|  |
| --- |
| 南农校徽  南农大  本科生毕业论文（设计）  题 目: 作物生长环境监测系统设计与实现  姓 名: 方定磊  学 号: 19219202  学 院: 人工智能学院  专 业: 计算机科学与技术  指导教师: 伍艳莲 职称 副教授  2023年5月30日 |

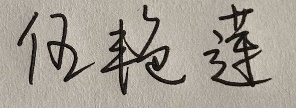
南京农业大学本科生毕业论文（设计）原创性声明

本人郑重声明：所呈交的毕业论文（设计），是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

论文作者签名：  日期： 2023 年 5 月30 日

南京农业大学本科生毕业论文（设计）使用授权声明

本学位论文作者完全了解学校有关保留、使用毕业论文（设计）的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权南京农业大学教务处可以将本毕业论文（设计）的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编毕业论文（设计）。

论文作者签名： 导师签名： 

日期： 2023 年 5 月 30 日 日期： 2023年 5 月 30 日

作物生长环境监测系统设计与实现

# 摘 要

我国是世界重要的农业大国之一，但现阶段我国部分地区的中小型农户在农业生产中存在一些问题。一个问题是对数据不够重视，导致产品分级定价难以让消费者信服，还有可能出现决策错误造成资源浪费。另一个问题是自动化程度低，大量的设备需要人工进行操作。而物联网技术是新一代的信息技术，能在很大程度上解决这两个问题。但现阶段，尚缺少操作简单、面向中小型农户的作物生长环境监测系统。

本文设计了一个作物生长环境监测系统的后端和客户端部分。系统整体采用C/S架构，后端使用C#WebAPI框架提供逻辑服务，Mysql存储数据，EF Core进行数据存取操作，客户端使用Winform和echarts。客户端和服务端使用gRPC进行信息传递。设备数据上传和命令接收使用软件进行模拟。系统共分为六个功能模块：用户模块、设备模块、设备数据模块、设备通讯模块、设备定时任务模块和冷数据处理模块。

系统的主要用户是中小型农户，用户可以通过此系统对物联网设备进行控制和对数据进行访问。告警机制方便了用户发现作物生长环境的异常，定时控制机制方便了用户进行设备控制。

关键词：环境监测系统；物联网；MQTT

**DESIGN AND IMPLEMENTATION OF CROP GROWTH ENVIRONMENT MONITORING SYSTEM**

# **ABSTRACT**

China is one of the world's major agricultural countries, but at this stage there are a number of problems in agricultural production for small and medium-sized farmers in some parts of the country. One problem is the lack of attention to data, which makes it difficult to convince consumers of the graded pricing of products, and there is also a risk of waste of resources due to wrong decisions. Another problem is the low level of automation, a large number of equipment requires manual operation. The Internet of Things technology is a new generation of information technology, which can solve these two problems to a large extent. However, at this stage, there is a lack of crop growth environment monitoring system that is simple to operate and oriented to small and medium-sized farmers.

In this paper, we design the back-end and client part of a crop growth environment monitoring system. The system uses C/S architecture, C#WebAPI framework for logical services, Mysql for data storage, EF Core for data access, and Winform and echarts for the client. gRPC is used for information transfer between the client and the server. Device data upload and command reception are simulated using software. The system is divided into six functional modules: user module, device module, device data module, device communication module, device timing task module and cold data processing module.

The main users of the system are small and medium-sized farmers, who can control the IOT devices and access the data through this system. The alarm mechanism facilitates users to detect abnormalities in the crop growth environment, and the timing control mechanism facilitates users to control the devices.

**KEY WORDS:** Environmental monitoring systems; Internet of Things; MQTT

# 目 录

[摘 要 I](#_Toc136383910)

[ABSTRACT II](#_Toc136383911)

[目 录 III](#_Toc136383912)

[[1] 绪论 1](#_Toc136383913)

[1 研究背景 1](#_Toc136383914)

[2 研究现状 1](#_Toc136383915)

[2.1 农业物联网 1](#_Toc136383916)

[2.2 相关软件 2](#_Toc136383917)

[2.3 通讯协议 2](#_Toc136383918)

[2.4 数据存储 2](#_Toc136383919)

[3 主要研究内容 2](#_Toc136383920)

[4 研究目标 3](#_Toc136383921)

[[2] 开发的关键技术 4](#_Toc136383922)

[1 开发语言 4](#_Toc136383923)

[2 开发框架 4](#_Toc136383924)

[3 数据库 4](#_Toc136383925)

[4 开发工具 4](#_Toc136383926)

[[3] 系统分析 6](#_Toc136383927)

[1 功能需求分析 6](#_Toc136383928)

[1.1 多租户 6](#_Toc136383929)

[1.2 用户与权限模块需求 6](#_Toc136383930)

[1.3 设备模块需求 7](#_Toc136383931)

[1.4 设备数据模块需求 8](#_Toc136383932)

[1.5 设备通讯模块需求 9](#_Toc136383933)

[1.6 定时控制模块需求 9](#_Toc136383934)

[1.7 站内信模块需求 10](#_Toc136383935)

[2 非功能需求分析 11](#_Toc136383936)

[2.1 安全性需求分析 11](#_Toc136383937)

[2.2 扩展性需求分析 11](#_Toc136383938)

[3 系统可行性分析 11](#_Toc136383939)

[3.1 技术可行性 11](#_Toc136383940)

[3.2 操作可行性 11](#_Toc136383941)

[[4] 系统设计与实现 12](#_Toc136383942)

[1 总体设计 12](#_Toc136383943)

[1.1 系统架构设计 12](#_Toc136383944)

[1.2 数据库表设计 13](#_Toc136383945)

[2 系统主要功能设计 19](#_Toc136383946)

[2.1 鉴权 19](#_Toc136383947)

[2.2 用户设备及分组 20](#_Toc136383948)

[2.3 设备数据访问 20](#_Toc136383949)

[2.4 定时控制设计 23](#_Toc136383950)

[2.5 上传数据处理 24](#_Toc136383951)

[2.6 设备通讯与数据处理设计 24](#_Toc136383952)

[2.7 用户关系查询优化 25](#_Toc136383953)

[2.8 告警功能设计 25](#_Toc136383954)

[2.9 租户和用户创建 25](#_Toc136383955)

[2.10 设备访问控制 26](#_Toc136383956)

[2.11 冷数据功能设计 27](#_Toc136383957)

[3 客户端设计与实现 28](#_Toc136383958)

[3.1 登录界面 28](#_Toc136383959)

[3.2 主界面 29](#_Toc136383960)

[3.3 全部设备界面 30](#_Toc136383961)

[3.4 设备分组编辑界面 31](#_Toc136383962)

[3.5 设备详情界面 31](#_Toc136383963)

[3.6 发送命令界面 32](#_Toc136383964)

[3.7 设备类型界面 32](#_Toc136383965)

[3.8 设备维修信息界面 33](#_Toc136383966)

[3.9 设备定时控制界面 34](#_Toc136383967)

[3.10 用户管理界面 34](#_Toc136383968)

[3.11 用户创建界面 37](#_Toc136383969)

[3.12 站内信界面 37](#_Toc136383970)

[3.13 设备数据界面 38](#_Toc136383971)

[3.14 设备冷数据界面 39](#_Toc136383972)

[[5] 功能测试 40](#_Toc136383973)

[1 初始化 40](#_Toc136383974)

[2 功能测试 40](#_Toc136383975)

[2.1 设备访问控制 40](#_Toc136383976)

[2.2 设备数据上传和显示 40](#_Toc136383977)

[2.3 设备定时控制 40](#_Toc136383978)

[2.4 冷数据 41](#_Toc136383979)

[2.5 集成测试 42](#_Toc136383980)

[3 局限性 45](#_Toc136383981)

[[6] 总结与展望 46](#_Toc136383982)

[1 总结 46](#_Toc136383983)

[2 问题与展望 46](#_Toc136383984)

[2.1 系统安全 46](#_Toc136383985)

[2.2 系统交互 46](#_Toc136383986)

[2.3 系统性能 46](#_Toc136383987)

[参考文献 47](#_Toc136383988)

[致 谢 49](#_Toc136383989)

# 绪论

## 研究背景

随着人口的不断增长和城市化的加速发展，农业生产的规模化和现代化已成为当前农业发展的重要趋势。人口迁移、宅基地空置、自然灾害等因素造成农村土地闲置[1]让拥有一定规模资金的个体农户得以租赁闲置的土地而成为中等规模的农户。

规模化种植给这些农户带来了更好的经济效益，但是其在农业生产中所存在的问题也影响了他们的进一步发展。其中一个重要的问题是其农业生产的信息化水平相对较低，缺乏数据支持。这主要是因为这些农业从业者大多数是传统的个体经营者，管理水平和文化素质普遍较低，他们往往缺乏对数据的重视而根据经验和直觉进行生产活动。一方面，这容易导致生产过程中出现错误决策而浪费资源，如在不合适的时候进行施肥和农药喷洒，进而导致生产效率和质量下降并影响最终的经济效益。另一方面，这些农户开始尝试对农产品进行分级定价以提高经济效益。但由于没有数据支撑，其往往使用大小、颜色等性状来对产品进行分级。而在很多情况下消费者对这种分级方式并不买账，其更关心农产品的口感是否合适、产品是否绿色健康。

此外，这些农户农业生产中的自动化程度相对较低，大量的设备需要人工操作，效率低下，这容易导致人力资源的浪费。相比于大规模农业生产者，这些农户往往不会对设备部署进行规划，这导致各种类型的设备分散在田地中，不能进行统一的管理。如果能够对设备进行远程操作，将会大幅提高生产效率，从而降低生产成本并提升经济效益。

这些农户的另一个特点是文化程度较低，对于提供给大规模农业生产者的软硬件系统，他们不能很好地掌握其使用方法。

基于以上背景和问题，结合物联网的发展趋势，本文旨在基于MQTT（Message Queuing Telemetry Transport）通讯协议和C#相关技术，以解决中等规模农户在数据采集和设备操控方面遇到的问题为主要目标，设计一个操作简单的多租户的作物生长环境监测系统的后端和客户端部分，使得农业数据采集和设备管理简单化。

## 研究现状

### 农业物联网

研究者们深入研究了使用物联网技术解决农业问题的意义、可行性。物联网技术是新一代信息技术的重要组成部分。现代化的网络服务和各种传感器技术，能够让使用者获取到海量、实时的环境数据，从而提高决策的科学性。同时，物联网技术还提供对边缘设备的实时控制[1]，可以大幅度提高设备对环境变化的响应速度并减少此方面的人力投入。因此，使用物联网来提高农业效率[3]是一种明智的选择。正如Jayavardhana Gubbi[4]预测的那样，物联网技术日趋完善，使用成本和使用难度不断降低，这使得如今物联网被应用在生活中的方方面面。在实践中，众多研究者使用物联网技术并结合大数据、人工智能等技术对农业数据进行采集和分析[5]。

### 相关软件

当前企业在农业物联网软件方面主要是提供一套物联网接入平台。一方面，一些软件公司，如腾讯、阿里、华为等，都提供物联网接入服务，支持MQTT等多种接入协议。但是这些服务仅实现了物联网的基础需求，包括设备管理、设备接入，但是没有提供可供用户直接使用的UI界面，如果使用他们提供的服务，仍然需要进行进一步的开发。另一方面，通讯公司，如移动，除了提供类似的平台服务外，还支持更多的设备通讯协议，有的甚至提供配套的硬件设备。当然，也有企业提供可直接使用的平台，一个比较受欢迎的、可供用户使用的、开源的物联网项目是物美智能，其提供基础的用户管理、设备管理、定时控制、数据可视化等服务，相比于上述物联网平台，其对用户更加友好，且不用二次开发就可以使用。但是其在用户上手难度上依然有很大的进步空间。同时，它们在设备权限管理上还有待提升，粗粒度的权限系统不能保护文化程度较低的用户远离危险操作。

### 通讯协议

通讯协议也是系统设计需要考虑的重要问题之一，是物联网设备通讯的基础。物联网设备工作环境通常比较恶劣，而通讯又是一个十分复杂且消耗能量的行为，因此对物联网设备来说，选择合适的协议至关重要。超文本传输协议（Hyper Text Transfer Protocol，HTTP）是一种单向通讯协议，即一个请求产生一次响应。而实际通信多以双向通信能力为主，故HTTP并不适用于物联网设备与云端的通信场景[6]。相对于HTTP，可扩展通信与存在协议（extensible messaging and presence protocol, XMPP）虽降低了开发成本，但是在安全和资源消耗上并没有得到提升[6]。目前，主流的应用层协议大多为发布-订阅机制。MQTT是其中最具有代表性的协议之一，其实现设备与平台之间的数据双向传递功能，其传输消耗小、信息传递高可靠、轻量化的特点[7]-[8]使之成为物联网通信协议的标准协议[9]。此外，MQTT还可以保证消息按正确顺序传输[10]。

