重庆邮电大学

学生课程设计报告册

学年学期：2023-2024学年 □春 ■秋学期

课程名称：课程设计（物联网节点设计）

学生学院：自动化学院/工业互联网学院

专业班级：08052103

学生学号：2021213198

学生姓名：简年杰

学生成绩：

指导教师：孟振亚

一、课程设计任务及要求

设计题目	基于 STM32 的智能家居系统

设计任务：

1.设计利用单片机系统技术设计智能家居系统。

2.利用 EMQ 云平台进行数据存储与转发。数据采集后将数据发送到 EMQ 平台，再通过

EMQ 平台发送到微信小程序上。

3.设计微信小程序来检测温湿度、光照度、控制 LED、报警器。

4.利用多个传感器进行检测和控制。

5.处理紧急情况，例如紧急报警和开灯。

6.夜晚自动开灯。

设计要求：

1、数据采集端采集数据、控制指令执行端处理数据、微信小程序作为可视化界面查看数据或者控制。

2、数据采集后使用 MQTT 协议的发布/订阅模式发送到 MQTT 服务器，MQTT 服务器信息再将数据发给微信小程序。

课程对培养目标的支撑：

1、能够通过系统优化和单元集成，进行传感器与智能终端、网络传输、智能信息处理、工业物联网等部件或系统的设计，体现创新意识；

2、能够正确采集、整理、分析与解释实验数据，获得合理有效的结论；

3、掌握现代工程工具，以及示波器、信号源、万用表等仪器仪表的使用方法，并能用于分析物联网复杂工程问题；

4、具有组建、协调和负责团队的能力，能够承担团队成员以及负责人的角色。

指导教师（签字）	

二、人员及分工

姓名	学号	班级	设计分工	联系电话
简年杰	**2021213198**	**08052103**	软件设计 **0.5**	**18225460871**
陈中印	**2021213240**	**08052104**	硬件以及测试 **0.5**	**19922237052**

目录

摘　要

随着物联网技术的飞速发展，物联网技术在家居中也有了更多的应用。由于物联网可以提升效率和便利性，提高生产效率和工作效率，同时又可以实现智能化管理，对物理设备进行实时监测和数据采集，可以获得更多的数据和信息，从而帮助企业、机构或个人做出更明智的决策。通过智能化管理和自动化控制，减少人工操作和管理的成本，提高资源利用率。因此物联网技术在智能家居中应用广泛。

本次设计采用多个传感器收起环境数据，例如光照，温湿度然后采集数据转发到服务器，系统可以通过传感器和监控设备收集数据上发服务器，然后在微信小程序实时监测家庭安全状态，并及时发出警报和通知。而便利性则体现在用户可以随时随地通过手机等终端设备，对家中的设备进行远程控制，方便快捷。

关键词：物联网，数据采集，远程监控

1 设计说明

1.1 设计概述

随着科技的不断进步和智能化的快速发展，智能家居系统成为了现代家庭生活中的热门话题。传统的家居设备往往需要人工操作和管理，缺乏智能化和自动化的特性，无法满足人们对高效、便利、安全的居住环境的需求。因此，设计一个智能家居系统来实现家居设备的互联互通和智能化管理变得迫切。在过去几年里，物联网技术的快速发展为智能家居系统的实现提供了技术基础。物联网技术通过连接家庭中的各种设备，并通过传感器、嵌入式系统和云计算等技术，实现了设备之间的数据共享和远程控制。这为智能家居系统的设计和实施提供了可行性。

1.2 原理分析

1.2.1 DHT11 温湿度传感器

DHT11 温湿度传感器是一种数字式的温湿度传感器，使用一对电阻温度传感器和湿度传感器来测量环境的温度和湿度。它通过一个简单的数字信号接口与微控制器或单片机连接，可以轻松地获取温度和湿度数据。

工作原理如下：

(1)温度测量：DHT11 传感器中的温度传感器基于热敏电阻的原理。随着温度的变化，热敏电阻的电阻值也会变化。传感器测量这种电阻值的变化，并将其转换成数字信号。

(2)湿度测量：湿度传感器采用一种湿度敏感的材料。这个材料的电阻值会随着周围环境湿度的变化而改变。传感器检测这种电阻值的变化，并将其转换成数字信号。

(3)单线制串行接口，使系统集成变得简易快捷。超小的体积、极低的功耗，信号传输距离可达 20 米以上，使其成为各类应用甚至最为苛刻的应用场合的最佳选则。传感器测量温度和湿度后，将数据转换成数字信号，然后通过其数字信号接

口输出给微控制器或单片机。

该传感器模块将 4 引脚封装的 DHT11 引出 3 个引脚，如图 1.1 所示。

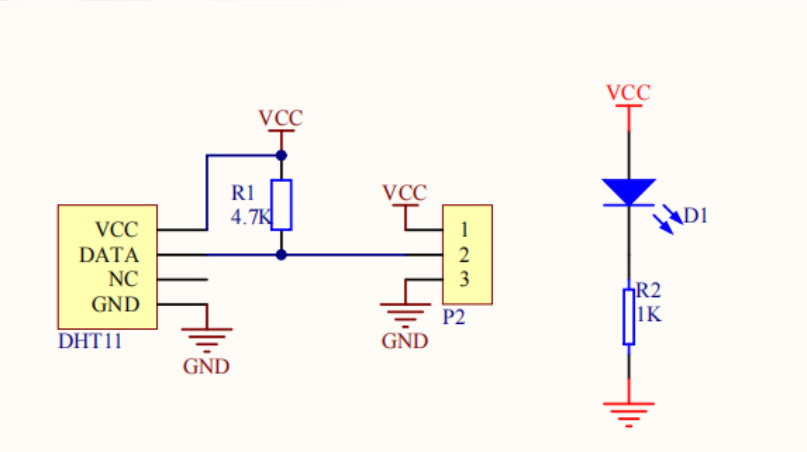

图 1.1 DHT11 模块原理图

VCC：通常连接到电源引脚（比如 3.3V 或 5V），提供传感器所需的电源。

GND：连接到地线或电源的地线，完成电路的闭环。

DATA：用于传输数字信号给单片机或微控制器，传输温度和湿度数据的引脚。

NC：未使用的引脚，不连接到任何功能。

VCC、GND、DATA 这三个引脚是 DHT11 传感器模块与其他电子设备，进行连接时需要用到的主要引脚。通过这些引脚，传感器可以获取电源、接地并与其他设备进行数据通信。

