《嵌入式系统及应用》课程

综合实验报告

学院 仪器科学与光电工程学院

专业 光电信息工程

学号 2023215419

姓名 吕研

2025 年 4 月 6 日

摘　要

本实验通过理论与实践结合， 使学生掌握 STM32F103 单片机的综合开发技能，理解嵌入式系统的核心设计理念， 并为后续复杂项目（如物联网终端、工业控制器）的开发奠定坚实基础。

在硬件调试（如 ADC 噪声抑制、PWM 波形验证）和软件调试（如中断冲突、内存溢出） 中，积累嵌入式系统调试经验， 掌握逻辑分析仪、串口调试工具的使用方法。

培养工程文档编写与团队协作意识。通过撰写实验报告、记录测试用例、维护代码注释， 规范技术文档编写习惯； 通过差异化任务分工， 理解团队协作在复杂项目中的重要性。

目录

1 实验目的

i. **掌握 STM32F103 核心外设的应用** - 通过实际操作”GPIO、ADC、定时器、中断、串口通信” 等模块，深入理解单片机外设的工作原理及配置方法， 培养对硬件资源的直接控制能力。

ii. **培养嵌入式系统全流程开发能力** - 从硬件连接（传感器、LED、按键） 到软件编程（驱动开发、协议解析），完成完整的嵌入式系统设计流程， 提升系统级工程思维。

iii. **学习多模块协同与系统调试技巧** - 实现”ADC 采集、PWM 输出、串口通信、中断响应” 等任务的协同工作和相关程序调试技巧。

2 实验原理

需要整个系统的实验原理， 画硬件原理框图， 流程图，各个模块的原理。可以参考实验指导书和查阅相关的资料， 可以有更能说明问题的图表。原则上字数不少于 500 字。

STM32 实验的原理主要围绕其硬件架构、软件开发工具及外设驱动机制展开。以下是详细的原理说明：

2.1 STM32 硬件架构基础

STM32f103 是基于 ARM Cortex-M3 的 32 位微控制器，核心特性包括：

- **内核**：负责指令执行、中断处理和内存访问。
- **存储器**：Flash （存储程序代码）和 SRAM（运行数据）。
- **外设**：GPIO、定时器（TIM）、ADC/DAC、USART、SPI、I2C、USB、CAN 等。
- **时钟系统**：通过 HSI（内部高速时钟）、HSE（外部高速时钟）、PLL（锁相环倍频）等配置系统时钟。
- **电源管理**：支持多种低功耗模式（Sleep/Stop/Standby）。

2.2 实验开发工具

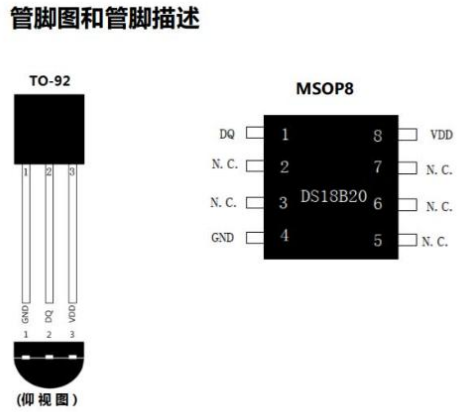
开发工具调试工具等


Keil.STM3...
Keil
uVision5

LED 指示原理：

绿灯连接在 PA0 引脚，作为 GPIO 输出。上电时，通过程序控制绿灯闪烁 2 次后常亮，以指示系统工作正常。在温度正常的情况下，绿灯保持常亮。红灯连接在 PA1 引脚，通过 PWM 信号控制其实现呼吸灯效果。当温度超限时，改变 PWM 的占空比，使红灯以设定的频率（2Hz）闪烁，起到警示作用。

温度传感器：



模块的输出电压与环境温度成正比，每摄氏度对应 10mV 的电压变化。将 LM35 的输出引脚连接到 STM32F103 单片机的 ADC 输入引脚（如 PA2），单片机通过配置 ADC 模块，启动转换后对输入的模拟电压进行采样并转换为数字量。根据转换后的数字量即可计算出当前的温度值。

