山西大学
SHANXI UNIVERSITY

2023 届 硕 士 学 位 论 文

基于 STM32 的智能家居控制系统的设计

作者姓名　王森民

指导教师　樊民革　讲师

学科专业　电子信息

研究方向　嵌入式系统应用

培养单位　自动化与软件学院

学习年限　2020 年 9 月至 2023 年 11 月

二〇二三年十一月

山西大学

2023 届硕士学位论文

基于 STM32 的智能家居控制系统的设计

作者姓名　王森民

指导教师　樊民革　讲师

学科专业　电子信息

研究方向　嵌入式系统应用

培养单位　自动化与软件学院

学习年限　2020 年 9 月至 2023 年 11 月

二〇二三年十一月

Thesis for Master's degree，Shanxi University，2023

Design of STM32-based smart home control system

Student Name Sen-min Wang

Supervisor Min-ge Fan

Major Electronic Information

Specialty Embedded System Applications

Department School of Automation and Software Engineering

Research Duration 2020.09-2023.11

November，2023

摘　要

随着科技的发展，现代人们开始追求高品质的精神生活。智能家居作为创新技术的融合，为消费者带来便利和智能化体验，具有市场潜力。本研究旨在设计一个基于 STM32 的智能家居控制系统，实现对家居环境的监测和设备的远程控制。该系统使用 WIFI 通信技术，用户可以通过手机 APP 进行实时控制和管理，提高家居的高效性和便捷性。

首先，本文阐述了研究背景研究意义以及研究现状和发展趋势，明确了智能家居研究在电子、自动化和通讯领域的交叉性质。通过对国内外智能家居发展的深入剖析，总结出该控制系统的核心需求：即实时控制家电和多媒体设备、通过传感器和摄像头进行数据采集以及实施实时监控和预警。针对上述需求，详细规划并设计了相关的系统功能。

其次本控制系统按照模块化由 WIFI 无线通信技术绑定云平台，STM32 硬件系统，基于 Android 平台的 APP 这几个主要模块组成。云平台利用 ONENET 进行串口调试进行配置，绑定设备 ID，WIFI 通信模块绑定到了 ONENET 云平台上;硬件系统是由单片机最小系统做控制中心，温湿度检测模块，光强检测模块，烟雾浓度检测模块，语音识别模块，继电器控制模块等组成 STM32 的硬件系统，其中控制系统的软件是利用 keil5 软件进行编写调试;APP 的开发是在 Android Studio 平台中完成的，采用 Java 语言开发完成，完成系统需求。

最后经过系统的硬件测试与软件测试。证明了该控制系统所实现的功能。可以通过手机和语音识别来控制窗帘，电视，空调和 LED 灯的开闭，针对温湿度模块，光照模块，烟雾监测模块达到阈值风扇会打开，也可以通过摄像头在线监测家居环境的情况。通过以上的测试，发现该控制系统符合日常生活需求，具有实用性和高度的稳定性。

关键词：STM32 单片机；智能家居；WIFI 无线通信；物联网

ABSTRACT

With the development of technology, modern people pursue a high-quality spiritual life. As a fusion of innovative technologies, smart homes bring convenience and intelligent experiences to consumers, and have market potential. This study aims to design a smart home control system based on STM32, achieving monitoring of the home environment and remote control of devices. The system uses WIFI communication technology, allowing users to control and manage in real-time through a mobile app, improving the efficiency and convenience of their homes.

Firstly, this article elaborates on the research background, significance, current status, and development trends, and clarifies the interdisciplinary nature of smart home research in the fields of electronics, automation, and communication.Through in-depth analysis of the development of smart homes both domestically and internationally, the core requirements of this control system are summarized: real-time control of household appliances and multimedia devices, data collection through sensors and cameras, and implementation of real-time monitoring and early warning. In response to the above requirements, we have planned and designed the relevant system functions in detail.

Secondly, this control system is modular and consists of several main modules: WIFI wireless communication technology bound to the cloud platform, STM32 hardware system, and Android platform based APP. The cloud platform utilizes ONENET for serial debugging and configuration, binds device IDs, and binds WIFI communication modules to the ONENET cloud platform;The hardware system of STM32 consists of a microcontroller minimum system as the control center, a temperature and humidity detection module, a light intensity detection module, a smoke concentration detection module, a speech recognition module, a relay control module, etc. The software of the control system is written and debugged using Keil5 software; The development of the app was completed on the Android Studio platform using the Java language to meet system requirements.

Finally, the system underwent hardware and software testing. Proved the functionality achieved by the control system. You can control the opening and closing of curtains, TV, air conditioning, and LED lights through mobile phones and voice

recognition. For temperature and humidity modules, lighting modules, and smoke monitoring modules, the fan will turn on when reaching the threshold. You can also monitor the situation of the home environment online through a camera. Through the above tests, it was found that the control system meets the needs of daily life, has practicality and high stability.

Key words: STM32 single chip; smart home; WIFI wireless communication; Internet of Things

目　录

1 绪论

1.1 研究背景

在全球经济迅猛发展和人类文明不断向前迈进的时代背景下，人们的物质生活得到显著改善[1]。随着生活品质的逐步提高，人们对家居的舒适度和科技含量都有了更高的追求。智能家居成为了新的趋势，人们不仅希望家中的设备能够相互连接、自动化操作，更期待通过技术的力量，创造一个更为便捷、环保和人性化的居住环境。智能家居作为现代家庭的重要组成部分，正逐渐走向人们生活，其中智能家居产品作为科技时代的重要象征之一，利用互联网进行家居间即时通讯成为了一个方向。智能家居通过将各种物联网技术与家居设施相融合，实现了家居设备的自动化、远程控制以及智能化管理，为人们带来了前所未有的便利和舒适。智能家居的研究旨在通过整合信息技术、通信技术和人工智能，实现家居环境的智能化，为居住者提供更加智能、高效、安全的居住体验[2]。

在 20 世纪 70 年代关于智能家居的研究，人们已经开始尝试将计算机技术应用于家居控制系统中，以提高生活的舒适性和便利性。美国引领提出了家庭智能系统的概念，并在其他地区如欧洲和日本等国家广泛推广，所取得的积极进展令人瞩目。在发达国家，许多公司纷纷积累了大量先进技术，并推出了智能家居的早些版本[3]。智能家居的发展让人们从繁忙的家务生活中得到解脱，然而这些产品往往针对高收入人群，价格普遍偏高，且受到时代的局限性，其高昂的成本与限制了智能家居的普及，难以适应大众群体的购买力。随着计算机、通信和传感器技术的不断进步，智能家居逐渐展现出巨大的潜力。

进入 90 年代末期，国内对智能家居领域的研究取得了显著的进展。在信息技术领域的快速发展推动下，智能家居得以实现更加智能化和互联化。部分社区和家庭早已开始引入智能家居系统，显现出这一领域的潜在前景。物联网技术的兴起，使得各种家居设备能够通过互联网相互通信，实现集中控制和远程操作。传感器技术的提升，使得智能家居能够更加准确地感知环境变化，从而自动调节温度、照明、安防等系统，提升居住体验的同时也实现了能源的有效利用。

人工智能在智能家居中的应用也是推动研究的重要因素之一。机器学习和数据分析使得智能家居系统能够逐渐适应居住者的习惯和需求，实现个性化的智能化控

制。语音助手技术的兴起，使得居住者可以通过语音指令控制家居设备，进一步提升了用户体验。

智能家居的研究涉及多个学科领域，包括计算机科学、电子工程、通信技术、人工智能等[4]。研究人员需要解决诸如设备互操作性、数据隐私保护、系统安全性等一系列技术难题。此外，智能家居的推广还需要考虑标准化、成本降低、用户接受度等社会和市场因素。

智能家居的研究和发展不仅仅关乎技术，也涉及到人们对于生活方式的重新思考。随着智能家居的普及，人们逐渐习惯了通过手机或其他智能终端来控制家居设备，这也在某种程度上改变了人们的生活习惯和社交方式。

如图 1-1 所示，中国智能家居市场呈现出稳健的增长态势。作为物联网、云计算和人工智能等前沿技术的实际应用，智能家居汇集了多个技术领域的力量。得益于国家政策的鼎力扶持，众多企业纷纷加大了对智能家居研发的投资，推出了一系列创新产品[5]。这不仅推动了市场规模的持续扩大，也为消费者带来了更为智能和便捷的生活体验。预计在未来，随着技术的进一步成熟和市场的深入拓展，智能家居将会成为家庭生活的标配，市场潜力巨大。

图 1-1 中国智能家居市场行业规模

Fig. 1-1 Scale of smart home market in China

因此，智能家居作为信息技术和家居生活的结合，其研究背景可以追溯到几十年前的尝试，但如今得以蓬勃发展。随着技术的不断进步和创新，智能家居必将在未来继续引领家居生活的发展方向，为人们创造更加便利、舒适的居住环境[6]。不同文化和地域的逐渐融合也进一步鼓励消费者更广泛地接受智能家居产品，满足多样化的需求。

1.2 研究意义

目前智能家庭的控制系统正处在一个不断发展的时期，使用范围和领域都相对有限，而且所面对的问题也是对智能家庭的发展产生了直接的制约，有着突出的特点：随着科技的飞速发展，智能家居作为现代生活的一个重要方面，正逐渐引起广泛的关注和研究。智能家居不仅仅是一种科技的体现，更是对于提升生活质量、优化资源利用以及推动社会进步的重要手段[7]。以下将从便利性、能源效率、安全性和可持续发展等方面，探讨智能家居研究的深远意义。

便利性：智能家居技术的引入使得人们的生活更加便捷。通过智能手机或其他终端设备，居住者可以远程控制家居设备，实现智能照明、智能窗帘、智能空调等功能。这种便利性不仅提升家居的实用性，还为居住者创造了更加舒适的生活环境。

能源效率：智能家居系统可以通过感应器和自动化控制，实现对家居设备的智能调控，从而减少资源的浪费。例如智能温控系统可以根据居住者的习惯和环境变化，自动调节温度，从而降低能源消耗。这有助于节约能源、降低能源成本，为可持续发展贡献一份力量。

安全性：智能家居系统可以集成安全监测技术，如智能摄像头、烟雾报警器、入侵检测器等，实现实时监控和预警。居住者可以通过手机随时查看家居情况，一旦发现异常，系统会及时发出警报。这种安全性的提升对于保护家庭成员和财产安全具有重要意义。

可持续发展：智能家居的能源管理和资源利用优化有助于实现可持续发展目标。通过智能化管理手段，可以减少不必要的能源浪费，降低碳排放，从而对环境产生积极影响。此外，智能家居还能鼓励人们养成节约能源和资源的生活习惯，从而为未来的可持续发展做出贡献。

智能家居的研究意义深远而多元，不只是带来了一个更为智能和便利的居住环境，更是在推动技术进步、提高资源使用效率和环境可持续性等领域中发挥了积极

作用[8]。通过智能家居的应用，能够更好地适应快速变化的现代生活，促进社会进步和可持续发展的目标。

1.3 研究现状及发展趋势

1.3.1 国内外研究现状

国外对于智能家居的研究相对国内较早，从最初的尝试到如今的普及应用，智能家居的发展历程充满了技术创新、市场竞争以及社会适应的过程。尽管如此，这一时期的探索为智能家居的后续发展奠定了基础。故对于智能家居的发展，可以概况为五个阶段。

（1）初期探索（20 世纪 70-80 年代）：智能家居的发展可以追溯到 20 世纪 70 年代。当时，研究者开始尝试将计算机技术应用于家居控制系统中，以提高生活的便利性和舒适性。国内在 20 世纪 80 年代末期，开始涌现出一些尝试将计算机技术应用于家居控制的项目。然而，那个时候的技术还非常有限，成本高昂，且安装复杂，限制了智能家居的推广。

（2）技术进步与商业化（20 世纪 90 年代-2000 年代）：随着计算机、通信和传感器技术的进步，智能家居逐渐迎来了技术的飞跃。在 20 世纪 90 年代，物联网技术开始应用于智能家居，使得家居设备可以通过互联网连接和通信。随着国际智能家居技术的逐渐成熟，国内也开始引进相关技术。中国一些企业也开始涉足智能家居领域，尝试开发智能化家居产品。这一时期，一些家庭自动化系统开始出现，如智能照明、温控、安防等[9]。然而，由于技术限制和高昂的成本和市场认知度不高，智能家居仍未实现大规模的商业化应用。

（3）物联网时代的崛起（2010 年代）：进入 21 世纪，物联网技术的崛起为智能家居的发展提供了巨大的机遇。中国政府开始重视智能家居的发展，出台了一系列政策支持措施。这为智能家居的快速发展提供了动力。国内企业开始大力投入研发，推出了一系列智能家居产品。物联网技术的普及使得智能家居在中国逐渐迎来商业化应用。智能家居设备逐渐变得更加智能化、互联化。智能手机的普及使得用户可以随时随地通过移动应用控制家居设备，实现远程控制和监测。同时，人工智能技术的突破，如语音识别和机器学习，为智能家居带来了更强大的智能化能力。

（4）市场竞争与商业蓬勃（2010 年代至今）：在过去的十年里，智能家居市场

迎来了蓬勃的发展。国内企业在技术创新和产品研发方面取得了显著进展。智能家居产品的种类日益丰富，包括智能门锁、智能照明、智能家电等。大型科技公司纷纷涌入这一领域，推出了各种智能家居产品和平台。Amazon 的 Echo、Google 的 Nest、Apple 的 HomeKit 等，都成为了智能家居市场的代表性产品。这些产品不仅丰富了智能家居的应用场景，还通过强大的生态系统，推动了智能家居的普及[10]。同时，移动互联网的普及也使得用户可以随时随地控制家居设备，推动了用户体验的提升。

（5）多元化应用场景（2020 年代及未来）：智能家居在国内外的发展历程都经历了从初期探索到快速发展的过程。从技术进步到市场竞争，从单一应用到多元化场景，智能家居不断推动着家庭生活的智能化和便捷化。随着技术不断演进，智能家居的应用场景也在不断扩展。除了传统的智能照明、温控、安防等，智能健康、智能娱乐、智能厨房等领域也逐渐受到关注。例如，智能健康设备可以监测用户的健康状况，智能厨房可以通过智能设备辅助烹饪等。未来，随着更多技术的融合，智能家居的多元化应用将持续扩展。国内智能家居有望在技术创新、市场推广等方面继续取得新的突破。

在国外，智能家居已经有了很长时间的发展，人们居住的地方，大部分都是以分散的方式生活，而智能家居的系统，大部分都是与市政系统相连接的。所以，在国外，智能家居中，大部分的智能家居都是单独的安装，具有很强的个性，更注重于对安全的保护。要想让智能家居在国内得到进一步的发展，就必须要学习并借鉴国外智能家居的先进技术，以及符合我国国情的特色[11]。

1.3.2 智能家居发展趋势

随着无线通信技术的迅猛发展，智能家居作为信息技术与家庭生活的结合，正经历着快速的发展和不断的创新。未来的智能家居发展趋势将涉及多个方面，从技术创新到用户体验，都将有显著的变化。以下是智能家居的一些发展趋势：

随着物联网技术的不断发展，越来越多的设备和家居用品将能够通过互联网连接。从家电到照明、安防、健康监测等，各类设备将形成一个更加紧密的智能家居生态系统，实现设备之间的互联互通。不同厂商的设备和系统将更容易实现互联互通，使用户能够更自由地选择不同品牌的智能家居设备，而不必担心兼容性问题。智能家居将与可持续发展的理念结合，致力于提高能源效率和资源利用[12]。通过智能的能源管理系统，家庭能够更好地监控和控制能源消耗，从而降低能源浪费。可以更好地满足老年人和特殊需求人群的健康关怀需求。智能健康监测设备可以实时

监测用户的健康状况，并在需要时提供警示或发送通知给医护人员。这在一定程度上能帮助老年人及其子女。智能家居还可以通过环境传感器监测室内空气质量，促进家居的环保和健康。未来的发展趋势之一是加强智能家居系统的安全性，以防止黑客入侵和个人信息泄露。设备制造商和开发人员将不断努力，提供更可靠的安全措施保护隐私。

人工智能（AI）将在智能家居中发挥越来越重要的作用。智能家居系统将能够更好地学习和理解居住者的行为、偏好和习惯，以提供更加个性化的体验。语音助手和虚拟助手将变得更加智能化，能够更准确地理解和响应用户的指令和需求。虚拟现实（VR）和增强现实（AR）技术将逐渐融入智能家居领域，为用户提供更丰富的体验。家居购物可以通过虚拟现实实现，用户可以在不出门的情况下体验家具和装饰品的摆放效果[13]。增强现实技术也可以用于家居维修和维护，通过 AR 引导用户进行简单的维修操作。

智能家居的发展趋势将会更加智能化、便捷化、个性化、安全化和可持续化。例如：以往手机数据线有着多种接口，但随着时代的发展逐渐普及 Type-c 接口，做到了统一的标准。过去冰箱有着保存食物，延长保质期的功能，而现在冰箱上添加了智慧管理，可以清楚的显示食品的名字，重量，产地及保质期限并能够对于食物到期进行及时提醒。从日常生活中看出具有广阔市场潜力，在全国科技会议上也指出将通过互联网及人工智能等高新技术助力智能家居产业的发展。

1.4 研究内容

本文针对目前智能家居存在的布线复杂，设备之间协议不互通，能源消耗功耗大，家庭老人的智慧监测报警及家庭环境需求提高为切入点，以智能家居控制系统为核心，通过对单片机、嵌入式和无线网络通的研究，通过分析 WIFI、蓝牙、ZigBee、NFC、RFID 等通信技术并比较其优缺点[14]，综合考虑选取 WIFI 作为无线通信方式，设计一种基于 WIFI 无线通信的智能家居监控系统，通过对家庭内部的环境数据、安防监控、家电设备等进行综合管理，并添加了手机操控，摄像头实时监控，主要由 WIFI 无线通信技术，STM32 硬件主控中心， Android 手机移动终端这几个主要模块构成。

从功能上来说，这个智能家庭控制系统可以利用手机 APP 来控制家里的电器的开关，还可以利用语音来控制窗帘、电视、空调以及 LED 灯的开启关闭，而对于温

湿度、光照、烟雾监控等模块，当各模块数值分别达到设定的阈值时，风扇就会自动开启，还可利用摄像机对家庭环境进行实时监控，这一系列创新使家庭生活更加舒适、安全且便利，提升了整个系统的智能化和人性化水平。

1.5 结构安排

本文共分六章介绍基于 STM32 单片机的智能家居控制系统的设计与实现过程：

第一章：绪论。本文一开始讲述本课题的研究背景及意义，在此基础上对国内外目前的研究的情况进行了阐述与分析并对未来智能家居的发展趋势作了一个，根据研究内容，提出了一种基于 STM32 单片机的智能家居控制系统，并对每章节的架构做出结构安排。

第二章：系统总体方案。在阅读关于智能家居控制系统的相关信息后，有了一个总体思路，对其进行了全方面的方案分析，最终确定了本次设计的系统总体方案以及各个模块的组成，确定了从家居环境温湿度监测、烟雾检测、窗帘电视空调自动控制以及远程监测控制智能家居控制系统功能，为后续章节的硬件电路设计和软件设计提供了坚实的基础。

第三章：硬件电路设计。主要介绍了系统的硬件功能。在此基础上，对整个系统的各模块进行了电路设计，并对各模块的电路图进行了具体说明。

第四章：系统软件设计。根据控制系统的功能需求，使用 C 语言在 keil5 软件进行了 MCU 控制系统的软件设计，使用 JAVA 语言在 Android Studio 上开发设计出来 APP，并画出了每个模块的软件流程图，从而实现了整体软件设计。

