WISE 监控系统总体设计

1.引言

随着中国制造 2025、智慧城市等理念的不断深入，使得工业领域企业对设备互联、集中管理、远程监控等数据处理需求愈加明显。本系统基于 Wise 设备，采用 Restful API 接口。实现了数据的采集、可视化展示以及物体控制等功能。本文档从系统的网络结构，通信方式，系统设计，模块设计等方面完整的介绍了系统的设计与实现。对后续的应用以及二次开发具有指导意义。

2.项目背景

当今世界，信息化的热潮已逐步蔓延到工作、生活、经济、政治、文化等各个角落，大数据的运用正飞速改变着我们的生活。

在蔬菜大棚内，无需人工操作，即可自动调节光照、温度、湿度等要素，为农作物高产、优质、高效创造最适宜的条件；在图书馆，无需刻意收放卷帘，也无需定时开关照明灯，无论昼夜，室内光线始终保持最佳状态；在生产车间，无需专门配置安全生产管理人员，即可清晰掌控机床运行、设备温度、管线压力、危险气体等信息。一旦数据超过阈值，报警装置则会启动，让管理者及时智能处理；

以上，不是美国大片中描绘的未来世界，它们已经或正在逐渐走进我们的日常生活……

近年来，随着中国制造 2025、智慧城市等理念的不断深入，使得工业领域企业对设备互联、集中管理、远程监控等数据处理需求愈加明显。越来越多的企业意识到数据管理的规范化、安全化及基于大数据的挖据分析实乃大势所趋。

为顺应广大工业物联网客户需求，研华 WISE-4000 系列产品应运而生，可满足数据采集工作无线化、低功耗、低成本等应用，同时可实现移动终端访问、智能化数据上传等功能。

3.系统环境

Flask 是一个 Python 微型框架。Flask 是一个使用 Python 编写的轻量级 Web 应用框架。基于 Werkzeug WSGI 工具箱和 Jinja2 模板引擎。 Flask 使用 BSD 授权。 Flask 也被称为“microframework”，因为它使用简单的核心，用 extension 增加其他功能。Flask 没有默认使用的数据库、窗体验证工具。然而，Flask 保留了扩增的弹性，可以用 Flask-extension 加入这些功能：ORM、窗体验证工具、文件上传、各种开放式身份验证技术。

axios 是基于 Promise 的 HTTP 请求客户端，可同时在浏览器和 node.js 中使用。Axios 的特点，特点：从浏览器中创建 XMLHttpRequests，从 node.js 创建 http 请求，支持 Promise API，拦截请求和响应，转换请求数据和响应数据
取消请求，自动转换 JSON 数据。

WeUI 是一个样式库，核心文件就是 weui.css，WeUI 是微信官方设计团队为微信 Web 开发量身打造的一个 UI 样式库，你可以把它理解为一个前端框架，类似于 Bootstrap。模块： 目前包含 12 个模块 (Button, Cell, Toast, Dialog, Progress, Msg, Article, ActionSheet, Icons, Panel, Tab, SearchBar).

4.网络结构

系统主要由 Wise 设备， 服务主机和无线热点三部分组成。Wise 设备，服务主机以及客户端在同一内网中。下图是系统的网络结构图：

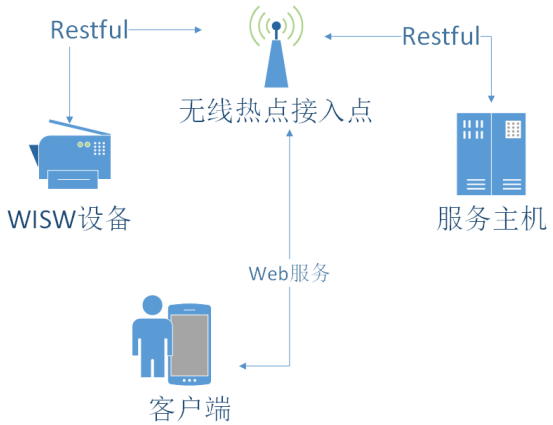

图 1 系统的网络结构图

如上图所示。服务主机通过 Restful 接口与 Wise 设备进行通信，从而达到设置 Wise 设备的目的。服务器请求方式有三种，分别为 GET，POST， PATCH：

GET：从服务端获取数据。

POST：向服务端提交数据，更新所有数据。

PATCH：向服务端提交数据，更新指定数据。

调用 API 时，需要附带权限信息。权限信息为 Wise 设备指定的用户密码加密后的字符串。加密的方式为 “用户名:密码”字符串经过 base64 加密， 加上 “base ” 前缀构成的最终字符串。

