边缘计算协同下基于Vue的智能家居控制系统实现

黄淑娣,黄艳贞,陈文丽,刘凤英

(广州南洋理工职业学院,广州 510900)

摘 要: 随着物联网技术的快速发展,智能家居系统在家庭生活中的应用日益广泛。然而,传统云端控制模式存在网络延迟高、数据安全性低及系统可靠性不足等问题。对此,文章提出一种基于边缘计算协同的智能家居控制系统,前端采用Vue框架实现用户交互界面,后端通过边缘计算节点实现设备数据处理与控制,实现本地快速响应和云端数据同步。通过实验验证,该系统能够有效加快设备响应速度、保障数据安全性,并支持多设备协同管理,为智能家居控制提供了一种高效、可靠的实现方案。

关键词: 智能家居;边缘计算;Vue;物联网;设备控制

中图分类号: TP399 **文献标识码:** A

Implementation of smart home control system based on Vue under edge computing collaboration

HUANG Shudi, HUANG Yanzhen, CHEN Wenli, LIU Fengying

(Guangzhou Nanyang Polytechnic College, Guangzhou 510900, China)

Abstract: With the rapid development of Internet of Things technology, smart home systems are increasingly being applied in family life. However, traditional cloud-based control models suffer from problems such as high network latency, low data security, and insufficient system reliability. This paper proposes a smart home control system based on edge computing collaboration, with the front-end using the Vue framework to implement the user interaction interface, and the back-end using edge computing nodes to achieve device data processing and control, realizing local rapid response and cloud data synchronization. Experimental verification shows that the system can effectively improve device response speed, ensure data security, and support multi-device collaborative management, providing an efficient and reliable implementation solution for smart home control.

Key words: smart home, edge computing, Vue, Internet of Things, device control

1 引言

智能家居系统可通过各种传感器、执行器和控制终端,实现对家居环境的智能化管理与控制。其中,边缘计算通过将数据处理和决策下沉至接近终端的边缘节点,可实现本地数据处理、设备协同控制和状态监测,并与云端同步,从而为智能家居系统提供低延迟、高安全性和高可靠性的解决方案[1]。基于此,本文提出一种基于Vue的智能家居控制系统,将边缘计算引入系统架构,实现设备本地控制和云端同步,以提升系统的响应速度、安全性与可靠性。

2 系统设计

2.1 系统架构

本文系统采用三层架构设计,各层职责明确、协同工作,形成从数据采集到应用服务的完整闭环。

(1)感知层由部署于家居环境中的各类智能传感器和执行器构成,包括温湿度传感器、光照传感器、门窗传感器等数据采集设备,以及灯光、空调、智能插座等控制执行设备。(2)边缘计算层部署于家庭网络内部,以树莓派或工业级网关作为边缘节点,承担设备数据的实时采集、协议解析与本地处理功能,能够在云端连接中断时独立执行本地控制逻辑,确保灯光调节、空调控制等关键操作的毫秒级响应。(3)应用层分为云端和前端两部分:云端基于Node.js与MongoDB构建,提供数据持久化存储、历史数据分析及远程控制接口;前端采用Vue框架开发,为用户提供设备状态监控、控制命令下发、历史数据可视化等交互界面[2]。

2.2 功能模块设计

系统围绕智能家居管控需求,划分为5个核心功能模块,各模块高内聚、低耦合,协同实现完整业务闭环。

(1)设备管理模块负责家居设备的全生命周期管

基金项目: 2025年广州南洋理工职业学院自然科学课题:基于Vue的智能家居控制系统(NY-2025KYYB-15)

理，支持设备注册和配网、在线状态实时查询、设备分组（如按房间或场景分组）及设备信息编辑、删除等功能。（2）控制管理模块是用户交互的核心，用户通过前端界面（Web 或 APP）发送单设备控制、场景联动或定时任务等命令，云端 RESTful API 接收命令后转发至边缘节点，由边缘节点解析协议并下发至具体执行器。（3）数据采集与分析模块由边缘节点按固定周期采集各类传感器数据，经数据清洗、格式转换和本地聚合后上传至云端；云端提供历史数据趋势分析、环境变化统计报表等功能，为节能优化与异常预警提供数据支撑。（4）安全管理模块贯穿系统各层，实现用户身份认证、设备接入鉴权、数据传输加密（TLS）及细粒度访问控制策略，防止未授权访问和数据泄露[3]。（5）告警与通知模块监测设备离线、传感器超阈值、安全事件等异常情况，触发分级告警并通过前端消息推送、微信小程序通知等方式实时告知用户，支持告警记录查询和规则配置。

2.3　技术选型

上述功能模块的实现依赖具体的技术选型，按前端、边缘节点与云端 3 个部分展开。

前端选用 Vue.js，其组件化与响应式特性便于界面模块复用与状态自动更新，结合 Vue Router 与 Vuex/Pinia 状态管理，可满足设备监控面板、历史图表展示等复杂交互需求[4]。

