工作环境温度	T _A	-20~+85	℃
高温贮存温度	T _S	150	℃

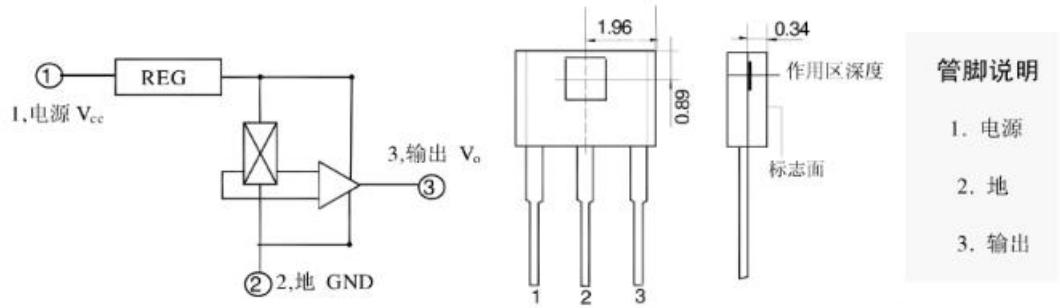
电特性

T_A=25 ℃

1mT=10Gs

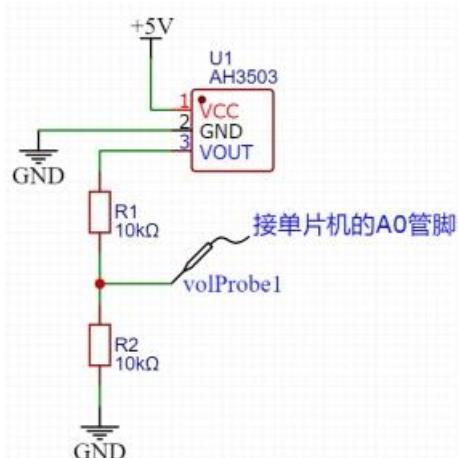
参 数	符 号	测 试 条 件	型 号 与 量 值			单 位
			最 小	典 型	最 大	
电源电压	V _{cc}		4.5	-	6	V
电源电流	I _{cc}		-	9	14	mA
线性范围			-90	-	+90	mT
静态输出电压	V _{out}	B=0	2.25	2.5	2.75	V
灵敏度	S	B=±90mT	7..5	13..5	17..5	mV/mT
输出电阻	R _O		-	50	-	Ω

注:输出电压应用输入阻抗大于 10KΩ 的电压表来测量; 磁感应强度应在器件最灵敏的区域(见外形图)测量.



2.5 软硬件环境及设计:

管脚连接及配置: 1--+5V; 2--GND; 3--A0.



AH503 在磁铁贴近测量面进行正向测量时，3 脚输出约为 3.30 V，对应磁场为 500 mT；磁铁翻过来贴近测量面进行反向测量时，3 脚输出约为 0.77 V，对应磁场为 -470 mT；由于 STM32G474 的 ADC 最高测量电压是 3.3V，故将输出电压串联进行分压后再送入单片机进行测量。根据以上推导出磁场 B 与 Vad 的关系，从而测量 B。如取典型值 13mV/5mT 作为灵敏度，取上电时输出电压 V0 作为静态输出电压，则测得磁场 B 与测得电压 V 有如下线性关系：

$$B = \frac{V - V_0}{0.013} \times 5 \quad (\text{仅供参考, 自行推导})$$

1) Arduino 代码: (现场完成)

```
#include <Arduino.h>
#include <Wire.h>
#include <U8g2lib.h>

// OLED 屏幕 (I2C 接口, SSD1306)
U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /* reset= */ U8X8_PIN_NONE);

// 霍尔传感器接到 PA0 (ADC)
#define HALL_SENSOR_PIN PA0
```

```

// ADC 相关参数
#define ADC_RESOLUTION 4095 // 12-bit ADC
#define VCC 3.3 // STM32 供电电压 3.3V
#define VREF 3.3 // ADC 参考电压
#define HALL_ZERO_V 2.5 // 霍尔传感器无磁场时的电压 (2.5V)
#define HALL_SENSITIVITY 0.0075 // AH3503 灵敏度 (7.5mV/mT)

void setup() {
    Serial.begin(115200);

    // 初始化 OLED
    u8g2.begin();

    // 初始化 ADC 引脚
    pinMode(HALL_SENSOR_PIN, INPUT_ANALOG);
}

void loop() {
    // 读取 ADC 值
    int adcValue = analogRead(HALL_SENSOR_PIN);

    // 转换成电压
    float voltage = (adcValue / (float)ADC_RESOLUTION) * VREF;

    // 计算磁场强度 B (单位: mT)
    float magneticField = (voltage - HALL_ZERO_V) / HALL_SENSITIVITY;

    // OLED 显示数据
    u8g2.clearBuffer();
    u8g2.setFont(u8g2_font_ncenB08_tr);
    u8g2.setCursor(10, 20);
    u8g2.print("ADC: ");
    u8g2.print(adcValue);
    u8g2.setCursor(10, 40);
    u8g2.print("Voltage: ");
    u8g2.print(voltage, 2);
    u8g2.setCursor(10, 60);
    u8g2.print("B: ");
    u8g2.print(magneticField, 2);
    u8g2.print(" mT");
    u8g2.sendBuffer();

    // 串口输出数据
    Serial.print("ADC: ");
}

```

```

Serial.print(adcValue);
Serial.print(" -> Voltage: ");
Serial.print(voltage, 3);
Serial.print("V -> B: ");
Serial.print(magneticField, 3);
Serial.println(" mT");

delay(500);

```

2) Keil+STM32 代码: (做不完回去做)

```

/* USER CODE BEGIN Header */
/**



*****



***



* @file          : main.c
* @brief         : Main program body



*****



***



* @attention
*
* Copyright (c) 2025 STMicroelectronics.
* All rights reserved.
*
* This software is licensed under terms that can be found in the LICENSE file
* in the root directory of this software component.
* If no LICENSE file comes with this software, it is provided AS-IS.
*



*****



***



*/
/* USER CODE END Header */
/* Includes
-----*/
#include "main.h"
#include "adc.h"
#include "gpio.h"

/* Private includes
-----*/
/* USER CODE BEGIN Includes */
#include "headfile.h"

```

```

/* USER CODE END Includes */

/* Private typedef
-----*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define
-----*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro
-----*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables
-----*/
/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes
-----*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code
-----*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
 * @brief The application entry point.
 * @retval int
 */

```

```

int main(void)
{
    /* USER CODE BEGIN 1 */

    /* USER CODE END 1 */

    /* MCU
    Configuration-----*/
    /* Reset of all peripherals, Initializes the Flash interface and the Systick.
*/
    HAL_Init();

    /* USER CODE BEGIN Init */

    /* USER CODE END Init */

    /* Configure the system clock */
    SystemClock_Config();

    /* USER CODE BEGIN SysInit */

    /* USER CODE END SysInit */

    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_ADC1_Init();
    /* USER CODE BEGIN 2 */

    OLED_Init();
    OLED_ColorTurn(0);
    OLED_DisplayTurn(0);

    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
        get_vol_B();
        oled_show();
    }
    /* USER CODE END WHILE */
}

```

```

    /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
}

/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

    /** Configure the main internal regulator output voltage
    */
    HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1);

    /** Initializes the RCC Oscillators according to the specified parameters
    * in the RCC_OscInitTypeDef structure.
    */
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
    RCC_OscInitStruct.HSEState = RCC_HSE_ON;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
    RCC_OscInitStruct.PLL.PLLM = RCC_PLLM_DIV2;
    RCC_OscInitStruct.PLL.PLLN = 12;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
    RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
    RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
    if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
    {
        Error_Handler();
    }

    /** Initializes the CPU, AHB and APB buses clocks
    */
    RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                                |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
    RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
    RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
    RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
    RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
}

