

中国大学生计算机设计大赛



物联网应用类作品技术文档

作品名称： SmartAgri-领航助农“智”平台

版本编号： v1.2

填写日期： 2023 年 4 月

填写说明：

- 1、 本文档适用于物联网应用类作品
- 2、 正文一律用五号宋体，一级标题为二号黑体，其他级别标题如有需要，可根据需要设置；
- 3、 本文档为简要文档，不宜长篇大论，简明扼要为上。
- 4、 提交文档时，以 PDF 格式提交本文档；
- 5、 本文档内容是正式参赛内容组成部分，务必真实填写。如不属实，将导致奖项等级降低甚至终止本作品参加比赛。

目录

第一章 技术方案	
第二章作品概述.....	1
1. 摘要.....	1
2. 主要功能.....	1
3. 产品性能.....	1
4. 竞品比较.....	2
第二章 技术方案	
第一章 技术方案.....	2
1. 概要设计.....	2
1.1 技术路线.....	2
1.2 使用展示.....	3
2. 详细方案.....	7
2.1 物联网节点部分.....	7
2.2 物联网协议部分.....	7
2.3 Web 平台部分.....	8
2.3.1 技术路线.....	8
2.3.2 架构图.....	8
2.3.3 数据库 E-R 图.....	9
2.3.4 功能特色.....	9
第二章 测试报告.....	11
1. Web 高并发设计.....	11
2. Web 性能测试.....	11
3. Web 安全测试.....	12
4. 硬件系统测试.....	12
第三章 安装及使用.....	12
1. 最低环境要求.....	13
2. Web 基本部署.....	13
1.1 目录结构.....	13
1.2 快速入门.....	13
2.2.1 安装依赖.....	13
2.2.2 修改配置.....	13
2.2.3 启动数据库.....	13
2.2.4 启动前端.....	14
2.2.5 启动后端.....	14
2.2.6 启动 EMQ X 服务器.....	14
2.2.7 启动传感器前端.....	14
2.2.8 启动传感器后端.....	14
第四章 项目总结.....	14
1. 项目收获.....	14
2. 使用前景.....	14

第一章 作品概述

本作品是一个软硬件结合的智能作物生长环境监控平台。其主要应用领域为农业生产和研究。其主要功能是实时监测和分析作物生长环境中的关键参数，如温度、湿度，以帮助农民或研究人员更好地控制作物生长环境、优化生产和研究效果。

这个平台的创新性在于其整合了软件和硬件两方面的技术。在硬件选择方案，采用支持 mqtt 协议的传感器节点，能无线部署，通过 4G 上传数据，并将数据反馈到软件平台，进行分析和处理。在硬件部署方面，采用 k8s 部署，提高管理效率。其软件方面采用了深度学习和数据挖掘等先进技术，能够通过对作物生长环境数据的分析，提供更为准确和个性化的生长环境控制建议。此外，该平台还具有远程控制和多终端支持等应用创新性。

第二章 需求分析

1. 摘要

目前，市场上缺乏农田环境监控与调控管理方面的系统，需要靠人力来检测环境情况进行相应的调节。经常需要工作人员四处奔波使用各种仪器实时地监控数据，安排专人留宿监管，导致成本高，耗时长。此外，为编制土壤，湿度等环境报告，还需要定期记录数据，采用人工的方式工作量大且容易出错。总的来说，目前需要一款系统，来进行自主监控和预警以及数据分析与预测的功能。

因此，开发一种农田环境监控系统称为迫切需要，本项目应运而生。结合物联网，智能数据分析，Web 这三大技术。一改作物环境检测的成本高，耗时长等缺点，使之成为具有智慧能力的一站式环境监控平台，挖掘数据本身的价值，赋能智慧农业，守护一方水土；旨在实现低成本，高可靠，面向商用的物联网落地方案，为农业环境监管打造“高效感知、互联互通、资源共享、智能分析、智能调控”的一体化智能化综合监控平台。

2. 主要功能

本项目是软硬件兼备，围绕智能作物生长环境的监控平台。该系统具备温湿度集中监控，历史数据回放，数据研判与展示、智能数据预测等功能。

- ◆各传感器实时上传数据，存放至 EMQX 服务器，并实时并发写入数据库。
- ◆后端实现对数据的分析处理，在前端以图表形式展示处理后得出的数据。
- ◆通过智能数据分析算法对历史数据进行分析，进行预测性调控。
- ◆上传区域内作物图片，通过深度学习算法对区域的作物生长情况进行打分，智能分析出现异常的环境因子，并对其自动化做出调整。
- ◆高效的告警功能，快速感知异常信息，通过微信/钉钉通知至手机。



3. 产品性能

在物联网协议方面，结点通过 MQTT 协议与服务器进行通信，开放，功耗低，保证稳定连接。

传感器部分，采用支持 mqtt 协议的传感器节点，4G 数据传输，且能耗低，续航时间久。采用无线 k8s 分布式部署，高效地管理传感器集群。

数据接收服务器方面，采用 EMQX 服务器，可支持高并发，多连接。

软件方面，中心服务器采用 Python Flask 框架，MySQL 服务器，Redis 缓存，性能强大，安全性高。

所有源码参数统一配置，均具备高鲁棒性。

```
db:
  mysql: #mysql的
    host: localhost
    port: 3306
    username: agriculture
    password: pNMTmxCHxmSa76N5
    db: agriculture
  redis:
    db1: 1
    db2: 2
    host: localhost
    port: 6379

host: #EMQX提供的port
ip: 127.0.0.1
port: 1883

msg: #手机推送的sendkey
SendKey: SCT205285T3m6qMIoLqHnRXYF1Ch2FRaJV
```

(图 1 配置文件)