1.2.2 BH1750 光照强度传感器

BH1750 光照传感器是一种数字式传感器，用于准确测量环境中的光照强度。其工作基于光敏电阻的原理，能够对不同光照条件下的光线进行敏感检测。

该传感器内部结构包括光敏元件和模数转换器（ADC）。光敏元件对周围环境光产生响应，其电阻值随光照强度变化而变化。通过内部的 ADC，光敏元件感知到的光照强度被转换成数字信号，进而输出光照数据。

BH1750 传感器通常通过 I2C 或 SPI 等数字接口与微控制器连接，以进行数据通信和光照数据读取。微控制器发送命令以初始化传感器并获取光照数据。数据的解析通常需要根据传感器规格表进行，以将数字化的数值转换为实际的光照强度数值。

其模块化设计和易于集成的特性，使得 BH1750 传感器在自动照明系统、环境光感应调节等领域得到广泛应用。它能够快速、准确地测量光照强度，并与数字系统进行高效的通信与集成。本系统中使用的 BH1750 使用的数字接口是 I2C 总线，其原理如图 1.2 所示。

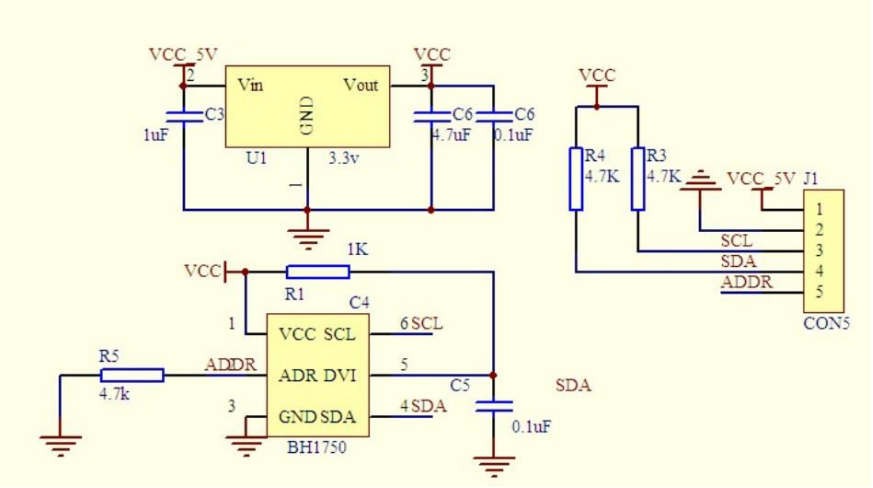

图 1.2 BH1750 光照强度传感器模块原理图

VCC：连接到电源引脚（通常为 3.3V 或 5V），为传感器提供电源。

GND：连接到地线，完成电路的闭环。

SDA：串行数据线，用于与 MCU 进行数据传输的双向串行数据线。

SCL：时钟线，与 SDA 线一同工作，用于与主控制器进行同步的时钟信号传输。

ADDR：地址选择线，用于选择传感器的 I2C 地址，允许连接多个传感器到同一个 I2C 总线上。通常通过将 ADDR 引脚接地或电源线来设置不同的地址。

1.2.3 蜂鸣器模块

三脚低电平触发蜂鸣器是一种用于产生声音信号的电子元件。其设计简洁，由三个引脚组成，分别是电源引脚（VCC）、地线引脚（GND）以及信号引脚（Signal）。

蜂鸣器的工作原理基于电磁感应原理。当在信号引脚提供低电平信号时，蜂鸣器内部的线圈或震动片会受到电流的作用，形成一个磁场。这个磁场将导致蜂鸣器内部的振动片或压电陶瓷振动，从而产生声音。

在控制方面，通过给信号引脚提供低电平信号，来触发蜂鸣器发声。反之，

去除这个低电平信号或将信号引脚断开连接，将会使蜂鸣器停止发声。

因其简便的控制特性，这种类型的蜂鸣器被广泛应用于各种电子设备中，用于产生声音信号，例如警报、提醒或指示特定状态的变化。其易于集成和控制的特性使其成为许多嵌入式系统中重要的声音输出设备。

1.2.4 SSD1306OLED 模块

四脚 I2C OLED 模块是一种采用 I2C 通信协议的 OLED 显示屏，其引脚构成包括 VCC、GND、SCL 和 SDA。

VCC（电源）：连接到电源引脚，供应 OLED 显示屏所需的电力。

GND（地线）：连接到地线，确保电路完整性。

SCL 和 SDA：SCL 是串行时钟线，SDA 是串行数据线。它们是 I2C 总线上的两根数据线，负责 OLED 模块与微控制器之间的双向通信。

OLED 模块通过 I2C 通信协议与 MCU 进行数据传输和控制。该协议支持多设备在同一总线上进行通信。SCL 和 SDA 引脚负责在 OLED 模块和微控制器之间传输数据和时钟信号。

