综合性项目报告：基于 openharmony 和启航开发板的智能风扇设计

一、 设计任务和要求

实现一个基于 openharmony 和启航开发板的智能风扇设计。要求实现手动挡和自动挡。本项目能够实时显示当前温度、湿度。在手动档下支持调节开关风扇和调节挡位，支持显示风速挡位，支持调节 led 灯。在自动模式下，支持按钮调节不同阈值，是风扇达到阈值后自动启动和关闭。

基于以上功能，还实现了在华为云上显示全部参数并进行调节。实现了基于局域网的 socket 调节风扇设置。

二、 系统分析

对于本项目，需要实现风扇的基础手动控制逻辑，自动控制逻辑，参数显示逻辑，华为云链接逻辑和 socket 控制逻辑。

本小组分工实现了不同的功能模块，并进行了整合。我主要负责的是设计项目整体结构，线程逻辑、空间和优先级设计，实时整合各个功能模块和将系统各个参数合理显示在 led 屏幕。

三、 设计方案

对于温度湿度传感器，需要获取芯片的 IIC 接口获取温度和湿度。采用线程循环获取温湿度，能够实现实时获取温湿度。openharmony 系统提供的线程调度和优先级功能能够很好的实现我们项目设计的需求。

对于屏幕显示，调用 GPIO 接口输出屏幕内容。调用 GUI.c 封装的画线、输出字符串和数字的功能显示屏幕内容。

四、硬件电路图

TLP521GB-S：是一个光耦合器，用于电气隔离和信号传输。

SS8050：是一个 NPN 型晶体管，用于放大和开关电子信号和电源。

SHT30-DIS-B：是一个湿度和温度传感器，通过 IIC 接口与其他设备通信。