中间件方面，docker 进行打包部署，nginx 服务器对前端进行部署，celery 提供对数据的异步与定时处理。实现高效处理与部署。

4. 竞品比较

在阿里巴巴中搜索关键字“作物生长环境监控”，选取能监控数据的 50 套项目与本项目比较，并将传统的作物生长环境检测方法与本项目进行比较，结果如下。

	竞品	本品
数据采集	人工采集数据	自动采集数据

特点	无	Ai 预测，自动调控、消息推送
数据存储	人工抄录	自动存储分析归档,生成分析图表
结点部署	有线部署	无线部署
人力成本	高	低

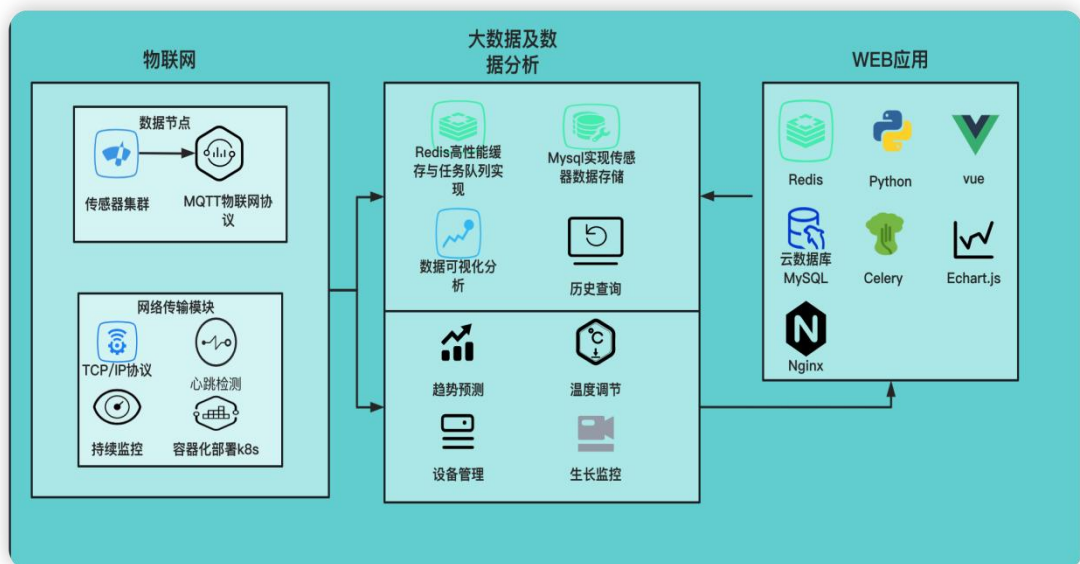
(表 1 竞品比较)

第三章 技术方案

1. 概要设计

系统的三大技术方向分别是**物联网、大数据及数据分析、Web 应用**。技术之间依赖紧密，但又结构清晰，毫无冗杂。三大组件间大部分通过 HTTP 的 API 进行交互，大大降低功能拓展的难度，同时便于各组件内的维护。

所涉及的技术可用如下图表大致表示。



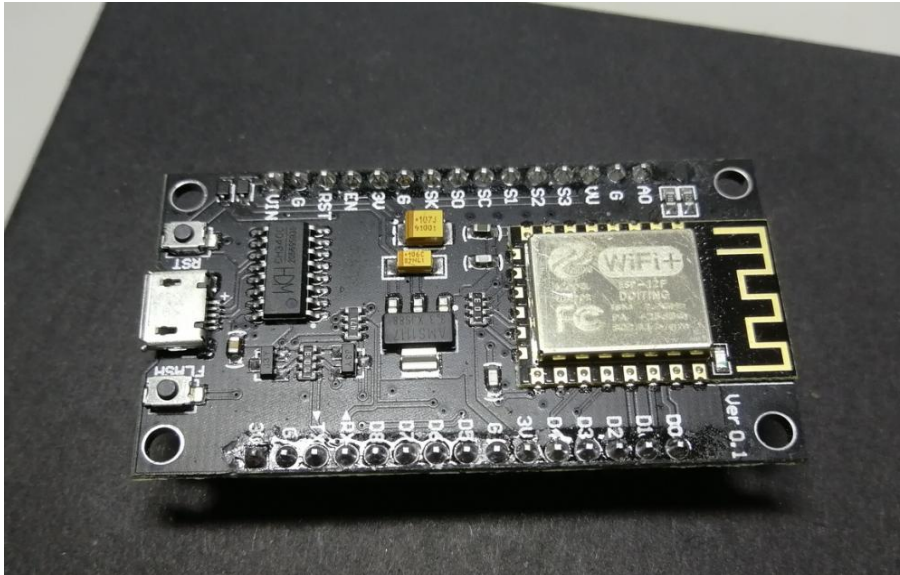
(图 2 技术路线)

第四章 方案实现

1. 详细方案

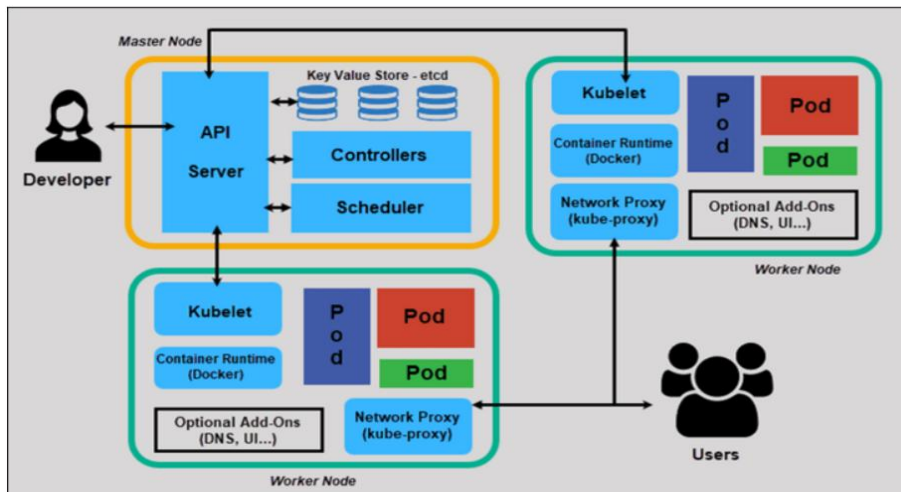
1.1 物联网节点部分

◆ 采用支持 mqtt 协议族的 esp8266 节点,esp8266 是一种功能强大、易于使用的 Wi-Fi 模块，可以满足各种物联网和嵌入式系统的需求。



(图 3 ESP8266 节点)

◆ 使用 K8S 技术进行部署



(图 4 K8S 部署原理图)