### 数据存储

数据存储方式对物联网平台的影响也是巨大的。常用的关系型数据库管理系统在时序数据压缩方面表现不佳，维护成本高，单机写入吞吐量低，难以应对交易处理等需要对海量数据进行聚合分析的场景[11]。因此，为冷数据寻找合适的存储方式，将大大增加平台能够管理的数据的规模。在物联网数据存储方面，可用的技术主要为非关系型数据库、时序数据库、分布式文件系统。

## 主要研究内容

本研究首先需要了解相关的系统的设计[12]-[15]，然后利用相关技术，研究、设计并实现这样的一个系统的后端和客户端部分，为用户提供高质量的服务。研究内容如下：

1、分析现有的功能需求和未来可能的扩展方向。

2、按照需求，设计系统功能和逻辑。

3、划分系统模块并完成开发工作。

## 研究目标

本文的主要目的是设计一个为农业生产者提供便利的作物生长环境监测系统的后端和客户端部分，以降低农业数据和设备的管理的成本，具体目标如下：

1、实现作物生长环境监测系统所需的基础功能，如用户管理、设备管理、数据管理、设备控制等。

2、实现严格的设备访问控制，确保数据安全和设备访问安全。

3、实现冷数据管理。

# 开发的关键技术

## 开发语言

本系统使用C#进行开发，C#是一门简单、现代、面向对象和类型安全的编程语言。C# 的语法和特性受到Java和C++等语言的启发，但设计得更简洁且易于理解。C#是.NET框架的主要编程语言。在本项目中，C#作为主要的编程语言。

## 开发框架

为了提高开发效率，系统还使用了一些框架，具体情况如下：

DotNet WebApi：.NET WebAPI是一个用于创建基于HTTP的RESTful Web服务的框架。它允许开发人员使用C#构建高性能的Web服务。本系统中其主要用于承载gRPC服务。

gRPC：gRPC是一个高性能、开源的远程过程调用框架，使用Protocol Buffers作为数据交换格式。在本系统中，其用于服务端和客户端通讯。相比HTTP，GRPC的数据即使被拦截也很难破解协议结构，可以极大地提高系统安全性。

EF Core：Entity Framework Core（EF Core）是一个对象关系映射（ORM）框架。它允许开发人员使用C#等编程语言操作数据库。本项目中，数据库操作完全通过其进行，项目中不直接使用SQL语句。

C#Winform：C#WinForms（Windows Forms）是一个基于.NET框架的图形用户界面（GUI）库，是用于建立客户端程序的丰富的程序设计框架[16]。本系统中，其用于客户端，为用户提供图形化的界面，在主流操作系统win7、win10上用户点击exe即可使用，没有安装学习成本。

Echarts：Echarts是一个由百度开发的开源数据可视化库，用于创建交互式的图表。Echarts的API设计简洁易用，使开发人员能够快速地构建高性能、美观的数据可视化界面。本系统中其配合js用于数据显示。

## 数据库

本系统使用数据库来存储大部分数据，具体情况如下：

MySQL：MySQL是一个关系型数据库管理系统。本系统使用其在生产环境中存储数据。

SQLite：SQLite是一个轻量级关系型数据库，其内存数据库功能配合EF Core可以快速初始化测试数据并在连接关闭时自动释放空间。本系统使用其在测试环境中存储测试数据。

## 开发工具

本系统使用Microsoft Visual Studio（简称VS）开发，VS是一个基本完整的开发工具集，它包括了整个软件生命周期中所需要的大部分工具。本设计使用其进行C#、js代码编写和调试。

# 系统分析

## 功能需求分析

经过查阅资料和相关系统使用体验，确认主要功能包括：多租户、用户管理、设备、设备类型、物模型、设备数据、设备控制、站内信、冷数据管理。主要用例角色为系统管理员和租户。

### 多租户

对小规模的农业从业者来说，部署一个完整的系统时十分昂贵且没有必要的。而SaaS服务具有成本低、方便快捷等优点[17]，能很好地解决这一痛点。因此系统需要提供多租户功能，让多个租户能够共用同一个系统以节约成本。用例图如图3‑1。

C:\Users\Lenovo\Desktop\论文图片\用例\用例-租户管理.emf

图3‑1 租户管理用例图

Fig.3‑1 Tenant management use case diagram

### 用户与权限模块需求

权限管理是该系统的最重要的需求之一。该系统多租户的性质决定了权限在该系统中的极高重要性。

在本系统中，用户按权限分为租户账号和系统管理员账号，租户账号又分为顶级用户账号和普通用户账号。对于一个租户，只能有一个顶级用户账号，不同的租户之间的数据和权限互不干扰，一个租户的操作不会影响另一个租户。系统管理员可创建租户并授予其顶级用户的权限，租户能创建普通用户作为自己的子用户。对租户而言，系统使用树形的权限系统。顶级用户拥有系统的大部分权限，顶级用户通过将自己的权限授权给自己的子用户来让更多的用户获得权限。因此权限只能由用户的创建者授权给用户和转授给子用户，对于一个用户，其子用户之间不能相互授予权限。在用户权限被撤销时，其所有子用户的权限也一并被撤销。因此，一个用户的权限不会超过其父用户。

该模块需要为用户提供以下3方面的功能：

1、子用户管理。用户可以在系统中查看子用户，创建新的子用户，设置子用户的基础信息、分配子用户的权限。

2、子用户设备权限管理。用户可以查看子用户所拥有的设备，同时也可以将自己拥有的设备和权限授予子用户。

3、用户日志。用户可以查询自己和子用户的日志，如登陆情况，包括登陆时间和是否成功。

用例图如图3‑2。

C:\Users\Lenovo\Desktop\论文图片\用例\用例-用户模块.emf

图3‑2 用户管理用例图

Fig. 3‑2 User management use case diagram

### 设备模块需求

设备模块是该系统的最核心的模块，提供以下服务：

1、设备管理。用户可以查看设备基础信息，其中拥有权限的用户可以对设备信息进行修改。

2、设备类型管理。用户可以创建设备类型并基于设备类型创建新的设备。

3、设备分组管理。当设备数量较大时，找到需要的设备就变得很困难，系统要为用户提供设备分组服务，用户可以将设备分为多个组并能更容易地查看在同一分组的多个设备

4、设备维修信息管理。部署在户外的物联网设备容易损坏，系统提供设备维修信息管理。用户可以按设备和时间查询设备维修记录以为维修提供便利。

用例图如图3‑3。



图3‑3 设备管理用例图

Fig. 3‑3 Device management use case diagram

### 设备数据模块需求

该模块用于管理设备上传的数据，主要功能如下：

1、数据查看，用户需要查看一个设备最近所有数据情况，以判断设备或设备所在环境是否存在异常。不同的数据的范围不同，系统要提供直观的方式以让用户能更轻易地了解情况。

2、设备数据分析，用户可能需要对比多个设备同一时间段的数据，也可能需要了解一个设备不同日期的同一时间的数据情况，系统要为用户提供图表以便用户进行数据分析。

3、由于农业物联网设备通常需要采集多种数据且需要大量的设备，同时又需要长期存储，长时间后系统数据量将会变得很大，这将大大影响系统性能。因此系统需要能够将庞大的数据从数据库中分离出来，但用户又可以在需要的时候进行查询。

用例图如图3‑4。

C:\Users\Lenovo\Desktop\论文图片\用例\用例-设备数据模块.emf

图3‑4 设备数据管理用例图

Fig. 3‑4 Device data management use case diagram

### 设备通讯模块需求

本系统中，系统与设备使用MQTT协议进行通讯。包括两个部分：

1、数据接收和筛选，系统需要接收设备通过MQTT代理传来的数据并存储，考虑到设备并不属于系统，设备可能会上传非法的数据，因此系统还要对设备数据进行过滤。

2、命令下发，对于一个物联网设备系统，设备控制是必不可少的。本系统需要把命令发送给MQTT代理并由MQTT代理发送给对应的设备。

用例图如图3‑5。

C:\Users\Lenovo\Desktop\论文图片\用例\用例-设备通讯.emf

图3‑5 设备通讯用例图

Fig. 3‑5 Device communication use case diagram

### 定时控制模块需求

农业生产通常需要在固定的时间进行固定的操作，因此系统还需要提供定时控制功能。用户需要对某个设备配置定时控制，系统在对应的时间发送命令以使设备在预定的时间做出预定的行为。

考虑到设备所处的网络环境较差，可能受到干扰或不可用而不能接收到指令，在预定的时间段要多次下发命令直到时间超过预定时间段。

用户需要能够查看和编辑设备的定时控制。

用例图如图3‑6。

C:\Users\Lenovo\Desktop\论文图片\用例\用例-定时控制.emf

图3‑6 定时控制用例图

Fig. 3‑6 Timing control use case diagram

### 站内信模块需求

站内信模块是大部分管理系统具备的功能，可以方便用户进行信息传递。需要提供以下功能：

1、站内信发送和查看。为了用户能在系统上进行沟通，系统提供站内信功能，允许用户发送邮件给子用户和查看接收到的信件。

2、已读状态。信息通常具有时效性，用户有时希望了解自己的信息是否被查阅，系统要提供已读状态功能。

3、邮件推送。用户可能不及时登陆系统，为了让站内信被及时阅读，系统提供站内信转邮件推送功能，发送者可以将站内信转为邮件发送给接收者，避免接收者错过重要信息。

用例图如图3‑7。

C:\Users\Lenovo\Desktop\论文图片\用例\用例-站内信.emf

图3‑7 站内信用例图

Fig.3‑7 Internal mail use case diagram

## 非功能需求分析

### 安全性需求分析

系统要实现安全的访问控制。访问控制分为设备安全、流量安全、应用安全、云安全、接口安全。本系统主要考虑应用安全和接口安全。一方面要实现租户的隔离，对所有用户进行严格的权限限制，避免用户访问到其他租户的资源或者租户内的用户越权访问。另一方面要对设备进行权限限制，防止用户绕开权限访问设备而造成数据泄漏甚至是控制设备导致设备损坏。

### 扩展性需求分析

科技迅速发展，新的技术和需求层出不穷，要对系统的核心功能进行抽象设计，降低系统的耦合性，为未来可能的变化预留接口以提高可扩展性。当系统功能需要扩展或者改变时，只对增加的模块进行设计，只要改动少量原有代码就可以完成需要的功能。如为其他的设备通讯协议预留接口等。

## 系统可行性分析

### 技术可行性

该项目使用的技术被广泛使用，拥有良好的社区生态和丰富的文档，技术可行性较高。

### 操作可行性

系统为C/S架构，用户使用Windows系统的电脑打开客户端就可以访问系统。同时界面简单明了，方便用户使用，学习成本较低。因此在操作上具有较高的可行性。

# 系统设计与实现

## 总体设计

### 系统架构设计

系统采用C/S架构，架构如图4‑1所示：

C:\Users\Lenovo\Desktop\论文图片\IOT软件架构.emf

图4‑1 系统架构

Fig.4‑1 System architecture

系统分为四个部分：

外部依赖：系统对QQSMTP服务和MQTT服务进行依赖。

数据库：系统使用Mysql数据库进行数据存储，使用本地Sqlite数据库进行开发和测试。

客户端：为用户提供服务，使用gRPC与服务端通讯。

服务端：使用C#WebAPI提供逻辑服务，使用MQTT与设备通讯，使用QQSMTP发送邮件。

### 数据库表设计

以下为部分ER图（不含实体属性和部分关系）：

C:\Users\Lenovo\Desktop\论文图片\ER图.emf

图4‑2 部分实体关系图

Fig.4‑2 Partial entity relationship dsiagram

系统使用EFCore操作数据库，只要按规则定义对象，框架会自动创建、操作数据库，因此数据库表设计中的类型使用C#的类型，表设计如下：

表4‑1 用户表

Tab.4‑1 User table

|  |  |  |
| --- | --- | --- |
| 类型 | 字段 | 描述 |
| long | Id | 自增主键 |
| string | Name | 用户名 |
| string | Pass | 密码 |
| string | Phone | 电话号码 |
| string | EMail | 邮箱 |
| byte | Status | 账号状态（预留） |
| long | LastLogin | 最后一次登陆的时间戳 |
| int | UserTreeId | 租户ID |
| int | TreeDeep | 子用户深度 |
| int | MaxSubUser | 最大用户数量 |
| int | MaxSubUserDepth | 最大用户深度 |
| string | Authoritys | 用户权限，json数组 |
| long | CreatorId | 父用户 |

表4‑2 用户关系表

Tab.4‑2 User relationship table

|  |  |  |
| --- | --- | --- |
| 类型 | 字段 | 描述 |
| long | User1Id | 用户1id |
| long | User2Id | 用户2id |
| bool | IsFather | 用户1为父用户 |
| bool | IsSelf | 用户1和用户2相同 |
| int | UserTreeId | 租户id |

表4‑3 用户日志表

Tab.4‑3 User log table

|  |  |  |
| --- | --- | --- |
| 类型 | 字段 | 描述 |
| long | Id | 自增ID |
| long | CreatorId | 用户ID |
| int | UserTreeId | 租户ID |
| short | Type | 日志类型 |
| long | Time | 时间戳，精确到毫秒 |
| bool | Success | 操作是否成功 |
| string | Data | 记录信息 |

