

**嵌入式系统原理课程设计**

**题目：** 基于STM32F103C8T6和OneNet云平台的物联网智能家居系统

物联网工程 学院 物联网2301

学 号 1034230105

学生姓名 梁开妍

指导教师 孙顺远

2025 年 7 月

目录

[一、 项目概述 3](#_Toc202261465)

[二、系统设计 3](#_Toc202261466)

[2.1硬件设计 3](#_Toc202261467)

[2.1.1元器件选择 3](#_Toc202261468)

[2.1.2硬件原理图设计 3](#_Toc202261469)

[2.1.3原理图和PCB图 5](#_Toc202261470)

[2.1.4系统硬件图 6](#_Toc202261471)

[2.2软件设计 6](#_Toc202261472)

[2.3 APP设计 10](#_Toc202261473)

[三、系统调试 10](#_Toc202261474)

[四、项目总结和优缺点 11](#_Toc202261475)

[五、项目自我评价以及未来展望 12](#_Toc202261476)

# 项目概述

本项目聚焦于开发基于 STM32 单片机的温湿度监测及远程控制系统。借助 DHT11 传感器获取环境温湿度数据，依托 ESP8266 模块上传数据至 OneNET 云平台，支持用户远程操控设备 LED 灯，搭配 OLED 显示屏实时呈现温湿度与 LED 状态。

# 二、系统设计

## 2.1硬件设计

### **2.1.1元器件选择**

选用 STM32F103C8T6 作为主控芯片，其高性能、低功耗特性适配本系统需求，具备丰富外设接口，可高效连接并控制各类模块。

采用 DHT11 传感器采集温湿度，该传感器具备成本低、接口简单、响应速度快等优势，能精准获取环境温湿度数据，为系统提供基础监测信息。

ESP8266（ESP - 01S）模块承担网络通信职责，基于 MQTT 协议实现与 OneNET 云平台的数据交互。其体积小巧、功耗低，内置 TCP/IP 协议栈，便于快速搭建网络连接，助力数据上传与指令接收。

0.96 寸 OLED 显示屏用于实时呈现温湿度数据与 LED 灯状态。OLED 具备自发光、对比度高、响应速度快、视角广等优点，能清晰展示信息，提升用户交互体验。

按键用于手动复位，LED灯作为被控对象与状态指示。

### **2.1.2硬件原理图设计**

1.主控电路（STM32F103C8T6）

引脚 40 接 3.3V 电源，引脚 37、38、20、19 等多处接地（GND），构建稳定供电与参考地，保障芯片正常工作电压（3.3V 为 STM32 核心电压）。

串口通信：PA2（引脚 7）、PA3（引脚 8）分别连 ESP8266 的 RX、TX，实现 STM32 与 WiFi 模块的 UART 串口通信，用于传输 MQTT 协议数据（如温湿度上传、远程控制指令接收）；PA9（引脚 25）、PA10（引脚 26）可作为 USART1 引脚（若代码需调试打印，可复用为调试串口）。

I2C总线：PB6（引脚 33）、PB7（引脚 34）配置为 I2C 功能（SCL、SDA ），连接 OLED 显示屏，通过 I2C 协议传输显示指令与温湿度数据，简化硬件布线（仅需两根线实现通信）。

GPIO控制：PA0（引脚 5）、PA1（引脚 6）等可扩展为通用输入输出（如按键检测、额外传感器接口 ）；PB12（引脚 21）等控制 LED 灯，通过高低电平切换实现亮灭，响应远程控制指令。

2.WiFi模块电路（ESP8266 01S）

VCC（引脚 7）接 3.3V 电源，GND（引脚 2）接地，保障模块供电；EN（引脚 3）为使能引脚，接 3.3V 保持模块常启（也可通过 GPIO 控制模块休眠 / 唤醒，优化功耗）。

RST（引脚 6）为复位引脚，可外接复位电路（或直连 3.3V 上拉，依赖模块内部复位 ），用于模块异常时重启，保障网络连接稳定。

TX（引脚 1）、RX（引脚 8）分别连 STM32 的 PA3、PA2，通过 UART 串口收发数据，遵循 AT 指令集（或 MQTT 协议封装 ），实现温湿度数据上传 OneNET 云平台、接收远程控制指令。