```

```

    if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
    {
        Error_Handler();
    }
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
 * @brief This function is executed in case of error occurrence.
 * @retval None
 */
void Error_Handler(void)
{
    /* USER CODE BEGIN Error_Handler_Debug */
    /* User can add his own implementation to report the HAL error return state
     */
    __disable_irq();
    while (1)
    {
    }
    /* USER CODE END Error_Handler_Debug */
}

#ifndef USE_FULL_ASSERT
/**
 * @brief Reports the name of the source file and the source line number
 * where the assert_param error has occurred.
 * @param file: pointer to the source file name
 * @param line: assert_param error line source number
 * @retval None
 */
void assert_failed(uint8_t *file, uint32_t line)
{
    /* USER CODE BEGIN 6 */
    /* User can add his own implementation to report the file name and line number,
       ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line)
    */
    /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
#include "headfile.h"

```

```

float voltage;
float V_0=1.257;
float B;
char text[20];

void get_vol_B()
{
    HAL_ADC_Start (&hadc1);
    uint32_t adc_value=HAL_ADC_GetValue (&hadc1);
    voltage=3.3*adc_value/4096;
    B=(voltage-V_0)/0.013*5.0;
}

void oled_show()
{
    sprintf(text, "V=%3fV", voltage);
    OLED_ShowString(30, 16, (uint8_t *)text, 16, 1);
    sprintf(text, "B=%3fmT", B);
    OLED_ShowString(30, 32, (uint8_t *)text, 16, 1);
    OLED_Refresh();
}

#include "oled.h"
#include "stdlib.h"
#include "oledfont.h"

u8 OLED_GRAM[144][8]; // 择
// • 'Yo-'c

void OLED_ColorTurn(u8 i)
{
    if(i==0)
    {
        OLED_WR_Bye(0xA6, OLED_CMD); // u³ f
    }
    if(i==1)
    {
        OLED_WR_Bye(0xA7, OLED_CMD); // • 'H
    }
}

// BL_H 180n¶ð
void OLED_DisplayTurn(u8 i)
{

```

```

if (i==0)
{
    OLED_WR_Byte (0xC8, OLED_CMD) ;//う^えY
    OLED_WR_Byte (0xA1, OLED_CMD) ;
}

if (i==1)
{
    OLED_WR_Byte (0xCO, OLED_CMD) ;// • ' nY
    OLED_WR_Byte (0xA0, OLED_CMD) ;
}

//リフ
void IIC_delay(void)
{
    u8 t=3;
    while(t--);

}

//う'S°
void I2C_Start(void)
{
    OLED_SDA_Set();
    OLED_SCL_Set();
    IIC_delay();
    OLED_SDA_Clr();
    IIC_delay();
    OLED_SCL_Clr();
    IIC_delay();
}

//%nS°
void I2C_Stop(void)
{
    OLED_SDA_Clr();
    OLED_SCL_Set();
    IIC_delay();
    OLED_SDA_Set();
}

//μlýS° お 番

```

```

void I2C_WaitAck(void) //2 v34zWj 爐
{
    OLED_SDA_Set();
    IIC_delay();
    OLED_SCL_Set();
    IIC_delay();
    OLED_SCL_Clr();
    IIC_delay();
}

// d öh, ög½t
void Send_Byte(u8 dat)
{
    u8 i;
    for(i=0;i<8;i++)
    {
        if(dat&0x80)//%datuk λ 'ě λ ζ'ř' ö
        {
            OLED_SDA_Set();
        }
        else
        {
            OLED_SDA_Clr();
        }
        IIC_delay();
        OLED_SCL_Set();
        IIC_delay();
        OLED_SCL_Clr(); //%S'o'n 藩 İμ 爐
        dat<<=1;
    }
}

// • ø_h, ög½t
//mode: c¾çüÁ 0 -, ± ¼üÁ , ± ¼c¾ç
void OLED_WR_Byte(u8 dat, u8 mode)
{
    I2C_Start();
    Send_Byte(0x78);
    I2C_WaitAck();
    if(mode){Send_Byte(0x40);}
    else{Send_Byte(0x00);}
    I2C_WaitAck();
    Send_Byte(dat);
}

```

```

    I2C_WaitAck();
    I2C_Stop();
}

//a OLED
void OLED_DisPlay_On(void)
{
    OLED_WR_Byte(0x8D, OLED_CMD); //μ 繽±Ê¹
    OLED_WR_Byte(0x14, OLED_CMD); //a yμ 繽±Í
    OLED_WR_Byte(0xAF, OLED_CMD); //μ あ BL
}

//1 SLED
void OLED_DisPlay_Off(void)
{
    OLED_WR_Byte(0x8D, OLED_CMD); //μ 繽±Ê¹
    OLED_WR_Byte(0x10, OLED_CMD); //1 J 繽±Í
    OLED_WR_Byte(0xAE, OLED_CMD); //1 JÁL
}

//, üB' 滂 OLED
void OLED_Refresh(void)
{
    u8 i, n;
    for(i=0; i<8; i++)
    {
        OLED_WR_Byte(0xb0+i, OLED_CMD); //íh A ur'mo •
        OLED_WR_Byte(0x00, OLED_CMD); //íhµI xø ño •
        OLED_WR_Byte(0x10, OLED_CMD); //íh, fI xø ño •

        I2C_Start();
        Send_Byte(0x78);
        I2C_WaitAck();
        Send_Byte(0x40);
        I2C_WaitAck();
        for(n=0; n<128; n++)
        {
            Send_Byte(OLED_GRAM[n][i]);
            I2C_WaitAck();
        }
    }
}

```

```

        }
        I2C_Stop();
    }
}

//清屏
void OLED_Clear(void)
{
    u8 i, n;
    for(i=0; i<8; i++)
    {
        for(n=0; n<128; n++)
        {
            OLED_GRAM[n][i]=0;//清屏
        }
    }
    OLED_Refresh(); //刷新
}

//>>-μ集
//x:0~127
//y:0~63

//t:1 ^ 跳, g i Q

void OLED_DrawPoint(u8 x, u8 y, u8 t)
{
    u8 i, m, n;
    i=y/8;
    m=y%8;
    n=1<<m;
    if(t) {OLED_GRAM[x][i]|=n;}
    else
    {
        OLED_GRAM[x][i]=^OLED_GRAM[x][i];
        OLED_GRAM[x][i]|=n;
        OLED_GRAM[x][i]=^OLED_GRAM[x][i];
    }
}

//>>-L

//x1, y1:起始坐标 //x2, y2:结束坐标 void OLED_DrawLine(u8 x1, u8 y1, u8 x2, u8 y2, u8
mode)
{

```

```

u16 t;
int xerr=0, yerr=0, delta_x, delta_y, distance;
int incx, incy, uRow, uCol;

delta_x=x2-x1; //计算差值
delta_y=y2-y1;
uRow=x1;//设置行和列
uCol=y1;

if(delta_x>0) incx=1; //初始化incx
else if (delta_x==0) incx=0;//如果相等
else {incx=-1;delta_x=-delta_x;}
if(delta_y>0) incy=1;
else if (delta_y==0) incy=0;//如果相等
else {incy=-1;delta_y=-delta_x;}

if(delta_x>delta_y) distance=delta_x; //计算距离
else distance=delta_y;
for(t=0;t<distance+1;t++)
{
    OLED_DrawPoint(uRow, uCol, mode); //绘制点
    xerr+=delta_x;
    yerr+=delta_y;
    if(xerr>distance)
    {
        xerr-=distance;
        uRow+=incx;
    }
    if(yerr>distance)
    {
        yerr-=distance;
        uCol+=incy;
    }
}
}

//x, y:圆心坐标 //r:圆半径

void OLED_DrawCircle(u8 x, u8 y, u8 r)
{
    int a, b, num;
    a = 0;
    b = r;
    while(2 * b * b >= r * r)
    {
        OLED_DrawPoint(x + a, y - b, 1);