表4‑4 设备类型表

Tab.4‑4 Device type table

|  |  |  |
| --- | --- | --- |
| 类型 | 字段 | 描述 |
| long | Id | 自增ID |
| int | UserTreeId | 租户ID |
| string | Name | 类型名称 |
| string | Script | 备用 |
| long | CreatorId | 创建者ID |

表4‑5 物模型表

Tab.4‑5 Thing model table

|  |  |  |
| --- | --- | --- |
| 类型 | 字段 | 描述 |
| long | Id | 自增ID |
| long | UserTreeId | 租户ID |
| long | DeviceTypeId | 所属的设备类型 |
| string | Name | 物模型名称 |
| byte | Type | 数据类型 |
| string | Remark | 备注 |
| string | Unit | 单位 |
| float | MinValue | 最小值 |
| float | MaxValue | 最大值 |
| float | AlertLowValue | 安全范围最小值 |
| float | AlertHighValue | 安全范围最大值 |
| bool | Abandonted | 是否弃用 |

表4‑6 设备表

Tab.4‑6 Device table

|  |  |  |
| --- | --- | --- |
| 类型 | 字段 | 描述 |
| long | Id | 自增ID |
| int | UserTreeId | 租户ID |
| string | Name | 设备名称 |
| int | Status | 设备状态 |
| string | LatestData | 最近上传的数据 格式，{物模型id：值[，重复]} |
| string | LocationStr | 位置信息 |
| long | DeviceTypeId | 设备类型ID |
| long | CreatorId | 创建者ID |

表4‑7 用户设备分组表

Tab.4‑7 User device group table

|  |  |  |
| --- | --- | --- |
| 类型 | 字段 | 描述 |
| long | Id | 自增ID |
| string | Name | 名称 |
| long | CreatorId | 创建者ID |
| int | UserTreeId | 租户ID |

表4‑8 用户设备表

Tab.4‑8 User device table

|  |  |  |
| --- | --- | --- |
| 类型 | 字段 | 描述 |
| long | UserId | 用户ID |
| long | DeviceId | 设备ID |
| long | User\_Device\_GroupId | 设备分组 |
| int | UserTreeId | 租户ID |
| int | Authority | 权限 |

表4‑9 冷数据信息表

Tab.4‑9 Device cold data table

|  |  |  |
| --- | --- | --- |
| 类型 | 字段 | 描述 |
| long | Id | 自增ID |
| long | DeviceId | 设备ID |
| long | StreamId | 数据ID |
| long | CreatTime | 压缩时间 |
| long | StartTime | 第一个数据的时间 |
| long | EndTime | 最后一个数据的时间 |
| int | Count | 数量 |
| string | Pars | 参数 |
| byte | status | 数据状态 |
| string | ManagerName | 管理器名称 |
| long | TreeId | 租户ID |

表4‑10 设备数据表

Tab.4‑10 Device data point table

|  |  |  |
| --- | --- | --- |
| 类型 | 字段 | 描述 |
| long | Id | 自增ID |
| long | DeviceId | 设备ID |
| long | StreamId | 数据ID |
| long | Time | 时间 |
| float | Value | 值 |

表4‑11 设备定时控制信息表

Tab.4‑11 Device time plan table

|  |  |  |
| --- | --- | --- |
| 类型 | 字段 | 描述 |
| long | Id | 自增ID |
| bool | Open | 是否开启 |
| byte | Order | 优先级 |
| string | Name | 名称 |
| byte | TriggerType | 触发器类型 |
| long | TimeStart | 开始时间 |
| long | TimeEnd | 结束时间 |
| long | OwnerID | 所属的设备ID |
| byte | Week | 周信息 |
| string | Cmd | 执行的命令 |

表4‑12 设备维修表

Tab.4‑12 Device repair table

|  |  |  |
| --- | --- | --- |
| 类型 | 字段 | 描述 |
| long | Id | 自增主键 |
| int | UserTreeId | 租户ID |
| long | DeviceId | 设备ID |
| long | DiscoveryTime | 设备损坏发现时间 |
| long | CompletionTime | 设备维修完成时间 |
| string | Context | 维修详情 |
| long | CreatorId | 创建者ID |

表4‑13 冷数据配置表

Tab.4‑13 Cold data setting table

|  |  |  |
| --- | --- | --- |
| 类型 | 字段 | 描述 |
| long | TreeId | 租户ID |
| string | ManagerName | 使用的管理器 |
| long | ColdDownTime | 冷却时间 |
| long | MinCount | 一个批次最小的数据量 |
| bool | Open | 是否启用 |

表4‑14 设备命令记录表

Tab.4‑14 Device command table

|  |  |  |
| --- | --- | --- |
| 类型 | 字段 | 描述 |
| long | ID | 自增主键 |
| long | DeviceId | 设备ID |
| string | Cmd | 命令 |
| byte | SenderType | 发送者类型 |
| long | SenderId | 发送者ID |
| long | SendTime | 发送时间 |
| bool | Success | 是否成功 |

表4‑15 站内信表

Tab.4‑15 Internal mail table

|  |  |  |
| --- | --- | --- |
| 类型 | 字段 | 描述 |
| long | Id | 自增ID |
| int | UserTreeId | 租户ID |
| string | Title | 主题 |
| string | Context | 内容 |
| long | Time | 时间 |
| bool | Readed | 已读状态 |
| long | SenderId | 发送者ID |
| long | ReceiverId | 接受者ID |
| long | LastEMailTime | 最近一次邮件推送时间 |

## 系统主要功能设计

### 鉴权

系统接口统一地对用户请求进行鉴权，鉴权流程较简单不做赘述，活动图如图4‑3：

C:\Users\Lenovo\Desktop\论文图片\登陆.emf

图4‑3 请求鉴权

Fig.4‑3 Request authentication

此外，对特定的接口，还需要按对应的请求资源进行鉴权，流程在每个接口中各有不同，以设备相关接口为例，其流程如图4‑4：

C:\Users\Lenovo\Desktop\论文图片\设备鉴权.emf

图4‑4 设备访问鉴权

Fig.4‑4 Device access authentication

### 用户设备及分组

设备数据属于租户，但是并非租户的所有用户都可以访问。当一个设备被创建时，用户和父用户都获取到设备的全部权限。而不是父用户和当前用户的用户则不能访问相应的设备。

为了方便管理，用户可以对设备进行分组，设备分组仅对当前用户有效，不同的用户可以按照设备位置、设备功能、自己的职责等因素对设备进行分组。设备的分组信息记录在用户设备信息中。

### 设备数据访问

设备数据访问分为三种，分别是一个设备最近的所有数据、同时段的不同设备的同一种数据、同一设备同一时间的不同日期的数据，其活动图如下：

C:\Users\Lenovo\Desktop\论文图片\查看最新数据.emf

图4‑5 展示设备最近的所有数据

Fig.4‑5 Show all recent data from the device

C:\Users\Lenovo\Desktop\论文图片\不同设备同一时间.emf

图4‑6 同时段的不同设备的同一种数据

Fig.4‑6 Same data from different devices at the same time

C:\Users\Lenovo\Desktop\论文图片\同一设备不同时间段.emf

图4‑7 同一设备同一时间的不同日期的数据

Fig.4‑7 Data from the same device on different dates at the same time

图4‑5为一个设备最近的所有数据的处理流程，流程如下：

1、用户选择想要查看的设备，客户端发起请求。

2、后端鉴权，如果没有对应设备的权限，则禁止访问。

3、后端从设备表获取最近的设备（为了方便查询，设备的最新数据会记录在设备信息表中而不是从数据表中查询每种数据最新的一条记录）。

4、客户端根据设备类型信息，渲染每个数据，数据以表盘的形式显示。

图4‑6为同时段的不同设备的同一种数据的处理流程，流程如下：

1、用户选择一个设备类型下的多个设备和一种数据。

2、客户端依次查询每个选中设备的相应数据并由后端响应。

3、对异常请求直接忽略，将可用数据进行合并。

4、客户端将数据转为json格式传递给内嵌的浏览器。

5、内嵌的浏览器将json解析为js对象。

6、内嵌的浏览器使用echarts将数据以折线图的方式绘制，横轴为选定的时间段，纵轴为数值，图例包含多个设备。

图4‑7为同一设备同一时间的不同日期数据的处理流程，流程如下：

1、用户选择一个设备和一种数据及数据的时间范围。

2、客户端发起请求，传递设备编号，数据编号，时间段。

3、服务端鉴权后返回数据。

4、客户端将数据按天进行聚合。

5、客户端将数据转为json格式传递给内嵌的浏览器。

6、内嵌的浏览器将json解析为js对象。

7、内嵌的浏览器使用echarts将数据以折线图的方式绘制，横轴为0-24小时，纵轴为数值，图例包含多个日期。

其中上述流程存在一个难点是考虑时区的情况下对时间戳按天聚合，方法为天=(时间戳-时区时间戳差值)/一天时间戳。

C:\Users\Lenovo\Desktop\论文图片\时区处理.emf

图4‑8 时间处理

Fig.4‑8 Time processing

同时为了绘制在同一天，还要将时间戳转为在同一天的时间戳，方法如下

1、特征=时间戳%一天时间戳（此时特征落在UTC时间的0-24小时中）

2、如果特征大于24-时区差，则特征=特征-一天时间戳

3、特征即为需要的值，时间段为[-时区时间戳差值，-时区时间戳差值+一天时间戳）

### 定时控制设计

定时控制由分组和分组中的配置组成，配置包含任务、触发方式、触发时间，任务为系统需要向设备发送的命令，触发方式有总是、时间段，总是表示一直生效，时间段表示在选定的开始时间和结束时间之间生效。触发时间为触发方式的补充数据。同一时间有多个配置符合时间要求则只有优先级最高的配置生效。

定时控制的处理流程如下图：

C:\Users\Lenovo\Desktop\论文图片\定时控制.emf

图4‑9 定时控制处理

Fig.4‑9 Time plan processing

系统处理定时控制的流程如下：

1、每隔一段时间执行下述步骤。

2、获取系统中所有设备ID。

3、对每个设备执行下述操作。

4、获取该设备的定时配置信息，按配置名称聚合，按优先级排序。

5、对每个名称的定时配置组按序查看第一个符合的时间段。

6、如果有符合的时间段，则获取其中的命令。

7、向设备发送命令和写命令日志。

上述流程在执行中出错会忽略该设备继续执行下一个设备。

### 上传数据处理

设备上传的数据只能是其类型中所拥有的物模型中定义的数据类型，对不在其中的数据则抛弃而只记录有效的数据。

### 设备通讯与数据处理设计

MQTT协议可以将数据在设备和后端之间传递，其中数据包含话题和数据体。服务端根据话题对数据进行处理，而话题传输错误导致解析错误的概率比较小，而更大可能是在传输错误的情况下被系统忽略，不会造成较大的影响，故不进行完整性校验，但数据体传输错误可能导致数据分析时产生不良影响，故对其进行完整性校验。使用CRC16进行校验，传输时小端在前。

设备数据：该话题用于设备向服务器传输监测数据。话题格式为 “/设备ID/data”，载荷由CRC16校验数据和UTF8编码的字符串组成，UTF8字符串格式为 "设备数据点ID,数据[,可多次重复前者]"

设备命令：该话题用于服务器向设备传输命令。话题格式为“/设备ID/cmd”，载荷由CRC16校验数据和UTF8编码的字符串组成，UTF8字符串为所需发送的命令。

### 用户关系查询优化

在树形的用户关系中，查询一个用户的所有子用户和所有父用户需要进行多次数据库查询，本设计中设计了一个独立的表用于记录每一个用户与其所有相关的用户的关系。为了提高效率，自己与自己的关系也被记录以用于支持查询自己和所有子用户和查询自己和所有父用户。

但是该设计会导致较多的冗余数据，尤其是每个用户只有一个子用户的情况下，共约产生用户数量平方数量的记录，为此，在系统中对子用户的层级进行限制，以使得用户不能通过恶意创建大量的用户来增加数据库查询压力。

### 告警功能设计

告警机制使用邮件来通知用户，其时序图如图4‑10。设备发出数据到MQTT代理服务器，随后代理服务器将数据转发给后端进行处理。后端获取数据库中的相关数据并进行告警判定，包括数值是否异常、是否达到告警时间、是否已经完成告警，如果需要告警则向SMTP服务器推送告警邮件。用户可查看邮件以了解设备告警情况。



图4‑10 告警时序图

Fig.4‑10 Time sequence diagram of alarm

### 租户和用户创建

租户只能由系统管理员创建，流程如下：

1、创建用户，并初始化权限为顶级用户。

2、创建用户关系。

用户能创建子用户，流程如下：

1、校验用户层级是否过高。

2、校验当前用户和上级用户是否达到子用户数量限制。

3、创建新用户，并初始化权限为无权限。

4、创建新用户与自己和所有上级的用户关系。

### 设备访问控制

设备访问权限分为三个级别：读信息、写信息、设备控制，包括基础信息、设备采集的数据、设备维修信息、设备命令记录、设备定时配置、发送命令等功能的权限。除了这些对设备的权限，还有对这个权限是否可转授的权限。

