基于WebGL的3D物联网智能监测-控制平台

摘要

随着“物联网+”概念的发展，物联网在各行业与生活中的应用也越来越广泛。然而，当前存在的物联网软件平台在庞大、专业化的大规模软件平台与小型软件之间存在断层，因此中小规模的物联网应用常常难以找到合适的软件平台，只能在收费高昂且较为臃肿的大型物联网平台与功能不足、配置繁琐的小型软件之间做出选择。而中小型的物联网应用实际上又占据了巨大的市场，因此开发一套适用于中小规模物联网应用，操作简单、功能强大的轻量级物联网平台在各领域都具有巨大潜力。

为解决这一问题，我们开发了一个涵盖前后端的轻量级通用物联网平台，以期待在中小规模的物联网应用中取代当前许多操作繁琐、臃肿且收费较高的大型物联网平台。该平台聚焦于聚焦于“轻量”、“通用”、“高效”、“易用”、“高度自定义”、“可扩展”，实现了一套较为完整的物联网应用解决方案。该平台对多种物联网协议提供了较为完善的支持，并支持复杂的可自定义操作，例如定时传感器监听、实验室安全控制、智能恒温恒湿等。

该平台由三大部分组成，分别是后端控制平台、网页端控制台与3D模拟观测平台。该平台有两大核心亮点，分别是后端控制平台的事件系统，支持通过图形化界面拖拽自定义较为复杂的设备交互操作，与3D模拟观测平台，以支持更直观的可视化设备监控。

在后端控制平台层面，我们引入了对设备的高层次抽象，使该平台不再局限于对物理设备的管理，同时还能与模拟量、网络API、外部数据库甚至程序脚本进行交互，以使得当前许多繁琐的操作能够在同一抽象层进行，简化了交互与配置成本，具有很高的自定义能力与高可扩展性。后端控制平台主要基于PyQt进行UI开发，对常用的物联网协议有较为完善的支持，在数个月的测试下仍能保持长期稳定运行，具有较高的可靠性。

此外，考虑到当前大部分物联网平台缺乏较为直观的可视化支持，我们在后端控制平台之外加入了一个可直观观察数据的可视化前端以显示相关图表，并使用WebGL技术构建了3D模拟观测平台，以直观地显示物联网设备的实际位置并进行实时监控，这使得对物联网设备的管理能够以更加直观的方式进行。控制台与3D模拟观测平台均使用基于Express.js开发的后端进行数据交互，其中控制台的前端界面使用Vue开发，3D模拟观测平台的前端界面使用WebGL库Three.js开发，并支持灵活的外部模型导入，具有较好的可扩展性。

关键词：物联网；传感器监测；数据可视化；WebGL；3D建模

目录

第一章 需求分析

1.1 行业现状

当前随着“物联网+”概念的不断发展与行业需求的不断增长，各领域对稳定可靠的物联网监测平台的需求越来越大，同时由于物联网技术的不断普及化，各领域对于物联网平台的易用性要求也越来越高。然而，当前来说，在专业大型物联网平台和小型传感器连接软件之间还存在较大的空缺有待填补，缺少稳定可靠、功能足够的轻量级物联网平台，而这恰恰是当前许多物联网应用场景所迫切需要的。

于是，为了解决当前流行的物联网应用软件平台普遍存在的操作繁琐、比较臃肿、订阅成本较高等问题，我们开发了一个涵盖前后端的轻量级通用物联网平台，以期待在中小规模的物联网应用中取代当前许多操作繁琐、臃肿的大型物联网平台。此外，我们还加入了3D模拟观测系统，以实现清晰直观的设备监控。

1.2 市场分析

物联网软件平台前景广阔，当前随着物联网应用的不断普及化，对易用稳定的物联网软件平台的需求将会越来越大。而对于中小规模的物联网应用，尤其是主要涉及传感器监测的应用来说，臃肿庞大的传统物联网平台由于订阅费用高、操作繁琐等原因将不再是很好的解决方案，而会转向对轻量化平台的需求。

1.3 产品简介

我们的平台由后端控制平台、网页端控制台、3D模拟观测平台三大部分组成，涵盖了物联网应用部署过程中的整个前后端过程。

我们的后端控制平台使用PyQt进行UI开发，支持设备的增删操作，可以对设备源、传感器数据处理、设备报警等进行详细的配置，也支持将读取到的数据直接上传至数据库。同时也支持动态读取传感器数据，并实时显示超时率和平均响应时间，以进行便捷的调试。也支持对多种不同的传感器进行轮询监听并将数据实时上传至服务器，支持多线程与高并发，时延低，稳定性高。此外，我们还引入了事件系统设计，可以通过拖拽操作便捷生成可执行的事件脚本，以进行更为复杂的操作，例如定时监听、实验室安全控制、智能恒温恒湿等，同时这些脚本可以保存后重新导入程序继续调试。最后，后端控制平台还提供了一系列小工具以简化操作流程，例如快速建表工具和数据库预览工具。

我们的网页端控制台主要使用JavaScript框架Vue开发，其后端采用Node.js框架Express.js编写。其主页面负责展示重点监测数据、警告信息以及相关统计数据；数据管理页负责历史数据的显示与查询操作；而设备管理页负责对传感器信息进行可视化展示。通过网页端控制台，许多工作不再需要运维人员线下调试服务器，只需通过网络即可进行一定程度的设备管理，并能够实时查看可视化数据。

我们的3D模拟观测平台基于WebGL框架Three.js开发，可通过网页端控制台的“查看模型“入口直接进入，其后端由Express.js编写。通过3D模拟观测平台，我们可以从多个视角直观地看到传感器与执行器的位置并进行实时数据监控。此外，3D模拟观测平台的房间模型与桌椅等装饰可以通过外部文件直接导入，具有很高的可扩展性。

我们构建的不仅仅是单一的物联网设备连接软件，而是可以支持复杂物联网应用的完整平台，兼具小型软件的小巧轻量和大型平台较为完备的功能，以期待在实际应用替代当前一些中小规模的物联网应用中收费高昂、较为臃肿的大型软件平台的使用场景。

