# 《物联网与嵌入式技术》

# 大作业课题报告

## 基于手势和语音的小灯开关控制系统


学 生 姓 名：刘厚麟

学科、 专业：材料科学与工程

学　　号：22305020


完 成 日 期：2024.05.05


大连理工大学

Dalian University of Technology

# 目　　录

# 1 项目背景

## 1.1 单片机概述

单片机（Single-Chip Microcontroller，简称 MCU），又称微控制单元(Microcontroller Unit)，是一种集成化的微型计算机系统，它将中央处理器（Central Process Unit；CPU）的频率与规格做适当缩减，并将随机存取存储器（RAM）、只读存储器（ROM）、输入/输出端口（I/O Ports）、定时器/计数器(Timer)、USB、A/D 转换、UART、PLC、DMA 等周边接口，甚至 LCD 驱动电路都整合在单一芯片上，形成一个可独立工作的微型控制系统单元——芯片级的计算机，为不同的应用场合做不同组合控制。

CPU 核心：单片机的核心部分是一个微处理器，执行指令，负责算术逻辑运算和控制整个系统的操作流程。

存储器：内部包含程序存储器（如 Flash ROM 用于存放用户程序代码）和数据存储器（RAM 用于临时存储中间数据和变量）。现代单片机一般允许在线编程和擦写，便于程序修改和升级。

I/O 端口：单片机有多个可配置的 I/O 引脚，用于与外部设备交互，实现数据的采集与输出控制，例如开关量信号、模拟量信号处理等。

中断系统：单片机支持中断功能，使得其能在响应外部事件时快速切换任务，提高实时性。

定时器/计数器：内置的定时器和计数器模块可以产生定时中断，用于精确控制周期性任务或者对外部信号进行计数。

通信接口：诸如 UART、SPI、$I^2C$ 等通信接口可用于与其他设备进行串行通信。

模数转换器（ADC）和数模转换器（DAC）：对于涉及模拟信号处理的应用，单片机可能还集成有 ADC 和 DAC，用于实现数字信号与模拟信号之间的转换。

## 1.2 MQTT 概述

MQTT(Message Queuing Telemetry Transport，消息队列遥测传输协议)，是一种基于发布/订阅(publish/subscribe)模式的“轻量级”通讯协议，该协议构建于 TCP/IP 协议上，由 IBM 在 1999 年发布。MQTT 最大优点在于，可以以极少的代码和有限的带宽，为连接远程设备提供实时可靠的消息服务。作为一种低开销、低带宽占用的即时通讯协议，使其在物联网、小型设备、移动应用等方面有较广泛的应用。

MQTT 是一个基于客户端-服务器的消息发布/订阅传输协议。MQTT 协议是轻量、简单、开放和易于实现的，这些特点使它适用范围非常广泛。在很多情况下，包括受限

的环境中，如：机器与机器(M2M)通信和物联网(IoT)。其在，通过卫星链路通信传感器、偶尔拨号的医疗设备、智能家居、及一些小型化设备中已广泛使用。

# 2　项目整体流程

## 2.1　项目流程

本项目由六部分组成，分别是硬件准备，模块识别，STM32 单片机编程，微信小程序开发，通信机制以及综合测视与调试部分。硬件准备方面有 STM32 单片机、ESP8266 模块等。识别模块通过 MQTT 协议传输到单片机上，STM 单片机通过读取识别数据来进行灯的控制。我所负责的主要内容是 ESP8266 模块和 STM32 单片机编程，来读取识别数据进行灯的控制，以及服务器与单片机的通信。