IO0、IO2（引脚4、5）：IO0 用于模块烧录模式切换（接地进入烧录，浮空/上拉为运行），IO2 可复用为通用IO（如状态指示、额外功能扩展），实际需依固件配置调整。

3.显示电路（0.96’OLED IIC）

VCC 接3.3V、GND 接地，SCL（B1）、SDA（B0 ）连 STM32 的 PB6、PB7（I2C 总线引脚 ），通过 I2C 协议与主控通信。OLED 作为从设备，接收 STM32 发送的显示指令（如初始化、字符/图形绘制），实时呈现温湿度、LED 状态。

4.其他模块

按键一端接GND，另一端连 STM32的PA0（A1），通过上拉/下拉电阻（原理图未画，实际需在PCB补全，如PA0配10kΩ上拉电阻到3.3V）实现电平检测。

### **2.1.3原理图和PCB图**

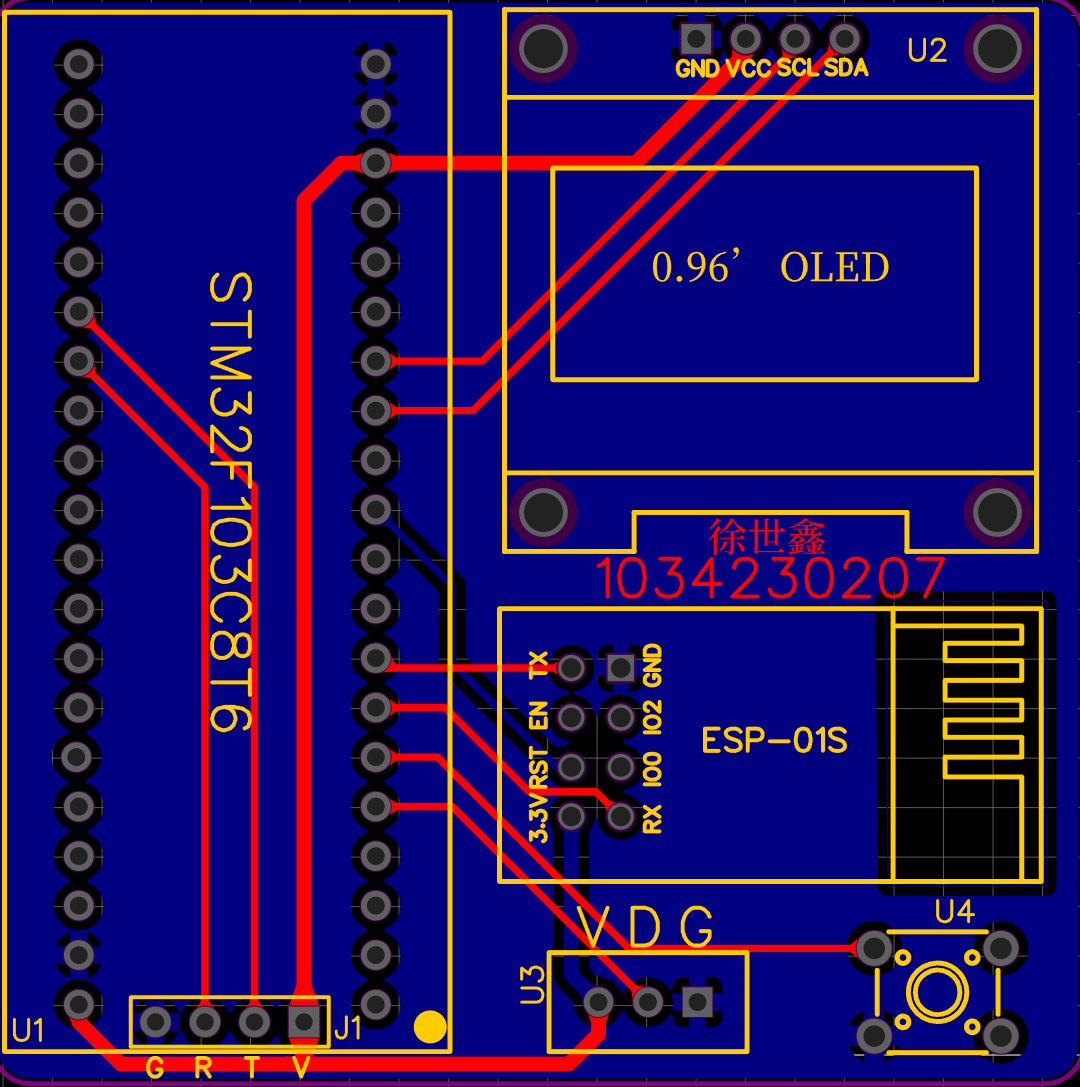


图1 PCB图

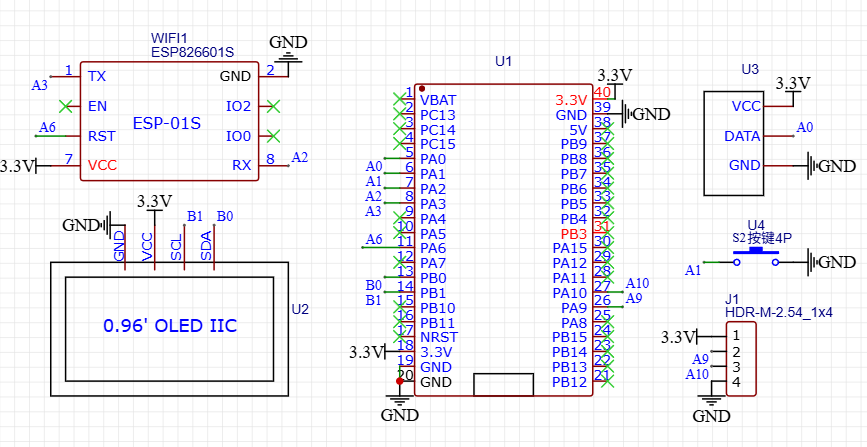


图2 原理图

### **2.1.4系统硬件图**

根据PCB图打板焊接后系统的硬件实物图如图3所示

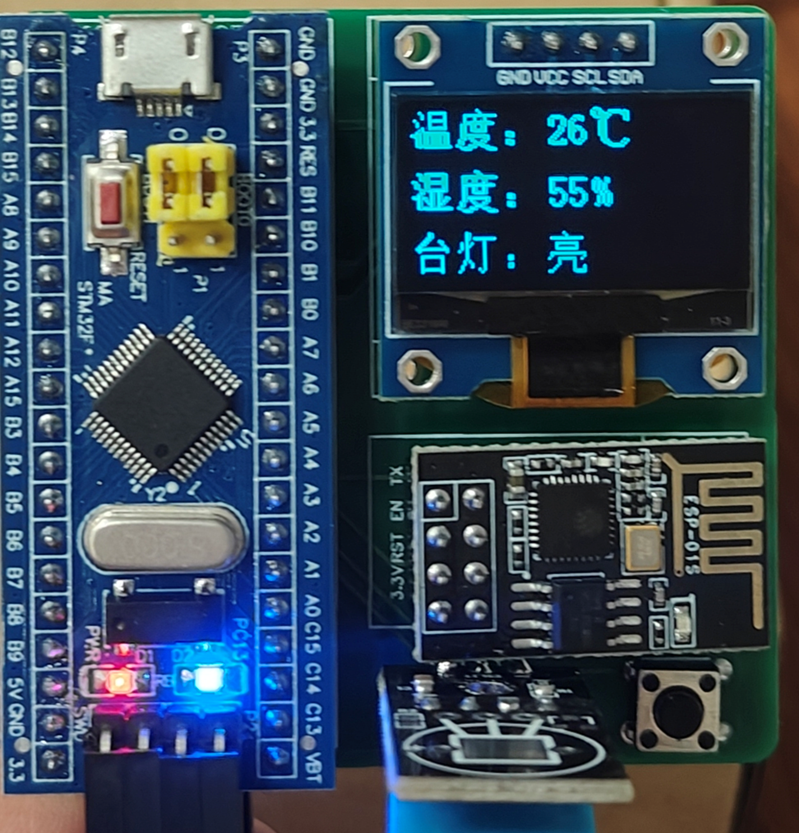


图3 系统硬件图

## 2.2软件设计

系统总体设计流程图如图3所示。

