

单位代码： 10293 密 级： 公开

南京邮电大学

专业学位硕士学位论文



论文题目： 基于物联网的智慧医疗空气消毒监控系统的
设计与实现

学 号 1219053918

姓 名 王严晖

导 师 庞宗强

专业学位类别 工程硕士

类 型 全日制

专业（领域） 仪器仪表工程

论文提交日期 2022.06

Design and Implementation of Intelligent Medical Air Disinfection Monitoring System Based on Internet of Things

Thesis Submitted to Nanjing University of Posts and
Telecommunications for the Degree of
Master of Engineering



By

Wang Yanhui

Supervisor: Prof.Zongqiang Pang

June 2022

摘要

智慧医疗是通过将无线通信、传感器、互联网等多种技术综合应用于整个医疗管理体系中，建立起信息高度共享化的医疗监测、控制和管理系统。智慧医疗相关应用就是在现有医疗设备、医疗服务的基础上利用物联网技术进行信息化升级。医疗设备信息化不仅能够促进医疗产业的高速发展，对医疗服务的质量和效率也有比较积极意义。

本文在分析了现有智慧医疗子系统相关设计的基础上，针对传统循环风紫外线消毒机信息化程度较低的缺陷，以消毒机的功能需求为切入点，采用物联网层级架构的思想，从软件设计与硬件配置两个维度构建完整的空气消毒监控系统，包括消毒机设备终端和监控平台。消毒机设备终端采用 STM32F103VET6 作为微控制器，负责接收、封装传感器数据，解析监控平台下发指令并控制执行器。选用 NB-IoT 模块实现 TCP 客户端，与监控平台 TCP 服务器进行双向的数据通信。执行器包括紫外线灯和离心风机，由离心风机将室内空气带入消毒机内部，在紫外线灯的照射下达到空气消毒的目的。并结合消毒机的实际功能需求，在设备终端加入触摸屏和红外接收模块，作为终端补充的控制交互方式。监控平台使用 Node.js 开发，主要分为两个部分：第一部分是 TCP 服务器，用于接收设备终端通过移动通信网络传输的传感器数据，对数据进行处理后存储到 MongoDB 数据库。其中，设备终端与监控平台之间的数据传输采用 JSON 格式。第二部分为 Web 服务器，负责向应用层服务提供数据接口。

最后部署系统并进行了相关测试，测试结果表明该系统能够稳定运行，实现了设备终端与监控平台双向的实时数据通信。验证了医用空气消毒监控系统具有室内环境监测、远程控制工作、设备终端交互、监控平台多设备终端管理、数据存储、数据显示、用户管理等功能。

关键词:物联网，消毒，NB-IoT，Node.js，监控平台

Abstract

Intelligent medical is to establish a medical monitoring, control and management system with a high degree of information sharing by comprehensively applying various technologies such as wireless communication, sensors, and Internet to the entire medical management system. The application of intelligent medical care is to use Internet of Things for information upgrading on the basis of existing medical equipment and medical services. Medical equipment informatization can not only promote the rapid development of the medical industry, but also have a positive impact on the quality and efficiency of medical services.

Based on the analysis of the relevant designs of the existing intelligent medical subsystems, this paper proposes to take the functional requirements of the disinfection machine as the breakthrough point and adopt the idea of the hierarchical architecture of IoT to build a complete air disinfection monitoring system from the two dimensions of software design and hardware configuration, aiming at the defects of the low informatization of traditional circulating air ultraviolet disinfection machine. The air disinfection monitoring system includes disinfection machine and monitoring platform. STM32F103VET6 is used as the microcontroller in the disinfection machine, which is responsible for receiving and packaging sensor data, analyzing the instructions issued by the monitoring platform and controlling the actuator. NB-IoT module is selected to realize the TCP client, and conduct two-way data communication with TCP server of monitoring platform. The actuator includes an ultraviolet lamp and a centrifugal fan. The centrifugal fan brings indoor air into the interior of the disinfection machine, and achieves the purpose of air disinfection under the irradiation of the ultraviolet lamp. At the same time, combined with the actual functional requirements of the machine, a touch screen and an infrared receiving module are added to the equipment as a supplementary control interaction mode of it. The monitoring platform is developed using Node.js and is mainly divided into two parts: The first part is the TCP server, which is used to receive and process the sensor data transmitted by the machine through the mobile communication network, and then store it into the MongoDB database. Among them, the data transmission between the machine and the monitoring platform adopts JSON format. The second part is the Web server, which is responsible for providing data interfaces to the application layer services.

Finally, the system is deployed and tested. The test results show that the system can run stably and realize two-way real-time data communication between the equipment terminal and the

monitoring platform. It is verified that the medical air disinfection monitoring system has the functions of indoor environment monitoring, remote control, equipment terminal interaction, monitoring platform multi-equipment terminal management, data storage, data display, user management and so on.

Key words: IoT, Disinfection, NB-IoT, Node.js, Monitoring Platform

目录

第一章 绪论	1
1.1 研究背景及意义	1
1.2 物联网在智慧医疗应用中的研究现状	2
1.3 主要研究内容	4
第二章 系统总体设计及关键技术分析	6
2.1 系统需求及功能指标	6
2.1.1 系统需求	6
2.1.2 功能指标	6
2.2 物联网通信的实现	7
2.2.1 无线通信技术介绍	7
2.2.2 无线通信方式对比分析	8
2.2.3 NB-IoT 关键技术介绍	9
2.3 监控平台功能实现	11
2.3.1 Node.js	12
2.3.2 MongoDB 数据库	14
2.4 总体设计方案	15
2.5 本章小结	16
第三章 空气消毒监控系统硬件设计	17
3.1 设备终端硬件框架	17
3.2 微控制器及硬件模块接口连接方式	17
3.3 传感器模块	19
3.4 NB-IoT 通信模块	20
3.5 设备终端交互模块	20
3.5.1 触摸屏	21
3.5.2 红外遥控	21
3.6 执行器模块	22
3.6.1 风机	22
3.6.2 紫外线灯	22
3.7 电源电路及 PCB 设计	23
3.8 本章小结	24
第四章 空气消毒监控系统软件设计	25
4.1 开发环境	25
4.2 消毒机设备终端软件设计	26
4.2.1 设备终端软件初始化	26
4.2.2 传感器数据采集	29
4.2.3 设备终端交互模块软件设计	30
4.2.4 通信模块软件开发	31
4.3 数据交换格式	34
4.4 监控平台软件设计	35
4.4.1 数据库设计	36
4.4.2 通信服务器设计	37
4.4.3 Web 服务器设计	40
4.5 本章小结	43
第五章 系统测试	44

5.1 基于监控平台的功能验证 44

 5.1.1 系统部署 44

 5.1.2 监控平台功能验证 46

5.2 控制功能验证 49

5.3 系统稳定性与延时测试 50

5.4 本章小结 51

第六章 总结与展望 52

 6.1 总结 52

 6.2 展望 53

参考文献 54

附录 1 攻读硕士学位期间撰写的论文 56

附录 2 攻读硕士学位期间申请的专利 57

致谢 58

第一章 绪论

1.1 研究背景及意义

随着互联网技术的广泛应用和无线通信技术的日益成熟，物联网正在成为下一代互联网的演化方向。物联网的概念是使用传感器、RFID（射频识别）等各种设备进行信息采集，再通过各种通信方式接入网络，并依照事先约定好的协议进行信息通信，实现智能化识别、监测、控制和管理网络，能够实现人与人、人与物、物与物之间的信息交换^[1]。

物联网已经应用在包括工业、农业、医疗等多个领域。在医疗领域中，物联网技术的应用前景巨大。2014年的物联网高峰论坛上首次提出“健康物联网”的概念，标志着政府支持在医疗行业中广泛应用物联网技术，表明了未来医疗领域将会与先进的物联网技术相结合，最终实现医疗领域的智慧化。智慧医疗是通过将无线通信、传感器、互联网等多种技术综合应用于整个医疗管理体系中，建立起信息高度共享化的医疗监测、控制和管理系统。简单的说，智慧医疗应用就是在现有医疗设备、医疗服务的基础上利用物联网技术进行信息化升级。智慧医疗不仅能够促进我国医疗设备信息化产业的高速发展，对医疗服务的质量和效率也有比较积极意义。与此同时，还可以提高医院的经济效益和好评度，对医患关系的改善也起到一定的帮助作用^[2]。

当前医疗领域信息化的发展模式已经在逐步推进，智慧医疗所倡导的整个大的系统是由各个小的智慧医疗子系统构成^[3]，如智慧病房、药品管理、远程监护、医疗设备的跟踪与管理、病房智能陪护设施、静脉输液监测、医院消毒管理、医疗废弃物管理等。本文以医疗场所空气消毒监控系统为例进行具体研究。

空气是病毒进行传播的一种主要载体，改善医疗机构的空气质量可有效避免医院内人员交叉感染^[4]。医疗机构采用的空气消毒方式包括照射紫外线消毒、化学消毒、使用专门的消毒机器等。照射紫外线消毒使用移动式或者悬吊式紫外线灯在室内直接照射进行消毒工作。化学消毒又分为超低容量喷雾法、熏蒸法两种，其中超低容量喷雾法是将消毒液雾化成20 μm 以下的微小粒子，均匀地喷洒在空气中，使消毒液与空气中微生物颗粒充分接触，达到杀灭病毒或微生物的目的。熏蒸法的原理是利用化学消毒剂所具有的挥发性，在一定空间内通过加热或其他方法使消毒剂挥发，接触到空气中微生物后使之灭活。照射紫外线消毒和化学消毒会危害到人体安全，因此仅适用于无人状态下室内空气的消毒。目前使用比较广泛的消毒机器有静电吸附式空气消毒机和循环风紫外线消毒机。静电吸附式空气消毒机是利用机器风

道中过滤材料和静电吸附的原理,来消除空气中的尘埃和微生物。循环风紫外线消毒机由风机、过滤系统和紫外线灯等组成,可有效杀灭空气中的微生物,并有效地过滤空气中的尘埃粒子^[5]。静电吸附式空气消毒机和循环风紫外线消毒机都适用于有人状态下的室内消毒,因为消毒的机理的不同,静电吸附式空气消毒机消毒能力一般弱于循环风紫外线消毒机。近年来,循环风紫外线消毒机在医院得到了逐步地推广与应用,其具有灭活率高、灭活速度快的优势,受到了广大医务人员和患者的认同^[6]。

传统的循环风紫外线消毒机需要由工作人员开启或关闭,并使用专用的仪器测量室内空气质量并记录在册,需耗费工作人员大量的时间和精力,室内空气质量数据的记录也难以长期保存和管理。本文在《常规医疗器械的智能化设计与开发》的横向科研项目的基础上,设计开发了一种符合智慧医疗理念的空气消毒监控系统。该系统可以和上文中提到的智慧医疗其他的子系统实现关联组合,以模块化的结构嵌入整个智慧医疗大系统当中。系统利用物联网、传感器等技术,在实现循环风紫外线消毒机的功能基础上进行重新设计,结合监控平台,开发完整的消毒监控系统,实现远程控制、室内空气质量数据的实时监测、数据显示等功能。新技术的应用往往带来工作方式的改变和工作效率的极大提高,系统应用后,工作人员可根据监控平台的监测信息配合消毒工作的开展,科学安排工作优先级。综上,本课题的研究符合当代智慧医疗系统的设计理念,并且具有很高的应用价值、巨大的市场潜力和良好的发展前景。

1.2 物联网在智慧医疗应用中的研究现状

随着计算机、物联网、传感器等技术的发展,智慧医疗下各种子系统被设计实现,包括药品管理、远程医疗、输液监测系统、消毒监控系统等等。

在药品管理方面主要使用的是物联网中射频识别技术。射频识别系统一般由3部分组成,标签(Tag)、阅读器(Reader)和数据管理系统^[7]。Shieh等人使用射频识别技术、软件技术和数据库技术相结合,开发出了一个图形化的药瓶管理系统,护士在拿取药品时需要核对患者的RFID标签和药品编号,避免错误用药的情况^[8]。T. Suzuki等将射频识别技术应用于监测老年慢性病患者服药,开发出iMEC系统。通过移动分布式RFID阅读器、摄像头和传感器,测量标签对象的移动(包括药物、水、牛奶等),每次患者服用药物时,系统会确认药物的类型、数量、时间,并判断药物服用是否规范。如果不符合规范,系统将通过服务器发送邮件通知护理人员不正确用药的警告,护理人员可以通过访问服务器随时监控服药的规范性^[9]。

基于物联网技术的远程医疗也是智慧医疗领域一个重要的应用。SW-SHMS系统由

Al-khafajiy 等人研究设计,是专用于老年人的智能医疗检测系统。系统使用可穿戴式的传感器采集用户的生理信息数据(脉搏),利用 Arduino Uno 作为微控制器,智能手机作为功能上的“网关”,将数据传输到云端进行数据分析和处理,一旦发现患者数据异常,就会通过医院平台上报给医生,让病患得到及时的治疗^[10]。沈阳理工大学的邓燕楠在远程医疗领域应用 NB-IoT(窄带物联网)技术,设计了一种智慧医疗系统。该系统实现将患者的血糖值信号通过 NB-IoT 模块实时、自动地远距离上传,使得糖尿病患者采集血糖值信号不受地点约束,并且系统智慧医疗软件平台能够对患者历史医疗数据进行信息化管理,有利于医生对患者病情做出准确判断^[11]。

智慧医疗输液监测子系统相关研究也有很多。Zhang H 在其研究中概述了智慧医院中医疗物联网设备的应用场景和特性,然后提出了一种使用 NB-IoT 的架构,包括终端感知层、基站、云服务器、边缘计算服务器。作为案例研究,设计了一个输液监控系统,使用 NB-IoT 监测静脉输液期间的实时滴液速率和剩余药物量^[12]。魏文博设计了一种基于线性 CCD(Charge Coupled Device,电荷耦合器件)的方法的输液监测系统。监测设备利用线性 CCD 传感器、温湿度传感器完成病房输液状态等信息的检测,通过 LoRa 模块和网关实现监测终端输液状态信息的上传,开发的监测管理平台用于实现输液数据的处理、显示和存储,实现了远程的输液数据存储、终端远程管理以及远程报警等功能^[13]。

物联网技术应用在医用消毒领域的研究包括空气消毒设备的设计和消毒过程的监控等。苗荣霞等提出了一种基于红外遥控和低功耗单片机的动态紫外线空气消毒机控制系统。在其设计的空气消毒控制系统中针对传统空气消毒方法存在的不足进行了相关改进,该系统具有消毒无死角,低耗电、适用范围广等特点^[14]。Hung 在其研究中实现医疗器械消毒过程监控,通过在消毒仪器中放置传感器监测灭菌温度和工作时间,结合物联网通信技术将数据传至平台管理系统进行显示和存储^[15]。牟强善提出采用消毒机器、Wi-Fi 通信及软件应用相结合的解决方案来实现循环风空气消毒器产品功能的升级和空气参数的监测^[16]。Yang L 等研究并设计出一款基于物联网的医用空气消毒机,使用紫外线灯管照射加上物理过滤装置达到净化消毒医院空气的目的。在功能上加入医疗环境监测,监测量包括循环空气量、臭氧浓度、温湿度、泄露紫外线强度,并且将监测量传到用户端实时显示。当有监测量超过所设阈值时,会触发报警,关闭消毒机^[17]。Gerson 等人设计一种监测臭氧消毒过程中气体浓度、温度和湿度的无线传感器网络,保证了消毒操作的安全性^[18]。Zhao Y L 等人开发了一种适用于手术室或病房消毒的智能消毒机器人系统。该系统集成了半自动遥控模块、物联网通信、自动智能消毒功能,使用二氧化氯杀菌技术,可以杀灭空气和物体表面的细菌和病毒^[19]。

1.3 主要研究内容

针对传统循环风紫外线消毒机信息化程度较低,本课题在广泛参阅了相关参考文献的基础上对以下内容展开研究,主要可以概括为以下几个方面的内容:

- (1) 设计一种具有普适性的基于物联网的智慧医疗监控系统架构。
- (2) 研究微控制器、传感器、通信模块及通信协议等,设计基于物联网的智慧医疗空气消毒机设备终端。
- (3) 分析对比软件技术、数据交换格式、数据库技术等,选择在特定场景下最优的应用技术,设计监控平台,实现数据显示、持久化存储、远程控制等功能。

本文以目前医疗机构空气消毒应用需求为出发点,评估了现有相关设计的不足,设计并实现了一种符合智慧医疗特点的空气消毒监控系统。本系统包括:

(1) 消毒机设备终端,利用微控制器,结合传感器及外围电路实现室内空气质量数据的采集。通过通信模块实现与监控平台的双向数据通信,设备终端采集传感器数据,通过向上数据通道传至监控平台;监控平台通过向下通道推送控制指令到设备端,控制执行器工作。执行器包括消毒用紫外线灯和将空气带入过滤系统的风机。并结合消毒机的实际功能需求,在消毒机设备终端加入触摸屏和红外接收模块,作为终端补充的控制交互方式。

(2) 消毒监控平台,主要分为两个部分:第一部分是通信服务器,用于接收设备终端通过移动通信网络传输的传感器数据,对数据进行处理后存储到数据库中。第二部分为 Web 服务器,负责向应用层服务提供数据接口,实现网页端数据显示和用户指令下发的功能。

根据以上的研究内容及系统的设计实现,论文的组织结构如下:第一章为课题绪论,就本文的研究背景及意义进行阐述,并对物联网在智慧医疗子系统研究应用现状进行分析和讨论。

第二章是系统的总体设计及关键技术分析,首先对消毒监控系统的需求和功能指标进行阐述。然后对实现物联网通信的无线通信技术进行对比和优选。对实现监控平台所使用的相关技术进行分析和概述。最后给出空气消毒监控系统总体设计方案。

第三章为空气消毒监控系统硬件设计,在该章节中对消毒机设备终端各个硬件模块的功能、相关参数、接口连接方式和电路设计进行详细介绍,包括微控制器、传感器模块、通信模块、终端交互模块、电路及 PCB 设计等。

第四章是空气消毒监控系统的软件设计部分,包括开发环境介绍、消毒机设备终端嵌入式软件设计、数据交换格式、监控平台软件设计。消毒机设备终端软件设计主要实现传感器数据解析、封装和上传,终端交互以及远程控制指令解析,驱动执行等功能。监控平台软件

设计实现数据通信、数据存储及数据显示。

第五章为系统测试，系统测试包括基于监控系统的功能验证和系统稳定性测试及延时测试。功能验证又分为监控平台相关功能的验证、远程控制功能验证。

第六章是总结与展望，总结本文完成的工作，并针对系统现有设计的局限性，展望以后研究开展的方向。

第二章 系统总体设计及关键技术分析

2.1 系统需求及功能指标

2.1.1 系统需求

传统的医院空气消毒依赖人工操作，需要由工作人员将消毒设备带入相关场所进行消毒工作。消毒方式更多的是使用紫外线灯照射或熏蒸法等化学方法。使用化学方法和紫外线灯直接照射方法进行空气消毒的方式，存在一定的安全隐患，并且会对人体造成伤害。随着一些消毒仪器，类似于循环风紫外线消毒机、静电吸附式消毒机的出现，可以做到进行消毒过程人机共存，减少在消毒过程中对人体的伤害。但这些消毒机器信息化程度较低，工作人员在消毒结束后还需要会对消毒持续时间、空气质量等参数进行测量记录，室内空气的质量数据的记录也难以长期保存和管理。

参考上文中提到的智慧医疗子系统的相关设计，其中多数子系统只完成了监测的功能，如远程医疗中，患者体征参数的监测上传，输液监测系统中滴液速率和剩余药物量的上传。实现的只有上传数据通路，系统整体结构还可以再扩展。在医用消毒领域的研究也只是完成了单一的消毒设备的开发或者消毒环境的监测。本文在广泛参阅了现有智慧医疗子系统相关应用的基础上，旨在设计和实现一种应用于医院消毒相关的监控系统，实现感知层设备与应用层平台的双向数据通信。系统包括消毒机设备终端和监控平台。消毒机设备终端在实现消毒功能的基础上，增加室内环境数据信息采集上传、远程控制消毒机工作的功能；配合的监控平台实现传感器数据接收、存储、显示和控制指令的下发。在系统的设计和实现过程中，使其具有普适性，可以为其他智慧医疗子系统的设计提供思路。

