GREEN BELT

基于大数据和物联网的可视化城市绿化带自动灌溉系统

A Visualized Urban Greenbelt Automatic Irrigation System Based on Big Data and Internet of Things

队伍：笔架山

成员：孙达明、李亮

指导教师：马征

目录

一、 项目背景

随着数字化时代的到来，城市绿化建设成为了人们和政府的共同关注点，但城市绿化管理也面临着成本高、效率低、难以覆盖全面等问题。为了解决这些问题，本项目利用互联网、物联网、大数据等技术，开发了一款创新的城市绿化带管理系统——**GreenBelt**。

城市绿化是提升城市生态环境和人居品质的重要措施，但城市绿化也面临着许多困难和挑战。城市绿化涉及到多种植物、多种生态功能、多种管理要求，因此具有复杂多样的特点。然而，城市绿化管理却往往缺乏科学规划、专业技术和有效监督，导致很多绿化带因管理不善而发生了枯萎的现象。这不仅影响了城市的美观和舒适度，还造成了资源的浪费和环境的恶化。因此，加强城市绿化管理，提高城市绿化的质量和效益，是城市建设和发展的迫切需要。

城市绿化带不好管理，是一个普遍存在的问题。由于城市绿化带的分布广泛、种类繁多、需求多变，传统的人工管理方式难以满足其

高效、精准、节能的要求。为了解决这个问题，亟需一种采用物联网和大数据的自动化管理系统进行管理，来解决这种现象。这种系统可以通过安装在绿化带上的各种传感器和控制器，实时收集和分析绿化带的环境数据和生长状态，根据预设的规则和算法，自动调节绿化带的灌溉、施肥、病虫防治等管理措施，实现对绿化带的智能化、精细化、节能化的管理。这样不仅可以提高城市绿化带的生态效益和美观度，还可以降低城市绿化带的管理成本和资源消耗。

城市绿化带的管理是一项复杂而繁琐的工作，需要投入大量的人力资源和各种资源，如水、肥料、农药等。这些资源的使用不仅增加了城市绿化带的管理成本，还可能造成环境污染和资源浪费。采用物联网和大数据的自动化管理系统进行管理城市绿化带，可以有效地解决这些问题。这种系统可以根据绿化带的实际情况，自动调节管理措施，避免过度或不足的使用资源，从而达到节水、节肥、节药的目的。同时，这种系统也可以减少人工干预和巡查的次数和范围，降低人力资源的需求和风险。综上所述，采用物联网和大数据的自动化管理系统进行管理城市绿化带，会大大减少成本和人力资源和各种资源。

我们研发的基于大数据、物联网、分布式绿化带管理系统能够很好地解决全国甚至全球城市绿化带复杂、难以管理的情况。我们的系统叫做 GreenBelt，它可以通过物联网硬件设备进行实时的实地状态的读取，之后有一个庞大的分布式可拓展的后端获得各个设备的状态和管理设备，并且可以自动操作物联网设备上的可控组件实现自动浇水施肥等功能。我们的系统的优势是价格低廉，可行性高，可以减少

人力成本。它可以用在全国或者全世界的绿化带，实现绿化带智能化、精准化管理，得益于分布式的后端结构，它的服务性能有着很好的可拓展性。我们的系统不仅可以提高城市绿化带的生态效益和社会效益，改善城市空气质量和居民生活环境，还可以借鉴其他领域的智能化管理经验，如住宅、园林、交通等，为城市建设和发展提供更多的可能性和创新。

二、 项目实施

概括

针对数量庞大的城市绿化带的管理问题，我们的项目的重点在于设备集群的可拓展性和管理服务器性能的可拓展性和稳定性。

我们的物联网设备上的可控组件可以由后端的轮询设备检测状态并且自动的触发，可控设备可以是继电器，小水泵，施肥装置等，为多种管理环境提供了可能。在轮询设备的功能上我们采用了一种基于 Java 多线程底层的队列思想和 go 程信道的特性进行，可行性、效率、安全性、稳定性得到了保证。

我们也提供了前端实时可视化的界面，让管理人员更

实施背景

随着城市化进程的加快，城市绿化带在提升城市生态环境、美化城市风貌、改善城市气候等方面发挥着重要作用。然而，城市绿化带的数量庞大、分布广泛、管理复杂等特点给城市绿化管理带来了巨大

挑战。传统的人工管理方式效率低下、成本高昂、难以保证绿化质量。

因此，我们需要利用物联网技术对城市绿化带进行智能化管理，实现远程监控、自动控制、精细化管理等功能。

实施目标

我们的项目旨在开发一套基于物联网技术的城市绿化带智能管理系统，通过在绿化带上部署各种传感器和可控设备，实现对绿化带的实时数据采集、分析、处理和反馈。我们的项目目标包括：

(1) 提高城市绿化带管理效率和质量，降低管理成本和风险。

(2) 提高城市绿化带设备集群的可拓展性和管理服务器性能的可拓展性和稳定性。

(3) 提供多平台的控制监控端软件，满足不同管理人员的使用需求。

(4) 提供多种可控设备，适应不同绿化环境的管理需求。

我们将通过以下指标来衡量和评估我们的项目成功：

(1) 系统可用性：系统正常运行时间占总时间比例。

(2) 系统性能：系统响应时间、并发数、吞吐量等指标。

(3) 系统功能：系统提供的功能是否完善、易用、准确。

(4) 系统安全：系统是否具有足够的安全保障机制，防止数据泄露、篡改等风险。

实施方案

我们的项目方案主要包括以下几个部分：

1. 物联网设备：我们在城市绿化带上部署各种传感器和可控设备，如温湿度传感器、光照传感器、土壤水分传感器、继电器、小水泵、施肥装置等，实现对绿化带的实时数据采集和自动控制。我们的物联网设备上的可控组件可以由后端的轮询设备检测状态并且自动的触发，可控设备可以是继电器，小水泵，施肥装置等，为多种管理环境提供了可能。在轮询设备的功能上我们采用了一种基于 Java 多线程底层的队列思想和 go 程信道的特性进行，可行性、效率、安全性、稳定性得到了保证。

2. 后端服务：我们采用了后端功能性分层的架构，将后端服务按照功能划分为不同的层次，如数据层、业务层、控制层等，提高了代码的可读性和可维护性。我们还采用了单功能集群分布式的部署方式，将同一功能的服务部署在多台服务器上，形成一个集群，提高了服务的可用性和负载均衡能力。我们使用 Nacos 作为注册中心，管理各个集群中大量的设备地址，简化了设备的注册和发现过程，也方便了设备的添加和删除。我们使用 redis 缓存技术实现信息读取的高并发，并且考虑到了缓存击穿的情况，一旦缓存击穿我们就会向 go 缓存流转中提交数显状态的请求，并获取当前的设备状态。