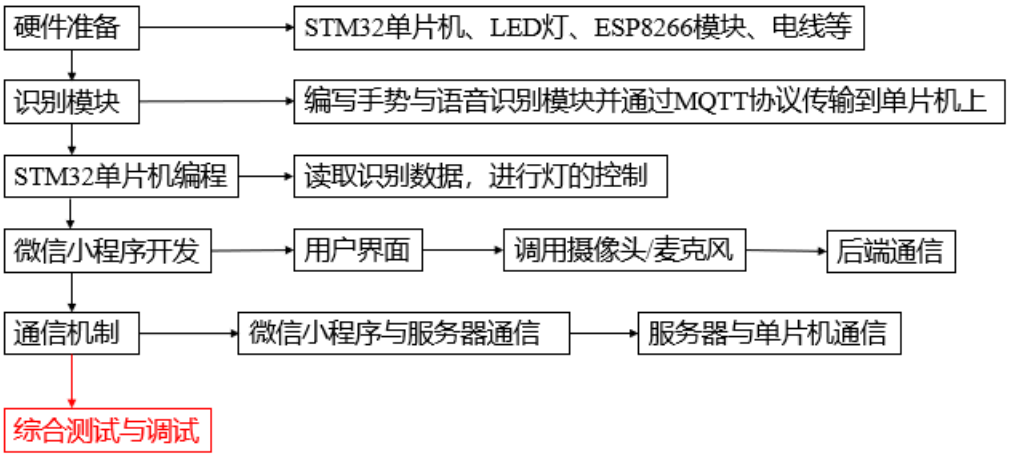


图 2.1　项目流程

3　第三部分

# 3　实现

## 3.1　远程 LED 控制实践

基于 STM32 单片机与 ESP8266 模块的远程 LED 控制实现旨在利用嵌入式技术和物联网通讯协议实现远程 LED 灯的智能控制。选用的核心控制器为意法半导体公司的高性能微控制器 STM32F103C8T6，该单片机因其强大的处理能力和丰富的 IO 资源而广泛应用于各类嵌入式项目中。同时，结合 ESP8266 WiFi 模块，我们搭建了一套能够接入互联网并实现 MQTT 协议通信的控制系统。

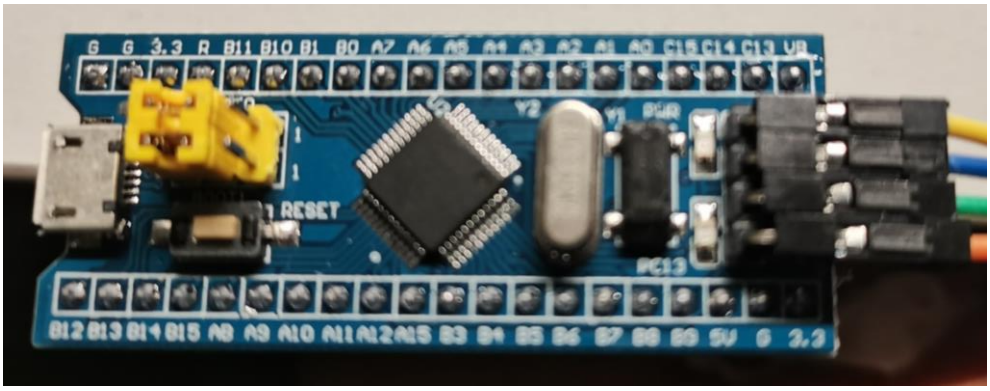


图 3.1　STM32F103C8T6 单片机

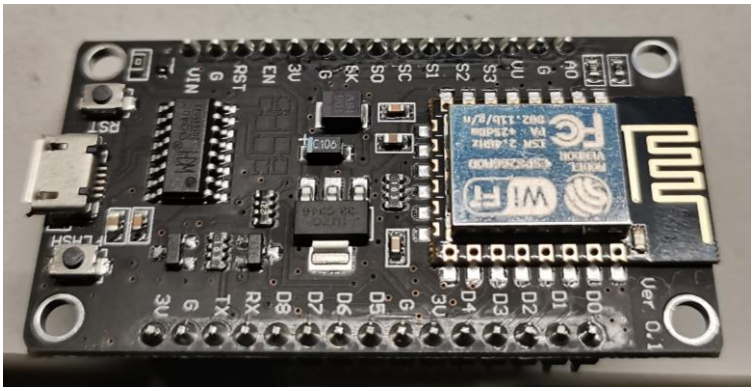


图 3.2　ESP8266 模块

项目的具体架构如下：STM32F103C8T6 单片机通过其内置的 USART2 串行接口与 ESP8266 模块建立连接，确保了两者间稳定的数据交换。在硬件层面，STM32 单片机的

PC13 引脚被配置为输出模式，并直接连接到一个 LED 灯，作为实验中的被控对象。通过改变 PC13 引脚的电平状态，即可控制 LED 灯的亮灭。

在软件设计环节，首先在 ESP8266 端实现了 MQTT 客户端的功能，确保其成功订阅主题并实时接收到来自服务器的控制消息。随后，这些消息通过串口 2 传输至 STM32 单片机。STM32 单片机侧的程序则持续监听串口数据，一旦接收到有效指令，立即解析 JSON 数据中的"LED"参数值，并据此更新 PC13 引脚的输出状态。

在网络通信部分，ESP8266 模块被预先编程以订阅指定的 MQTT 服务器的主题“STM32”。服务器发送的消息数据采用 JSON 格式封装，包含两种不同的控制指令，即{"params":{"LED":0}}表示关闭 LED 灯，{"params":{"LED":1}}则指示打开 LED 灯。ESP8266 接收到此类数据包后，解析 JSON 内容并提取出 LED 控制指令。

实验过程中，针对接收到的不同数值（0 或 1），STM32 会做出相应的响应：若接收到数值 1，则通过设置 PC13 引脚为低电平来驱动 LED 灯亮起；反之，若接收到数值 0，则将 PC13 引脚置为高电平，从而关闭 LED 灯。

整个项目过程不仅验证了 STM32 单片机与 ESP8266 模块间串行通信的有效性，也展示了基于 MQTT 协议的物联网应用如何实现远程设备的精确控制。通过此基础实验，我们可以进一步扩展到更为复杂的智能家居系统或工业自动化场景，实现更多设备的远程监控和智能化管理。