第二章 特色与创新

2.1 基于事件系统的复杂设备交互

通过为后端控制平台加入事件系统，该平台可以真正实现复杂的设备交互操作。事件系统支持通过图形化界面拖拽生成高度可自定义的事件脚本，并配套有相应的调试功能，可将事件脚本保存成独立的文件后再次导入调试。事件系统同时支持运行多个事件脚本。通过合理的配置，事件系统可以适应但不局限于以下场景：传感器定时监测、实验室安全控制、智能恒温恒湿、智能流水线控制等。

2.2 直观的3D模拟观测平台

通过加入3D模拟观测平台并与后端控制平台进行联动，实现了对设备信息的实时监控与直观展示。3D模拟观测平台也支持通过拖拽操作直接调整传感器与执行器的位置，并支持导入外部模型生成新的房间，具有较高的通用性。

2.3 良好的可扩展性

通过加入多层抽象与设备源的概念，该平台不再仅限于能够与物理设备进行交互，而是能够将广义上的任何传感器、执行器与控制器都纳入平台的管理范围。例如任何能够获取信息的数据源都可以作为传感器，如可以通过一个能够获取当地温湿度数据的网络API代替实体温湿度传感器；任何能够响应操作的对象都可以作为执行器，例如可以使用一个能够向微信发送报警提醒的接口作为执行器对接传感器。如遇到该平台尚未支持的物联网协议，也可以通过对接外部脚本以实现支持。

2.4 较为完整的解决方案

通过后端控制平台、网页端控制台与3D模拟观测平台的集成，该平台实现了一套涵盖前后端的较为完整的解决方案，从后端设备的对接到可视化信息的展示都有涉及，可以一定程度上做到“开箱即用”，而无需对接其他软件。

第三章 概要设计

3.1 系统概观

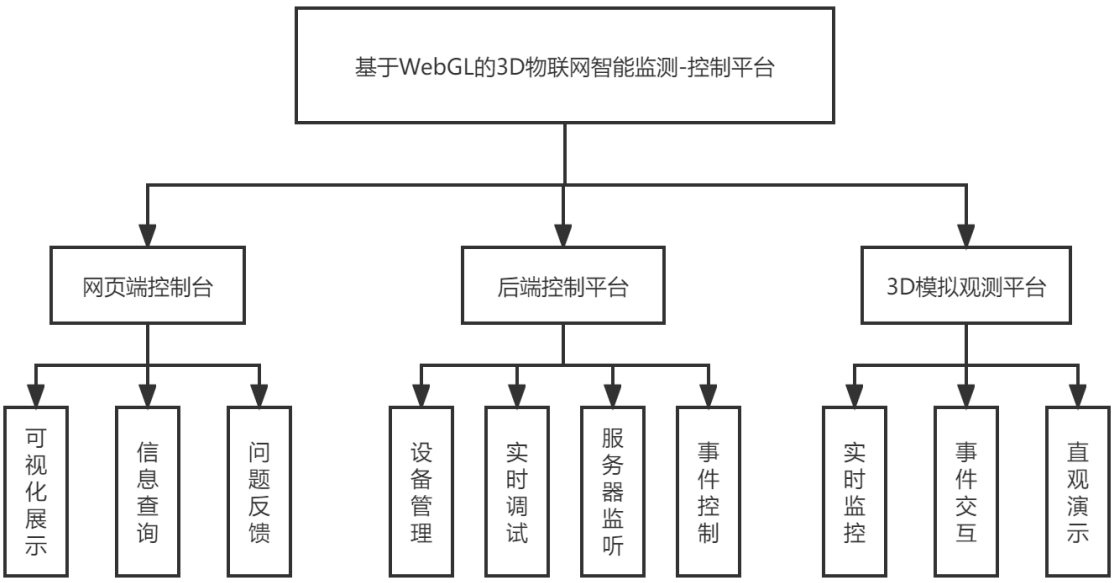


图3-1 系统概观

该平台的系统概观如图3-1所示。该平台主要包含三个部分：后端控制平台、网页端控制台与3D模拟观测平台，分别负责设备管理与交互、可视化展示与数据查询，和直观的实时信息监控。

在后端控制平台层面，我们进行了硬件抽象，将所有物联网设备抽象为传感器、控制器、执行器三种，以对设备交互操作进行统一，提高平台的可扩展性与通用性。在这一抽象层级，传感器指代一切可用于获取数据的数据源，包括但不限于物理设备（如温湿度传感器）、模拟设备、网络API、外部数据库以至于程序脚本；执行器指代一切可用于相应操作的对象，包括但不限于物理设备（如继电器）、模拟设备、网络操作、外部数据库以至于程序脚本；控制器负责对接传感器与执行器，例如定时向执行器中写入传感器接收的数据、进行逻辑运算、条件判断等。

后端控制平台具有设备管理、实时调试、轮询监听、事件控制等多种功能，可实现一个完整的后端监控平台，支持主流传感器协议，并支持高度可自定义。

在网页端控制台层面，由于数据展示功能需要适应不同用户共同使用的需求，我们将其部署到网络服务器，保证同时异地多人使用并查看数据。另外还带有权限分级功能，将普通的数据查询用户和网页、传感器维护与管理人员的视图进行划分，最大程度实现控制台功能的精准实现。