3. 前端软件：我们考虑到管理人员使用的设备可能会多种多样，于是我们尽力解决掉一切平台请求的跨域问题，并且开发了多平台的控制监控端软件，包括 web 网页端，APP 端，小程序端。其中：

(1) Web 端可以对集群中众多设备进行统一的编辑和管理，实现了对整个城市绿化带的全面控制。

(2) APP 和小程序端是针对单台物联网设备进行操作，可用于故障排查和单设备特殊控制，实现了对每个绿化带的精细化管理。

我们的项目方案具有以下优势：

(1) 可拓展性强：我们的项目方案可以根据城市绿化带数量和规模进行灵活的扩展，只需要增加或减少物联网设备和后端服务的数量，就可以实现对不同大小的城市绿化带的智能管理。

(2) 节能环保：我们的项目方案可以根据绿化带的实际情况，自动调节水量、光照、肥料等参数，实现对植物的最佳生长条件的保障，同时也节省了资源和成本，减少了浪费和污染。

(3) 智能高效：我们的项目方案可以实现对城市绿化带的远程监控和控制，提高了管理人员的工作效率和便利性，也减少了人为的误操作和失误，提升了城市绿化带的管理水平和品质。

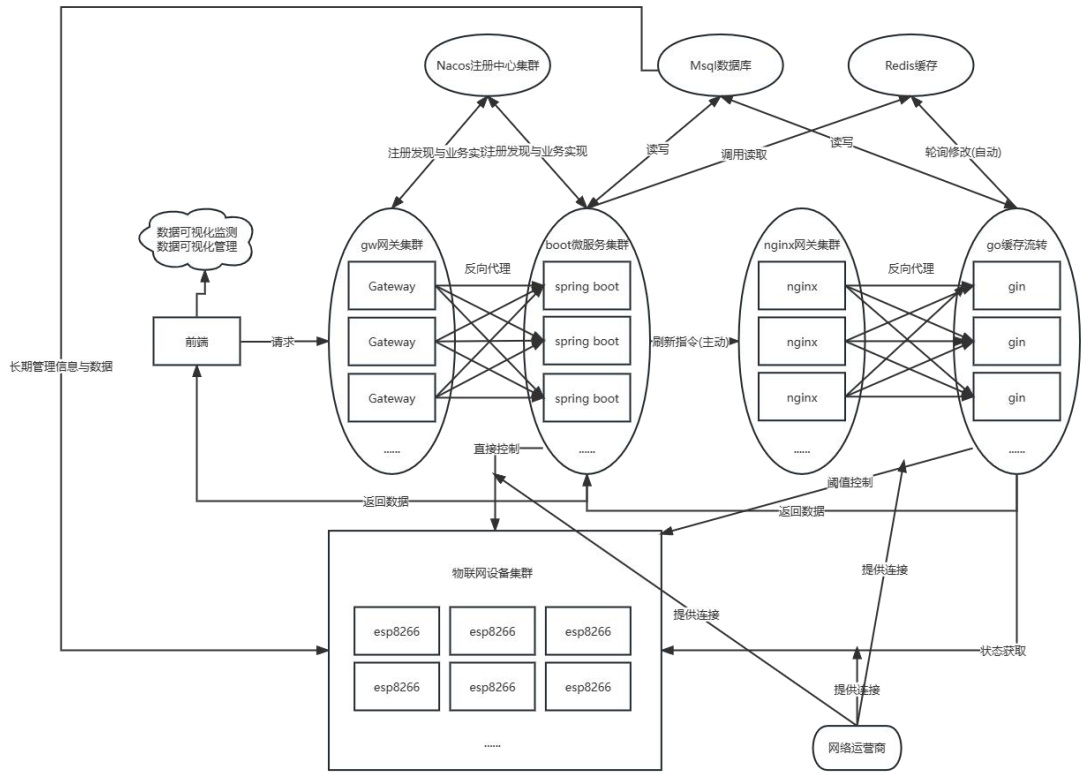
我们的项目方案参考了以下案例：

(1) 浅谈智慧城市之移动物联网建设：该文章介绍了移动物联网在智慧城市建设中的重要作用和应用场景，如智慧交通、智慧环境、智慧安防等，为我们提供了一些思路和启发。

(2) 物联网在智慧城市管理中的深度应用：该文章分析了物联网技术在智慧城市管理中的发展与深度应用，如物联网体系架构、物联网数据处理、物联网应用平台等，为我们提供了一些技术指导和参考。

(3) 舟山市城市管理物联网专项规划（2021-2025）：该文件是舟山市制定的关于城市管理物联网建设的专项规划，涉及了物联网建设目标、重点领域、主要任务、保障措施等内容，为我们提供了一些政策支持和借鉴。