第五章：实物焊接与调试。制作实物，对该智能家居控制系统进行多次测量比对后，将实测结果与理论值进行对比。得出其测试结果，本次设计的控制系统能够满足其功能要求。

第六章：总结与展望。对本文的研究成果进行总结，并对其中的创新点进行深入探讨[15]，同时也将提出未来的发展建议，以期达到更好的效果。

2 智能家居控制系统的方案设计

本文的主要目的就是通过建立一套综合性的智能家庭控制系统，让最终用户获得良好的用户体验。因此需要了解大众需求，设计原则，总体设计方案再进行无线网络通信技术的选取，控制终端的选取是本系统的设计要点，下面将从以下几个方面进行逐一说明。

2.1 系统需求分析

该智能家居控制系统可以根据其功能划分为三个主要部分，以满足用户的各种需求，其中监测采集部分主要通过传感器实时采集和监测该系统，显示室内的温湿度、烟雾浓度以及摄像头周围情况；阈值报警部分是根据用户设定的各类阈值范围，当烟雾浓度及光照达到预定的阈值时，系统会自动启动风扇，确保系统的安全；控制部分允许用户以多种方式控制智能家居。用户可以通过相应的手机应用程序进行控制，同时也可以使用语音命令进行操作[16]。通过继电器，用户能够远程操控窗帘、电视和空调等设备。

2.2 设计原则

为了使所设计的智能家居控制系统接近家庭使用需要，在设计的过程中遵循了四条原则，具体包括：

方便性：智能化家庭系统的概念就是要为人们创造一个具有高安全、高舒适度、强便捷智能化的家庭生活环境。该智能系统及其相关的产品，操作简便，实用。

可靠性：智能家庭控制系统的每个子系统都可以安全、稳定、高效地工作，这是确保整个系统稳定运行的前提。只有对所有子系统采取必要的补救预防措施，才能确保系统的良好运行，应对各种不确定性因素。

标准性：由于各大厂商设计的产品兼容性差，所以为了能让智能家庭控制系统在未来的扩展性和兼容性得到充分的保证，在系统的初期建设阶段，就应当按照该工业标准来进行设计，以保证各子模块之间的兼容和互联。

综合性：目前智能家庭控制系统的功能单一，且多种设备功能交叉，故需要设计一套完整的、集成的系统，且这个综合系统种类要多，具有多种方案供人们选取。

因此要解决的核心问题是：突破目前市面上普遍存在的高价低质的缺点，以最小的代价满足人的智能化需要，为使用者提供高舒适、安全、便捷的居住环境[17]。

2.3 系统总体设计方案

系统的整体设计由数据采集、电子控制、ESP32 摄像头、单片机、网络服务器、移动 APP 等模块组成。其单片机是整个系统的核心，既可以通过网络与远程手机客户端进行通信，又可以通过传感器对家庭联网电器进行环境数据进行采集，随时进行电器的开启或关闭。另外，在烟雾浓度较高的情况下，单片机设置好阈值也会开启风扇来应对突发状况。使用者也可以使用配置好数据的手机 APP 客户端，来实时地了解到家庭环境中的信息变化情况，窗帘、电视、空调和 LED 灯的状态都会在 APP 上实时地显示出来，还可以对 APP 进行相应的操作控制。手机 APP 客户端的主要功能是通过简单易用的界面实现复杂的功能，同时还可以通过语音控制家中的电器，为用户提供更加便捷的智能家居体验。该系统的整体结构框图如图 2-1 所示。

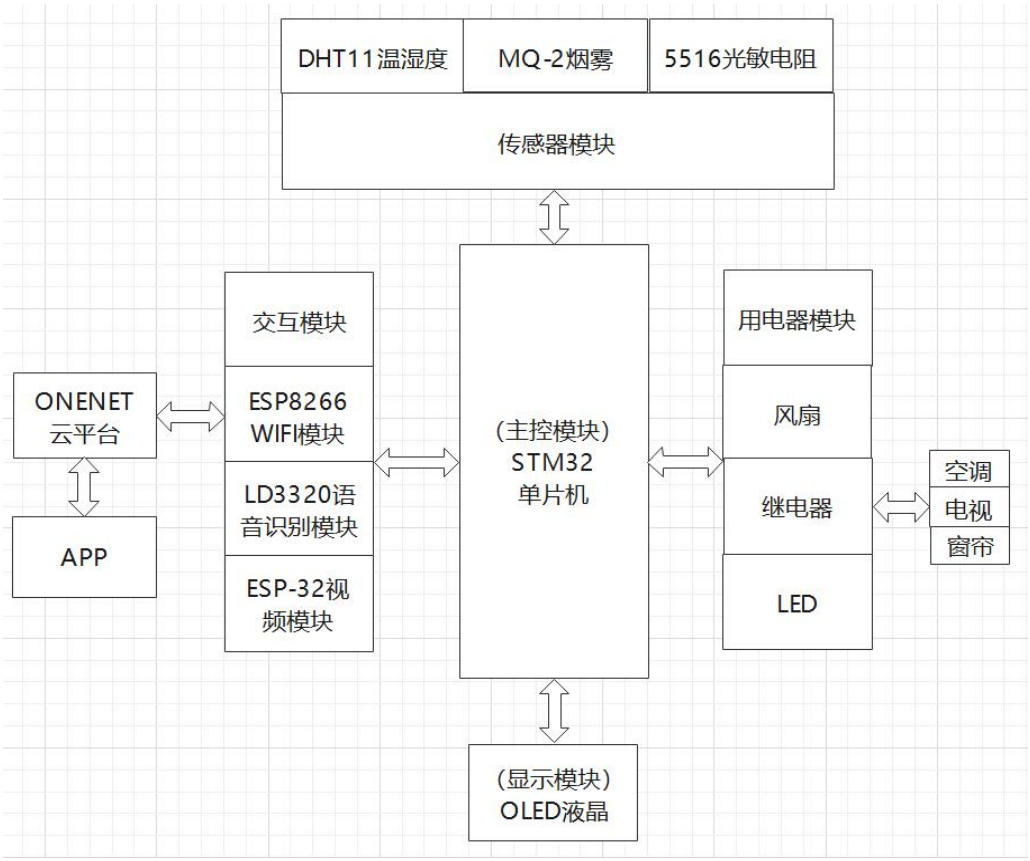

图 2-1 整体结构框图

Fig. 2-1 Overall structure block diagram

2.4 网络通信技术

2.4.1 通信技术的选取

随着互联网技术的快速进步，现代智能家居市场的大部分设备都可以通过移动终端进行远程控制。用户在操作时，追求简单、便捷的控制体验。这些移动终端与智能家居硬件之间的通信，主要依赖于网络技术来实现数据的传输。考虑到多种可选的无线通信技术，为了使系统更加实用并与日常生活紧密结合，选择了无线通信作为主要的通信手段[18]。下表 2-1 列举了行业间常用的无线通信技术，并针对以上四种通信技术之间进行功能比较。

表 2-1 无线通信技术之间比较

Tab. 2-1 Comparison between wireless communication technologies

无线通信技术	蓝牙	WIFI	ZigBee	NFC
传输速度	1Mbps	11-54Mbps	100Kbps	424K
通信距离	10-100m	20-200m	2-20m	20cm
工作频段	2.4GHz	2.4GHz，5GHz	2.4GHz	13.56GHz
国际标准	IEEE80.15.1x	IEEE802.11b IEEE802.11g	IEEE802.15.4	ISO/IEC18092 ISO/IEC21481
网络拓扑结构	点对点，小型网络	基础网络，自组网络	点对点，星状/网格网络	卡片模式，点对点
安全性	高	低	中等	极高
功耗	20mA	10-50mA	5mA	10mA
典型应用	通信、汽车、IT、工业、医疗、多媒体	无线上网、PC、手机	无线传感器、医疗	手机、地铁、社区

在智能家居控制系统中，选择适合的无线通信技术对于确保系统的稳定性、性能和用户体验至关重要。在众多无线通信技术中，NFC、WIFI、ZigBee 和蓝牙都是常见的选择，各自具有特定的优势和适用场景。以下针对这些技术进行比较和适用性分析。

NFC(近场通信)是一种短距离通信技术，通常用于近距离的数据传输和交换。在智能家居控制系统中有一些特定的应用场景，如智能门锁、物品标签等。NFC 通信在非常短的距离内进行，提供了较高的安全性，难以被远程窃取数据且设备之间的连接通常只需将其靠近，操作简便，但 NFC 的通信距离非常有限，适用于近距离的数据传输，限制了其在智能家居中的应用范围，与其他技术相比，NFC 的数据传输速率较低，不适用于需要高带宽的应用。

WIFI 是一种高速、长距离的无线通信技术，已广泛应用于家庭网络中。在智能家居控制系统中，WIFI 提供高速数据传输能力，适用于涉及视频流、音频流和大量数据传输的场景，保证了稳定的控制和监控体验，WIFI 信号覆盖范围大，可以覆盖整个家庭，确保设备的连接稳定性而且普及度高，多数家庭已经配置 WIFI 网络，无需额外投资，节省成本。但也有其不足，WIFI 设备通常需要更多的电力供应，可能在一些低功耗设备上不太适用。

ZigBee 是一种短距离、低功耗通信技术，适用于电池供电的设备[19]，如传感器和遥控器，支持自组织的 Mesh 网络，提供了更好的网络覆盖和鲁棒性，适合大规模设备连接。但与 WIFI 相比，ZigBee 的数据传输速率较低，不适合需要高带宽的场景。在实际应用过程中，ZigBee 技术往往依赖于一个相对复杂的网关来进行通信，这使得难以实现直接的端到端控制，可能导致一定的延迟。

蓝牙在特别是在短距离、低功耗设备的连接方面有一定使用。蓝牙低能耗（BLE）技术适用于电池供电设备，如智能传感器、健康设备等。几乎所有现代智能手机和平板电脑都支持蓝牙连接，提供了方便的设备互联性。但受到设备连接数量和通信距离短的限制，不适合应用中智能家居控制系统中。

综合考虑，针对智能家居控制系统，WIFI 技术是一个较为全面的选择。其高速数据传输、广范围覆盖和现有网络基础设施的利用，使其能够满足智能家居中多种应用的需求。然而在特定场景下，如低功耗设备互联，可以考虑使用 Zigbee 或蓝牙等技术来弥补 WIFI 的一些不足。因此，在选择无线通信技术时，需根据智能家居控制系统的实际需求，综合考虑各种技术的优劣，以达到最佳的性能和用户体验。

2.4.2 WIFI 通信协议的设计

智能家居系统利用 WIFI 无线通信技术连接各个模块，实质上是基于 TCP 的网络传输，主要负责字符串的传递。而根据 OSI 参考模型和 TCP/IP 参考模型，知道两个参考模型传输层在本质上是一致的[20]。这一层的核心任务是确保源主机与目标主

机之间能够有效地通信。传输层不仅负责建立、管理和终止连接，还处理错误恢复和流量控制，确保数据的有序和完整传输，为应用层提供透明、可靠的数据传输服务。在传输层中，有两种主要的端到端的通信协议：传输控制协议（TCP）和用户数据报协议（UDP）。这两种协议各自拥有独特的特性和应用场景。TCP 是基于连接的、可靠性强的协议，保证了从一台设备发送的数据能够完整且正确地被另一台设备接收，并采用了重传机制来确保数据传输的准确性。而 UDP，与之相对的是无连接的、不保证可靠性的协议，允许应用直接传送数据，而不必事先建立通信连接。由于 UDP 的这种特性，在实时通信、如音视频流、广播等场景中，常被优先选择，因为在这些场景中，速度和实时性往往比数据的完整性更为关键。下表 2-2 为 TCP 协议和 UDP 协议的对比。

表 2-2 TCP 协议和 UDP 协议对比

Tab. 2-2 Comparison between TCP protocol and UDP protocol

TCP	UDP
TCP 协议面向连接	UDP 协议面向非连接
TCP 协议传输速度慢	UDP 协议传输速度快
TCP 协议采用字节流方式，若字节流太长，可将其分段，保证数据顺序，不会导致乱序或数据丢失	UDP 协议不保证，尽最大能力交付
TCP 协议对系统资源要求多	UDP 协议要求少
协议保证数据正确性，超时重发，会丢弃重复数据，有可靠性	UDP 协议可能丢包
协议提供端到端，全双工通信	UDP 协议可能丢包
都是网络层协议	

因此 TCP 协议提供了稳定且可靠的数据传输，但可能需要更多的资源开销。相比之下，UDP 协议虽然不如 TCP 可靠，但其简洁和高效。鉴于本系统要求数据的双向交互和对通信的安全性有一定的要求，综合各种因素，TCP 协议更适合本系统的需求。为了实现各模块间的实时通信，将采取 TCP 协议进行数据传输。

2.4.3 云平台介绍

云平台是一种基于云计算技术构建的软件和服务平台，作为一种现代计算模型，允许用户通过互联网访问和使用远程服务器上的计算能力、存储容量和多种应用服

务。这种模型避免了用户在本地部署和维护物理服务器的需求，从而降低了初始投资和运营成本。基于云计算技术，云服务为用户提供了一个虚拟化的、高度可定制的计算环境，其核心优势在于按需分配资源、高度的可伸缩性和灵活性。云服务在智能家居系统中的应用已经成为一种趋势，为家居环境带来了前所未有的便利性和智能化体验，给用户提供了强大的后端支持，使得家居环境更加智能、高效和友好。

云平台的提供的服务形式多种多样，主要包括基础设施即服务（IaaS）、平台即服务（PaaS）和软件即服务（SaaS）[21]。这些模型分别满足了不同层次的计算需求，从底层的硬件资源到高层的应用服务。特别是在智能家居领域，云服务为设备间的数据交互和远程控制提供了强大的后端支持。

从部署角度来看，云平台可以分为公有云、私有云、混合云和多云。公有云，由第三方云服务提供商管理，为广大用户提供共享的计算资源。私有云，通常为特定组织定制，提供更高的数据安全性和自定义能力。混合云结合了公有云和私有云的特点，为组织提供了更大的灵活性。而多云策略则允许组织利用多个云服务提供商的优势，确保业务连续性和避免供应商锁定。

云平台为现代企业和个人用户提供了一种高效、经济和灵活的计算模式，推动了技术创新和业务转型，是数字化时代的关键支撑技术。本次设计所选取的云平台为 ONENET 云平台，这是一个由中国移动推出的 PaaS 级别的物联网开放平台[22]。可为开发者提供了便捷的工具，使得设备的接入和连接变得简单，从而加速产品的开发和部署过程。对于智能硬件和智能家居领域，ONENET 都能提供一站式的物联网解决策略。平台支持标准 MQTT、CoAP、LwM2M 和 HTTP 协议接入，是物联网的重要组成部分。此次选用 MQTT 协议在云平台上接收智能家居系统发来的消息。

2.4.4 MQTT 协议介绍

MQTT 是 IBM 于 1999 年推出的一种基于 TCP 的轻量级消息传输协议。是为有限带宽的网络环境设计的，并能维持长连接，具有一定实效性。其主要特点是通过最小的代码和带宽实现远程设备的实时、稳定的消息传输。由于其低成本和低带宽使用，在物联网、微型设备和移动应用中得到了广泛应用。

在实现 MQTT 协议的过程中，必须确保客户端与服务器端之间的有效通信。在 MQTT 协议结构中，存在三个主要的角色：发布者（Publish）、消息代理（Broker，即服务器）和订阅者（Subscribe）[23]。发布者和订阅者均作为客户端存在，而消息

代理则扮演服务器的角色。MQTT 客户端不仅具有向服务器发布消息的能力，还可以从服务器端接收消息。当客户端向服务器发送消息时，此行为被定义为“发布”。为了从服务器端接收消息，客户端首先需要执行“订阅”操作。此订阅机制可以类比于在线视频平台上的订阅功能：当新内容发布时，平台会通知已订阅该内容的用户。服务端通常是一台服务器，也称为“消息代理”（Broker）。充当信息传输的中心节点角色，负责接收来自一个客户端的消息并将其转发给另一个客户端。此外，消息代理还承担了管理客户端的责任，确保客户端间的通信流畅，并保障消息的准确性和可靠性。MQTT 协议身份如图 2-2 所示。

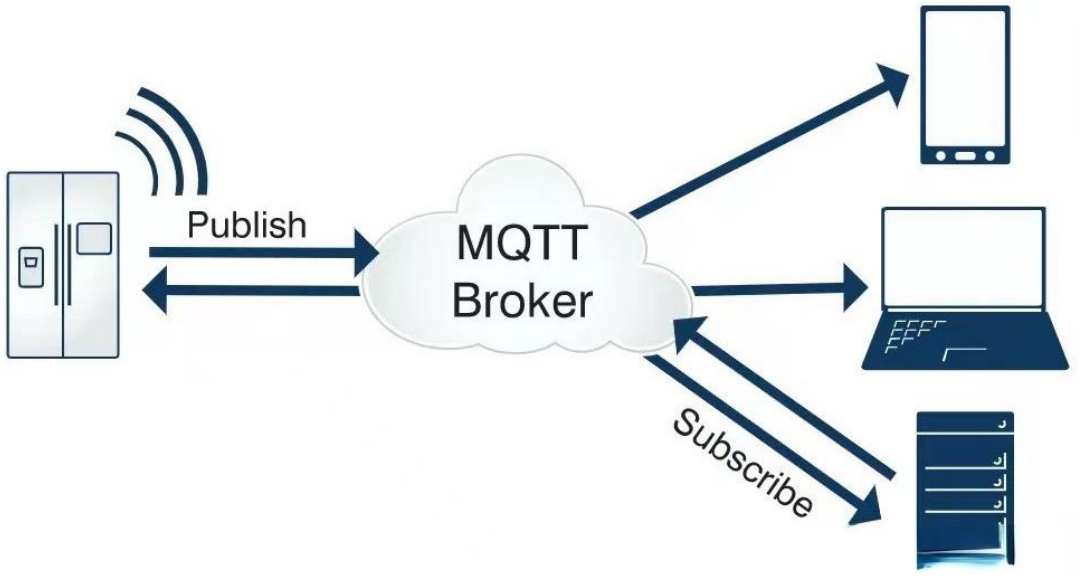

图 2-2 MQTT 协议身份

Fig. 2-2 MQTT protocol identity

其次，MQTT 的设计非常注重效率和可靠性。为了实现这一点，提供了三种不同的服务质量（QoS）级别，以确保消息的正确交付。从只交付一次但不进行确认的 QoS0，至少交付一次并由接收方确认的 QoS1，再到只交付一次并通过双向握手进行确认的 QoS2，用户可以根据自己的需求选择合适的 QoS 级别。

此外，MQTT 还提供了一种称为“遗嘱”（Last Will and Testament）的机制。这允许客户端在连接到 MQTT 代理时设置一个特定的消息。如果代理检测到客户端由于某种原因而意外断开连接，将会自动发布这个遗嘱消息，通知其他客户端发生了什么。安全性也是 MQTT 设计的一个重要方面。尽管 MQTT 本身不包含加密或认证机制，但可以很容易地与现有的安全协议（如 TLS/SSL）结合，以确保数据的机密

性和完整性。

MQTT 的简单性、高效性和可靠性使其在各种应用场景中都得到了广泛的应用，从简单的家用设备到复杂的工业系统。从智能家居系统中的灯泡和开关，到工业自动化中的机器人和传感器，再到健康监测中的心率监测器和血糖计，MQTT 都有其应用之处。其简单的设计和低功耗特性使其成为这些应用的理想选择。随着物联网的持续发展，可以预料到 MQTT 的重要性和普及度将继续增长。