motor-2p-610：是一个电机驱动模块。

U1 E53 标准-小板接口：这是一个通用接口，包括 SPI、GPIO、ADC、DAC、IIC、UART 等多种接口。

GND：接地，用于提供参考电位。

D5V、3V3：分别表示 5V 和 3.3V 的电源。

SDA、SCL：这是 IIC 接口的数据线和时钟线。

SPI_SCK、SPI_NSS、SPI_MISO、SPI_MOSI：是 SPI 接口的时钟线、片选线、主设备输入从设备输出线、主设备输出从设备输入线。

GPIO：通用输入输出接口。

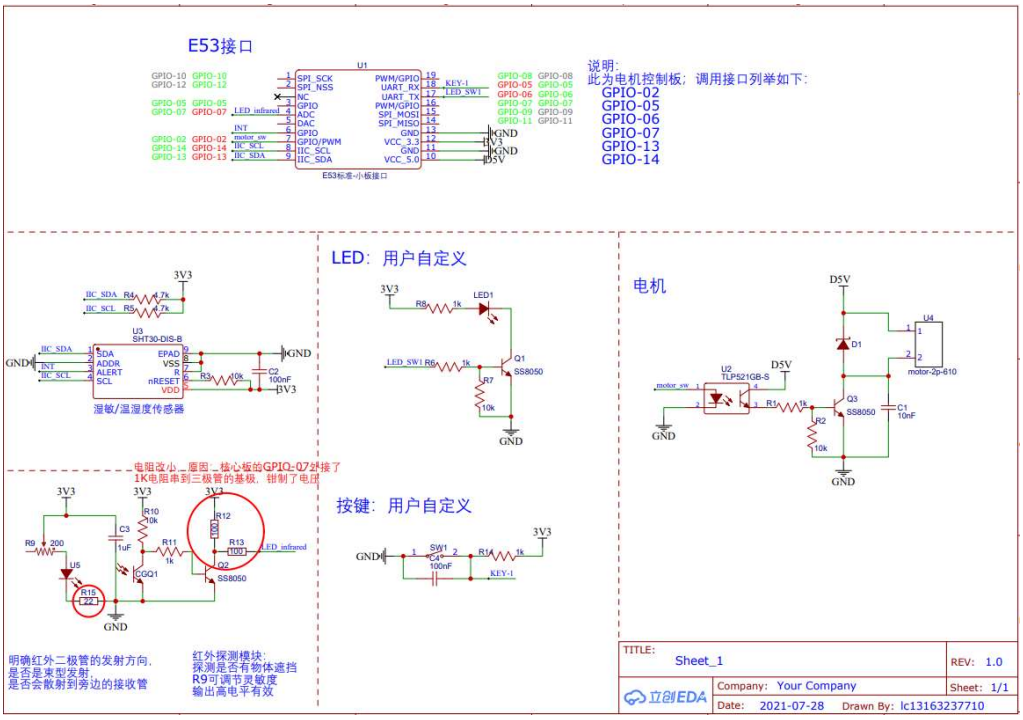
ADC、DAC：分别表示模数转换器和数模转换器。

UART_TX、UART_RX：是 UART 接口的发送线和接收线。

PWM：脉冲宽度调制接口。

INT：中断接口。

motor_sw、LED_SW1、KEY-1：分别表示电机开关、LED 开关和按键。

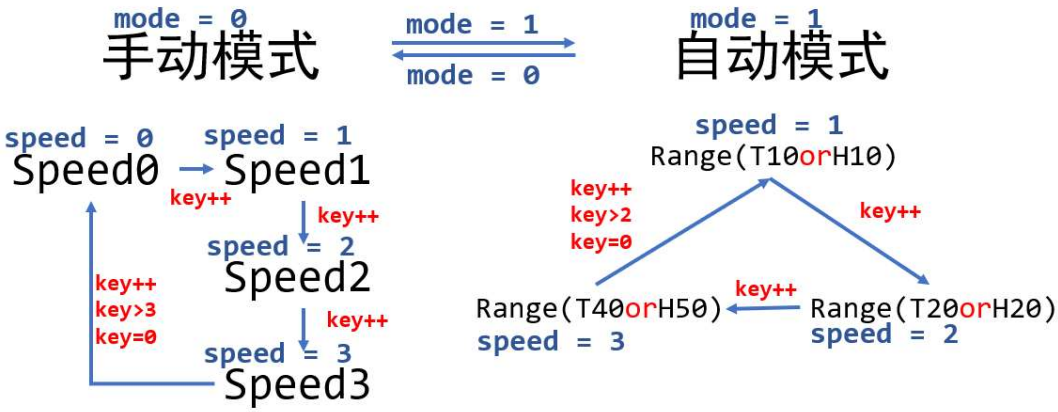


五、软件设计

5.1 项目主体结构和流程：

程序主体分为三个线程，task_main_entry 是主线程，进行各个功能的初始化，wifi 热点的连接，华为云的连接，mqtt 初始化和通过消息队列处理接收和发送。task_sensor_entry 是主要的功能实现进程，实现了读取当前温湿度，将温湿度加入消息队列，在自动模式进行自动开关和调整风速，在开发板显示风速，温度，自动和手动模式和当前湿度。WifiAPTask 是处理 socket 连接的线程。