设备权限的转授和撤销合并为一个操作，流程如下图所示：

C:\Users\Lenovo\Desktop\论文图片\设备权限转授.emf

图4‑11 权限转授流程

Fig.4‑11 Delegation of permissions

### 冷数据功能设计

系统压缩冷数据的流程如下图4‑12：

C:\Users\Lenovo\Desktop\论文图片\冷数据处理.emf

图4‑12 冷数据处理

Fig.4‑12 Cold data processing

流程描述如下：

1、每隔一段时间执行下述步骤

2、获取系统所有的租户

3、对每个租户执行下述流程

4、获取租户冷数据配置

5、如果租户未开启冷数据功能，则忽略该用户

6、获取租户所有设备ID

7、对每个设备执行下述操作

8、获取其类型及其物模型

9、统计该设备所有物模型下的实例数据

10、如果数据不多，则忽略

11、抽取一定数量的数据并按时间排序

12、记录该组数据的最早时间，最晚时间，数量等信息并插入冷数据信息（此操作是为了获取一个ID）

13、将数据重新构建为多组时间偏移（原时间戳-第一个数据的时间戳）\数据的形式，以二进制方式存储

14、对二进制数据进行压缩

15、用租户选定的方式进行数据存储

16、重新插入冷数据信息（此操作是为了将冷数据管理器做出的更新加入数据库）

17、删除已经压缩的数据

## 客户端设计与实现

界面仅以完成功能为主，在美观性上没有过多关注。以下为界面设计和实现：

### 登录界面

登录界面如图4‑13，输入账号密码点按钮即可进入系统。

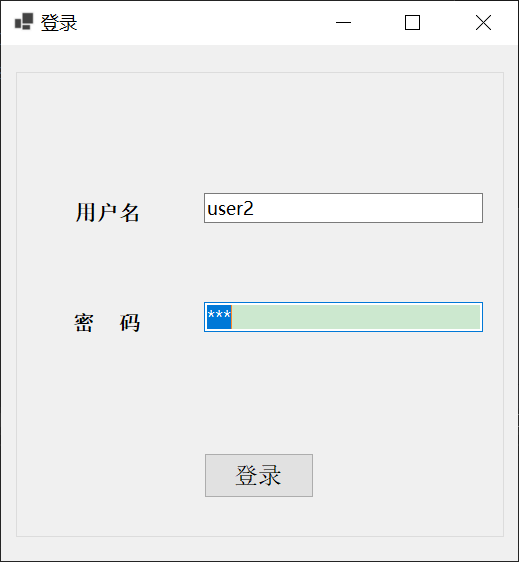


图4‑13 登录界面

Fig.4‑13 The interface of login

### 主界面

系统主界面如图4‑14 主界面，点击左侧菜单栏即可打开对应的功能页面，旁边的按钮可在打开的界面有多层时回到上一个界面。

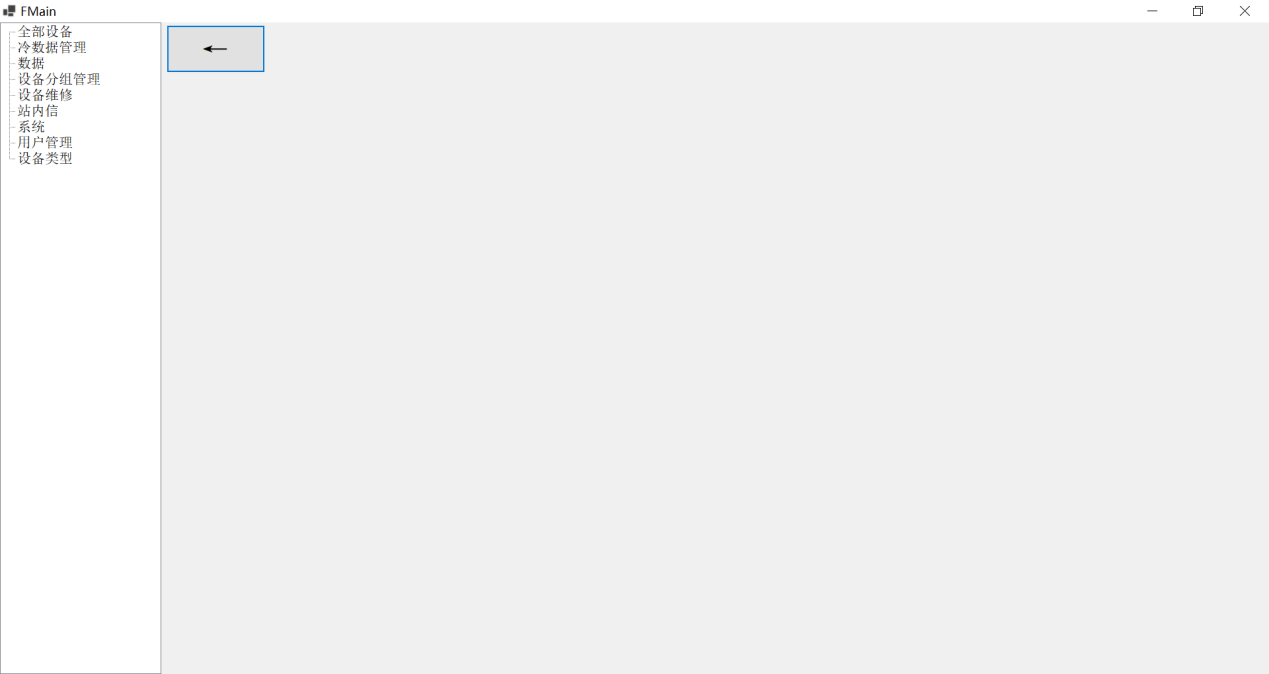


图4‑14 主界面

Fig.4‑14 The main interface

其中菜单使用反射的方式自动加载，原理如：

C:\Users\Lenovo\Desktop\论文图片\流程\流程-主界面初始化.emf

图4‑15 界面初始化流程图

Fig.4‑15 Interface initialization flow chart

### 全部设备界面

全部设备界面如图4‑16，左侧为设备分组，设备分组只针对一个用户，不同的用户可以将设备分到不同组中。一个用户对设备进行分组对另一个用户不可见。点击对应的分组可对设备进行筛选，中间为设备信息，展示了当前符合条件的所有设备。右侧显示在线和离线为预留按钮，暂无功能。右侧的按钮可以完成对应的功能或者进入对应的界面。全选可全选或者撤销全选设备，分组可进入分组编辑界面，发送命令可向若干个设备发送命令，移动分组可以选择若干个设备并通过设备拖拽移动设备的分组，定时配置可进入若干个设备的定时配置界面。

此外中间的按钮也可以直接进入对应的设备功能界面。

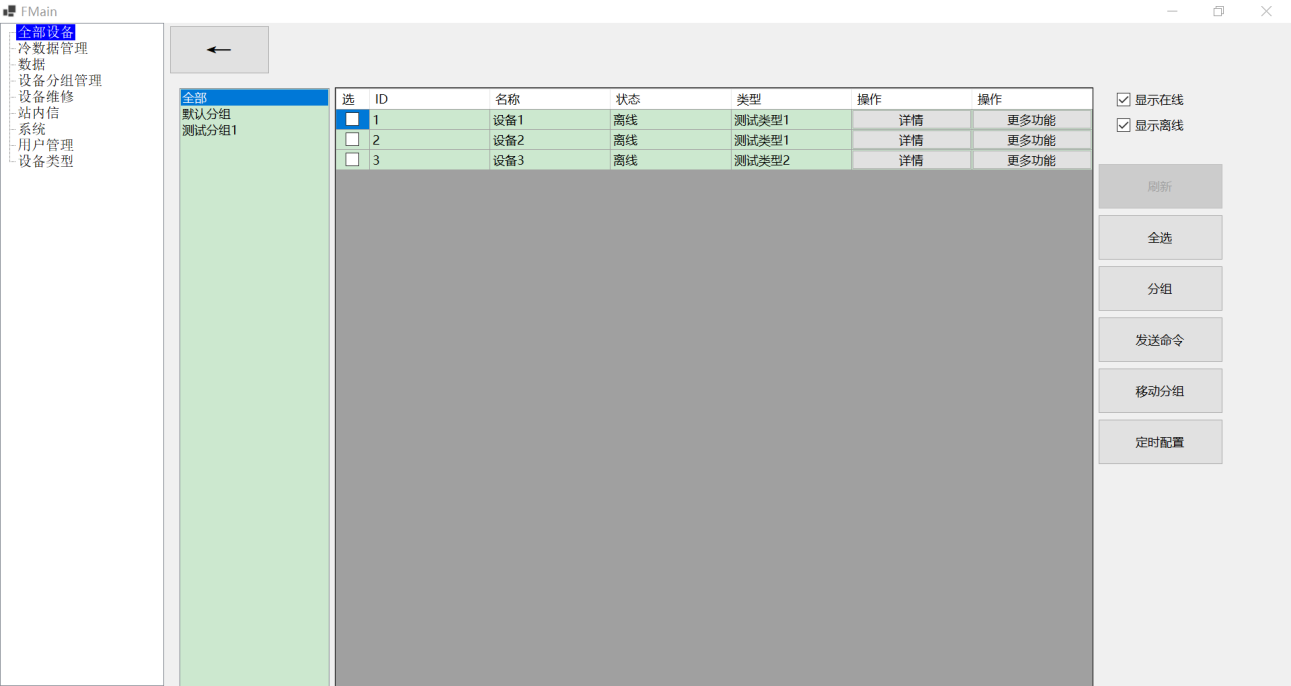


图4‑16 全部设备

Fig.4‑16 The interface of all devices

### 设备分组编辑界面

分组编辑界面如图4‑17，可添加、删除分组，也可选中分组后修改其名称



图4‑17 设备分组编辑

Fig.4‑17 The interface of device group

### 设备详情界面

设备详情界面如图4‑18，可从全部设备页面进入，显示设备基础信息和最近一次数据，其中数据以echarts图标显示，仪表上包括当前值，最大值最小值，警戒最大值和最小值，红色部分为警戒范围，绿色部分为正常范围。点击左下方按钮可进入对应的功能界面。