2.1.2 功能指标

本系统所要完成的具体功能指标如下：

（1）准确、实时地室内环境监测：通过选取合适的传感器，实现在误差可接受范围内室内环境参数的信息采集；利用可靠的通信方式实现数据传输；在监控平台显示室内环境参数。

（2）远程控制消毒机设备终端工作，实现消毒功能：控制离心风机和紫外线灯工作，在风机离心力的作用下，将室内空气循环带入消毒通道，让空气中的病毒或细菌在紫外线灯的照射

下解体消亡。

(3) 消毒机设备终端交互: 在消毒机设备终端增加除监控平台下发指令控制以外的控制方式, 实现完整的循环风紫外线消毒机的功能。

(4) 监控平台多设备终端管理、数据管理、用户管理: 满足多个消毒机设备终端的接入和控制; 传感器数据存储; 监控平台的用户权限管理。

(5) 系统稳定性、数据实时性、低成本: 在保证上述功能实现的前提下, 具体设计时考虑系统的稳定性、数据传输的实时性以及整个系统的成本。

2.2 物联网通信的实现

基于物联网的智慧医疗消毒监控系统分为消毒机设备终端和监控平台两个部分, 使用无线通信技术实现消毒机设备终端与监控平台双向的数据通信。

2.2.1 无线通信技术介绍

无线通信是物联网的核心技术, 当前在物联网领域中应用较为广泛的无线通信技术包括 RFID、Bluetooth、LoRa、GPRS、Wi-Fi、ZigBee、Sigfox、NB-IoT 等。这些无线通信技术从通信速率上划分, 可以大致分为高速率、中速率与低速率三种类型, 三种类型都有各自应用的场景^[20]。高速率是指传输速率大于 1Mbps, 特点是流量高, 并且对功耗不敏感, 一般应用于视频监控、远程医疗等场景, 主要使用的通信技术为 4G、Wi-Fi 等。中速率是指传输速率大于 100kbps, 小于 1Mbps, 传输数据以语音、图片为特征, 如内置语音功能的智能家居设备等, 使用的无线通信方式有 GPRS 等。低速率是指传输速率小于 100kbps, 传输的数据一般为文本信息, 流量不高, 但对功耗敏感, 如智能仪表、环境监测、智能门锁等, 使用的无线通信方式有 LoRa、NB-IoT 等。根据本系统要实现远距离网络通信的功能需求, Bluetooth、RFID 等近距离通信方式无法满足, 在下文中将着重介绍可实现远距离网络通信的技术。

(1) Wi-Fi

Wi-Fi 是基于 802.11 协议的无线局域网接入技术^[21], 属于近距离通信方式的一种。在公共场所大都布设了无线访问接入点 AP, Wi-Fi 设备可借助 AP 接入网络。Wi-Fi 比较突出的特点在于其有较广的覆盖范围, 相比较于蓝牙, Wi-Fi 的覆盖范围更广, 覆盖半径在 100 米左右。Wi-Fi 属于高速率的无线通信技术, 传输速度可以达到 54Mbps (802.11.a) 或 11Mbps (802.11b), 因此适用于高速数据传输的应用, 如上文提到的视频监控等使用场景。

(2) LoRa

LoRa 是 LPWAN (Low Power Wide-Area Network, 低功率广域网络) 通信技术的一种^[22]。LoRa 主要在全球免费频段运行, 包括 433MHz、868MHz、915MHz 等频段。LoRa 技术的特点: 1) 传输距离远, 城镇覆盖距离为 2-5km, 传输速率一般在几十 kbps, 传输速率与传输距离成反比, 低速率下传输距离较远; 2) 工作功耗低、组网节点多, 理论上来说单个的 Lora 网关能够连接数万个 Lora 节点。

(3) ZigBee

ZigBee 采用 IEEE 802.15.4 标准规范, 是一种短距离传输的无线网络协议^[23]。主要特点有低速、低功耗、低成本、低复杂度等。在 ZigBee 网络中, 设备的角色分为协调器 (Coordinator)、路由器 (Router)、终端设备 (End Device) 三种。其中协调器负责启动整个网络, 路由器负责将其他设备接入网络。ZigBee 网络支持大量网络节点, 并且有多种网络拓扑结构, 包括星型、树型、网型等。

(4) GPRS

GPRS 的全称是 General Packet Radio Service (通用分组无线电服务), 2.5 代移动通信技术, 用于终端和通信基站之间的远程通信^[24]。GPRS 通信技术的特点在于: 1) 传输速率较快, 可达到 56kbps-114kbps; 2) 传输距离远, 但是受基站覆盖范围的限制, 在通信条件差的地方信号比较弱。3) 成本低, 由于移动通信终端的普及, 在物联网中采用 GPRS 通信模块的成本已经大大降低, 硬件成本相较于 Wi-Fi 或者 ZigBee 都有着较大的价格优势。

(5) NB-IoT

NB-IoT 全称为 Narrow Band-Internet of Things, 窄带物联网, 也是低功率广域网络通信技术的一种, 可直接部署于 GSM 网络或 LTE (Long Term Evolution, 长期演进) 网络。NB-IoT 的传输距离可达十几公里, 每个网络单元可以支持接入 50000 个设备终端^[25]。NB-IoT 传输速率一般在 160kbps-250kbps 之间。NB-IoT 优点有: 1) 连接多, 每个单元可以支持 50000 个设备终端连接管理; 2) 信号好, 相比于 LTE, NB-IoT 有 20dB 增益的提升, 相当于发射功率增大了 100 倍, 即覆盖能力增强了 100 倍; 3) 不需要网关组网, 直接接入移动通信网络; 4) 成本低。

2.2.2 无线通信方式对比分析

上文中介绍的 5 种通信方式, 可以实现远距离网络通信, 具体参数对比如表 2.1 所示。Wi-Fi、Lora、ZigBee 设备需要通过网关 (或 AP) 接入网络, 使用 NB-IoT 和 GPRS 则不需要网关组网。例如在物联网应用中使用 Wi-Fi 作为无线通信方式, 系统架构如图 2.1 所示, 终端通过 Wi-Fi 模块接入网关进而连接到互联网, 与应用服务器实现数据通信。使用 NB-IoT 作为

无线通信方式，直接通过电信运营商的通信基站连接到互联网。考虑到在实际应用场景下，消毒机设备可能会在医院多个区域使用，使用网关接入网络更适用于在固定场所使用的设备。所以考虑使用直接能接入运营商网络的通信方式。NB-IoT 与 GPRS 的传输速率都可满足本应用需求，NB-IoT 模块价格与 GPRS 模块价格接近，但是 NB-IoT 在信号覆盖、时延、功耗方面更加优于 GPRS，所以选择 NB-IoT 作为无线通信方式。

表 2.1 物联网通信方式对比

通信方式	Wi-Fi	LoRa	ZigBee	GPRS	NB-IoT
覆盖范围	100m	2-5km	150m	/	1.7-20km
传输速率	11Mbps 或 54Mbps	0.3kbps- 37.5kbps	250kbps	56kbps-114 kbps	160kbps- 250kbps
工作频段	2.4GHz、 5.0GHz	433MHz、 868MHz、 915MHz	2.4GHz- 2.4835GHz	890MHz- 960MHz	900MHz
是否借助网关接入网络	是	是	是	否	否

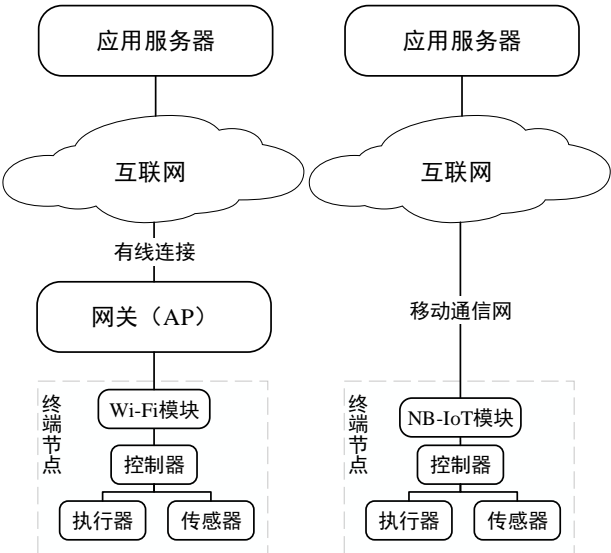


图 2.1 设备接入网络的两种方式

2.2.3 NB-IoT 关键技术介绍

NB-IoT 是在超窄带、低成本的理念下设计的全新技术，牺牲了一定程度的速率和时延性能，但是补充了更加适用于物联网场景的特性，包括覆盖能力增强、低功耗、大连接。

NB-IoT 在许可频段下可共存于 LTE 或 GSM，系统带宽为 200kHz，在 GSM 和 LTE 传输中对应一个资源块。如图 2.2 所示，在频段选择的情况下支持三种工作模式^[26]。1）独立部署模式（Stand-alone）：部署在一个或多个现有的或改造的 GSM 频段中。在独立部署方式下，NB-IoT 的配置限制较少，传输功率较高；2）LTE 带内部署模式（In-band）：在 LTE 带宽内

部署 NB-IoT，带内部署使用与 LTE 相同的 PRB（Physical Resource Block，物理资源包）。这种部署方式有一定的限制，LTE 业务会占用一些资源，所以传输功率由 LTE 和 NB-IoT 共享。

3) LTE 保护带部署模式（Guard band）：在 LTE 频谱边缘的保护带内部署 NB-IoT，不占用 LTE 资源。但是由于是部署在 LTE 的保护带内，所以需要考虑与 LTE 系统的兼容问题^[27]。

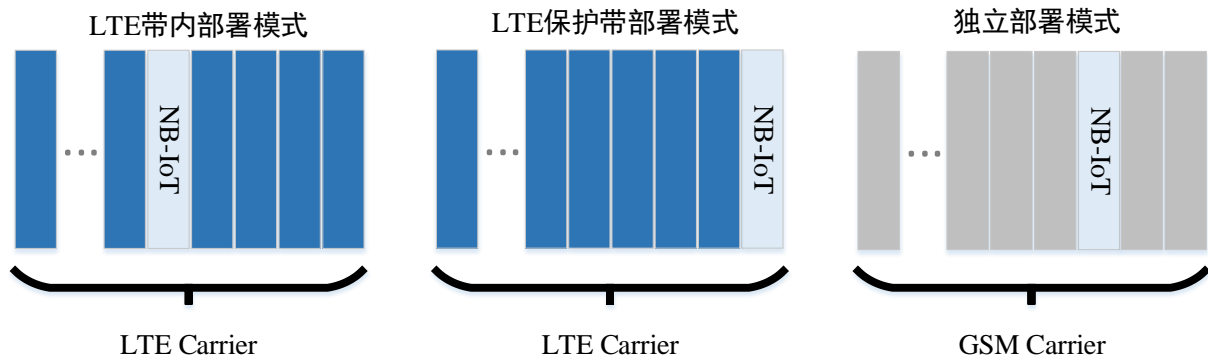


图 2.2 NB-IoT 支持的三种部署模式

在物联网相关的应用场景下，对于覆盖能力的要求比传统通信更高。NB-IoT 的覆盖增强，在上行方向和下行方向使用的策略不同。覆盖增强在上行方向上是通过提高上行功率谱密度和重复发送实现的。独立部署、LTE 带内部署和 LTE 保护带部署三种工作模式的上行覆盖性能相差不大。在下行方向使用的策略是重复发送。3GPP 协议定义 NB-IoT 不同信道的重复次数不同，通过重复传输实现覆盖增强。在独立部署模式下，基站的发射功率独立配置。LTE 带内部署和 LTE 保护带部署，因为 NB-IoT 与 LTE 系统共享基站发射功率，所以需要更多的重复发送次数才能达到与独立部署方式相近的覆盖能力。NB-IoT 独立部署模式下，下行发射功率可独立配置。如表 2.2，在典型的功率配置下，NB-IoT 下行功率谱密度比 GSM 高 0.45dB，比 LTE 功率谱密度高 14dB 左右。NB-IoT 在 LTE 带内部署和 LTE 保护带部署时，因为与 LTE 共享基站发射功率，功率谱密度下降。此时，NB-IoT 下行功率谱密度比 GSM 低约 7.5dB，比 LTE 功率谱密度高 6dB 左右^[28]。

表 2.2 GSM、LTE 与 NB-IoT 上行功率谱密度比较

上行方向	GSM	LTE	NB-IoT	
			Stand-alone	In-band、Guard band
上行最大发射功率/dBm	33	23	23	
终端最小调度带宽/kHz	200	180	15	3.75
上行最大功率谱密度 / (dBm/kHz)	10	0.45	11.2	17.3

NB-IoT 实现海量连接的技术手段主要有两种，一种是降低控制信令开销，另一种就是其最大的特性窄带传输^[28]。物联网业务中数据包较小，如果采用 LTE 的传输流程，每次数据传输都需要 10 条信令，资源开销大。比起 LTE，NB-IoT 在传输流程上做了简化，传输流程分为控制面数据传输方案与用户面数据传输方案。控制面数据传输方案传输流程如图 2.3（a）

所示, 相比于 LTE 信令流程, 可节省 5 条信令。在长间隔小数据包的传输场景下, 效率更高。用户面传输方案的传输方案如图 2.3 (b), 与 LTE 信令传输流程相比, 节省 4 条信令, 适用于传输大数据包和数据频繁发送的业务场景。

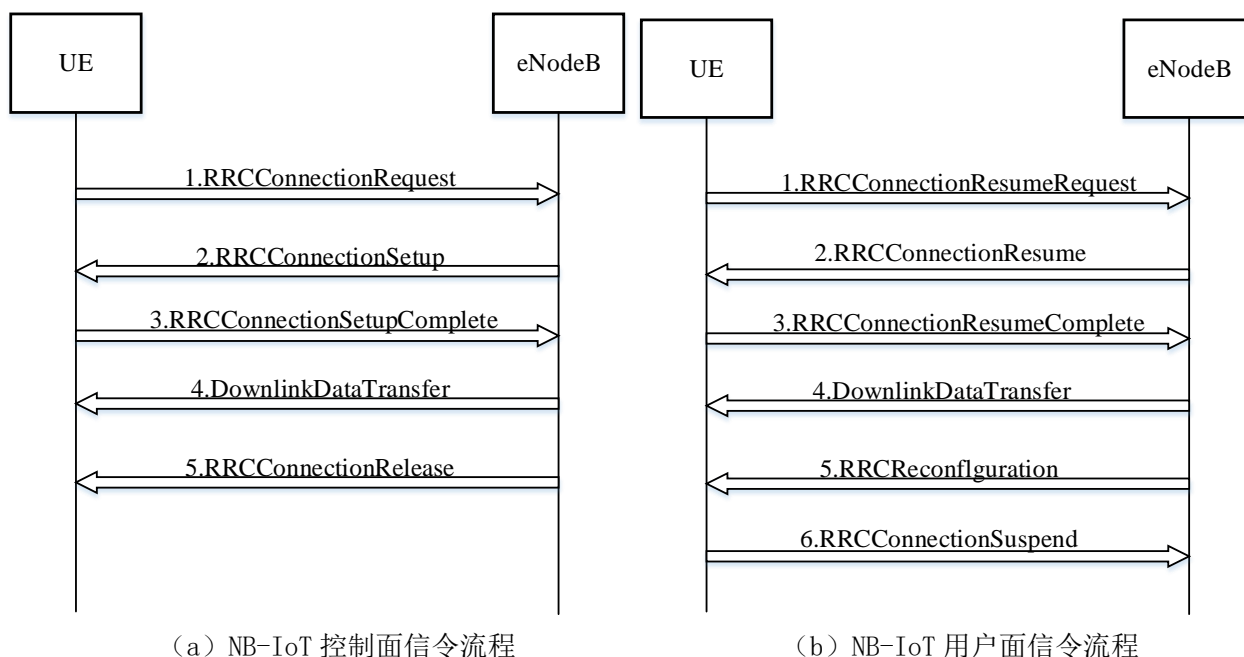


图 2.3 NB-IoT 信令流程

NB-IoT 的窄带传输模式适合小数据包、时延不敏感的物联网场景下使用。传统的 LTE 系统带宽值大多为 5MHz、10MHz、15MHz、20MHz 等, NB-IoT 的系统带宽为 200kHz, 占用的频带资源显著降低。在相同的总资源下, NB-IoT 资源利用率更高, 基站支持的连接数更多。

NB-IoT 为了降低功耗, 增加了节电模式 (Power Saving Mode, PSM) 和扩展的非连续接收 (Extended Discontinuous Reception, eDRX) 两大功能^[28]。节电模式允许终端在数据传输完成后进入深度睡眠状态, 此时仅有时钟等少量电路工作, 耗电量在 μA 级别。节电模式与关机不同的是, 终端处于节点模式下时在网络中还是已注册状态。eDRX 通过延长终端的唤醒周期降低终端在连接或者空闲状态下的功耗, 在每个寻呼周期中, 设置一段寻呼窗口, 终端在寻呼窗口内按照寻呼周期监听下行的寻呼消息。eDRX 与节电模式的差异在于, 不管是否有数据上报, eDRX 都会周期性监听寻呼, 但不使用信令。eDRX 状态下的功耗大于节电模式状态下的, 这两种状态可以配合使用, 达到最低的电源消耗。

2.3 监控平台功能实现

监控平台所要实现的功能包括传感器数据接收、数据存储、数据显示、用户管理、远程控制指令下发等。实现上述功能, 监控平台需要包括通信服务器、数据库、web 服务器。其

中通信服务器用于与消毒机设备终端实现双向数据通信，数据库用于数据存储，web 服务器用于向应用层服务提供数据接口。

监控平台需要使用服务器端编程语言开发，可以使用的语言有 Java、Python、PHP、Node.js 等。选择 Node.js 相较于其他编程语言来说，有如下优点：1) Node.js 是服务器端 JavaScript 运行环境，所以服务器端代码和客户端代码可以使用相同的语言开发，并且 JavaScript 作为一种解释性语言，语法难度小，更易掌握。2) Node.js 开发速度更快，开发生态繁荣，使用其提供的模块，仅仅几行代码就可以构建服务器。3) 在物联网应用场景下，Node.js 事件驱动机制可以用来实现自动触发，如传感器数值达到阈值，自动触发控制执行器工作，还可以用来实现 M2M 通信。

数据库按存储方式可分为关系型数据库和非关系型数据库。关系型数据库使用关系模型来组织数据，在实际的关系数据库中的关系也称表。一个关系数据库就是由若干个表组成。非关系型数据库以键值对存储，且结构不固定。物联网所生成的数据具有连续性、海量、非结构化等特点，如 RFID 标签数据、传感器数据等，会在监测过程中不断地实时生成。现有的关系型数据库技术由于处理速度和存储扩展的局限性，不适用于处理物联网相关数据^[29]。非关系型数据库通过牺牲关系数据库模型的一些属性（ACID 事务属性），可以实现更快的处理速度和可伸缩性。MongoDB 是非关系型数据库中功能最丰富的，监控平台数据存储使用 MongoDB 数据库。对于物联网应用来说，选用 MongoDB 数据库，有以下的一些优点，首先是灵活性，MongoDB 数据库采用非结构化的数据模型，可以存储和处理任何结构的数据^[30]。MongoDB 数据库支持分布式存储架构，有着优秀的水平扩展性。MongoDB 数据库包含丰富的查询和索引支持，可以针对物联网中大量数据在本地运行报告分析。MongoDB 的存储结构类型类似于 JSON，Node.js 也是基于 JavaScript 的环境，所以 Node.js 搭配 MongoDB 数据库使用还能大大减少因数据转换所带来的时间空间开销。

2.3.1 Node.js

Node.js，也称 Node，基于 Google Chrome V8 引擎，是一个开源、跨平台的服务器端 JavaScript 运行环境^[31]。Node.js 底层主要由 C 和 C++ 编写，所以具有极高的运行效率。在 Node.js 出现之前，JavaScript 主要应用在客户端程序设计当中，所写出的程序在浏览器上执行。Node.js 为 JavaScript 程序语言提供了操作文件、创建 HTTP 服务、创建通信服务等功能接口，使得 JavaScript 也能作为服务端程序设计语言使用^[32]。Node.js 应用程序的基础组成部分是模块。Node.js 提供了 exports 和 require 两个对象，exports 用于将模块公开，如 module.exports = em,

就是将模块 `em` 公开供其他程序文件使用。`require` 用于从外部获取一个模块，如 `var em = require("./eventemitter")`，即获取 `em` 模块的 `exports` 对象。使用 `require` 获取模块时，查找步骤如图 2.4 所示。