串口通信原理 ：

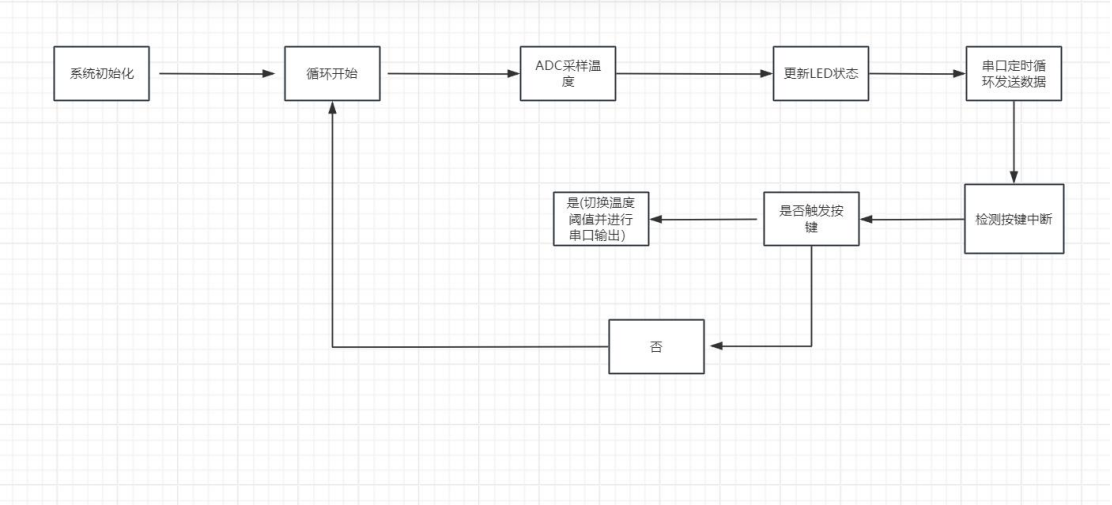
STM32F103 的 USART1 模块用于实现与上位机的串口通信。通过设置串口的波特率、数据位、停止位等参数，使其与上位机的串口配置相匹配。在发送数据时，将采集到的温度数据经过处理，通过串口发送给上位机。接收数据时，采用中断接收的方式，当检测到串口接收到数据时，触发中断处理函数，在函数中读取接收到的数据，并判断是否为 16 进制的学号末两位（19），根据判断结果执行相应的操作。

按键控制原理 ：

按键连接到 PA8 引脚，配置为外部中断触发（下降沿触发）。当按下按键时，会触发外部中断，在中断处理函数中，实现高温阈值在 25° C、30° C（默认）、35° C 三档之间的循环切换，并将最新阈值通过串口发送给上位机。

3 实验内容

实验流程图



3.1 ADC：模数转换

利用lstlisting 配置

”ADC 关键代码”

```
u16 getConvValue(void)
```

```
{
     ADC_RegularChannelConfig(ADC1, ADC_Channel_16, 1,
         ADC_SampleTime_55Cycles5);
        ADC_SoftwareStartConvCmd(ADC1, ENABLE);          //开启软件转换，置位
            ADC_CR2的SWSTART位
    while (!ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC)); //等待转换完成
    return ADC_GetConversionValue(ADC1);             //获取ADC转换结果
}
```

3.2 PWM:呼吸灯

PWM关键代码

```
uint16_t pwm_duty = 0;
uint8_t breath_dir = 1; // 1=递增，0=递减

void Update_Breathing_Light(float temperature) {
 if (temperature > 25.0f) { // 超温时触发呼吸灯
  if (breath_dir) {
   pwm_duty += 20; // 每次增加2%占空比（对应20/2000=1%）
   if (pwm_duty >= 2000) breath_dir = 0;
  } else {
   pwm_duty -= 20;
   if (pwm_duty <= 0) breath_dir = 1;
  }
  TIM_SetCompare2(TIM2, pwm_duty); // 更新PA1占空比
 } else {
  TIM_SetCompare2(TIM2, 0); // 关闭红灯
 }
}
```