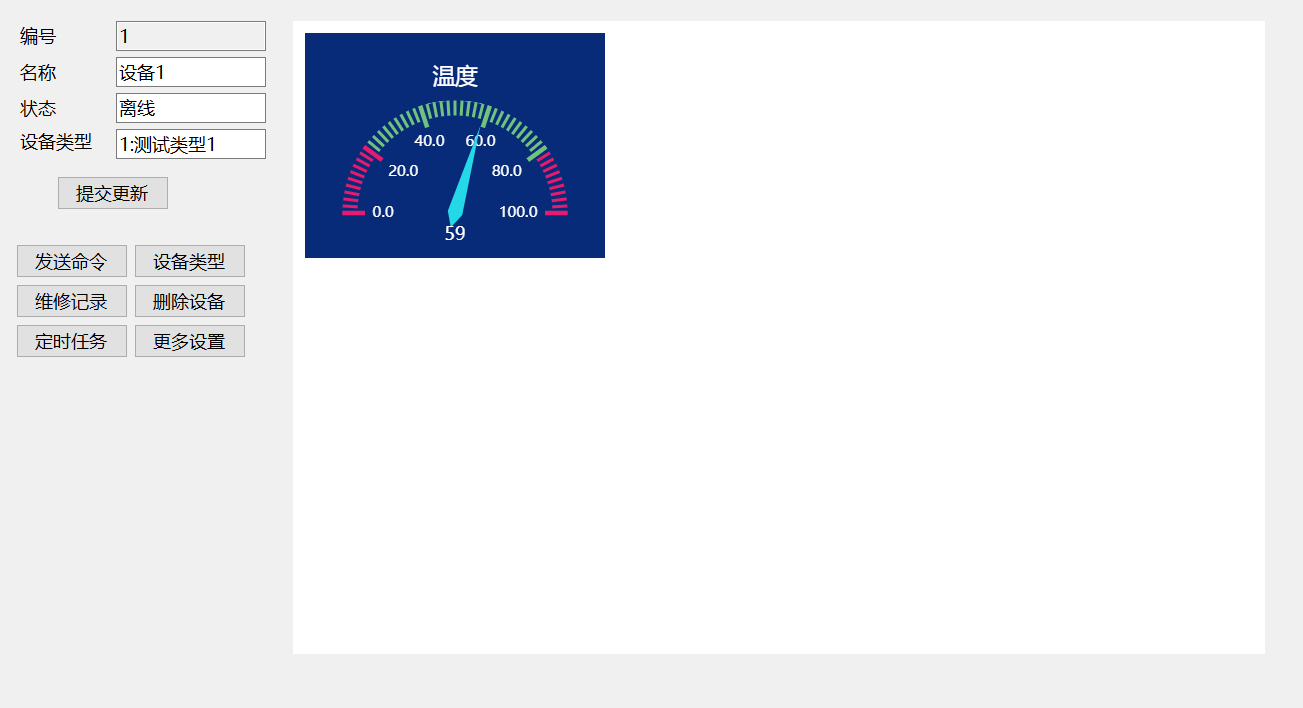


图4‑18 设备详情

Fig.4‑18 The interface of device detail

### 发送命令界面

发送命令可从全部设备页面或者设备详情界面进入，支持对单个或者多个设备发送命令。界面如图4‑19。

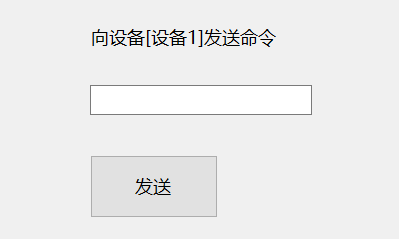


图4‑19 发送命令界面

Fig.4‑19 The interface of send command

### 设备类型界面

界面如图4‑20，可修改设备类型名称，以及其对应的物模型。点击创建设备按钮可以此设备类型创建一个新的设备。如从设备详情界面进入该界面，则不显示左侧类型选择栏。



图4‑20 设备类型界面

Fig.4‑20 The interface of device type

### 设备维修信息界面

界面如图4‑21，该界面可以筛选一定范围内的维修记录和对维修记录进行增改。

该界面可从设备详情界面进入，也可从主界面进入，从设备详情界面进入只显示对应设备的维修记录，否则显示所有设备的维修记录

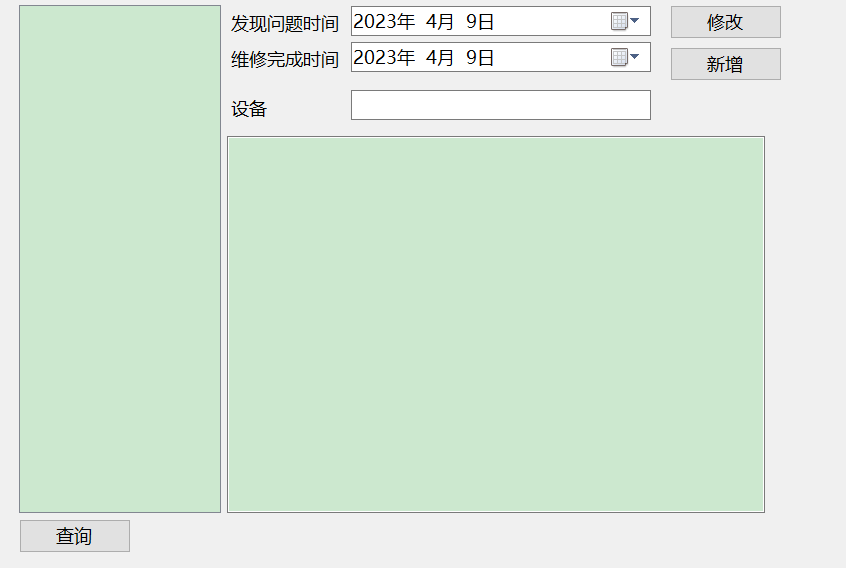


图4‑21 设备维修信息界面

Fig.4‑21 The interface of device rapair

### 设备定时控制界面

设备定时控制界面如图4‑22，展示了设备定时控制数据，在左边为分组，每个分组同一时间最多有一个配置生效。在中间显示了分组中的配置，分组中可以有多个配置。右侧为功能按钮，对配置进行增删改操作。箭头用于修改配置的优先级，即其在列表中的上下位置。确定用于提交修改。测试用于获取当前生效的配置以便判断定时配置设置是否正确。

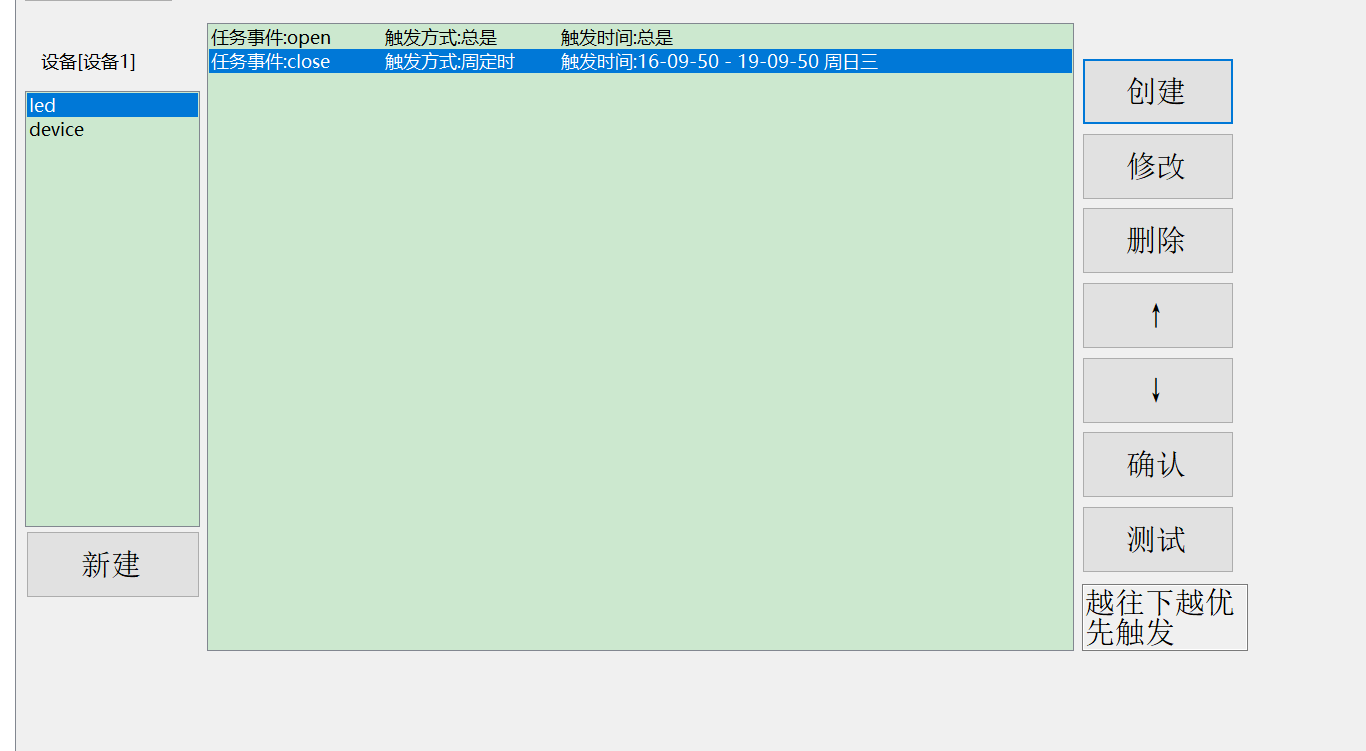


图4‑22 设备定时控制界面

Fig.4‑22 The interface of time plan

### 用户管理界面

用户管理界面如图4‑23，展示了用户的基础信息和子用户的基础信息，包括用户名、ID、电话、邮箱等信息。

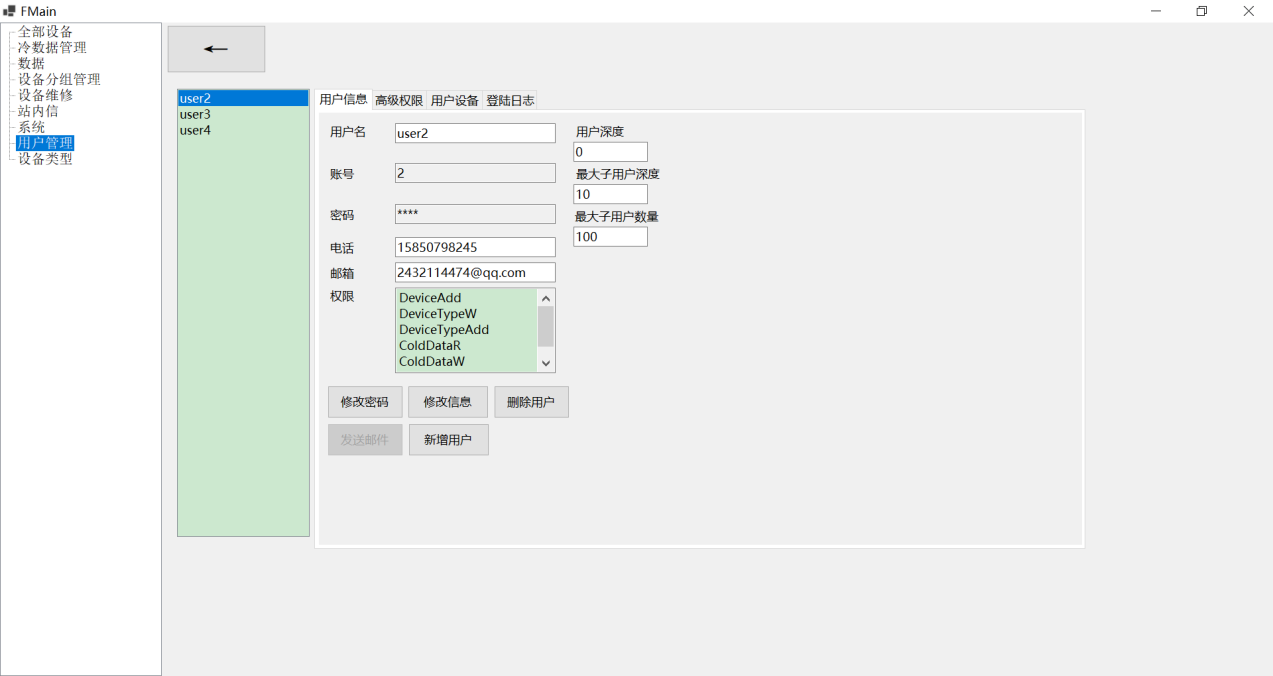


图4‑23 用户信息界面

Fig.4‑23 The interface of user info

用户权限管理界面如图4‑24，显示了当前用户所有的权限，选择子用户即可展示子用户当前拥有的权限，打钩的为子用户拥有的权限。点击已有权限，可撤销权限，该行额外显示减号表示撤销，点击没有分配的权限将显示加号表示新增权限，点击下方按钮进行提交。对于撤销权限，也会一并撤销其子用户的多有子用户的权限。



图4‑24 用户权限界面

Fig.4‑24 The interface of user permissions

其中，权限的文字显示逻辑如表4‑16。

表4‑16 权限文本伪代码

Tab.4‑16 Permission text pseudo code

|  |
| --- |
| 权限文本伪代码 |
| 输入：用户权限和多选框操作  输出：多选框文本   1. 如果多选框选中状态与是否拥有权限的真假性一致，返回原文本 2. 如果原本没有，现在选中，则返回加号和原文本 3. 返回减号和原文本 |

用户设备界面如图4‑25，显示了用户所有有权限的设备，设备以分组方式进行筛选和展示，右侧显示了所有的权限，打钩为已拥有的权限，只能撤销设备权限或者将自己已有的权限分配给子用户。

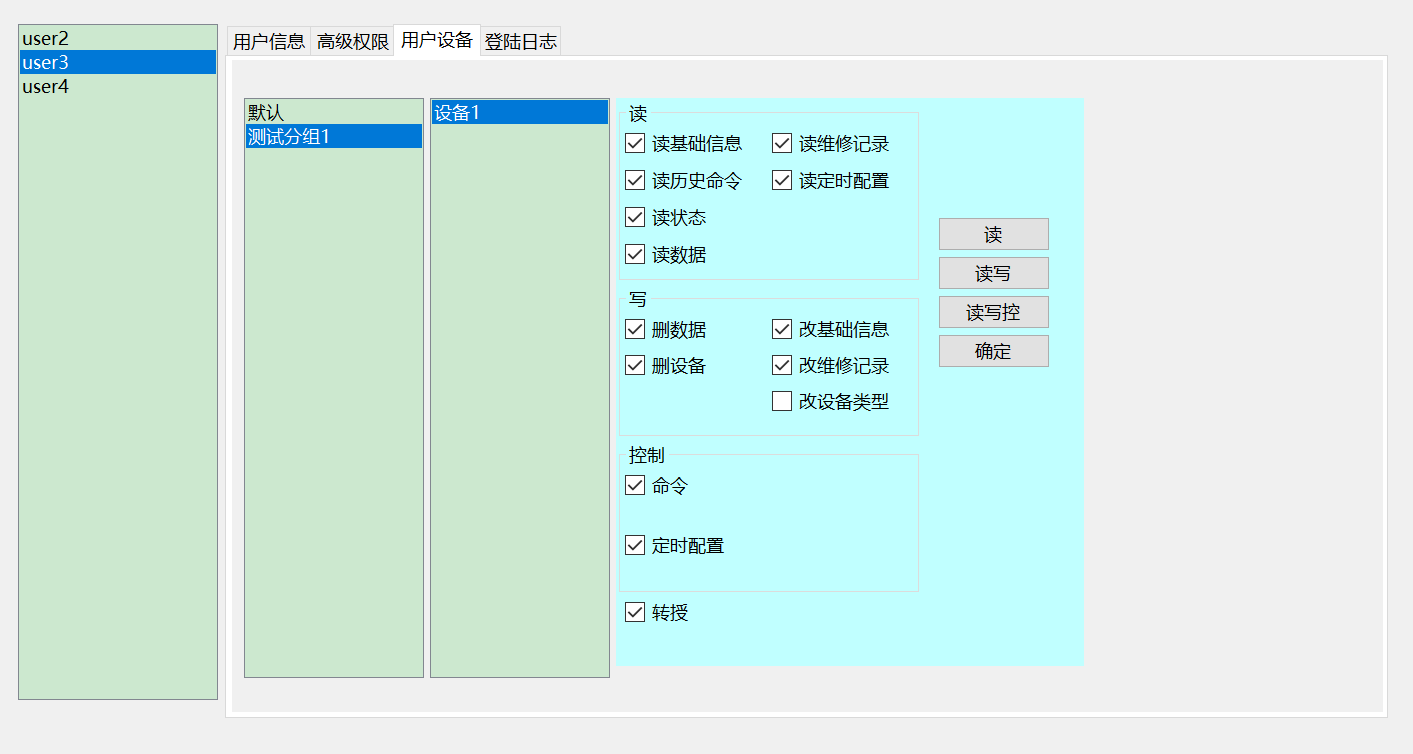


图4‑25 用户设备界面

Fig.4‑25 The interface of user device

其中界面上的多选框状态初始化逻辑如表4‑17。

表4‑17 用户设备界面初始化伪代码

Tab.4‑17 User device interface initialization pseudo code

|  |
| --- |
| 用户设备界面初始化伪代码 |
| 输入：用户设备权限和子用户设备权限  输出：多选框的选择状态   1. 如果当前用户没有该设备的转授权限，返回不可用 2. 如果当前用户没有该权限，返回不可用 3. 返回选中状态为子用户是否有该权限 |