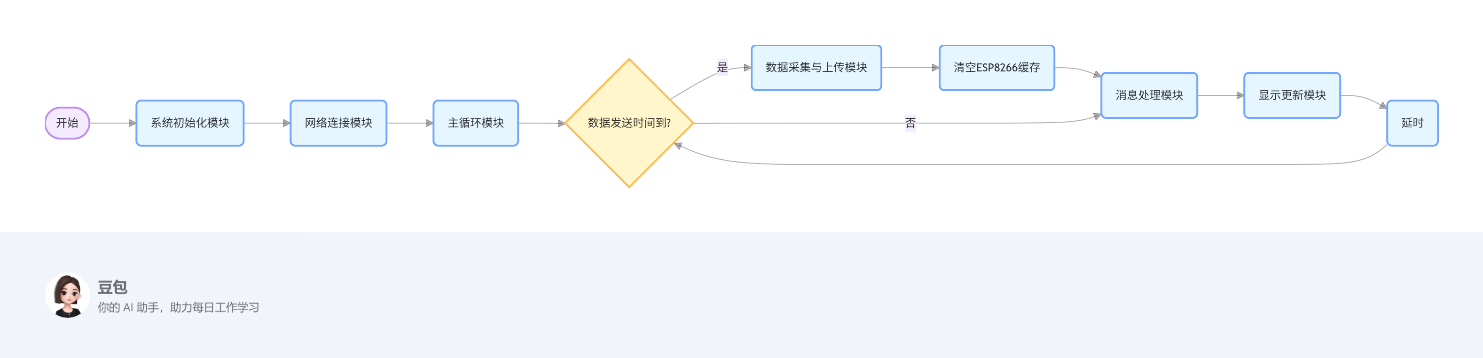


图4 系统总体设计流程图

1.系统初始化模块

功能：初始化单片机的各个外设和中断控制器，为系统的正常运行做好准备。

调用NVIC\_PriorityGroupConfig函数设置中断控制器分组。

调用Delay\_Init函数初始化systick定时器。

调用Usart1\_Init和Usart2\_Init函数分别初始化串口1和串口2。

调用Key\_Init和Led\_Init函数初始化按键和 LED 灯。

调用OLED\_Init函数初始化 OLED 显示屏。

调用DHT11\_Init函数初始化 DHT11 温湿度传感器，如果初始化失败则在OLED 上显示错误信息并等待。

代码模块如下：

void Hardware\_Init(void)

{

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2); //中断控制器分组设置

Delay\_Init(); //systick初始化

Usart1\_Init(115200); //串口1，打印信息用

Usart2\_Init(115200); //串口2，驱动ESP8266用

Key\_Init();

Led\_Init(); //蜂鸣器初始化

OLED\_Init(); //初始化OLED

while(DHT11\_Init())

{

OLED\_ShowString(0,0,"DHT11 Error",16);

DelayMs(1000);

}

OLED\_Clear(); OLED\_ShowString(0,0,"Hardware init OK",16); DelayMs(1000);

}

2.网络连接模块

功能：初始化ESP826 模块，连接到 MQTT 服务器，并登录 OneNET 平台。

调用ESP8266\_Init函数初始化 ESP8266 模块。

发送 AT 指令ESP8266\_ONENET\_INFO连接到 MQTT 服务器，如果连接失败则重试。

调用OneNet\_DevLink函数登录 OneNET 平台，如果登录失败则重新连接 MQTT 服务器并重试。

登录成功后，调用OneNET\_Subscribe函数订阅消息。

代码模块如下：

ESP8266\_Init(); //初始化ESP8266

OLED\_Clear(); OLED\_ShowString(0,0,"Connect MQTTs Server...",16);

while(ESP8266\_SendCmd(ESP8266\_ONENET\_INFO, "CONNECT"))