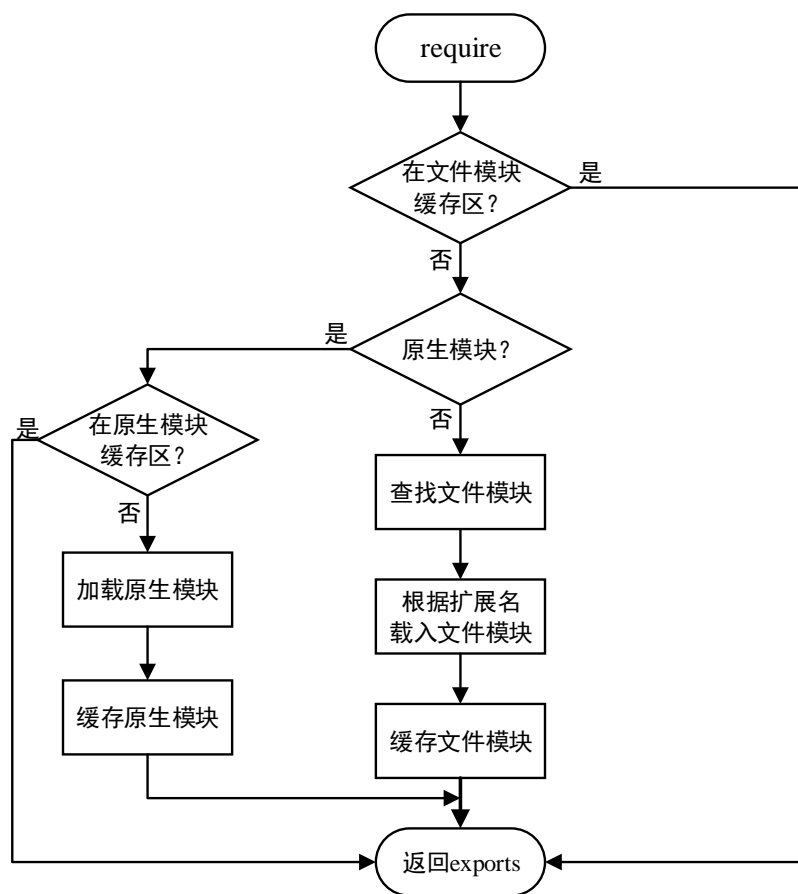


图 2.4 Node.js 模块加载路径

与大多数后端语言运行需求不同，Node 进程不依赖于多线程支持业务的并发执行，而是基于异步 I/O 事件模型。Chrome V8 引擎提供了异步执行回调接口，虽然 Node.js 是单进程单线程应用程序，但是通过异步回调可以处理大量的并发。Node.js 事件机制是基于设计模式中观察者模式构建的，如图 2.5 所示，Node.js 单线程运行原理就类似于进入一个 `while(true)` 的事件循环，每个事件都生成一个事件观察者，当有事件发生就调用该事件的异步回调函数，直到没有事件观察者后退出事件循环。

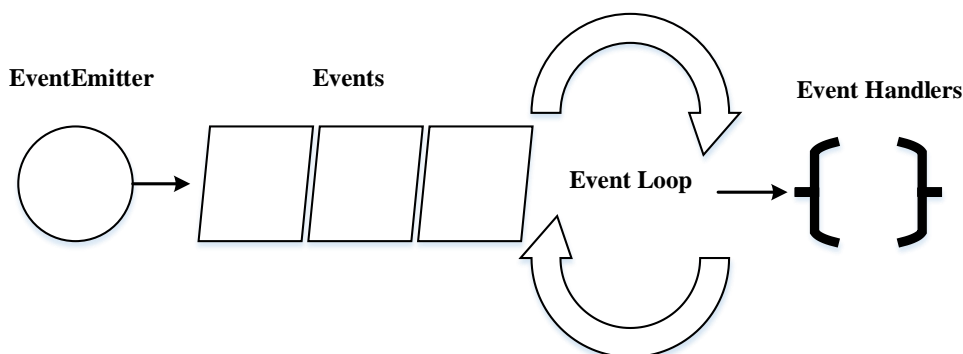


图 2.5 Node.js 事件驱动机制

Node.js 有很多的内置的事件，还可以创建自定义事件。通过引入 `events` 模块，并实例化 `EventEmitter` 类来绑定和监听自定义事件。`EventEmitter` 类的核心就是事件触发与事件监听器功能的封装。自定义事件的创建步骤如下：

(1) 引入 `events` 模块

```
var events = require ( 'events' );
```

(2) 建立 `eventEmitter` 对象

```
var eventEmitter = new events.EventEmitter();
```

(3) 绑定事件及事件的处理程序

```
eventEmitter.on( 'eventName' , eventHandler);
```

(4) 触发事件

```
eventEmitter.emit( 'eventName' );
```

2.3.2 MongoDB 数据库

MongoDB 是一个灵活的、可扩展性强的非关系型数据库。非关系数据库的主要优势在于可以有效地处理非结构化数据，如文档、电子邮件等^[33]。传统的关系型数据库系统，如 SQL 等，专注于一致性，强调事务（一个或多个 sql 语句组成的独立工作单元）是一个整体，事务执行结果必须是使数据库保持一致性状态，如果只有部分事务完成则会被会丢弃。并且事务需要在没有任何干扰的情况下独立处理完成。所有提交的事务都保存在日志中，不会丢失。而非关系型数据库系统更关注可用性和分区，数据库系统的状态是动态的，数据库的所有副本不必一直保持一致。在任何写入、更新或删除操作之后，系统可能不会立即反映所做的修改。但是最终在所有副本中显示修改后的数据一致即可^[34]。所以非关系型数据库的效率要高于关系型数据库。

MongoDB 数据库是用 C++编写的，具有很强的可移植，可以在运行在 Linux、MacOS、Windows 等多种操作系统上。MongoDB 数据库使用集合来组织数据，集合（collection）就类似于关系型数据库中的表（Table），但是表是严格定义的，只能将指定的项放入表中，而集合是灵活的，其中的数据项可以不相同。文档（document）就相当于关系型数据库的记录（Record）。在 MongoDB 数据库中，集合不执行严格的模式，在同一个集合中的文档可以有不同的结构，而关系型数据库表中的结构需要保持一致。文档的存储数据为 BSON 格式，二进制 JSON 的缩写。使用 BSON 格式使得 MongoDB 数据库处理速度更快，BSON 还添加了一些标准 JSON 中无法提供的特性，包括数据的扩展类型(如 int32 和 int64)，以及对处理二进

2.4 总体设计方案

近年来，物联网技术应用在包括工业、农业、医疗等各个领域中，相关应用的架构总体可分为三层：感知层、网络层和应用层^[35]。感知层主要由各种传感器、执行器和通信模块等组成，主要完成信息采集与控制。网络层的主要作用是作为感知层与应用层之间的数据交换通道，将感知层接入网络，进行信息的远程传输。应用层类似于传统的上位机，作用是对感知层上传的相关数据进行接收、存储、计算和处理等。上传的数据被用来支持在各种应用中提供特定的服务^[36]；或者在某些应用场景下，向感知层发送指令控制感知层执行器工作，满足特定的控制需求^[37]。

本文基于物联网的三层架构对整个消毒监控系统进行设计。如图 2.6 所示，该系统包括消毒机设备终端和监控平台两个部分。数据通道包括向上通道和向下通道，消毒机设备终端采集传感器数据，通过向上数据通道传至监控平台；监控平台通过向下通道推送指令到设备端，控制消毒机工作。现将系统简单介绍如下：

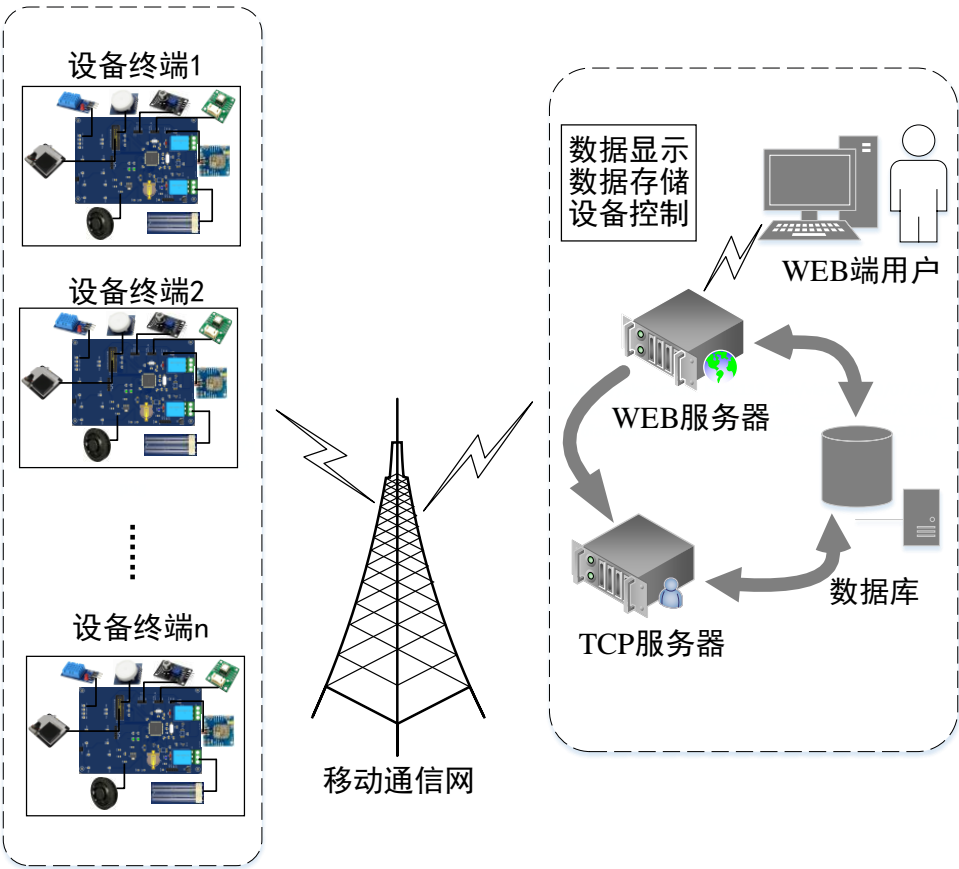


图 2.6 系统整体框架结构图

消毒机设备终端采用 STM32F103VET6 芯片作为微控制器，负责接收、封装传感器数据，

解析监控平台下发指令并控制执行器。选用 NB-IoT 模块实现 TCP 客户端，与监控平台 TCP 服务器进行双向的数据通信。执行器包括紫外线灯和离心风机，由离心风机将室内空气带入消毒机内部，在紫外线灯的照射下达到空气消毒的目的。并结合消毒机的实际功能需求，在设备终端加入触摸屏和红外接收模块，作为终端补充的控制交互方式。

监控平台使用 Node.js 开发，主要分为两个部分：第一部分是 TCP 服务器，用于接收设备终端通过移动通信网络传输的传感器数据。对数据进行处理后存储到 MongoDB 数据库。其中，设备终端与监控平台之间的数据传输采用 JSON 格式。第二部分为 web 服务器，负责向应用层服务提供数据接口，实现网页端数据显示和用户指令下发的功能。

2.5 本章小结

本章首先分析了系统需求，明确了功能指标，随后对实现物联网通信的无线通信技术进行了简单介绍和对比，并对 NB-IoT 实现覆盖增强、海量连接、低功耗的技术进行阐述。而后对实现监控平台功能所使用的相关技术进行了概述。最后给出了消毒监控系统总体的设计方案。

第三章 空气消毒监控系统硬件设计

3.1 设备终端硬件框架

本章主要说明消毒监控系统中消毒机设备终端的硬件设计。如图 3.1 所示，其硬件部分大致可分为：1) 微控制器，负责用于接收、封装传感器数据，解析监控平台下发指令并控制执行器；2) 环境参数采集模块，由多种传感器组成，采集的参数包括 PM2.5、PM10、甲醛、挥发性有机物、二氧化碳、温度和湿度；3) NB-IoT 无线通信模块，用于构建 TCP 客户端，实现与监控平台 TCP 服务器进行双向的数据通信；4) 执行器模块，包括紫外线灯和离心风机，由离心风机将室内空气带入消毒机内部，在紫外线灯的照射下达到空气消毒的目的；5) 终端交互模块，包括触摸屏和红外接收器，使用触摸屏或红外遥控器对消毒机设备终端进行相关功能控制，作为远程控制以外的控制方式；6) 电源管理模块，包括开关电源和降压电路，由开关电源将市电 AC220V 转换为 DC24V，给离心风机供电。再通过降压电路将 24V 分别转换为 5V 和 3.3V 给相应模块供电。

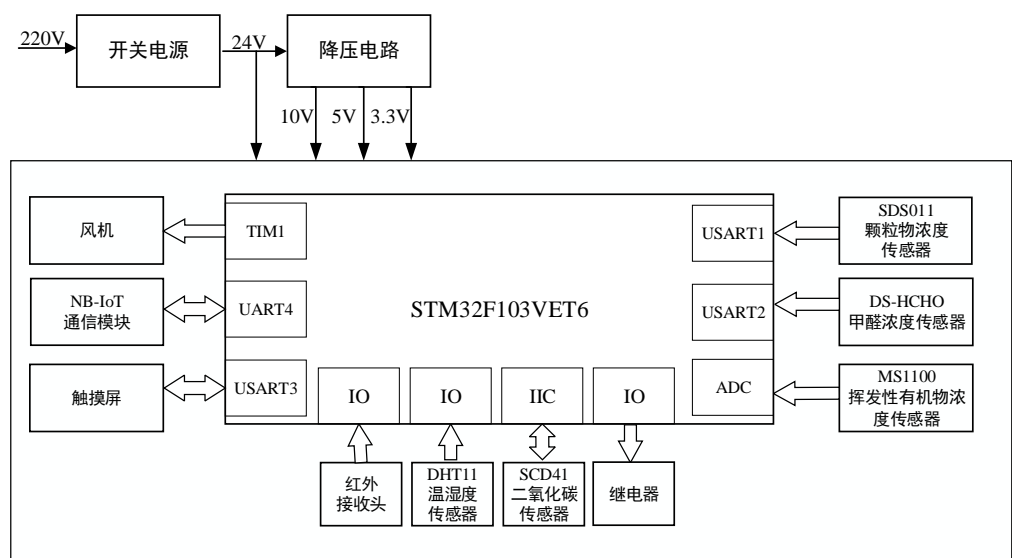


图 3.1 设备终端硬件框架

3.2 微控制器及硬件模块接口连接方式

消毒机设备终端采用 STM32F103VET6 芯片作为微控制器。STM32F103VET6 是一款 32 位高性能的微控制器，采用 ARM Cortex-M3 内核，最高运行速度为 72MHz。芯片内部集成了高速存储器以及各种增强型 I/O 和外围设备^[38]，丰富的片上外设大大增强了可扩展性，满

足消毒机设备终端信息采集和控制的需求。在本设计中，微控制器与传感器连接使用到的片上外设就包括 IO、USART、I²C、ADC 等。在下文中，将简单介绍微控制器及硬件模块接口连接方式以及使用到的微控制器片上外设。

IO（Input/Output）是输入输出并行接口，可分为 GPIO（General-purpose input/output，通用型输入输出）和专用 IO。IO 有 8 种工作模式，分别是浮空输入、带上拉输入、带下拉输入、模拟输入、开漏输出、推挽输出、复用开漏输出、复用推挽输出^[39]。

USART（Universal Synchronous/Asynchronous Receiver and Transmitter，通用同步/异步收发器）用于设备之间进行全双工数据通信^[40]。在串口通信的协议层规定了数据包的构成，包括起始位、主体数据、校验位以及停止位，如图 3.2 所示。STM32F103VET6 有 3 个 USART 和 2 个 UART，在外设控制寄存器中可设置波特率大小、数据帧字长、停止位长度、奇偶校验位等数据包信息。通讯双方的数据包格式必须在事先要约定一致，才能正确地进行数据的收发。

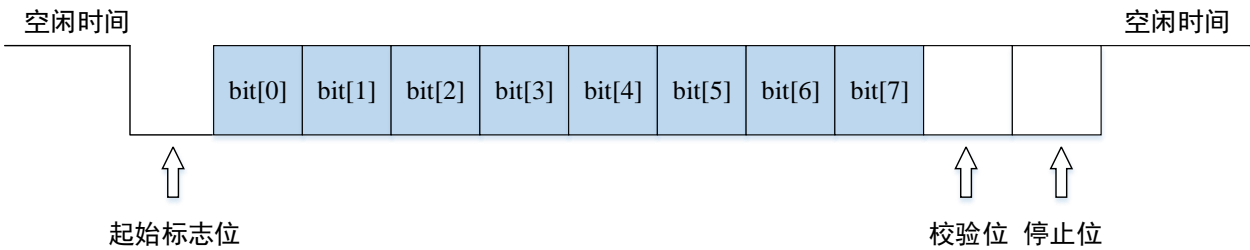


图 3.2 串口通信数据包组成

I²C 通信协议由飞利浦公司开发，特点在于使用引脚少，硬件实现简单，可扩展性强，广泛地应用在微控制器与传感器或系统 IC 之间通讯。如图 3.3 所示，I²C 协议物理层使用两条总线，双向串行数据线（SDA）用来传输数据，串行时钟线（SCL）用于通讯双方数据收发同步。STM32 的 I²C 片上外设用于实现 I²C 通信协议。在使用时，配置 I²C 片上外设相关控制寄存器，就可以自动根据协议要求产生通讯信号。在程序中通过检测外设的状态和数据寄存器，就能完成数据收发。

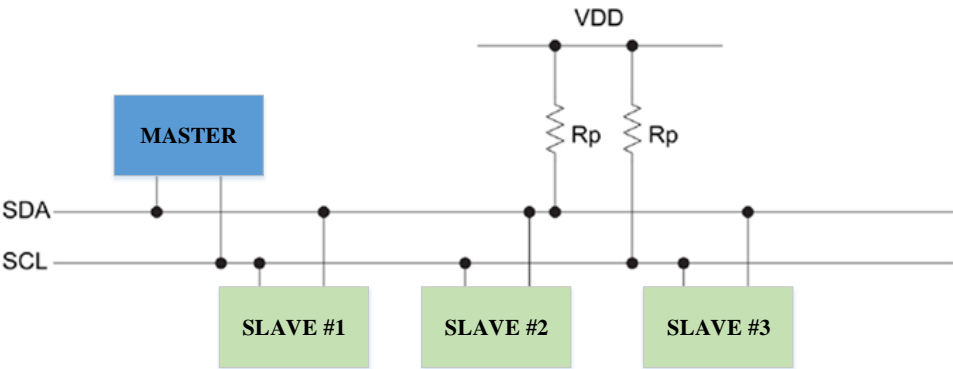


图 3.3 常见的 I²C 通讯系统






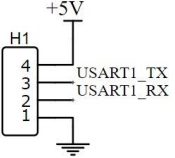
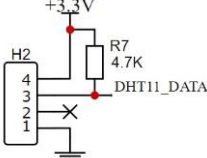
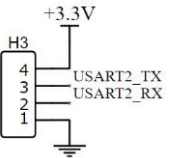
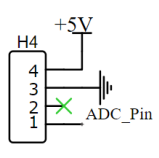
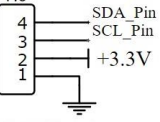
STM32F103 系列有 3 个 ADC（Analogue to Digital Conversion 模数转换）外设,外部信号由通道输入至 ADC，模拟电压经过转换后，存储到数据寄存器中，再通过输入电压范围，可计算出模拟电压值^[41]。

STM32F103VET6 有 4 个通用定时器、2 个高级定时器和 2 个基本定时器。基本定时器只支持向上计数的定时功能。通用定时器支持向上或向下计数，可以实现定时、输出比较和输入捕获功能，并且有 4 个外部 IO。高级定时器在通用定时器功能的基础上，还可以用于生成三相电机互补输出信号，每个高级定时器有 8 个外部 IO。在本设计中，除了用到定时器的定时功能，还用到了通用定时器的输出比较功能。输出比较就是通过定时器的外部引脚对外输出控制信号^[42]。在控制离心风机时用到了通用定时器 PWM 输出模式。