3D模拟观测平台为设备管理与数据查询提供了更直观的方式，使得运维人员可以对设备情况有更为直观的把握，实现更好的事件交互与实时监控。

3.2 设计目标

- 实现物联网设备集成控制系统
- 实现网页端可视化信息控制台
- 构建网页端3D模拟观测平台

3.3 系统架构

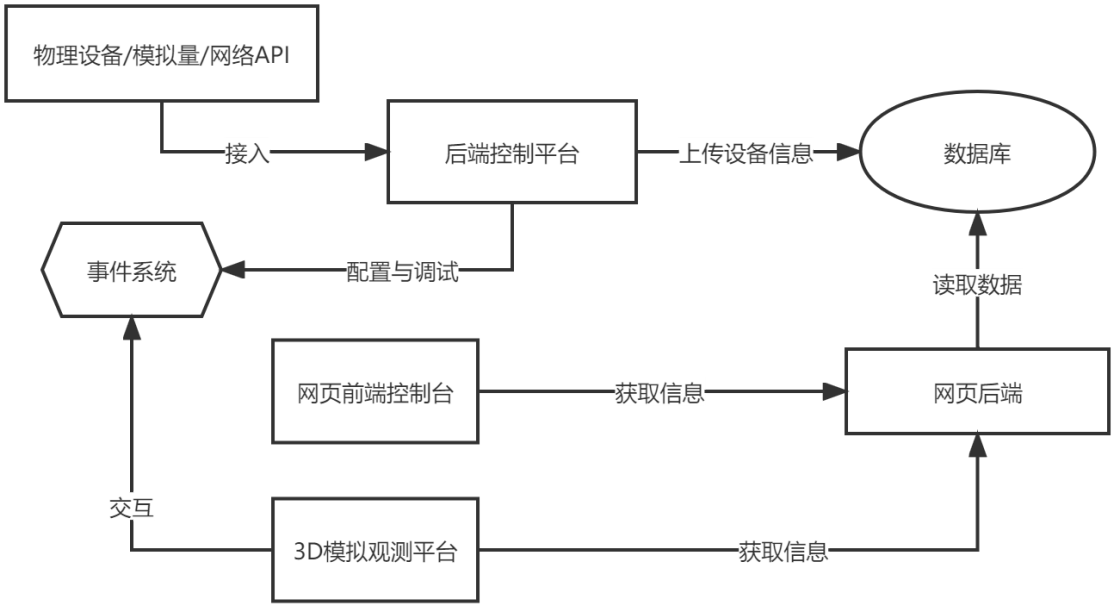


图3-2 系统架构

系统整体架构如图3-2所示。后端控制平台与后台数据库有直接交互，实时将设备信息直接上传至数据库。而网页端控制台与3D模拟观测平台通过REST API向同一个服务器后端发送请求并得到数据，而服务器后端则与数据库进行交互并返回相应数据。

第四章 系统实现

4.1 后端控制平台

在后端控制平台层面，我们将所有物联网设备抽象为传感器、执行器、控制器三类，分别负责数据获取、执行相应操作以及进行逻辑运算三大功能。

为了使得传感器接收的数据与执行器能够涵盖的对象进一步拓展，我们引入了设备源的概念。对于传感器来说，设备源是对一切可获取信息的数据源的抽象，通过物理设备接受的信息、模拟量、网络API甚至程序脚本都可以作为设备源。例如一个具体的LoRa温湿度传感器可以作为一个设备源，通过网络API读取的温湿度数据可以作为一个抽象的设备源。对于执行器来说，设备源是对一切可响应操作对象的抽象，可响应命令的物理设备、模拟设备、网络API甚至一个可运行的程序脚本都可以作为设备源。例如一个具体的继电器可作为设备源，一个可向微信发送报警的接口也可作为设备源。

后端控制平台共有五个标签页，分别用于实现不同的功能。

4.1.1 设备管理页

设备管理页负责进行设备的详细配置，支持增加/删除设备、重命名设备以及调整排列顺序。对于传感器而言，可以进行设备源设置、默认动作设置、数据处理设置、通用设置、报警设置共五个配置部分。对于执行器而言，可进行设备源设置、默认动作设置与通用设置共三个配置部分。

4.1.1.1 设备源设置

图4-1 设备管理页-设备源设置

对于传感器来说，设备源是传感器的原始数据来源。设备源可以是一个能够读取数据的物理设备，例如温湿度传感器或光线传感器；也可以是一个抽象的数据源，例如外部数据库、网络API等；也可以是一个模拟数据源，例如简单正弦函数、随机模拟量等。

对于不同类型的设备源可以进行详细的配置。例如使用Modbus-TCP协议的物理设备源需要配置其端口、波特率、地址、超时，使用正弦波模拟量作为设备源需要设置其频率、振幅等。