VCC 和 GND 引脚提供必要的电源支持，确保 OLED 显示屏正常运行。微控制器通过 I2C 总线向 OLED 模块发送命令和数据，控制显示内容、亮度和其他显示参数。OLED 模块接收到的数据被转换为像素点，并在屏幕上呈现相应的图像、文本或图形。

在 SSD1306OLED 模块上，用于图像显示的 RAM，GDDRAM 是位映射静态 RAM，大小为 128x64 位。GDDRAM 分为 8 页（PAGE0~PAGE7），每页内 1 个 SEG 对应 1Byte 数据，一页由 128 Byte 组成。一帧显示数据为 1024 Byte（1KB）。即屏幕每 8 行像素点（8*PIXEL）记为一页（PAGE），64 行即为 8 页，则屏幕变为 128 列（ROW）8 页（PAGE），若要显示整个屏幕，则需要 128*8 个 1 字节数。图 1.3 是其内部结构。

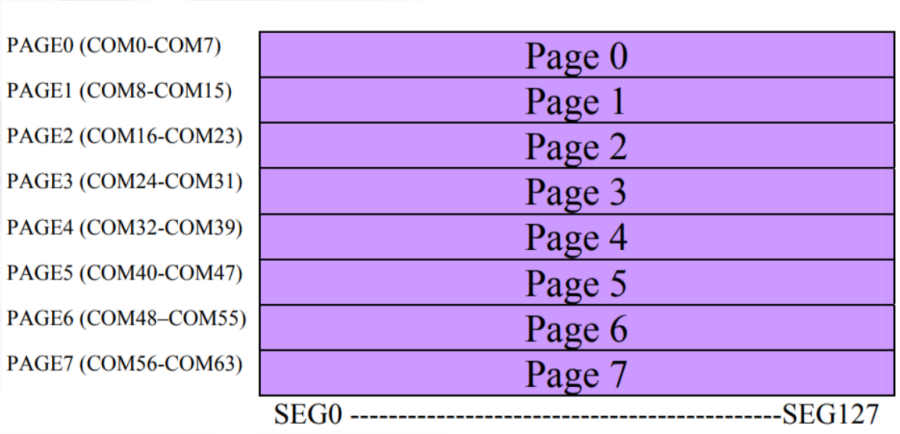

图 1.3 GDDRAM(Graphic Display Data RAM)内部结构

1.2.5 esp8266-01s 无线模块

ESP8266-01S 是 ESP8266 系列的一款芯片，它集成了 WiFi 模块，能够支持 802.11b/g/n 标准，实现无线网络连接。通过 WiFi 连接，它能够接收和发送数据，为各种智能设备和传感器提供无线连接的能力。

作为嵌入式系统，ESP8266-01S 内部整合了处理器和内存等核心功能，使其具备独立运行和执行代码的能力。这让它可以在无需外部设备支持的情况下完成特定的任务。

ESP8266-01S 通过串行通信接口进行数据的接收和发送。这种通信方式用于与其他设备进行数据交互，实现数据传输。

使用支持 ESP8266 的集成开发环境编写程序并上传至 ESP8266-01S 芯片中。通过编写的代码，可以实现 WiFi 连接控制、数据传输管理、传感器数据接收等功能。ESP8266-01S 还可以通过 AT 指令进行控制和配置，无需编写程序。通过串行通信，可以发送特定格式的 AT 指令给 ESP8266-01S，实现各种功能和配置。在本系统中，使用 STM32 作为主控芯片，通过串口发送 AT 指令给 esp8266-01s 连接 WiFi 并进行数据上传和接收。

以下是 esp8266-01s 的管脚功能图。

脚序	名称	功能说明
1	GND	接地
2	IO2	GPIO2/UART1_TXD
3	IO0	GPIO0；下载模式：外部拉低；运行模式：悬空或者外部拉高
4	RXD	UART0_RXD/GPIO3
5	TXD	UART0_TXD/GPIO1
6	EN	芯片使能端，高电平有效
7	RST	复位
8	VCC	3.3V 供电（VDD）；外部供电电源输出电流建议在 500mA 以上

图 1.4 esp8266-01s 管脚功能定义

本系统中使用到了 VCC、GND、RXD、TXD 四个管脚，其中 VCC、GND 为 esp8266-01s 供电，RXD、TXD 管脚与单片机进行数据传输。

1.2.6 LED 灯电路

LED 是一种半导体器件，其发光基于电子在半导体材料中重新组合时释放能量的原理。LED 具有两个引脚，一个是正极（阳极），另一个是负极（阴极）。当正极施加正电压、负极施加负电压时，电流从正极流向负极，LED 便会发光。

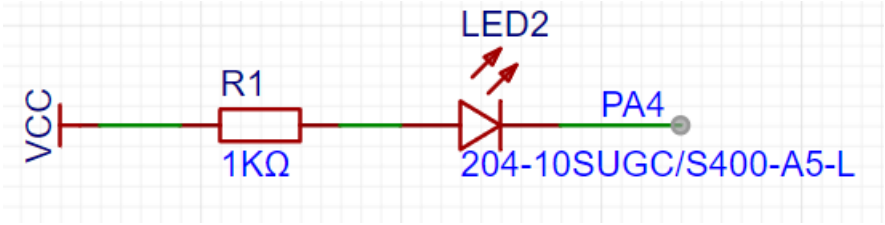

图 1.5 LED 控制电路

加电阻防止电流过大烧毁芯片，当控制引脚输出低电平时，LED 被点亮。

1.2.7 MQTT 协议原理

MQTT（Message Queuing Telemetry Transport）是一种轻量级、基于发布/订阅模式的消息传输协议，被广泛应用于面向物联网设备和传感器网络的通信[4]。该协协议设计用于低带宽、高延迟或不可靠网络环境中的设备间通信，其基本原理依赖于中介服务器（Broker）和发布者-订阅者模型。