登陆日志界面如图4‑26，展示了用户的登陆情况，可以基于时间进行筛选。

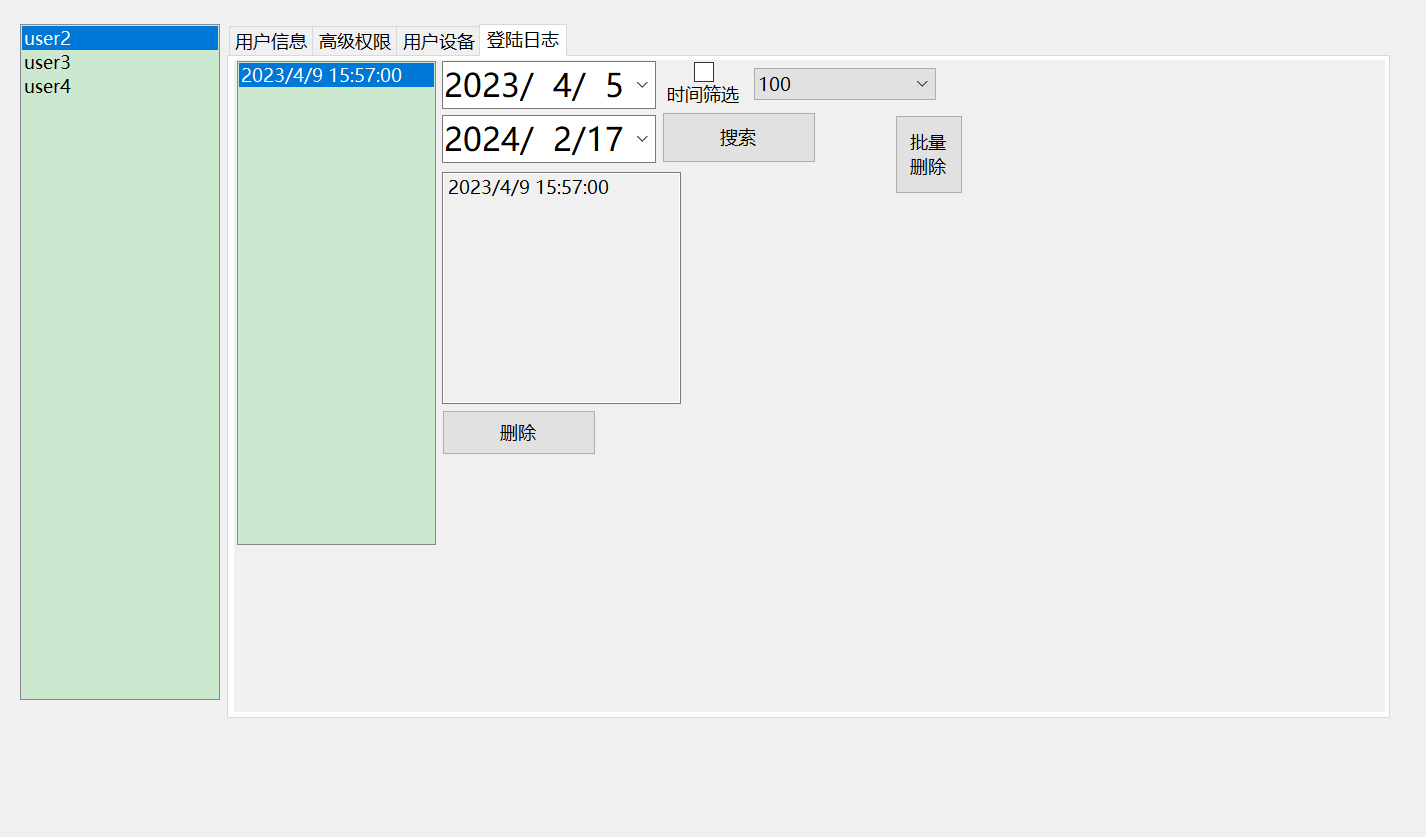


图4‑26 用户日志

Fig.4‑26 The interface of user log

### 用户创建界面

用户创建界面如图4‑27，可创建用户，普通子用户初始化没有权限，需要自行添加。顶级用户创建租户则默认拥有所有的租户权限。

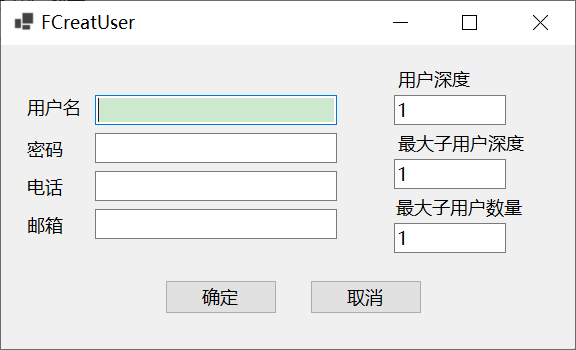


图4‑27 创建用户界面

Fig.4‑27 The interface of add user

### 站内信界面

站内信界面如图4‑28，展示了自己收到的和发送的站内信，点击指定的标题可查看对应的邮件，查看后邮件变成已读状态，接收者可以选择删除邮件。发送者可以选择向用户的邮箱地址推送一封邮件以便更好地提醒用户。



图4‑28 站内信界面

Fig.4‑28 The interface of internal mail

### 设备数据界面

设备数据界面，展示了设备数据。有两种模式：

第一种为同一个类型的不同设备的同一时间的数据之间的对比，数据以折线图的方式显示，如图4‑29。

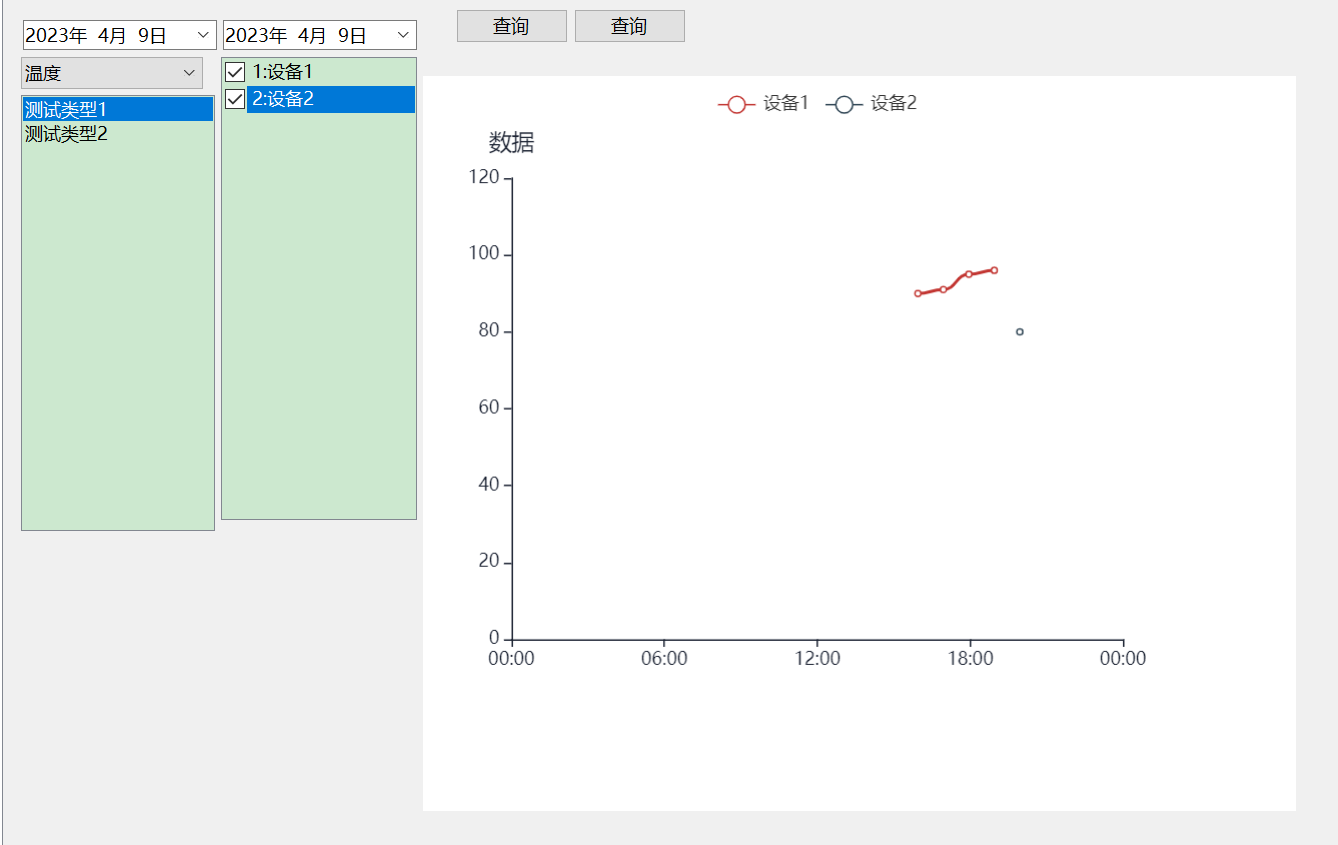


图4‑29 设备数据界面1

Fig.4‑29 The interface of device data 1

第二种为同一设备的同时间的不同天的数据展示，如图4‑30。

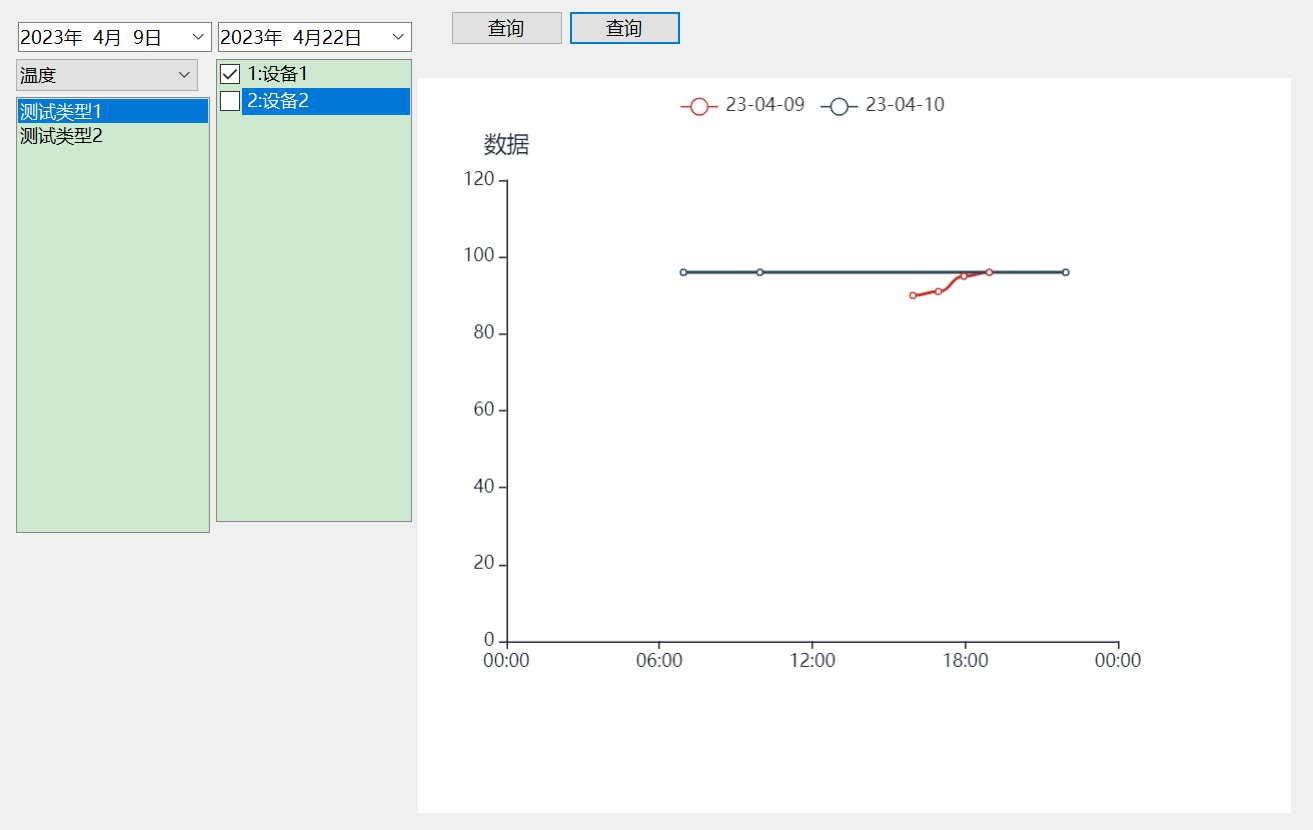


图4‑30 设备数据界面2

Fig.4‑30 The interface of device data 2

### 设备冷数据界面

冷数据管理界面如图4‑31，用于管理冷数据，在最右侧为全局配置，配置了将设备数据转为冷数据的条件和使用的转换方式。在左侧为当前的冷数据，最左侧可以对数据进行筛选，包括按时间和设备，在中间为冷数据的相关信息和用户可进行的操作。