对于执行器来说，设备源意味着可以响应操作的对象，例如以继电器为代表的物理设备，以POST API为代表的抽象操作对象等。

4.1.1.2 默认动作设置

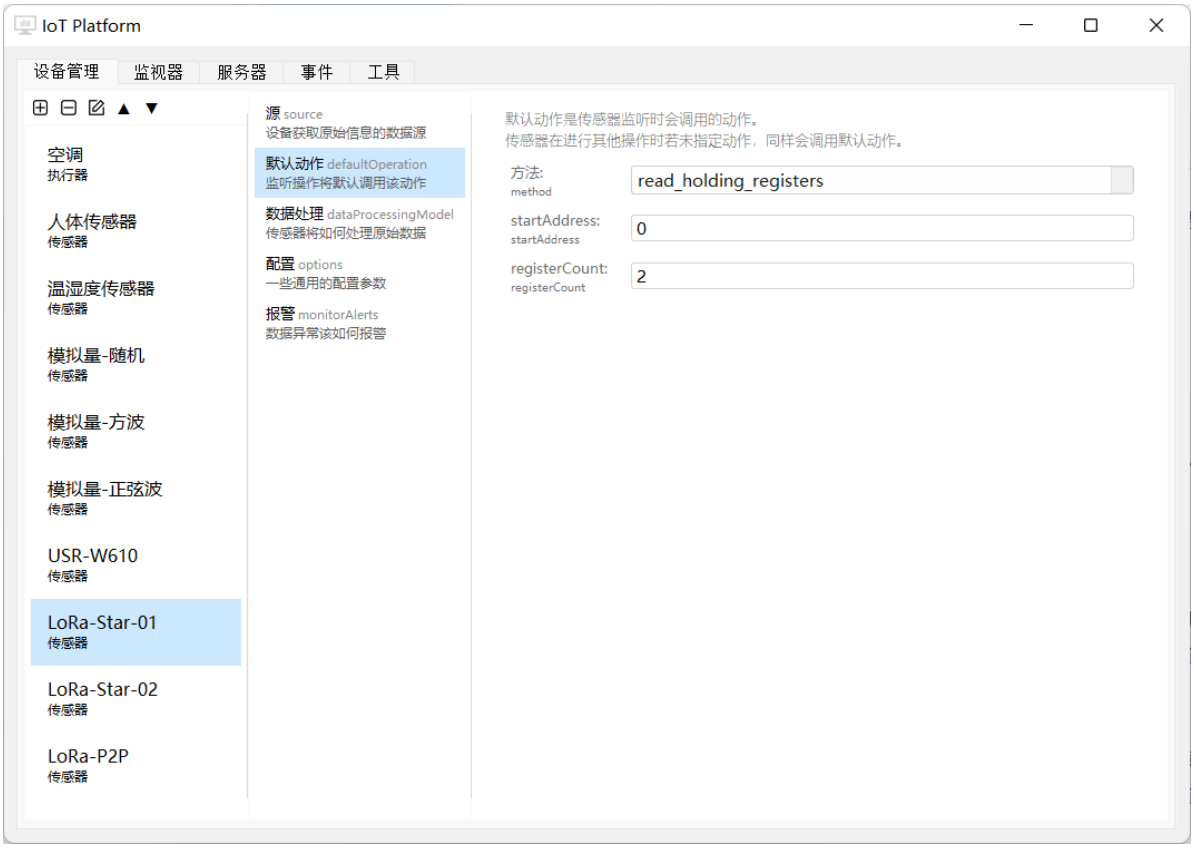


图4-2 设备管理页-默认动作设置

对于传感器来说，默认动作是传感器监听时默认调用的操作，服务器轮询监听即调用该操作读取原始数据。例如对使用Modbus协议的传感器来说，默认动作可以是读取线圈、离散寄存器、输入寄存器等；对于网络API来说，默认动作可以是一个GET/POST请求。

对于执行器来说，默认动作是其响应命令时默认调用的操作，事件系统中的“激活执行器”调用的就是该操作。例如对使用Modbus协议的传感器来说，默认动作可以是写入线圈、离散寄存器、输入寄存器等；对于网络API来说，默认动作可以是一个GET/POST请求。

4.1.1.3 数据处理设置

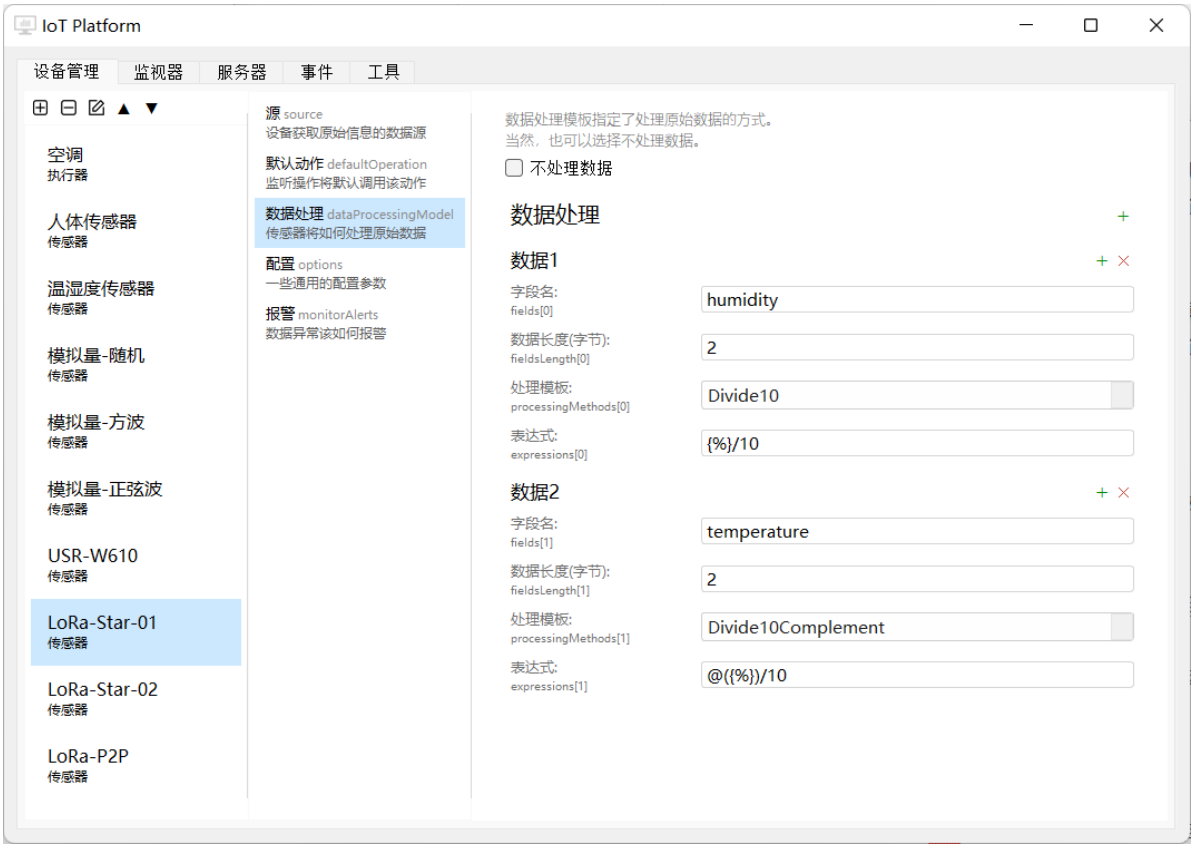


图4-3 设备管理页-数据处理设置

数据处理设置决定了传感器读取到的十六进制原始数据将以何种方式被处理。未配置该设置时，默认的处理方式是直接输出数据而不进行处理。

在数据处理设置中，可以将读取到的数据分为多个字段，并设置每个字段所占字节数，并指定对每个字段的具体处理方式。例如对于一个返回四字节十六进制数据的温湿度传感器，可以将其分为两个字段"湿度"和"温度"，分别占两个字节，对湿度进行除以十操作，对温度进行取补码再除十的操作（因为温度可能是负值）。数据处理设置支持高度可自定义，可以执行任何合法的Python代码。