3.3 传感器模块

参考《室内空气质量国家标准 GB/T18883-2002》，选择监测参数包括 PM2.5、PM10、甲醛、甲苯、二氧化碳、温度和湿度。结合性能及成本考虑，最终选用的传感器如下：SDS011 激光传感器测量 PM2.5 和 PM10，DHT11 传感器测量温度和湿度，DS-HCHO 传感器测量甲醛，MS1100 传感器测量甲苯浓度，SCD41 传感器测量二氧化碳浓度。选用的传感器参数如表 3.1 所示。

表 3.1 传感器参数

传感器	SDS011	DHT11	DS-HCHO	MS1100	SCD41
量程	0-999.9ug/m ³	湿度：20%-95%RH 温度：0-50℃	0-7.5mg/m ³ 或 0-10ppm	0-75ppm	0-40000ppm
工作电压	5V	3.3V~5V	3.3V	5V	2.4V-5.5V
误差	5%	湿度：±5% 温度：±2℃	±5%	±7%	±40ppm
连接方式	UART	单总线	UART	ADC	I ² C
模块实物图					
接口电路					

SDS011 使用激光检测原理, 测量空气中悬浮颗粒物的浓度^[43]。本设计中使用的 SDS011 传感器模块采用数字化串口输出方式, 直接输出检测到的 PM10 和 PM2.5 浓度。

DS-HCHO 使用电化学原理实时检测室内环境中的甲醛含量,通过串口数字输出浓度信息^[44]。

DHT11 传感器内置电阻式感湿元件和 NTC 测温元件，用于测量湿度和温度。

MS1100 是挥发性有机化合物传感器,可以用于检测空气中 0.1ppm(parts per million, 百万分比)以上的挥发性气体,包括苯、甲醛、甲苯等^[45]。

SCD41 是使用光声传感技术设计的二氧化碳传感器，能在保持高性能的同时大幅缩小传感器尺寸。与红外气体测量感应技术不同的是，光声传感技术的信号是随着二氧化碳的浓度增加而增强，所以传感器可以实现更高的精度。

3.4 NB-IoT 通信模块

NB-IoT 具有低成本、低功耗、广覆盖和海量连接的特点^[46]。消毒机设备终端选用移远通信设计的 BC20 模组。BC20 是一款低功耗、高性能的 NB-IoT 无线通信模块，支持多家运营商。如图 3.4，BC20 模组的大小为 18.7mm×16.0mm×2.1mm，能够最大限度地满足产品对尺寸的需求，有效地优化产品成本。BC20 有着丰富的外部接口，并且支持很多的网络协议栈，包括 TCP/UDP/MQTT/CoAP 等。可以使用丰富的 AT 命令与 BC20 模组进行交互控制。BC20 终端通过接入运营商网络进行数据传输，需要使用运营商提供的 SIM 卡，图 3.4 为 BC20 与 SIM 卡连接原理图。

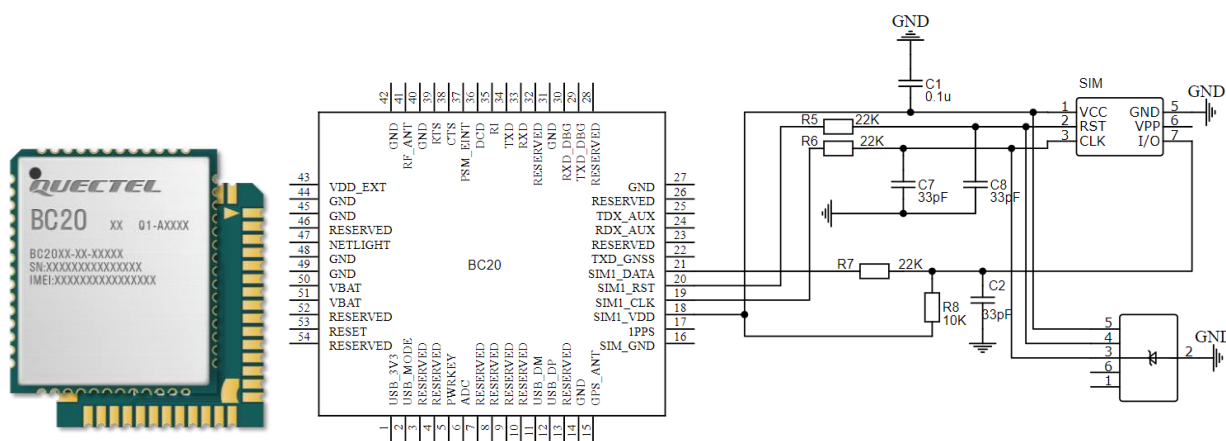


图 3.4 BC20 模块及 SIM 卡接口电路

3.5 设备终端交互模块

考虑到消毒机设备终端展示信息的全面性及交互的便利性，满足设备单独使用的功能，

选择触摸屏和红外遥控作为设备终端控制交互的额外方式。

3.5.1 触摸屏

触摸屏选用大彩科技生产的型号 DC10600M070 医用级显示屏，与微控制器通过 UART 连接。显示屏内置 400MHz 独立处理器，上电后即可运行。触摸屏可实现任意大小多种字库显示或自定义字体显示，还支持图片、音频、动画播放的功能。显示画面使用配套的上位机 VisualTFT 软件进行设计，支持包括按钮、文本、下拉框、进度条、仪表、曲线图、滑块、选择框、图标、动画、音频、时钟和二维码等控件。DC10600M070 显示屏实物图如图 3.5 所示。



图 3.5 触摸屏

3.5.2 红外遥控

红外遥控一般由红外发射装置和红外接收装置两个部分组成。消毒机设备终端配有红外接收装置，使用配套的红外遥控器完成控制交互。红外遥控器采用 NEC 编码，960nm 的波长，使用的晶振频率为 455kHz，对应的发射频率(载波频率)为 38kHz。终端配套的接收装置为 1838 脉冲型一体化红外接收头，电路集成在一个封装里，体积更小，可靠性更高。1838 红外接收头有三只引脚，即电源正、电源负和数据输出。红外接收头、遥控器和接口电路如图 3.6 所示。



图 3.6 红外接收头、遥控器及接口电路

3.6 执行器模块

循环风紫外线消毒机由紫外线灯、风机和过滤系统等组成，本设计是在传统的循环风紫外线消毒机的基础上，设计基于物联网的消毒监控系统，所以消毒机设备终端完成了相关执行模块的设计，包括离心风机和紫外线灯。在风机离心力的作用下，室内空气会被循环带入消毒通道，病毒或细菌在紫外线灯的照射下会被解体消亡。

3.6.1 风机

在本设计中选用直流离心风机，风机的额定电压为 24V，额定电流 2.4A，额定转速 2500RPM，风量最高可以到达 821.8m³/h，并且噪音控制在 62.0dB。图 3.7 为风机实物图和 PWM 升压电路，微控制器输出不同频率的 PWM 信号，增大其幅值至 10V 后控制风机工作在不同风速，将室内空气带入消毒机消毒通道。

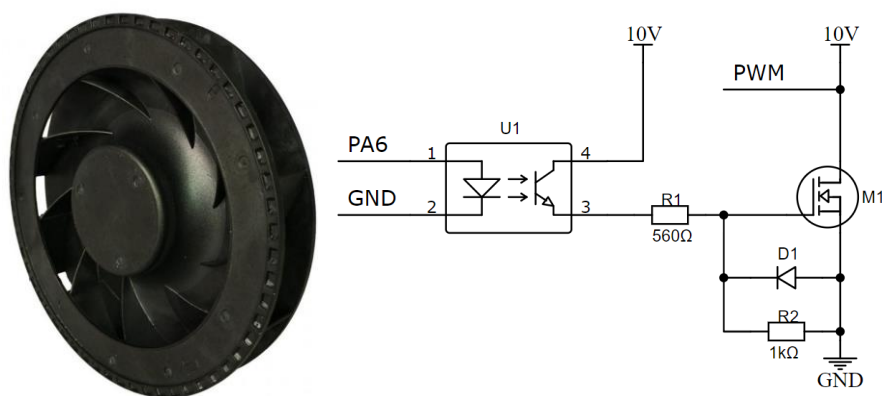


图 3.7 离心风机和 PWM 升压电路

3.6.2 紫外线灯

紫外线的消毒机理是通过短波长紫外线(UVC)照射破坏微生物的 DNA 来杀死或降低其活性。杀菌效果由照射剂量决定，同时与紫外线灯的种类、强度以及使用时长也存在关系。紫外线照射剂量公式如下：

$$K = I \times T \quad (3.1)$$

式中：I 为紫外线灯的强度(W/m²)，T 为照射时间(s)，K 为照射剂量(J/m²)。

本设计中使用 2 支无臭氧高强度紫外线灯，功率为 2.5W。能够保证在使用较长时间后，不会因为其中一只灯的老化而明显降低整体的消毒效率。由继电器控制紫外线灯开闭。继电器采用低电平触发，控制电路如图 3.8 所示。

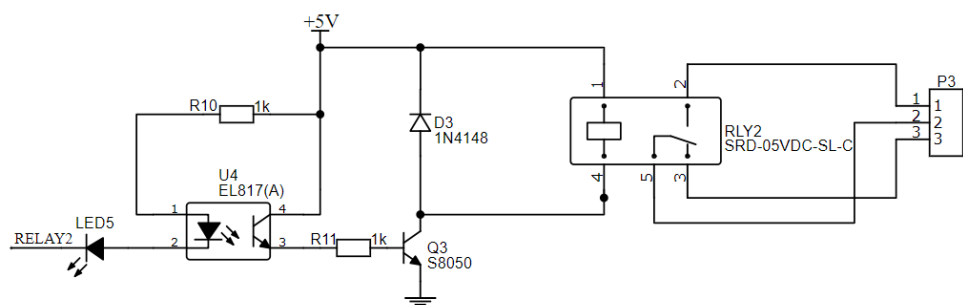


图 3.8 继电器控制电路

3.7 电源电路及 PCB 设计

在硬件系统中，STM32F103VET6 芯片、红外模块的工作电压为 3.3V，离心风机工作电压为 24V，紫外线灯、继电器、传感器工作在 5V。考虑到设备终端的安全性，选择带有过电压、过负载保护的开关电源将市电 AC220V 转换为 DC24V，直接给离心风扇供电；再通过 TPS54331D 降压芯片将 24V 分别转换为 5V 和 3.3V 给相应模块供电。TPS54331D 降压芯片的输出电压幅值通过改变输出端电阻阻值大小可计算得出。其中，24V 转 5V、24V 转 3.3V 电路原理图如图 3.9 所示。

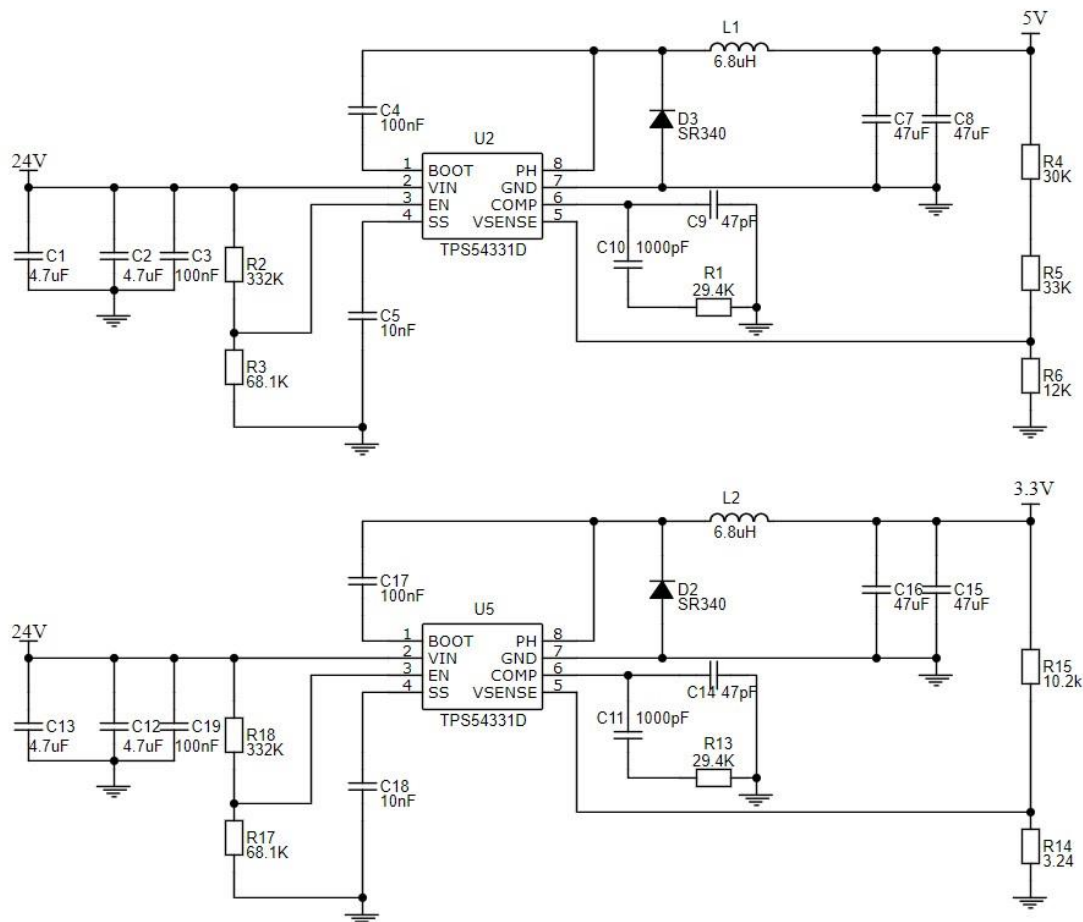


图 3.9 降压电路

以上分别介绍了控制系统各个硬件模块，根据所需要的调试电路、控制电路、驱动电路和电源电路等设计系统原理图，再根据原理图绘制 PCB。调试电路如图 3.10 所示。电路原理图和 PCB 采用立创 EDA 软件设计，采用双层板结构，将 STM32F103VET6 放置在 PCB 板中心，晶振靠近芯片。与传感器接口电路放在外围，电源线加粗，顶层和底层布线尽量垂直，降低干扰，并预留出相应的调试接口、下载接口，便于扩展功能。PCB 设计图如图 3.11。

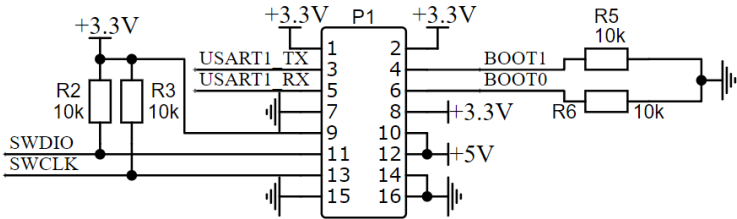


图 3.10 调试电路

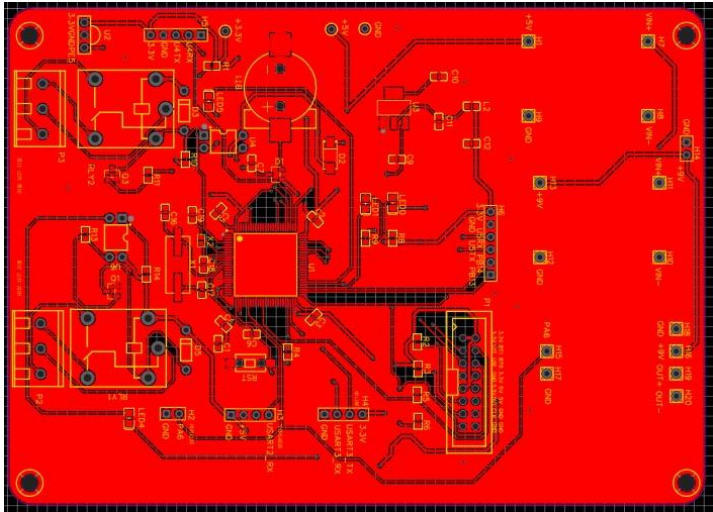


图 3.11 PCB 设计图

3.8 本章小结

本章根据系统需求和功能指标，给出了消毒机设备终端的硬件设计。从整体设备终端硬件框架到分模块的介绍，包括微控制器选型和微控制器与各模块接口连接方式、传感器模块选型和接口电路、通信模块参数介绍及典型 SIM 卡电路设计、交互模块设备参数、执行器模块电路设计，最后对系统的电源电路和 PCB 设计进行了阐述。

第四章 空气消毒监控系统软件设计

4.1 开发环境

本系统的软件设计主要包括设备终端软件设计和监控平台软件设计。其中设备终端软件在 Keil MDK-ARM uVision5 IDE 下开发。Keil MDK-ARM 包含了 Keil C 编译器、宏汇编器、调试器、实时内核等组件,支持 Cortex-M、Cortex-R4、ARM7 和 ARM9 系列芯片^[47]。Keil IDE 界面如图 4.1。

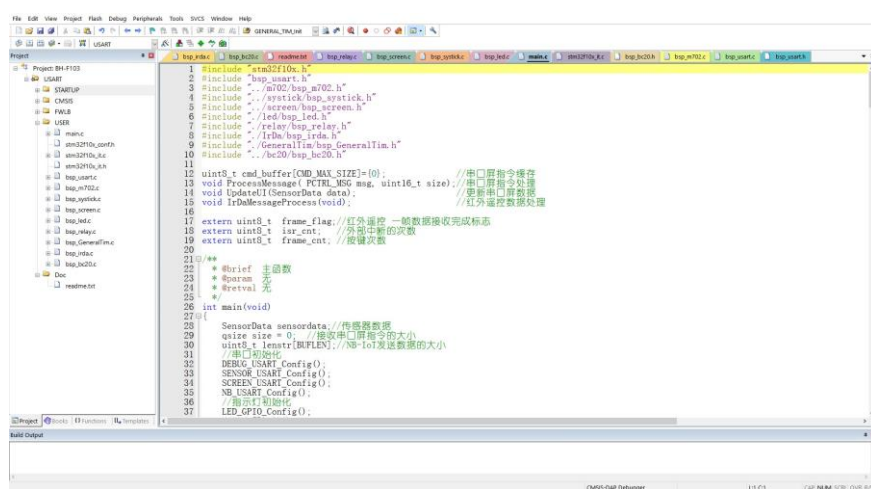


图 4.1 Keil IDE 界面

监控平台软件本地开发环境为 Node.js, Node.js 是一个开源的、跨平台的运行环境。使用 NPM 用于 Node 插件管理(包括安装、卸载、管理依赖等)。监控平台软件使用 Visual Studio Code (VS Code) 进行代码编辑。VS Code 是一款跨平台的免费源代码编辑器,可以使用内置命令行工具进行 Node.js 模块下载、程序运行等^[48]。VS Code 代码编辑器界面如图 4.2。

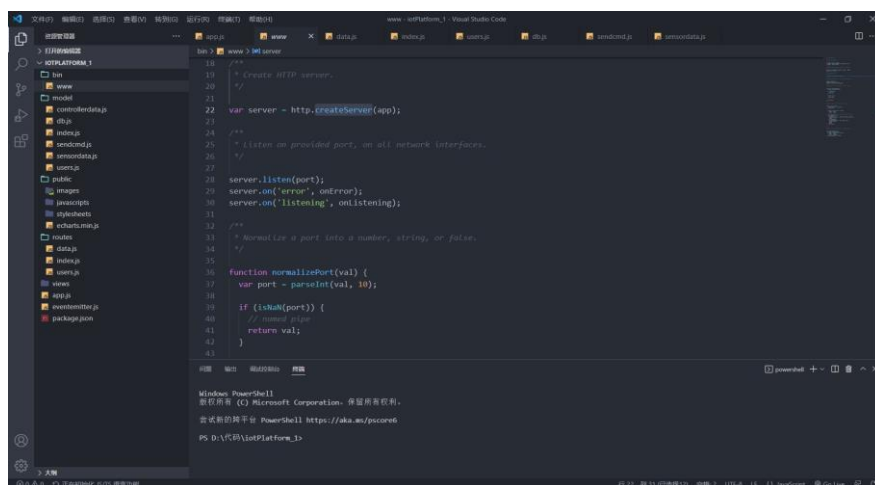


图 4.2 VS Code 编辑器界面

4.2 消毒机设备终端软件设计

消毒机设备终端软件采用 C 语言编写，将相关功能代码模块化，实现传感器数据解析、封装和上传以及控制指令解析，驱动执行等功能。嵌入式软件程序采用前后台结构设计，如图 4.3，结构包括一个无限循环和若干中断服务程序，应用程序是一个无限的循环，循环中调用相应的函数完成相关操作(后台)，中断服务程序用于处理软件系统的异步事件(前台)^[49]。在无限循环之前进行初始化工作，包括初始化 IO、定时器、USART、I2C、ADC 等外设和相关数据结构。前台中断服务程序用于处理异步事件，包括读取传感器数据、交互模块信息、通信模块数据等，所使用到包括定时中断、串口接收中断、EXTI 外部中断等等。在轮询中处理指令数据并控制相应的执行器，封装采集到的传感器数据发送到通信模块，完成数据上传。