用户通过浏览器访问 Web 页面的方式去查看 Wise 设备的实时信息，以及对设备进行相应的设置。

5. 网络通信

下图是网络通信时的序列图。

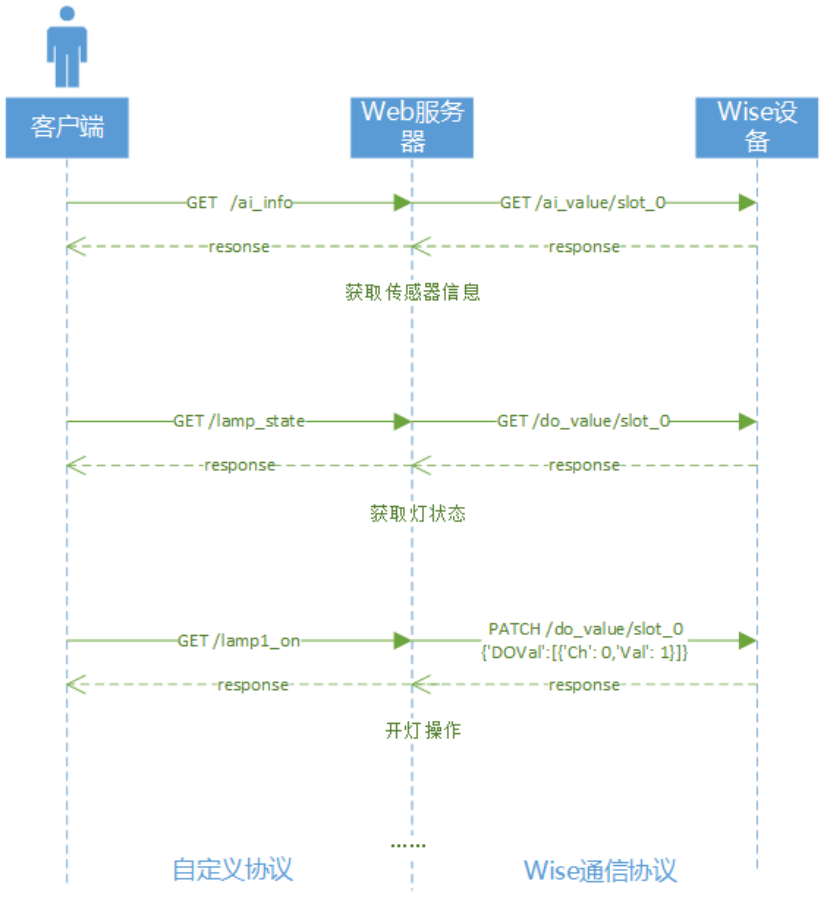

图 2 网络通信序列图

如图 2 所示，系统操作过程中，由服务器转发客户端所有请求并提交到具体的 Wise 设备。完成通信。

5.系统结构

下图是系统的系统结构图。

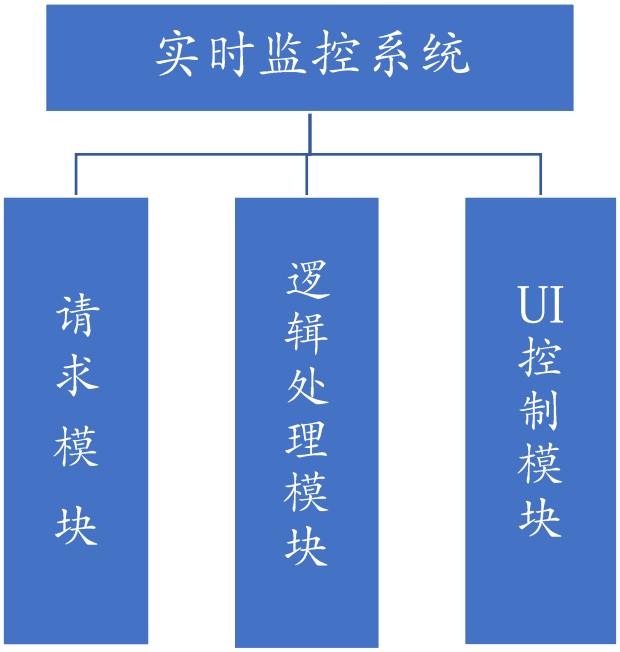

图 2 系统的功能模块

如图 2 所示，系统主要由请求模块，逻辑处理模块，UI 模块构成。

请求模块是对 http request 的二次封装，加入了用户名，密码的身份认证信息。根据不同的请求参数打包成对应的请求获得应答返回上层。

逻辑处理模块主要封装了不同的请求所对应的具体请求参数，并用函数包装为具体动作。该模块根据前端传到的具体参数调用不同的动作，在底层形成具体的请求得到应答并返回。

UI 控制模块解析好后台返回的数据后，格式化为前端 Ecahrt 控件所适应的数据格式，并更新前段 UI。

6. 功能模块

本系统主要分为对 Wise 数据的获取以及对 Wisw 的设置两个功能。通过获取 Wise 信息，我们可以得到传感器的实时数据，在前端中用折线图的形式动态表示。对于 Wise 的设置。通过提交信息，我们可以控制灯的亮灭。