边缘节点以树莓派或工业级网关为载体，运行轻量级 Linux 系统与 Node.js/Python 服务，负责设备数据采集、本地控制逻辑执行及与云端的数据同步；通信协议采用 MQTT，其发布/订阅模式与低开销特性适合低带宽、不稳定的网络环境[5]。

云端基于 Node.js 与 Express 构建 RESTful API 服务层，数据库选用 MongoDB，其文档型模型与 JSON 格式匹配传感器数据异构特性，内置 TTL 索引支持时序数据自动过期。同时，可选用 Redis 缓存热点数据。

3　系统实现

3.1　前端实现

前端采用 Vue.js 框架结合 ElementUI 组件库构建用户交互界面，实现设备列表展示、控制操作、数据可视化及响应式布局四大功能。

（1）设备列表展示。前端按房间与设备类型分组渲染设备卡片，实时显示设备名称、在线状态、当前读数及心跳时间，并通过 WebSocket 与边缘节点保持长连接，设备状态变更时自动更新界面。

（2）控制操作。用户可通过开关、滑条、设定值等方式下发控制命令。前端将命令封装为 JSON 报文，经 RESTful API 发送至边缘节点，并根据返回结果更新界面或提示错误。

（3）数据可视化。前端采用 ECharts 绘制温度、湿度及能耗等环境数据趋势图，支持缩放与数值悬停等交互，帮助用户掌握环境波动规律。其中，能耗数据的累计值可通过数值积分近似计算，设采样时间间隔为 Δt（min），各时刻瞬时功率为 $P(t_i)$（W），则时段 $[t_0, t_n]$ 内的总能耗 E（kW·h）可表示为：

$$E=\frac{1}{60\,000}\sum_{i=1}^{n}P(t_i)\cdot\Delta t \tag{1}$$

（4）响应式布局。在布局层面，项目使用 postcss-px-to-viewport 插件将设计稿中的像素单位自动转换为视口单位（vw/vh），配合 ElementUI 响应式断点，使界面在手机竖屏、平板横屏及桌面宽屏下均能自适应排版；在交互层面，移动端优化包括增大触摸热区、简化导航结构及手势滑动支持，确保用户在手机 APP 内嵌 WebView 或独立浏览器中均获得流畅的操作体验。

3.2　边缘节点实现

边缘节点基于树莓派或工业级网关硬件，运行轻量级 Linux 系统，采用 Node.js 或 Python 编写服务程序，通过 MQTT 协议与云端及本地传感器通信，承担数据采集、处理、命令转发及本地缓存四大职责。

（1）数据采集模块：使用 MQTT 客户端库订阅固定主题。当温湿度、光照等传感器以 1 Hz 频率发布数据时，边缘节点触发回调函数，解析 JSON 载荷中的设备 ID、数值及时间戳，存入内存中的最近状态映射表。为保证采集可靠性，节点配置 MQTT 的 QoS 级别为 1，确保消息至少被接收一次，并在断线重连后自动恢复订阅。

（2）数据处理模块：节点对原始数据执行异常检测、统计聚合与事件判断。异常检测采用阈值与变化率双重校验；统计聚合在滑动时间窗口内计算均值、最大值等指标；事件判断通过内置规则引擎实现本地自动控制。设当前温度为 T_{curr}，空调状态为 S_{ac}（1 表示自动模式），触发阈值 $T_{\text{th}}=26$，则开启制冷的触发逻辑为：

$$\text{Trigger}=1_{\{T_{\text{curr}}>T_{\text{th}}\}}\cdot 1_{\{S_{\text{ac}}=1\}} \tag{2}$$

（3）命令转发模块：接收来自前端的 HTTP 请求，将请求转换为符合设备协议的 MQTT 消息，发布至 devices/[device_id]/command 主题。若设备为 Zigbee 或红外控制类型，节点需通过外接射频或红外发射模块进行协议转换，如将 JSON 命令 {"cmd":"power_on"} 转化为二进制编码的红外脉冲序列。

（4）本地缓存与同步模块：节点将采集数据写入本地数据库，采用 FIFO 策略淘汰旧记录，并定时将未同步数据批量上传至云端。网络中断时持续本地缓存，恢复后自动续传，保证数据完整性与可追溯性。

3.3　云端与数据管理

云端基于 Node.js 与 Express 构建 RESTful API 服务层，采用 MongoDB 作为持久化数据库，并可选配 Redis 缓存热点数据，主要负责数据存储、远程控制与管理、数据分析与可视化三大职能。