2.5 控制终端系统的选择

随着智能手机技术的迅速发展和广泛应用，其已成为现代人类生活中不可或缺的工具。在智能家居领域，用户对于能够实时监控家居系统状态的需求日益增强，因此选择智能设备作为控制终端显得尤为重要。本研究对 Windows Phone、Linux、Android 和 Mac OS 四种操作系统进行了深入的比较分析。

Windows Phone，作为微软推出的移动操作系统，曾经拥有一定的市场份额在用户界面和设计哲学上与桌面版的 Windows 有许多相似之处，与微软的其他产品，如 Office 和 Outlook，有着无缝的集成。但由于微软已停止其进一步的开发，并且市场占有率逐渐下降，因此不建议将其作为首选的智能家居控制终端。

Linux，一个开源的操作系统，广泛应用于多种嵌入式系统中。尽管其具有高度的自定义性和灵活性，在服务器和开发环境中有着广泛的应用，但对于大多数消费者而言，其复杂性和一定技术要求可能会成为一大障碍，目前主流的商业软件可能不支持 Linux 环境。

Android，由 Google 推出，是当前全球移动设备中最为普及的操作系统。其丰富的应用生态、广大的开发者社区以及对各种通信技术的完美支持使其成为智能家居控制的理想选择。

Mac OS，主要应用于苹果的 Mac 电脑中，虽然其在桌面领域表现出色，与 Apple 的其他产品，如 iPhone 和 iPad，有着深度集成，为用户创造了一个连贯的 Apple 生态系统，但在移动设备和智能家居控制终端的应用上并不常见。而且只能在 Apple 的设备上运行，且价格昂贵。

综上所述，考虑到用户基数、技术支持和应用生态等因素，Android 被认为是智能家居控制终端的最佳选择，为用户提供了一个稳定且高效的平台。

2.6 本章小结

本章主要讲述设计智能家居控制系统的方案分析，并确定本次设计的总体方案以及需要实现的模块功能，确定了从家居环境温湿度监测、烟雾检测、窗帘电视空调语音控制以及远程监测进行模块化设计，对比了当前网络通信技术并确定以 WIFI 作为无线通信技术，讲述了当前 WIFI 通信协议以及当前云平台上的功能介绍了 MQTT 协议，最后进行了终端控制系统的对比并最终选择 Android 平台作为智能家居控制终端，为第三章中硬件电路的设计和第四章中软件开发奠定了基础。

3 硬件电路设计

本系统以模块化设计为主，选用 STM32 作为主控制器。首先，需要根据每个模块的功能需求来选择合适的传感器和控制模块。接着，将这些模块与 STM32 控制器连接。然后，将分模块进行程序编写和调试，确保每个模块可以正常运行。最后是实现 WIFI 模块与云平台的连接和通信，以实现远程数据传输功能[24]。

3.1 主要芯片选择

STM32 芯片选择是智能家居设计的第一步，该芯片的微控制器是意法半导体(ST Microelectronics)公司基于 ARM Cortex-M3 核心设计所推出的产品[25]。这款微控制器以其高性价比、低功耗特性受到广泛关注，被认为是性价比高、功能全面且可靠的解决方案，涵盖了从基础到高性能的多个系列。ARM Cortex-M3 核心旨在满足高效、经济和低能耗的嵌入式应用需求。Cortex 是 ARM 公司的最新处理器核心系列，为满足多样化市场需求而生，不仅提供标准的 CPU 核心，还有统一的硬件系统架构。其中，Cortex 系列有针对高级应用的“A”系列和针对微控制器应用的“M”系列。STM32 微控制器正是基于 Cortex-M3 设计，旨在实现高性能、低功耗和成本效益。STM32 微控制器拥有各种内存容量、外设选项和封装类型，能满足各种应用需求，从简单的消费电子产品到复杂的工业设备。其主要特点包括低功耗工作模式、丰富的外设集成、灵活的时钟系统、及对实时操作的强大支持。加之 STM32 提供了大量的开发工具、固件库和硬件开发板，配合着一个庞大且活跃的开发者社区，使得 STM32 在全球嵌入式领域中脱颖而出，成为工程师们首选的微控制器平台之一。

STM32 有多个系列，主要包括基本型和增强型。基本型的最大时钟频率为 36MHz，而增强型可达 72MHz，性能领先于同类产品。两者的内置闪存大小不同，分别为 32K 和 128K。另外，基本型和增强型的 SRAM 容量分别为 48K 和 64K。增强型还配备了更多的外设，并能在 72MHz 主频下运行，而基本型的最大主频为 36MHz。在本研究中，考虑到成本、外设配置和智能家居系统的需求，选择了增强型的 STM32F103C8T6 作为系统的主控制器。

3.2 硬件电路模块

3.2.1 单片机最小系统

STM32F103C8T6 是一款基于 ARM 的 Cortex-M3 核心设计的高性能微控制器。这款微控制器集成了 Cortex-M3 的强大处理能力，使其在各种应用中都能提供出色的性能[26]。Cortex-M3 核心是 ARM 公司专为嵌入式应用设计的，因此 STM32F103C8T6 在低功耗和高效处理方面都有出色的表现。

除了其核心性能外，STM32F103C8T6 还具有很强的抗干扰能力。这意味着在各种噪声环境中，都能稳定工作，不易受到外部因素的影响。这一特点使得其特别适合用于工业控制、医疗设备、通信设备等需要高可靠性的应用中。

此外，STM32F103C8T6 还提供了丰富的外设和接口，如 GPIO、UART、SPI 和 I2C 等，可以轻松与其他设备进行通信和交互。64KB 的 Flash 和 20KB 的 SRAM 为大部分嵌入式应用提供了充足的存储。另外，其灵活的电源选项和低功耗模式使其尤其适合于电池供电的设备。

STM32F103C8T6 是一种非常适用于新手的 MCU 模型。加上 ST Microelectronics 良好的市场声誉，以及 STM32 系列活跃的开发者社区和丰富的资源支持，使得 STM32F103C8T6 成为众多工程师在设计中的优先选择。STM32F103C8T6 最小系统配置了电源、外部晶振、复位电路等，其电源配置如下表所示。

表 3.1 电源配置

Tab. 3-1 Power configuration

STM32F1	供电范围	引脚连接
VDD	2.0V-3.6V	接去耦电容到稳压源
VBAT	1.8V-3.6V	外接电池
VDDA(ADC 工作)	2.4V-3.6V	外接两个去耦电容
VDDA(ADC 不工作)	2.0V-3.6V	
VREF+	2.4V-VDDA	引脚连到 VDDA

其中 VBAT 直接外接电容到稳压源。根据上表，本系统选择 3.3V 作为系统的稳压源。启动模式设置，启动模式由 BOOT0、BOOT1 两个引脚决定，通常设置为

BOOT0=0，BOOT1=0 为一般闪存方式[27]。

在图 3-1 中显示了 STM32F103C8T6 的引脚配置。

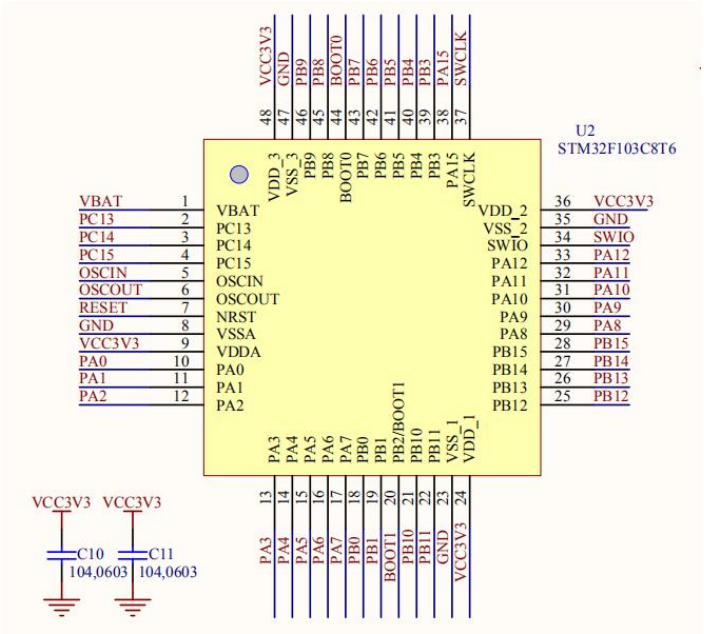

图 3-1 STM32 F103C8T6 引脚配置

Fig. 3-1 STM32 F103C8T6 pin configuration

STM32F103C8T6 单片机主要基本特性:根据所选择的模型，可用的特性是不同的。如果是外围设备模组比较少，则通常是含有比较小数量的功能模组。举例来说，一种机型仅有一个 SPI 与两个 USART，这两种机型分别为 SPI1 与 USART1 与 USART2。PC13、PC14、PC15 的插头是由一个功率开关来提供电力的，但是该功率开关只能吸收一定的（3 毫安）电流[28]。所以，当这三个引脚用作输出引脚的时候，存在着如下的局限性：一次只有一个引脚可以用作输出，而用作输出脚时，只能工作在 2 MHz 的模式下，最大驱动负荷是 30pF，而且不能用作电流源（比如驱动 LED）。不像 LQFP64，TFBAG64 没有 PC3，但是有一个 VREF+管脚。在输出方式中，PD0、PD1 仅可设定在 50 MHz 的输出方式。

单片机加上一些外围硬件电路，可以组成最小系统。最小系统是基于单片机完成各种各样功能所必不可少的，在最小系统的基础上可以自由添加各种模块，从而可完成各种各样的功能。STM32F103C8T6 最小系统配置了电源、晶振电路、复位电路等。

复位电路：常见的复位电路有上电复位和按键复位两种电路。其复位电路是单片机引脚 RST 上外接一些电容和电阻组成的，而上电复位则是上电后接通电源从而

靠着给电容 C2 充电来自动完成。按键复位则是在接通电源的情况下按下按键将电阻 R1 与 VCC 连通来实现的复位，运行状态是按下按键传输信号到 I/O 口 RST 然后控制单片机改变电平的高低。

时钟电路：外围硬件电路由晶振及电容组成，晶振在单片机内起着无可取代的作用，单片机完成程序给予的指令都需要靠晶振所产生的时钟频率来决定的[29]。电容的作用主要是滤掉晶振所产生的高频信号，让时钟频率更加稳定准确。

在图 3-2 中显示了 STM32F103C8T6 的中的复位电路和晶振电路。

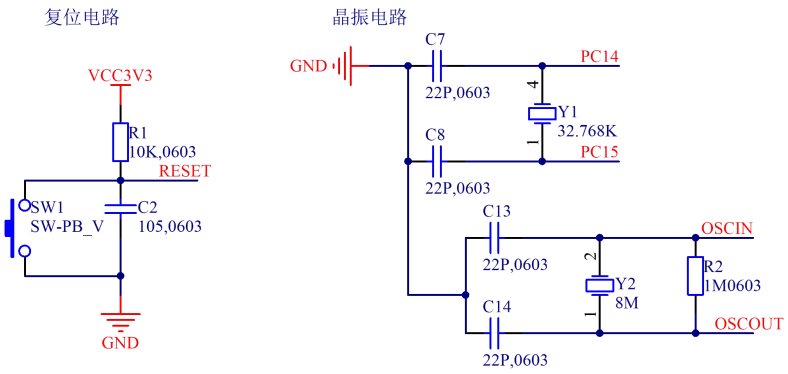

图 3-2 复位电路和晶振电路

Fig. 3-2 Reset circuit and crystal oscillator circuit

3.2.2 液晶显示模块电路

在该控制系统中，选择了 1.3 英寸的有机发光二极管（OLED) LCD 屏为主要显示设备。其中 OLED 显示器提供了出色的对比度和色彩饱和度，可使显示内容更为生动和真实，且提供了宽广的视角，无论从哪个角度查看，都能获得一致的显示效果。与传统的 LCD 显示器相比，OLED 显示器在显示深色内容时消耗的功耗更低。OLED 的基本原理就是利用 LED 的特点，在不同时刻，OLED 所发射出来的光的颜色的差异，来制作一个 LED 的显示屏。这个系统的显示速度非常快，稳定性强，亮度高，分辨率高，数据显示直观。其内部集成 8 个存储地址，可用于保存汉字，并直接显示这些汉字，无需额外添加液晶模块，这不仅可以降低成本，还简化了使用流程。

OLED 显示器模组具有四种不同的界面模式，可以通过模组 BS1/BS2 进行调整，而接口模式则有四种：并行模式 6800 和 8080，串行 SPI 模式 4 线，以及 IIC 模式。为了满足该控制系统的需求，选择了 4 管的 IIC 接口 OLED 显示器，引脚名称及功能说明如表 3-2 所示。

表 3-2 OLED 引脚功能说明

Tab. 3-2 OLED Pin Function Description

引脚号	引脚名称	功能
1	GND	电源负极
2	VCC	电源正极
3	SCL	时钟信号线
4	SDA	双向数据线

而 IIC 总线是由飞利浦公司推出的一种串行、同步、半双工通信协议。分别由时钟线(SCL)和数据线(SDA)组成。IIC 总线设备上的串行数据 SDA 需要接到总线的 SDA 上，各设备的时钟线 SCL 需要接到总线的 SCL 上。其中 SCL 是 IIC 总线时钟信号，连接 STM32 单片机的 PB6 口，SDA 是 IIC 总线数据信号，连接 STM32 单片机的 PB7 口，VCC 接 3.3v 电源，GND 接地。图 3-3 中说明了发光二极管的显示电路的设计。

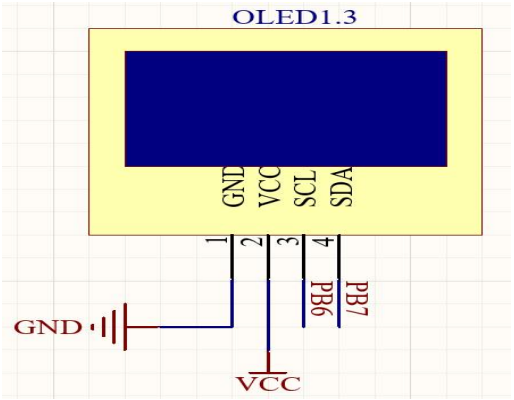

图 3-3 1.3 寸 OLED 显示屏电路图

Fig. 3-3 1.3 inch OLED display circuit diagram

3.2.3 温湿度检测电路

温湿度检测模块的器件采用 DHT11 型温湿度探测装置，该装置具有湿、温两种检测功能，是目前市面上较为流行的温湿度复合传感器。应用专用的数字模块采集技术和温湿度传感技术，确保产品具有极高的可靠性与卓越的长期稳定性。该产品最大的优点是，不需要进行模数转换，可以直接输出控制系统可接收的数字信号，这对本次设计开发有很大的帮助，还可以省去一些不必要的转换模块，可大幅简化

电路。DHT11 拥有 4 个引脚，各引脚的功能如表 3-3 所示。

表 3-3 DHT11 引脚功能说明

Tab. 3-3 DHT11 Pin Function Description

引脚号	引脚名称	功能
1	GND	电源负极
2	NC	空脚
3	DATA	串行数据，单总线接口
4	VCC	电源正极

DHT11 温湿度传感器和控制系统之间采用单线连接，数据通过一条线传输。传送格式和规则如下。

传感器采集到的湿温度信息后，其发出的数字信号为包含 16 bit 大小的湿度数值、16 bit 大小的温度数值和 8 bit 大小的效验数值。其中，湿温数值以整数和小数各占半个位数的方式存储。所有的数据都是二进制的。有效性值为以上四个数值之和。如果效验是正确的，那么这个值就是正确的，温湿度校验如图 3-4 所示。

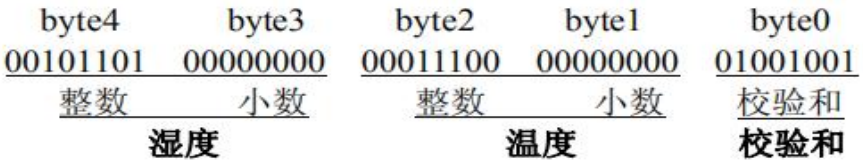

图 3-4 温湿度校验

Fig. 3-4 Temperature and humidity calibration

按照以上的计算方法，将湿度值整数部分转化为十进制，数值为 45，小数部分为零，则得出其湿度为 45%。将温度值的整数部分转换成十进制的数字，数值是 28，十进制的小数部分是 0，从而得出的温度是 28℃。同理可以得出效验值的大小为 73，将湿温度数值整数和小数部分分别相加，若得到的为 73，则可以得出此次传输数值是正确的。

DHT11 传感器的通信流程是在控制中心向传感器发出开始信号后，传感器收到信号后会做出回应，将输入信号拉高并开始进行传送。对获取到的信号进行处理，以 1 bit 为单位，按顺序进行传送，直至 40 bit 尺寸的完整数据包全部发送完毕，而这时，输出信号会进入拉低模式，DHT11 通讯如图 3-5 所示。

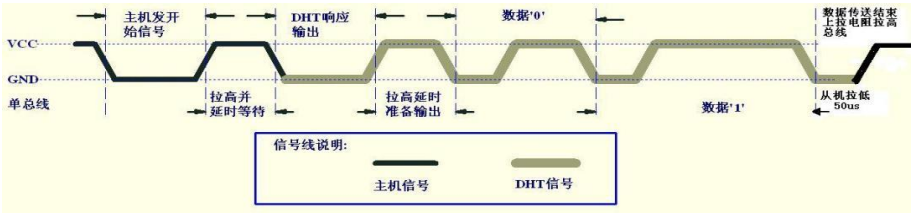

图 3-5 DHT11 通讯

Fig. 3-5 DHT11 communication

DHT11 通讯起始阶段传送数据如图 3-6 所示。

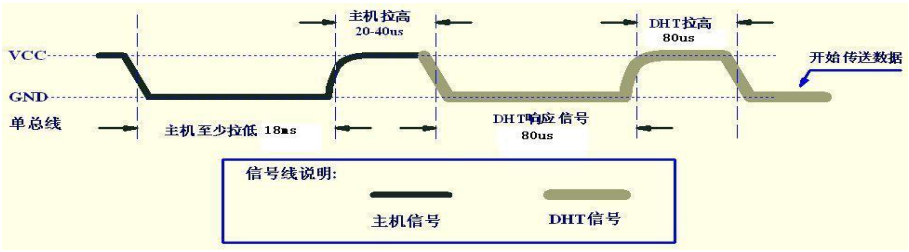

图 3-6 DHT11 传送数据

Fig. 3-6 DHT11 transmission data

在传输给传感器的过程中，需要一定的时延。在单总线方式下，为了保证总线传送的进程不会出现错误，可以对总线传送的方位进行快速而准确的变换，从而保证了总线的使用权变换的顺利进行。当传感器收到初值时，需要将传送延迟设置为 18 ms 或更高，不然将会断开连接，从头启动初值。同样，当总线在繁忙的情况下，并且在将命令发送到所述传感器的情况下，需要将所述总线的使用恢复到 DHT11 上，并且需要将所述延迟时间大于 20 微秒，小于 40 微秒，才能恢复所述的 DHT11。

STM32 单片机经 IO 管脚 PA2 与与 DHT11 数据端 Data 相连接，DHT11 检测到的温湿度数据通过 DHT11 串口输出，一次完整的数据传输为 40bit，通过转换得出温度和湿度的数值。DHT11 发送完 40 位数据后，STM32 单片机可以结束通讯，或者继续发送启动信号读取下一次的数据。

温湿度传感器 DHT11 的电路原理图如图 3-7 所示。其中 DHT11 的工作电压为 3 至 5V，所以在与 3.3V 的 STM32 单片机连接时，可以直接供电。其中 Data 是 DHT11 的数据发送端，连接 STM32 单片机的 PA2 口，GND 接地。