手动和自动各模式转换逻辑如下：

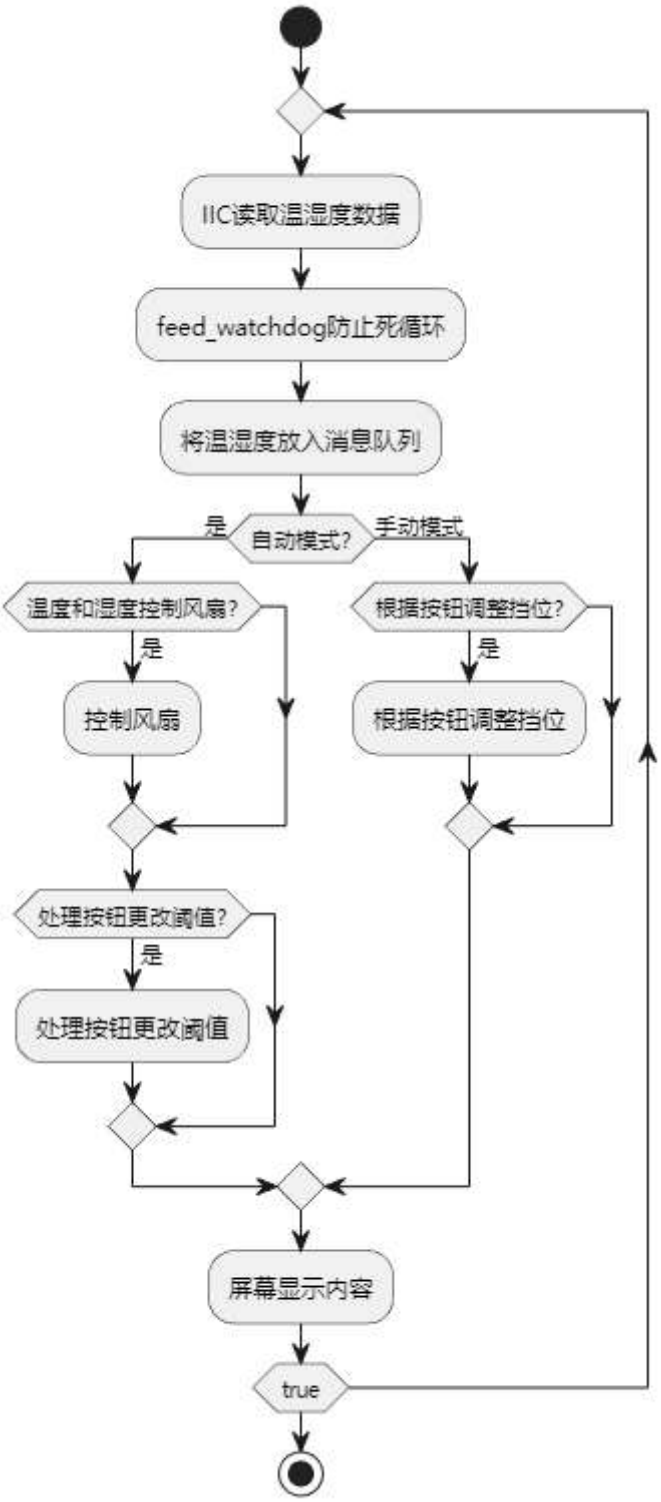


5.2 主线程流程

主线程主要完成各种初始化和消息队列的处理。


进入主线程
OLED初始化和清屏
WIFI连接
华为云连接
GPIO初始化
消息队列接收和发送

5.3 传感器线程流程



传感器线程:

```
static int task_sensor_entry(void)
```

```
{
    app_msg_t *app_msg;
    E53_IA1_Data_TypeDef data;
    // E53_IA1_Init();
    // OLED_Init();   // 初始化 OLED
    // OLED_Clear(0); // 清屏（全黑）

    while (1) {

        E53_IA1_Read_Data(&data); // 此处读取温度湿度数据

        hi_watchdog_feed(); // feed dog 防止死机
        app_msg = malloc(sizeof(app_msg_t));

        if (NULL != app_msg) { // 将数据放入消息队列
            app_msg->msg_type = en_msg_report;
            app_msg->msg.report.hum = (int)data.Humidity;
            app_msg->msg.report.temp = (int)data.Temperature;
            if (0 != osMessageQueuePut(mid_MsgQueue, &app_msg,
0U, 0U)) {
                free(app_msg);
            }
        }

        if (g_app_cb.mode == 1) {
            gpio_getval_auto();
            if (data.Humidity > thh.h || data.Temperature >
thh.t) {
                Speed_StatusSet(g_app_cb.speed);
                printf("speed is %d\n", g_app_cb.speed);
            }
            else {
                hi_pwm_stop(HI_PWM_PORT_PWM2);
            }
        }

        else if (g_app_cb.mode == 0) {
            g_app_cb.speed = gpio_getval1(g_app_cb.speed);
            printf("speed is %d\n", g_app_cb.speed);
        }

        printf("data en queue\r\n"); // 打印数据入队列

        printf("before main page clear \r\n");

        int speed01 = g_app_cb.speed;
        int motor01 = g_app_cb.motor;
        int mode01 = g_app_cb.mode;
        TEST_tempAndHumMenu(speed01, motor01, mode01);

        // OLED_Clear(0);
```

```
    }
    return 0;
}
```