（1）数据存储模块：设计 sensor_data 存储传感器历史记录（包含设备 ID、数值类型、时间戳、数值），device_logs 记录设备上下线、命令执行结果等操作日志，user_actions 保存用户登录、配置修改等行为轨迹。针对传感器数据的时序查询特性，在 MongoDB 中建立复合索引（device_id+timestamp）以加速范围检索。为控制存储成本，采用 TTL 索引自动删除超过 90 天的原始采样数据，同时保留按小时聚合的统计记录。对于落在同一小时窗口内的原始数据序列 $\{x_1, x_2, \cdots, x_n\}$，小时均值 $\dot{x}_{\text{hour}}$ 与方差 s^2 可通过预计算公式实现：

$$\dot{x}_{\text{hour}}=\frac{1}{n}\sum_{i=1}^{n}x_i,\quad s^2=\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\dot{x}_{\text{hour}})^2 \tag{3}$$

（2）远程控制与管理模块：提供统一的鉴权与路由中间件。用户通过前端登录后获取 JWT 令牌，后续请求需在 HTTP 头部携带该令牌。云端 API 首先校验令牌有效性及用户对目标设备的访问权限（基于设备-用户绑定表），随后将控制命令转发至对应家庭的边缘节点。若边缘节点不在线，云端将命令存入命令队列（Redis List），待节点上线后通过 MQTT 保留消息或按需拉取方式补发。

（3）数据分析与可视化模块：基于聚合数据提供多维度洞察。能耗优化分析中，云端计算各设备或房间的能耗占比，并和历史同期（如上月、去年同期）比较，识别异常耗电设备。使用建议生成方面，云端运行轻量级规则模型，如检测到“用户离家模式下空调仍持续运行超过 30 min”则推送节能提醒。所有分析结果支持 PDF 或 CSV 导出，满足用户数据归档与进一步处理需求。

4 系统测试与结果分析

4.1 实验环境

实验在智能家居模拟环境中搭建，硬件设备包括树莓派 4B 作为边缘节点，一台配备主流浏览器的 PC 及两部不同品牌智能手机作为前端终端，用于模拟用户远程控制与本地访问场景。受控设备包括可调光智能灯泡、支持通断控制的智能插座及温湿度传感器。所有设备通过同一家庭无线路由器接入局域网，边缘节点通过 MQTT 协议与设备通信，同时通过互联网与云端服务器保持连接。环境布置上，将传感器置于房间不同位置以模拟真实分布，确保测试数据覆盖信号强度与通信距离的多种组合。实验前对所有设备进行配网注册，并配置边缘节点的基础规则（如温度超限自动关灯），为后续指标测试提供稳定、可重复的运行环境。

4.2 测试指标

实验选取 3 个核心指标评估系统性能。响应延迟定义为用户在前端界面点击控制按钮（如关闭灯泡）到设备状态实际发生变化所经历的总时间，该指标直接反映用户的操作感知体验。数据同步延迟指传感器数据从边缘节点产生到成功写入云端数据库的时间差，用以衡量系统在数据汇集和远程监控方面的实时能力。系统可靠性测试重点关注网络中断场景：人为断开边缘节点的互联网连接，观察本地控制功能可否正常执行，以及网络恢复后数据能否自动续传。

4.3 实验结果

实验数据显示，相较于传统纯云端控制架构下约 500 ms 的平均延迟，系统平均响应延迟约为 120 ms，性能提升超过 70%。用户操作时几乎感觉不到等待，实现了近似本地直控的流畅体验。数据同步延迟平均为 2 s，即边缘节点采集的温湿度读数可在 2 s 内出现在云端历史记录中，基本满足远程监控的准实时需求，用户打开手机查看环境数据时看到的是数秒前的状态。网络中断测试中，拔掉路由器外网网线后，前端界面虽提示云端连接失效，但用户通过局域网 IP 访问边缘节点提供的本地 Web 服务时，仍可正常控制灯泡开关和调节亮度，说明边缘节点的独立运行机制有效。重新接通网络后，离线期间产生的约 300 条传感器数据自动补传至云端，无数据丢失。

5 结束语

本文设计并实现了一种基于 Vue 的智能家居控制系统，结合边缘计算实现本地快速响应与云端数据管理。实验结果表明，该系统在响应速度、数据安全性及系统可靠性方面均优于传统云端控制模式。未来可进一步引入边缘 AI 技术，提升智能家居的自动化和智能化水平，为智能家居系统的实际应用提供参考。

参考文献：

[1] 杨永强，张思，刘智敏，等. 基于 ZigBee 和 WiFi 的智能语音家居控制系统[J]. 计算机科学与应用，2024，14(1)：41-53.

[2] 冷宇. 基于 ESP32 的远程家居控制系统设计[J]. 无线互联科技，2022，19(24)：86-88.

[3] 刘一帆，田生丁，姜纪煌，等. 智能家居控制系统研究[J]. 电子制作，2026，34(2)：36-41.

[4] 丁常松，丁庆松，张广学，等. 电子技术在智能家居灯光控制系统中的应用[J]. 消费电子，2026(1)：215-217.

[5] 李乾赫. 智能家居环境下的机电一体化控制系统的集成与优化设计[C]//2024 精益数字化创新大会，2024：1-4.

作者简介：

黄淑娣（1983—），硕士，讲师，研究方向：Web 前端开发、软件应用开发。