```

```

    OLED_DrawPoint(x - a, y - b, 1);
    OLED_DrawPoint(x - a, y + b, 1);
    OLED_DrawPoint(x + a, y + b, 1);

    OLED_DrawPoint(x + b, y + a, 1);
    OLED_DrawPoint(x + b, y - a, 1);
    OLED_DrawPoint(x - b, y - a, 1);
    OLED_DrawPoint(x - b, y + a, 1);

    a++;

    num = (a * a + b * b) - r*r; //计算圆心到点的距离
    if(num > 0)
    {
        b--;
        a--;
    }
}

//设置显示窗口，坐标原点在左上角，x轴向右，y轴向下
//x:0~127
//y:0~63
//size1:wU6 jx8/6x12/8x16/12x24

//mode:0, 1为反显
void OLED_ShowChar(u8 x, u8 y, u8 chr, u8 size1, u8 mode)
{
    u8 i, m, temp, size2, chr1;
    u8 x0=x, y0=y;
    if(size1==8) size2=6;
    else size2=(size1/8+((size1%8)?1:0))*(size1/2); //计算字符的宽度
    //将字符转换为十六进制数
    chr1=chr-' ';
    for(i=0;i<size2;i++)
    {
        if(size1==8)
            {temp=asc2_0806[chr1][i];} //将字符转换为十六进制数

```

```

    else if(size1==12)
        {temp=asc2_1206[chr1][i];} //μ÷K1206
    else if(size1==16)
        {temp=asc2_1608[chr1][i];} //μ÷K1608
    else if(size1==24)
        {temp=asc2_2412[chr1][i];} //μ÷K2412
    else return;
    for(m=0;m<8;m++)
    {
        if(temp&0x01)OLED_DrawPoint(x, y, mode);
        else OLED_DrawPoint(x, y, !mode);
        temp>>=1;
        y++;
    }
    x++;
    if((size1!=8)&&((x-x0)==size1/2))
    {x=x0;y0=y0+8;}
    y=y0;
}
}

//显示字符串
//x, y:显示位置
//size1:字符大小
//*chr:显示字符
//mode:0, 显示一个字符;1, 显示字符串
void OLED_ShowString(u8 x, u8 y, u8 *chr, u8 size1, u8 mode)
{
    while((*chr>=' ')&&(*chr<='~'))//O¶Dz»•p .. •û!
    {
        OLED>ShowChar(x, y, *chr, size1, mode);
        if(size1==8)x+=6;
        else x+=size1/2;
        chr++;
    }
}

//m^n
u32 OLED_Pow(u8 m, u8 n)
{

```

```

u32 result=1;
while(n--)
{
    result*=m;
}
return result;
}

//显示一个数
//x,y :存储地址 //num :要显示的数
//len :字符长度
//size1:显示的位数
//mode:0, • ' ;1, u³ f

void OLED_ShowNum(u8 x, u8 y, u32 num, u8 len, u8 size1, u8 mode)
{
    u8 t, temp, m=0;
    if(size1==8)m=2;
    for(t=0;t<len;t++)
    {
        temp=(num/OLED_Pow(10, len-t-1))%10;
        if(temp==0)
        {
            OLED_ShowChar(x+(size1/2+m)*t, y, '0', size1, mode);
        }
        else
        {
            OLED_ShowChar(x+(size1/2+m)*t, y, temp+'0', size1, mode);
        }
    }
}

//显示一个数
//x,y :存储地址 //num:显示的数
//mode:0, • ' ;1, u³ f

void OLED_ShowChinese(u8 x, u8 y, u8 num, u8 size1, u8 mode)
{
    u8 m, temp;
    u8 x0=x, y0=y;

    u16 i, size3=(size1/8+((size1%8)?1:0))*size1; //计算汉字的偏移量
    for(i=0;i<size3;i++)
    {

```

```

if(size1==16)
    {temp=Hzk1[num][i];}//μ÷16*16
else if(size1==24)
    {temp=Hzk2[num][i];}//μ÷24*24
else if(size1==32)
    {temp=Hzk3[num][i];}//μ÷32*32
else if(size1==64)
    {temp=Hzk4[num][i];}//μ÷64*64
else return;
for(m=0;m<8;m++)
{
    if(temp&0x01)OLED_DrawPoint(x, y, mode);
    else OLED_DrawPoint(x, y, !mode);
    temp>>=1;
    y++;
}
x++;
if((x-x0)==size1)
{x=x0;y0=y0+8;}
y=y0;
}
}

//num 單字數
//space 純空格數
//mode:0, • ' ;1, u³ f

void OLED_ScrollDisplay(u8 num, u8 space, u8 mode)
{
    u8 i, n, t=0, m=0, r;
    while(1)
    {
        if(m==0)
        {
            OLED_ShowChinese(128, 24, t, 16, mode); // 單字顯示
            OLED_GRAM[][]='v
            t++;
        }
        if(t==num)
        {
            for(r=0;r<16*space;r++)           //清屏
            {
                for(i=1;i<144;i++)
                {

```

```

        for (n=0;n<8;n++)
        {
            OLED_GRAM[i-1][n]=OLED_GRAM[i][n];
        }
    }
    OLED_Refresh();
}
t=0;
}
m++;
if(m==16) {m=0;}
for(i=1;i<144;i++) //\x0f
{
    for (n=0;n<8;n++)
    {
        OLED_GRAM[i-1][n]=OLED_GRAM[i][n];
    }
}
OLED_Refresh();
}
}

```

//x, y£º uµ嚮士//sizex, sizey, ¢T³ Øi//BMP[]£º C d Õµç¼T'w

//mode:0, • 'H';1, u³ fY'

```

void OLED_ShowPicture(u8 x, u8 y, u8 sizex, u8 sizey, u8 BMP[], u8 mode)
{
    u16 j=0;
    u8 i, n, temp, m;
    u8 x0=x, y0=y;
    sizey=sizey/8+((sizey%8)?1:0);
    for(n=0;n<sizey;n++)
    {
        for(i=0;i<sizex;i++)
        {
            temp=BMP[j];
            j++;
            for(m=0;m<8;m++)
            {
                if(temp&0x01)OLED_DrawPoint(x, y, mode);
                else OLED_DrawPoint(x, y, !mode);
                temp>>=1;
                y++;
            }
        }
    }
}

```

```

        }
        x++;
        if((x-x0)==sizex)
        {
            x=x0;
            y0=y0+8;
        }
        y=y0;
    }
}

//OLEDµíjjó'»-
void OLED_Init(void)
{
//  GPIO_InitTypeDef  GPIO_InitStructure;
//  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);      //'GA¶,±'
//  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1;
//  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_OD;           //Íd³ ö
//  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;// · ¶n,0MHz
//  GPIO_Init(GPIOA, &GPIO_InitStructure); //³ ö'» PA0, 1
//  GPIO_SetBits(GPIOA,GPIO_Pin_0|GPIO_Pin_1);

//  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
//  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;           //Íd³ ö
//  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;// · ¶n,0MHz
//  GPIO_Init(GPIOA, &GPIO_InitStructure); //³ ö'» PA2
//  GPIO_SetBits(GPIOA,GPIO_Pin_2);

__HAL_RCC_GPIOB_CLK_ENABLE();
GPIO_InitTypeDef GPIO_InitStruct = {0};

GPIO_InitStruct.Pin = GPIO_PIN_8;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_PULLUP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

GPIO_InitStruct.Pin = GPIO_PIN_9;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_OD;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

//  OLED_RES_Clr();
//  delay_ms(200);
}