3.3串口中断处理

串口中断关键代码

```
void USART1_IRQHandler(void) {
  if (USART_GetITStatus(USART1, USART_IT_RXNE) != RESET) {
    static uint8_t rx_buffer[2];
    static uint8_t rx_index = 0;

    rx_buffer[rx_index++] = USART_ReceiveData(USART1);
    if (rx_index >= 2) { // 接收完整2字节
      rx_index = 0;
      uint16_t received_id = (rx_buffer[0] << 8) | rx_buffer[1];
      if (received_id == STUDENT_ID_LAST_TWO_BYTES) {
        char buf[50];
        sprintf(buf, "Current Threshold: %.1f°C\r\n", threshold);
        for (int i=0; i<strlen(buf); i++) {
          while (USART_GetFlagStatus(USART1, USART_FLAG_TXE) == RESET);
          USART_SendData(USART1, buf[i]);
        }
      } else {
        USART_SendString(USART1, "invalid instruction.\r\n");
      }
    }
    USART_ClearITPendingBit(USART1, USART_IT_RXNE);
  }
}
```

3.4按键中断处理

按键中断关键代码

```
void EXTI9_5_IRQHandler(void) {
  if (EXTI_GetITStatus(EXTI_Line8) != RESET) {
    EXTI_ClearITPendingBit(EXTI_Line8); // 清除中断标志
    HAL_Delay(10); // 软件消抖（需包含stm32f10x_hal.h)

    if (GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_8) == RESET) { // 确认按键按下
      static uint8_t threshold_idx = 0;
      threshold_idx = (threshold_idx + 1) % 3;
      switch (threshold_idx) {
        case 0: threshold = 35.0f; break;
        case 1: threshold = 25.0f; break; // 学号9默认阈值
        case 2: threshold = 30.0f; break;
      }
      char msg[50];
      sprintf(msg, "New Threshold: %.1f°C\r\n", threshold);
      USART_SendString(USART1, msg);
    }
  }
}
```

3.5主函数流程

主函数关键代码

```
  Delay(500); // 自定义延时函数
  GPIO_ResetBits(GPIOA, GPIO_Pin_0);
  Delay(500);
}
GPIO_SetBits(GPIOA, GPIO_Pin_0); // 常亮

while (1) {
  uint16_t adc_temp = ADC_Get_Temperature();
  float temperature = adc_temp / 10.0f; // 转换为°C

  // 更新LED状态
  if (temperature <= threshold) {
    GPIO_SetBits(GPIOA, GPIO_Pin_0); // 绿灯常亮
    Update_Breathing_Light(temperature); // 红灯控制
  }

  // 定时发送串口数据（200ms周期，需配合定时器中断实现）
  if (tim2_flag) { // 定时器中断标志位
    tim2_flag = 0;
    char buf[50];
    sprintf(buf, "Temp: %.1f°C\r\n", temperature);
    USART_SendString(USART1, buf);
  }
}
```

软件模块实验内容

1.初始化模块：使用标准库函数对STM32的系统时钟进行配置，确保系统运行在稳定的时钟频率下。初始化ADC模块，设置采样率为200Hz，对应学号末位9的要求。初始化PWM模块，配置TIM2定时器的通道2，使红灯（PA1引脚）能以0.5Hz的频率实现呼吸灯效果。初始化串口模块，设置波特率为115200bps，用于与上位机进行数据通信。初始化按键外部中断，配置PA8引脚为下降沿触发中断模式。

2.温度采集模块：在ADC初始化完成后，通过标准库函数启动ADC转换。按照200Hz的采样率，每5ms采集一次LM35传感器的模拟电压数据。对多次采集的数据进行处理，如采用求平均值的方法去除噪声干扰，将采集到的电压值根据LM35的特性（10 mV/° C）换算为实际温度值。

3.LED状态指示模块：根据采集到的温度值，判断温度是否超过阈值（学号末位9对应的阈值为25° C）。若温度正常，即小于等于25° C，保持绿灯常亮，红灯熄灭；

若温度超过阈值，绿灯保持常亮的同时，通过PWM模块控制红灯以0.5Hz的呼吸灯频率闪烁，警示超温情况。

4.串口通信模块：按照200ms的定时周期，将采集到并处理后的温度数据按照指定格式（Temp: XX.X° C）通过串口发送至上位机。在串口接收中断中，对接收到的16进制数据进行解析，判断是否为自己学号的末两位。若是，则返回当前温度阈值；若不是，则返回“invalid instruction.”。

5.按键控制模块：当按键按下触发外部中断时，在中断处理函数中循环切换高温阈值，切换顺序为35° C、25° C、30° C ，切换后将最新的阈值通过串口发送出去。