DelayXms(500);

OLED\_ShowString(0,4,"Connect MQTT Server Success",16); DelayXms(500);

OLED\_Clear(); OLED\_ShowString(0,0,"Device login ...",16);

while(OneNet\_DevLink()) //接入OneNET

{

ESP8266\_SendCmd(ESP8266\_ONENET\_INFO, "CONNECT");

DelayXms(500);

}

OneNET\_Subscribe();

3.主循环模块

功能：在系统运行过程中，不断进行数据采集、上传、消息处理和显示更新等操作。

使用timeCount变量作为计数器，每 10ms 加 1，当计数器达到 100（即 5s）时，触发数据采集和上传操作。

调用ESP8266\_GetIPD函数检查 ESP8266 是否接收到数据，如果接收到则调用OneNet\_RevPro函数进行消息解析和处理。

调用Refresh\_Data函数更新 OLED 显示屏上的温湿度数据和 LED 灯状态。

代码模块如下：

while(1){

if(++timeCount >= 100) //发送间隔5s

{

DHT11\_Read\_Data(&temp,&humi);

OneNet\_SendData(); //发送数据

timeCount = 0;

ESP8266\_Clear();

}

dataPtr = ESP8266\_GetIPD(0);

if(dataPtr != NULL)

OneNet\_RevPro(dataPtr);

Refresh\_Data();

DelayMs(10);

}

4.数据采集与上传模块

功能：使用 DHT11 传感器采集环境温湿度数据，并通过 MQTT 协议将数据上传至 OneNET 云平台。

调用DHT11\_Read\_Data函数读取温湿度数据。

调用OneNet\_SendData函数将数据封装成 MQTT 数据包并发送到 OneNET 平台。

代码模块如下：

if(++timeCount >= 100) //发送间隔5s

{

DHT11\_Read\_Data(&temp,&humi);

OneNet\_SendData(); //发送数据

timeCount = 0;

ESP8266\_Clear();

}

5.消息处理模块

功能：解析和处理从 OneNET 平台接收到的消息，实现对设备的远程控制。

实现细节：

调用ESP8266\_GetIPD函数获取接收到的数据。

调用OneNet\_RevPro函数对数据进行解析，判断消息类型（如 Publish 消息），并根据消息内容控制 LED 灯的开关。

代码模块如下：

dataPtr = ESP8266\_GetIPD(0);

if(dataPtr != NULL)

OneNet\_RevPro(dataPtr);

6.显示更新模块

功能：实时更新 OLED 显示屏上的温湿度数据和 LED 灯状态。

实现细节：

使用sprintf函数将温湿度数据转换为字符串。

调用OLED\_ShowString和OLED\_ShowCHinese函数在 OLED 上显示温湿度数据和 LED 灯状态。

代码模块如下：

void Refresh\_Data(void)

{

u8 buf[3];

sprintf((char \*)buf, "%2d", temp);

OLED\_ShowString(54, 0, buf, 16); // 温度值

sprintf((char \*)buf, "%2d", humi);

OLED\_ShowString(54, 3, buf, 16); // 湿度值

if(led\_info.Led\_Status) OLED\_ShowCHinese(54, 6, 8); // 亮

else OLED\_ShowCHinese(54, 6, 9); // 灭

}

## 2.3 APP设计

设计一款远程控制 App，用于实时获取设备的温湿度数据，并实现对台灯的远程开关控制。该 App 通过与 OneNet 云平台进行数据交互，利用 HTTP 请求获取设备属性信息并进行设备属性的设置。

App 界面会实时展示当前设备的温度和湿度数据，数据格式为 {{temp}} ℃ 和 {{humi}} %。并通过一个开关按钮展示台灯的当前状态（开启或关闭），并允许用户通过点击开关按钮来控制台灯的开关，如图4所示。