画图函数:

```
void TEST_tempAndHumMenu(int speed, int motor, int mode)
{
    E53_IA1_Data_TypeDef data;
    E53_IA1_Init();
    printf("e is test temp and hum menu \n");

    u8 i;
    if (ledInitSigh == 0) {
        ledInitSigh = 1;
        GUI_DrawLine(0, 10, WIDTH - 1, 10, 1);
        GUI_DrawLine(WIDTH / 2 - 1, 11, WIDTH / 2 - 1, HEIGHT -
1, 1);
        GUI_DrawLine(1, 10 + (HEIGHT - 10) / 2 - 1, WIDTH - 1,
10 + (HEIGHT - 10) / 2 - 1, 1);
        GUI_ShowString(0, 1, "2023-07-08", 8, 1);
        GUI_ShowString(78, 1, "Saturday", 8, 1);
        GUI_ShowString(0, 39, "MODE", 8, 1);
        GUI_ShowString(WIDTH / 2 - 1 + 2, 13, "TEMP", 8, 1);
        GUI_DrawCircle(WIDTH - 1 - 19, 25, 1, 2);
        GUI_ShowString(WIDTH / 2 - 1 + 2, 39, "Hum", 8, 1);
        GUI_ShowString(WIDTH - 1 - 14, 20, "C", 16, 1);
        GUI_ShowString(WIDTH - 1 - 14, 46, "%", 16, 1);
    }

    E53_IA1_Read_Data(&data); // 此处读取温度湿度数据
    hi_watchdog_feed();       // feed dog 防止死机

    char tempStr[10], humStr[10];              // 字符串数组，用
于存储转换后的字符串
    sprintf(tempStr, "%.2f", data.Temperature); // 使用 sprintf
将浮点数转换为字符串，保留两位小数
    sprintf(humStr, "%.2f", data.Humidity);
    GUI_ShowString(WIDTH / 2 - 1 + 10, 20, tempStr, 16, 1);
    GUI_ShowString(WIDTH / 2 - 1 + 10, 46, humStr, 16, 1);

    // }

    if (mode == 0) {
        GUI_ShowString(10, 46, "MANUAL", 16, 1);
        GUI_ShowString(0, 13, "SPEED :>", 8, 1);
        int speed_rlt;
        speed_rlt = speed;
        GUI_ShowString(3, 20, "     ", 16, 1);
        GUI_ShowNum(3, 20, speed_rlt, 2, 16, 1);
    }
    else if (mode == 1) {
```

```
        GUI_ShowString(10, 46, "AUTO  ", 16, 1);
        GUI_ShowString(0, 13, "RANGE :>", 8, 1);
        int speed_rlt;
        speed_rlt = speed;
        if (speed == 1)
        {
            // show 10,10
            GUI_ShowString(3, 20, "10,10", 16, 1);
        }
        else if(speed == 2)
        {
            // show 20,20
            GUI_ShowString(3, 20, "20,20", 16, 1);
        }
        else if(speed == 3)
        {
            // show 40,50
            GUI_ShowString(3, 20, "40,50", 16, 1);
        }
    }

    delay_ms(200);
}
```