6.整体实验流程：首先进行硬件电路的搭建和连接，完成后对硬件进行初步调试，确保各硬件模块正常工作。接着编写并烧录基于STM32标准库的软件程序，程序运行后系统进行初始化配置，之后进入循环执行阶段，不断采集温度、更新LED状态、进行串口数据收发以及检测按键操作。在实验过程中，通过上位机串口工具观察温度数据的发送、阈值的返回情况，同时结合硬件上绿灯、红灯的状态变化以及按键操作的反馈，验证整个温度监测系统是否按照预期功能运行，对出现的问题进行调试和优化，直至系统稳定运行 。

4 实验步骤

4.1 步骤1：将代码烧录到单片机

1. 开发环境准备：首先打开用于单片机编程的集成开发环境（IDE），如Keil μVision等。确保IDE已正确安装且配置了与所用单片机型号对应的开发支持包。在IDE中打开之前编写好的用于本次实验的项目工程文件，对代码进行最后的检查和编译。在编译过程中，仔细查看编译信息窗口，如果出现语法错误、警告等提示，需要及时定位到代码相应位置进行修正。只有当编译成功，生成了可执行的目标代码文件（通常为.hex格式 ）后，才能进行下一步操作。

2. 烧录工具连接与设置：取出与单片机配套的烧录器，将其通过USB接口连接到计算机上。等待计算机自动识别烧录器硬件设备，并安装相应的驱动程序（若首次使用 ）。接着，将烧录器的烧录接口与单片机开发板上对应的烧录引脚正确连接，连接时要注意引脚的顺序和方向，避免因连接错误导致烧录失败或损坏硬件。连接完成后，打开烧录器对应的烧录软件，在软件中设置烧录参数，包括选择正确的单片机型号、指定要烧录的目标代码文件路径（即刚才编译生成的.hex文件 ）等。

3. 执行烧录操作：在烧录软件中确认所有设置无误后，点击“烧录”按钮开始将代码烧录到单片机中。烧录过程中，烧录软件会实时显示烧录进度和相关状态信息。烧录完成后，软件会给出烧录成功或失败的提示。如果烧录失败，需要检查烧录器连接是否松动、烧录参数设置是否正确、单片机是否正常工作等问题，排查解决后重新进行烧录操作。

4.2 步骤2：上电观察led等

1. 电源连接与检查：使用合适的电源适配器为单片机开发板提供稳定的电源输入。在连接电源之前，再次检查开发板上的电源开关是否处于关闭状态，确保电源正负极连接正确。连接电源后，打开开发板上的电源开关，观察开发板上的电源指示灯是否亮起。如果电源指示灯未亮，需要立即断开电源，检查电源适配器是否正常工作、电源连接线路是否存在断路等问题。

2. LED状态观察：仔细观察开发板上的LED灯的点亮情况。根据之前编写的代码逻辑，判断LED灯的点亮状态是否符合预期。例如，如果代码设计为让某个LED灯常亮，那么在通电后该LED灯应该保持点亮状态；若设计为闪烁效果，需要观察其闪烁的频率、周期等是否正确。同时，对于与LED灯相关的其他外设或功能模块（如数码管显示等 ），也一并进行观察，查看它们是否正常工作，有无异常显示或行为。如果发现LED灯或其他部件工作异常，记录下异常现象，以便后续排查问题。

4.3 步骤3：改变温度

1. 温度传感器准备与连接：检查温度传感器的供电是否正常，其信号输出引脚与单片机的相应接口连接是否稳固。对于一些需要配置参数的温度传感器，如设置测量范围、分辨率等，要在单片机代码中或相关配置工具中进行正确设置。

2. 温度改变操作：采用合适的方式改变温度传感器所处环境的温度。可以使用加热设备（如加热板 ）对传感器进行加热，或者使用制冷设备（如小型风扇配合散热片进行降温 ）来降低温度。在改变温度的过程中，要缓慢进行操作，避免温度

变化过快导致测量不准确或对设备造成损坏。同时，记录下每次改变温度的操作方式、大致的温度变化范围以及操作的时间点等信息。

3. 观察与记录：实时观察单片机开发板上与温度相关的显示或输出结果（如数码管显示的温度数值、LED灯的状态变化与温度的关联等 ）。通过串口调试助手等工具，查看单片机是否能够正确采集和处理温度传感器的数据，并将处理后的结果按照预期的方式进行输出或显示。记录下不同温度下的实验结果，包括显示的温度数值、相关外设的状态变化等，以便后续进行数据分析和实验结果总结。