仅传感器可配置这一设置。

4.1.1.4 通用设置

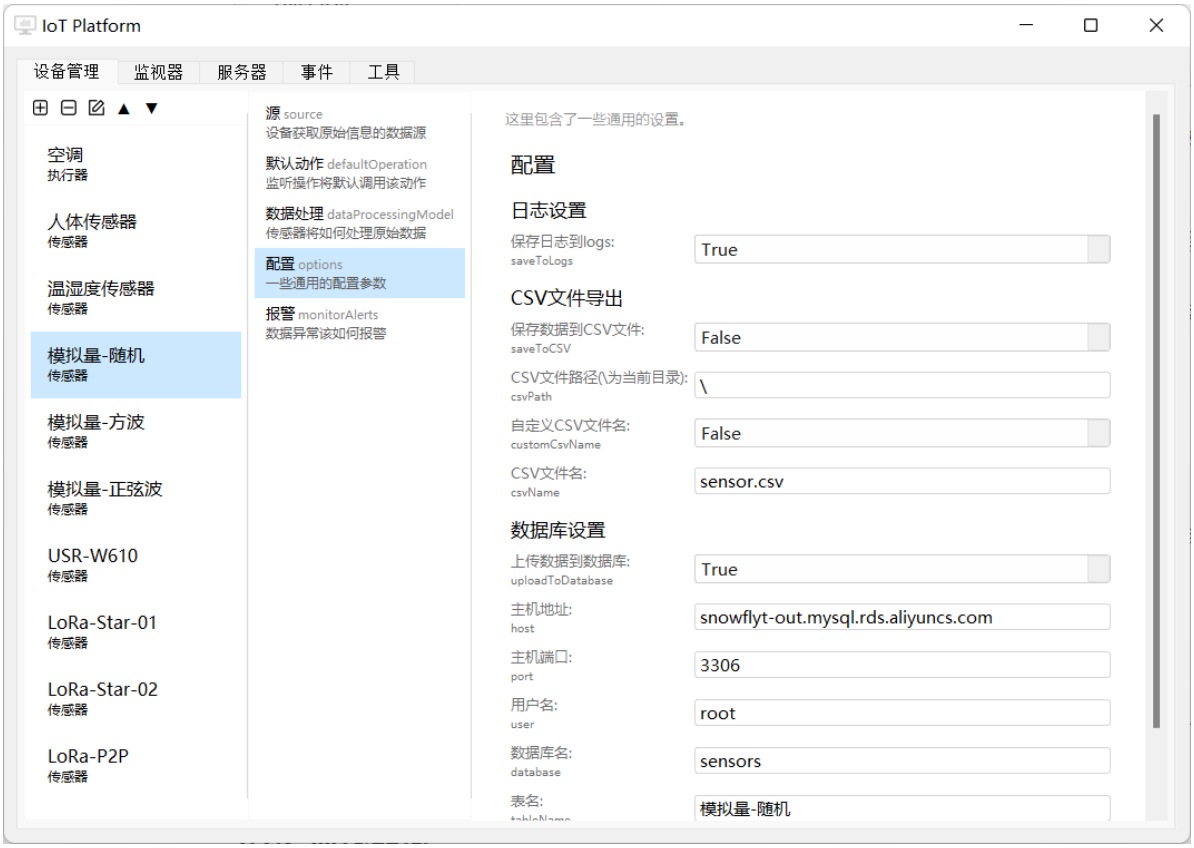


图4-4 设备管理页-通用设置

通用设置中包含了一些常见的设置项。例如是否输出到日志文件，是否生成csv文件，是否上传至数据库以及有关数据库的参数设置。将数据上传至数据库是该平台进行前后端信息交互的主要方式。

4.1.1.5 报警设置

图4-5 设备管理页-报警设置

在报警设置中可配置多个报警项以及报警条件、报警方式（单次触发、连续多次触发、周期内多次触发）等。以温湿度传感器为例，可以将”连续一分钟内读取到的温度大于35度超过十次“设置为一个报警项，将”温度超过四十度触发一次“设置为另一个报警项。报警数据同样会上传至数据库供前端或其他接口调用。