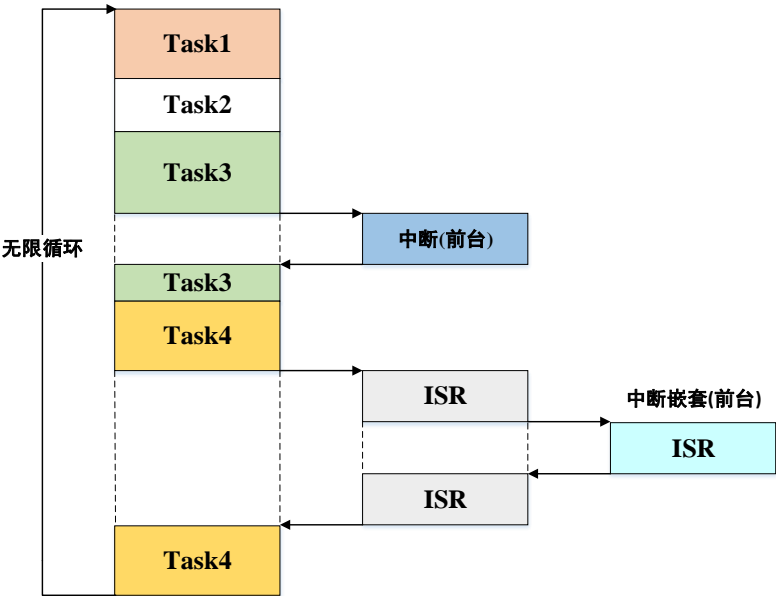


图 4.3 前后台结构

4.2.1 设备终端软件初始化

初始化包括微控制器片上外设的初始化、硬件模块的初始化、相关数据结构的初始化等等。片上外设的初始化包括所使用到的 IO、USART、定时器、ADC 等外设，外设的初始化大同小异，以通用定时器初始化和中断配置初始化为例大致介绍一下。

定时器在上一章节中有过介绍，通用定时器可以用做定时、输出比较和输入捕获，并且有外部 IO 引脚。在本设计中，使用通用定时器输出比较功能，输出 PWM 控制离心风机工作。初始化通用定时器需要初始化 IO、初始化时基和输出比较功能。初始化 IO 包括使能 IO 端口的时钟、配置 IO 口的工作模式，设置引脚速率等。代码如下，此处配置 IO 口的工作模式为

复用推挽输出。

```
static void GENERAL_TIM_GPIO_Config(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;
    RCC_APB2PeriphClockCmd(GENERAL_TIM_CH1_GPIO_CLK, ENABLE);
    GPIO_InitStructure.GPIO_Pin = GENERAL_TIM_CH1_PIN;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GENERAL_TIM_CH1_PORT, &GPIO_InitStructure);
}
```

初始化时基和输出比较功能代码如下，在此处设置时基，需要使能定时器时钟、配置自动重载的值、配置驱动计数器的时钟、选择计数器的计数模式。设置完成后，计数器的时钟频率为 1MHz。输出比较功能需要配置工作模式为PWM、输出通道的极性配置为高电平有效。最终引脚输出的PWM频率和占空比由函数参数GENERAL_TIM_Period和CCR_Val决定，输出频率等于 $1\text{MHz}/\text{GENERAL_TIM_Period}$ ，占空比等于 $\text{CCR_Val}/\text{GENERAL_TIM_Period}$ 。控制离心风机转速的PWM频率范围为 1-10kHz，测试多次后，预设好三组 10kHz频率的PWM，调节占空比使离心风机工作在三挡转速。

```
static void GENERAL_TIM_Config(uint16_t GENERAL_TIM_Period, uint16_t CCR_Val)
{
    GENERAL_TIM_APBxClock_FUN(GENERAL_TIM_CLK, ENABLE);
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
    TIM_TimeBaseStructure.TIM_Period = GENERAL_TIM_Period - 1;
    TIM_TimeBaseStructure.TIM_Prescaler = GENERAL_TIM_Prescaler;
    TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
    TIM_TimeBaseStructure.TIM_RepetitionCounter = 0;
    TIM_TimeBaseInit(GENERAL_TIM, &TIM_TimeBaseStructure);
    TIM_OCInitTypeDef TIM_OCInitStructure;
    TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
    TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
```

```
TIM_OCInitStructure.TIM_Pulse = CCR_Val;
TIM_OC1Init(GENERAL_TIM, &TIM_OCInitStructure);
TIM_OC1PreloadConfig(GENERAL_TIM, TIM_OCPreload_Enable);
TIM_Cmd(GENERAL_TIM, ENABLE);
}
```

在程序设计中，中断服务程序用于处理系统的异步事件。在使用相关中断前需要进行相应的初始化配置。STM32F103ZET6 中使用 NVIC (Nested Vectored Interrupt Controller, 嵌套向量中断控制器) 控制整个芯片中断相关的功能^[50]。NVIC 的结构包括中断使能寄存器、中断清除寄存器、中断使能悬起寄存器、中断清除悬起寄存器、中断有效位寄存器、中断优先级寄存器、软件触发中断寄存器。在中断初始化配置时，一般只需要配置中断使能寄存器、中断清除寄存器和中断优先级寄存器即可。中断的使能与清除分别使用中断使能寄存器、中断清除寄存器来控制。中断优先级寄存器，顾名思义，用来配置中断的优先级。优先级又分为抢占优先级和子优先级，数字越小，优先级越高。在中断初始化编程时，首先需要选择中断源，再配置中断优先级分组，设置中断的抢占优先级和子优先级，最后使能中断请求。每个中断源都有对应的中断服务函数进行处理。例如与 NB-IoT 连接串口中断配置和中断服务函数如下。

```
static void NB_USART_NVIC_Configuration(void)
{
    NVIC_InitTypeDef NVIC_InitStructure;
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);
    NVIC_InitStructure.NVIC_IRQChannel = NB_USART_IRQ ;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}

void NB_USART_IRQHandler(void)
{
    if(USART_GetITStatus(NB_USARTx, USART_IT_RXNE)==SET)
    {
        nbiot_receive_process_event(USART_ReceiveData(NB_USARTx));
    }
}
```

```
        USART_ClearITPendingBit(NB_USARTx,USART_IT_RXNE);
    }
    if(USART_GetFlagStatus(NB_USARTx,USART_FLAG_ORE)==SET)
    {
        nbiot_receive_process_event(USART_ReceiveData(NB_USARTx));
        USART_ClearFlag(NB_USARTx,USART_FLAG_ORE);
    }
}
```

4.2.2 传感器数据采集

在第三章中已经介绍了使用的传感器模块选型,包括 SDS011 颗粒物浓度传感器、DHT11 温湿度传感器、DS-HCHO 甲醛传感器、MS1100 甲苯传感器、SCD41 二氧化碳传感器。其中 SDS011、DS-HCHO 与微控制器之间通过串口通讯, DHT11 使用单总线的方式传输数据, MS1100 输出模拟电压值由微控制器 ADC 读取, SCD41 与微控制器通过 I²C 总线连接。使用串口通讯与下文介绍的触摸屏连接方式相同, 单总线传输数据与红外接收类似, 这里就不过多介绍。下面以 MS1100 为例, 介绍一下传感器数据采集软件设计。

MS1100 使用 ADC 读取, 外设的初始化与 TIM 类似, 包括初始化 IO 和配置 ADC 外设控制相关寄存器。程序编写时还使用到了 DMA 功能, 将数据直接从 ADC 外设数据寄存器传输到内部 SRAM 中的变量 ADC_ConvertedValue[0]中, 然后将数字量转换为电压值显示, 再根据气体浓度与模拟电压的计算公式, 进行气体浓度的计算。模拟电压转换为气体浓度代码如下:

```
void MQ_Value_Conversion()
{
    MQ_X_NUM=ADC_ConvertedValue[0];
    MQ_X=((MQ_X_NUM*3300.0)/4096)/0.66;
    GAS_ppm=MGGetPercentage(MQ_X,GASCurve);
}
```

每个传感器数据采集到后, 都会进行相应的数值校验, 如果数值异常, 数据被丢弃。传感器的数据会被保存到数据结构体中, 转换为字符格式后发送到触摸屏更新显示数据, 转换为 JSON 字符串发送到 NB-IoT 模块进行数据上传。

4.2.3 设备终端交互模块软件设计

触摸屏与微控制器通过串口连接，微控制器与触摸屏之间通过指令传递信息。一条完整的触摸屏指令帧格式如表 4.1 所示。完整指令的最大长度为 1024 字节，数值均为十六进制。当指令参数的长度大于 1 个字节，发送时高字节在前。串口数据格式约定数据位为 8 位、停止位 1 位、无校验位。

表 4.1 指令帧格式

指令	EE	XX	XX XX...XX	FF FC FF FF
说明	帧头	指令	指令参数	帧尾

下面介绍一下主要使用的两个触摸屏指令，一个是文本控件显示指令，由微控制器发送到触摸屏，用于显示传感器采集到的数据；另一个是按钮控件通知指令，触摸屏回传到微控制器，用于控制执行器工作。微控制器向触摸屏发送文本显示指令，将采集到的传感器数据显示到触摸屏上，例如需要显示采集到的 CO₂ 浓度值为 430ppm，画面 ID 为 0，控件 ID 为 1，微控制器发送的具体指令解析如表 4.2。

表 4.2 文本控件显示指令

微控制器发送指令	EE 【B1 10 00 00 00 01 34 33 30】 FF FC FF FF
命令解析	EE 表示帧头
	B1 10 表示发送的组态控件指令
	00 00 00 01 表示画面 ID 为 0，控件 ID 为 1
	34 33 30 表示数字 430 的 ASCII 码
	FF FC FF FF 表示帧尾

触摸屏向微控制器回传按钮控件通知指令，例如按下控制紫外线灯打开或关闭的按钮，画面 ID 为 0，控件 ID 为 12，具体指令解析如表 4.3。

表 4.3 按钮控件通知指令

屏幕回传指令	EE 【B1 11 00 00 00 0C 10 01 01】 FF FC FF FF
命令解析	EE 表示帧头
	B1 11 表示回传的组态控件指令
	00 00 00 0C 表示画面 ID 为 0，控件 ID 为 12
	10 表示控件为按钮控件
	01 表示按钮控件属性为开关类型
	01 表示按钮状态由弹起变成按下
	FF FC FF FF 表示帧尾

在编写微控制器发送触摸屏指令代码时，只需要将数据信息和控件指令组合逐一通过串口发送即可。在编写读取屏幕回传指令时，需要先对指令进行缓存再进行解析。回传指令在串口接收中断中被缓存至循环队列中，首先判断循环队列是否有完整的指令，再通过触摸屏指令解析函数处理具体指令，控制执行器工作。

触摸屏显示画面通过配套的上位机 VisualTFT 软件进行图形化设计。如图 4.4 所示。



图 4.4 触摸屏显示画面

消毒机设备终端还可以通过红外遥控来控制，设备终端配套的接收装置为 1838 脉冲型一体化红外接收头。程序原理：在 IO 中断中，检测与红外接收头连接的引脚电平，通过 SysTick 延时中断来检测高电平的时间。初始化时配置 SysTick 延时中断的优先级高于 IO 中断，SysTick 延时中断才能抢占 IO EXTI 中断，起到计时的作用。高电平的时间为 4ms-4.5ms 表示开始信号，0.2ms-1ms 之间表示数据 0，1ms-2ms 表示数据 1，2ms-4ms 表示一帧数据接收完成。红外帧数据有 4 个字节，第 1 个字节是遥控的 ID，第 2 个字节为第 1 个字节的反码，第 3 个字节是遥控的真正的键值，第 4 个字节是第 3 个字节的反码。遥控信号处理程序根据键值的不同，控制不同的执行器工作。

4.2.4 通信模块软件开发

微控制器通过 AT 命令控制 NB-IoT 模块工作。AT 命令是以 AT 开头，响应数据包在中，字符结尾的字符串。每个 AT 命令不管是否执行成功都有相应的返回。BC20 模块实现的 AT 命令可以在语法上分为两类：基础类和拓展类。基础类 AT 命令的格式为 AT<x><n>或 AT+<x><n>（尖括号中为参数名称,实际命令行中不包含尖括号），其中<x>是命令，<n>是该命令的参数。拓展类 AT 命令可以在多种模式下运行，如表 4.4 所示：

表 4.4 AT 命令格式

测试命令	AT+<cmd>=?	返回相应设置命令或内部程序可支持的参数取值列表或范围。
查询命令	AT+<cmd>?	返回相应设置命令的当前参数设置值。
设置命令	AT+<cmd>=<p1>[,<p2>[,<p3>[...]]]	设置用户可自定义的参数值。
执行命令	AT+<cmd>	主动执行内部程序实现的功能集。

BC20 模块每次仅支持执行一个 AT 命令，在上一个命令执行完成后，才能执行下一个命令。在程序中编写 AT 命令时，直接使用字符串的形式即可。相关的特殊符号需要加上转义字符。在 AT 命令的末尾需要加上回车换行符，即\r\n，通知模块一个 AT 命令发送完毕。

BC20 模块支持命令和数据两种串口模式。在命令模式下，通过串口输入的数据会被视为 AT 命令去解析。在数据模式下，通过串口输入的数据则被视为普通数据。

设备终端通信程序设计：微控制器通过 UART 向 NB-IoT 模块发送 AT 命令，检测模块返回数据。相关程序可分为 4 个功能函数模块 BC20_Initialize、BC20_ConnectTCP、BC20_SendData、BC20_ReceiveData。

在 BC20_Initialize 功能函数模块中检测模块是否正常工作并接入网络，确保后续 TCP 传输操作是在模块获取到 IP 地址后再进行。使用的 AT 命令如表 4.5 所示：

表 4.5 BC20_Initialize 功能模块使用的 AT 命令

命令	响应	功能
AT	成功，“OK”字符串返回 执行失败，“ERROR”字符串返回。	检测模块是否正常连接
AT+CIMI	响应: <IMSI> OK 若出现任何错误: ERROR 或者 +CME ERROR: <err>	查询(U)SIM 卡的国际移动用户识别码 (IMSI)
AT+QBAND?	响应:+QBAND: (支持的 <band_number>范围),[(支持的 <operating_band>列表)] OK	查询当前注册的频段
AT+CSQ?	响应: +CSQ:(支持的<rssi>列表),(支持的<ber>列表) OK	返回接收信号强度指示<rssi>和信道误码率<ber>
AT+CEREG?	响应: +CEREG: <n>,<stat> OK 若出现任何错误: ERROR 或者 +CME ERROR: <err>	查询网络注册状态
AT+CGPADDR=1	响应: +CGPADDR: <cid>[,<PDP_addr_1>[,<PDP_addr_2>]] OK	返回设备的 IP 地址。

BC20_ConnectTCP 功能函数模块用于创建 socket（套接字），建立 TCP 客户端，与监控平台 TCP 服务器建立连接。socket 是在网络上运行的两个程序之间的双向通信链路的一个端点，端点是 IP 地址和端口号的组合^[51]。套接字绑定到端口号，TCP 层可以识别数据要发送到的应用程序。图 4.5 为 socket 通信的流程，通信服务器侦听客户端的连接请求，客户端通过正在运行通信服务器的计算机 IP 地址以及服务器正在侦听的端口号进行连接请求，连接成功后会创建一个套接字，客户端和服务端可以通过写入或从套接字中读取来进行通信。

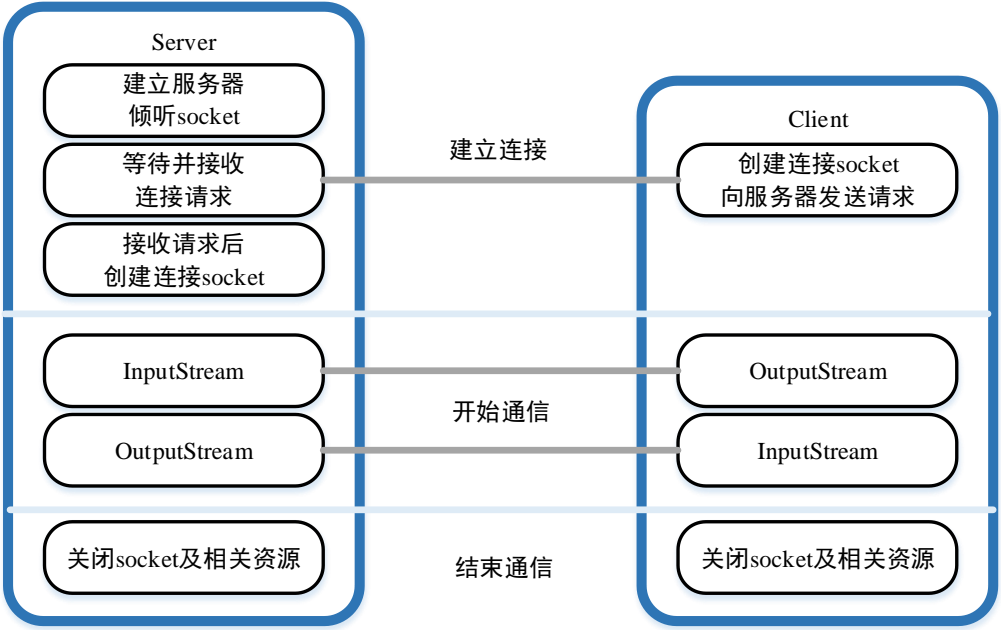


图 4.5 socket 通信流程

BC20_ConnectTCP 功能函数模块使用到的 AT 命令为 AT+QIOPEN，功能是为了打开 Socket 以创建 TCP 或 UDP 连接。如

```
AT+QIOPEN=1,0,"TCP","120.26.15.240",3307
OK
+QIOPEN: 0,0
```

第一个参数 0 为 socket 标识号，第二个参数选择使用 TCP 还是 UDP 进行连接，第三个参数为部署通信服务器计算机的 IP 地址，第四个参数为通信服务器程序正在侦听的端口号。返回 OK +QIOPEN: 0,0，表示建立连接成功。消毒机设备终端与监控平台之间采用 TCP 连接，IP 地址为 120.26.15.240，TCP 服务器监听的端口号为 3307。

BC20_SendData 功能函数模块用于在建立 TCP 连接后发送传感器数据至监控平台的 TCP 服务器，所使用到的 AT 命令是 AT+QISEND，用于发送十六进制/文本字符串数据。发送数据前可配置发送的数据格式，默认格式为 Text 字符串。发送 Text 字符串数据又分为普通模式、定长数据模式和不定长数据模式。普通数据模式就如上文提到的 BC20 串口模式中的命令模式一样，在 AT 命令中包含所要传输的数据。定长数据模式和不定长数据模式会进入数据模式，等待数据的输入，定长数据模式会指定发送数据的大小，不定长数据模式则不用。程序中使用的普通模式来发送数据，如：

```
AT+QISEND=0,93,"{"CO2":"420","CH2O":"1","TVOC":"2","PM25":"12","PM10":"16","temperature":"26","humidty":"40"}"
OK
```


SEND OK

命令中第二个参数为发送数据的长度，第三个参数为数据，数据长度参数和数据实际长度保持一致。数据使用 JSON 格式传输，在发送 JSON 格式等特殊字符数据时，必须使用双引号将数据包围，在程序编写时注意使用双引号等特殊字符时需要加转义符。

BC20_ReceiveData 功能函数模块用于接收远程控制指令，使用的 AT 命令是 AT+QIRD，用于读取 TCP/IP 数据，同样也可以配置接收的数据格式，与发送一样使用的也是 Text 字符串。

```
AT+QIRD=0,34
```

```
+QIRD: 34,0
```

```
{"uvlStatu": true, "speed": "1"}
```

OK

其中 AT 命令后 0 表示 socket 标识号，第一个参数 34 表示读取缓存中的数据长度(最大为 512 字节)，+QIRD:后跟的参数表示读取的长度为 34 字节，还剩余 0 字节未读取，读取到的 34 字节数据为: "{"uvlStatu": true, "speed": "1"}"。