项目架构图



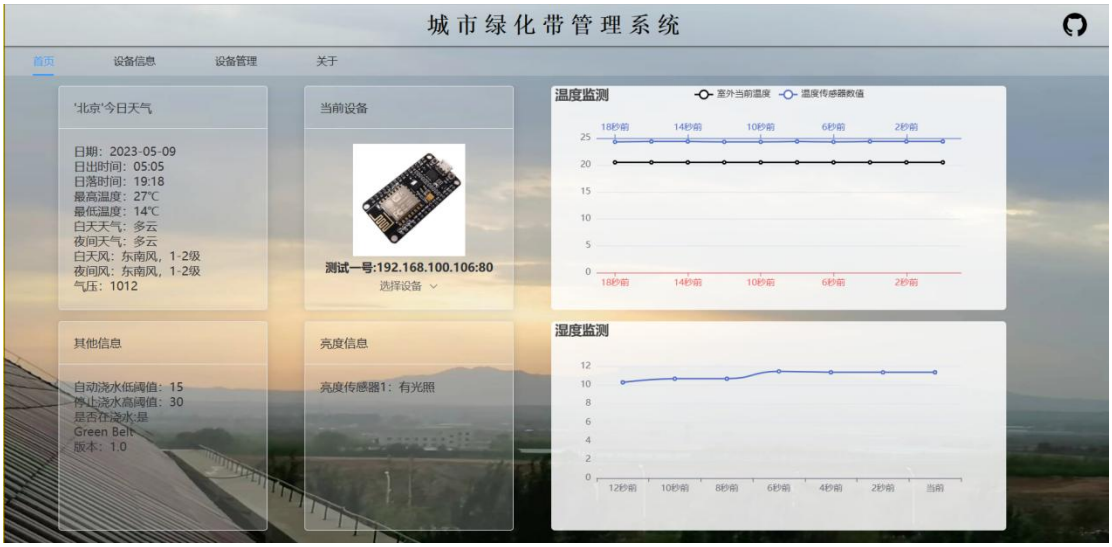
三、 项目展示

网页前端界面

用户登录

用户注册

设备动态可视化



设备信息

设备管理

APP/小程序前端界面

APP 界面

图 1　APP 登录

图 2　APP 管理

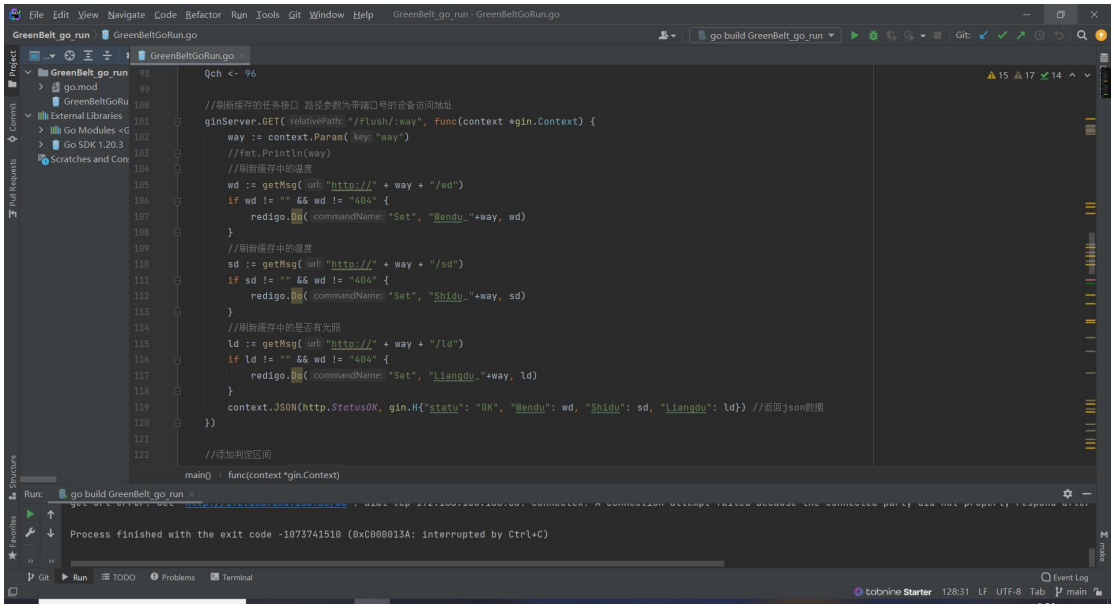
小程序界面

图 1　小程序登录

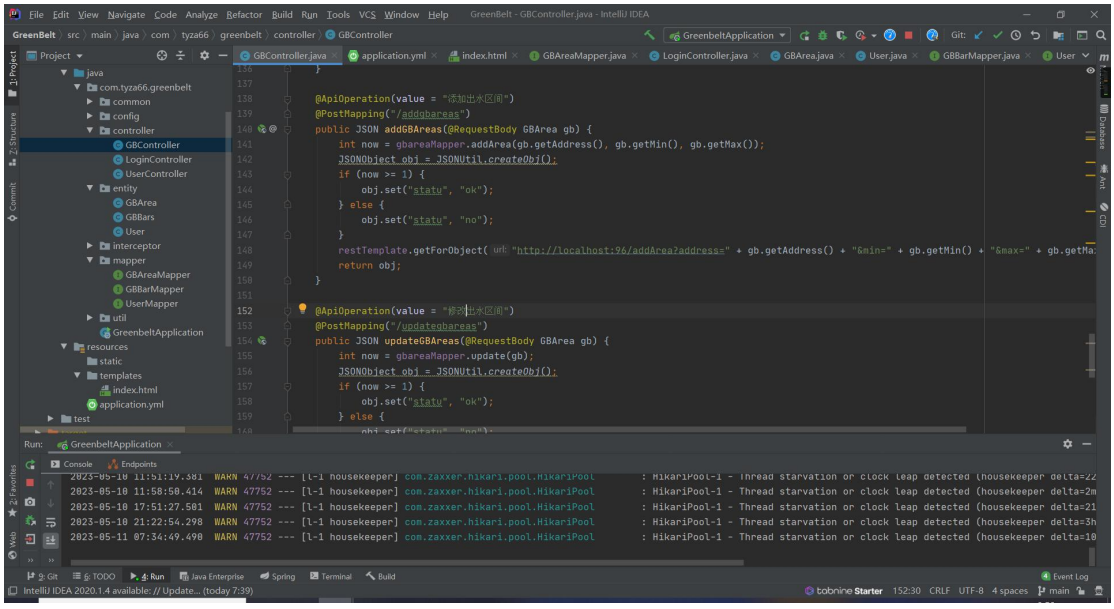
图 2　小程序管理

后端部分

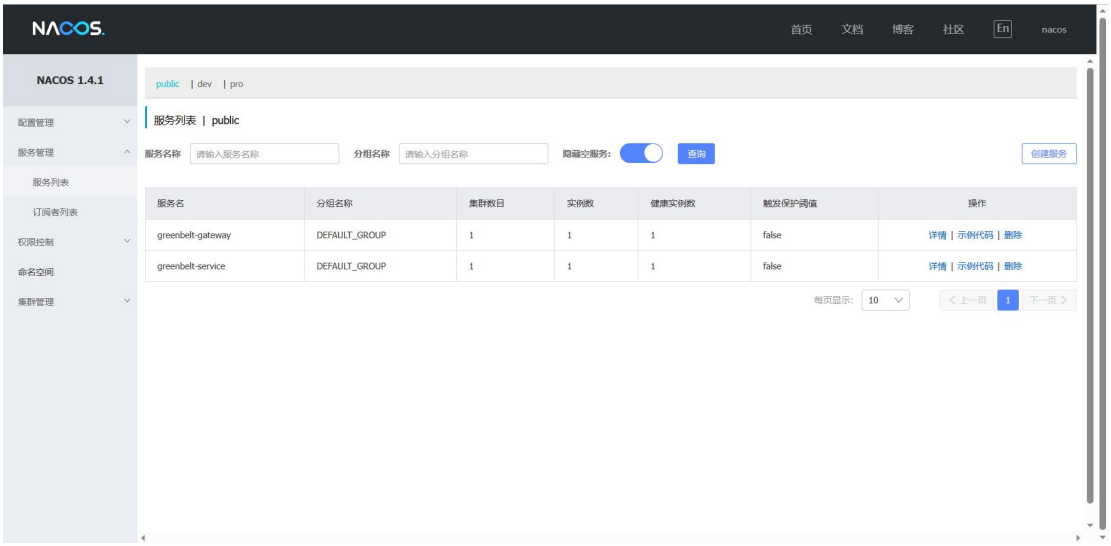
Go 缓存流转



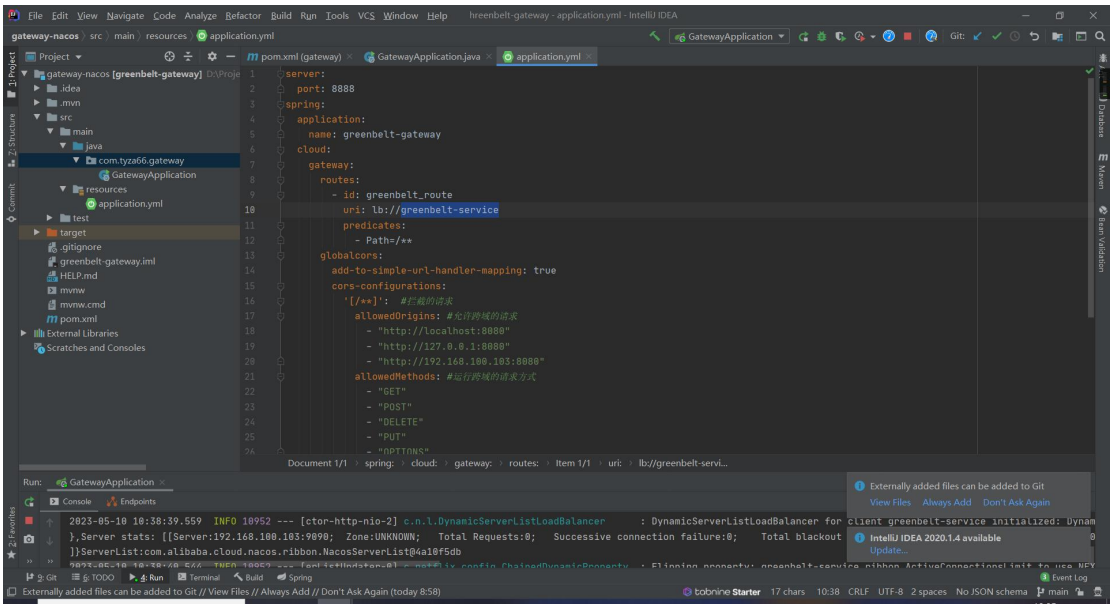
Boot 集群单机