仅传感器可配置这一设置。

4.1.2 监视器页

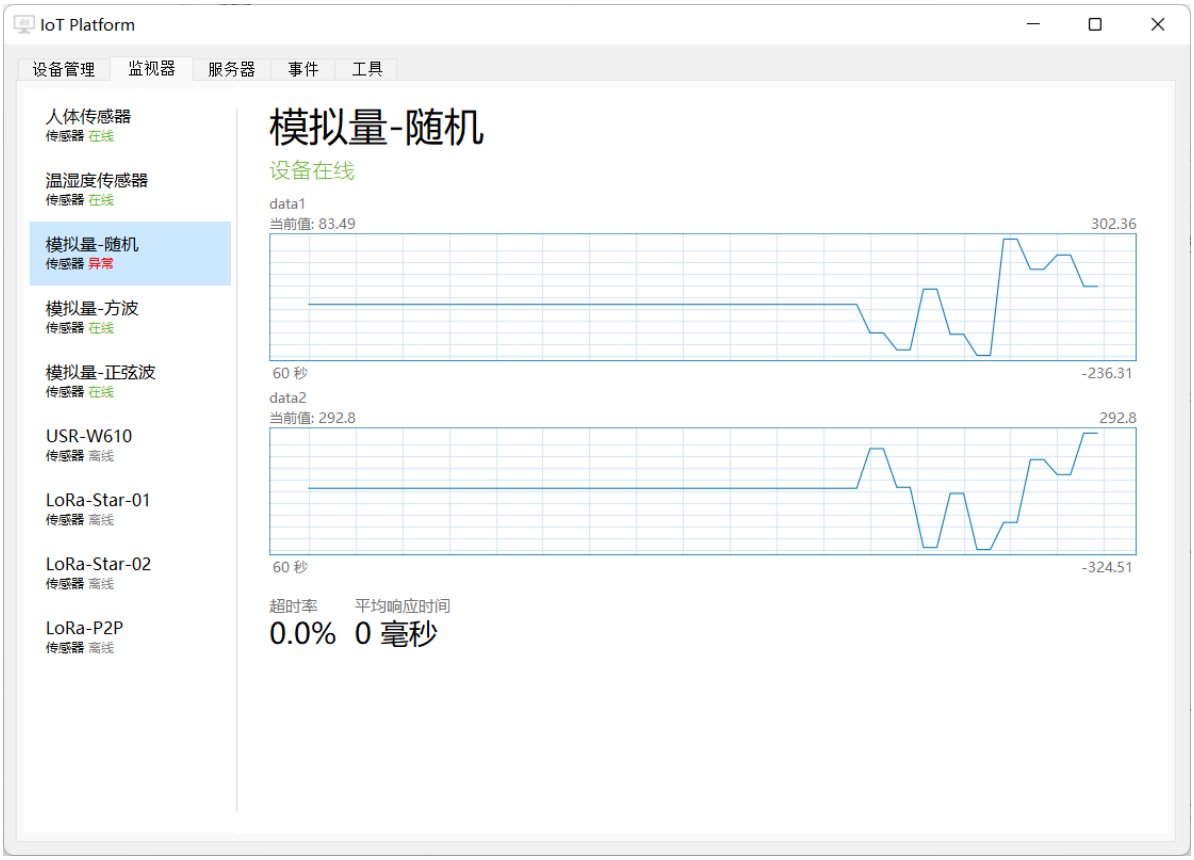


图4-6 监视器页

监视器页可以动态连续读取传感器数据（每秒一次），显示实时数据图像，并实时显示超时率和平均响应时间。监视器页可用于配合设备管理页用来进行便捷的调试，也很适合进行物理设备调试。

4.1.3 服务器页

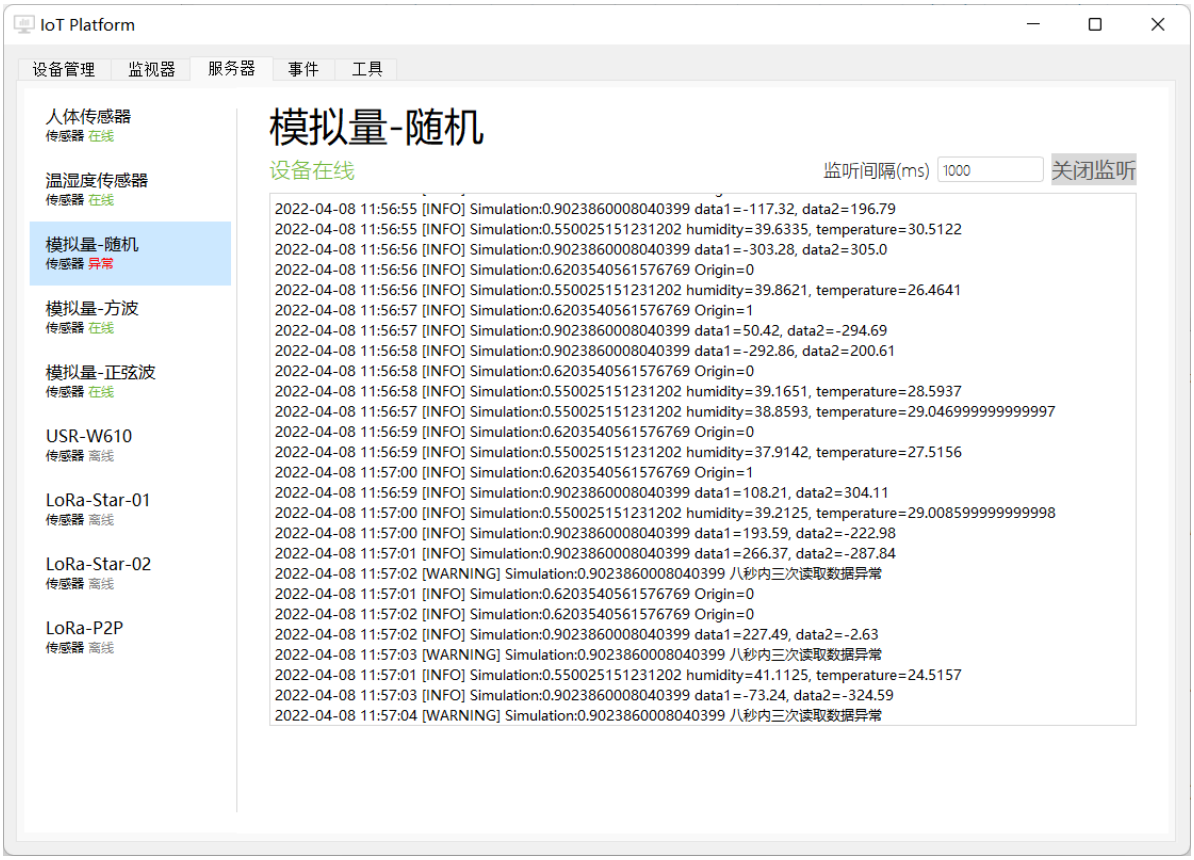


图4-7 服务器页

服务器页可以对多种不同的传感器进行轮询监听并将数据实时上传至数据库，支持多线程与高并发，时延低，稳定性高。支持对每个传感器设置不同的循环监听周期。同时服务器页还支持与事件页联动进行更为复杂的监听操作。