图5 APP界面

App 在进入前台显示后，每3秒向云平台发送一次请求，以获取最新的设备属性数据，并更新界面上的温湿度和台灯状态信息。

采用HBuilder X平台开发和运行，登录OneNet平台创建产品和设备并代码中进行接口请求，最终实现数据交互。

# 三、系统调试

采用ST Link和CH340下载程序。

串口调试：利用串口调试工具，输出硬件初始化、网络连接、数据收发等环节的调试信息，分析指令执行结果、数据格式是否正确，定位硬件交互异常（如串口波特率不匹配、指令发送失败）。

MQTT 协议调试：抓包分析 ESP8266 与 OneNET 云平台之间的 MQTT 报文，检查连接、订阅、发布等操作的协议格式，确保硬件传输的数据符合协议规范，解决通信失败、数据丢失问题。

远程控制调试：在云平台发送 LED 控制指令，通过串口调试信息与硬件状态（LED 亮灭），验证指令接收、解析、执行流程，排查 GPIO 控制逻辑、指令解析代码问题。

# 四、项目总结和优缺点

本项目成功实现了基于 STM32 的温湿度监测及远程控制系统。通过 DHT11 传感器实时采集环境温湿度数据，并通过 ESP8266 模块将数据上传至 OneNET 云平台。用户可以通过云平台远程控制设备上的 LED 灯。同时，使用 OLED 显示屏实时显示温湿度数据和 LED 灯的状态。在开发过程中，遇到了一些问题，如程序无法下载、网络连接不稳定、数据解析错误等，通过查阅资料和调试代码，最终解决了这些问题。

1.优点

实现了核心功能：温湿度实时监测、远程控制LED灯、数据定时更新（3秒/次）。

支持与OneNET云平台的MQTT通信，保证数据可靠传输。

采用模块化设计，STM32 与 ESP8266、DHT11、OLED 等模块接口清晰，便于替换升级。

PCB布局合理，分离数字与射频部分，减少干扰。

代码结构清晰，按功能模块化组织（初始化、网络通信、数据处理）。

注释完善，关键函数和逻辑有明确说明，便于维护。

App 界面简洁直观，温湿度与 LED 状态一目了然。

支持开关状态实时同步，操作响应迅速。

2.缺点

网络依赖性强

完全依赖 WiFi 连接，断网时功能失效。

未实现本地数据存储，离线后历史数据丢失。

ESP8266 持续联网功耗较高，未优化休眠机制，不适合电池供电场景。

安全性不足：MQTT 通信未启用TLS加密，数据明文传输，存在安全风险。App令牌生成逻辑简单，易被逆向工程获取。

扩展性限制：传感器接口有限，难以快速扩展新功能（如空气质量监测）。

App 界面固定，未支持自定义设备或场景配置。

用户交互单一：仅支持开关控制，缺乏更丰富的交互（如定时任务、阈值报警）。数据展示仅为数值，未提供图表分析或历史趋势。

# 五、项目自我评价以及未来展望

1.自我评价：良。

评分依据：核心功能实现：完成温湿度实时采集、云端上传、远程控制及本地显示，功能链路完整；技术架构合理性：采用 STM32+ESP8266 的经典物联网方案，硬件模块化设计清晰；代码质量：固件与 App 代码结构规范，关键逻辑注释完整；用户体验：App 界面简洁，数据更新与控制响应较及时但是交互功能不丰富；创新性：传统物联网方案，并未引入AI预测或边缘计算等创新技术。

2.未来展望：将 App 扩展至 Web 端与微信小程序，支持多终端访问；完善用户交互功能，与门锁、空调等设备联动，实现“智能家居系统”；开发 PC 端管理后台，支持多设备分组、数据导出与报表生成；启用 MQTT TLS 加密传输（需 ESP8266 支持 SSL）实现数据安全强化；App 添加数据可视化图表（温度趋势折线图）并支持阈值报警设置（如温度>30℃时 App 推送通知）；训练温湿度预测模型（基于历史数据预测未来 24 小时趋势）；实现异常数据智能识别（如传感器故障自动检测）等等。