具有如下好处：

- 允许用户在各种环境中使用相同的部署、扩展和管理工具。
- 提供了丰富的 API 和工具，使用户可以更轻松地进行监控、调试和管理容器应用程序。

1.2 物联网协议部分

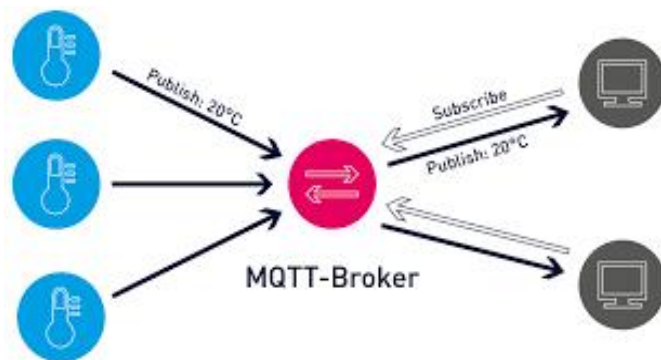
◆ 节点与服务器的通信采用 mqtt 协议

相较于 HTTP 等其他网络传输协议具有以下优点：

1. 轻量级：MQTT 协议是一种轻量级的协议，它的通信开销非常小，适合在带宽有限的网络中进行通信。
2. 可靠性高：MQTT 协议支持 QoS 服务质量等级，可以保证消息的可靠传输。
3. 灵活性强：MQTT 协议可以通过订阅/发布模式，实现对多个客户端的消息传输，并支持多种消息格式，包括二进制数据、文本数据、JSON 等。
4. 易于集成：MQTT 协议的客户端和服务端实现非常简单，易于集成到各种应用程序中。

5. 跨平台支持：MQTT 协议支持各种不同的操作系统和编程语言，可以在多种平台上进行通信。

6. 节省能源：MQTT 协议可以通过心跳包来维持连接，减少通信过程中的开销，从而节省设备的能源。



(图 5 传感器结点与 EMQX 交互图)

◆ 服务器采用 EMQ X 服务器接收节点上传的数据。

相较于 Mosquitto 等其他服务器，EMQ X 服务器具有以下优点：

1. 高度可扩展性：EMQ X 可以很容易地扩展到支持大量的连接和消息。EMQ X 可以横向扩展，通过添加更多的服务器来处理更多的连接和消息。

2. 支持多种协议：EMQ X 支持 MQTT 和 MQTT-SN 等多种协议，因此可以适应各种 IoT 设备和应用场景的需求。

3. 安全性：EMQ X 提供了安全的连接和身份验证机制，包括 TLS/SSL 支持、用户名密码认证、ACL（访问控制列表）等。

4. 高可靠性：EMQ X 支持主从架构和集群部署，可以保证消息传递的高可靠性和可用性。

5. 高性能：EMQ X 是一个轻量级的消息服务器，具有高性能和低延迟。它使用了异步 IO 和线程池等技术，可以处理大量的消息。

6. 开源免费：EMQ X 是一个完全开源的物联网服务器，提供社区版和企业版，可免费使用，也提供付费服务和技术支持。

1.3 Web 平台部分

1.3.1 技术路线

后端采用 Python Flask 作框架，MySQL 数据库作存储库。

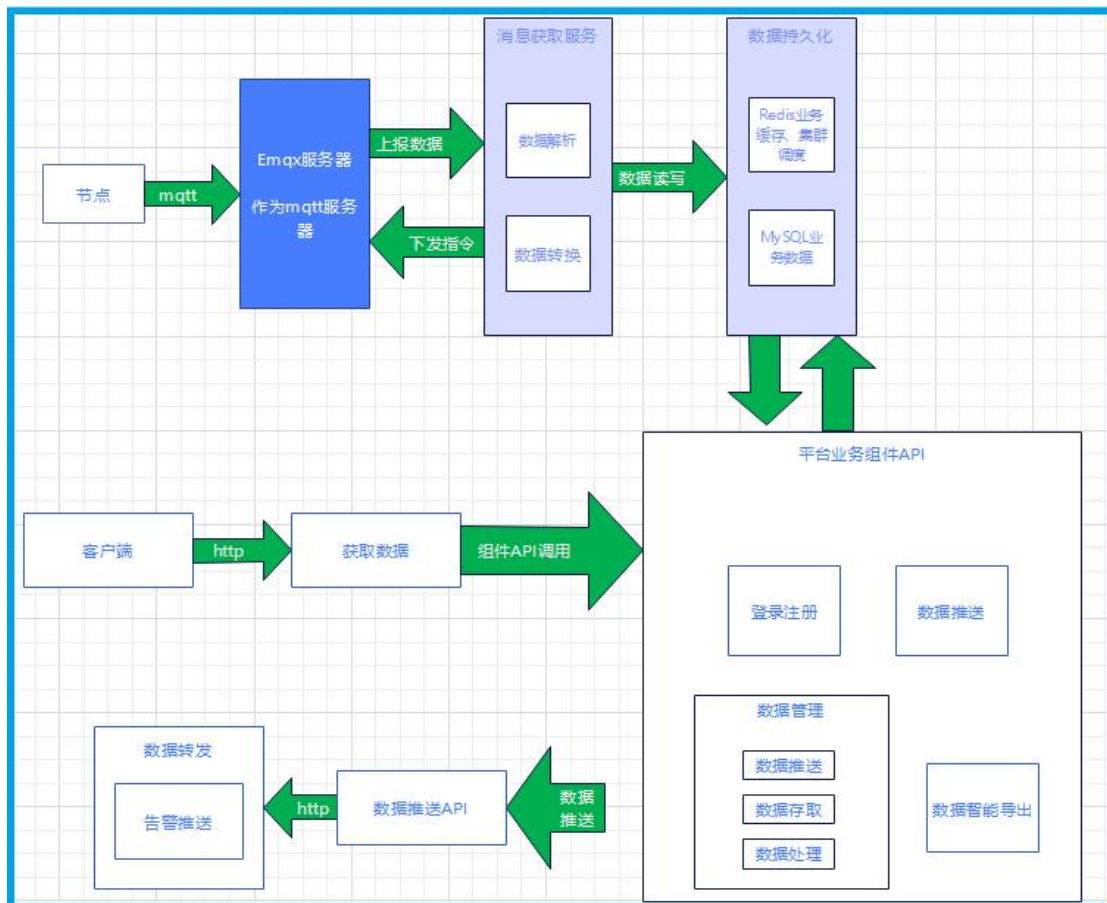
中间件使用 Redis，实现高效存取，数据持久化。使用 Celery，实现定时预警推送以及对数据库中过期数据的清除。