4.1.4 事件页

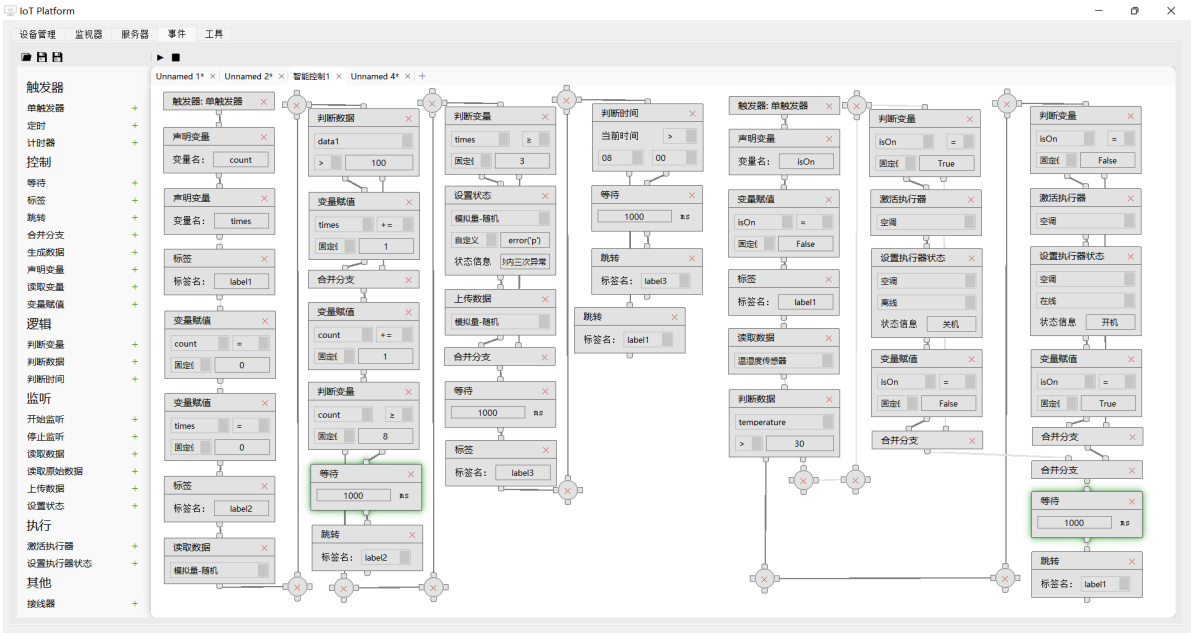


图4-8 事件页

事件页支持通过拖拽操作便捷生成可执行的事件脚本，以进行更为复杂的操作，例如定时监听、智能恒温恒湿、实验室安全控制等。同时这些脚本可以保存后重新导入程序继续调试。这也是我们整个后端控制平台开发过程中耗时最长也是投入最大的部分，同时也是该项目的主要亮点之一。

图4-8演示的事件脚本有两个功能。第一个功能是在每天早上八点至晚上十二点反复监听一个传感器，当读取到的数据大于100时在八秒内超过三次时进行一次报警并上传数据；第二个功能是反复监听一个温湿度传感器的温度数据，当温度大于三十度时开启空调，否则关闭空调。

事件页支持同时编辑多个事件脚本并且支持多脚本同时并发执行。事件脚本还可保存为JSON文件以供后续再次导入继续调试。

我们考虑将来使事件系统支持更多操作，例如集成更完善的数据处理操作等。同时我们也考虑优化相关的调试工具，如支持断点调试等。

4.1.5 工具页

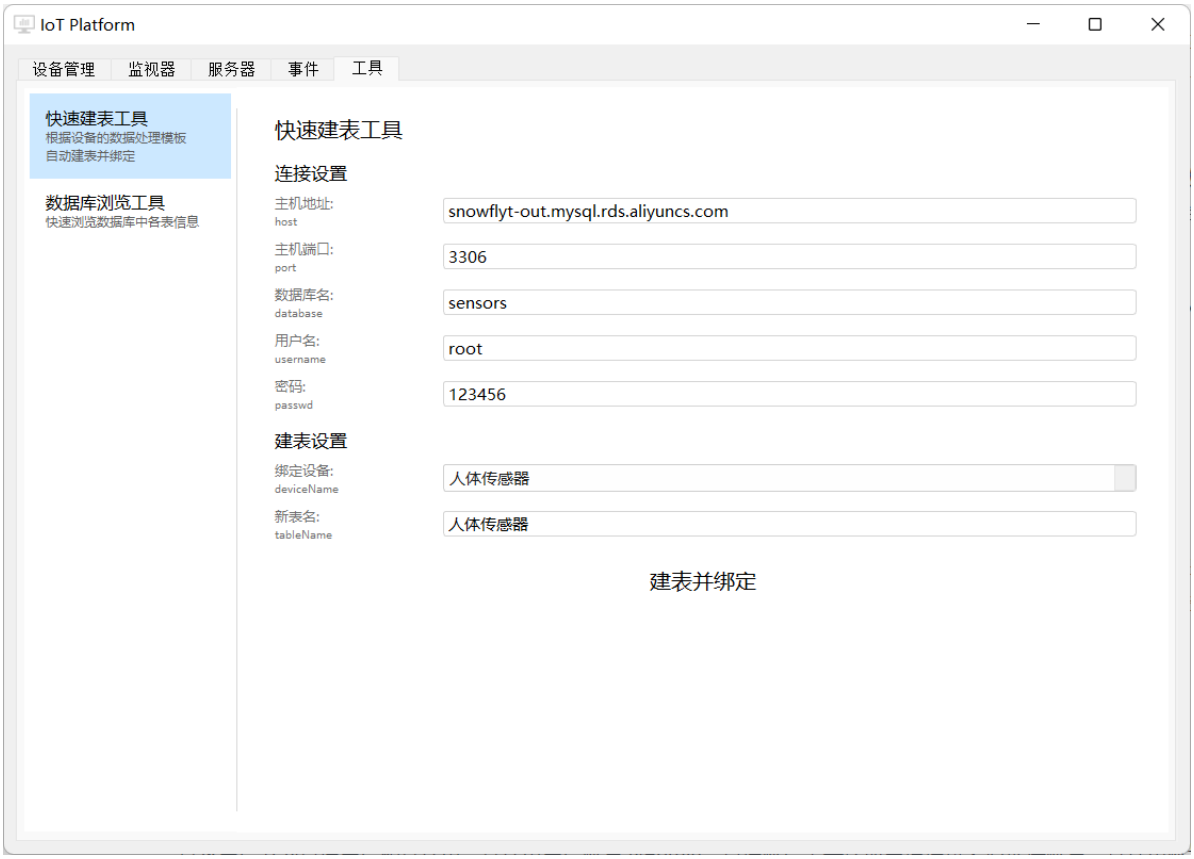


图4-9 事件页

工具页提供了一些利于简化操作的小工具，例如快速建表工具可以根据设备管理页中的设备配置在数据库中快速建表并绑定，降低了运维人员的心智负担。数据库浏览工具可以对数据库中的各个数据表进行简单的预览，可实时察看数据是否已上传至数据库。