注册中心

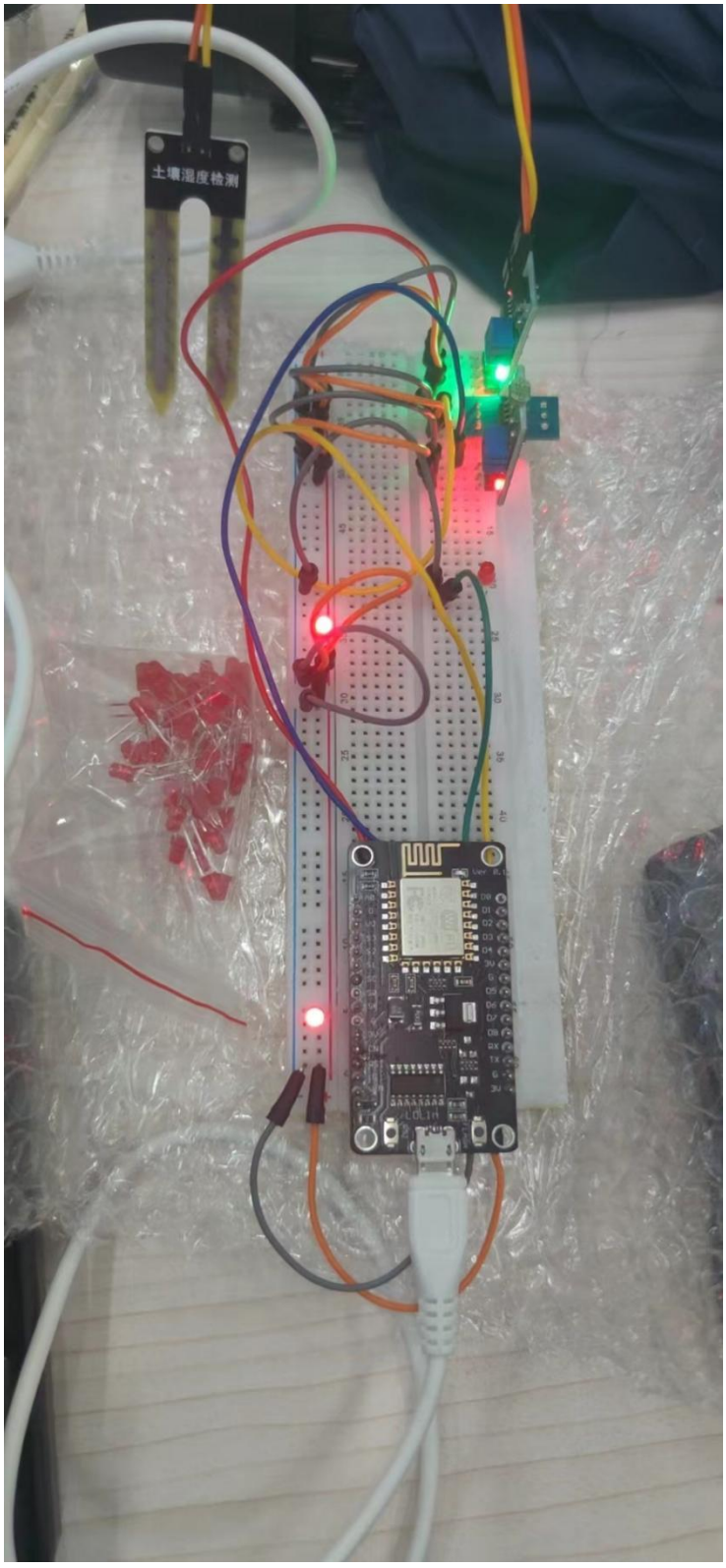

NACOS
首页
文档
博客
社区
En
nacos
NACOS 1.4.1
public | dev | pro
配置管理
服务管理
服务列表
订阅者列表
权限控制
命名空间
集群管理
服务列表 | public
服务名称
请输入服务名称
分组名称
请输入分组名称
隐藏空服务:
查询
创建服务
服务名
分组名称
集群数目
实例数
健康实例数
触发保护阈值
操作
greenbelt-gateway
DEFAULT_GROUP
1
1
1
false
详情 | 示例代码 | 删除
greenbelt-service
DEFAULT_GROUP
1
1
1
false
详情 | 示例代码 | 删除
每页显示:
10