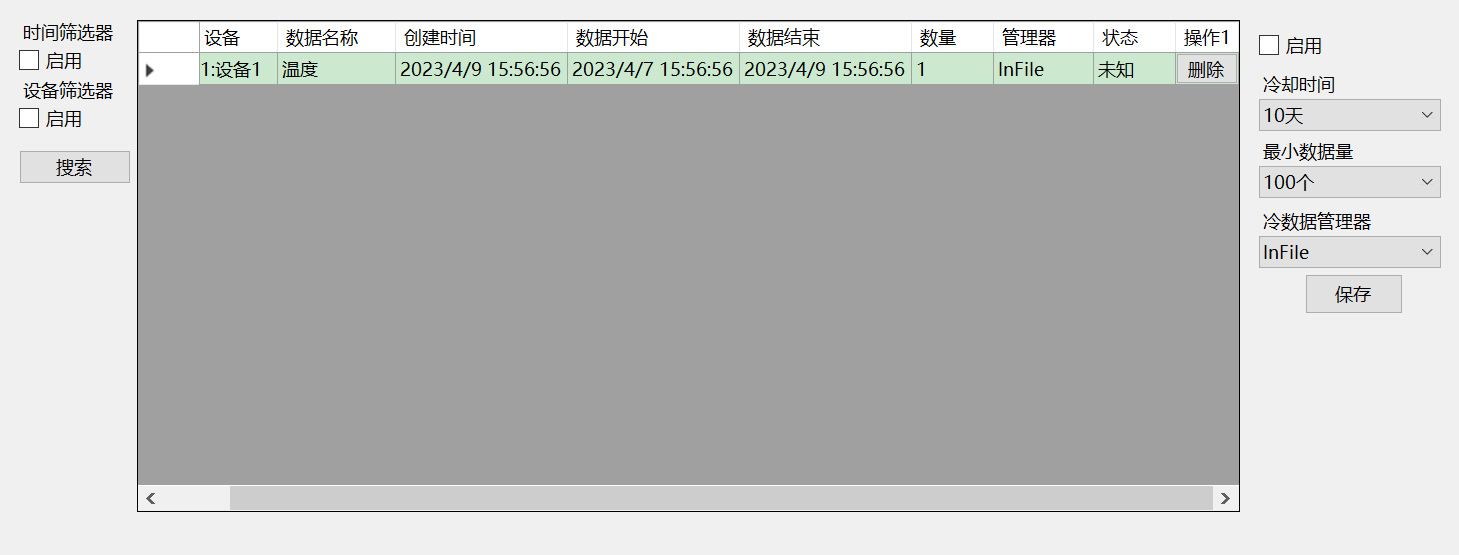


图4‑31 设备冷数据界面

Fig.4‑31 The interface of device cold data

# 功能测试

## 初始化

EFCore提供了面向对象的初始化功能，通过函数调用进行初始化，初始化数据包括用户、设备类型、物模型、设备等数据。

对系统进行测试难免要修改数据库，如果每次测试都修改数据库为初始值，将大大影响测试效率。因此，在本系统中，除了可以使用mysql数据库来持久化存储数据，还使用了sqlite的内存数据库功能，配合EFCore的自动初始化功能，可以让测试数据更快地完成初始化。

## 功能测试

项目功能较多，需要的测试也较多，本文只详细介绍部分功能的测试方式、结果。

### 设备访问控制

设备权限是本项目的核心，使用的测试数据为：一个设备、一个设备类型、几个设备数据、一个拥有该设备完全控制权限的用户（以下称为用户1）、一个只拥有该设备基础信息查看权限的用户（以下称为用户2）。

测试结果如表5‑1，符合预期结果：

表5‑1 设备访问控制测试结果

Tab.5‑1 Device access control test results

|  |  |  |
| --- | --- | --- |
| 操作 | 用户1 | 用户2 |
| 查看设备 | 成功 | 成功 |
| 查看设备数据 | 成功 | 失败 |
| 发送命令 | 成功 | 失败 |
| 查看和修改设备维修记录 | 成功 | 失败 |
| 查看和修改定时配置 | 成功 | 失败 |
| 转授设备权限 | 成功 | 失败 |

### 设备数据上传和显示

该部分测试系统是否与MQTT代理正常通讯和设备数据是否被正确处理。

测试流程如下：

1、使用设备模拟器上传数据。

2、在系统中查看设备最新数据。

3、在系统中查看设备近期数据。

测试结果：符合预期。

### 设备定时控制

该部分测试设备定时控制功能是否符合预期。

测试流程如下：

1、修改设备定时控制运行的时间间隔，以方便测试。

2、创建一个一直生效的配置，查看设备模拟器收到的命令。

3、创建一个时间段定时，查看设备模拟器收到的命令。

4、将其创建在一起，测试不同排列顺序下设备模拟器收到的命令。

5、删除所有定时配置，查看设备模拟器是否收到的命令。

测试结果：符合预期。

### 冷数据

该部分测试冷数据配置处理、冷数据压缩是否符合预期以及压缩效果。

测试流程：

1、修改冷数据处理的时间间隔，以方便测试。

2、生成大量数据，在系统中查看是否创建成功。

3、修改冷数据配置，将其启用并设置最小数据量为较大值，查看数据是否被转为冷数据。

4、修改冷数据配置，将其最小数据量设为较小值，查看数据是否被转为冷数据

5、查看被转为冷数据的数据是否被删除。

6、查看冷数据是否能被正确解析并显示。

7、删除冷数据，查看其对应的数据和文件是否被删除。

测试结果如表5‑2：

表5‑2 冷数据压缩测试结果

Tab.5‑2 Cold data compression test results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 数据量（万） | 不处理（MB） | 1百条（KB） | 1千条（KB） | 1万条（KB） | 7z压缩（KB） |
| 1 | 1.9 | 450 | 85 | 80 | 跳过 |
| 30 | 36.1 | 跳过 | 跳过 | 2415 | 1623 |

其中数据占用空间为不处理时数据在数据库中的占用空间和处理时产生的文件大小加冷数据基础信息大小。测试数据值为0-1范围内的随机数，产生时间为连续、间隔为1s的时间序列，数据和冷数据基础信息都在设备ID，物模型ID，时间上建立索引。文件大小为实际占用空间，受到操作系统最小文件大小的限制。测试发现冷数据基础信息约为200B/条，压缩后数据约为8B/条，数据约为128B/条。

从理论上进行分析，单条数据不压缩时占用的空间为记录ID（8字节）+设备ID（8字节）+物模型ID（8字节）+时间（8字节）+数据值（4字节）+索引（ID8字节+设备ID 8字节+物模型ID 8字节+时间8字节），总共68字节，而单条数据进行压缩后只包含压缩后的时间（4字节）和数据值（4字节），在数据量大的情况下可忽略冷数据基础信息占用的空间，总共8字节，差距约为8倍，但是实际上差距约为16倍，这可能是受到了Mysql本身的存储机制的影响。另一方面，使用7z压缩后，可以再节约约为30%的空间。

从结果可以看出，设备数据存放在数据库中占用的空间远高于压缩后。可见将数据压缩后存储可以极大得节约存储空间。

### 集成测试

假设有一个农户拥有两个温湿度传感器，经改造后可以接入该系统。现要将其部署在两个区域中来分别采集周围的环境数据。已知某作物，要求温度在5-25°之间，如果温度异常则要及时处理。

在此情况下对系统进行测试，测试流程如下：

1. 创建一个设备类型为温湿度采集器，物模型为温度和湿度。
2. 按照要求配置温度的预警信息，预警时间设置为1分钟。
3. 在此设备类型上创建两个设备。
4. 配置设备模拟器分别连接两个设备并模拟数据上传，其中一个设备上传不在范围内的温度。
5. 等待其中一个设备的温度异常时间超过设定值，在此期间可改变设备模拟器上传的数值。
6. 观察设备在此段时间采集的数据。