```

```

// OLED_RES_Set();

OLED_WR_Byte(0xAE, OLED_CMD); //--turn off oled panel
OLED_WR_Byte(0x00, OLED_CMD); //---set low column address
OLED_WR_Byte(0x10, OLED_CMD); //---set high column address
OLED_WR_Byte(0x40, OLED_CMD); //--set start line address Set Mapping RAM
Display Start Line (0x00~0x3F)
OLED_WR_Byte(0x81, OLED_CMD); //--set contrast control register
OLED_WR_Byte(0xCF, OLED_CMD); // Set SEG Output Current Brightness
OLED_WR_Byte(0xA1, OLED_CMD); //--Set SEG/Column Mapping      0xa0'• 'ψ
0xa1u³ £

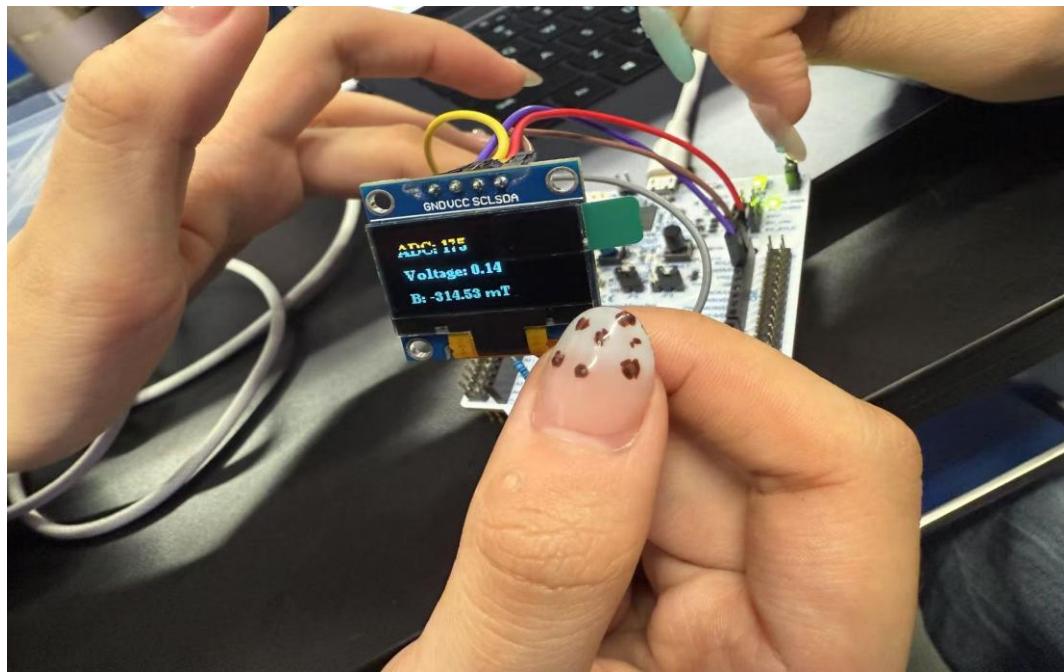
OLED_WR_Byte(0xC8, OLED_CMD); //Set COM/Row Scan Direction      0xc0'• 'ψ
0xc8u³ £

OLED_WR_Byte(0xA6, OLED_CMD); //--set normal display
OLED_WR_Byte(0xA8, OLED_CMD); //--set multiplex ratio(1 to 64)
OLED_WR_Byte(0x3f, OLED_CMD); //--1/64 duty
OLED_WR_Byte(0xD3, OLED_CMD); //--set display offset Shift Mapping RAM
Counter (0x00~0x3F)
OLED_WR_Byte(0x00, OLED_CMD); //--not offset
OLED_WR_Byte(0xd5, OLED_CMD); //--set display clock divide ratio/oscillator
frequency
OLED_WR_Byte(0x80, OLED_CMD); //--set divide ratio, Set Clock as 100
Frames/Sec
OLED_WR_Byte(0xD9, OLED_CMD); //--set pre-charge period
OLED_WR_Byte(0xF1, OLED_CMD); //Set Pre-Charge as 15 Clocks & Discharge as
1 Clock
OLED_WR_Byte(0xDA, OLED_CMD); //--set com pins hardware configuration
OLED_WR_Byte(0x12, OLED_CMD);
OLED_WR_Byte(0xDB, OLED_CMD); //--set vcomh
OLED_WR_Byte(0x30, OLED_CMD); //Set VCOM Deselect Level
OLED_WR_Byte(0x20, OLED_CMD); //--Set Page Addressing Mode (0x00/0x01/0x02)
OLED_WR_Byte(0x02, OLED_CMD); //
OLED_WR_Byte(0x8D, OLED_CMD); //--set Charge Pump enable/disable
OLED_WR_Byte(0x14, OLED_CMD); //--set (0x10) disable
OLED_Clear();
OLED_WR_Byte(0xAF, OLED_CMD);
}

```

2.6 测试结果：

1、液晶显示照片；(换为自己的图)



2、串口接收截图。(换为自己的图)

V = 1.28	B = 8.61
V = 1.74	B = 185.86
V = 1.56	B = 116.45
V = 1.32	B = 25.97
V = 1.25	B = -3.78
V = 0.44	B = -316.13
V = 1.08	B = -68.23
V = 1.17	B = -34.77

3、列表给出磁场测量距离范围 (上下限均需测量，磁场有正负，磁铁反过来测量为负值)；

距离 (mm)	正向磁场 (mT)	距离 (mm)	反向磁场 (mT)
0	500.00	0	-470.03

3	500.00	3	-465.12
6	500.00	6	-430.44
9	478.12	9	-395.56
12	416.33	12	-340.80
15	372.46	15	-260.45
18	205.22	18	-105.67
21	127.19	21	-74.05
24	88.34	24	-53.36
27	40.02	27	-15.12
30	10.64	30	-8.45

2.7 总结与讨论：

1、 你认为本设计有哪些可以改进的地方？

本设计可以从以下几个方面进行改进：首先，优化硬件电路设计，例如采用更高精度的 ADC 模块或增加信号调理电路(如放大和滤波)，以提高测量精度和抗干扰能力；其次，引入动态校准机制，通过软件算法实时修正零点漂移和温漂，确保数据稳定性；此外，可以结合多传感器数据融合技术，比如使用多个霍尔传感器或搭配磁阻传感器，以扩展测量范围并提高可靠性。

2、 研究文献并给出 uT 乃至 nT 的微弱磁场测量方案。

对于微弱磁场 (uT 乃至 nT) 的测量，可以采用高灵敏度传感器，例如磁通门传感器（适用于 uT 级测量，基于磁芯饱和特性）、SQUID 超导量子干涉器件（可达 nT 甚至 pT 级，但需低温环境）、原子磁力仪（利用原子能级跃迁，灵敏度极高）或各向异性磁阻 (AMR) 传感器（适合便携式应用）。这些方案需结合低噪声放大、屏蔽干扰和数字信号处理技术，以满足微弱磁场的精确测量需求。