MQTT 协议的工作流程由发布者将消息发布到特定主题，而订阅者则根据主题订阅感兴趣的消息构成。这些消息通过 Broker 中介进行传输。主题（Topic）作为消息的标识符，允许订阅者选择接收特定主题下的消息。协议支持多种质量服务级别（QoS），从保证最多传递一次的 0 级到确保仅传递一次的 2 级，以满足不同场景下的可靠性需求。

MQTT 的设计注重轻量级和资源效率，其协议头相对较小，适应于受限的网络带宽和资源环境。设备与 Broker 之间维持持久连接，以保证设备在需要时可以随时发送或接收消息。

在实际应用中，MQTT 被广泛应用于物联网、传感器网络以及需要高效、可靠通信的移动设备领域。其高度灵活性和可靠性使得其成为解决低带宽或不稳定网络条件下设备间通信需求的一种有效协议

1.3 方案论证及可行性分析

1.3.1 单片机的选择

在本次课程设计中，我们需要建立一个控制中心，以单片机为核心来控制各种传感器数据的收集，以及开关 LED 灯和电机等设备。因此，在这个系统设计中，选择一款合适的单片机非常关键，它需要具备高效的计算和处理能力，同时支持丰富的外设集成，以便于与各种传感器和设备进行连接和通信。只有选择了适合的单片机，才能保证整个系统的性能和可靠性。

STM32F103C8T6 是一款由意法半导体生产的 32 位 ARM Cortex-M3 内核微控制器，相比于深圳市三个科技有限公司生产的 8 位 STC89C51 单片机，它具有更高的处理能力、更大的存储容量以及丰富的外设集成。此外，STM32F103C8T6 拥有强大的开发生态系统和先进的架构和特性，使其在处理复杂算法、存储大型项目和实现各种功能需求方面表现出色。因此本次采用 STM32F103C8T6 作为本次系统的主控制器[6]。

1.3.2 无线模块的选择

ESP8266-01s 作为无线模块的选择具有低廉的价格、强大的 Wi-Fi 功能和灵活

的串口通信，通过简单的 AT 指令即可实现配置和控制，支持固件升级和兼容多种开发平台和编程语言。此外，丰富的资源和活跃的社区为开发者提供了便利，使他们能够快速上手并加速应用程序的开发。综合而言，ESP8266-01s 是一款性价比高、功能强大且易于使用的无线模块，非常适合构建无线控制系统和物联网应用。它有如下优点：

①低成本：ESP8266-01s 是一款经济实惠的无线模块，价格相对较低，适合在成本敏感的项目中使用。

②强大的功能：尽管 ESP8266-01s 是一个小型模块，但它内置了强大的 Wi-Fi 功能，可以快速连接到无线网络，并支持 TCP/IP 协议栈，使其能够与互联网进行通信。

③灵活性：ESP8266-01s 可以通过串口与单片机或其他设备进行通信，使用简单的 AT 指令即可实现配置和控制。它还支持固件升级，可以根据需要进行功能扩展和更新。

④资源丰富：ESP8266 社区非常活跃，有大量的文档、示例代码和库可供参考和使用。这使得开发者能够快速上手，解决问题，并且可以利用已有资源加速开发过程。

⑤良好的兼容性：ESP8266-01s 可以与各种主流开发平台和编程语言配合使用，如 Arduino、MicroPython 等。这意味着开发者可以使用他们熟悉的工具和语言来开发基于 ESP8266-01s 的应用程序。

1.3.3 传输协议的选择

MQTT 协议是一种轻量级的发布/订阅式消息传输协议，具有简单和轻量级、低功耗和资源占用少、异步通信和发布/订阅模型、可靠性和持久性、灵活性和扩展性等优点。这使得它非常适合在带宽有限或网络连接不稳定的环境中使用，同时也适用于各种设备和平台，包括嵌入式系统和传感器等资源受限设备。而 HTTP 是基于请求/响应模式的协议，客户端需要向服务器发送。基于这些，MQTT 协议广泛应用于物联网、传感器网络和即时通讯等领域，非常适合作为本次设计的传输协议[4]。

1.3.4 方案经济成本分析

材料清单：usb 转 ttl、光照传感器 bh1750、温湿度传感器 DHT11、STM32F103C8T6、ESP8266-01S、LED 灯、按键、0.96 寸 OLED 显示屏、stlink 烧写器。