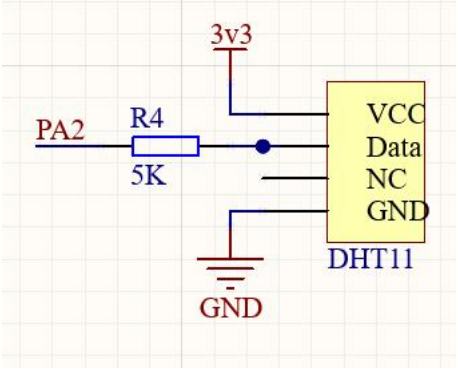

图 3-7 DHT11 电路

Fig. 3-7 DHT11 Circuit Diagram

3.2.4 光强检测电路

光敏电阻器是一种特殊的光学管道，主要由硫化镉制成，但也可以使用硒、铝、铅、铋等其他元素。当光照射到某一特定波长时，这种材料的电阻会迅速降低[30]。这是因为，光照产生的载流子会在外加电场的作用下，发生飘移运动，导致电子和空穴都朝着电源的正极移动，从而使光敏电阻器的阻值迅速降低。

模数转换器（ADC）是将连续的模拟信号转换为离散的数字值的核心组件。其工作原理基于两个主要步骤：采样和量化。首先，采样过程将连续的模拟信号以一定的时间间隔进行离散化。通过采样定理，也就是尼奎斯特定理，确定了采样频率，以确保在采样过程中不会丢失重要的信号信息。其次，量化过程将每个采样值映射到一系列离散的数字值中。这个映射根据事先设置的分辨率将连续的模拟信号分为有限数量的离散级别。更高的分辨率能够提供更准确的数字表示，但也需要更大的存储空间和处理能力。模数转换器的输出结果是一串数字数值，代表了原始模拟信号在特定离散时间点上的振幅。这些数字数值可以被进一步处理、传输或储存，以便进行后续的数字信号处理。

本文所设计的系统需收集周围的照明光强，才能进行照明光强的计算。通过使用光敏电阻测量光照强度，将光照模拟量通过用 ESP8266 的一路内部 ADC 转换电压，通过换算得到光照强度数据，并传输到单片机进行处理，从而获得准确的测量结果。ESP8266 内置了一个 10 位 ADC，只有一个 ADC 通道，即只有一个 ADC 输入引脚可读取来自外部器件的模拟电压，预留的 ADC 引脚为 PA0，能够读取 0 到 1V

范围内的电压，可将这个电压输入信号映射为一个介于 0 到 1023 之间的数值。

光敏电阻的特点是在不同的光线下，光敏电阻的电阻会随着光线的不同而不同，而变成随着光线而不同的电压。STM32 单片机利用模拟量采集管脚 PA0 来获取光电阻的分压值，再经过运算将 0-VCC 电压转换为 0-100%光照，并将数值显示到 OLED 屏幕上。当其检测数据值超过了设定的阈值时，STM32 主控中心就会发出报警信号，这时就会启动风扇。光强检测电路图 3-8 所示。

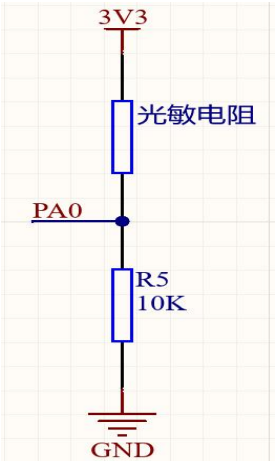

图 3-8 光强检测电路

Fig. 3-8 Light intensity detection circuit

3.2.5 烟雾浓度检测电路

在本设计中，采用 MQ-2 型气体传感器来检测城市煤气、天然气、液化石油等以氢气为主要成分的气体浓度，该传感器灵敏度高，有较好的稳定性，使用寿命较长，具有良好的抗干扰性能，即使是水蒸气、烟雾等干扰气体也能被有效抑制。

MQ-2 气体传感器的结构和轮廓是用小型 AL2O3 陶瓷管、SnO2 敏感层、测量电极和加热器组成的，被固定在塑料或不锈钢腔中[31]，气体传感器需要加热器才能启动。气体传感器已经完成包装，共有 6 根针，其中 4 根用于收集信号，另外 2 根用于供电。具体工作原理是当烟雾传感器的感应器受到烟雾颗粒的刺激时，感应器的温度会逐渐升高，同时感应器的响应时间也会变得更迅速。MQ-2 传感器内部采用了高灵敏的热传感材料作为导热材料。当烟雾颗粒被吸附到传感器表面时，这会导致电信号迅速变化，虽然这种变化微弱，但经过信号放大后，就能获得明显的电压变动。MQK-2 型元件外形结构图如图 3-9 所示。

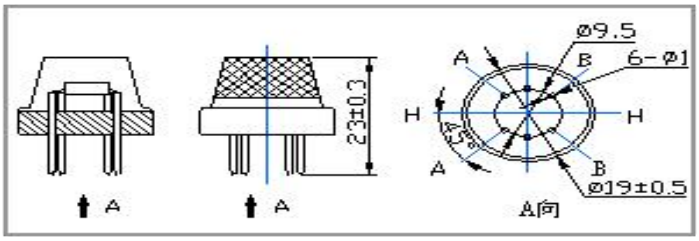

图 3-9 MQK-2 型元件外形结构图

Fig. 3-9 Outline structure diagram of MQK-2 type components

ADC0832 是一种 8 位模数转换器（ADC），能够将模拟信号高效地转换为 8 位二进制数字，具有两个独立的模拟输入通道，允许用户选择不同的输入源进行转换。这一特性使其非常适合需要同时监测多个信号源的应用，如数据采集系统和传感器接口。此外，ADC0832 支持外部参考电压，使用户能够根据具体需求定制输入电压范围，确保精准的数据转换。为了确保准确性和控制，ADC0832 需要外部时钟信号，用户可以根据应用的要求配置时钟频率。ADC0832 拥有 8 个引脚，各引脚的功能如表 3-4 所示。

表 3-4 ADC0832 引脚功能说明

Tab. 3-4 ADC0832 Pin Function Description

引脚号	引脚名称	功能
1	CS	片选端，低电平有效
2	CH0	模拟信号输入端
3	CH1	模拟信号输入端
4	GND	电源负极
5	D1	两路模拟输入选择输入端
6	D0	模数转换结果串行输出端
7	CLK	串行时钟输入端
8	VCC	电源正极

烟尘浓度越高，烟尘传感器输出的模式电压也就越高。通过将 MQ-2 烟雾传感器输出的模拟信号接入 CH0 引脚，将芯片选择引脚 CS 连接 PA4 置低电平，然后通过时钟输入引脚 CLK 连接 PA14 并发送 8 个脉冲，最后从数据输出引脚 DOUT 读取结

果。经过运算，将 0-VCC 电压转换为 0-100%的烟雾浓度，通过将 PA13 读取到的数值显示到 OLED 屏幕上。当其检测数据值超过了设定的阈值时，STM32 主控中心就会发出报警信号，这时就会启动风扇。烟雾浓度检测电路如图 3-10 所示。

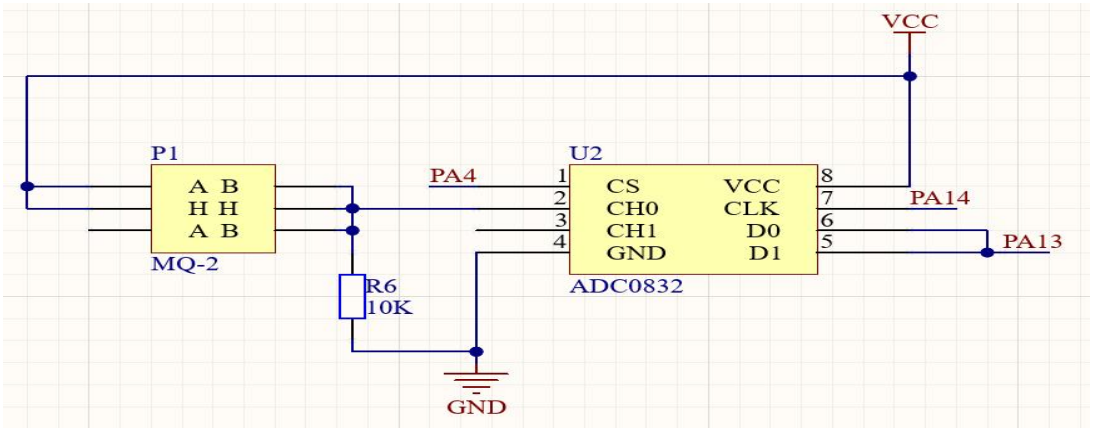

图 3-10 烟雾浓度检测电路

Fig. 3-10 Smoke concentration detection circuit

3.2.6 风扇驱动控制电路

散热风扇的工作原理是按照能量转换的方式进行的，也就是：电能→电磁能→机械能→动能。因为单片机不能直接驱动振动风扇，所以为了控制散热风扇，选用了三极管 9012，其中的电阻是一种限流电阻，起到了限制电流的作用，起到了保护三极管的作用。在相应的 MCU 中，当 MCU 中的相应的控制针为 Low-Low 时，三极管接通，风机运行。风扇驱动控制电路如图 3-11 所示，排针部与 5 V 的风扇电源相连，当风扇通电时，将运行旋转。1 K 的电阻器是电流限制器。通过控制 PA7 管脚的高、低电平，实现了对风扇旋转的控制。当高电平为 1 的时候，三极管打开，风机接地并起动。当输出为零时，表示三极管断开导电，风机未接地，启动停止。

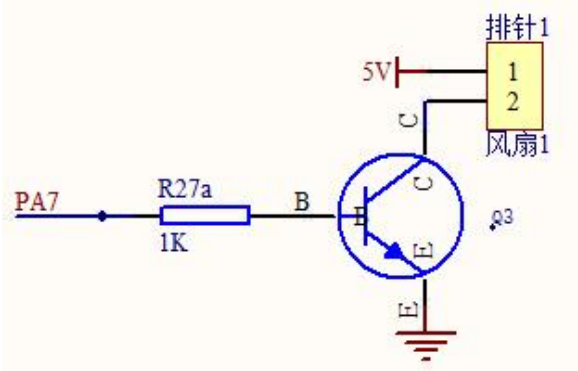

图 3-11 风扇驱动控制电路

Fig. 3-11 Fan drive control circuit

3.2.7 继电器控制电路

继电器属于电力控制装置。该设备有受控体系和受控体系。常用于自控回路，其基本思想是以较低的电流来控制较高的电流动作，等同于“自动开关”” 。因而，继电保护被广泛地用于电路的自动调节，安全保护，变换回路等。

本设计中采用了三个继电器，分别实现空调、窗帘以及电视的开关控制，其电路图如图 3-12 所示当继电模组的绕组通电时，则将 kj3 接点拉入，当没有通电时，则将接点切断。1 K 的电阻器是电流限制器。由单片机分别控制管脚 PB13、PB14、PB15 输出的高低电平，从而实现对继电器的开闭。当高电平为 1 的时候，三极管被接通，保护线圈被激励，接触器被关闭。当高电平为 0 的时候，保护绕组不能被激活，接触器会被切断。

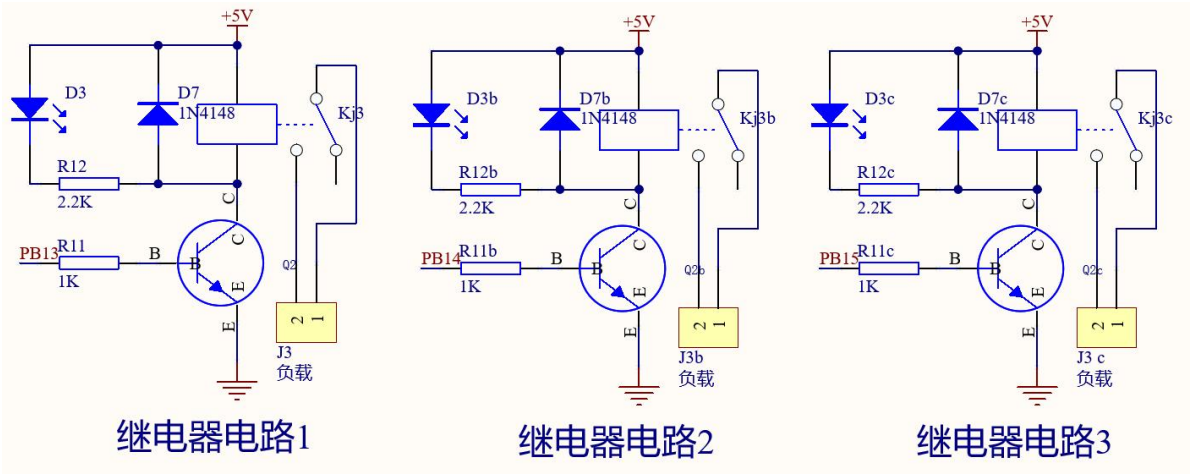

图 3-12 继电器电路控制模组

Fig. 3-12 Relay circuit control module

3.2.8 WIFI 模块电路

本次设计的无线 WIFI 模块使用的是 ESP8266 芯片，在 DIY 和智能家居项目中非常受欢迎。内置了 Wi-Fi 功能，这使得其非常适合用于需要网络连接的智能家居项目。支持多种编程平台和语言，如 Arduino、Lua 等。由于其广泛的应用，网上有大量的教程、项目例子和开源代码，这为初学者和高级开发者提供了丰富的资源。市面上有多种基于 ESP8266 的模块可供选择，例如 ESP-01、ESP-12 等，这为不同的项目需求提供了便利。而且开发者不需要从零开始建立网络通信，因为 ESP8266 已经内置了 TCP/IP 协议栈。可以让用户的电脑和其他设备连接到 Wi-Fi 无线网络，实现

远程通信，从而实现网络连接。

ESP8266 提供三种不同的天线连接方式：板载 PCB，IPEX 天线，印章孔型，用户不需要额外的匹配电路就可以直接连接[32]。若用户要在一块基片上进行天线部件的设计，则可以采用 ESP8266 的小孔天线界面，在此设计中，基片要保留一块基片的匹配电路，具体射频参考电路如图 3-13 所示。

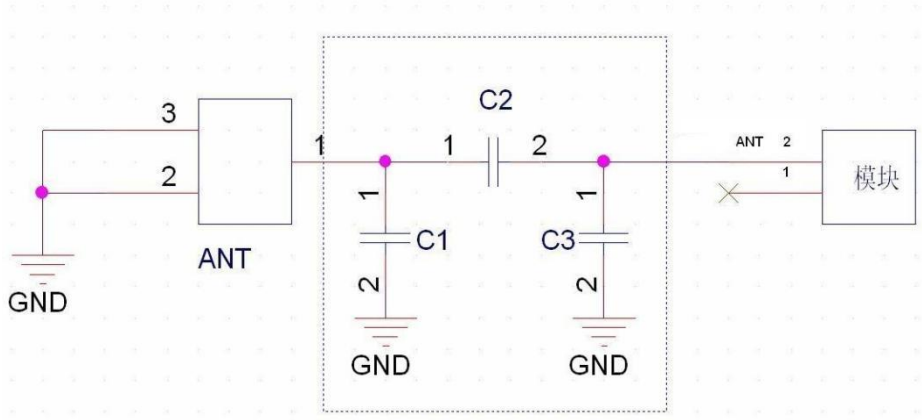

图 3-13 WIFI 射频参考电路

Fig. 3-13 WIFI RF reference circuit

ESP8266 拥有 8 个引脚，各引脚的功能如表 3-5 所示。

表 3-5 ESP8266 引脚功能说明

Tab. 3-5 ESP8266 Pin Function Description

引脚号	引脚名称	功能
1	GND	电源负极
2	TX	串口通信中的输出引脚
3	GPIO2	IO 口
4	CH_PD	模块使能端，高电平有效
5	GPIO0	IO 口
6	RST	重启，低电平有效
7	RX	串口通信中的输入引脚
8	VCC	电源正极

ESP8266 是一个串口控制模块，STM32 单片机通过串口 PA9 和 PA10 与 WIFI 模块进行通信，来控制 WIFI 模块的工作。首先将 WIFI 接入网络，获得 IP 地址，再用 MQTT 协议登陆网络创建设备，而产品、设备创建时，平台为每类产品、每个设备均分配了唯一的 key，设备登录时需要使用通过 key 计算出的访问 token 来进行访问安全认证。只有通过安全认证，才能够订阅 APP，用户才能收到根据订阅的主题得到 APP 的消息。每隔一段时间，每个芯片都会将自己的数据（以 MICROCU 的名义）上传到 ONENET 云，如果 app 上有 MICROCU 的话，那么就会将这些数据传输到 app 上。本设计中 WIFI 模块硬件电路图如图 3-14 所示。

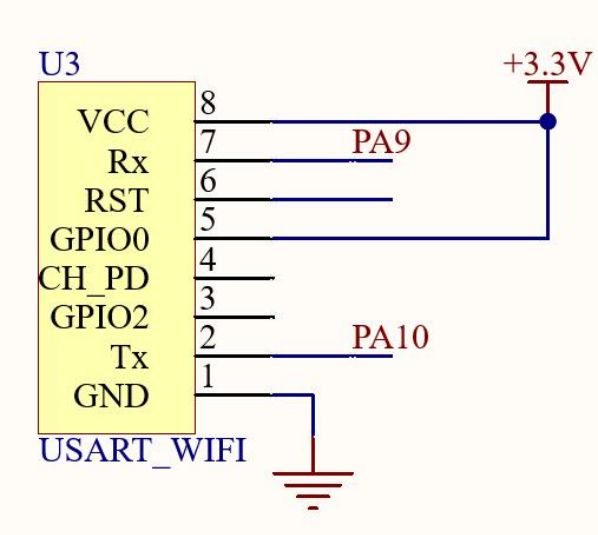

图 3-14 WIFI 模块电路

Fig. 3-14 WIFI module circuit

3.2.9 视频模块电路

本文选择了 ESP32-CAM 开发板卡，该板卡具有诸多优势，其集成了高性能的 ESP32 微控制器和摄像头模块，为开发者提供了一个低成本、低功耗且功能丰富的解决方案。ESP32-CAM 内置了 Wi-Fi 和蓝牙功能，使其能够轻松地与其他设备进行通信和数据传输，而且还具有小巧的尺寸，适合各种紧凑型应用。其内置的摄像头模块使得开发者可以为其设计各种图像处理和视频流应用，如家庭监控、人脸识别和物体检测等。支持各种图像尺寸，包括 VGA、SVGA 和其他更高分辨率的格式。此外，ESP32-CAM 还提供了丰富的 GPIO 接口，允许开发者连接其他传感器或连接单片机，进一步扩展其功能[33]。由于内置了摄像头、微型 SD 卡、 LED 补光灯，所

以这些基础的连接，都是轻而易举的事情。如果有什么人从旁边走过，用感应器触碰一下，就会被拍成照片，储存在记忆卡里，或是上传到网络上。

ESP32- CAM 是安信可公司推出的一款小型相机模块，该模块可作为最小系统单独工作。其体积只有 27*40.5*4.5mm，具有 6 毫安的最小休眠电流。ESP32-CAM 使用了 DIP 封装技术，可以直接插入到底板中进行工作，从而可以实现产品的快速生产，给用户提供了一种高可靠的连接方式，可以很容易地用于各种物联网硬件终端的场合。虽然 ESP32-CAM 提供了一个非常经济的解决方案来实现基于摄像头的项目，但也有一些限制，包括较低的处理功率和内存限制。因此，可能不适合需求非常高的应用。