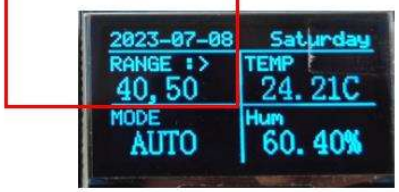
Manual速度显示0/1/2/3

手动/自动模式显示MANUAL/AUTO

温度显示

湿度显示

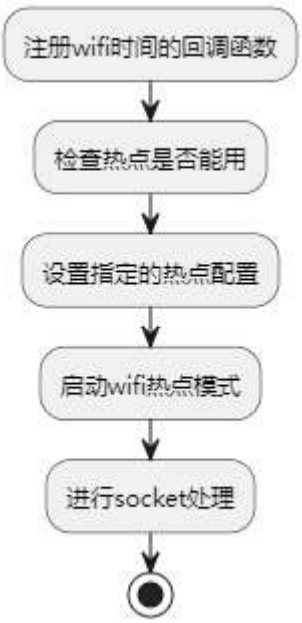
Auto模式下显示温度湿度阈值（或逻辑）



温度显示

湿度显示

5.4 socket 流程处理



5.5 程序清单

gui.c 绘图函数，拥有画线，输出字符串，输出数字到显示屏的功能。 iot_cloud_oc_sample.c 主程序，处理项目主要功能调用其他程序函数。 test.c 调用绘图函数，输出温度湿度挡位模式。 src/E53_IA1.c 处理 IIC 连接。 motor_module.c 风扇控制 oc_mqtt.c mqtt 连接处理 sht3x_i2c.c IIC 连接处理 wifi_connect.c WiFi 连接处理

六、系统调试过程

6.1 屏幕显示刷新不及时问题：

如果每次都刷新整个屏幕会导致刷新很不及时，延迟极大。在 test.c 中创建一个全局变量检测，在第一次执行时刷新整个页面，在之后的执行过程中，由于全局变量值已经更改，之刷新数据部分，可以极大的加快刷新速度，减少屏幕显示延迟。

6.2 线程优先级问题

线程优先级一开始设置的是主线程高于其他线程，每个小功能单独创建了一个线程。这样会导致各个线程之间相互抢断，各个功能延迟相当大。减少代码中 sleep 的数量，将主要功能集成到一个线程中，减少重复初始化的数量，可以显著减少延迟时间。

6.3 接口复用问题

led 和屏幕使用了同样的 gpio8 接口，关闭 led 时会导致屏幕关闭，需要避免使用 gpio8 接口的 led。

6.4 温湿度输出格式

使用 springf 将 float 格式的温湿度转化为字符串再使用 string 输出函数，使得输出格式更加美观。

七、设计总结与心得体会

这次的设计任务深化了我们对 OpenHarmony 系统和启航开发板的理解，并且增强了我们在实际环境下处理多任务调度的能力。OpenHarmony 系统的多线程、优先级以及消息队列设计都为我们提供了强大的工具，在设计和实现过程中，我们更深刻的理解了这些知识的应用。对于硬件，通过控制 GPIO、ADC、DAC 等多种接口，我们了解到了在实际中如何应用这些基本的电子技术。

我们在项目中遇到了一些挑战，比如屏幕刷新不及时、线程优先级设置不合理以及接口复用问题，通过一次次的调试和优化，我们发现，优化是一个持续和迭代的过程，它需要我们不断的尝试、调整和改进，优化不仅仅体现在算法层面，还包括了代码的结构、设计的模式、以及最终的执行效果。同时，这也让我们认识到了良好的编程习惯和代码规范的重要性。

整个项目设计过程，特别是在功能模块整合时，我们更加深刻的认识到了团

队协作的重要性。通过 git 和 github 进行版本管理，使得我们能够有效的同步工作进度和代码回溯，减少了因为代码版本问题导致的困扰。这些工具在今后的学习和工作中，将会给我们带来极大的便利。

通过这次项目的实践，我们认识到了理论知识和实际操作之间的关系，理论知识为我们提供了基础，而实践让我们将这些理论应用到实际中，两者是相辅相成的。项目实践让我们更好的理解了所学知识，也为我们提供了解决实际问题的经验和能力。

总的来说，这次的项目实践是一次非常宝贵的经验，它不仅提高了我们的技术能力，更增强了我们的团队协作能力，为我们在未来的学习和工作中带来了很大的帮助。

八、参考资料

[1] 老师提供的 Schematic_E53 标准板-2-电机+温湿度-V0.4_2021-12-21.pdf

[2] 老师提供的软通启航 KP_IOT 物联网开发板快速入门手册 v2.1(新板).pdf