预估成本大约 60，故此方案造价低廉，符合低经济成本预期。

1.4 总体设计

总体设计如图所示：

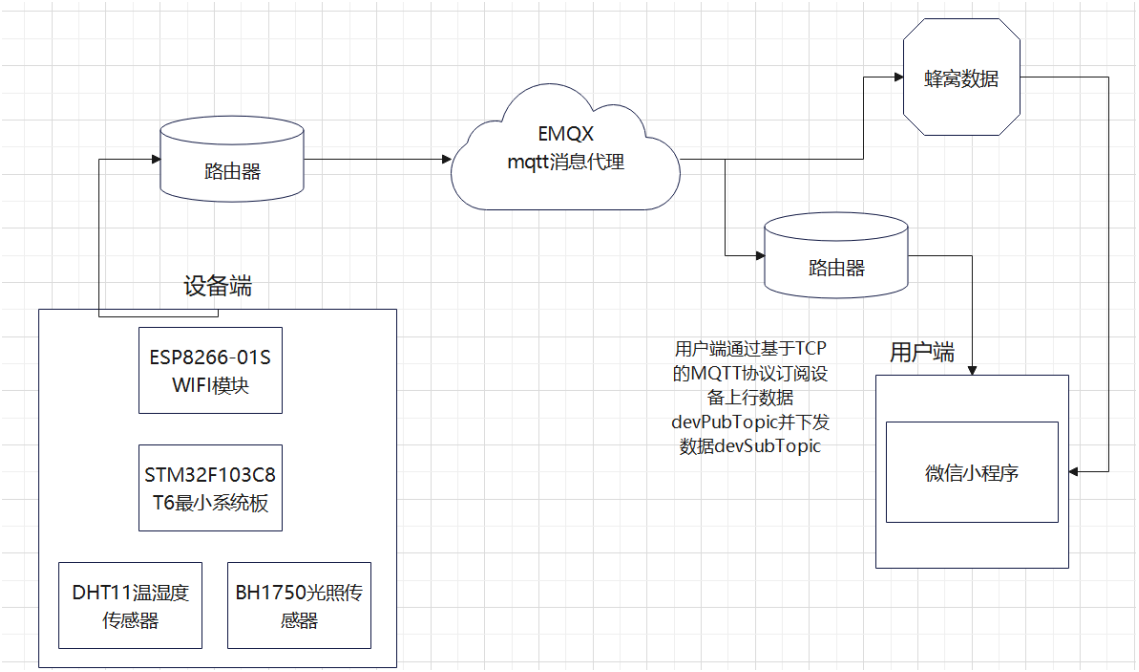

图 1.6 系统总体设计图

1.5 软件设计

1.5.1 微信小程序设计

在本实验中，通过微信小程序作为可视化界面，来观察来自服务器的数据下发，和控制数据上传。分为以下设计部分：

(1)环境的安装和开发工具的安装

本次微信小程序使用的开发工具为 微信开发者工具，并使用了 VUE 脚手架，安装的环境为 NODE。

(2)主页面图形搭建

整体采用蓝白配色，界面整洁清晰，导航栏背景选用纯白色，标题为智能家居

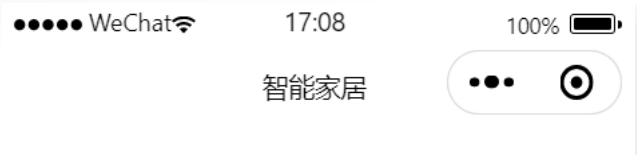

图 1.7 微信小程序导航栏部分

然后是微信小程序实时天气部分，需要获取用户当前地址，来更新当前的地区位置和天气信息

图 1.8 微信小程序天气部分

使用和风天气接口 api，引用官方文档中的示例 js 代码，选择自己需要的数据部分用 mustache 语法与标签绑定数据，即可正常显示出数据。定位默认获取的是登录微信的位置信息，当没有正确登录时，会报错提示未获取定位信息

接着是微信小程序数据部分，采用方块设计，一个盒子一个功能，更直观并且有利于后续添加功能，每个盒子里的图标通过 Iconfont 阿里巴巴矢量图标库获得

图 1.9 微信小程序数据部分

在相应的模块部分，其数据与平台收到的数据进行绑定，实现实时数据的更新。

(3)个人主页界面

个人主页界面，顶部是用户登录获取用户的头像和名字，中间为轮播图可以放照片用于装饰，可以播放图片

图 1.9 微信小程序个人主页界面（1）

采用的微信小程序旧版登录方式，推荐使用 wx.getUserProfile 获取用户信息，开发者每次通过该接口获取用户个人信息均需用户确认。但是该开放接口的具体使用过程中登录有时候会登录不上，此为待优化的地方

```
getuser: function(e) {
  wx.getUserProfile({
    desc: '用于完善会员资料',
    success: (res) => {
      this.setData({
        name: res.userInfo.nickName,
        pic: res.userInfo.avatarUrl,
        showButton: false, // 成功获取用户信息后隐藏按钮
      })
    }
  })
},
```