BC20_Initialize、BC20_ConnectTCP 功能函数在初始化时调用，BC20_SendData 和 BC20_ReceiveData 在后台轮询中调用。当传感器数据读取封装完毕调用 BC20_SendData 将数据传递至 NB-IoT 模块进行发送。BC20_ReceiveData 调用后，在串口中断中读取监控平台发送的数据，调用数据解析函数解析为具体的控制指令，控制执行器工作。

4.3 数据交换格式

消毒机设备终端与监控平台之间进行数据交换需要使用一种数据交换格式，目前比较通用的数据交换格式有 XML、JSON、CSV 等。CSV 使用新行分隔行，用逗号分隔字段，格式本身不带有结构信息，需要自己定义每个域表示的意义。XML 格式复杂、数据量大，无论是在服务器端还是在客户端生成、处理、解析 XML，都需要花费大量代码，降低开发效率^[52]。JSON 是一种轻量级的文本数据交换格式，与 CSV 不同，可以存储结构化内容，相较于 XML 来说更小，更快，更容易解析和理解。在设备终端与监控平台数据通信中，使用合适的数据交换格式可大大减少数据处理的时间开销。考虑到监控平台所使用的编程语言是 Node.js，处理解析 JSON 格式十分方便，并且采用的是类 JSON 存储结构的 MongoDB 数据库，因此 JSON 是消毒机设备终端与监控平台之间理想的数据交换格式。

JSON 的三种语法：第一种键/值对 key: value 的形式，比如"deviceId":"1234"。第二种文

档对象，可以包含多个键/值对，比如{"deviceId":"1234","ip":"121.0.0.1"}。第三种为数组，JSON 数组在方括号中书写，数组成员可以是对象，值，也可以是数组。

设备终端程序使用 C 语言编写，在 C 语言中有专门实现 JSON 格式的解析库 cJSON，在程序中参照相关函数实现了数据与 JSON 相互转换的功能，将上传的数据转换为 JSON 格式发送，接收到 JSON 形式的控制指令数据后进行解析，得到想要的指令数据。监控平台程序采用 Node.js 开发，在 JavaScript 中处理 JSON 不需要任何特殊的 API 和工具包，使用 JSON.stringify()和 JSON.parse()方法就可以实现 JSON 字符串与对象之间的转换。

4.4 监控平台软件设计

消毒监控平台软件设计主要分为两个部分，如图 4.6 所示。第一部分是通信服务器，即构建 TCP 服务器用于接收设备终端通过移动通信网络传输的传感器数据，对数据进行处理后存储到数据库中；用于发送控制指令，控制消毒机设备终端执行器工作；第二部分为 web 服务器，负责向应用层服务提供数据接口，实现设备管理、网页端数据显示和用户指令下发的功能。

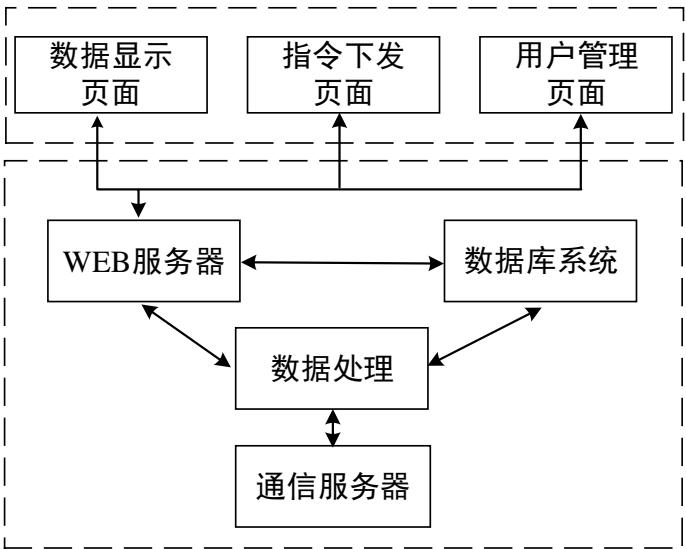


图 4.6 监控平台架构

监控平台采用 Node.js 开发，使用 express 框架构建 web 服务器，net 模块构建 TCP 服务器。监控平台实现的主要功能包括数据通信、数据存储、数据显示等。以 MVC 开发模式进行代码的编写^[53]。如图 4.7 所示，Model 层处理数据库相关事务，Controller 层负责执行不同的业务流程，包括路由配置、登录拦截、网络通信等，View 层负责将 Controller 层返回的结果通过模板引擎的方式渲染到页面。

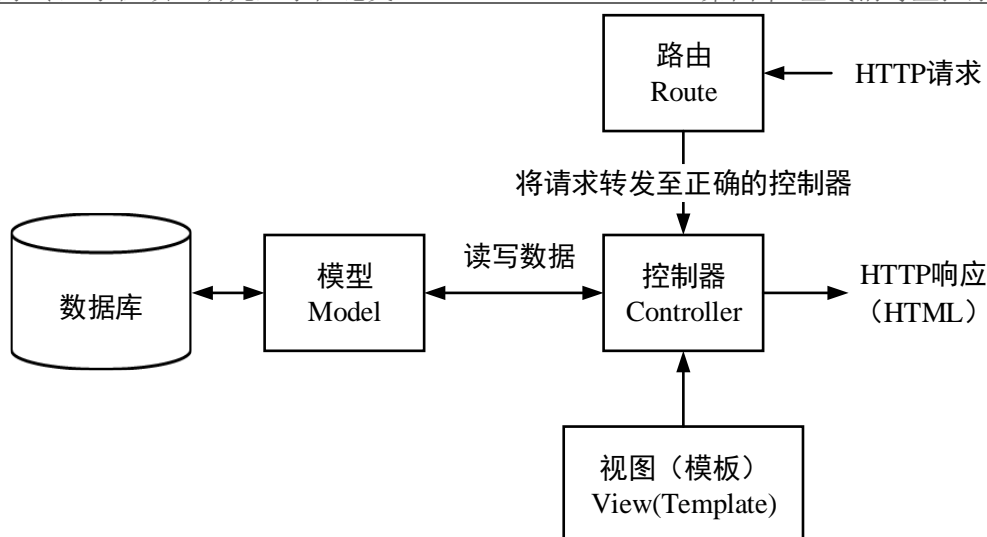


图 4.7 MVC 模式

4.4.1 数据库设计

与 MongoDB 数据库交互有两种方法，第一种是使用数据库的原生查询语言，第二种是使用对象数据模型或对象关系模型^[54]。对象数据模型或对象关系模型能将数据表示为 JavaScript 对象，再将它们映射到底层数据库。使用对象数据模型或对象关系模型的好处在于可以继续用 JavaScript 对象的思维而不用转向数据库语义的思维。在监控平台软件设计中，使用 Mongoose 操作 MongoDB 数据库。Mongoose 是一种对象模型工具，可以在异步的环境下执行。Mongoose 封装了 MongoDB 数据库对文档增删改查的常用方法，使得 Node.js 操作 MongoDB 数据库变得更加灵活简单。Mongoose 的使用步骤首先需要使用上文提到的 NPM 包管理工具安装 Mongoose，然后在程序中连接到 MongoDB 数据库并定义和添加模型，最后使用模型进行数据库操作。连接数据库代码如下：

```
const mongoose = require('mongoose');
const url = 'mongodb://localhost:27017/iot'
mongoose.connect(url, { useNewUrlParser: true, useUnifiedTopology: true }, function(err) {
  if(err) {
    console.log("数据库连接失败", err);
  } else {
    console.log("数据库连接成功");
  }
})
```

如图 4.8，数据库创建了 controllerdata、sendcmd、sensordata、users 四个集合，分别用来

存储执行器状态数据、发送的历史指令、传感器数据、以及监控平台用户信息。

```
> use iot
switched to db iot
> show collections
controllerdata
sendcmd
sensordata
users
```

图 4.8 数据库集合

4.4.2 通信服务器设计

Node.js 中 net 模块提供了用于底层网络通信工具的创建方法。监控平台使用 net 模块提供的方法创建 TCP 服务器，与消毒机设备终端实现双向数据通信。Node.js 的主要特点在于非阻塞机制和事件驱动架构。事件驱动的实质是让线程一直循环执行事件轮询，当事件发生后便触发相应的事件处理函数^[55]。在 Node.js 提供的 API 中，都支持回调函数。net.Server 类中“connection”事件在新连接建立时触发，“close”事件在连接关闭时触发，“error”事件用于捕获错误。通信服务器接收数据流程如图 4.9 所示。程序具体实现监控平台 TCP 服务器与消毒机设备终端 TCP 客户端数据通信步骤如下：

- (1) 在监控平台代码中引入 net 模块：`var net = require('net');`
- (2) 创建 TCP 服务器：`var tcpServer = net.createServer();`
- (3) 设置监听端口，启动连接服务器：`tcpServer.listen(3307, function(){.....})`
- (4) 在创建 TCP 服务器后，对“connection”、“close”、“error”事件进行监听，等待消毒机设备终端的连接。消毒机设备终端连接后触发 server 的“connection”事件，socket 传给触发的“connection”事件的监听器，然后就可以使用 socket 与客户端进行交互。

```
tcpServer.on('connection',function(socket){
    .....
    dealRecivedData(socket);
    .....
})
```

- (5) 当接收到数据时触发 socket 类中“data”事件，在“data”事件中接收消毒机设备终端上传的数据，解析数据及设备 IP 地址等信息并将数据存储在数据库中。

```
socket.on("data", async function(data){
```

```
.....  
receivedData = JSON.parse(data);  
.....  
model.connect(function(db){  
    db.collection('sensordata').insertOne(receivedData, function(err, doc){  
        .....  
    })  
})  
})
```

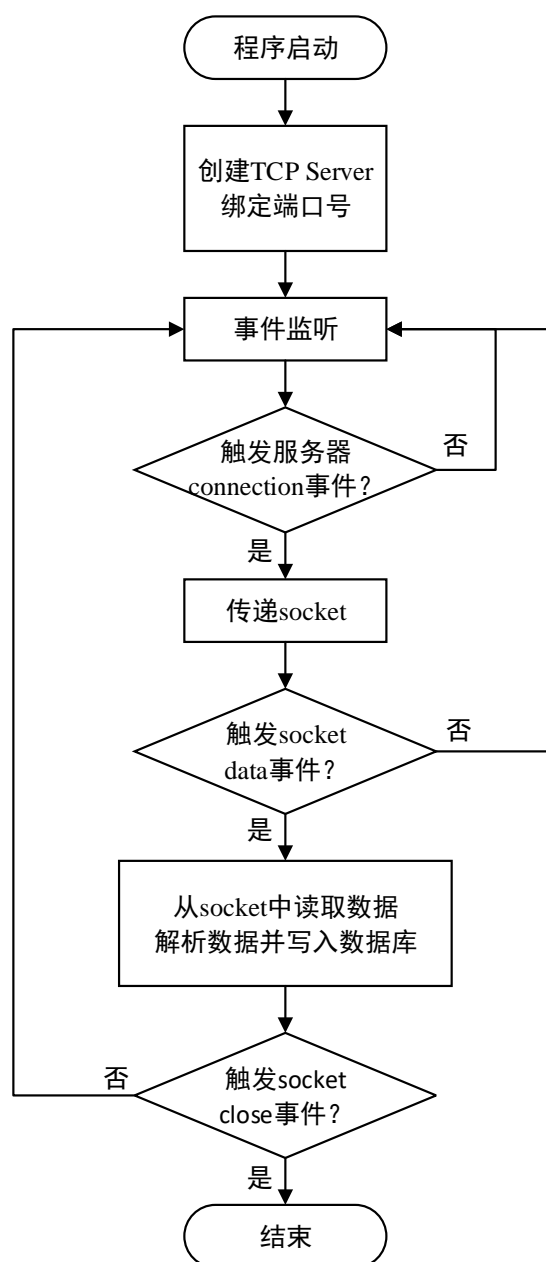


图 4.9 通信服务器接收数据流程图

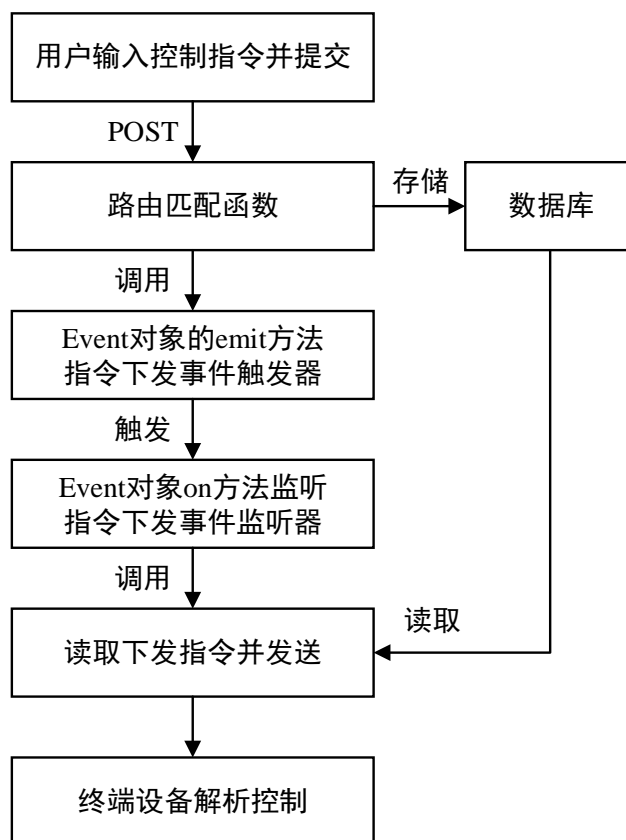


图 4.10 远程控制指令下发流程

远程控制指令下发流程如图 4.10 所示。功能具体步骤如下（1）首先引入 `events` 模块创建对象，注册指令下发事件监听器。

```
var events = require('events');
var em = new events.EventEmitter();
```

（2）指令下发页面将用户输入指令通过 `post` 请求传至路由匹配函数处理，在路由匹配函数中将指令存储至数据库后触发指令下发事件。

```
router.post("/send", function(req,res,next){
    .....
    sendCmd.save(function(err){
        if(err){
            console.log(err);
        }else{
            em.emit('dataSend');
            res.redirect('/sendcmd');
        }
    })
})
```

```
})
```

(3) 在监控平台与设备终端建立连接后, 指令下发事件监听器在其回调函数中读取数据库中指令并调用发送函数完成控制指令下发的操作, 并对数据库中存放指令的文档进行更新, 增加发送时间, 发送是否完成等信息。

```
em.on('dataSend',function(){  
    .....  
    socket.write(sendDataString);  
    model.connect(function(db){db.collection('sendcmd').updateOne(.....)});  
});
```

4.4.3 Web 服务器设计

Web 服务器, 负责向应用层服务提供数据接口, 实现用户管理、数据显示、指令下发等功能页面。express 是 Node.js 中一个简洁且灵活的 Web 应用框架, 基于 Node.js 内置的 http 模块^[56]。监控平台软件的文件架构是基于 express 框架的, express 使用的大致步骤如下:

(1) 创建项目, 项目目录结构如图 4.11, 其中 model 文件夹表示数据, 代码文件用于实现数据库相关的业务逻辑。public 文件夹下存储页面所使用到的所有静态文件, 包括图片、样式和脚本等。routes 文件夹下存放监控平台路由相关文件。views 文件夹下放置页面模板文件, 模板文件通过模板引擎渲染, package.json 文件是整个项目依赖配置, 记录了项目中所使用的模块和依赖。app.js 是项目入口, 也是应用核心配置文件, 其中的代码包括创建通信服务器、使用 express 框架、配置路由等。

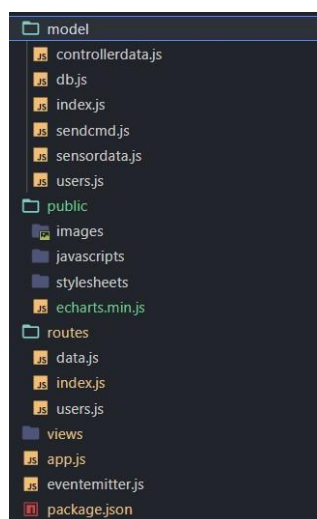


图 4.11 项目目录结构

(2) 安装并引入 express, 使用 npm 工具安装即可。var express = require('express')。

(3) 创建应用对象 `var app = express()`，启动服务器 `express.listen(3000)`，配置静态资源 `app.use(express.static("public"))`，设置路由。

`express` 框架中最主要的是路由和中间件。路由是由一个 URI（路径）和一个特定的 HTTP 方法（GET、POST 等）组成，用于响应客户端对网站节点的访问。每一个路由都可以有一个或多个处理函数。中间件是在匹配路由之前或在匹配路由完成所做的一系列操作。使用 `app.use()` 函数使用中间件，下文中使用的 `session` 和 `cookie-parser` 就属于中间件的一种。

在 HTTP 请求中，可以提取出请求的 URL 以及 GET/POST 参数^[57]。路由匹配函数有三个参数 `request (res)`、`response (req)` 和 `next()`。`request` 对象表示 HTTP 请求，常见的属性有：`req.query` 用于获取 GET 参数，`req.body` 获取 POST 请求参数，`req.ip` 属性用于获得 HTTP 请求的 IP 地址。`response` 对象表示 HTTP 响应，常见的方法有：`res.redirect()` 用于网址重定向，`res.render()` 方法用于渲染网页模板。监控平台用户管理和数据显示相关路由及说明如表 4.6 所示。

表 4.6 监控平台用户管理和数据显示相关路由

URL	http 方法	是否需要登录	说明
/login	get	否	用户登录
/users/login	post	是	登录验证
/regist	get	否	用户注册
/users/regist	get	是	新建用户
/delete	get	是	删除一条传感器数据
/sendcmd	get	是	控制指令下发
/sensordata	get	是	传感器数据显示

监控平台通过登录验证实现用户管理。因为 HTTP 是无状态协议，所以每一次访问页面都是无关的。基于 `session` 认证的流程：用户在页面中输入账号和密码，向 Web 服务器发送请求。服务器接收到请求后，对账号和密码进行验证。如果验证成功的话，会生成一个随机的 `token`，保存并发送给浏览器。浏览器将 `token` 保存在 `cookie` 中，在接下来的每个请求都会携带 `cookie` 信息。服务器接到带有 `cookie` 的请求，解析 `cookie` 确定用户的身份。`session` 的配置如下：

```
app.use(session({
  secret: 'iot',
  resave: false,
  saveUninitialized: true,
  cookie: { maxAge: 1000 * 60 * 5 }
}))
```


session 配置相关参数的解释如下表 4.7。

表 4.7 session 配置相关参数

参数	作用
secret	一个 String 类型的字符串，作为服务器端生成 session 的签名
resave	强制保存 session 即使它并没有变化
saveUninitialized	强制将未初始化的 session 存储
cookie	设置返回到前端 key 的属性。maxAge: cookie 的超时时间，表示当前时间（Date.now()）之后的毫秒数，即指定登录会话的有效时间，这里设置为 5 分钟

基于 express 框架搭建 web 服务器应用，服务器响应浏览器请求的具体流程如图 4.12 所示。首先接收浏览器请求，如果请求合法，根据请求匹配路由转到对应的路由处理函数进行事务处理，包括调用数据库查询等操作。事务处理完成，组合请求结果，将其渲染到浏览器页面。

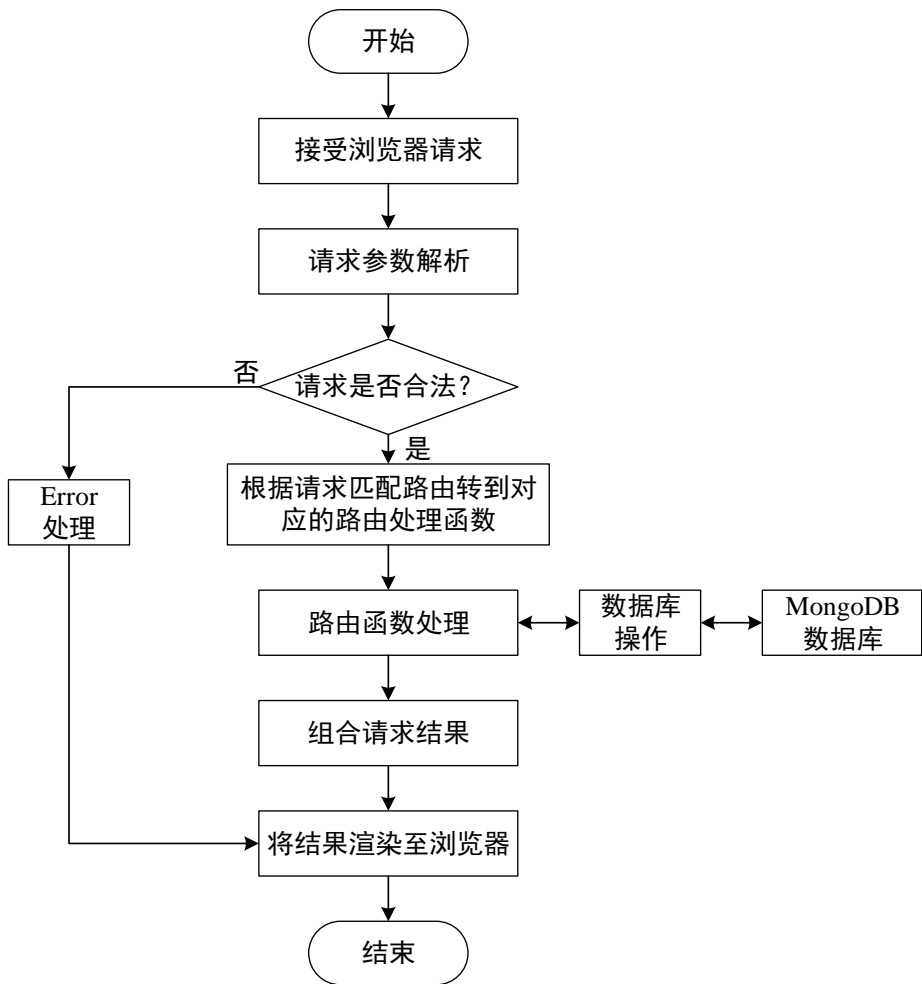


图 4.12 Web 服务器响应浏览器请求的流程

监控平台页面采用前端相关技术开发，包括用户登录页面、数据显示页面、指令下发页面等，在项目 views 文件夹下存放。使用模板引擎来实现监控平台页面显示。模板引擎是用来解析对应类型模板文件，然后动态生成由数据和静态页面组成的视图文件的工具。在模板文件中通过标签（tag）来响应各种解析动作，动态的将对应数据展示到指定位。express 框架

支持多个版本的模板引擎，在本应用中，使用的模板引擎是 `ejs`。`ejs` 是一个开源的 `JavaScript` 模板库或工具，可以将数据和模板整合最终生成 `html` 文件。`ejs` 与上述使用的模块类似，通过 `npm` 安装，然后在文件中引入，通过下述代码设置模板引擎为 `ejs`，并设置模板文件的位置。

```
app.set('views', path.join(__dirname, 'views'));
```

```
app.set('view engine', 'ejs');
```

路由匹配函数中通过 `res.render(view)` 方法渲染网页模板，如传感器数据显示页面，其路由匹配函数中会同步读取数据库中存放传感器数据的集合，将数据渲染到页面中预先放置占位符的展示位置。

4.5 本章小结.