测试结果：符合预期。相关截图及介绍如下：

图5‑1为设备类型定义界面，按照需求先定义设备类型，并创建设备。



图5‑1 集成测试-物模型定义

Fig.5‑1 Integration test - thing model definition

设备信息如图5‑2所示，只展示其中一个，另一个同理。

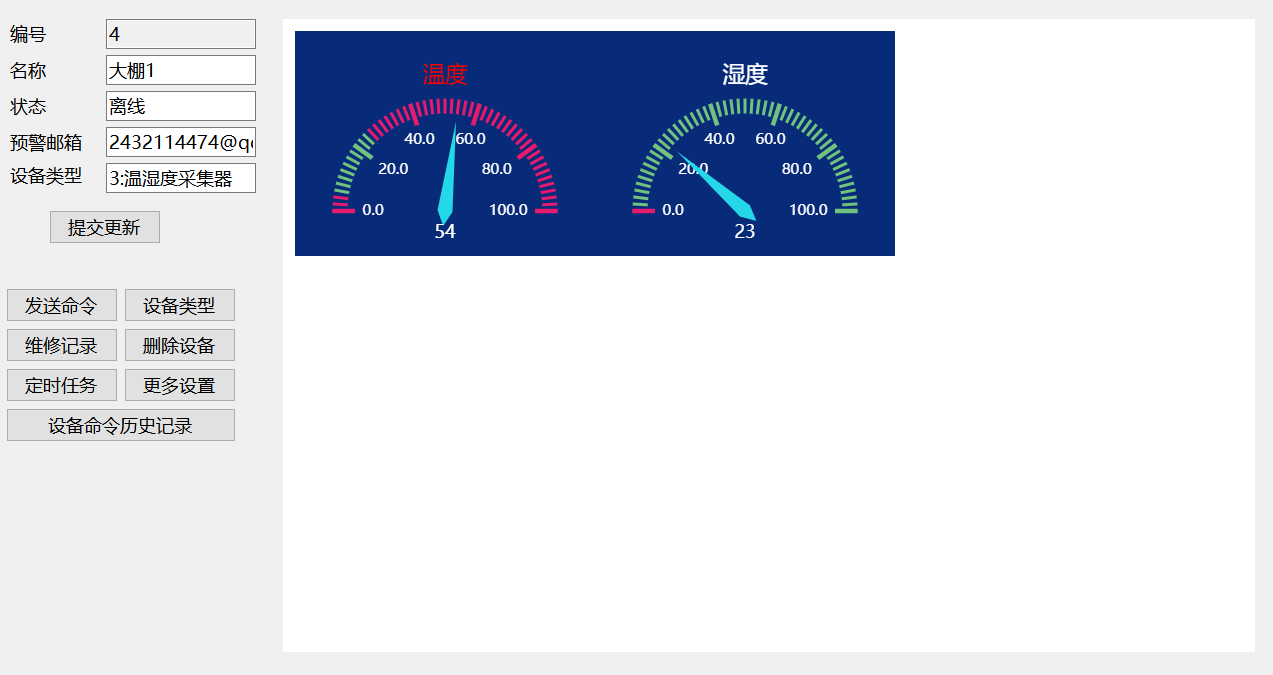


图5‑2 集成测试-设备详情

Fig.5‑2 Integration test – device details

在全部设备界面进行分组后如图5‑3，其中ID为4的设备因为接收到不在范围中的温度显示了星号作为标记，方便查看。



图5‑3 集成测试-全部设备

Fig.5‑3 Integration test – all devices

经过一段时间后查看设备采集的数据，如图5‑4所示为温度，图5‑5所示为湿度。

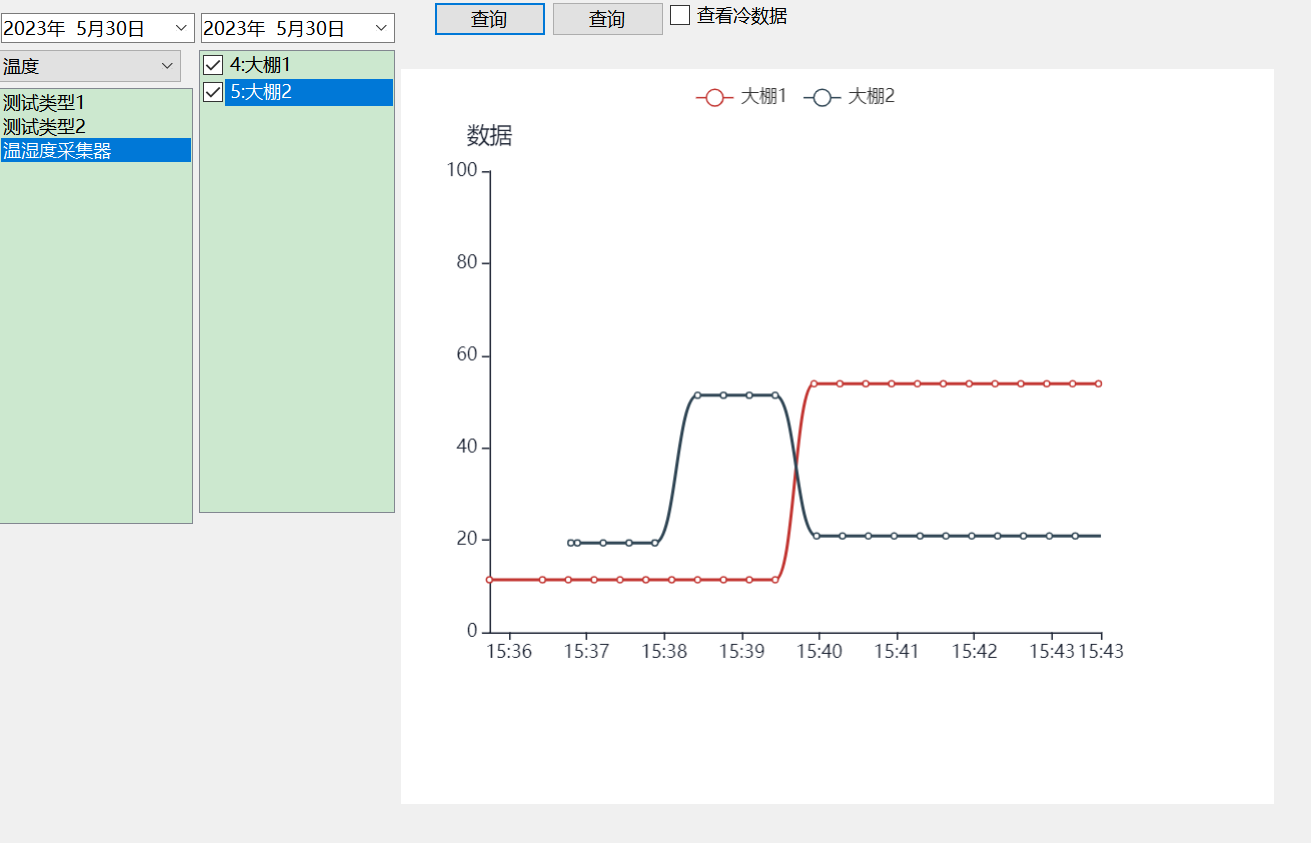


图5‑4 集成测试-温度

Fig.5‑4 Integration test – temperature

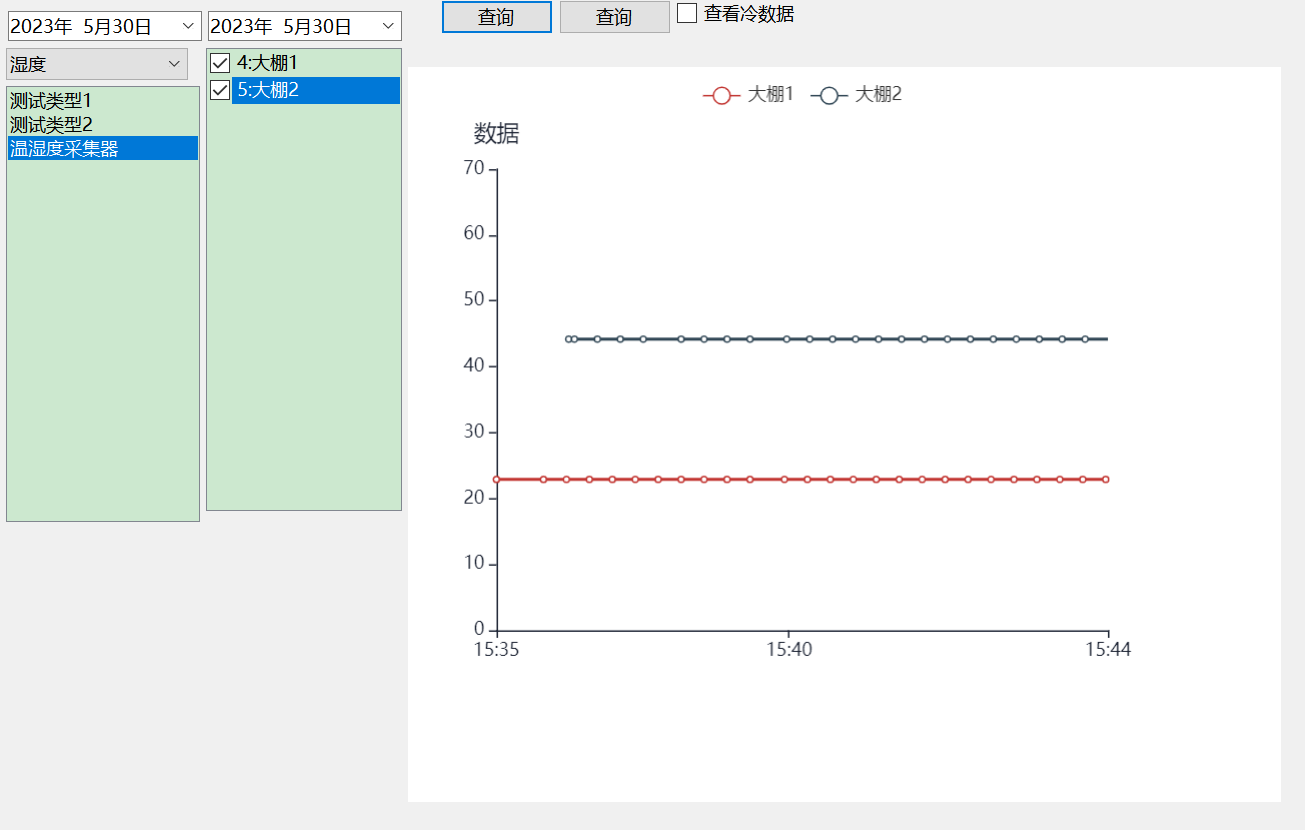


图5‑5 集成测试-湿度

Fig.5‑5 Integration test – humidity

由于其中一个设备长时间温度异常，邮箱收到了预警邮件，如图5‑6。



图5‑6 集成测试-告警邮件

Fig.5‑6 Integration test – alert email

## 局限性

由于时间原因，本系统以实现功能为主，参数校验是一个十分耗费时间的工作，因此本系统暂时没有对参数进行校验，测试也主要局限在功能上，包括功能是否与预期相符、系统是否能阻止用户越权访问。非法的参数校验、操作是否友好，不在本测试范围内。

# 总结与展望

## 总结

我国是世界重要的农业大国之一，物联网技术可以很好地解决农业数据采集和设备控制的问题。但是目前市场上还没有一个被广泛认可的作物生长环境监测系统，为此，本文设计并实现了一个作物生长环境监测系统，主要内容包括：

1.需求收集和分析。

2.进行软件设计和开发。

3.对系统进行部署和测试。

本文在以下两个方面进行了探索和实践：

1.细粒度的设备权限控制

本文对设备权限进行更细粒度的划分，用户访问设备时不仅需要设备的所有权，还要根据访问类型拥有对应的访问权限。

2.冷数据处理

系统提供冷数据处理功能，对长时间以前的数据使用其他存储方式以降低数据占用存储空间大小并减少因数据规模过大造成的数据库数据查询压力。

## 问题与展望

由于时间有限、系统功能较多，因此系统还有很多缺陷，有如下三个方面有待完善：

### 系统安全

系统没有进行参数校验，这很可能导致系统在接收到异常参数时产生错误的行为，影响系统稳定性；系统没有使用分布式锁，在并发环境下有极低的可能性出现并发问题；系统只考虑了软件方面的访问控制，并没有考虑MQTT的安全问题，应当为设备分配单独的MQTT账号和限制发送和订阅的话题。

### 系统交互

系统主要以实现功能为主，有很多功能按钮、页面布局不够合理，交互不够友好。

### 系统性能

系统对高并发的考虑欠佳，没有使用缓存和分库分表等技术，当设备量过多、查询量过大时，系统可能不能及时响应用户。

总之，系统还存在很多问题，要达到投入使用的程度，还需要很多的时间来完善功能和优化细节。在未来笔者将会把系统开源，以给其他研究者参考。

# 参考文献

1. 张容军, 段建南. 供给侧改革背景下农村闲置土地的概况与利用研究[J]. 江苏农业科学, 2017, 45(19): 39-45.
2. 贾敬敦, 鲁相洁, 黄峰, 等. 远程控制与无线通信技术在农业中的应用分析与展望[J]. 农业机械学报, 2021, 52(S1): 351-359.
3. Sisinni E, Saifullah A, Han S, et al. Industrial Internet of Things: Challenges, Opportunities, and Directions[J]. Ieee Transactions on Industrial Informatics, 2018, 14 (11): 4724-4734.
4. Gubbi J, Buyya R, Marusic S, et al. Internet of Things (IoT): A vision, architectural elements, and future directions[J]. Future Generation Computer Systems-the International Journal of Escience, 2013, 29 (7): 1645-1660.
5. Muangprathub J, Boonnam N, Kajornkasirat S, et al. IoT and agriculture data analysis for smart farm[J]. Computers and Electronics in Agriculture, 2019, 156: 467-474.
6. 陈文艺, 梁宁宁, 杨辉. 基于MQTT的物联网网关双向通信系统设计[J]. 传感器与微系统, 2022, 41(08): 100-103.
7. 孔垂跃, 陈羽, 赵乾名. 基于MQTT协议的配电物联网云边通信映射研究[J]. 电力系统保护与控制, 2021, 49(08): 168-176.
8. 马跃, 孙翱, 贾军营, 等. MQTT协议在移动互联网即时通信中的应用[J]. 计算机系统应用, 2016, 25(03): 170-176.
9. Al-Fuqaha A, Guizani M, Mohammadi M, et al. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications[J]. Ieee Communications Surveys and Tutorials, 2015, 17 (4): 2347-2376.
10. Hwang H C, Park J S, Jin G S. Design and implementation of a reliable message transmission system basedon MQTT protocol in IoT[J]. Wireless Personal Communications, 2016, 91(4): 1-13.
11. 郑博, 王煜彤, 燕钰, 等. 时间序列数据库管理：技术、系统与展望[J]. 工业技术创新, 2022, 9(04): 12-21.
12. 陈海明, 崔莉, 谢开斌. 物联网体系结构与实现方法的比较研究[J]. 计算机学报, 2013, 36(01): 168-188.
13. 于金刚, 耿云飞, 杨海波, 等. 基于MQTT协议的消息引擎服务器的设计与实现[J]. 小型微型计算机系统, 2016, 37(10): 2238-2243.
14. 李响, 李彤, 谢仲文, 等. 一种面向SaaS多租户的多层模型[J]. 计算机科学, 2017, 44(11): 56-63.
15. 李清宇. 基于物联网的智慧化工园区系统的设计与实现[D]. 济南：山东大学, 2021.
16. 赵春玲. .NET平台下开发三层架构WinForm应用程序简介[J]. 信息技术与信息化, 2010, 148(04): 33-35.
17. 陈睿哲. 中国SAAS市场发展问题探析[J]. 科学决策, 2023(01): 149-160.

# 致 谢

首先，我要感谢我的导师伍艳莲。她给了我很多指导。我的论文一开始从内容到格式都有很多问题，导师耐心地指导我，让我的论文更规范。在此期间，工作上的问题和对未来迷茫让我的心情很糟，我和导师说话时不能很好地控制情绪，也感谢导师的包容。

然后，我要感谢我的父母。感谢他们在我学习道路上的帮助。感谢父亲辛苦工作供我读书，感谢母亲在我小学、中学期间全职照顾我。

最后，我要感谢哔哩哔哩视频网站和制作编程视频的博主。我大部分有关编程的知识都是通过观看网站上的视频学习的。