摄像头通过硬件板子 5V 供电，上电后将会与热点相连。从手机热点中，可以查看到 ESP32 摄像头的 IP 地址，如果手机连接了同一热点，只需要将该 IP 地址输入进去，就可以 APP 或网页上看到实时视频。有一点需要注意由于该软件只能在安卓系统 9.0 以下的版本下看到实时视频，若该手机系统是 9.0 及以上网页会出错。

ESP32-CAM 电路图如图 3-15 所示，由于是已经集成了该模块电路，只需要在该电路板上通电和接地即可。

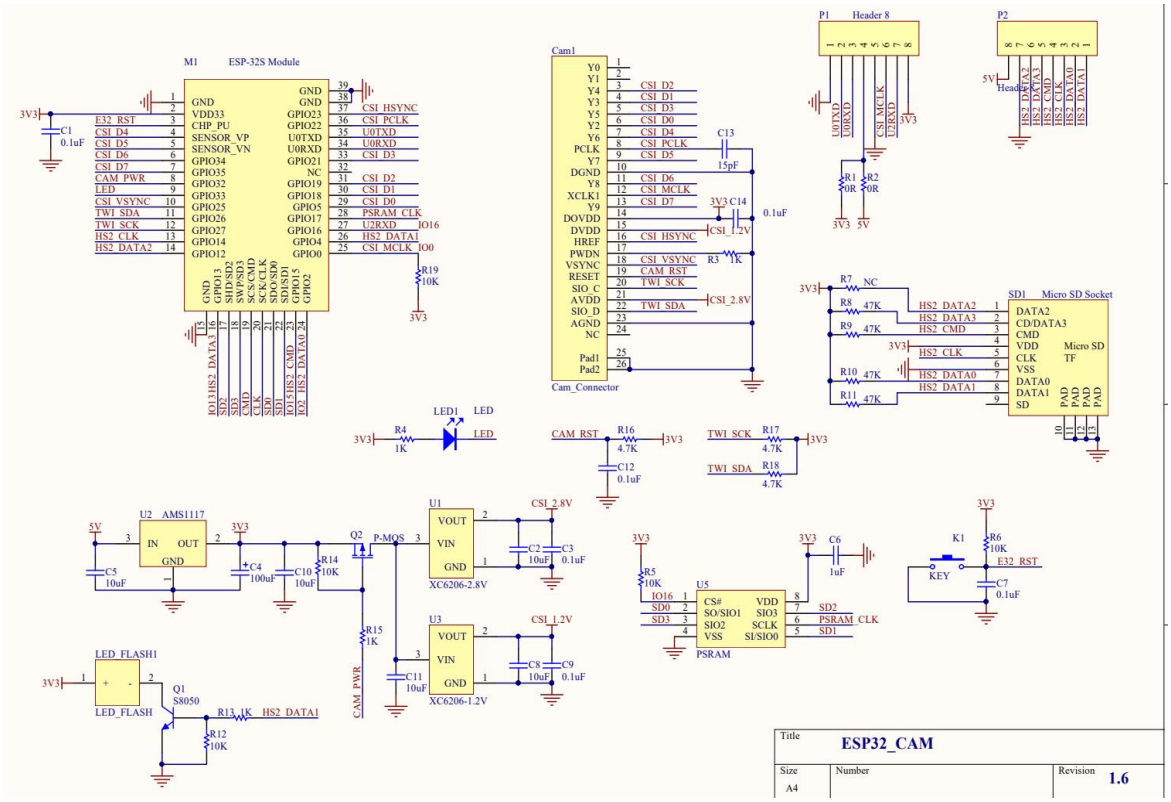

图 3-15 视频模块电路

Fig. 3-15Video module circuit

3.2.10 语音识别模块电路

该系统的语音识别系统使用了一种特殊的“语音识别”芯片—LD3320，具有快速和稳定的优化算法可完成非特定人语音识别。为开发者提供了一个集成度高、性价比优越的音频处理解决方案[34]。LD3320 不仅支持多种语言的语音识别，还能实现高质量的语音合成，使得设备能够与用户进行自然的语音交互。此外，其内置的数字信号处理功能确保了音频信号的清晰度和稳定性。LD3320 还具有低功耗和小尺寸的特点，使其适合于各种便携式和嵌入式应用。无论是为智能家居产品增加语音控制功能，还是为玩具和消费电子产品提供语音交互体验，LD3320 都是一个理想的选择。芯片内包括了语音识别器、模数转换、话筒接口、音频输出接口等外围电路。该芯片可以根据用户需要，对所识别的关键词进行动态修改。

LD3320 语音识别模块是串口输出指令的，将语音识别模块 LD3320 的发送管脚 TXD 接到单片机串口 2 接收端 PA3 口，这样语音模块有指令过来的时候，单片机串口 1 的接收端就可以接收到数据，其设计的电路如图 3-16 所示。

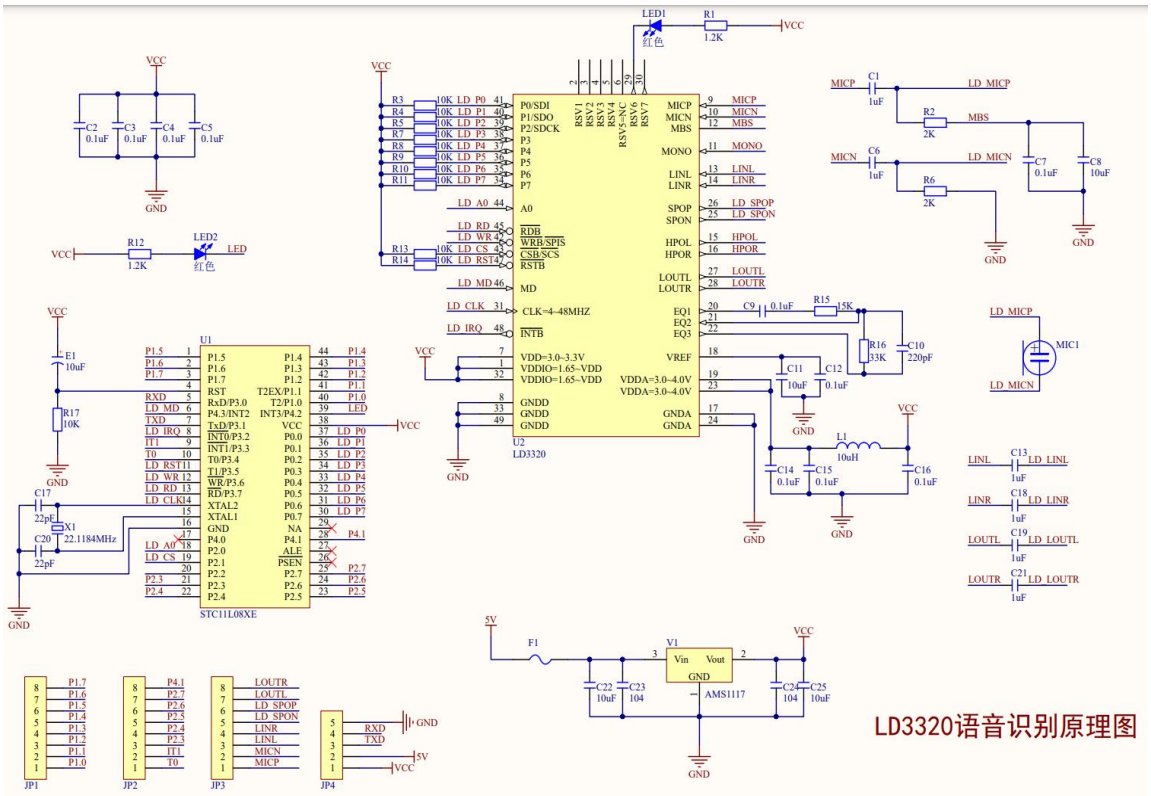

图 3-16 语音识别模块电路

Fig. 3-16 Speech recognition module circuit

3.3 引脚使用统计

针对本章所使用的引脚统一对外设模块所需引脚进行统一分配，可避免一个引脚重复分配到多个外设模块，其引脚使用统计如表 3-6 所示。

表 3-6 引脚使用统计

Tab. 3-6 Pin usage statistics

模块电路	引脚名称	对应 GPIO 口	复用功能
OLED	SCK	PB6	IIC_SCL
	SDA	PB7	IIC_SDA
DHT11	Data	PA2	USART2_TX
光强检测	ADC	PA0	ADC12_INO
风扇	DATA	PA7	SPI1_MOS1
烟雾检测（MQ-2）	CS	PA4	USART2_CK
	CLK	PA14	SWCLK
	SDA	PA13	SWDIO
继电器	DATA	PB13	GPIOPB13
	DATA	PB14	GPIOPB14
	DATA	PB15	GPIOPB15
WIFI	TX	PA9	USART1_TX
	RX	PA10	USART1_RX
LD3320	RX	PA3	USART2_RX

3.4 本章小结

本章介绍了主控制器芯片的选取并说明选取的原因，接着根据智能家居所需要完成的功能分模块对各个电路进行了解释说明，根据相应的模块电路完成总体硬件部分电路的设计，为后面第四章的软件设计及 IO 口的设计，功能的调用奠定了基础。

4 系统软件设计

4.1 单片机软件程序设计

4.1.1 Keil5 软件介绍

Keil5 是 KEIL Software 公司专为单片机设计的程序开发工具，基于 ARM 架构，特别适用于 Cortex—M、Cortex—R4、ARM7 和 ARM9 处理器设备。这款软件提供了一个完整的开发环境,专为微控制器应用而设计[35]，包括μVision IDE、调试器和 Arm C/C++编译器，支持 C 语言开发。Keil5 的优势在于其广泛的微控制器支持和丰富的中间件组件，使得开发过程从原型设计到产品发布都更为高效。

STM32 系列均采用 ARM Cortex 结构，使得 Keil5 与前一代 Keil4 相比具有更高的兼容性。但是在 Keil4 的开发过程中，有许多的类库文件被整合在一起，这样就可以方便地被调用。但是，Keil5 并不具备这样的特性。因此，开发者在需要使用某个类库文件时，需要自己安装这个类库文件，以便能够正常地使用。在这个例子中，Keil5 在性能上似乎要逊色于 Keil4，但是，因为开发人员添加了更多的类库，所以其性能远超 Keil4。

Keil5 MDK 集成了软件系统开发的全套工具，包括但不限于程序的编写、编译和调试。不仅允许开发者顺利完成代码的编写工作，还可以通过内置的检测工具来验证代码的正确性[36]。其强大的仿真功能可以有效检验代码错误，同时支持单步或多步检验代码，并能烧录进单片机进行硬件功能检测。当代码编写和验证完成后，可以利用下载器将其连接到单片机系统进行在线调试，确保其在实际硬件上的可行性和正确性。通过使用 J-LINK，可以将程序烧录到 STM32 上进行在线仿真，并通过单步和断点进行调试。

在成功编译系统软件后，该软件会将所编写的程序转化为。hex 目标文件。接着，利用编程工具将其传输到 STM32F103C8T6 微控制器内。完成这一步后，将微控制器安装到 PCB 板的专用插槽中，并为其提供电源，从而实现程序的下载[37]。在下载的过程中，如果遇到下载失败的情况，首先要检查串口设置和所选的微控制器型号是否匹配；如果上述设置都正确，那么建议断开微控制器的电源，然后再次上电以进行下载。下图 4.1 为 keil5 软件编译成功后的界面。

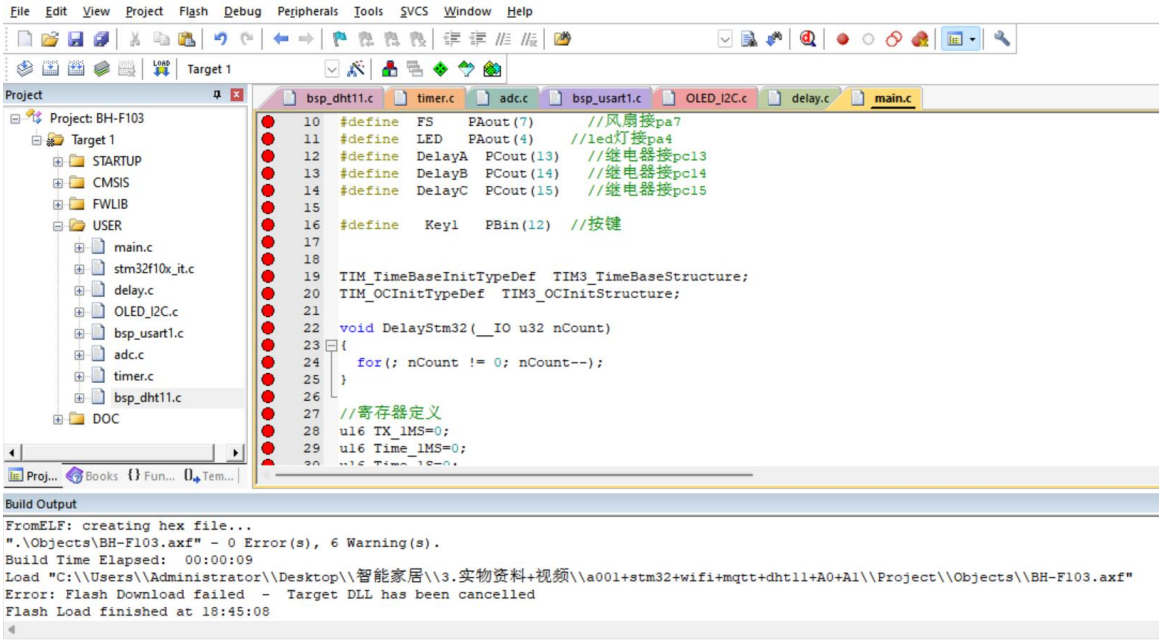

图 4-1 软件编译成功界面

Fig. 4-1 Software Compilation Success Interface

4.1.2 STM32 主程序流程设计

在程序开发中，传统的顺序编程方式能直观地反映开发者的思维，使得读懂代码就能理解其背后的逻辑。但这种方法在多项目环境下缺乏移植性，导致重复劳动和低效率。现代开发趋向于强调代码的可读性和可移植性，其中模块化和结构化设计思想应运而生。模块化开发将常用功能封装为特定模块，避免重复编写当需要使用到该功能的时候，只需调用该模块即可，而结构化开发则将面向对象理念融入其中，每个模块都具有特定功能，在将来的开发过程中，如果开发人员要使用到对拥有特定特性的对象进行赋值、读取等操作，就可以直接使用该模块，增强了代码的可移植性。尽管结构化开发在单项目中可能显得繁琐，但其长远优势在于提高代码复用率和简化项目分析[38]。在本次设计中，采用了模块化思想，以提高效率和降低代码重复性。

主控中心是整个系统的核心，既可以通过网络与远程手机客户端进行通信，又可以通过传感器对家庭联网电器进行控制，并对环境数据进行采集。另外，在烟气浓度较高的情况下，主控制中心也会开启风扇。使用者也可以利用手机 APP 客户端，来实时地了解到家庭环境中的信息变化，窗帘、电视、空调和 LED 灯的状态都会被实时地显示在 APP 上，还可以对 APP 进行相应的操作控制。STM32 主程序流程图

如图 4-2 所示。

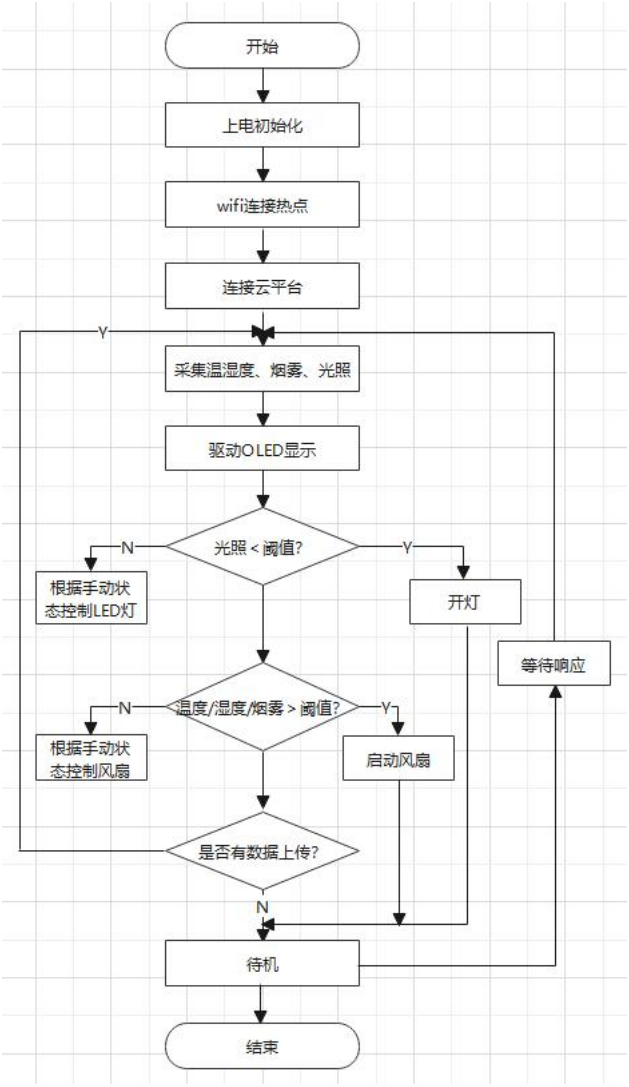

图 4-2 STM32 主程序流程图

Fig. 4-2 STM32 main program flowchart

4.1.3 温湿度检测模块流程设计

当温湿度传感器 DHT11 初始化工作结束时，温湿度传感器就会以一种极快的速度对其进行信息的采集，并对其做出相应的反应。将所采集到的湿度温度信息以 40bit 的数据的形式发送出去，并对其进行读出，同时读出的数据通常会通过串口显示进行转换在 OLED 屏幕上，而在读出完毕之后，DHT11 传感器便会自动地转入低压状态，并将其返回[39]。而 DHT11 有时可能会返回无效的数据，因此在代码中可添加错误检查机制。如果读取的数据无效，可以尝试重新读取或显示错误消息。

温湿度数据采集程序流程图如图 4-3 所示。

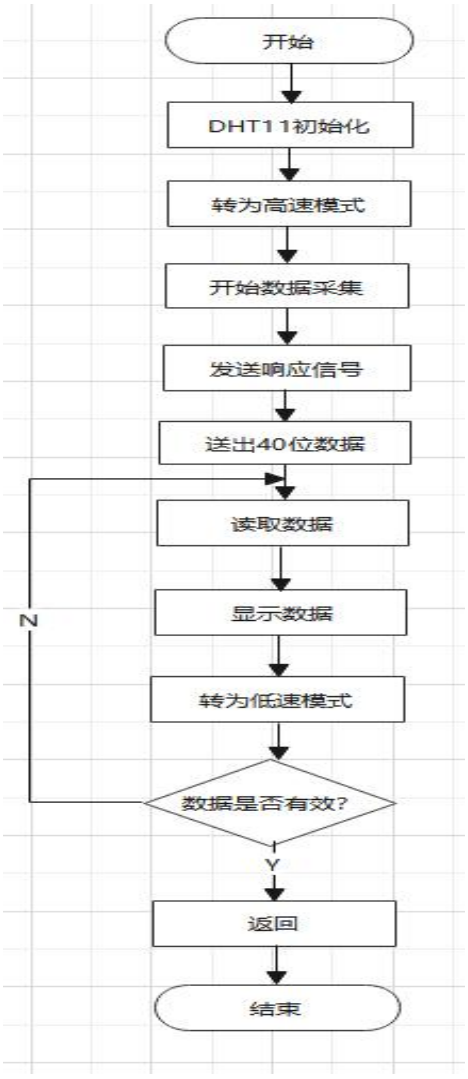

图 4-3 温湿度数据采集程序流程图

Fig. 4-3 Flow chart of temperature and humidity data collection program

4.1.4 显示模块流程设计

在本设计中的显示模块扮演了一个关键的角色，负责将关键信息呈现给用户。为了实现这一目标，选择了 OLED 显示屏，这种显示屏以其 128x64 的点阵分辨率为特点，工作原理相对直观：每个像素点可以独立控制，当像素值为 1 时，该像素点会发光；而当像素值为 0 时，该像素点则不发光。这种二进制的控制方式使得图像显示既简单又直观。值得注意的是，OLED 的数据组织方式是按列存储的，且每一列对应一个字节，低位在最上面，高位在下面，从左到右依次写入命令[40]。