7.核心代码

下面是封装后的 Wise 请求核心代码。

```
class WiseRequest:

    def __init__(self, ip, account, password):
        self.sess = requests.session()
        self.headers = config.HEADERS
        self.ip = ip
        self.account = account
        self.password = password
        self.init_headers()

    def get_auth(self):
        step1 = '%s:%s'%(self.account, self.password)
        step2 = base64.b64encode(step1.encode('utf-8'))
        return 'Basic %s'%(str(step2, encoding='utf-8'))

    def init_headers(self):
        self.headers['Authorization'] = self.get_auth()

    @str2dict
    def request(self, url, data, method='patch'):
        data = json.dumps(data)
        if method == 'patch':
            json_str = requests.patch(url%self.ip, data=data,
headers=self.headers)
        elif method == 'get':
            json_str = requests.get(url%self.ip, data=data,
headers=self.headers)
        elif method == 'post':
            json_str = requests.post(url % self.ip, data=data,
headers=self.headers)
        else:
            return {'success': False, 'msg': 'method is not define,
method: %s'%(method)}
        return json_str.text
```

对于 Wise 来说,每个 IP，用户名和密码唯一确定一个 Wise 设备并拥有完全控制权限。一个无线传感网可能有多个 Wise 设备。

8.界面设计

8.1 开关设置

下图是控制灯的亮灭的界面图。

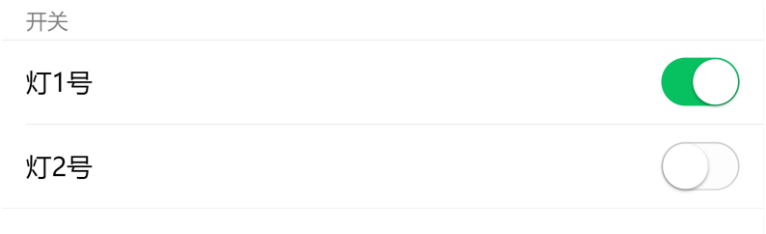

图 4 灯控制界面

如图所示，系统会每隔 0.25s 向后台实时获取灯的状态并控制前端的 checkbox 是否变绿。变绿代表亮，变灰代表暗。点击按钮可以控制灯的亮灭。

8.2 数据监控

下图是实时监控折线图。

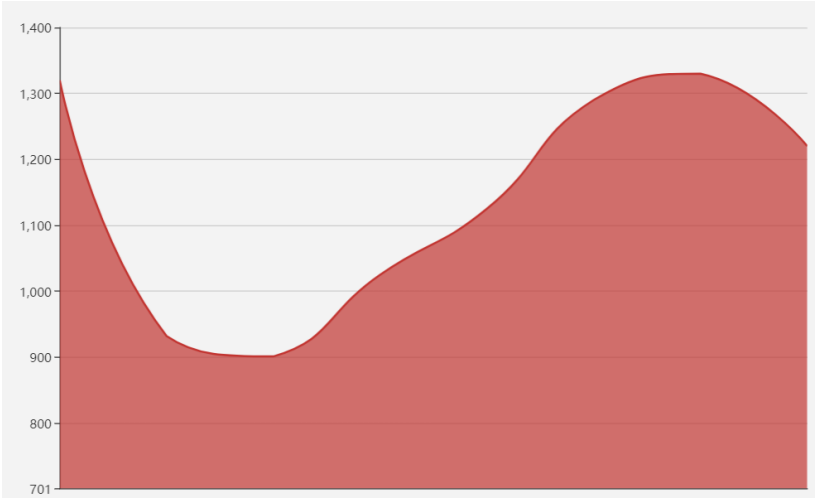

图 5 实时监控

系统每隔 0.25s 向后台获取最新的数据，并将数据以可视化的形式展现出来。

9.总结

针对无线传感网路中数据采集功耗高，效率低的问题，本系统基于 Wise 设备进行二次开发与实现。该系统采用第三方服务器作为主控管理节点，Restful API 通信。合理有效的利用资源。完成了数据的采集与处理以及最终的可视化展示。对后续的开发具有指导意义。