图 1.10 wx.getUserProfile 接口使用图

创建一个获取用户信息的 button 并绑定这个 getuser 的方法，当点击获取用户信息的时候可以弹出用户信息框

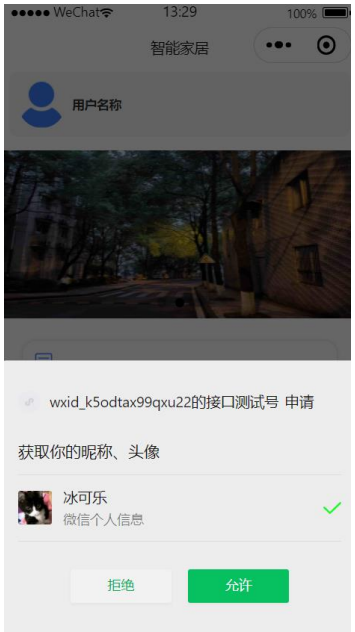

图 1.11 用户获取信息弹窗示意图

当用户允许登录后即可进入后续状态

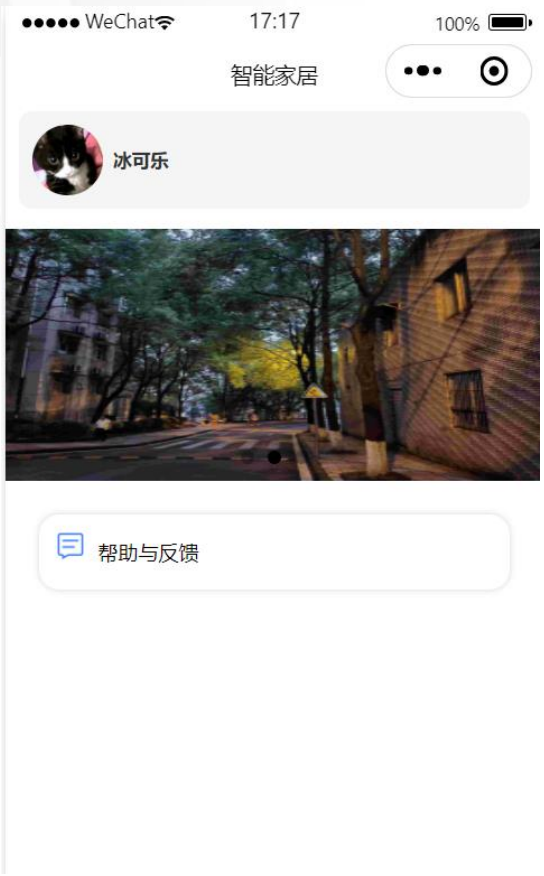

图 1.14 个人主页界面

将“帮助与反馈”这个盒子也绑定一个点击事件，点击“帮助与反馈”后可以弹出开发者信息，或者提供产品说明

图 1.15 帮助与反馈消息提示框

至此便是我小程序部分的设计。

1.5.2MQTT 传输协议程序设计

首先连接 MQTT 服务器，并在 STM32F103C8T6 端利用 C 语言 MQTT 客户端库实现连接、订阅和发布传感器数据等功能；同时在微信小程序端使用 JavaScript MQTT 客户端库实现连接、订阅 STM32F103C8T6 发布的主题和接收数据，保证两端发布主题和接收主题相同。单片机通过订阅的主题，在云平台服务器中获取微信小程序发送的消息，同时，微信小程序以同样的方式获取单片机发送的消息

1.5.3ESP8266 端设计

使用 ESP8266 模块完成 WIFI 通讯，整个 ESP8266 的程序设计流程图如图:

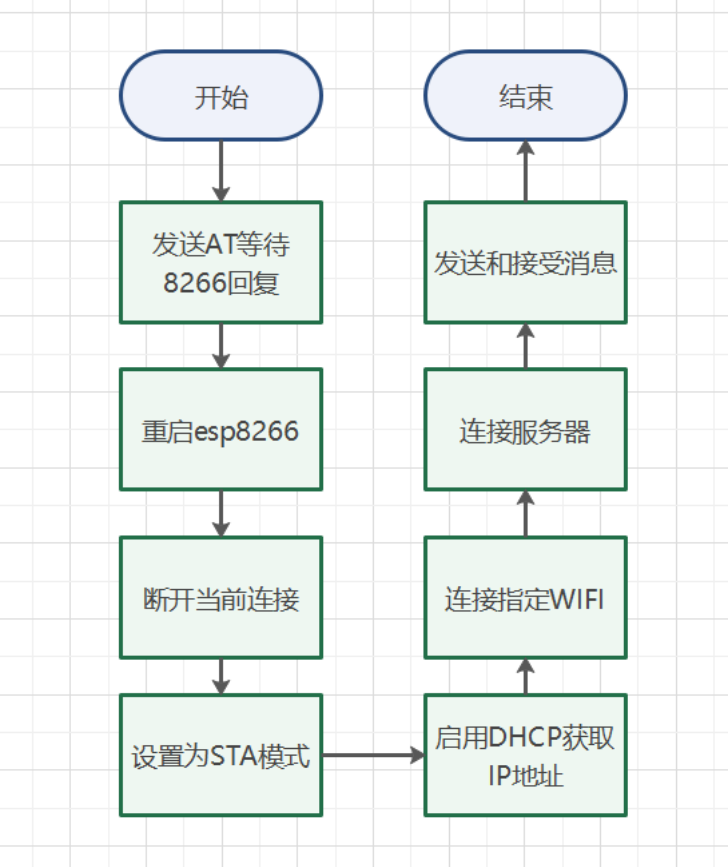

图 1.16 ESP8266 程序设计流程图

首先确保 ESP8266 烧录了 MQTT 固件库，然后发送 AT 指令"AT\r\n"，等待 ESP8266 回复"OK"，重复发送直到收到回复。然后重启 ESP8266 保证 ESP8266 可以清除 ESP8266 的状态和缓存，以确保其能够正确地执行后续的指令。有时，

ESP8266 可能会出现异常状态或缓存堵塞等问题，这些问题可能会导致 ESP8266 无法正常工作。通过重启 ESP8266，可以清除这些问题并恢复 ESP8266 的正常工作状态。然后通过设置 STA 模式，ESP8266 可以连接到路由器或者接入点（AP），从而能够访问互联网或者局域网中的其他设备，接着连接 WIFI 和服务器，以进行后续数据上云的操作等。

1.5.4 主程序的设计

主程序中首先主要定义各类变量，并且定义接受主题和发送主题，然后初始化各个硬件，通过 DHT11 传感器读取温湿度数据，通过 BH1750 传感器获取光照强度数据，并通过 USART1 和 OLED 显示屏显示相关信息。同时，该应用程序还通过 ESP8266 模块连接到 MQTT 服务器，实现与云端的通信。