前端使用 vue 脚手架做基本框架，vue+axios 实现响应式布局，Echarts 绘制图表。

1.3.2 架构图

团队考虑了如下设计：

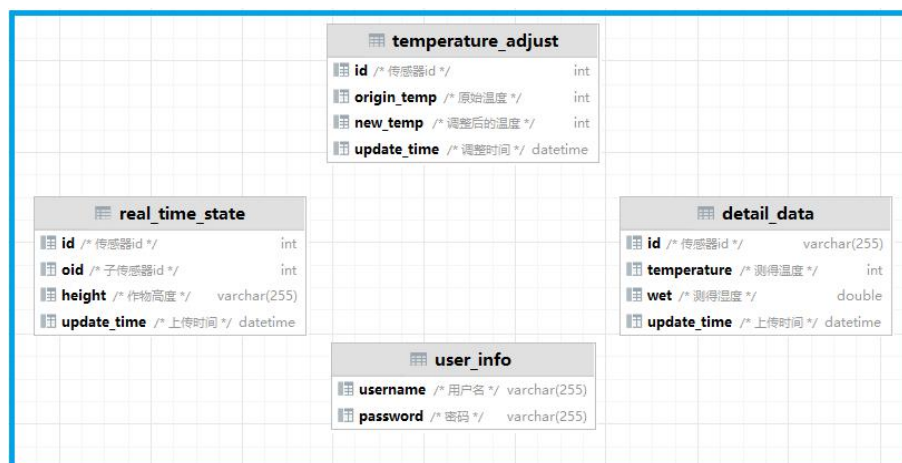
- ◆ 使用前后端分离设计思想，设计规范的接口 API，以便后序对接。
- ◆ Flask 后端实现各种组件 API，方便各种业务的调用。
- ◆ 后端实现智能数据处理，可以定时自动进行进行数据分析与预测。
- ◆ 实现微信/钉钉消息推送。



(图 6 系统架构图)

1.3.3 数据库 E-R 图

总计 4 张表，分别为传感器实时数据，作物实时数据，温度调节记录，用户信息。各表和字段均已加规范注释，使用 DataGrip 导出 ER 图如下。



(图 7 数据库 E-R 图)

2. 功能特色

特色 1: 监控后台

■ 对数据多维度分析与可视化

Web 页面以图表的形式展示所有结点的近十分钟平均温度与湿度。

展示所有数据中的最高温度与湿度

展示各结点的历史温度变化

展示各分区作物生长情况

■ 远程调节结点所处区域的环境参数。

可在 web 页面中手动调节各传感器区域的温度，以实现农场环境的无人调节。

■ 数据警告推送

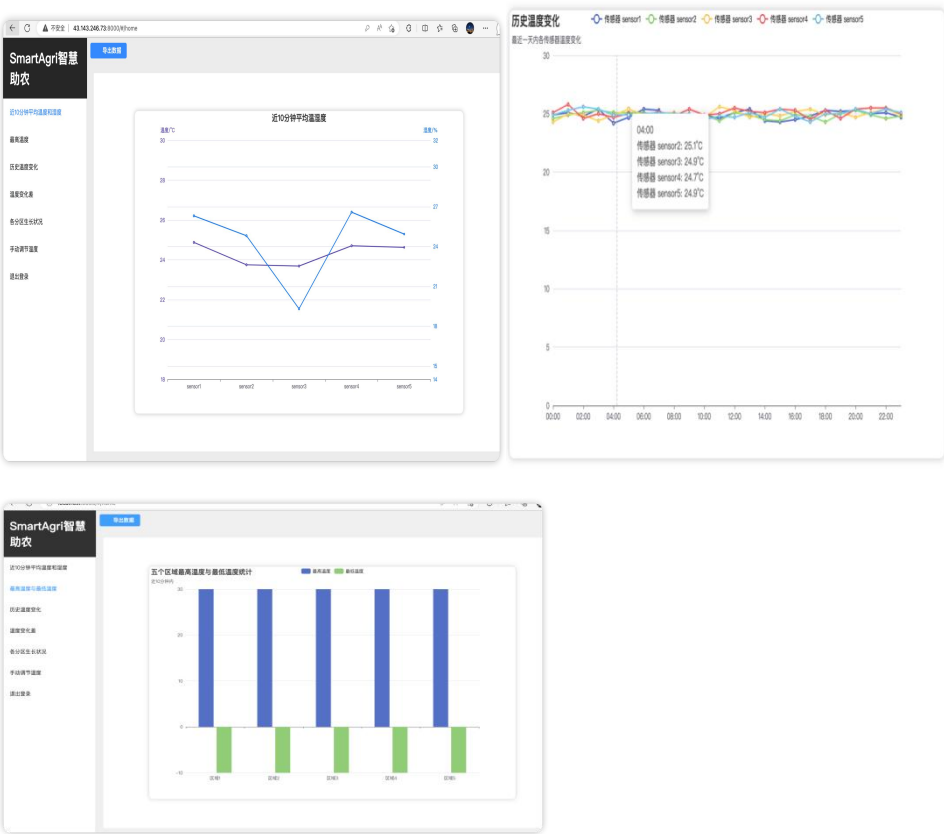
当某片区域的温度或者湿度过高或过低时，后端主动向微信/钉钉推送警告消息，并主动对该区域的温度或湿度进行智能调节。

■ AI 智能调节

后端通过深度学习，获得对作物生长情况进行打分，可根据作物的分数，进行数据探索式分析，获知可能的原因，并主动做出智能调节。

■ 报表功能

可从服务器获得近期的环境数据，以 EXCEL 文件形式返回。



(图 8 数据展示界面)



(图 9 远程调节温度界面)

		temperatur	wet
1			
2	2023-04-14 16:10:02	12.5	27.5
3	2023-04-14 16:10:03	12.83333	28.16667
4	2023-04-14 16:10:03	16.5	27.66667
5	2023-04-14 16:10:04	16.83333	28.66667
6	2023-04-14 16:10:04	16.5	26
7	2023-04-14 16:10:05	16.5	26
8	2023-04-14 16:10:05	16.33333	23.66667
9	2023-04-14 16:10:06	16.33333	22.66667
10	2023-04-14 16:10:06	16.16667	23.5
11	2023-04-14 16:10:07	14	22
12	2023-04-14 16:10:07	19.66667	23
13	2023-04-14 16:10:08	24	23.66667
14	2023-04-14 16:10:08	18.5	27.83333
15	2023-04-14 16:10:09	17.5	24.83333
16	2023-04-14 16:10:09	18.5	21.66667
17	2023-04-14 16:10:10	20.66667	26.16667
18	2023-04-14 16:10:10	18.66667	27.66667
19	2023-04-14 16:10:11	17.16667	25.83333
20	2023-04-14 16:10:11	21.66667	21.33333
21	2023-04-14 16:10:12	22.66667	25.5
22	2023-04-14 16:10:12	19.83333	29.66667
23	2023-04-14 16:10:13	20.66667	25.5
24	2023-04-14 16:10:13	19	22.5
25	2023-04-14 16:10:14	19.66667	23.5
26	2023-04-14 16:10:14	14.83333	27.83333
27	2023-04-14 16:10:15	14.83333	26
28	2023-04-14 16:10:15	14.16667	25.66667
29	2023-04-14 16:10:16	13.5	28.83333
30	2023-04-14 16:10:16	11.16667	32.33333
31	2023-04-14 16:10:17	11.33333	31.83333
32	2023-04-14 16:10:17	14.16667	28.16667
33	2023-04-14 16:10:18	14	30.33333