网关


gateway-nacos
src
main
resources
application.yml
GatewayApplication
pom.xml (gateway)
GatewayApplication.java
application.yml
server:
port: 8888
spring:
application:
name: greenbelt-gateway
cloud:
gateway:
routes:
- id: greenbelt_route
uri: lb://greenbelt-service
predicates:
- Path=/**
globalcors:
add-to-simple-url-handler-mapping: true
cors-configurations:
'[/**]': #拦截的请求
allowedOrigins: #允许跨域的请求
- "http://localhost:8888"
- "http://127.0.0.1:8888"
- "http://192.168.100.103:8888"
allowedMethods: #运行跨域的请求方式
- "GET"
- "POST"
- "DELETE"
- "PUT"
- "OPTIONS"
Run:
GatewayApplication
Console
Endpoints
2023-05-10 10:38:39.559 INFO 10952 --- [ctor-http-nio-2] c.n.l.DynamicServerListLoadBalancer : DynamicServerListLoadBalancer for client greenbelt-service initialized: Dynam
},Server stats: [[Server:192.168.100.103:9898; Zone:UNKNOWN; Total Requests:0; Successive connection failure:0; Total blackout
]}ServerList:com.alibaba.cloud.nacos.ribbon.NacosServerList@4a10f5db
Externally added files can be added to Git
View Files
Always Add
Don't Ask Again
IntelliJ IDEA 2020.1.4 available
Update...

物联网设备部分

面包板工程开发板



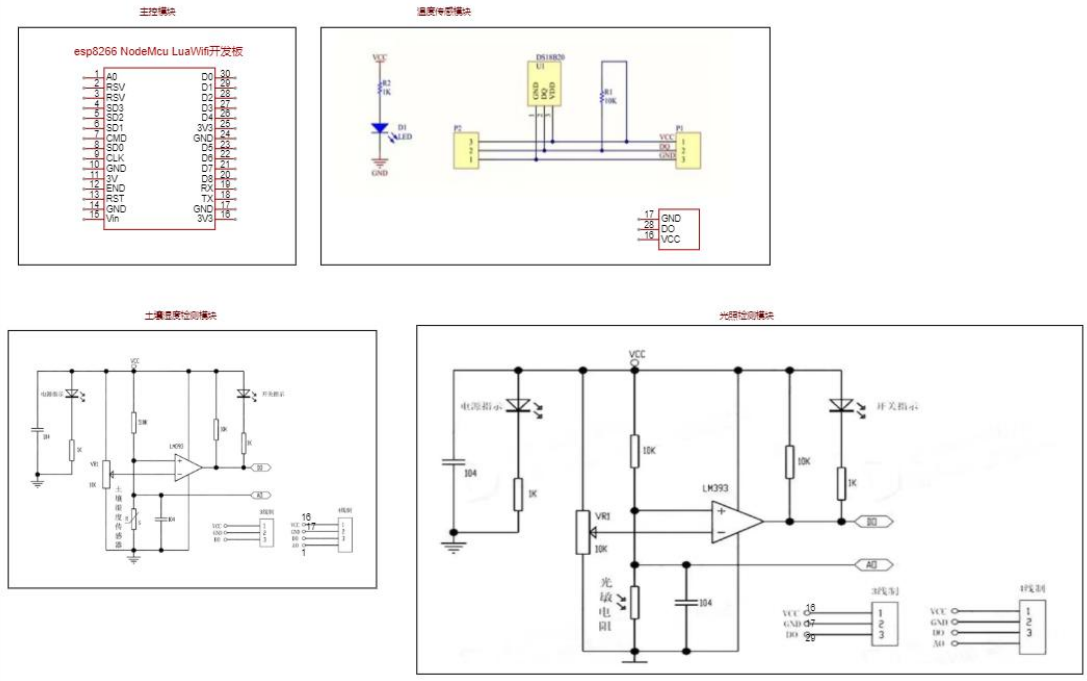
实物演示

四、 物联网集群设备介绍

概括

我们的物联网芯片使用的是 esp8266，它能够通过编程实现开启一些 tcp 接口实现请求接口获得设备上的引脚信息或控制引脚上的电位。这个物联网芯片很适合制作我们物联网设备。

我们使用到的设备有：土壤湿度传感器、温度传感器、亮度传感器、二极管。其中我们用一个二极管来代表这个集群中的单机设备控制的模块，这个模块可以是继电器，可以是小水泵，可以是各种的有用的可控模块。



如果访问了物联网设备的根目录，会看到一个指令提示界面，这个界面是物联网单机设备的欢迎界面，也可以用这个界面是否可访问来判断物联网设备是否在线。

功能拓展说明

我们在演示项目中使用了一个红色的发光二极管来表示物联网单机设备上可控的元件，这个原件根据物联网芯片性能可以替换成任意的功能控制模块，比如洒水、施肥、展开遮光布、打开雨伞等。并且我们还在数据库管理层面为设备添加了分组和命名，您可以根据这些命名依据区分各种不同的集群设备。设备在此基础上，各司其职，各自进行自己的自动化操作。

如果想拓展不同的设备类型，只需要根据对应可控原件的控制规则，修改硬件烧录的程序，并在对应的引脚更换可控原件。这样同样可以使用我们的流转后端进行自动化启动设备和其他监控操作。

集群模式

点模式:像共享单车一样每台物联网设备都加入联网模块把物理机本地接口映射到互联网上,每台机器的公网 ip 注册到服务列表中

伞模式:附近区域一些物联网设备连接同一个中枢设备统一进行以端口号区分设备的端口映射实现一定物理范围内设备服务可被发现

开发环境和实用技术

开发环境：ESP8266 开发板，ArduinoIDE，嘉立创 EDA 专业版

应用技术：C 语言，模块图纸及相关文档，常用硬件开发库，ESP8266WiFi，ESP8266WiFiMulti，ESP8266WebServer，OneWire，DallasTemperature，stdio