先初始化 OLED 屏幕，并发送一系列的命令来启动和配置屏幕。在编写显示数

据时，首先需要确定要显示的内容，然后决定这些内容在屏幕上的具体位置。一旦定义了要显示的内容，而这些内容可以是简单的文本、图形或自定义的像素阵列就需要发送一个命令来更新屏幕。通常是通过一个名为 display（）或类似名称的函数完成[41]。这个函数会将定义的所有内容发送到 OLED 屏幕上。而如果数据序列有误，必须进行调整以确保显示内容的准确性。由于 OLED128x64 的分辨率限制，可能需要进行一些调整和优化，以确保数据在屏幕上正确显示。这可能涉及到调整字体大小、图形位置或像素颜色等。显示模块流程图如图 4-4 所示。

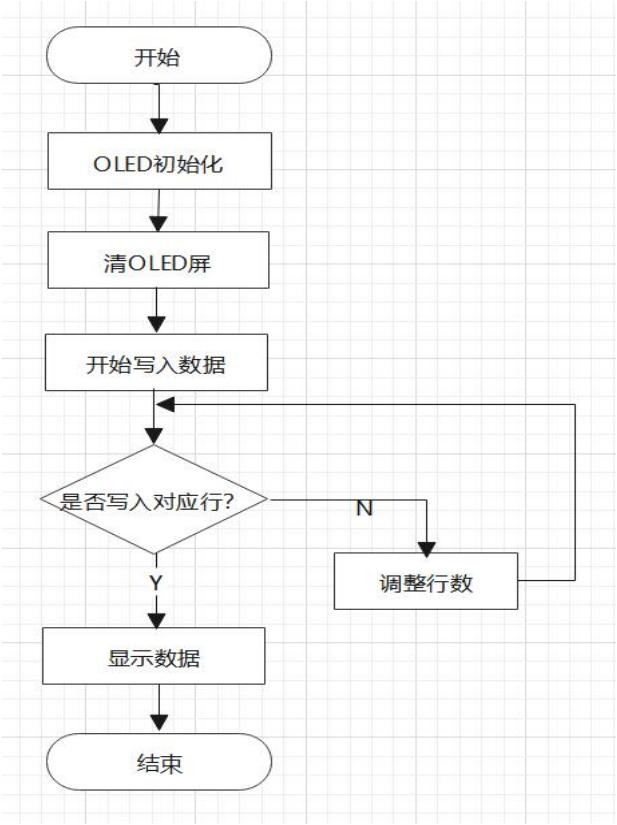

图 4-4 显示程序流程图

Fig. 4-4 Display program flowchart

4.1.5 WiFi 串口通讯模块流程设计

无线传输技术在数据交换中经常采用串行接口的中断机制来确保数据的准确性和完整性。当检测到中断信号时，系统会立即关闭该中断，避免因数据信号的连续波动导致的多次中断，从而确保数据传输的稳定性。为了进一步优化数据处理，指令会被解码成相应的二维数组格式，这种结构化的数据形式使得主程序能够更加高效地调用相关功能。特别是在 WIFI 串口数据接收过程中，系统首先会检测是否存在

连接断开的情况。一旦检测到连接中断，系统会立即激活数据缓冲区，开始接收待处理的数据。数据接收完毕后，通过特定的算法进行分析处理，或根据有效的控制指令进行相应的操作。这种综合利用中断管理和数据结构化处理的方法，不仅提高了无线传输的效率，还确保了数据的安全性和准确性。

ESP8266 采用串口方式与 STM32 进行通信，首先要初始化 ESP8266 模块。可发送 AT 命令来配置 ESP8266 的工作模式、连接到指定的 Wi-Fi 网络、设置 IP 地址等。一旦 ESP8266 成功连接到路由器，STM32 可以通过串口查询 ESP8266 的状态，进行判断是否有数据需要接收或发送[42]。若 ESP8266 接收到来自服务器的数据，会通过串口将这些数据发送到 STM32。STM32 需要解析这些数据，并根据应用的需求进行相应的处理。若没有将数据发送到串口则是发送到服务器，可以通过串口发送指令给 ESP8266，指示其将数据发送到指定的服务器地址。ESP8266 串口通讯流程图如图 4-5 所示。

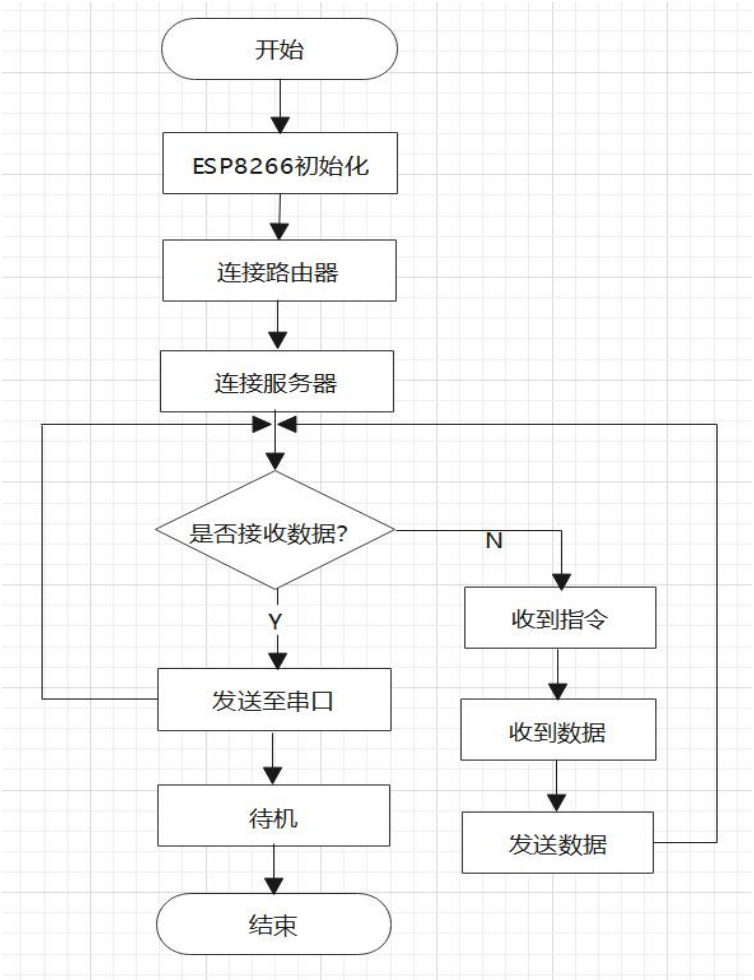

图 4-5 ESP8266 串口通讯流程图

Fig. 4-5 ESP8266 serial communication flowchart

4.1.6 语音识别模块流程设计

语音识别芯片 LD3320 是一款语音识别芯片，其稳定性和可靠性对于确保语音识别的准确性至关重要。在其工作过程中，系统首先会对芯片进行状态检查，确保其正常运行。若芯片出现异常或故障，如无响应信号，系统会从语音模块获取中断请求，并在相应的中断服务程序中设置特定标志位[43]。这个标志位是芯片健康状态的关键指示器。如果标志位不为 0，这意味着芯片可能出现了问题，没有正常响应；而标志位为 0 则表示芯片正常工作。为了确保系统的稳定运行，一旦检测到芯片的异常状态，系统会立即启动复位程序，将芯片恢复到初始状态。同时，语音模块也会对应地进入复位状态，确保整个系统的同步和协调。这种自我检测和自动恢复的机制大大增强了 LD3320 在各种应用场景中的稳定性和可靠性。流程图如图 4-6 所示。

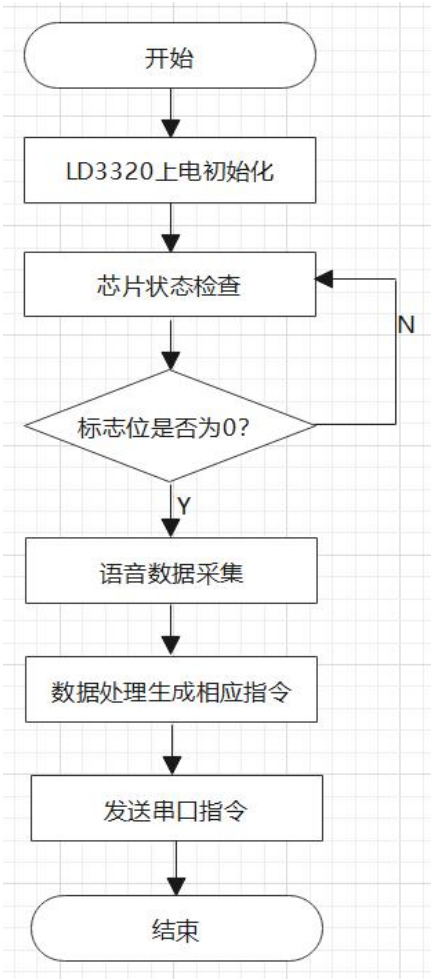

图 4-6 语音识别子程序流程图

Fig. 4-6 Speech recognition subroutine flowchart

4.2 WIFI 通信模块与 ONEENT 云平台

在本设计中，为降低自主搭建服务器的难度和节省开发成本，选择通过 ESP8266WIFI 通信模块连接 ONENET 云平台，从而实现对相关数据的实时通讯，运用 ONENET 云平台可以实时进行摄像头数据的采集[44]。首先在官网注册账号，注册成功后打开 ONENET 云平台官网的首页控制台[50]。云平台控制台如图 4-7 所示。

图 4-7 云平台控制台

Fig. 4-7 Cloud Platform Console

然后在基础服务中选择多协议接入。点击添加产品，输入相关参数后点击确定，之后点击“添加设备”，分别添加“MCU”设备以及“APP”设备，其中“MCU”是单片机创建的设备，“APP”是手机创建的设备。云平台设备列表如图 4-8 所示。

图 4-8 云平台设备列表

Fig. 4-8 Cloud Platform Device List

使用串口调试助手对串口端进行配置，然后选择自己的设备 ID，与管理员 APIKEY 进行匹配，在完成匹配之后，ESP8266WIFI 通信模块就被绑定到了 ONENET 云平台上，之后就可以使用该设备来访问 ONENET 云平台[45]。

通过 MQTT Simulate Device 和 MQTT.fx 两个软件用来模拟两个设备，从而实现上位机与下位机通讯传输。在 MQTT Simulate Device 中的 DeviceID 输入 1043512515 其为设备 ID 中的 mcu 模块，ProductID 中输入 570894 其为产品 ID，AuthInfo 输入 mcu 其为鉴权信息，用于设备登录参数之一参与设备登录鉴权，SeverAddr 输入 183.230.40.39 其为 IP 地址，Port 输入 6002 其为端口地址。在 MQTT。fx 中的 Broker Address 输入 183.230.40.39 其为 IP 地址，Broker Port 输入 6002 其为端口地址，Client ID 输入 1043512473 其为设备 ID 中的 app 模块，User Name 输入 570894 其为产品 ID，Password 输入 app 其为鉴权信息。下图 4-9 为软件间进行模拟通讯传输。

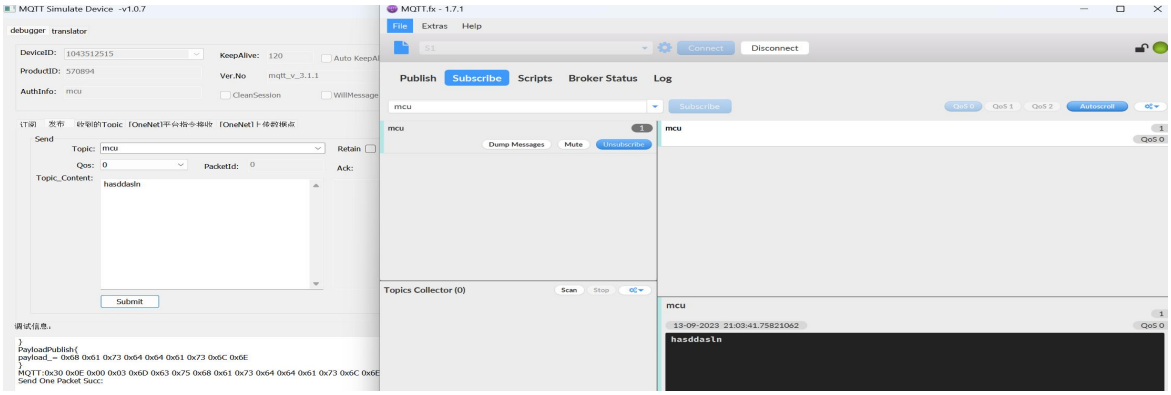

图 4-9 软件间进行模拟通讯传输

Fig. 4-9 Simulated communication transmission between software

4.3 手机 APP 软件设计与实现

4.3.1 JDK 安装与配置

在进行 app 编写之前，需要配置 JDK（Java Development Kit）。Android 操作系统的应用层大部分是用 Java 编写的，需要 JDK 来运行和执行某些开发任务。提供了运行 Java 应用所需的类库和其他资源，使得开发者可以轻松地创建功能丰富的应用。

在官网下载安装完成后需要进行环境的配置，通过控制面板打开高级系统设置，选择环境变量，点击“新建”，新建系统变量 JAVA_HOME，其中“值”为 JDK 安装根目录。通过编辑 PATH 变量，将刚新建的 JAVA_HOME 变量加上 bin 目录设置到 PATH 中[46]。

之后打开 cmd 命令行，进行 JDK 的配置验证，在命令行中输入 Java-version，下图 4-11 是 Java-version 验证成功。

图 4-11 Java-version 验证成功

Fig. 4-11 Java-version verification successful

4.3.2 手机 app 的软件设计

在设计过程中，需要面临一个选择难题，基于 IOS 系统还是基于 Android 系统进行程序的开发。据 2022 年全球智能手机的总出货量进行对比分析，IOS 系统的苹果仅有 19%的份额[47]，Android 系统因为开源的特性，基于 Android 系统的进行开发的是满足绝大多数人的需求的。因此手机 APP 的开发是在 Android Studio 平台中完成的，采用 Java 语言开发完成。

Android Studio 开发流程如下所示。

首先，创建 APP 项目：

（1）打开软件，在菜单中选择 file-》new project 打开创建向导。

（2）配置项目，确定各个项目名称和存放项目存放路径，如下表 4-1 所示；

表 4-1 项目名称和存放路径

Tab. 4-1 Project Name and Storage Path

名称	注释
Application name	项目名称
Company Domain	域名
Package name	app 打包名称
project location	存放路径

（3）设定兼容的安卓的最小版本，默认情况下为 API21 即 Android5.0 系统的。

（4）确定最小兼容版本后，选择一个模板来开始应用（例如 "Empty Activity"）。

（5）设定活动名称，界面布局的名称，以及界面标题，最后点击 finish 完成项目的创建。

（6）创建完后，在项目名》app》src》main》res》layout 下双击 xml 文件即为打开活动界面设计窗口[48]。

然后，进行代码的编写。主要的代码文件通常位于 app》java》 your.package.name》MainActivity.java。双击 MainActivity.java 并打开编写 Java 代码。而 Android 的界

面是通过 XML 文件定义的。可以在 app》res》layout 目录下找到这些文件。双击 activity_main.xml 打开，直接编辑 XML 来设计应用界面[49]。

新建工程项目后 Android Studio 的 Product 目录结构如下表 4-2 所示。

表 4-2 Product 目录结构表

Tab. 4-2 Product Directory Structure Table

名称	注释
idea	Android Studio 生成的工程配置文件
app	Android Studio 创建工程中的一个 Module
gradle	构建工具系统的 jar 和 wrapper 等
External Libraries	不是一个文件夹，只是依赖 lib 文件，如 SDK 等

新建工程项目后 Android Studio 的 Module 目录结构如下表 4-3 所示。

表 4-3 Module 目录结构表

Tab. 4-3 Module directory structure table

名称	注释
build	构建目录，编译生成的 apk 也在此目录的 outs 子目录
libs	依赖包，包含 jar 包和 jni 等包
src	源码，相当于 eclipse 的工程
main	主文件夹
java	Java 代码，包含工程和新建是默认产生的 Test 工程源码
res	资源文件
layout	App 布局及界面元素配置
menu	App 菜单配置
Dimens.xml	定义 css 的配置文件
Strings.xml	定义字符串的配置文件
Styles.xml	定义 style 的配置文件
AndroidManifest.xml	App 基本信息（Android 管理文件）
ic_launcher-web.png	App 图标
Build.gradle	Module 的 Gradle 构建脚本

代码编写完成后，下图 4-12 为软件进行调试测试部分。

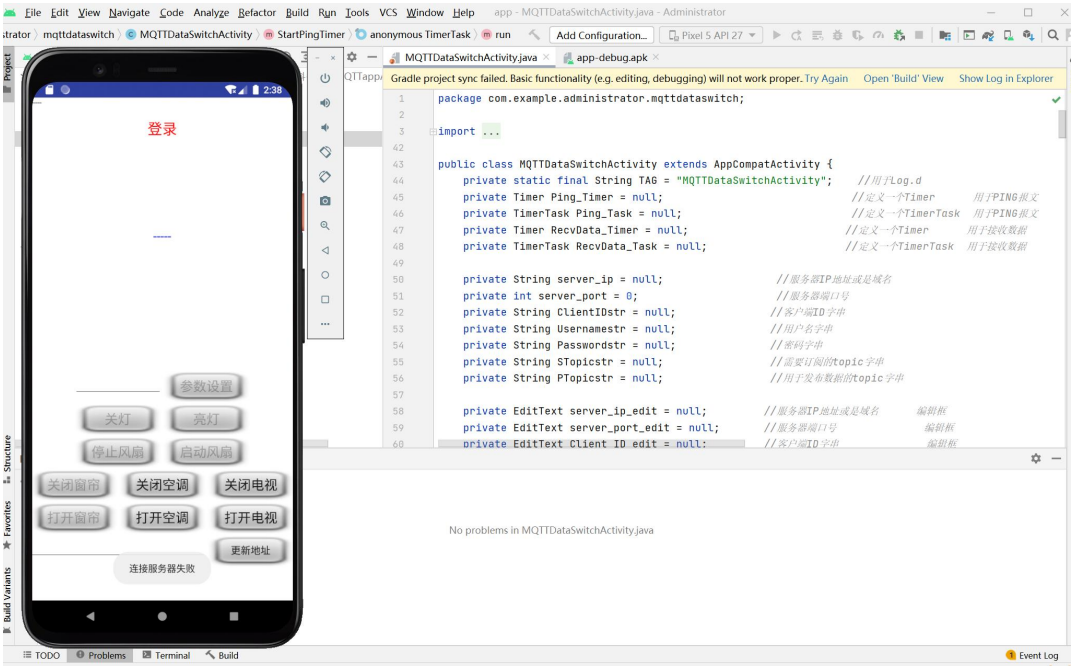

图 4-12 软件调试

Fig. 4-12 Software Debugging

4.3.3 手机 app 的调试

在完成代码的编写后，进行 APK 文件的安装调试，打开 APP 后，首先需要对登录里面的链接信息进行设置，如图 4-13 所示。

图 4-13 设置链接信息

Fig. 4-13 Set Link Information

填入相应的服务器 IP/域名，对服务器端口号以及 ONENET 云平台中的客户端 ID 进行填写，包括用户名、密码以及订阅和发布的 Topic。完成配置设置后，在手机