3 实验三 激光测距与雷达成像

3.1 实验目的：

学习使用 VL53L1X 传感器测距，学会 Processing 的图像化使用。

3.2 程序代码

(1) Arduino 代码：

```
#include <Wire.h>
```

```

#include <VL53L1X.h>
#include <Adafruit_GFX.h>
#include <Adafruit_ST7735.h>
// 屏幕引脚定义（根据实际接线修改）
#define TFT_CS      10
#define TFT_DC      9
#define TFT_RST     8
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

VL53L1X sensor;
const int ledPin = PA5;

// 上次显示的距离值和危险状态
static int lastDistance = -1;
static bool lastWarning = false;

// 缓冲区用于数字转字符串
char distanceStr[10];

void setup() {
    Serial.begin(115200); // 高速串口通信
    Wire.begin();

    // 初始化传感器
    if (!sensor.init()) {
        Serial.println("传感器初始化失败！");
        while (1);
    }
    sensor.setTimeout(500);
    sensor.setMeasurementTimingBudget(200000);
    sensor.startContinuous(50);

    // 初始化屏幕
    tft.initR(INITR_BLACKTAB);
    tft.setSPISpeed(2000000); // 降低 SPI 速度至 2MHz
    tft.setRotation(1);
    tft.fillScreen(ST7735_BLACK);

    pinMode(ledPin, OUTPUT);

    // 显示初始化信息
    tft.setTextColor(ST7735_WHITE);
    tft.setTextSize(2);
    tft.setCursor(10, 10);
}

```

```

tft.print("System Ready");
delay(1000);
tft.fillScreen(ST7735_BLACK);
}

// 安全转换数字为字符串并显示
void safePrintDistance(int distance) {
    sprintf(distanceStr, "%d", distance);

    // 清空距离数值显示区域
    tft.fillRect(30, 35, 100, 20, ST7735_BLACK);

    // 居中显示距离数值和单位
    tft.setCursor(30, 35);
    tft.print(distanceStr);
    tft.print(" mm");
}

void loop() {
    uint16_t distance = sensor.readRangeContinuousMillimeters();

    // 仅在距离变化超过阈值时刷新显示
    if (abs(distance - lastDistance) > 10) {
        lastDistance = distance;

        // 清空标题区域（如果需要）
        tft.fillRect(10, 10, 128, 20, ST7735_BLACK);

        // 显示距离标题（上一行）
        tft.setCursor(10, 10);
        tft.setTextColor(ST7735_GREEN);
        tft.setTextSize(2);
        tft.print("Distance:");

        // 显示距离数值（下一行）
        safePrintDistance(distance);

        // 串口输出（调试用）
        Serial.print("Distance(m): ");
        Serial.println(distance);
    }

    // 危险提示逻辑（显示在第三行）
    bool currentWarning = (distance < 1000);
}

```

```

if (currentWarning != lastWarning) {
    lastWarning = currentWarning;

    // 清空危险提示区域
    tft.fillRect(10, 60, 100, 20, ST7735_BLACK);

    if (currentWarning) {
        digitalWrite(ledPin, HIGH);

        // 显示英文危险提示
        tft.setCursor(10, 60);
        tft.setTextColor(ST7735_RED);
        tft.setTextSize(2);
        tft.print("WARNING1");
    } else {
        digitalWrite(ledPin, LOW);
    }
}

delay(100); // 保持适当延时
}

```

(2) Processing:

```

import processing.serial.*;

Serial myPort;
int distance = 0;
float scanAngle = 0; // 雷达扫描角度
boolean showRadar = true; // 是否显示雷达扫描线
int maxDistance = 2000; // 最大显示距离（用于缩放）
PFont font; // 字体变量

void setup() {
    size(600, 400); // 窗口尺寸

    // 加载支持 ASCII 字符的字体（确保英文显示正常），调整字体大小为 20
    font = createFont("Arial", 20);
    textAlign(CENTER);

    // 打印可用串口列表（用于调试）
    println("Available ports: " + Serial.list());

    try {

```

```

// 初始化串口（修改为你的实际串口号，如 "COM3")
myPort = new Serial(this, "COM9", 115200);
myPort.bufferUntil('\n'); // 以换行符为数据结束标志
} catch (Exception e) {
    println("Serial port connection failed: " + e.getMessage());
}

background(255);
}

void draw() {
background(255);

// 绘制距离数值，修改为 "DISTANCE: 数字" 格式，颜色改为蓝色，移到底部
fill(0, 0, 255); // 蓝色，RGB 值为 (0, 0, 255)
textSize(20); // 设置字体大小
text("DISTANCE: " + distance + " mm", 10, height - 20); // 底部显示

// 绘制距离状态指示，增大圈的尺寸
fill(getDistanceColor());
ellipse(100, 15, 30, 30); // 原尺寸 15 改为 30

// 绘制雷达扫描效果
if (showRadar) {
    drawRadar();
}

// 更新扫描角度
scanAngle += 0.05;
if (scanAngle > TWO_PI) {
    scanAngle = 0;
}
}

// 绘制雷达扫描效果
void drawRadar() {
// 计算圆的半径（根据距离和最大距离缩放）
float radius = map(distance, 0, maxDistance, 50, min(width, height)/2 - 20);

// 绘制雷达底座（同心圆）
noFill();
stroke(0, 100);
strokeWeight(1);
ellipse(width/2, height/2, radius*2, radius*2);
}

```

```

ellipse(width/2, height/2, radius*1.5, radius*1.5);
ellipse(width/2, height/2, radius/2, radius/2);

// 绘制十字线
line(width/2 - radius, height/2, width/2 + radius, height/2);
line(width/2, height/2 - radius, width/2, height/2 + radius);

// 绘制扫描线
pushMatrix();
translate(width/2, height/2);
rotate(scanAngle);

// 扫描线渐变效果
for (int i = 0; i < 30; i++) {
    float alpha = map(i, 0, 30, 150, 0);
    stroke(0, alpha);
    strokeWeight(2);
    line(0, 0, radius, 0);
    radius -= 5;
    if (radius < 0) break;
}
popMatrix();

// 绘制距离指示点，调整绘制逻辑让显示更明显
fill(getDistanceColor());
noStroke();
float indicatorSize = map(distance, 0, maxDistance, 10, 50); // 根据距离
映射指示点大小
ellipse(width/2, height/2, indicatorSize, indicatorSize);
}

// 根据距离返回不同颜色
color getDistanceColor() {
    if (distance < 500) {
        return color(0, 0, 255); // 红色（近）
    } else if (distance < 1000) {
        return color(255, 165, 0); // 橙色（中）
    } else {
        return color(0, 255, 0); // 绿色（远）
    }
}

// 处理串口数据（关键修改：解析 "Distance: X" 格式）
void serialEvent(Serial myPort) {

```

```

String data = myPort.readStringUntil('\n');

if (data != null) {
    data = data.trim(); // 去除首尾空白字符

    // 打印原始数据（用于调试）
    println("Received: " + data);

    // 解析 "Distance: X" 格式的数据
    String[] parts = data.split(": ");

    if (parts.length == 2) {
        try {
            distance = int(parts[1]); // 提取并转换距离值
            distance = constrain(distance, 0, maxDistance); // 限制距离范围
        } catch (Exception e) {
            println("Invalid data format: " + data);
        }
    } else {
        println("Unexpected data format: " + data);
    }
}

// 鼠标点击切换雷达显示
void mousePressed() {
    showRadar = !showRadar;
}

// 键盘控制：+/- 调整最大距离
void keyPressed() {
    if (key == '+') {
        maxDistance += 100;
        maxDistance = constrain(maxDistance, 500, 5000);
        println("Max distance: " + maxDistance + " mm");
    } else if (key == '-') {
        maxDistance -= 100;
        maxDistance = constrain(maxDistance, 500, 5000);
        println("Max distance: " + maxDistance + " mm");
    }
}