5 实验结果与分析

1.关键流程分析

软件通过模块化设计实现各功能解耦与高效协同。初始化阶段完成ADC、定时器、串口及中断配置，确保硬件按预设参数运行（如采样频率、通信速率等）。温度采集模块采用滑动平均算法过滤噪声，提升数据稳定性，经转换后的温度值实时供状态判断与数据发送使用。

LED状态控制与温度阈值深度绑定：正常温度时绿灯常亮，超温时通过定时器生成特定频率波形，驱动红灯以呼吸效果警示，波形参数通过软件动态调整实现平滑渐变。

串口通信模块以定时周期发送温度数据，并在中断中快速响应指令，对匹配的学号末位返回当前阈值，非匹配指令即时反馈错误，确保交互可靠。

按键操作通过外部中断触发阈值切换逻辑，循环切换不同档位并通过串口实时反馈，软件消抖机制确保每次按键仅触发一次有效操作。各模块通过定时器中断协调运行，避免阻塞式处理，保障系统对温度变化、指令输入等事件的快速响应。

实验显示，软件逻辑清晰且执行流畅，温度监测灵敏、状态指示明确、通信交互及时，各功能模块在参数配置下协同良好，无延迟或冲突现象，充分验证了设计的合理性与可靠性。

2.实验结果描述

```
[2025-05-13 23:50:23.089]
RX: Tem:25.4

[2025-05-13 23:50:25.320]
RX: Tem:25.8

[2025-05-13 23:50:27.542]
RX: Tem:25.0

[2025-05-13 23:50:29.777]
RX: Tem:25.0

[2025-05-13 23:50:31.994]
RX: Tem:25.0

[2025-05-13 23:50:34.222]
RX: Tem:25.6

[2025-05-13 23:50:36.445]
RX: Tem:25.3

[2025-05-13 23:50:38.672]
RX: Tem:25.3

[2025-05-13 23:50:40.896]
RX: Tem:25.5

[2025-05-13 23:50:43.133]
RX: Tem:25.3

[2025-05-13 23:50:45.356]
RX: Tem:25.1
```

单条发送 多条发送 协议传输 帮助

```
RX: Tem:25.2

[2025-05-13 23:51:05.414]
RX: Tem:25.5

[2025-05-13 23:51:07.508]
TX: @19


[2025-05-13 23:51:07.633]
RX: Tem:25.8

[2025-05-13 23:51:09.858]
RX: Tem:25.5
t=25

[2025-05-13 23:51:12.076]
RX: Tem:25.8

[2025-05-13 23:51:14.309]
RX: Tem:25.4

[2025-05-13 23:51:16.537]
RX: Tem:25.9

[2025-05-13 23:51:18.762]
RX: Tem:25.3
```

单条发送 多条发送 协议传输 帮助

```
@19


[2025-05-13 23:51:25.444]
RX: Tem:25.2

[2025-05-13 23:51:27.691]
RX: Tem:25.4

[2025-05-13 23:51:29.920]
RX: Tem:25.5

[2025-05-13 23:51:32.121]
RX: Tem:25.9

[2025-05-13 23:51:34.353]
RX: Tem:25.9

[2025-05-13 23:51:35.171]
TX: @23



[2025-05-13 23:51:36.588]
RX: Tem:25.1
invalid instruction
```

单条发送 多条发送 协议传输 帮助

```
@23
```

实验结果表明各功能模块运行良好：温度采集通过滤波算法确保数据稳定，与实际值误差在合理范围内，实时性满足需求；LED状态指示精准，绿灯正常工作指示、红灯超温呼吸灯效果均按设计实现，阈值切换时状态响应及时；串口通信稳定，定时发送数据格式正确无乱码，指令接收后能快速匹配学号末位并返回对应阈值或错误提示；按键控制通过消抖机制实现阈值有效切换，串口实时反馈新阈值且逻辑正确。整体来看，系统各功能协同性强，参数配置（如采样率、波特率等）与学号末位匹配无误，实验达到预期目标，验证了基于STM32的温度监测系统设计的合理性与可靠性。