(图 10 导出数据生成 excel 界面)



(图 11 消息推送手机端界面)

特色 2：高可用设计

物联网设备的鲁棒性，很大程度上依赖终端设备自身的设计，如软件看门狗、自动重启

重连策略等。本平台自身也有一定的鲁棒性设计，具体如下：

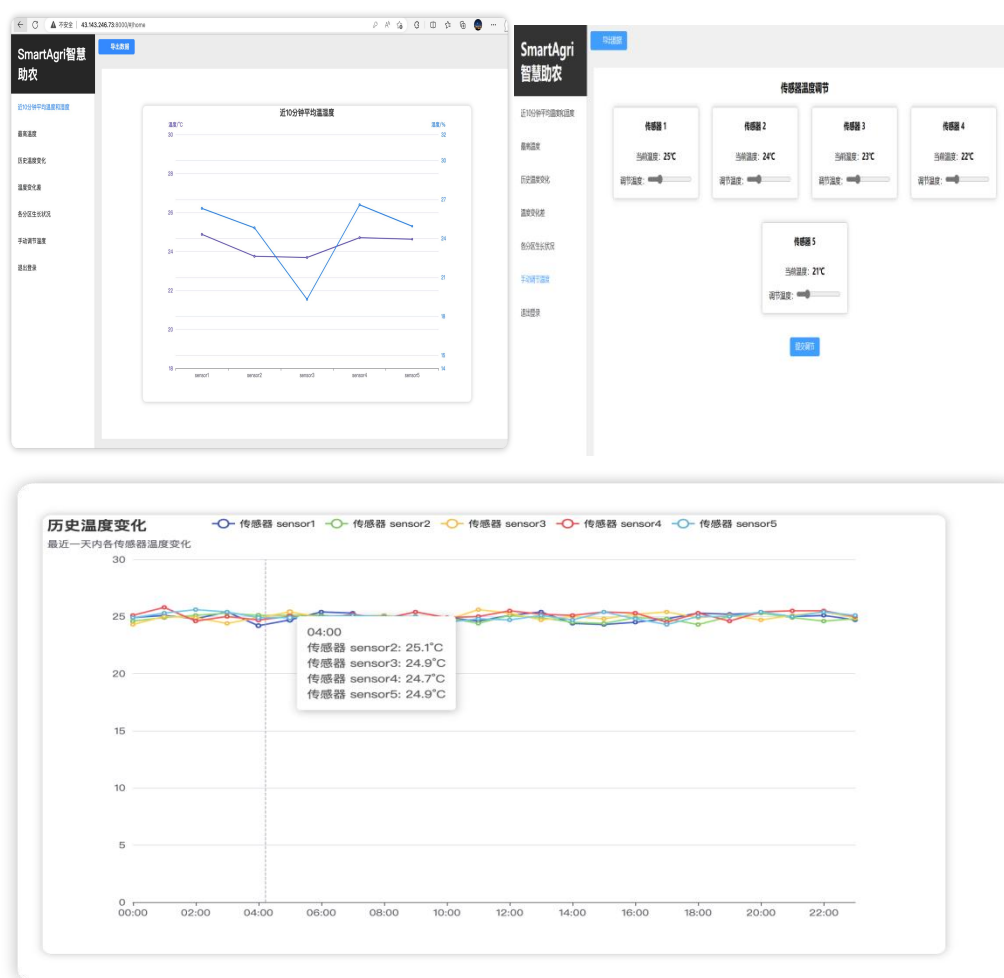
- 心跳包检测，心跳包是由终端设备定时发出，用于服务器验证设备在线状态的方法。
- 设备离线告警。当终端设备超过 10 倍轮询时间未上报时，即触发设备离线告警，在平台首页显示，并推送微信平台。
- 传感器异常侦测。当传感器距离、温度测量值与前一次变化超过 20%，以及 2 次查询无数据响应时，即触发传感器异常侦测。系统将尝试重启站点设备，并推送微信平台。
- 双向绑定。当上报终端与服务器建立 TCP 连接后，终端即发送一条注册包，内容为 4 位 HEX 字符串，作为设备 ID 来让服务器做识别。