操作界面对系统进行调试[50]。手机操作界面如图 4-14 所示。

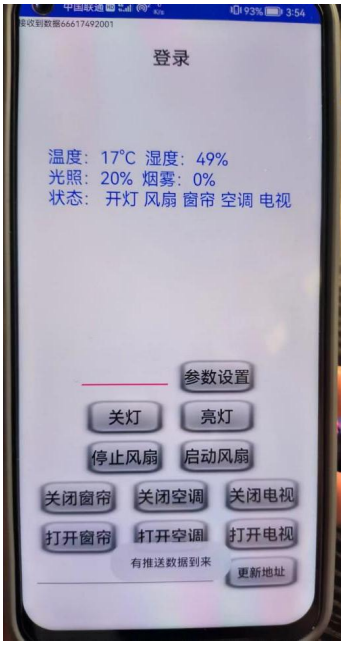

图 4-14 手机操作界面

Fig. 4-14 Mobile operation interface

4.4 本章小结

本章主要通过介绍智能家居软件方面的情况，根据上一章节硬件部分所描述的功能进行程序的编写，详细介绍了 ONENET 云平台的使用原理，并通过 Android Studio 进行手机程序 APP 的编写，通过编写调试，验证软件的可行性。通过手机安装进行调试运行。

5 系统调试

5.1 系统实物图

根据上述的硬件电路设计进行实物的焊接与程序的下载，制作完成的实物正面图如图 5-1 所示，实物背面图如图 5-2 所示。

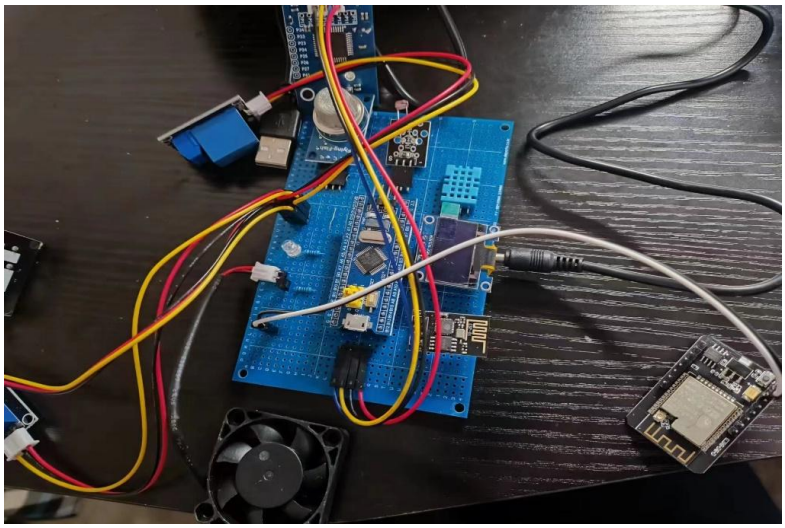

图 5-1 实物正面图

Fig. 5-1 Physical front view

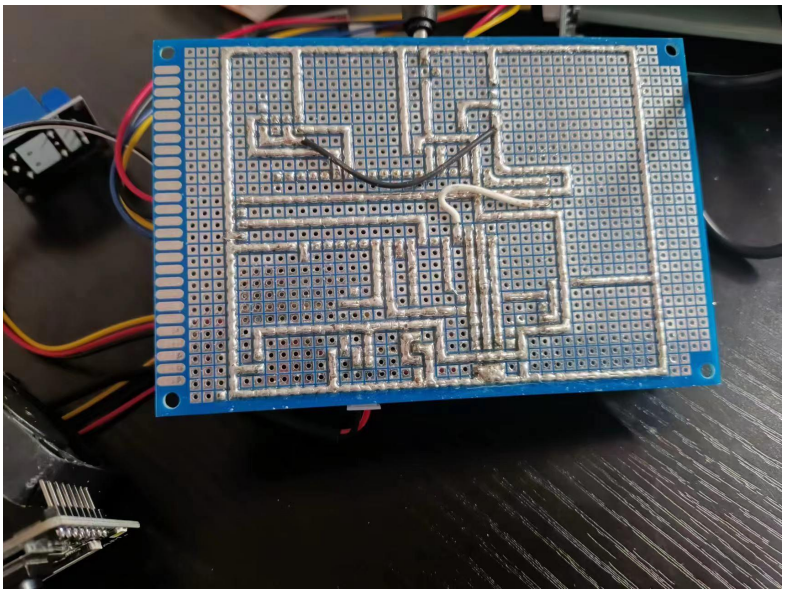

图 5-2 实物背面图

Fig. 5-2 Physical back view

5.2 功能测试

首先，对系统进行上电初始化，如图 5-3 所示。

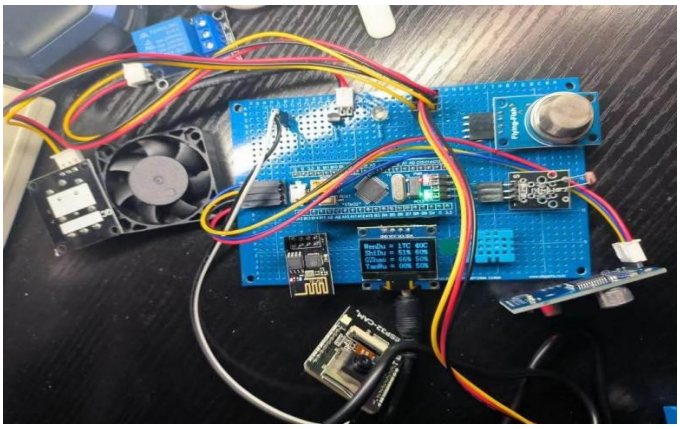

图 5-3 上电初始化

Fig. 5-3 Power on initialization

其功能说明如下：初始设置的温度为 40，湿度为 60，光照为 50，烟雾为 50。温度调节 t；湿度调节 h；光照调节 g；烟雾调节 m。大于阈值风扇会自动启动，这时候无法通过语音和 APP 控制风扇的开关，开机若低于光照的阈值，LED 灯会启动，这时候也无法通过语音和 APP 控制 LED 灯的开关，低于阈值风扇会关闭，当大于光照阈值时，LED 灯熄灭，这时候可以正常开启关闭 LED 灯。

对于烟雾浓度检测，可用打火机释放可燃气体丁烷来进行模拟火灾发生时候的场景，当烟雾传感器检测到环境周边的可燃气体丁烷高于设置的阈值时风扇启动[51]，随着打火机停止释放丁烷后，传感器检测到低于阈值时，风扇会关闭，如图 5-4 所示。

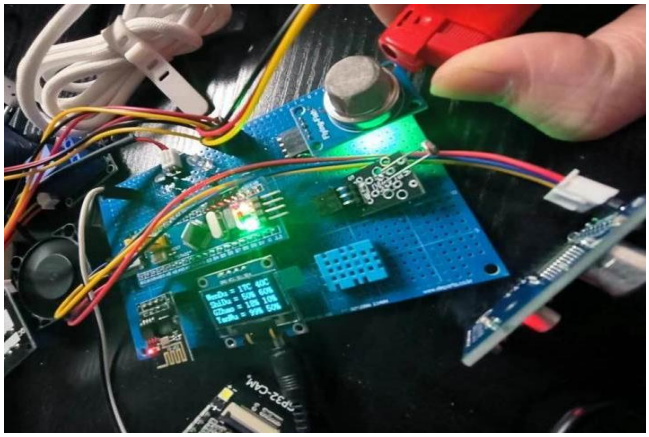

图 5-4 烟雾浓度高于阈值风扇启动

Fig. 5-4 Smoke concentration above threshold fan start

对于温度检测，可用吹风机加热的方式来进行实验，随着吹风机加热通电后周围的温度随着升高，当温度高于阈值时，风扇启动，而随着吹风机断电温度随着降低，低于阈值时，风扇会关闭，如图 5-5 所示。

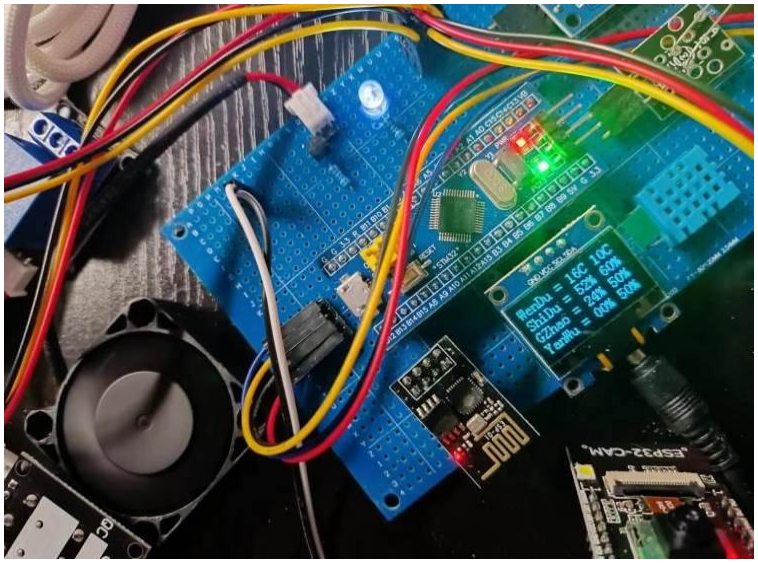

图 5-5 温度高于阈值风扇启动

Fig. 5-5 Temperature above threshold fan start

对于温度检测，可用嘴吹气的方法来改变环境的湿度。当用嘴吹气时，会使周边环境的湿度升高当湿度高于阈值风扇同样会启动，而不用嘴吹气时，会使得湿度逐渐低于阈值，会使风扇关闭，如图 5-6 所示。

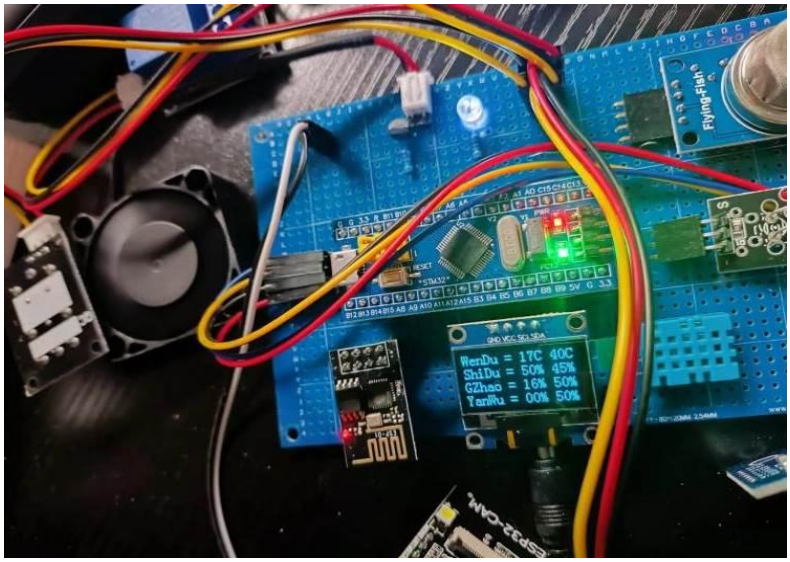

图 5-6 湿度高于阈值风扇启动

Fig. 5-6 Humidity above threshold fan start

对于光照检测，可用手机摄像头闪光灯来改变环境的光照。当用闪光灯时，会使周边环境的光照变高，当光照高于阈值时，LED 灯熄灭，而随着闪光灯关闭时会使得光照低于阈值时[52]，LED 灯会打开，如图 5-7 所示。

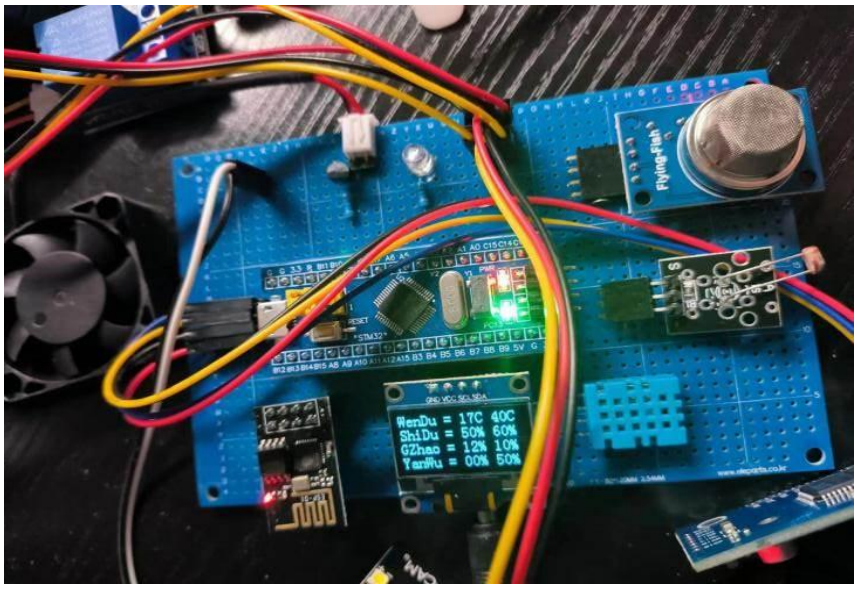

图 5-7 光照高于阈值 LED 灯熄灭

Fig. 5-7 Humidity above threshold fan start

对于摄像头模块，首先需要进行摄像头 IP 地址的选取，上面 ESP-8129BB 是 WiFi 模块的 IP 地址，下面 esp32-arduino 是摄像头的 IP 地址如图 5-8 所示。

图 5-8 IP 地址的选取

Fig. 5-8 Selection of IP address

接着在 APP 的最下面输入摄像头的 IP 地址 http://192.168.43.48，就可以实时读取当前图像，移动 ESP-8266 摄像头模组，图像会随着发生改变，如图 5-9 所示。

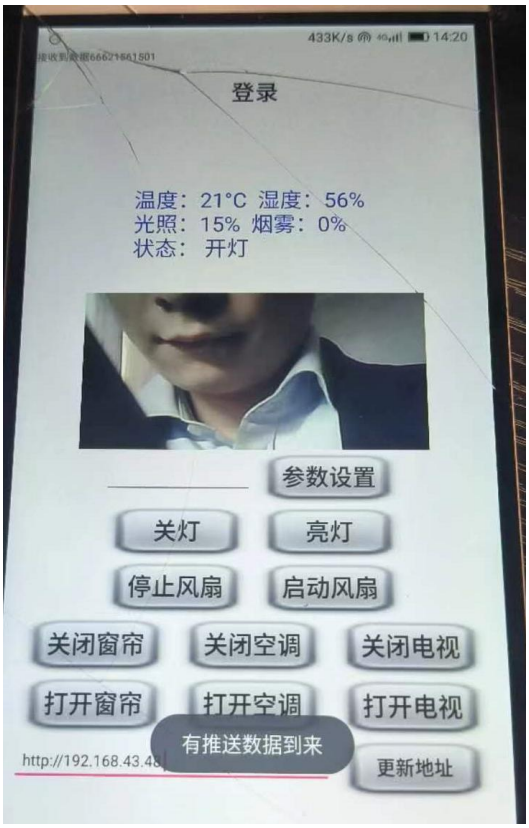

图 5-9 摄像头实时读取图像

Fig. 5-9 Real time image reading by camera

5.3 本章小结

本章首先对实物进行正面和背面的展示，其次对于所实现的功能进行单项测试，通过对摄像头的 IP 地址进行选取，验证其监视的功能，通过软硬件连调以及多次测试，证明了智能家居控制系统的稳定性及可行性。

6 总结和展望

6.1 本文的主要成果

随着智能家居市场的逐渐扩大，无线通信扮演者越来越重要的角色，作为智能家居控制系统的一个重要组成部分，合理进行通信方式的选取，能为智能家居走向千家万户打下坚实的基础。

智能家居的研究与开发涉及到电子，自动化，通讯三个方面。智能家居控制系统可以实现对家庭环境的远程监控，可以通过安装传感器和摄像头，实时采集家庭内的温度、湿度、采光等数据，并且可以根据这些数据进行调整，从而保障家庭的安全。智能家居系统可以为人们提供一个安全、舒适、便捷的家庭环境，让他们在家中享受到更加安全、舒适的生活体验。智能家庭是利用无线通讯将家庭内的各种与资讯有关的装置联结在一起，并对其进行集中的管理与控制。相对于一般的居住区环境，智能化的居住环境更加健康，方便及舒适，极大程度的提升了人们的生活品质。

本论文的主要工作重点如下:

第一，介绍了智能家居的研究背景，给生活带来便利，在结合国内外研究情况，展望未来发展的基础上，形成了这篇论文的设计思路，设计一个基于 APP 控制的智能家居控制系统。

第二，分析比较论证得出一个具体的设计方案选择 WIFI 技术作为系统主要的网络通信技术方案，选定 Android 作为控制终端的操作系统。设计了由控制终端、服务器、中央控制器和子控制器四部分组成的无线智能家居系统的总体方案。

第三，根据系统所要实现的功能，采用电路模块化设计的方式，画出了电路原理图，完成以 STM32 单片机控制的硬件电路。

第四，根据硬件功能用 Keil5 写出了所对应的软件并通过 JLINK 写入单片机中。介绍了移动端 APP 项目流程并完成了 Android 系统应用的开发。

第五，根据功能进行逐项调试，成功实现了基于 Android 的无线智能家居系统的设计，实现了预期的功能设计。

6.2 创新之处

随着云平台的出现，智能家居系统的设计可以有效地实现超长距离的数据传输，这一技术的发展为人们提供了一种新的可能性，使得智能家居系统的设计可以更加有效地满足用户的需求，让科技的便利普及到千家万户。

通过利用 ONENET 云平台，本系统实现了本地控制系统与 Android 客户端之间的数据交互，实现了远程监测和控制功能;此外，为了提升用户体验，还开发了一款手机 APP，使得用户无论在何时何地都可以轻松访问本地家居系统。

6.3 研究展望

经过多日的研究，设计的智能家居系统已经具备了智能控制的基本要求，但仍有一些不足之处，需要进一步改进，以满足智能家居系统所需的广泛知识和巨大工作量的需求。

1. 由于时间的关系，没能做出一个红外检测物体的模块，这可以用来防盗，这是可以下一步进行改进的地方。

2. 本文的智能家居系统缺乏软件安全，没有进行一个用户的注册，只需要连接特定的 IP 就能实时接收到数据。

3. 硬件方面的选取对于外界环境未能做到很好的抗干扰性，所选器件都对环境有要求，对于该系统的精确度不够。

4. 在系统功能方面，当传感器检测到环境变化超过阈值时，系统这时候只会启动风扇来处理。在实际当中，可以通过蜂鸣器报警和发送短信来通知用户。

参考文献

[1] 朱利娟.基于 STM32 的智能家居控制系统设计与研究[D].西藏：西藏大学，2018.

[2] 袁明兰，龙颖．基于环境感知的智能家居用户体验实验系统设计[J]．科技风，2019(27)：13．

[3] Yan Changshun,Shao Yong. Research on Security Evaluation Indicator System of Smart Home[J]. American Journal of Networks and Communications,2020,23-26.