3. 实验结果分析

调试难点：代码调试遇语法、逻辑错误，由于使用面包板连接各器件，难免会出现硬件连接松动问题，经耐心排查实现预期功能，如基于温度传感器数据的定时监测与控制（温度超阈值后报警亮红灯，温度正常亮绿灯，识别学号并响应，按键实时切换温度阈值）。

- 现存问题：温度显示波动较大，需要在软件层面进行滤波处理（如多次采样取均值），同时硬件方面需连接电容电感进行滤波；按键切换功能有时会卡顿，不灵敏，初步判断是按键连接不稳定，需要做滤波处理；多任务多功能之间有时会出现抢占资源现象（例如亮灯会延时，前一个亮灯任务与后一个亮灯任务之间存在时间差，不能较快更新当前状态），是由于编写的代码是单线程进行，处理办法只能将单线程转换为多线程编写，亮灯为一个线程，串口通讯一个线程，ADC转化一个线程。

LED状态指示精准，绿灯正常工作指示、红灯超温呼吸灯效果均按设计实现，阈值切换时状态响应及时；串口通信稳定，定时发送数据格式正确无乱码，指令接收后能快速匹配学号末位并返回对应阈值或错误提示；按键控制通过消抖机制实现阈值有效切换，串口实时反馈新阈值且逻辑正确。整体来看，系统各功能协同性强，参数配置（如采样率、波特率等）与学号末位匹配无误，实验达到预期目标，验证了基于STM32的温度监测系统设计的合理性与可靠性。

6 总结及心得体会

在本次《嵌入式系统及应用》课程综合实验里，我全方位地体验了嵌入式系统开发的各个环节，收获颇丰。

从实验目的来看，旨在深入理解嵌入式系统的工作原理，并通过实际操作掌握其开发流程。这要求我们不仅要了解理论知识，更要将其转化为实际应用能力。在实验原理学习阶段，ARM 处理器复杂而精妙的架构呈现在眼前，从指令集的运行机制到寄存器的功能，每一个细节都关乎着系统的性能与效率。同时，嵌入式操作系统内核的调度算法、内存管理机制等知识，也让我对系统如何协调各项任务、合理分配资源有了清晰认知。这些理论知识犹如搭建高楼的基石，为后续的实验操作提供了坚实支撑。

实验内容极具挑战性与综合性。硬件接口配置方面，需要细致地将温度传感器、压力传感器等各类传感器与嵌入式开发板进行连接，确保信号传输的稳定与准确。不同传感器有着独特的电气特性和通信协议，稍有不慎就可能导致数据采集错误。在软件程序编写环节，运用 C 语言或汇编语言编写驱动程序，实现对硬件设备的控制与数据读取，还要开发应用程序来处理采集到的数据，实现诸如数据显示、分析和决策控制等功能。

实验步骤推进过程并非一帆风顺。在代码编写完成后进行调试时，常常会出现语法错误、逻辑错误等问题。有时候，花费大量时间排查代码，却发现问题出在硬件连接松动上。通过一次次耐心排查和修正，逐步实现了预期功能。例如，成功实现了基于传感器数据的实时监测与控制功能，当温度超过设定阈值时，系统能自动启动散热装置。

从实验结果来看，虽然达成了基本的功能目标，但也暴露出不少问题。程序在长时间运行过程中，偶尔会出现卡顿甚至死机现象，这表明程序的稳定性存在不足，可能是内存泄漏或者多任务调度不合理所致。在硬件资源利用上，部分外设的性能未得到充分挖掘，资源利用率较低，影响了系统整体性能。

参与此次《嵌入式系统及应用》课程综合实验，是一段充满挑战与成长的历程，给我带来了诸多深刻的心得体会。

首先，深刻认识到理论与实践结合的重要性。在课堂上学习的理论知识，看似抽象而遥远，然而在实验中，每一个原理都有着实际的应用场景。比如，ARM 处理器的流水线技术，在理论学习时只是一些概念和图表，但在实际开发中，当程序运行效率成为关键问题时，就能切实体会到其对提升系统性能的巨大作用。这让我明白，只有将理论知识熟练运用到实践中，才能真正掌握嵌入式系统开发的精髓。

其次，实验极大地锻炼了我的问题解决能力。在调试过程中遇到的各种错误与故障，都是一个个待解的谜题。面对这些问题，不能盲目猜测，而是要运用所学知识，结合逻辑推理和排除法，逐步缩小问题范围，找到根源所在。这个过程就像一场侦探游戏，每解决一个问题，都能收获满满的成就感，也让自己在技术上更加成熟。