五、 Go 缓存流转介绍

概括

为了保证我们的大集群在前端访问时高并发、高可用，我们使用了 redis 作为缓存。但是我们还要获取实时信息，这就需要专门对设备集群进行遍历，并将实时的设备信息写入 redis 缓存中。Go 语言在并发编程中有着极为显著的优势，结合我们自己实现的使用队列方式的设备轮询和处理机制，实现了这种特别的缓存流转机制。除了可以按照设定的时间进行设备状态轮询并将信息写入缓存中，设备上的可操控原件阈值性控制也在这里实现。

我们使用 xorm 连接 mysql 数据库，使用 Redigo 操作 redis 缓存，并且使用 Gin 提供了刷新、插入新设备、插入新的区间等功能的 tcp 请求接口。实现了轮询服务强制更新和热更新等。

其中最值得一提的是我们基于 java 多线程底层的队列思想和 go 程信道机制制作的设备轮询机制。首先我们定义了一个储存设备地址的自定义字符串队列，之后我们启动了一个 go 程，并在 go 程中的逻辑执行之前进行信道值等待，只有当上一个 go 程结束后，才会有值传入信道，使得当前在等待信道传值的 go 程继续执行。go 程中逻辑即从队列中取出一个设备地址，之后对设备进行读取、控制之后将设备信息写入 redis 缓存，最后在将刚刚操作完成的设备放到队列尾部实现环形读取。

Go 缓存流转也可可以使用 grpc 等技术或直接相互请求的形式形成集群。

各功能模块

“刷新状态”

请求方式：GET
请求路径：/flush/:way
参数类型：路径参数
参数实例：/flush/192.168.100.106:80
说明：刷新指定设备当前信息并且写入 redis 缓存，返回从设备中读取到的所有数据

“添加设备”

请求方式：GET
请求路径：/addAddress
参数类型：address
参数实例：/addAddress?address=192.168.100.106:80

说明：添加设备地址到遍历的队列中

“添加区间”

请求方式：GET
请求路径：/addArea
参数类型：address,min,max
参数实例：/addArea?address=192.168.100.106:80&min=10&max=20
说明：添加或更新设备上可控原件的触发区间

开发环境和实用技术

开发环境：Goland2021，Golang1.20，Git

应用技术：Gin，xorm，Redigo，一种采用 Java 多线程底层队列思想和 Go 程信道特性制作的物联网硬件状态检查轮询机制

六、 Boot 微服务集群介绍

概括

首先，我们使用了经典的 SSM 架构，因为业务本身并不复杂，单机就能承载一定数量的访问量。我们采用分布式架构有两个目的：一是将轮询设备和随设备状态控制设备的业务功能交给更擅长并发的 go 语言（go 缓存流转集群）来进行；二是为了实现高性能和高可拓展性，以达到大面积的应用覆盖。

在 boot 设备单机上，我们使用了 swigger2 生成接口文档，展示了已经实现的业务接口。可以通过访问”[IP]:[端口]/doc.html”，指向

单台后端设备查看对应文档。注意不要直接访问网关的地址来浏览接口文档，因为我们的网关使用的是轮询服务集群的方案，您可能查看的并不是您想看到的 boot 服务的接口文档。

如果想测试单台服务器是否在运行，您可以直接请求单台服务器的根路径地址，即”[IP]:[端口]/”页面。我们使用 Thymeleaf 给集群中单设备准备了一个欢迎页面，其中包含了项目名称和我们开源项目的地址。如果请求这个路径报错，说明该单机设备没有被启动。当然，也可以在 nacos 注册中心查看相关情况。

Green Belt

基于大数据和物联网的可视化城市绿化带自动灌溉系统

开源地址：https://github.com/tyza66/GreenBelt

我们曾使用 jedis 与 redis 缓存交互，但是期间遇到了一些问题。当我们两个接口同时收到前端的请求并且同时读取缓存的时候会出现”串台”的现象，即同一个 jedis 对象对缓存读取出来的信息混淆，可能最终返回的不是我们需要的数据。因此我们改用了全新稳定的 redis 中间件 Lettuce 与 redis 缓存之间进行交互，实现了可并发，同时无冲突。

在登录功能中，我们初期打算使用 session 进行登陆状态持久化，但是因为我们使用的网关不支持 session，我们更换成了 token 的方式。在此基础上，我们想到了撞库的问题，因此提出了一个安全的等状态验证方案，即使用多层 hashmap 才会映射到用户信息（这一点在我们的演示项目中没有直接实现出来，处于设想阶段），而不是直接通过 token 的读取就断定用户登陆状态。

用户注册登录模块介绍

“用户登录”

请求方式：POST
请求路径：userl/login
参数类型：对象
参数实例：{
"account": "",
"id": 0,
"nickname": "",
"password": "",
"statu": 0,
"username": ""
}
说明：接受一个 post 表单并且在后台访问 mysql 数据库查询是否有用户与所传递过来的信息相匹配，如果匹配就在 json 信息中返回 ok

“生成图片二维码”

请求方式：GET
请求路径：userl/qcr
参数类型：无参数
说明：返回一个由 BufferImage 转换的图片二进制文件，其中的信息是使用随机方法产生的随机验证码

“注册新用户”

请求方式：POST
请求路径：userl/sign
参数类型：对象
参数实例：{
"account": "",
"id": 0,
"nickname": "",
"password": "",
"statu": 0,
"username": ""
}
说明：这里的 nickname 用来暂时传递验证码用，其他几个是注册账号的信息，请求这个接口后首先会验证存在 nickname 中的验证码正确与否，之后会访问数据库插入对应的表单信息，如果都正确就会以 json 的形式返回成功信息

用户登陆状态模块介绍

“验证登录状态”

请求方式：GET
请求路径：/user/logined
参数类型：无参数
说明：访问这个接口时，首先拦截器拦截请求并验证 token，如果 token 过期或无效则会返回错误信息，如果 token 有效并且没有过期，那么返回 json 信息表示有登陆状态

设备状态模块介绍

“添加出水区间”

请求方式：POST
请求路径：/gb/addgbareas

参数类型：对象
参数实例：{
"address": "",
"max": "",
"min": ""
}
说明：服务器接受请求后向出水区间（或其他设备控制区间）表单中添加一个阈值，并通过 restTemplate 向组织 go 轮询机的网关对应接口发送添加请求，若都成功则以 json 形式返回成功信息

“添加物联网设备”

请求方式：POST
请求路径：/gb/addgbs
参数类型：对象
参数实例：{
"address": "",
"group": 0,
"id": 0,
"name": ""
}
说明：接收到请求后向数据库中设备表单中添加信息，并通过 restTemplate 向组织 go 轮询机的网关对应接口发送添加请求，若都成功则以 json 形式返回成功信息