```

3.3 讨论

(1) 测量的准确度和环境光有什么关系？

激光测距的准确度受环境光强度显著影响。其核心原因在于传感器通过发射激光并检测目标反射回来的光来计算距离。强环境光会形成强烈的背景噪声，淹没微弱的反射激光信号，导致传感器难以准确识别和计时有效信号。这会引起两种主要误差：一是测量值波动增大，在明亮环境下连续测量同一距离时结果可能出现较大跳变；二是最大有效测距显著缩短，传感器可能无法探测到较远或反射率低的目标（超出其信噪比能力）。虽然 VL53L1X 内置了光学滤光片和算法来抑制环境光干扰，但在极端明亮或具有强烈红外光源的环境下，其测量性能仍会明显劣化。因此，在室内或光照可控条件下通常能获得最佳精度。

(2) 如果实现 3 维激光雷达扫描成像，你认为软硬件应该怎样改进？

1. 硬件改造：

增加二轴步进电机云台（水平+俯仰）驱动单点传感器进行空间扫描，或直接采用多线激光雷达模块；主控升级至高性能处理器处理实时点云数据。

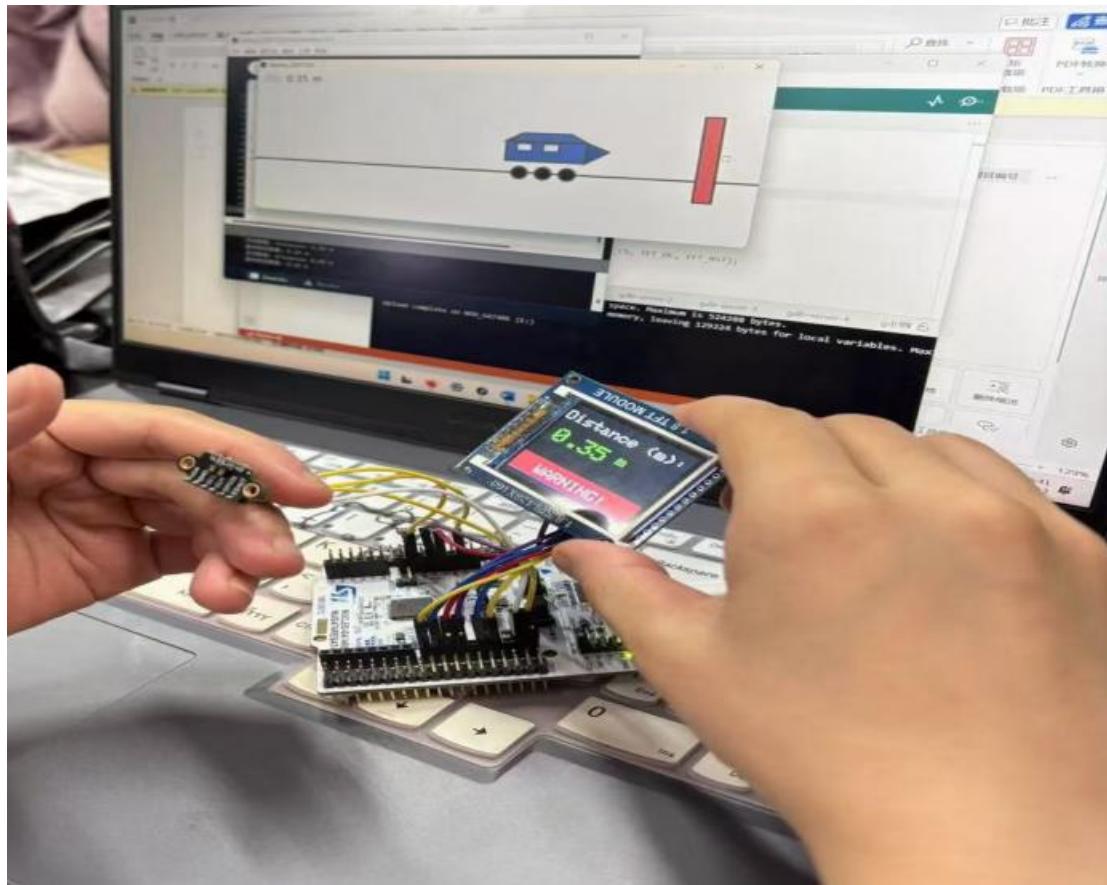
2. 软件优化：

开发球坐标转换算法构建三维点云，在 Processing 中调用 PeasyCam 库实现动态 3D 渲染。

3. 抗干扰与简化：

增加红外滤光片抑制环境光，通过卡尔曼滤波降噪；教学可先实现水平轴二维扫描，再扩展俯仰轴分层验证。

3.4 结果展示



图表 3 结果展示

4 实验四 MPU9250 使用及姿态可视化成像

4.1 实验目的:

学习使用 MPU9250 299DOF 九轴/9 轴姿态或 MPU6050、加速度、陀螺仪，使用 Processing 的图像化显示物体姿态。

4.2 程序代码

(1) Arduion

```

#include <Adafruit_GFX.h>
#include <Adafruit_ST7735.h>
#include <SPI.h>
#include <MPU6500_WE.h>
#include <Wire.h>

// 硬件引脚定义
#define TFT_CS      10
#define TFT_RST     8
#define TFT_DC      9
#define LED_PIN     13

// 初始化组件
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);
MPU6500_WE mpu = MPU6500_WE(0x68);

// 全局变量
xyzFloat accel;           // 加速度原始数据
xyzFloat gyro;            // 陀螺仪原始数据
float accelX, accelY, accelZ; // 转换后的加速度(g)
int stepCount = 0;         // 步数
bool fallDetected = false;
unsigned long lastStepTime = 0;

// 配置参数
const float STEP_THRESHOLD = 1.2;
const float FALL_THRESHOLD = 120.0;
const unsigned long STEP_LOCK = 300;

void setup() {
    // 初始化串口
    Serial.begin(115200);
    Serial.println("System Initialization...");

    // 初始化显示屏
    tft.initR(INITR_BLACKTAB);
    tft.fillScreen(ST77XX_BLACK);
    tft.setTextColor(ST77XX_WHITE);
    tft.setTextSize(1);
    tft.setCursor(0, 0);
    tft.println("Initializing MPU6500...");

    // 初始化 LED
    pinMode(LED_PIN, OUTPUT);
}

```

```

digitalWrite(LED_PIN, LOW);

// 初始化 I2C 总线
Wire.begin();

// 检测 MPU6500 连接
if (!mpu.init()) {
    errorHandler("MPU6500 Communication Error");
}
Serial.println("MPU6500 Detected");
tft.setCursor(0, 15);
tft.println("MPU6500 Detected");

// 执行传感器校准（无返回值版本）
tft.setCursor(0, 30);
tft.println("Calibrating... Keep Still!");
Serial.println("Starting Calibration...");
mpu.autoOffsets(); // 直接调用校准函数
delay(2000); // 等待校准完成
Serial.println("Calibration Completed");
tft.fillScreen(ST77XX_BLACK);

// 配置传感器参数
mpu.setAccRange(MPU6500_ACC_RANGE_2G);
mpu.setGyrRange(MPU6500_GYRO_RANGE_250);
mpu.enableAccDLPF(true);
mpu.setAccDLPF(MPU6500_DLPF_6);
mpu.enableGyrDLPF();
mpu.setGyrDLPF(MPU6500_DLPF_6);

// 初始化完成提示
tft.setCursor(0, 0);
tft.println("System Ready");
delay(1000);
tft.fillScreen(ST77XX_BLACK);
}

void loop() {
    // 读取原始传感器数据
    accel = mpu.getGValues();
    gyro = mpu.getGyrValues();

    // 转换加速度为 g 单位 (MPU6500_WE 库已处理量程转换)
    accelX = accel.x;
}