主要功能包括：

①初始化各个外设，如 USART、OLED、LED、按键、定时器、温湿度传感器、光照传感器等。

②初始化 ESP8266 模块并连接到 OneNet 平台。

③定时读取温湿度和光照强度数据，并更新显示屏上的信息。

④定时将数据发布到 OneNet 平台。

⑤接收来自 OneNet 平台的命令并执行相应操作。

使用定时器，用于控制 LED 灯和蜂鸣器的闪烁，并在 OLED 屏幕上显示温湿度和光照强度。按键实现的功能通过外部中断实现，实现按键亮灯操作和控制蜂鸣器。

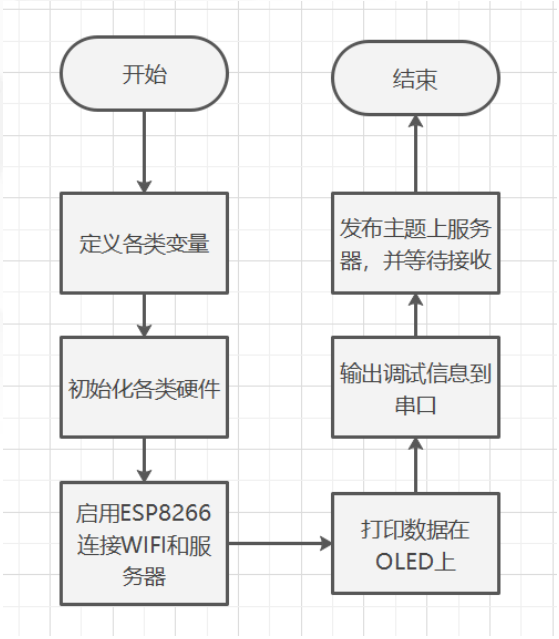

图 1.17 主程序设计流程图

2 设计总结

对于我设计的这部分，我感觉可完善的点还很多，比如小程序的界面还可以美化，还可以增加手动添加设备的功能，然后个人主页中内部的功能还可以丰富一些，以及小程序中的数据还可以实现上微信小程序的云存储。对于数据上云的代码部分，采用的 mqttkit 包和 onenet 的连接示例包，理解实在是有点困难，仅仅是勉强会用的程度。期间由于不熟悉小程序的接口也有区分版本，导致在引入天气接口和引入 mqtt 包的时候处处碰壁，出现许多奇奇怪怪的问题，最后排查很久才解决。

在本次课程设计学习过程中，通过查阅资料、小组讨论、并且查看教学视频完成了本次设计。将我之前感兴趣的前端知识有所应用，这样的兴奋感让我内心激动许久，不可否认的是，我们的方案还存在许多不足，比如传感器太少、功能太少等，作为一个智能家居系统，应该需要有更多的设备，但是碍于时间和经费等原因，我们的方案只得到此结束，如果后续还有机会，我们会继续完善该项目。

总体而言，这次课程设计是一次宝贵的经验，让我深入了解了理论和实践的区分，并提升了自己能力。我将继续努力学习和探索。

致谢

首先感谢学校有让我有次机会去学习到本次设计所涉及的知识和内容，以及让我运用以前所学的知识到实际中，以及感谢授课的老师们，老师教给我的不仅仅是知识，更多是热爱和坚持，让我遇到问题可以坚持不懈地去解决。

其次感谢 B 站的 up 主提供了如此多的教程，让我有所头绪，从而顺利完成此设计，他们专业的知识和经验给我提供了莫大的帮助

还要感谢我的搭档，有着彼此的陪伴，互相勉励，我们分工合作，发挥各自优势从而顺利进行本次设计。

众多帮助和支持让我充满感激，没有你们的支持和鼓励无法完成本次设计，最后再对以上人员致以我深深的谢意和祝福！

感谢走得很慢但一直向前的自己。

参考文献

[1] 谢希仁. 计算机网络(第五版）[M]. 北京: 电子工业出版社, 2008: 349-352.

[2] 姜凤娟.物联网技术在智能家居中的应用分析[J].智能建筑与智慧城市,2023(08):182-184.

[3] 曾海燕,郑鑫,韦燚.基于物联网技术的智能家居控制系统设计研究[J].电子制作,2023,31(01):116-120.

[4] 高亮海,于世龙,赵少坤等.基于 MQTT 协议的嵌入式联网报备考勤系统[J].物联网技术,2023,13(10):75-79.

[5] 向敏等. 单片机原理与工程应用. 北京：电子工业出版社,2021,3.

[6] 肖榆瀞,柯玮翔.基于 STM32 的物联网智能家居控制系统[J].现代计算机,2023,29(10):87-92.

[7] 腾立国.基于物联网的农业温室大棚智能控制系统研究[J].智慧农业导刊,2023,3(23):1-5.

[8] 赵鹏博.基于单片机的火灾报警的设计与实现[J].中国水运(下半月),2023,23(01):41-43.