本章节结合系统需求与功能指标，对系统的软件设计进行了详尽的介绍。首先介绍了消毒机设备终端嵌入式软件设计和监控平台软件设计的开发环境。然后从软件初始化、传感器数据采集、交互模块软件设计、通信模块软件设计对消毒机设备终端嵌入式软件设计进行了阐述。对监控平台与消毒机设备终端使用的数据交换格式进行了概述。最后描述了监控平台软件设计，包括数据库的设计、通信服务器设计、**Web** 服务器软件开发和相关功能的实现。

第五章 系统测试

本文设计了一种基于物联网的消毒监控系统，包括消毒机设备终端和监控平台，为了确保两部分是否能够正常稳定运行并实现相应的功能，本章对系统进行相应的功能性验证和通信相关测试。

实验整体框架如图 5.1 所示。向上的数据通道：传感器采集的室内环境数据信息统一由 STM32F103VET6 封装传至 NB-IoT 通信模块，再通过移动通信网络，发送到部署在阿里云服务器上的监控平台，在监控平台 TCP 服务器中处理数据，存储至 MongoDB 数据库。Web 服务器从数据库中读取数据，渲染至 Web 应用程序的前端页面。向下的数据通道：用户在 Web 应用程序中发送远程控制指令，由 HTTP 协议传递至 Web 服务器，Web 服务器将远程控制指令存储至数据库中，触发指令下发事件，由 TCP 服务器向指定的消毒机设备终端的 NB-IoT 通信模块发送，微控制器接收到远程控制指令后，进行解析并控制相应的执行器工作。

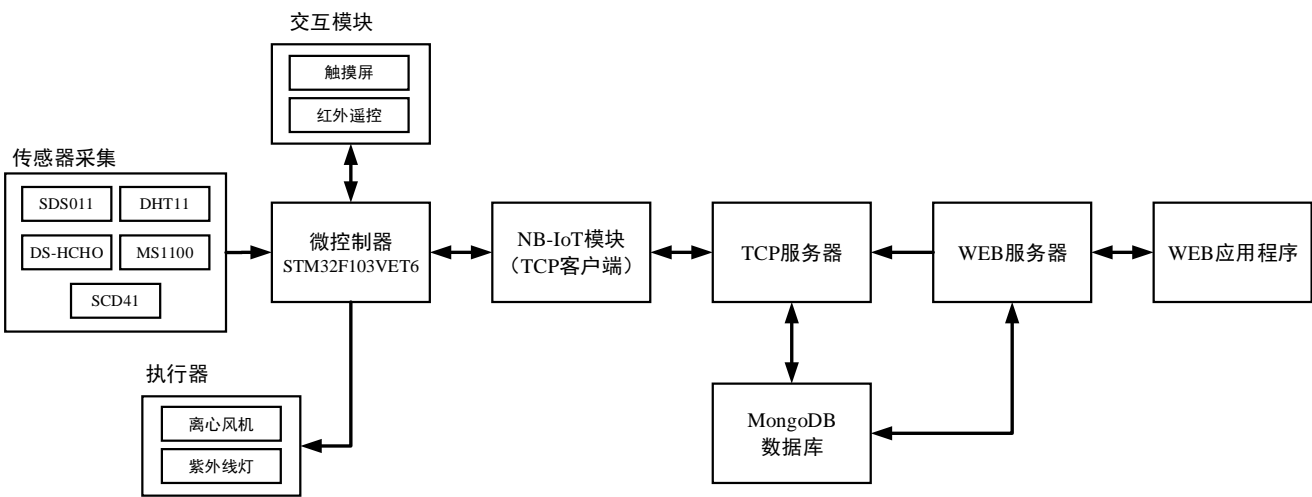


图 5.1 实验整体框架

5.1 基于监控平台的功能验证

5.1.1 系统部署

基于监控系统的功能验证是对监控平台相关功能的验证，包括用户登录、数据存储、数据显示、多设备终端连接管理等功能，。

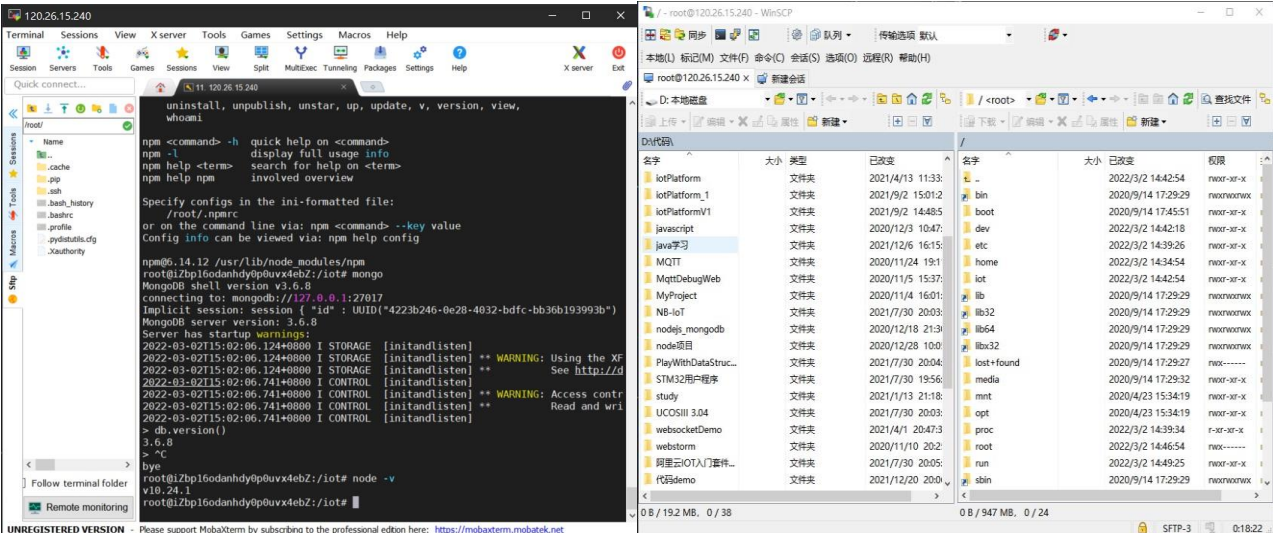
监控平台选用阿里云 ESC 服务器作为部署环境，服务器内存 2GB，系统盘 60GB，运行 Linux64 位 Ubuntu 20.04 64 位操作系统，公网 IP 地址为 120.26.15.240。部署步骤如下：

- (1) 在阿里云服务器安全配置创建防火墙规则，开启 TCP 连接、HTTP 连接的相应端口。
- (2) 使用 MobaXterm 软件作为阿里云服务器的远程操作台，如图 5.3 (a) 所示。在阿里云服务器上安装 node 运行环境和 MongoDB 数据库，使用 apt (Advanced Packaging Tool) 软件包管理器即可。如图 5.2 所示，安装的 node 版本为 10.24.1，npm 版本为 6.14.12，MongoDB 数据库的版本为 3.6.8。安装好 MongoDB 后新建数据库及相关集合。

```
root@iZbp16odanhdy0p0uvx4ebZ:~# node -v
v10.24.1
root@iZbp16odanhdy0p0uvx4ebZ:~# npm -v
6.14.12
root@iZbp16odanhdy0p0uvx4ebZ:~# mongo
MongoDB shell version v3.6.8
connecting to: mongodb://127.0.0.1:27017
Implicit session: session { "id" : UUID("dd6f80bd-c913-4408-8292-b7e17e31b10a") }
MongoDB server version: 3.6.8
```

图 5.2 node、npm 和 MongoDB 版本

- (3) 如图 5.3 (b)，WinSCP 软件用于本地与阿里云服务器间安全地传输文件，将编写的监控平台程序传输到阿里云服务器。



(a) MobaXterm 界面

(b) WinSCP 软件界面

图 5.3 MobaXterm 和 WinSCP 软件界面

- (4) 使用 npm 包管理工具，安装相关 modules 后运行程序，运行成功后会打印出相关信息，如图 5.4。

```
root@iZbp16odanhdy0p0uvx4ebZ:~/iotPlatform_1# npm start
> iotplatform@0.0.0 start /root/iotPlatform_1
> nodemon ./bin/www

[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./bin/www`
TCP Server Listening port 3307
数据库连接成功mongoose
```

图 5.4 监控平台启动运行

5.1.2 监控平台功能验证

监控平台实现的功能主要是数据通信、数据存储、数据显示、用户管理、消毒机设备终端管理等。在浏览器输入 IP 地址和端口号，进入用户登录页面。用户名和密码输入框会进行相关正则表达式的匹配，不符合输入规范，会提示重新输入。点击登录按钮时发起登录请求，在后端登录接口进行验证。如果用户名或密码错误，返回登录页面，用户名和密码验证通过，进入主页。登录功能测试正常，登录页面如图 5.5。



图 5.5 登录页面

用户在登录页面验证用户名和密码成功后，进入主页，主页为数据显示页面，页面对传感器数据进行分页显示。传感器数据按接收时间进行排列显示，数据显示功能正常，数据显示页面如图 5.6。

序号	CO2	CHCO	TVOC	PM2.5	PM10	温度	湿度	IP地址	接收时间	操作者
1	454	1	2	34	30	26	10	10.163.155.12	2021-05-18 10:27:03	admin 删除 更改
2	457	1	2	33	30	26	10	10.163.155.12	2021-05-18 10:26:55	admin 删除 更改
3	451	1	2	32	38	26	10	10.163.155.12	2021-05-18 10:26:32	admin 删除 更改
4	452	1	2	34	30	26	10	10.163.155.12	2021-05-18 10:26:17	admin 删除 更改
5	451	1	2	34	31	26	10	10.163.155.12	2021-05-18 10:26:08	admin 删除 更改
6	458	1	2	32	31	26	10	10.163.155.12	2021-05-18 10:25:28	admin 删除 更改
7	460	1	2	31	32	26	10	10.163.155.12	2021-05-18 10:24:59	admin 删除 更改
8	449	1	2	36	32	26	10	10.163.155.12	2021-05-18 10:24:34	admin 删除 更改
9	458	1	2	35	35	26	10	10.163.155.12	2021-05-18 10:24:07	admin 删除 更改
10	456	1	2	34	30	26	10	10.163.155.12	2021-05-18 10:22:38	admin 删除 更改

图 5.6 数据显示页面

主页中有导航栏，点击指令下发，进入指令下发页面，在指令下发页面可编辑远程控制指令，包括选择紫外线灯的开启或关闭，离心风机风速，还预留了额外的控制开关，消毒机设备终端可接额外的执行设备。在输入框中控制指令后，在数据库中能查询到，指令下发页

面如图 5.7。



图 5.7 指令下发页面

数据存储包括用户数据、传感器数据、控制指令数据、设备终端数据等，在监控平台页面中显示的传感器数据归登录用户所有，例如数据显示页面中只显示属于该用户的消毒机设备终端的传感器数据。TCP 服务器在接收到传感器数据时，会进行解析处理，添加 TCP 客户端 IP 地址、接收到数据的时间和设备终端管理者信息(默认 admin)后，存储到数据库 sensordata 集合中。数据库 sensordata 集合中一条文档（数据）如下所示，包括了传感器的数据、TCP 客户端的 IP 地址及接收到的时间，以及设备所有者信息。

```
{ "_id" : ObjectId("60a324eea04365031c2b582c"), "CO2" : "456", "CH2O" : "1", "TVOC" : "2", "PM25" : "34", "PM10" : "30", "temperature" : "26", "humidty" : "10", "tcpClientIp" : "10.163.155.12", "receivedTime" : 1621304558382, "owner" : "admin" }
```

数据库 users 集合中的一条文档（数据）如下所示，文档信息为监控平台用户的登录名和密码。用户在登录时，会在路由匹配函数中查询此集合中是否存在输入的登录用户信息，验证登录名和密码是否匹配。

```
{ "_id" : ObjectId("605b47a10e554ecc34d32212"), "username" : "admin", "password" : "123456" }
```

用户在指令下发页面完成控制指令的设置后，指令会被存储到数据库 sendcmd 集合中。sendcmd 集合的一条文档（数据）如下所示，文档是保存用户设置的远程控制指令，包括了设置紫外线开关、控制风机风速大小、要发送的消毒机设备终端 IP 地址、指令发送时间、是否发送完成的标志位等信息。

```
{ "_id" : ObjectId("606ef57fab32de42c0b6432b"), "plasmaStatus" : true, "uvlStatus" : true, "speed" : "2", "tcpClientIp" : "10.163.168.159", "sendTime" : 1617884543100, "flag" : "complete", "__v" : 0 }
```

数据库中存储的数据如图 5.8 所示，包括以上提到的传感器数据、用户数据、控制指令

数据和设备终端数据等。

```
> use iot
switched to db iot
> show collections
controllerdata
sendcmd
sensordata
users
> db.sensordata.find()
{"_id": ObjectId("60a324eea04365031c2b582c"), "CO2": "456", "CH2O": "1", "TVOC": "2", "PM25": "34", "PM10": "30", "temperature": "26", "humidity": "10", "tcpClientIp": "10.163.155.12", "receivedTime": 1621304558392, "owner": "admin"}, {"_id": ObjectId("60a32547a04365031c2b582d"), "CO2": "458", "CH2O": "1", "TVOC": "2", "PM25": "35", "PM10": "35", "temperature": "26", "humidity": "10", "tcpClientIp": "10.163.155.12", "receivedTime": 1621304647381, "owner": "admin"}, {"_id": ObjectId("60a32562a04365031c2b582e"), "CO2": "449", "CH2O": "1", "TVOC": "2", "PM25": "36", "PM10": "32", "temperature": "26", "humidity": "10", "tcpClientIp": "10.163.155.12", "receivedTime": 1621304674193, "owner": "admin"}, {"_id": ObjectId("60a3257ba04365031c2b582f"), "CO2": "460", "CH2O": "1", "TVOC": "2", "PM25": "31", "PM10": "32", "temperature": "26", "humidity": "10", "tcpClientIp": "10.163.155.12", "receivedTime": 1621304699951, "owner": "admin"}, {"_id": ObjectId("60a32598a04365031c2b5830"), "CO2": "458", "CH2O": "1", "TVOC": "2", "PM25": "32", "PM10": "31", "temperature": "26", "humidity": "10", "tcpClientIp": "10.163.155.12", "receivedTime": 1621304723130, "owner": "admin"}, {"_id": ObjectId("60a325c0a04365031c2b5831"), "CO2": "451", "CH2O": "1", "TVOC": "2", "PM25": "34", "PM10": "31", "temperature": "26", "humidity": "10", "tcpClientIp": "10.163.155.12", "receivedTime": 1621304768766, "owner": "admin"}, {"_id": ObjectId("60a325c9a04365031c2b5832"), "CO2": "452", "CH2O": "1", "TVOC": "2", "PM25": "34", "PM10": "30", "temperature": "26", "humidity": "10", "tcpClientIp": "10.163.155.12", "receivedTime": 1621304777962, "owner": "admin"}, {"_id": ObjectId("60a325d8a04365031c2b5833"), "CO2": "451", "CH2O": "1", "TVOC": "2", "PM25": "32", "PM10": "38", "temperature": "26", "humidity": "10", "tcpClientIp": "10.163.155.12", "receivedTime": 1621304792907, "owner": "admin"}, {"_id": ObjectId("60a325efa04365031c2b5834"), "CO2": "457", "CH2O": "1", "TVOC": "2", "PM25": "33", "PM10": "30", "temperature": "26", "humidity": "10", "tcpClientIp": "10.163.155.12", "receivedTime": 1621304815125, "owner": "admin"}, {"_id": ObjectId("60a325f7a04365031c2b5835"), "CO2": "454", "CH2O": "1", "TVOC": "2", "PM25": "34", "PM10": "30", "temperature": "26", "humidity": "10", "tcpClientIp": "10.163.155.12", "receivedTime": 1621304823491, "owner": "admin"}
```

图 5.8 数据库查询

监控平台支持多个消毒机设备终端的连接和管理。通过 Socket 调试工具创建两个 TCP 客户端和消毒机设备终端同时连接监控平台。图 5.9 为 Socket 调试工具界面。测试结果如图 5.10，监控平台成功与三个客户端建立连接并打印出了设备终端的 IP 地址。当三个客户端同时连接时，监控平台可以下发远程控制指令到指定客户端。



图 5.9 socket 调试工具

```
> iotplatform@0.0.0 start /root/iotPlatform_1
> nodemon ./bin/www

[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./bin/www`
TCP Server Listening port 3307
数据库连接成功mongoose
IP::ffff:36.152.116.90 connect
IP::ffff:36.152.115.155 connect
IP::ffff:117.136.45.27 connect
```