“删除物联网设备”

请求方式：POST
请求路径：/gb/delgbs
参数类型：对象
参数实例：{
"address": "",
"group": 0,
"id": 0,
"name": ""
}
说明：通过请求值中的 id，在数据库表单中删除对应的项

“刷新设备”

请求方式：GET
请求路径：/gb/flush/{address}
参数类型：路径参数
参数实例：/gb/flush/192.168.100.106:80

说明：通过 restTemplate 向 nginx 网关的对应接口和地址调用 go 缓存流转服务器中的刷新接口，更新缓存中的各设备状态和参数，并且拿到 json 格式的当前设备状态内容，重新整理之后返回

“获取指定设备亮度”

请求方式：GET
请求路径：/gb/getLiangdu/{address}
参数类型：路径参数
参数实例：/gb/getLiangdu/192.168.100.106:80
说明：直接通过 Lettuce 向缓存中读取对应信息，如果缓存中没有信息则通过缓存击穿方案获取信息，获取完信息后返回

“获取指定设备湿度”

请求方式：GET
请求路径：/gb/getShidu/{address}
参数类型：路径参数
参数实例：/gb/getShidu/192.168.100.106:80
说明：直接通过 Lettuce 向缓存中读取对应信息，如果缓存中没有信息则通过缓存击穿方案获取信息，获取完信息后返回

“获取指定设备温度”

请求方式：GET
请求路径：/gb/getWendu/{address}
参数类型：路径参数
参数实例：/gb/getWendu/192.168.100.106:80
说明：直接通过 Lettuce 向缓存中读取对应信息，如果缓存中没有信息则通过缓存击穿方案获取信息，获取完信息后返回

“获取物联网集群设备自动区间”

请求方式：GET
请求路径：/gb/getgbareas
参数类型：无
说明：读取数据库中的设备区间列表并且返回

“按地址获取物联网设备自动区间”

请求方式：GET
请求路径：/gb/getgbareasad/{address}

参数类型：路径参数
参数实例：/gb/getgbareasad/192.168.100.106:80
说明：通过参数向数据库中查询指定设备的设定区间并返回

“获取物联网设备集群”

请求方式：GET
请求路径：/gb/getgbs
参数类型：无
说明：读取数据库中的设备列表并且返回

“修改出水区间”

请求方式：POST
请求路径：/gb/updategbareas
参数类型：对象
参数实例：{
"address": "",
"max": "",
"min": ""
}
说明：接收到请求后向数据库中设备表单中修改信息，并通过 restTemplate 向组织 go 轮询机的网关对应接口发送修改请求，若都成功则以 json 形式返回成功信息

开发环境和实用技术

开发环境：Idea2020，Maven3.6，Java1.8，Nacos1.4.1
应用技术：Spring Boot2.7.11，Spring Cloud，Spring Cloud Alibaba，Nacos，Gateway，Mybatis，Lombok，Thymeleaf，nife4j-openapi2(swagger2)，Hutool，java-jwt，Lettuce

七、 Nginx 网关集群介绍

概括

我们使用 nginx 网关轮询代理 go 缓存流转的接口，主要是能反向代理一部分 go 缓存流转集群中的服务，并且实现轮询负载均衡。

八、 Gw 网关集群介绍

概括

Gateway 网关集群中每个服务同 boot 微服务集群中每个服务一样都注册到了 nacos 注册中心集群，Gateway 网关中选择轮询的形式访问其他服务端，并且可拓展。

我们在 Gateway 网关中设定所有请求来源允许跨域，从而达到在测试阶段、实用阶段的小程序、app、网页都可以访问网关反向代理的服务。

网关配置

```
  server:
 port: 8888
spring:
 application:
   name: greenbelt-gateway
 cloud:
   gateway:
     routes:
       - id: greenbelt_route
         uri: lb://greenbelt-service
         predicates:
```

```
        - Path=/**
    globalcors:
      add-to-simple-url-handler-mapping: true
      cors-configurations:
        '[/**]':  #拦截的请求
          allowedOrigins: #允许跨域的请求
            - "http://localhost:8080"
            - "http://127.0.0.1:8080"
            - "http://192.168.100.103:8080"
          allowedMethods: #运行跨域的请求方式
            - "GET"
            - "POST"
            - "DELETE"
            - "PUT"
            - "OPTIONS"
          allowedHeaders: "*" #允许请求中携带的头信息
          allowedCredentials: true #是否允许携带 cookie
          maxAge: 36000 #跨域检测的有效期,单位 s

nacos:
  discovery:
    server-addr: localhost:8848
```

```
username: nacos

password: nacos
```

九、 Nacos 注册中心介绍

概括

来自 Spring Cloud Alibaba 系列的注册中心，是目前功能最全面、可拓展性最强、相对稳定、开源的注册中心，常见于各种微服务项目和服务中。

我们使用 Nacos 注册中心注册 Boot 服务集群中的设备和网关集群中的设备，方便管理并借助心跳机制处理服务、服务器崩溃等问题。

十、 Mysql 数据库介绍

概括

我们在项目中使用的 Mysql 数据库在 docker 容器中，但是这与数据库在实体机上没有什么分别，我们使用的只是简单配置的 Mysql 数据库。

我们在项目的‘database’文件夹中提供了我们开发阶段测试的数据库表结构和内容的 sql 文件。

为了方便操作，和在 go 缓存流转服务器上实现阈值控制和轮询分别判断，我们的设备信息和阈值信息不是存放在同一张表中的。

十一、 Redis 缓存介绍

概括

为了能够响应密集的数据获取请求，我们使用 redis 作为缓存数据库，boot 集群中的单机直接从缓存中读取信息，而不是反复的去关系型数据库的表中查询，go 缓存流转也会往关系型数据库中的表中写入数据，这样大大减少了数据库服务器设备的压力，规避了关系型数据库因海量请求造成因查询修改过多造成关系型数据库表内容受损或服务器崩溃等。

在演示程序中，我的 redis 服务器是放在 docker 容器中的，并且 default 用户有一个密码 000415。

十二、 混合型分布式架构介绍

概括

我们使用手写的方式实现分布式请求和 Spring Cloud Alibaba 系列注册中心 Nacos 提供微服务可拓展性相结合，大幅度无上限提高运行稳定性和效率。我们的演示项目是单台设备运行，单台服务器部署，可能无法体现优势。