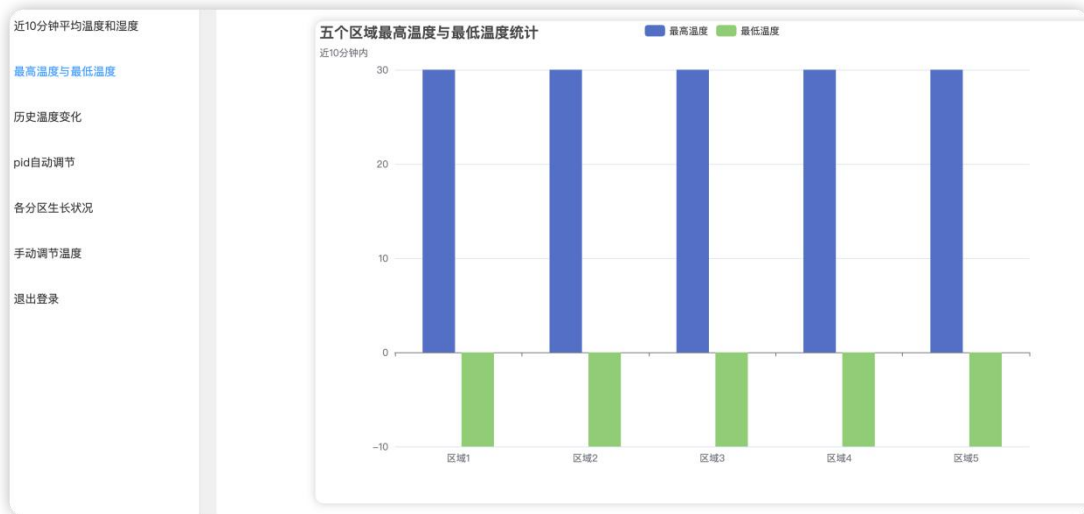
3. 系统实现展示

演示网址：43.143.246.73:8000/#/login

默认账号：admin，密码：admin

◆ Web 界面

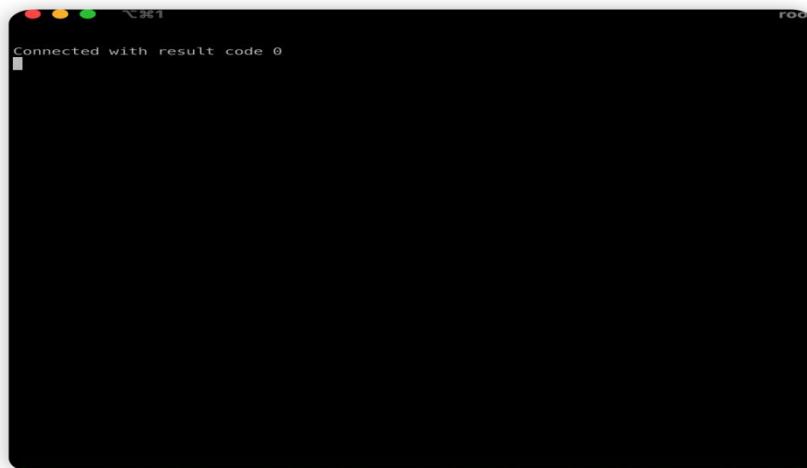




(图 12 Web 界面展示)

◆ 系统部署情况展示

● 传感器前端节点部署



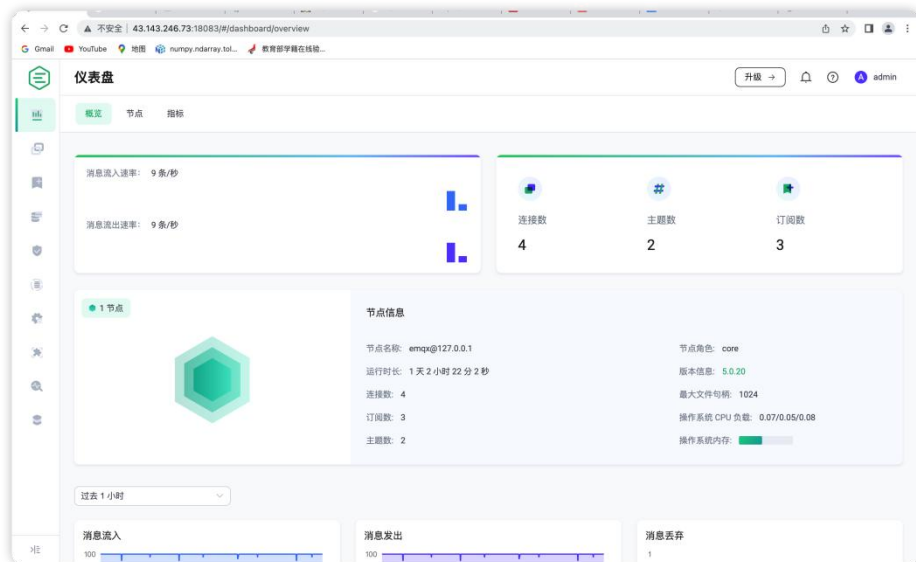
(图 13 传感器前端节点部署成功图)

● 传感器后端部署：

```
root@VM-24-4-ubuntu: ~  
[+]记录成功  
接收到传感器编号为 sensor5 的温度数据, 温度为: temperature = 24 °C,更新时间是 2023-04-12 10:43:40  
[+]记录成功  
接收到传感器编号为 sensor2 的温度数据, 温度为: temperature = 21 °C,更新时间是 2023-04-12 10:43:40  
[+]记录成功  
接收到传感器编号为 sensor1 的温度数据, 温度为: temperature = 29 °C,更新时间是 2023-04-12 10:43:40  
[+]记录成功  
接收到传感器编号为 sensor5 的温度数据, 温度为: temperature = 29 °C,更新时间是 2023-04-12 10:43:40  
[+]记录成功  
接收到传感器编号为 sensor2 的温度数据, 温度为: temperature = 24 °C,更新时间是 2023-04-12 10:43:40  
[+]记录成功  
接收到传感器编号为 sensor4 的温度数据, 温度为: temperature = 23 °C,更新时间是 2023-04-12 10:43:40  
[+]记录成功  
接收到传感器编号为 sensor3 的温度数据, 温度为: temperature = 20 °C,更新时间是 2023-04-12 10:43:41  
[+]记录成功  
接收到传感器编号为 sensor1 的温度数据, 温度为: temperature = 26 °C,更新时间是 2023-04-12 10:43:41  
[+]记录成功  
接收到传感器编号为 sensor3 的温度数据, 温度为: temperature = 24 °C,更新时间是 2023-04-12 10:43:41  
[+]记录成功  
接收到传感器编号为 sensor4 的温度数据, 温度为: temperature = 27 °C,更新时间是 2023-04-12 10:43:41  
[+]记录成功  
接收到传感器编号为 sensor5 的温度数据, 温度为: temperature = 25 °C,更新时间是 2023-04-12 10:43:41  
[+]记录成功  
接收到传感器编号为 sensor2 的温度数据, 温度为: temperature = 20 °C,更新时间是 2023-04-12 10:43:41  
[+]记录成功  
接收到传感器编号为 sensor1 的温度数据, 温度为: temperature = 25 °C,更新时间是 2023-04-12 10:43:41  
[+]记录成功  
接收到传感器编号为 sensor2 的温度数据, 温度为: temperature = 27 °C,更新时间是 2023-04-12 10:43:41  
[+]记录成功  
接收到传感器编号为 sensor5 的温度数据, 温度为: temperature = 29 °C,更新时间是 2023-04-12 10:43:41  
[+]记录成功  
接收到传感器编号为 sensor4 的温度数据, 温度为: temperature = 30 °C,更新时间是 2023-04-12 10:43:41  
[+]记录成功  
[2] 0:python3*
```