[4] 金纯用．基于 iOS 的智能家居系统设计与实现[D]．广西：广西师范大学，2022．

[5] 郭诗霖，江业峰，侯俊博，魏英楠．基于 Linux 与 ARM 架构的嵌入式智能家居系统[J]．软件，2022，43(05)：125-127．

[6] 黄科军． AMOLED 的驱动控制电路设计[D]．四川：电子科技大学，2004．

[7] 高丽娜，王晓阳，李昊霖，刘伟丽．基于嵌入式的车内空气质量检测仪的设计与实现[J]．南方农机，2022，53(10)：20-23．

[8] 王愿祥，程悦琪，孙先松．基于 WiFi 的无线测控终端系统设计[J]．物联网技术，2018，8(09)：23-26．

[9] 张俊朋，盛象飞．基于单片机的智能花盆系统设计[J]．电子世界，2016(12)：38．

[10]娄丽萍．基于 dsPIC30F 的移动式太阳能自动跟踪系统的设计[D]．天津：天津大学，2012．

[11]付孟林，姚圣男，殷倩如，张福鼎．基于 Arduino 的分布式智能家居安防系统设计[J]．电子世界，2019(09)：88-89．

[12]王雪英.基于 Android 系统与 ZigBee 技术的智能家居系统设计与实现[D].山西：山西大学，2018．

[13]陈淡宁.基于物联网的智能家居环境监测系统的研究[D].吉林：吉林大学，2014，33-34.

[14]Flores Miguel,Heredia Diego,Andrade Roberto,Ibrahim Mariam. Smart Home IoT Network Risk Assessment Using Bayesian Networks[J]. Entropy,2022,24-25.

[15]李诚．互联网科技趋势下智能家居设计研究[D]．广东：广东工业大学，2021．

[16]唐俊涛，刘谦．基于 ESP8266WiFi 模组的智能开关[J]．数字技术与应用，2022，40(02)：186-188．

[17]王磊，何勇，张宇. 智能语音控制系统的设计与实现[J]. 计算机测量与控制，2018，26(02)：109-112.

[18]Mart Home Energy Management Segment To Grow By 30% In 2021 – $8B Revenue Globally[J]. M2 Presswire,2021,41-43.

[19]孟锦涛．基于手机 APP 的智能家居系统设计[D]．安徽：安徽理工大学，2019.

[20]张小红．基于 Android 的无线智能家居系统设计与研究[D]．四川：电子科技大学，2016.

[21]IDC: Smart Home Device Growth Stalls[J]. Wireless News,2022,12.

[22]自宁，王宁. 基于云平台的智能家居系统设计与实现[J]. 电子制作，2022，30(20)：47-50.

[23]Zhen Ying,Maragatham T.,Mahapatra Rajendra Prasad. Design and implementation of smart home energy management systems using green energy[J]. Arabian Journal of Geosciences,2021,14-17.

[24]邓楷煊，张金尧，许彩望，孙朝鹏．ZigBee 技术下的智能家居系统设计[J]．物联网技术，2022，12(09)：91-93+97.

[25]黄超. 智能家居系统安全方案的技术研究[J]. 数字通信世界，2022(07)：43-46+50.

[26]Vivint Smart Home Taps David Bywater as CEO[J]. Wireless News,2021,74-76.

[27]莫先．基于 STM32 单片机家电控制及家居环境监测系统设计与实现[D]．重庆：重庆理工大学，2016.

[28]Smart Citizens Introduces UAE Residents to Smart Home Automation Solutions Via Artificial Intelligence[J]. M2 Presswire,2020,33-36.

[29]肖松飞．基于云计算与微服务的智能家居系统[D]．西安：长安大学,2022.

[30]Vivint Smart Home Posts Second Quarter 2022 Results[J]. Wireless News,2022,52.

[31]周宝玲，黄军豪，柳贵东．基于单片机的智能家居系统设计[J]．信息与电脑(理论版)，2021，33(14)：145-147.

[32]李亚慧，刘娜，刘国权，温丹丽. 基于物联网技术的智能家居系统设计与实现[J]. 电脑编程技巧与维护，2021(04)：125-126+139.

[33]王美荣，孙苗苗. 基于 Android 的智能家居系统设计与研究[J]．电脑知识与技术，2020，16(35)：43-44.

[34]Smart Home Demand Launches Platform With Fixes To Common Problems of Smart

Home Devices[J]. M2 Presswire,2022,78-80.

[35]黄晓斌．一种基于 STM32 单片机的多功能智能家居控制系统[D]．西安：西安电子科技大学，2021．

[36]张启龙，陈湘萍．OneNET 云平台 WiFi 远程控制的智能家居系统[J]．现代电子技术，2020，43(14)：25-29．

[37]王雅莹，蔡学森．基于 Python 语言的智能家居系统的设计[J]．科技资讯，2020，18(07)：25+27．

[38]陈梅芬，李伟权．基于语音控制技术的简易智能家居系统设计[J]．电子世界，2019(20)：131-132．

[39]任瑞斌．智能家居系统中人工智能的应用[J]．智能建筑与智慧城市，2019(05)：89-90+94．

[40]王萌．基于 Android 的智能家居系统设计与实现[D]．四川：电子科技大学,2019．

[41]Hospitality Turns to Smart Home Technology,Driving New Fast Growth Opportunities and 20 Million Device Shipments by 2027; Hospitality and coworking offer new but distinct opportunities for smart home players[J]. M2 Presswire,2023,88.

[42]朱玉萌．基于 ZigBee 技术与 WIFI 的智能家居系统设计[D]．辽宁：沈阳建筑大学，2019．

[43]Morris Lindsey L.,Fairman Andrea D.,Ding Dan,Messina Kristin. Providing Mainstream Smart Home Technologies as Assistive Technology for Persons With Disabilities:Current Trends and Needs[J]. AMERICAN JOURNAL OF OCCUPATIONAL THERAPY,2021,75-78.

[44]李志瑞，段元梅．基于物联网的智能家居系统设计[J]．无线互联科技，2023，20(12)：30-32．

[45]盛承强．智能家居系统的研究与应用[J]．房地产世界，2022，No.380(24)：155-157.

[46]Moses Jeban Chandir,Adibi Sasan,Angelova Maia,Islam Sheikh Mohammed Shariful. Smart Home Technology Solutions for Cardiovascular Diseases: A Systematic Review[J]. Applied System Innovation,2022,45-47.

[47]邱明松，王望望．家用智能开关的设计与制作[J]．计算机产品与流通，2019(07)：147．

[48]Smart Home Meets Smart Mobility.[J]. M2 Presswire,2022,89.

[49] Li Wenda,Yigitcanlar Tan,Liu Aaron,Erol Isil. Mapping two decades of smart home research: A systematic scientometric analysis[J]. Technological Forecasting & Social Change,2022,179.

[50] Ingram Micro partners with Arenti Smart Home Security System[J].M2 Presswire,2021,11-15.

[51] DTS Play-Fi Ranks as CES 2022 Innovation Awards Honoree in Smart Home and Software & Mobile App Categories[J]. Wireless News,2021,142.

[52] Tang Ruiyang,Inoue Yuki. Services on Platform Ecosystems in the Smart Home 2.0 Era: Elements Influencing Consumers' Value Perception for Smart Home Products[J]. Sensors,2021,88.

附　录

```
#include "stm32f10x.h"
#include "./usart/bsp_usart1.h"
#include "./OLED_I2C/OLED_I2C.h"
#include "./delay/delay.h"
#include "adc.h"
#include "string.h"
#include "bsp_dht11.h"
#include "./TIMER/timer.h"
#define    FS       PAout(7)
#define    LED    PAout(0)
#define    DelayA    PBout(13)
#define    DelayB    PBout(14)
#define    DelayC    PBout(15)
#define     Key1   PBin(12)
TIM_TimeBaseInitTypeDef    TIM3_TimeBaseStructure;
TIM_OCInitTypeDef    TIM3_OCInitStructure;
void DelayStm32(__IO u32 nCount)
{
   for(; nCount != 0; nCount--);
}
u16 TX_1MS=0;
u16 Time_1MS=0;
u16 Time_1S=0;
u8 WenDu=0;
u8 ShiDu=0;
u8 GZhao =0;
u8 YanWu =0;
```

```
u8 AlarmT=40;
u8 AlarmH=60;
u8 AlarmG=50;
u8 AlarmM=50;
DHT11_Data_TypeDef DHT11_Data;
u16 Cnt=0;
char dispBuff[10];
u8 FlagUpDara = 0;
u8 FlagLED = 0;
u8 FlagFS = 0;
u8 FlagDelayA = 0;
u8 FlagDelayB = 0;
u8 FlagDelayC = 0;
u8 StateLED = 0;
u8 StateDelayA = 0;
u8 StateDelayB = 0;
u8 StateDelayC = 0;
u8 StateFS = 0;
u16 KeyCnt = 0;
u8 RxBuf1[128];
u8 RxBuf2[128];
char MqttDataDL[37]=
{
    0x10, 0x23, 0x00, 0x04, 0x4D, 0x51, 0x54, 0x54, 0x04, 0xC0,
    0x00, 0x78, 0x00, 0x0A, 0x31, 0x30, 0x34, 0x33, 0x35, 0x31,
    0x32, 0x35, 0x31, 0x35, 0x00, 0x06, 0x35, 0x37, 0x30, 0x38,
    0x39, 0x34, 0x00, 0x03, 0x6D, 0x63, 0x75

};
char MqttDataDY[10]=
```

```
{
        0x82, 0x08, 0x00, 0x02, 0x00, 0x03, 'a', 'p', 'p', 0x00
};
char MqttDataFB[24]=
{
        0x30, 0x16, 0x00, 0x03, 0x6D, 0x63, 0x75, 0x06, 0x06, 0x06,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00
};
void SendASC1(u8 d)
{
    USART_SendData(USART1, d);
    while (USART_GetFlagStatus(USART1, USART_FLAG_TXE) == RESET);
}
void SendASC2(u8 d)
{
    USART_SendData(USART2, d);
    while (USART_GetFlagStatus(USART2, USART_FLAG_TXE) == RESET);
}
void DengLu(void)
{
        u8 k=0;
        printf("AT+CIPSEND=37\r\n");
        DelayS(1);
        for(k=0; k<37; k++)
        {
                SendASC1(MqttDataDL[k]);
        }
}
void DingYue(void)
```

```
{
        u8 k=0;
        printf("AT+CIPSEND=10\r\n");
        DelayS(1);
        for(k=0; k<10; k++)
        {
                SendASC1(MqttDataDY[k]);
        }
}
void FaBu(void)
{
        u8 k=0;
        printf("AT+CIPSEND=24\r\n");
        DelayS(1);
        for(k=0; k<24; k++)
        {
                SendASC1(MqttDataFB[k]);
        }
}
void GPIO_Config(void)
{
  GPIO_InitTypeDef GPIO_InitStructure;
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC,ENABLE);
  GPIO_InitStructure.GPIO_Pin =    GPIO_Pin_13| GPIO_Pin_14| GPIO_Pin_15;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(GPIOC, &GPIO_InitStructure);
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA,ENABLE);
  GPIO_InitStructure.GPIO_Pin =    GPIO_Pin_7 | GPIO_Pin_4;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
```

```
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
}
void DispOled(void)
{
        sprintf(dispBuff,"WenDu = %0.2dC %0.2dC", WenDu,AlarmT);
        OLED_ShowStr(0,0,dispBuff,2);
        sprintf(dispBuff,"ShiDu = %0.2d%% %0.2d%%",ShiDu,AlarmH);
        OLED_ShowStr(0,2,dispBuff,2);
        sprintf(dispBuff,"GZhao = %0.2d%% %0.2d%%",GZhao,AlarmG);
        OLED_ShowStr(0,4,dispBuff,2);
        sprintf(dispBuff,"YanWu = %0.2d%% %0.2d%%",YanWu,AlarmM);
        OLED_ShowStr(0,6,dispBuff,2);
}
 int main(void)
{
    USART1_Config();
     GPIO_Config();
     LED =0;
     DelayA = 0;
     DelayB = 0;
     DelayC = 0;
     FS = 0;
     Adc_Init();
     DelayInit();
     I2C_Configuration();
     OLED_Init();
     OLED_Fill(0xFF);
         DelayS(1);
         DelayS(1);
```

```
DelayS(1);
printf("AT\r\n");
DelayS(1);
printf("AT\r\n");
DelayS(1);
printf("AT\r\n");
DelayS(1);
printf("AT+CWMODE=3\r\n");
DelayS(1);
printf("AT+CWJAP=\"win\",\"123123123\"\r\n");
DelayS(1);
DelayS(1);
DelayS(1);
DelayS(1);
DelayS(1);
DelayS(1);
DelayS(1);
DelayS(1);
DelayS(1);
DelayS(1);
DelayS(1);
DelayS(1);
DelayS(1);
DelayS(1);
DelayS(1);
DelayS(1);
DelayS(1);
printf("AT+CIPSTART=\"TCP\",\"183.230.40.39\",6002\r\n");
DelayS(1);
DelayS(1);
DelayS(1);
DelayS(1);
```

```
DelayS(1);
DelayS(1);
DengLu();
DelayS(1);
DelayS(1);
DelayS(1);
DingYue();
DelayStm32(0xFFFFF);
OLED_Fill(0x00);
DelayStm32(0xFFFFF);
TIM4_Int_Init(9,7199);
while(1)
{
        DHT11_Read_TempAndHumidity ( & DHT11_Data );
        WenDu = DHT11_Data.temp_int;
        ShiDu = DHT11_Data.humi_int;
        GZhao = 100-Get_Adc_Average(0,20)*100/4096;
        YanWu = Get_Adc_Average(1,20)*100/4096;
        MqttDataFB[10] = WenDu;
        MqttDataFB[11] = ShiDu;
        MqttDataFB[12] = GZhao;
        MqttDataFB[13] = YanWu;
        MqttDataFB[14] = StateLED ;
        MqttDataFB[15] = StateFS;
        MqttDataFB[16] = StateDelayA;
        MqttDataFB[17] = StateDelayB;
        MqttDataFB[18] = StateDelayC;
        DispOled();
        if( (WenDu>AlarmT)||(ShiDu>AlarmH)||(YanWu>AlarmM) )
        {
```

```
                StateFS = 1;
                FlagFS=0;
        }
        else
        {
                StateFS =    FlagFS;
        }
        if(GZhao<AlarmG)
        {
                StateLED = 1;
                FlagLED=0;
        }
        else
        {
                StateLED =    FlagLED;
        }
        StateDelayA =    FlagDelayA;
        StateDelayB =    FlagDelayB;
        StateDelayC =    FlagDelayC;
        if(StateLED==1) LED=1; else LED=0;
        if(StateFS==1) FS=1; else FS=0;
        if(StateDelayA==1) DelayA=1; else DelayA=0;
        if(StateDelayB==1) DelayB=1; else DelayB=0;
        if(StateDelayC==1) DelayC=1; else DelayC=0;
        if(FlagUpDara==1)
        {
                FlagUpDara = 0;
                FaBu();
        }
}
```

```
}
void TIM4_IRQHandler(void)
{
        if (TIM_GetITStatus(TIM4, TIM_IT_Update) != RESET)
        {
                TIM_ClearITPendingBit(TIM4, TIM_IT_Update );
                TX_1MS++;
                if(TX_1MS>=5000)
                {
                        TX_1MS = 0;
                        FlagUpDara = 1;
                }
        }
}
void USART1_IRQHandler(void)
{
    uint8_t ucTemp;
        if(USART_GetITStatus(USART1,USART_IT_RXNE)!=RESET)
        {
                ucTemp = USART_ReceiveData(USART1);
            RxBuf1[0] = RxBuf1[1];
            RxBuf1[1] = RxBuf1[2];
            RxBuf1[2] = RxBuf1[3];
            RxBuf1[3] = ucTemp;
            if( (RxBuf1[0]=='F')&&( RxBuf1[1]=='D') )
            {
                    if( (RxBuf1[2]=='0')&&( RxBuf1[3]=='0') )
                    {
                            FlagLED = 0;
                    }
```

```
else if( (RxBuf1[2]=='0')&&( RxBuf1[3]=='1') )
{
        FlagLED = 1;
}
else if( (RxBuf1[2]==   '1')&&( RxBuf1[3]=='0') )
{
         FlagDelayA= 0;
}
else if( (RxBuf1[2]== '1')&&( RxBuf1[3]=='1') )
{
        FlagDelayA = 1;
}
else if( (RxBuf1[2]==   '2')&&( RxBuf1[3]=='0') )
{
         FlagFS= 0;
}
else if( (RxBuf1[2]== '2')&&( RxBuf1[3]=='1') )
{
        FlagFS = 1;
}
else if( (RxBuf1[2]==   '3')&&( RxBuf1[3]=='0') )
{
         FlagDelayB= 0;
}
else if( (RxBuf1[2]== '3')&&( RxBuf1[3]=='1') )
{
        FlagDelayB= 1;
}
else if( (RxBuf1[2]==   '4')&&( RxBuf1[3]=='0') )
{
```

```
                    FlagDelayC= 0;
                }
                else if( (RxBuf1[2]== '4')&&( RxBuf1[3]=='1') )
                    FlagDelayC= 1;
                }
            }
                else if( (RxBuf1[0]=='F')&&(RxBuf1[1]=='t') )
                {
                    AlarmT = (RxBuf1[2]-0x30)*10 + (RxBuf1[3]-0x30);
                }
                else if( (RxBuf1[0]=='F')&&(RxBuf1[1]=='h') )
                {
                    AlarmH = (RxBuf1[2]-0x30)*10 + (RxBuf1[3]-0x30);
                }
                else if( (RxBuf1[0]=='F')&&(RxBuf1[1]=='g') )
                {
                    AlarmG = (RxBuf1[2]-0x30)*10 + (RxBuf1[3]-0x30);
                }
                else if( (RxBuf1[0]=='F')&&(RxBuf1[1]=='m') )
                {
                    AlarmM = (RxBuf1[2]-0x30)*10 + (RxBuf1[3]-0x30);
                }
        }
}
void USART2_IRQHandler(void)
{
    uint8_t ucTemp2;
    if(USART_GetITStatus(USART2,USART_IT_RXNE)!=RESET)
    {
        ucTemp2= USART_ReceiveData(USART2);
```

```
RxBuf2[0] = RxBuf2[1];
RxBuf2[1] = RxBuf2[2];
RxBuf2[2] = RxBuf2[3];
RxBuf2[3] = ucTemp2;
if( (RxBuf2[0]=='F')&&( RxBuf2[1]=='D') )
{
        if( (RxBuf2[2]=='0')&&( RxBuf2[3]=='0') )
        {
                FlagLED = 0;
        }
        else if( (RxBuf2[2]=='0')&&( RxBuf2[3]=='1') )
                FlagLED = 1;
        }
        else if( (RxBuf2[2]==    '1')&&( RxBuf2[3]=='0') )
                 FlagDelayA= 0;
        }
        else if( (RxBuf2[2]== '1')&&( RxBuf2[3]=='1') )
                FlagDelayA = 1;
        }
        else if( (RxBuf2[2]==    '2')&&( RxBuf2[3]=='0') )
        {
                 FlagFS= 0;
        }
        else if( (RxBuf2[2]== '2')&&( RxBuf2[3]=='1') )
        {
                FlagFS = 1;
        }
        else if( (RxBuf2[2]==    '3')&&( RxBuf2[3]=='0') )
                 FlagDelayB= 0;
        }
```

```
        else if( (RxBuf2[2]== '3')&&( RxBuf2[3]=='1') )
        {
                FlagDelayB= 1;
        }
        else if( (RxBuf2[2]==   '4')&&( RxBuf2[3]=='0') )
                FlagDelayC= 0;
        }
        else if( (RxBuf2[2]== '4')&&( RxBuf2[3]=='1') )
                FlagDelayC= 1;
        }
}
```