图 5.10 多个设备终端连接

5.2 控制功能验证

消毒机设备终端硬件系统如图 5.11 所示。

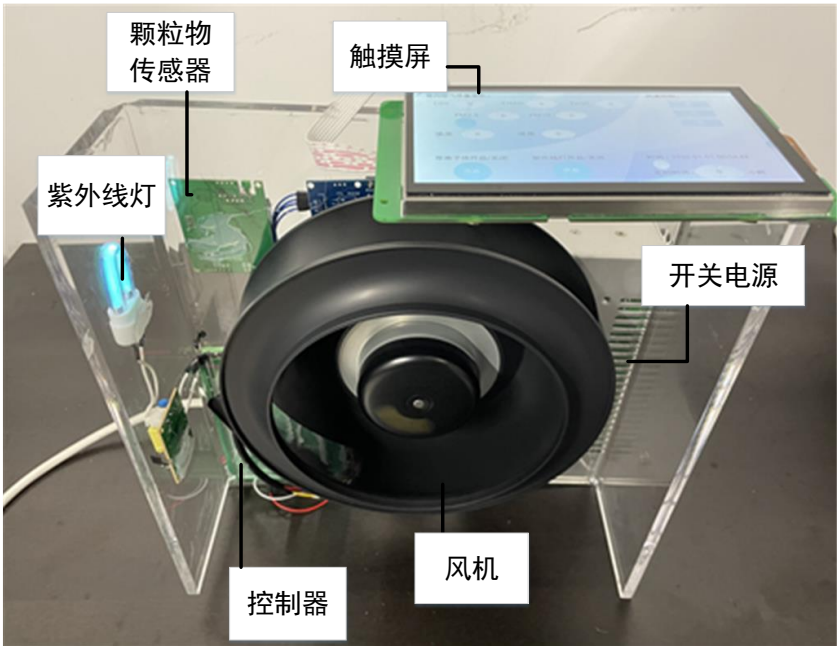
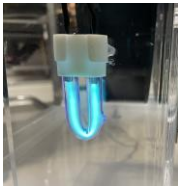
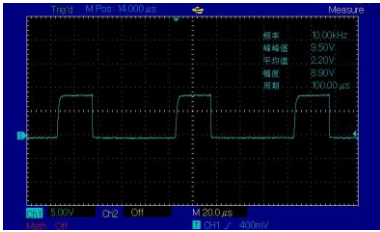


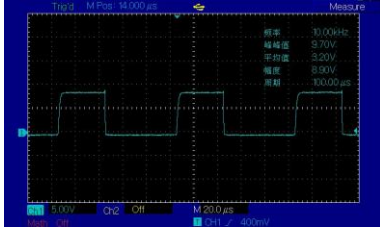

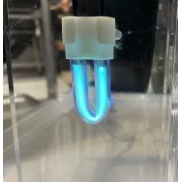
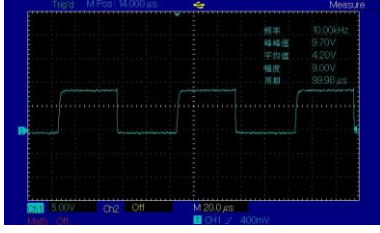



图 5.11 硬件系统实物图

监控平台与消毒机设备终端建立连接后，用户通过浏览器访问监控平台，通过下发不同的远程控制指令控制消毒机设备终端工作。表 5.1 显示的是在不同的远程控制下，紫外线灯和风机的运行情况。

表 5.1 远程控制功能验证

远程控制指令	紫外线灯情况	示波器测量 PWM 波形	离心风机转动情况
<code>{"uvlStatus": true, "speed": "1"}</code>			
<code>{"uvlStatus": false, "speed": "2"}</code>			
<code>{"uvlStatus": true, "speed": "3"}</code>			

消毒机设备终端还可以使用触摸屏和红外遥控器作为控制方式，得到的测试结果验证这两种方法也可以控制执行器工作。

5.3 系统稳定性与延时测试

由于消毒机设备终端微控制器处理速度、程序设计等原因，使接收到的控制指令数据得不到及时处理，或者由于网络传输等原因，最终出现数据丢失的情况。现进行相关测试，判断系统稳定性。测试环境：在楼高 5 层的建筑物室内进行测试；通信设备使用贴片式天线。测试方法如下：在设备终端处于随机的网络环境下，由监控平台下发指令，通过设备终端实际接收指令数据个数与理论接收指令数据个数的比值来判定数据接收的成功率。测试采用不同的发送时间间隔为分组，每组监控平台下发 1000 个指令数据，指令数据长度在 51 字节左右。每组分别在建筑物室内 1 楼、4 楼、5 楼重复测试三次，统计数据的接收成功率。指令数据接收成功率测试结果如表 5.2 所示。

表 5.2 远程控制指令接收成功率

发送间隔/ms	接收成功率/%
500	93.93
600	96.23
700	98.20

监控平台发送指令数据间隔在 500ms 以上，设备终端接收指令数据成功率均大于 90%以上，在 700ms 间隔以上，指令数据接收成功率接近 100%，表明在数据下发通道具有良好的稳定性，满足系统设计要求。

延时测试包括双向数据通道的测试，来表征数据的实时性。向上数据通道延时测试是通过微控制器接收到传感器数据的时间与数据存储到监控平台数据库的时间差，来判断传感器数据的实时性。用户传感器数据包的长度在 100 字节左右，设备终端发送 100 次传感器数据，比较时间差均值。向下数据通道延时测试通过监控平台 Web 服务器后台接收到用户输入指令到设备终端执行动作的时间差，来衡量控制指令的实时性，同样进行 100 次测试。在不同网络环境的情况下，测试结果如表 5.3 所示。

表 5.3 延时测试

数据通道	延时/ms
向上通道	1423.85
向下通道	2461.14

微控制器接收到完整传感器数据的时间到数据存储至数据库的时间差为 1424.85ms。监控平台后台接收到页面输入指令到消毒机设备终端完成执行动作的时间差为 2461.14ms，达到系统的设计要求。

5.4 本章小结

本章对消毒监控系统进行了相应的功能性验证和通信相关测试。功能性验证包括监控平台的用户管理, 数据显示、数据存储功能和远程控制功能, 以及设备终端的控制功能进行了验证。通信相关测试包括远程控制指令接收成功率和数据通道延时测试。测试结果表明, 系统设计完成了预期的功能需求。

第六章 总结与展望

6.1 总结

随着物联网技术的不断发展,在相关领域的应用研究也相继展开。在医疗领域中,物联网技术的应用前景巨大。智慧医疗应用就是在现有医疗设备、医疗服务的基础上利用物联网技术进行信息化升级。智慧医疗所倡导的整个大的体系系统是由一个个小的、耦合度不高的智慧医疗子系统构成,如智慧病房、药品管理、远程监护、医疗设备的跟踪与管理、病房智能陪护设施、静脉输液监测、消毒监控系统、医疗废弃物管理等。针对医院消毒领域传统循环风紫外线消毒机信息化程度较低,本课题在广泛参阅了相关技术及参考文献的基础上,开发设计了一种基于物联网的消毒监控系统。本文在系统的相关设计研究与系统设计上完成了以下工作:

(1) 阐述了智慧医疗、空气消毒相关的研究背景,延伸出本课题的研究意义。对物联网在智慧医疗子系统下的应用现状进行了分析。给出了系统功能需求,确定了功能内容。对本文中所涉及到的物联网通信技术进行概述,重点介绍了在设计中所使用到的关键技术,包括 NB-IoT、Node.js、MongoDB 数据库。对 NB-IoT 实现广覆盖、海量连接、低功耗所使用的技术手段和 Node.js 事件驱动机制以及 MongoDB 数据库应用在物联网场景下的优势进行了详细分析。

(2) 基于物联网的三层架构对整个消毒监控系统进行设计,包括消毒机设备终端和监控平台两个部分。数据通道包括向上通道和向下通道,设备终端采集传感器数据,通过向上数据通道传至监控平台;监控平台通过向下通道推送指令到设备端,控制消毒机工作。

(3) 完成消毒机设备终端的硬件设计。消毒机设备终端在实现传统循环风紫外线消毒机的基础上,增加室内环境数据监测、远程控制等功能。硬件设计包括微控制器、传感器模块、NB-IoT 模块、终端交互模块、紫外线灯、离心风机的选型,接口电路、电源电路和 PCB 的设计。

(4) 完成消毒机设备终端的软件设计和监控平台的软件设计。介绍了消毒机设备终端嵌入式软件和监控平台软件的开发环境。从初始化、传感器模块、设备终端交互模块、通信模块介绍了本系统嵌入式软件设计方法。对比了相关数据交换格式,优选出 JSON 作为消毒机设备终端和监控平台的数据交换格式。从数据库设计、通信服务器设计、Web 服务器设计三个方面,介绍了监控平台软件设计方法。

(5) 通过系统测试,对整个系统进行功能性验证和通信相关测试。功能性测试包括监控平台

用户管理、数据存储、数据显示、多设备连接管理和控制功能验证。通信测试通过远程控制指令接收成功率和双向数据通道的延时，来表征系统性能。测试结果给出了系统的功能展示和通信测试数据，验证了系统的可行性。

6.2 展望

本文参照了物联网应用相关参考文献和技术，设计了一种基于物联网的智慧医疗消毒监控系统，实现了准确、实时地室内环境监测、远程控制消毒机设备终端工作、消毒机设备终端交互、监控平台多设备终端管理、数据管理、用户管理等功能，并通过相关测试，验证了系统的可行性。但是由于时间和个人能力的有限，系统还存在一些不足之处，还需要进一步的优化才能投入到实际使用中，具体有以下几个方面：

- （1）系统只完成了相关环境数据的监测，没有对消毒效果的监测，无法反映准确的消毒情况。消毒机设备终端还可以增加除紫外线外的消毒模块，增强消毒效率。
- （2）消毒机设备终端与监控平台之间数据交互单一，虽然完成了双向的数据通信，但是上传通道只上传传感器数据，下发通道只下发远程控制指令。在后续的优化中，增加包括设备状态等数据的传输。
- （3）监控平台只实现了所需的基础功能，界面也不够美观，在后续还需要增加更多的功能，包括数据可视化，环境数据分析等。
- （4）系统在投入实际应用前，还需要进行更多的功能测试和稳定性测试。

参考文献

- [1] 苗凤娟, 惠鹏飞, 孙艳梅. 物联网技术导论[M]. 哈尔滨工程大学出版社, 2013.
- [2] 李新献, 田肖. 物联网技术在医疗行业中的应用[J]. 电子世界, 2017(6):1.
- [3] 何遥. 智慧医疗的新发展[J]. 中国公共安全, 2018(10):5.
- [4] 米丽娟. 医疗机构空气消毒净化处理技术进展[J]. 职业与健康, 2011, 27(2):3.
- [5] 中华人民共和国卫生部. 医院空气净化管理规范: WS/T368—2012[S].
- [6] 谢冰. 循环风紫外线消毒器手术室空气动态消毒的效果分析[J]. 工程技术研究, 2020, 5(8):2.
- [7] Xu Z , He J , Chen Z . Design and actualization of IoT-based intelligent logistics system[C]. IEEE International Conference on Industrial Engineering & Engineering Management. IEEE, 2014.
- [8] Shieh H L, Lin S F, Chang W S. RFID medicine management system[C]. International Conference on Machine Learning & Cybernetics. IEEE, 2012.
- [9] Suzuki T, Oyama Y, Nakauchi Y. Intelligent medicine case system with distributed RFID readers[J]. Conf Proc IEEE Eng Med Biol Soc, 2010, 2010(10):344-347.
- [10] Al-Khafajiy M, Baker T, Chalmers C, et al. Remote health monitoring of elderly through wearable sensors[J]. Multimedia Tools and Applications, 2019, 78(17):1-26.
- [11] 邓燕楠. 窄带物联网在智慧医疗中的应用研究[D]. 沈阳理工大学, 2020.
- [12] Zhang H, Li J, Wen B, et al. Connecting Intelligent Things in Smart Hospitals Using NB-IoT[J]. IEEE Internet of Things Journal, 2018.
- [13] 魏文博. 基于LoRa技术的智慧医疗输液监测系统的设计与实现[D]. 西安电子科技大学, 2020.
- [14] 苗荣霞, 陈忠孝, 戴宝华, 等. 基于单片机的空气消毒机控制系统[J]. 现代电子技术, 2008, 31(16):3.
- [15] Hung L, Chen K, Hsieh, N, et al. Using Internet of Things to Improve the Sterilization Process for Surgical Instruments in Healthcare[J]. Journal of Internet Technology, 2020, 21(6).
- [16] 牟强善. 基于物联网技术医用空气消毒参数监测系统[D]. 新乡医学院, 2018.
- [17] Yang L, Yao T, Liu G, et al. Monitoring and Control of Medical Air Disinfection Parameters of Nosocomial Infection System Based on Internet of Things[J]. Journal of Medical Systems, 2019, 43(5).
- [18] Gerson, Roberto, Luqueta, et al. Wireless Sensor Network to Monitoring an Ozone Sterilizer[J]. IEEE Latin America Transactions, 2016, 14(5):2167-2174.
- [19] Zhao Y L , Huang H P , Chen T L , et al. A Smart Sterilization Robot System with Chlorine Dioxide for Spray Disinfection[J]. IEEE Sensors Journal, 2021, PP(99):1-1.
- [20] 王宜怀, 张建, 刘辉, 等. 窄带物联网NB-IoT应用开发共性技术[M]. 北京: 电子工业出版社, 2019.
- [21] Sheth J, Dezfouli B. Enhancing the Energy-Efficiency and Timeliness of IoT Communication in WiFi Networks[J]. IEEE Internet of Things Journal, 2019.
- [22] Rashmi, Sharan, Sinha, et al. A survey on LPWA technology: LoRa and NB-IoT[J]. ICT Express, 2017.
- [23] 陆睿. 基于ZigBee的智能家居控制系统研究[D]. 南京邮电大学, 2017.
- [24] 张瑞增. 基于智能车位锁的共享停车位管理系统研究与设计[D]. 山东大学, 2017.
- [25] Mekki K, Bajic E, Chaxel F, et al. Overview of cellular LPWAN technologies for IoT deployment: Sigfox, LoRaWAN, and NB-IoT[C]. IEEE International Workshop on Mobile & Pervasive Internet of Things. IEEE, 2018.
- [26] Beyene Y D, Jantti R, Tirkkonen O, et al. NB-IoT Technology Overview and Experience from Cloud-RAN Implementation[J]. IEEE Wireless Communications, 2017, 24(3):26-32.
- [27] 杨观止, 陈鹏飞, 崔新凯, 等. NB-IoT综述及性能测试[J]. 计算机工程, 2020, 46(1):14.
- [28] 黄宇红, 杨光, 曹蕾, 等. NB-IoT技术解析与案例详解[M]. 北京: 机械工业出版社, 2018.
- [29] Kang Y S, Park I H, Rhee J, et al. MongoDB-based Repository Design for IoT-generated RFID/Sensor Big

- Data[J]. IEEE Sensors Journal, 2015, 16(2):1-1.
- [30] 张正颢. 基于云平台的IoT数据监控系统的设计与实现[D]. 电子科技大学, 2020.
- [31] Tilkov, Stefan, Vinoski, et al. Node.js: Using JavaScript to Build High-Performance Network Programs[J]. IEEE Internet Computing, 2010.
- [32] 卢培鹏. 基于Web技术的物联网数据云平台的设计与实现[D]. 哈尔滨工程大学, 2016.
- [33] C Györödi, R Györödi, Pecherle G, et al. A comparative study: MongoDB vs. MySQL[C]. 2015 13th International Conference on Engineering of Modern Electric Systems (EMES). IEEE, 2015.
- [34] Rautmare S, Bhalerao D M. MySQL and NoSQL database comparison for IoT application[C]. IEEE International Conference on Advances in Computer Applications. IEEE, 2016.
- [35] 毛文琪, 瞿少成, 赵亮, 等. 基于物联网的教室模糊AQI监测管理系统[J]. 电子测量技术, 2021, 44(3):5.
- [36] 华枝发, 张兰, 岳显昌, 等. 高频地波雷达无线数据传输系统设计[J]. 电子测量与仪器学报, 2021, 35(1):8.
- [37] 王灿. 基于云平台的物联网远程监控系统研究[D]. 华中科技大学, 2015.
- [38] 赵全, 徐光, 郝龙, 等. 基于LoRa的无线多参数环境监测系统设计[J]. 国外电子测量技术, 2019, 38(06):120-124.
- [39] 郑亮, 郑士海. 嵌入式系统开发与实践: 基于STM32F10x系列[M]. 北京航空航天大学出版社, 2015.
- [40] 刘火良. STM32库开发实战指南[M]. 机械工业出版社, 2013.
- [41] 李艳霞. 用于故障预警和诊断的滚动轴承检测仪设计[D]. 山东师范大学, 2020.
- [42] 王子权. 基于STM32的PWM调光器[J]. 电气自动化, 2018, 40(06):98-100.
- [43] 王登宏, 厉佳男, 贺雪辉, 等. 基于GPRS的无线室内空气质量监测系统的设计[J]. 工业控制计算机, 2017, 30(7):3.
- [44] 肖健超, 苏晋升, 郭格, 等. 基于车载WSN网络的车载环境监测系统设计[J]. 电子测量技术, 2020, 43(13):8.
- [45] 宋高峰. 基于ARM单片机的空气质量检测系统研究[D]. 吉林大学, 2018.
- [46] 党凯强, 姚金杰, 贺冠华, 等. 一种低功耗NB-IoT远程监测终端设计[J]. 国外电子测量技术, 2020, 39(12):136-140.
- [47] Nayyar A, Puri V. An Encyclopedia Coverage of Compiler's, Programmer's & Simulator's for 8051, PIC, AVR, ARM, Arduino Embedded Technologies[J]. International Journal of Reconfigurable and Embedded Systems (IJRES), 2016, 5(1).
- [48] 李进才. 5款优秀的前端开发工具[J]. 计算机与网络, 2020, 46(13):1.
- [49] Jean J. Labrosse. 嵌入式实时操作系统μC/OS-II[M]. 北京航空航天大学出版社, 2012.
- [50] 张雪松, 李驹光, 聂诗良, 等. 基于Cortex-M3处理器的.Net Micro Framework移植[J]. 微计算机信息, 2012(6):3.
- [51] Lin Y B, Huang C M, Chen L K, et al. MorSocket: An Expandable IoT-based Smart Socket System[J]. IEEE Access, 2018, PP(99):1-1.
- [52] 黄明亮. 物联网开放平台的研究与设计[D]. 中国海洋大学, 2013.
- [53] 赵宏林, 廉小亲, 郝宝智, 等. 基于物联网云平台的空调远程控制系统[J]. 计算机工程与设计, 2017, 38(01):265-270.
- [54] MDN contributors. Express 教程3: 使用数据库(Mongoose)[EB/OL]. (2022-03-13)[2022-03-14]. https://developer.mozilla.org/zh-CN/docs/learn/Server-side/Express_Nodejs/mongoose.
- [55] 陈瑶. 基于Node.js高并发web系统的研究与应用[D]. 电子科技大学, 2014.
- [56] Caset F, Vale D S, Viana C M. Measuring the Accessibility of Railway Stations in the Brussels Regional Express Network: a Node-Place Modeling Approach[J]. Networks and Spatial Economics, 2018.
- [57] 蔡立雄. 基于Node.js的智慧农业数据采集平台的设计与实现[D]. 浙江理工大学, 2018.

附录 1 攻读硕士学位期间撰写的论文

- [1] 庞宗强, 王严晖, 吴浩, 基于物联网的消毒监控系统设计, 国外电子测量技术, 已录用;
- [2] 庞宗强, 吴浩, 王严晖, 张方政, 基于 STM32 的智能物联空气净化系统, 国外电子测量技术, 已录用。

附录 2 攻读硕士学位期间申请的专利

- [1] 庞宗强,王严晖,吴浩,袁明. 一种基于物联网的智能锁系统, 202110636606.4, 2021.06, 2021.09(已公开);
- [2] 杜长青,王严晖,刘寅莹,庞宗强. 一种全自动调平三脚支架, 202110249652.9, 2021.03, 2021.06(已公开);
- [3] 范舟,王严晖,杜长青,刘寅莹,庞宗强. 一种基于物联网的室内环境参数监测装置, 202120222792.2, 2021.01, 2021.12。

致谢

三年的研究生时光转瞬即逝，在这三年里，我有幸结识了许多的良师益友，他们在生活与学习中给予了我很多的帮助，在此向他们表示诚挚的感谢。

首先要感谢的是我的研究生导师庞宗强老师。不管是在学习、科研还是生活中，庞老师都给予了我非常悉心的指导和帮助。在撰写论文、专利的过程中，庞老师始终认真负责地帮助我修改完善。在遇到问题时，帮助我开拓思路。并且在研究生期间，让我们能有机会参与到工程实践中，从中学到了很多的技能，为之后的工作提供了巨大的帮助。

同时感谢课题组的龚煜滔师兄、石佳文师兄、白莉萍师姐、吴浩同学、彭君同学、陆昂同学、王庆师弟、王勇师弟、赵月荷师妹，在学习和生活中都给予了我很大的帮助。

感谢父母，二十多年，始终不言辛苦，默默付出。