(图 14 传感器后端节点部署成功图)

● EMQX 部署：



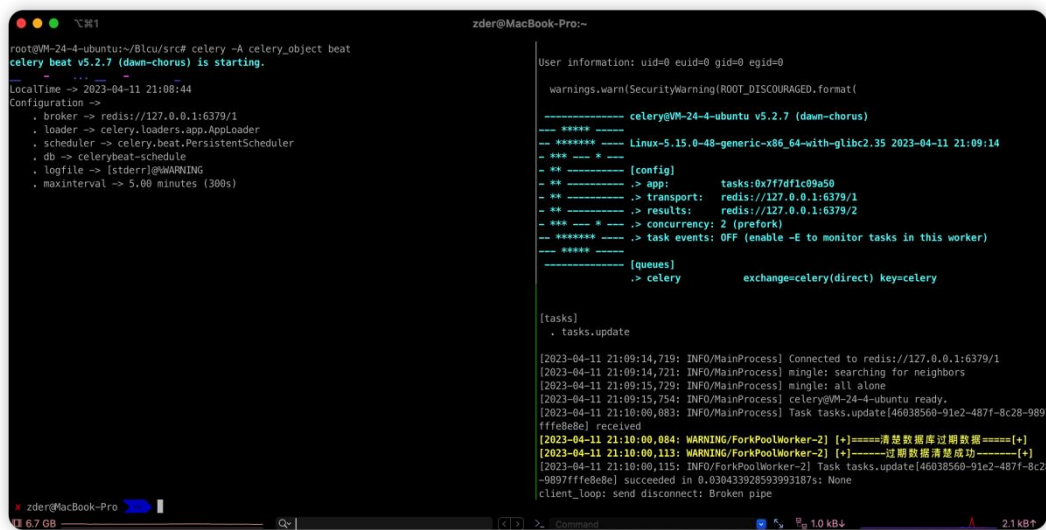
(图 15 EMQ X 部署成功图)

● 数据库部署：

```
root@VM-24-4-ubuntu:~# bash init.sh  
mysql: [Warning] Using a password on the command line interface can be insecure.  
mysql: [Warning] Using a password on the command line interface can be insecure.  
mysql: [Warning] Using a password on the command line interface can be insecure.  
mysql: [Warning] Using a password on the command line interface can be insecure.  
root@VM-24-4-ubuntu:~#
```

(图 16 数据库部署成功图)

● Celery 部署:



```
root@M-24-4-ubuntu:~/Blcu/src# celery -A celery_object beat
celery beat v5.2.7 (dawn-chorus) is starting.
-----
LocalTime -> 2023-04-11 21:08:44
Configuration ->
  . broker -> redis://127.0.0.1:6379/1
  . loader -> celery.loaders.app.AppLoader
  . scheduler -> celery.beat.PersistentScheduler
  . db -> celerybeat-schedule
  . logfile -> [stderr]@WARNING
  . maxinterval -> 5.00 minutes (300s)
-----
User information: uid=0 euid=0 gid=0 egid=0
warnings.warn(SecurityWarning(ROOT_DISCOURAGED),format(
-----
celery@M-24-4-ubuntu v5.2.7 (dawn-chorus)
-----
*****
-----
Linux-5.15.0-48-generic-x86_64-with-glibc2.35 2023-04-11 21:09:14
-----
*****
-----
[config]
-----
. app: tasks:0x7f7df1c09a50
. transport: redis://127.0.0.1:6379/1
. results: redis://127.0.0.1:6379/2
. concurrency: 2 (prefork)
. task events: OFF (enable -E to monitor tasks in this worker)
-----
*****
-----
[queues]
-----
. celery exchange=celery(direct) key=celery
-----
[tasks]
. tasks.update

[2023-04-11 21:09:14,719: INFO/MainProcess] Connected to redis://127.0.0.1:6379/1
[2023-04-11 21:09:14,721: INFO/MainProcess] mingle: searching for neighbors
[2023-04-11 21:09:15,729: INFO/MainProcess] mingle: all alone
[2023-04-11 21:09:15,754: INFO/MainProcess] celery@M-24-4-ubuntu ready.
[2023-04-11 21:10:00,083: INFO/MainProcess] Task tasks.update[46038560-91e2-487f-8c28-989fffe8e8e] received
[2023-04-11 21:10:00,084: WARNING/ForkPoolWorker-2] [*]=====清楚数据库过期数据=====[*]
[2023-04-11 21:10:00,113: WARNING/ForkPoolWorker-2] [*]-----过期数据清楚成功-----[*]
[2023-04-11 21:10:00,115: INFO/ForkPoolWorker-2] Task tasks.update[46038560-91e2-487f-8c28-989fffe8e8e] succeeded in 0.030433928593993187s: None
client_loop: send disconnect: Broken pipe
```

(图 17 Celery 部署成功图)

● Web 前端部署:

PS D:\pycharm2023\flask_server\Blcu\Blcu\src\Fronted> npm run serve

```
> demo0326@0.1.0 serve
> vue-cli-service serve
```

INFO Starting development server...

DONE Compiled successfully in 15829ms

App running at:

- Local: <http://localhost:8080/>
- Network: <http://10.19.60.238:8080/>

Note that the development build is not optimized.
To create a production build, run `npm run build`.

(图 1 Web 前端部署成功图)

● Web 后端部署:

```
D:\python\python.exe D:\pycharm2023\flask_server\Blcu\Blcu\src\app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 611-931-289
```

(图 19 Web 后端部署成功图)

4. 安装与使用

该项目的部署十分简洁，只需安装依赖、修改配置，即可完成最小化实例的部署。

4.1 最低环境要求

CPU:E5 虚拟化双核

内存: 2GB

硬盘: 20GB(Linux)/40GB(Windows)

网络: 2Mbps

系统: Ubuntu18.04+

软件环境:Python3.8+、Nginx、Redis

4.2.1 目录结构

具体代码详见附件中的代码包，以下仅给出二级目录说明。

—Docs	# 存放设计文档
—src—Fronted	# 存放前端代码
—app.py	# flask 后端程序
—backend.py	# 传感器后端程序
—Common.py	# 通用类
—config.py	# 配置文件
—Dbconn.py	# 数据库驱动程序
—fronted.py	# 传感器前端节点程序
—GetData.py	# 数据库驱动程序的封装
—init.sh	# 数据库建设脚本
—README.md	# 项目手册