附件 A 程序

```
#include "led.h"
#include "key.h"
#include "beep.h"
#include "delay.h"
#include "sys.h"
#include "timer.h"
#include "usart.h"
#include "dht11.h"
#include "bh1750.h"
#include "oled.h"
#include "exti.h"
#include "stdio.h"
#include "esp8266.h"
#include "onenet.h"

u8 alarmFlag = 0;//是否报警的标志
u8 alarm_is_free = 10;//报警器是否被手动操作，如果被手动操作即设置为 0
u8 LightFlag = 0;//是否自动亮灯的标志
u8 Light_is_free = 1;



u8 humidityH;   //湿度整数部分
u8 humidityL;   //湿度小数部分
u8 temperatureH;   //温度整数部分
u8 temperatureL;   //温度小数部分
extern char oledBuf[20];
float Light = 0; //光照度
u8 Led_Status = 0;
```

```
char PUB_BUF[256];//上传数据的 buf
const char *devSubTopic[] = {"/iot/4346/sub"};
const char devPubTopic[] = {"/iot/4346/pub"};
u8 ESP8266_INIT_OK = 0;//esp8266 初始化完成标志


int main(void)
{
  unsigned short timeCount = 0;  //发送间隔变量
  unsigned char *dataPtr = NULL;

  Usart1_Init(115200);//debug 串口
      DEBUG_LOG("\r\n");
      DEBUG_LOG("UART1 初始化          [OK]");
       delay_init();          //延时函数初始化
      DEBUG_LOG("延时函数初始化          [OK]");

       delay_ms(500);// 延迟一下等待 oled 识别
       OLED_Init();
       OLED_ColorTurn(0);//0 正常显示，1 反色显示
        OLED_DisplayTurn(0);//0 正常显示 1 屏幕翻转显示
       OLED_Clear();
      DEBUG_LOG("OLED1 初始化          [OK]");
      OLED_Refresh_Line("OLED");

  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);// 设置中断优先级分组 2
      DEBUG_LOG("中断优先初始化          [OK]");
      OLED_Refresh_Line("NVIC");
```

```
    LED_Init();          //初始化与 LED 连接的硬件接口
        DEBUG_LOG("LED 初始化        [OK]");
        OLED_Refresh_Line("Led");
    KEY_Init();              //初始化与按键连接的硬件接口
        DEBUG_LOG("按键初始化          [OK]");
        OLED_Refresh_Line("Key");
    EXTIX_Init();    //外部中断初始化
        DEBUG_LOG("外部中断初始化          [OK]");
        OLED_Refresh_Line("EXIT");
    BEEP_Init();
        DEBUG_LOG("蜂鸣器初始化        [OK]");
        OLED_Refresh_Line("Beep");
    DHT11_Init();
        DEBUG_LOG("DHT11 初始化        [OK]");
        OLED_Refresh_Line("DHT11");
BH1750_Init();
        DEBUG_LOG("BH1750 初始化        [OK]");
        OLED_Refresh_Line("BH1750");
    Usart2_Init(115200);//stm32-8266 通讯串口
        DEBUG_LOG("UART2 初始化        [OK]");
        OLED_Refresh_Line("Uart2");

        DEBUG_LOG("硬件初始化          [OK]");

    delay_ms(1000);

    DEBUG_LOG("初始化 ESP8266 WIFI 模块...");
    if(!ESP8266_INIT_OK){
        OLED_Clear();
        OLED_ShowString(0,0,(u8*)"WiFi",16,1);
```

```
        OLED_ShowChinese(32,0,8,16,1);//连
        OLED_ShowChinese(48,0,9,16,1);//接
        OLED_ShowChinese(64,0,10,16,1);//中
        OLED_ShowString(80,0,(u8*)"...",16,1);

        OLED_Refresh();
    }
    ESP8266_Init();                    //初始化 ESP8266

    OLED_Clear();
        OLED_ShowChinese(0,0,4,16,1);//服
        OLED_ShowChinese(16,0,5,16,1);//务
        OLED_ShowChinese(32,0,6,16,1);//器
        OLED_ShowChinese(48,0,8,16,1);//连
        OLED_ShowChinese(64,0,9,16,1);//接
        OLED_ShowChinese(80,0,10,16,1);//中
        OLED_ShowString(96,0,(u8*)"...",16,1);
    OLED_Refresh();

    while(OneNet_DevLink()){//接入 OneNET
        delay_ms(500);
    }

    OLED_Clear();

    TIM2_Int_Init(4999,7199);
    TIM3_Int_Init(2499,7199);

//  BEEP = 0;//鸣叫提示接入成功
//  delay_ms(250);
```

```
// BEEP = 1;

    OneNet_Subscribe(devSubTopic, 1);

    while(1)
    {
        if(timeCount % 40 == 0)//1000ms / 25 = 40 一秒执行一次
        {
            /********** 温湿度传感器获取数据**************/
            DHT11_Read_Data(&humidityH,&humidityL,&temperatureH,&temperatureL);

            /********** 光照度传感器获取数据**************/
            if (!i2c_CheckDevice(BH1750_Addr))
            {
                Light = LIght_Intensity();
            }

            /********** 读取 LED0 的状态 **************/
            Led_Status = GPIO_ReadInputDataBit(GPIOA,GPIO_Pin_4);

            /********** 报警逻辑 **************/
            if(alarm_is_free == 10)//报警器控制权是否空闲 alarm_is_free == 10 初始值为 10
            {
                if((humidityH < 80) && (temperatureH < 30) && (Light < 1000))alarmFlag = 0;
                else alarmFlag = 1;
            }
            if(alarm_is_free < 10)alarm_is_free++;
            /********** 光照太低自动开灯 **************/
```

```
            if(Light_is_free)
            {
                if(Light < 20) LightFlag = 1;
                else LightFlag = 0;
            }

            /********** 输出调试信息 **********/
            DEBUG_LOG(" | 湿度：%d.%d C | 温度：%d.%d %% | 光照度：%.1f lx | 指示灯：%s | 报警器：%s |
",humidityH,humidityL,temperatureH,temperatureL,Light,Led_Status?"关闭":"【启动】",alarmFlag?"【启动】":"停止");
        }
        if(++timeCount >= 100)   // 5000ms / 25 = 200 发送间隔 5000ms
        {
            Led_Status = GPIO_ReadInputDataBit(GPIOA,GPIO_Pin_4);//读取 LED0 的状态
            DEBUG_LOG("======================");
            DEBUG_LOG("发布数据 ----- OneNet_Publish");

    sprintf(PUB_BUF,"{\"Hum\":%d.%d,\"Temp\":%d.%d,\"Light\":%.1f,\"LED\":%d,\"Beep\":%d}",

    humidityH,humidityL,temperatureH,temperatureL,Light,Led_Status?0:1,alarmFlag);
            OneNet_Publish(devPubTopic, PUB_BUF);
            DEBUG_LOG("======================");
            timeCount = 0;
            ESP8266_Clear();
        }
```

```
            dataPtr = ESP8266_GetIPD(3);
            if(dataPtr != NULL)
                OneNet_RevPro(dataPtr);
            delay_ms(10);
        }
    }
```