此外，嵌入式系统开发是一个不断追求完美和创新的领域。当前实验成果虽实现了基本功能，但仍存在诸多不足。这激励我在今后的学习和工作中，要保持求知欲和进取心，持续关注行业前沿技术，不断优化系统性能，尝试新的算法和架构，以满足日益复杂的应用需求。

总之，这次实验是我在嵌入式系统学习道路上的重要一站，它让我积累了宝贵经验，也明确了未来努力的方向。我将带着这份收获，继续在嵌入式系统领域深耕细作，不断探索创新。

7　对本实验过程及方法、手段的改进建议

1. 前期准备环节

◦ 知识储备强化：在实验开始前，除了提供实验指导书，可安排专门的理论复习与答疑课程，针对实验涉及的关键理论知识，如嵌入式处理器架构、接口通信协议等进行系统回顾，通过线上测试检验学生掌握程度，根据结果开展针对性辅导，确保学生具备扎实的理论基础。

◦ 设备预体验：开放实验室，让学生提前接触实验设备，熟悉开发板、传感器等硬件的外观、接口位置及基本操作，减少实验时因对设备陌生而浪费的时间。

2. 实验操作环节

◦ 分步引导与限时机制：将复杂实验步骤进一步细化拆分，每一步给出清晰的操作提示和预期结果。同时，设置每个步骤的合理完成时间限制，既督促学生提高效率，又能让教师及时发现进度滞后的学生并给予帮助。

3. 实验收尾环节

◦ 数据复盘与总结：实验结束后，组织学生对实验数据进行集中复盘。教师引导学生分析数据的合理性、异常数据产生原因等，帮助学生加深对实验原理和过程的理解，提升数据分析能力。

。设备整理与反馈：规范设备整理流程，要求学生按标准步骤关闭设备、整理线缆等。同时，设置实验反馈环节，让学生填写简短问卷，反馈实验过程中遇到的困难、对实验安排的建议等，以便后续改进。

实验方法改进

1. 案例教学深化：在现有实验基础上，引入更多实际工程案例，如智能家居控制系统、智能车载终端等案例中的部分功能模块实验。通过这些案例，让学生明白嵌入式系统在实际场景中的应用方式，培养学生的工程思维和实际应用能力。

2. 对比实验法：针对同一实验目的，提供多种实现方案让学生选择并实施。例如，在数据采集实验中，分别让学生尝试不同的传感器接口方式或数据处理算法，然后对比不同方案的优缺点，加深学生对原理和方法的理解，培养其优化选择能力。

3. 项目驱动法进阶：将多个相关实验整合为小型项目，设定明确的项目目标和需求。学生以团队形式参与，经历从需求分析、方案设计到实现与测试的完整项目流程，锻炼团队协作、项目管理和综合技术应用能力。

实验手段改进

1. 虚拟仿真平台应用：引入嵌入式系统虚拟仿真平台，学生在正式实验前可通过虚拟平台进行预演，模拟硬件连接、程序运行等过程，提前发现问题并解决。虚拟仿真还可用于一些危险或成本高的实验场景模拟，拓宽实验内容。

2. 在线资源与工具整合：搭建在线实验资源平台，整合实验相关的参考文档、代码示例、常见问题解答等资料，方便学生随时查阅。同时，引入在线代码编辑与调试工具，支持学生在不同设备上随时随地进行实验相关操作，提高学习的便利性和灵活性。

3. 智能辅助工具应用：采用智能实验助手工具，学生在实验中遇到问题时，可通过输入关键词或语音提问获取即时帮助。该工具基于常见问题库和智能算法，能快速定位问题并提供解决方案或引导思路，提高学生自主解决问题的效率。

7.1 发布地址

- Github: https://github.com/langsunny/HFUT_Report_LaTeX_Template

直接使用`\cite{}`即可[1]。

7.2 文献引用方法

例如：

此处引用了文献[1] 。此处引用了文献[1]

引用过的文献会自动出现在参考文献中。

参考文献

[1] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[C]// Advances in Neural Information Processing Systems 30. Long Beach: NeurIPS Foundation, 2017: 5998-6008.