4.3 快速入门

4.3.1 安装依赖

进入源码包目录，执行命令

```
python3 -m pip update
```

```
Python3 -m pip install -r requirements.txt
```

4.3.2 修改配置

打开配置文件 config.py

■ 配置 mysql:

host: 设为数据库服务器的 IP 地址，通常设为 localhost

port: 设为提供数据库服务的端口号，通常为 3306

username: 设为数据库的登录账户

password: 设为登录密码

■ 配置 redis:

db1: 设为 1

db2: 设为 2

host: 设为 Redis 的 IP 地址，通常设为 localhost

port: 设为提供 redis 服务的端口号，通常为 6379

■ 配置 EMQ X

ip: 设为 EMQ X 服务器的 IP 地址，通常为 127.0.0.1

port: 设为提供 EMQ X 服务的端口号，通常为 1883

■ 配置消息推送的 sendkey

Sendkey: 设为推送手机的的 sendkey

4.3.3 启动数据库

启动本机的 MySQL 数据库

4.3.4 启动前端

进入前端目录，打开终端，输入 `npm run serve`

4.3.5 启动后端

进入 src 目录，打开终端，输入 `python3 app.py`

4.3.6 启动 EMQ X 服务器

本机打开终端，输入 `emqx start`

4.3.7 启动传感器前端

进入 src 目录，打开终端，输入 `python3 fronted.py`

4.3.8 启动传感器后端

进入 src 目录，打开终端，输入 `python3 backend.py`

第五章 测试报告

1. Web 高并发设计

对于获取实时数据这类高实时、高并发的 API，后台使用了 Redis 缓存。平峰通过 Crontab 定时执行，高峰自动缩短缓存周期，尽最大程度减少磁盘 I/O，提升网站响应速度。

对于历史记录的自动归档，使主数据库保持在较小规模，提升查表性能。

2. Web 性能测试

本次使用权威第三方测试平台—阿里云 PTS 平台对 We 接口进行压力测试。测试时间：2023-04-11 20:39:06。

平台运行于阿里云 2 核 4G 服务器，对最高频的三个 API 进行 50 用户并发测试，平均请求成功率 $\geq 98\%$ 。可见平台在轻量化部署场景下仍有较高性能表现，便于企业单位私有化部署该平台，节约运营成本，保障敏感数据安全。



(图 20 测试报告图)

3. Web 安全测试

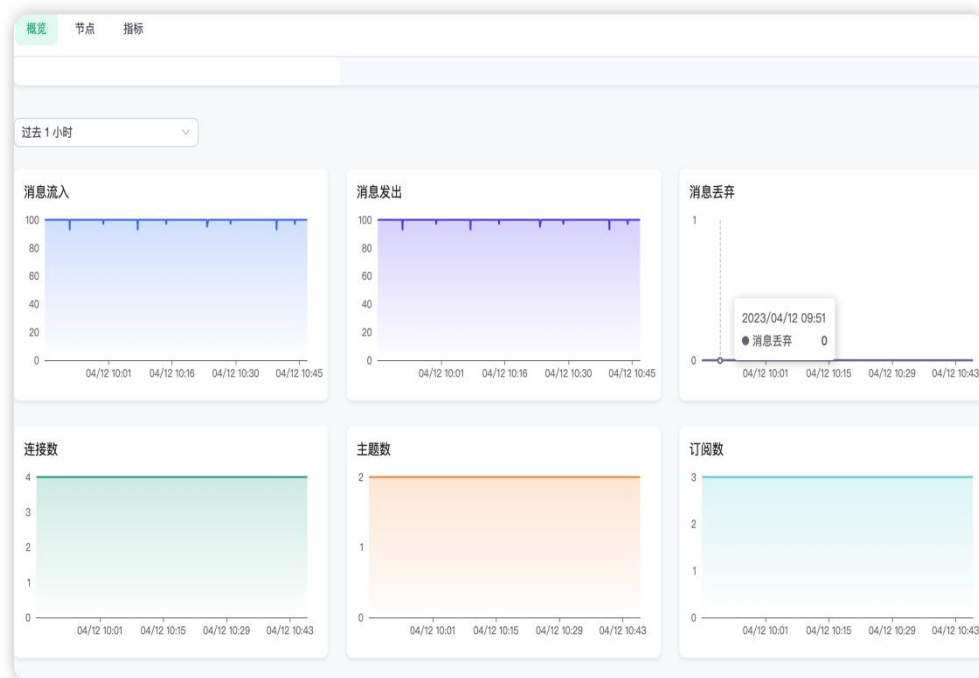
平台面向实际使用，由于作物环境以及作物信息的敏感性，团队对平台的安全做了针对性优化，各种漏洞的防护水平如下。

是否防护		防护策略
SQL 注入	是	使用 SQL 参数化查询，彻底杜绝该问题
XSS	是	先转义 HTML 字符在存入数据库
SSRF	是	部署时置于 Docker 环境中
模板注入	是	纯前后端分离，不存在 SSR，不存在该问题
未授权访问	是	完善的密码鉴权流程

(图 21 漏洞防护水平图)

4. 硬件系统测试

共安装 5 个节点进行测试，设备测试 10 天内，无设备离线情况，且数据丢失率极低。



(图 22 硬件测试结果图)

第六章 项目总结

1. 项目收获

本团队也是第一次尝试将物联网、人工智能、Web 结合起来，实现如此规模的一体化平台。最大的难度在于模块间的交互与配合。

而模块间的交互与配合，离不开三位成员的缜密合作、相互支持。经过一个多月的反复打磨，平台已较好地满足设计初衷，并有所创新和突破。同时，总结出几点经验：

- 熟练使用版本控制工具 git，进行版本回退，变化比较，提高解决问题的效率。同时要避免团队开发中的 git 冲突，如果要修改公共文件，提前沟通。

- 使用 API 管理工具，例如 APIFox，支持一键生成文档、Mock 调试、接口同步、自动测试等使用功能。

- 前后端开发需要提前指定需求文档，并以此制定接口文档。使得前后端得以同步开发。

2. 应用情况

本产品于 2023 年 4 月在山东省青岛市某大型农场单位安装使用，共为 5 个区域安装站点。负责监控各区域的环境情况。



(图 23 应用场景)

3. 使用前景

作为智能的作物环境监控系统，其实现了低人工，高智能，无人化管理地农场，且功能丰富。

农场环境检测系统可以大大提高农业生产的科学性、精准性和效率性，有着广泛的应用前景和市场需求。