## 3.2 通信部分

### 3.2.1 程序实现

这部分内容是基于 STM32 平台，通过 UART 接口与 ESP8266 WiFi 模组进行通信，实现 WiFi 联网、MQTT 协议的客户端功能，如连接服务器、发布和订阅消息。定义了四个全局变量用于存储串口 2 接收到的数据的缓冲区，标记串口 2 接收数据开始的标志位，记录串口 2 接收的数据字节数，标记串口 2 接收数据结束的标志位。

定义了一系列函数解析传入的 json 格式数据，查找键为"params.LED"的值，并将其转换为整数赋给 LED_status；串口 2 数据接收处理：每当串口 2 的 RXNE 标志位被置位时，接收 1 个字节数据，并将其存入接收缓冲区中，同时更新接收状态标志。清除串口 2 接收缓冲区内容，并重置接收相关标志位。向 ESP8266 发送命令并通过串口 2 接收回复，等待接收完指定的数据或超时，然后检查是否收到预期的回复。

配置 ESP8266 连接到指定的 Wi-Fi 网络。

ESP8266_connect_server：连接到 MQTT 服务器。

ESP8266_reset：复位 ESP8266。

ESP8266_send_msg：构造并发送一条 MQTT 发布消息。

ESP8266_receive_msg：检查接收缓冲区是否有新的 MQTT 订阅消息，并解析 JSON 数据。

ESP8266 的初始化过程包括设置为 Station 模式、关闭回显、禁用自动连接 Wi-Fi、复位 ESP8266、配置 Wi-Fi、设置 MQTT 用户信息、连接 MQTT 服务器、订阅主题等一系列步骤，并在 OLED 显示屏上显示当前步骤。初始化完成后，打印 ESP8266 初始化成功的消息。

```
/**
 * @brief          esp8266发送数据
 * @param[in]      none
 * @retval         返回0发送数据成功,返回1发送数据失败
 */
uint8_t esp8266_send_msg(void)
{
    uint8_t retval =0;
  uint16_t count = 0;
    static uint8_t error_count=0;
    unsigned char msg_buf[256];

  sprintf((char *)msg_buf,"AT+MQTTPUB=0,\"PUB_TOPIC\",\"JSON_FORMAT\",0,0\r\n",temp_value,humi_value);
    HAL_UART_Transmit(&huart2, (unsigned char *)msg_buf,strlen((const char *)msg_buf), 1000);
  HAL_UART_Transmit(&huart1, (unsigned char *)msg_buf,strlen((const char *)msg_buf), 1000);
    while((receive_start == 0)&&(count<500))
    {
        count++;
        HAL_Delay(1);
    }
    if(count >= 500)
    {
        retval = 1;
    }
    else
    {
        HAL_Delay(50);
        if(strstr((const char*)receive_buf, "OK"))
        {
            retval = 0;
            error_count=0;
```

图 3.6　与 ESP8266 WiFi 模组实现通信的部分程序

## 3.3　MQTT 服务器搭建

EMQX 是一个开源的 MQTT 消息服务器，支持高并发、低时延的消息传输。以下是搭建 MQTT 服务器的步骤：

(1) 下载并安装 EMQX，详细步骤可参考官方文档。

(2) 启动 EMQX 服务，确保服务器正常运行。

(3) 配置 WebSocket over SSL (WSS) 协议支持，以便小程序可以通过安全的 WebSocket 连接访问 MQTT 服务器。

### 3.3.1 小程序端设备连接与通信

在小程序中使用 MQTT.js 等 MQTT 客户端库，通过 WSS 协议连接至 EMQX 服务器。

订阅主题：“stm32”以接收来自单片机的消息。

发布消息格式示例：

熄灭 LED 灯：{"params":{"led":1}}

点亮 LED 灯：{"params":{"led":0}}

监听来自服务器的消息，处理单片机反馈的状态。

### 3.3.2 单片机端（ESP8266 模块）

使用 keil5 等相关开发工具开发固件。配置 ESP8266 连接至 EMQX 服务器，订阅主题：“stm32”以接收来自小程序的消息。解析接收到的 MQTT 消息，根据 led 参数控制单片机上的 LED 灯状态。

### 3.3.3 实现原理

小程序通过 WSS 连接到 EMQX 服务器，向主题“stm32”发布 LED 控制指令。ESP8266 模块订阅“stm32”主题，监听来自小程序的指令。当 ESP8266 收到消息后，解析消息内容，根据指令控制 LED 灯状态。ESP8266 将 LED 状态反馈至 EMQX 服务器，再由小程序订阅并获取 LED 状态信息。

## 4.总结

通过本次物联网与嵌入式技术课程，我学习到了真正的软硬件结合的嵌入式技能。从本此项目中，我学到了如何使用 MQTT 协议传输信息，学习了如何使用 Keil5 与 HAL 库编写 stm32 与 esp8266 程序，通过不断地 debug 达到在项目中学习知识的目的。经过这次宝贵的项目经验，让我对以后的嵌入式学习有了更深刻的了解。