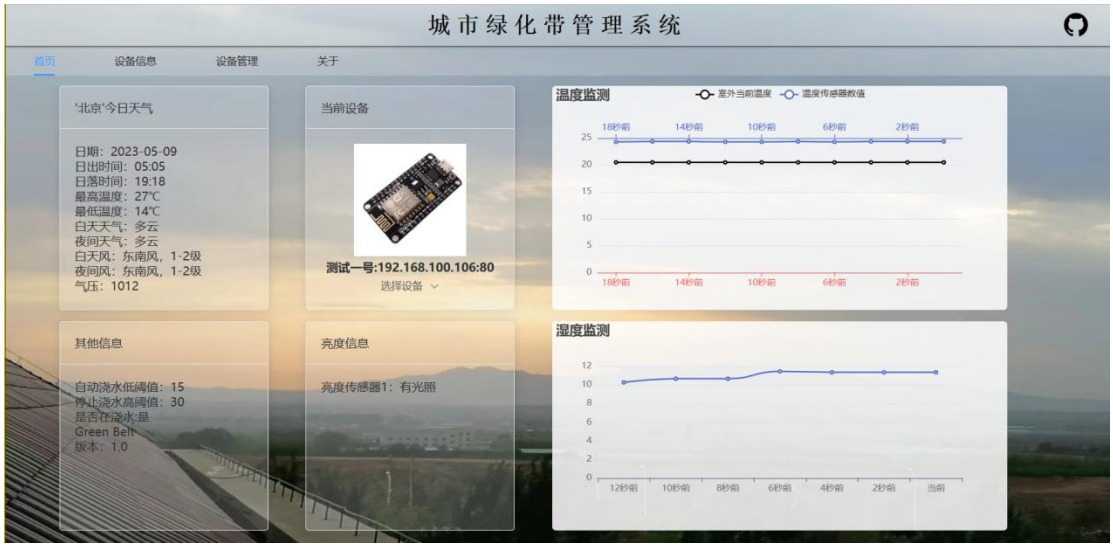
十三、 前端-Web 介绍

概括

Web 前端使用 vue 脚手架以及 vue 系列框架，ElementUI plus，echarts 等。前后端分离，前端与后端交互使用的是与 vue 搭配最好的请求框架 axios。

在数据实时显示的实现上，我们采用定时轮询的方式向网关发送请求，并由网关将请求转发到 boot 集群的请求接口上获得实时相应，之后再由定时轮询的 echarts 组件更新函数进行同步到页面上。

页面中的今日天气等信息来自和风天气提供的免费接口，温度检测中的黑线，也就是室外当前温度的数值来自和风天气实时天气预报中温度最高值和最低值的平均值，这个数值也会随着接口中温度的返回值而变化，只不过为了减少对收费接口的请求频率，我们将这个值的检测周期设置的很长。



在设备信息和设备管理界面中，我们的开关水阈值也是通过定时轮询的方式实现的实时信息，也就是说即便是从数据库层面修改信息也会实时更新进来。

其中编辑，添加，删除等操作就是简单 jdbc 操作实现的。

开发环境和实用技术

开发环境：Node，Vue3.0，HBuilderX，VS code，@vue/cli

应用技术：@vue/cliV，ue2.0，Elementui-Plus，Vuex，Vrouter，Axios，Echarts，uiverse.io，iconfont.cn，和风天气

十四、 前端-APP/小程序介绍

概括

APP 和小程序使用 uniapp 一站式开发，在 APP 和小程序中我们没有采用连接我们庞大的后端集群的方式，而是通过输入单机设备的地址和端口连接单台设备进行状态检测和操作，这样可以补充 web 端对单机操作的空缺。

有了 APP 和小程序端，单机设备直接使用，可以不依赖于后端自动体系。并且，在故障排查时直接使用 APP 或者小程序连接单台

设备检测故障极为方便。

开发环境和实用技术

开发环境：Node，Vue2.0，HBuilderX，@vue/cli，微信开发者工具

应用技术：uni-app，uiverse.io

图片

十五、 网络运营商介绍

概括

国内的网络运营商为了给物联网设备和物联网应用提供便利，推出了物联卡服务，我们的物联网设备可以独立安装物联卡支持的模块进行联网，也可以以集群的模式，几台物联网设备使用同一个映射中心，进行集中的端口映射。

十六、 FOREVER WIFI HUB 介绍

概括

FOREVER WIFI HUB 是我们测试阶段使用的局域网提供源，简单来说就是一个 WiFi 发射源，可以提供 WiFi 局域网和广域网的网络连接。

十七、 拓展开发

我们的项目仅为演示阶段，仅实现了部分的功能。但是我们的硬件设备和开发的框架以及代码中提供了很大的可拓展性，如果想使用我们的项目进行实际使用，只需修改业务需求细节和逻辑，并拓展相应部分的代码，相信我们的项目的拓展会意想不到的便捷，投入使用后也会符合您的设想。

十八、 开源

为了提供学习价值、复用价值，本项目将在比赛投稿截止日期当天将代码仓库设置为公共，开源地址为：tyza66/GreenBelt: 基于大数据和物联网的可视化城市绿化带自动灌溉系统 (github.com)。

十九、 参考资料

参考项目：

https://github.com/redis/redis
https://github.com/alibaba/spring-cloud-alibaba
https://github.com/tyza66/PetAdoption
https://github.com/tyza66/GreenBelt
https://github.com/tyza66/GoStart
https://github.com/tyza66/GoRun
https://github.com/tyza66/SpringCloudAlibabaStart

参考文章：

http://www.taichi-maker.com/homepage/esp8266-nodemcu-iot/iot-c/
https://blog.csdn.net/weixin_45821811/article/details/116211724
https://blog.csdn.net/qq_17351161/article/details/113573382
https://blog.csdn.net/qixiwl/article/details/128829146
https://blog.csdn.net/qq_42883074/article/details/123471586
https://m.php.cn/be/go/508196.html
https://blog.csdn.net/jeremy_ke/article/details/127524414
https://blog.csdn.net/hinsss/article/details/119981795
https://blog.csdn.net/qq_34458968/article/details/123132247
https://echarts.apache.org/handbook/zh/basics/import
https://www.ngui.cc/el/3129250.html?action=onClick
https://blog.csdn.net/cidaren/article/details/118759256
https://www.cnblogs.com/xielong/p/9996503.html
http://www.yuanchengzhushou.cn/article/8863.html
https://blog.csdn.net/m0_57963535/article/details/120390994
https://blog.csdn.net/qq_46124502/article/details/106575711

参考书籍：

https://www.bookstack.cn/books/nacos-2.0-zh