```

```

accelY = accel.y;
accelZ = accel.z;

// 计步检测
detectStep();

// 摔倒检测
detectFall();

// 更新显示屏
updateDisplay();

// 串口调试输出
debugPrint();

delay(100);
}

// 计步检测函数
void detectStep() {
    float totalAccel = sqrt(accelX*accelX + accelY*accelY + accelZ*accelZ);

    if (totalAccel > STEP_THRESHOLD && millis() - lastStepTime > STEP_LOCK) {
        stepCount++;
        lastStepTime = millis();
    }
}

// 摔倒检测函数
void detectFall() {
    if (abs(gyro.x) > FALL_THRESHOLD ||
        abs(gyro.y) > FALL_THRESHOLD ||
        abs(gyro.z) > FALL_THRESHOLD) {
        fallDetected = true;
        digitalWrite(LED_PIN, HIGH);
    } else {
        fallDetected = false;
        digitalWrite(LED_PIN, LOW);
    }
}

// 显示屏更新函数
void updateDisplay() {
    tft.fillRect(ST77XX_BLACK);
}

```

```

// 加速度显示区
tft.setCursor(5, 5);
tft.setTextColor(ST77XX_BLUE);
tft.println("Accelerometer:");
tft.setCursor(10, 25);
tft.print("X: "); tft.print(accelX, 2);
tft.setCursor(10, 40);
tft.print("Y: "); tft.print(accelY, 2);
tft.setCursor(10, 55);
tft.print("Z: "); tft.print(accelZ, 2);

// 步数显示区
tft.setCursor(5, 75);
tft.setTextColor(ST77XX_YELLOW);
tft.println("Steps:");
tft.setCursor(10, 90);
tft.setTextSize(2);
tft.print(stepCount);
tft.setTextSize(1);

// 状态显示区
tft.setCursor(5, 120);
tft.setTextColor(fallDetected ? ST77XX_YELLOW : ST77XX_GREEN);
tft.print("State: ");
tft.println(fallDetected ? "FALL" : "NORMAL");
}

// 串口调试函数
void debugPrint() {
    Serial.print("Acc: X="); Serial.print(accelX, 2);
    Serial.print(" Y="); Serial.print(accelY, 2);
    Serial.print(" Z="); Serial.print(accelZ, 2);
    Serial.print(" | Steps="); Serial.print(stepCount);
    Serial.print(" | State="); Serial.println(fallDetected ? "FALL" : "NORMAL");
}

// 错误处理函数
void errorHandler(String errorMsg) {
    Serial.println("Critical Error: " + errorMsg);
    tft.fillScreen(ST77XX_RED);
    tft.setTextColor(ST77XX_WHITE);
    tft.setCursor(0, 0);
    tft.println("ERROR:");
}

```

```

tft.setCursor(0, 15);
tft.println(errorMsg);

// 持续闪烁 LED 提示错误
while (1) {
    digitalWrite(LED_PIN, HIGH);
    delay(300);
    digitalWrite(LED_PIN, LOW);
    delay(300);
}
}

```

(2) Processing

```

// Enable lighting
lights();
directionalLight(150, 150, 150, 0, 0, -1);
ambientLight(80, 80, 80);

// Draw fuselage - 改为橙色
fill(#009997);
box(10, 10, 200);

// Draw nose cone - 改为紫色
fill(#0000EE);
pushMatrix();
translate(0, 0, -120);
rotateX(PI/2);
drawCylinder(0, 20, 20, 8);
popMatrix();

// Draw wings and tail - 改为天蓝色
fill(#00333B);

// Wings (triangles and quads)
beginShape(TRIANGLES);
// Top wing surface
vertex(-100, 2, 30);
vertex(0, 2, -80);
vertex(100, 2, 30);
// Bottom wing surface
vertex(-100, -2, 30);
vertex(0, -2, -80);
vertex(100, -2, 30);
// Left tail fin

```

```

vertex(-2, 0, 98);
vertex(-2, -30, 98);
vertex(-2, 0, 70);
// Right tail fin
vertex(2, 0, 98);
vertex(2, -30, 98);
vertex(2, 0, 70);
endShape();

// Wing sides and tail fin sides
beginShape(QUADS);
// Left wing side
vertex(-100, 2, 30);
vertex(-100, -2, 30);
vertex(0, -2, -80);
vertex(0, 2, -80);
// Right wing side
vertex(100, 2, 30);
vertex(100, -2, 30);
vertex(0, -2, -80);
vertex(0, 2, -80);
// Wing top-bottom connection
vertex(-100, 2, 30);
vertex(-100, -2, 30);
vertex(100, -2, 30);
vertex(100, 2, 30);
// Tail fin front
vertex(-2, 0, 98);
vertex(2, 0, 98);
vertex(2, -30, 98);
vertex(-2, -30, 98);
// Tail fin top
vertex(-2, 0, 98);
vertex(2, 0, 98);
vertex(2, 0, 70);
vertex(-2, 0, 70);
// Tail fin side
vertex(-2, -30, 98);
vertex(2, -30, 98);
vertex(2, 0, 70);
vertex(-2, 0, 70);
endShape();

popMatrix();

```

```

}

// Helper function to draw cylinders/cones
void drawCylinder(float topRadius, float bottomRadius, float height, int sides)
{
    float angle = 0;
    float angleIncrement = TWO_PI / sides;

    // Draw sides
    beginShape(QUAD_STRIP);
    for (int i = 0; i < sides + 1; ++i) {
        vertex(topRadius*cos(angle), 0, topRadius*sin(angle));
        vertex(bottomRadius*cos(angle), height, bottomRadius*sin(angle));
        angle += angleIncrement;
    }
    endShape();

    // Draw top (if not a cone)
    if (topRadius != 0) {
        angle = 0;
        beginShape(TRIANGLE_FAN);
        vertex(0, 0, 0);
        for (int i = 0; i < sides + 1; i++) {
            vertex(topRadius * cos(angle), 0, topRadius * sin(angle));
            angle += angleIncrement;
        }
        endShape();
    }

    // Draw bottom (if not a cone)
    if (bottomRadius != 0) {
        angle = 0;
        beginShape(TRIANGLE_FAN);
        vertex(0, height, 0);
        for (int i = 0; i < sides + 1; i++) {
            vertex(bottomRadius * cos(angle), height, bottomRadius * sin(angle));
            angle += angleIncrement;
        }
        endShape();
    }
}

// Draw coordinate system (optional, can be disabled)
void drawCoordinateSystem() {

```

```

pushMatrix();
translate(width/2, height/2, 0);
scale(100);

strokeWeight(0.05);

// X axis (red)
stroke(#FF0000);
line(0, 0, 0, 1, 0, 0);
fill(#FF0000);
text("X", 1.1, 0, 0);

// Y axis (green)
stroke(#00FF00);
line(0, 0, 0, 0, 1, 0);
fill(#00FF00);
text("Y", 0, 1.1, 0);

// Z axis (blue)
stroke(#0000FF);
line(0, 0, 0, 0, 0, 1);
fill(#0000FF);
text("Z", 0, 0, 1.1);

popMatrix();
}

// Keyboard controls
void keyPressed() {
    if (key == 'd' || key == 'D') {
        showDebugInfo = !showDebugInfo;
        println("Debug info: " + (showDebugInfo ? "enabled" : "disabled"));
    }

    if (key == 'm' || key == 'M') {
        useMouseControl = !useMouseControl;
        println("Mouse control: " + (useMouseControl ? "enabled" : "disabled"));
    }

    if (key == 'r' || key == 'R') {
        firstSample = true;
        println("Attitude calculation reset");
    }
}

```

```

// Adjust model size
if (keyCode == UP) {
    modelScale *= 1.1;
    println("Model scale: " + nf(modelScale, 1, 2));
}

if (keyCode == DOWN) {
    modelScale *= 0.9;
    println("Model scale: " + nf(modelScale, 1, 2));
}

// Adjust camera distance
if (keyCode == RIGHT) {
    cameraZ *= 1.1;
    println("Camera distance: " + nf(cameraZ, 1, 1));
}

if (keyCode == LEFT) {
    cameraZ *= 0.9;
    println("Camera distance: " + nf(cameraZ, 1, 1));
}

void draw() {
    background(0);
    camera(width/2, height/2, cameraZ, width/2, height/2, 0, 0, 1, 0);

    // Update attitude (critical for rotation)
    updateAttitude();

    // Draw coordinate system (commented to hide)
    // drawCoordinateSystem();

    // Draw airplane model with updated attitude
    drawPlane();

    // Draw debug information
    if (showDebugInfo) {
        fill(#FFFFFF);
        textSize(12);

        // Display sensor data
        text("Accelerometer (G):\nX: " + nf(RwAcc[0], 1, 2) +
            "\nY: " + nf(RwAcc[1], 1, 2) +

```

```

"\nZ: " + nf(RwAcc[2], 1, 2), 20, 20);

text("Gyroscope (° /s):\nX: " + nf(Gyro[0], 1, 2) +
"\nY: " + nf(Gyro[1], 1, 2) +
"\nZ: " + nf(Gyro[2], 1, 2), 150, 20);

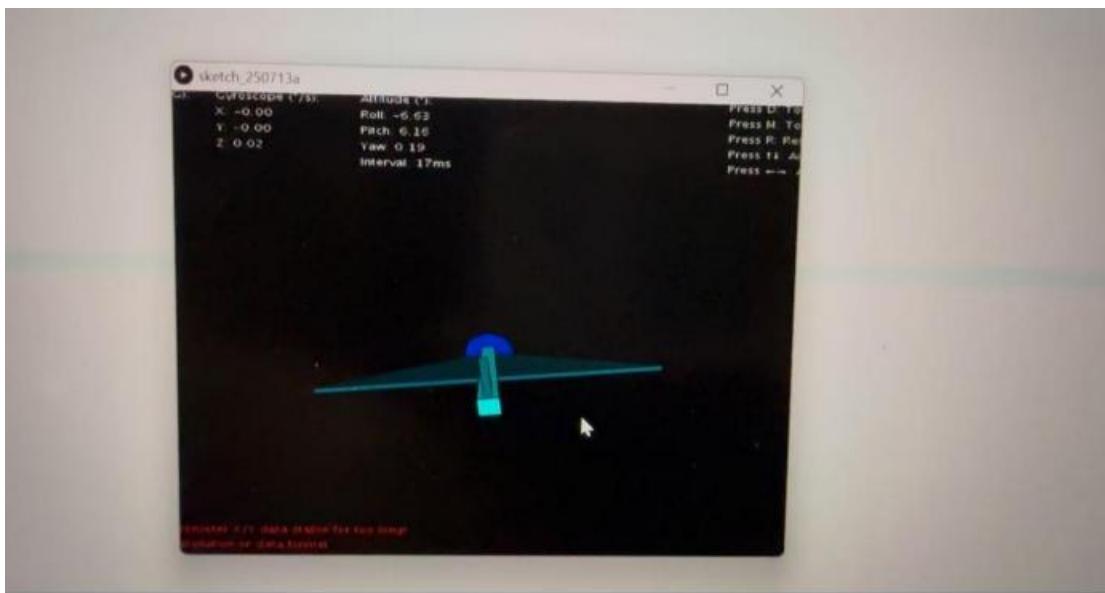
text("Attitude (°):\nRoll: " + nf(RwEst[0], 1, 2) +
"\nPitch: " + nf(RwEst[1], 1, 2) +
"\nYaw: " + nf(RwEst[2], 1, 2) +
"\nInterval: " + interval + "ms", 280, 20);

// Display control hints
text("Press D: Toggle debug info\nPress M: Toggle mouse control\nPress R:
Reset attitude\nPress ↑ ↓: Adjust model size\nPress ←→: Adjust camera
distance", width-180, 20);

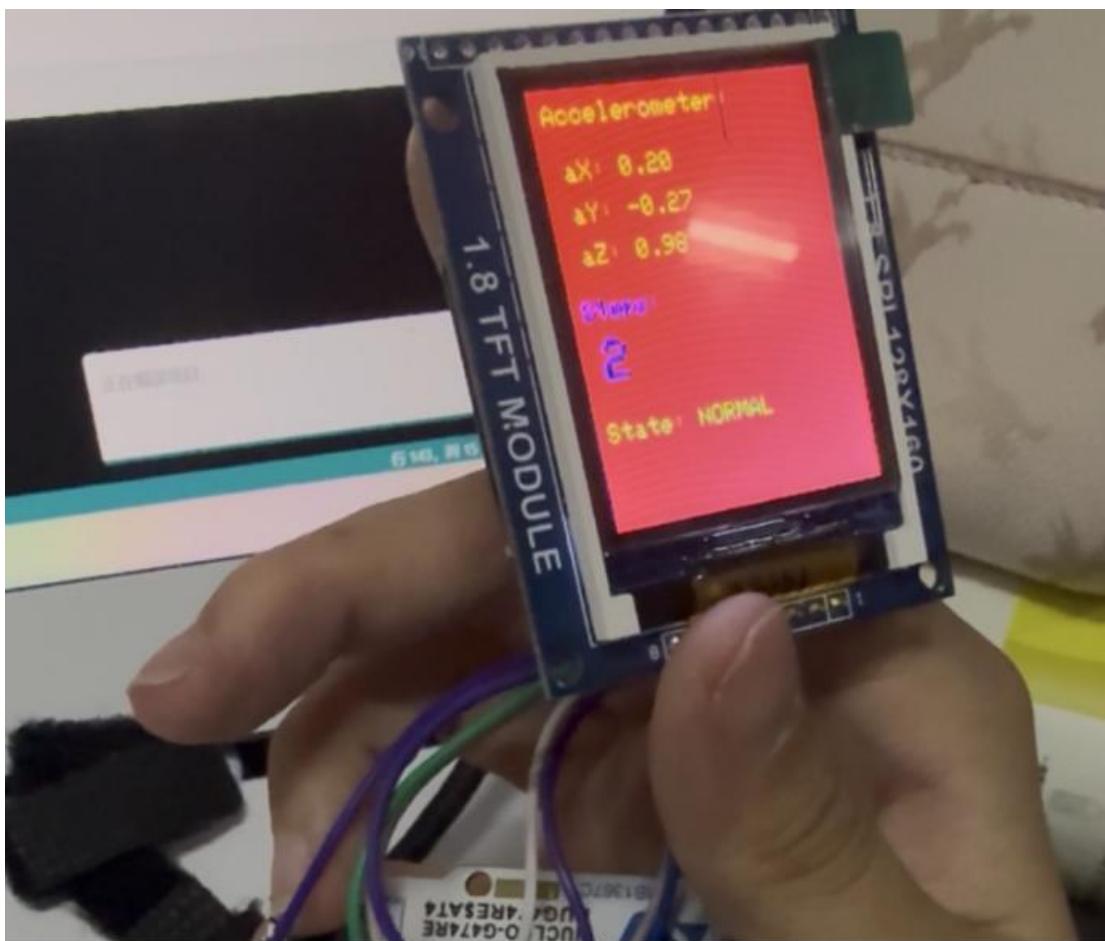
// Display data anomaly warning
if (dataStableCount > MAX_STABLE_COUNT) {
    fill(255, 0, 0);
    text("Warning: Accelerometer X/Y data stable for too long!\nCheck sensor
installation or data format", 20, height-40);
}
}
}

```

4.3 成果展示



图表 4Processing



图表 5 屏幕显示