4.2 网页端控制台

4.2.1 登录页

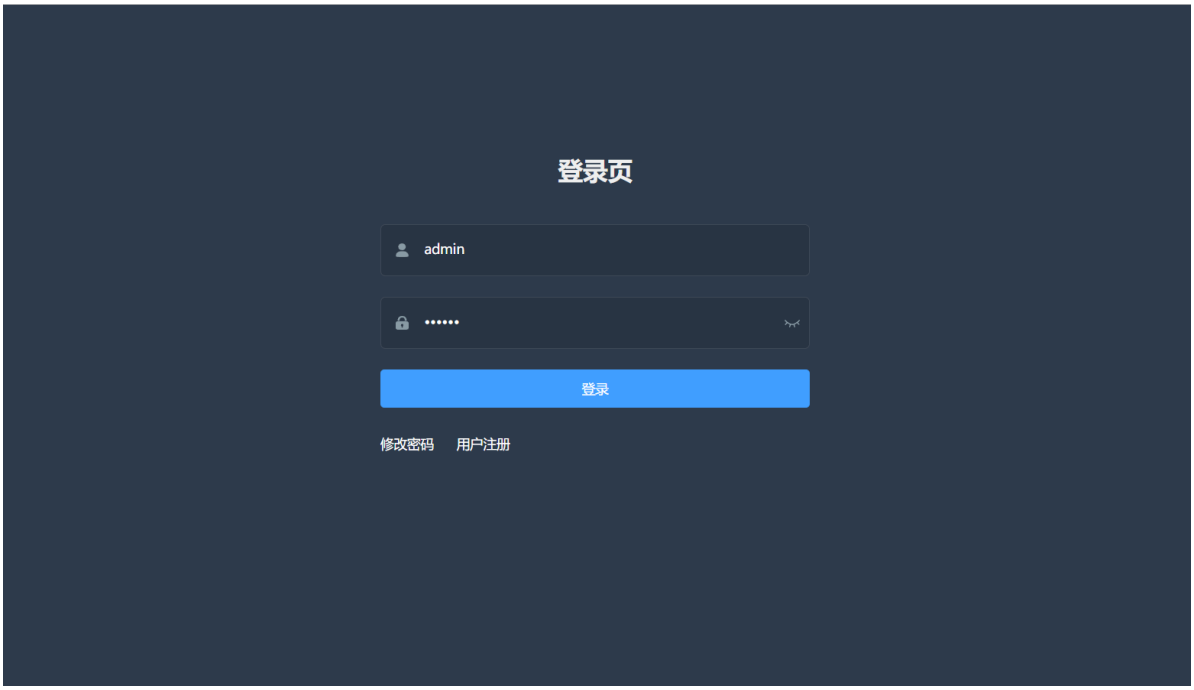


图4-10 登录页

登录用户分为普通用户和管理员，管理员用户账号为admin，普通账户需要注册申请通过之后提供账号。管理员账号拥有对传感器、故障日志进行编辑功能，而普通账号暂时只有数据查看、故障提交功能。

4.2.2 主控制台

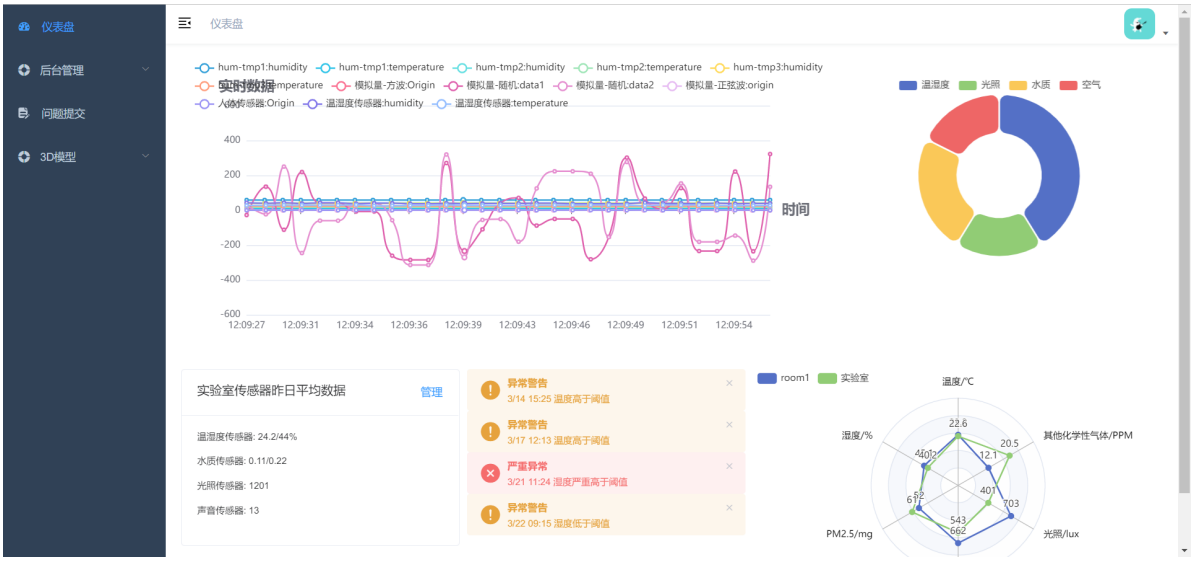


图4-11 主控制台

登陆成功会进入主控制台，对所有用户开放。主控制台的主要功能有四个：传感器数据实时监控，昨日传感器数据，报警日志以及设备统计。

4.2.3 数据及设备管理

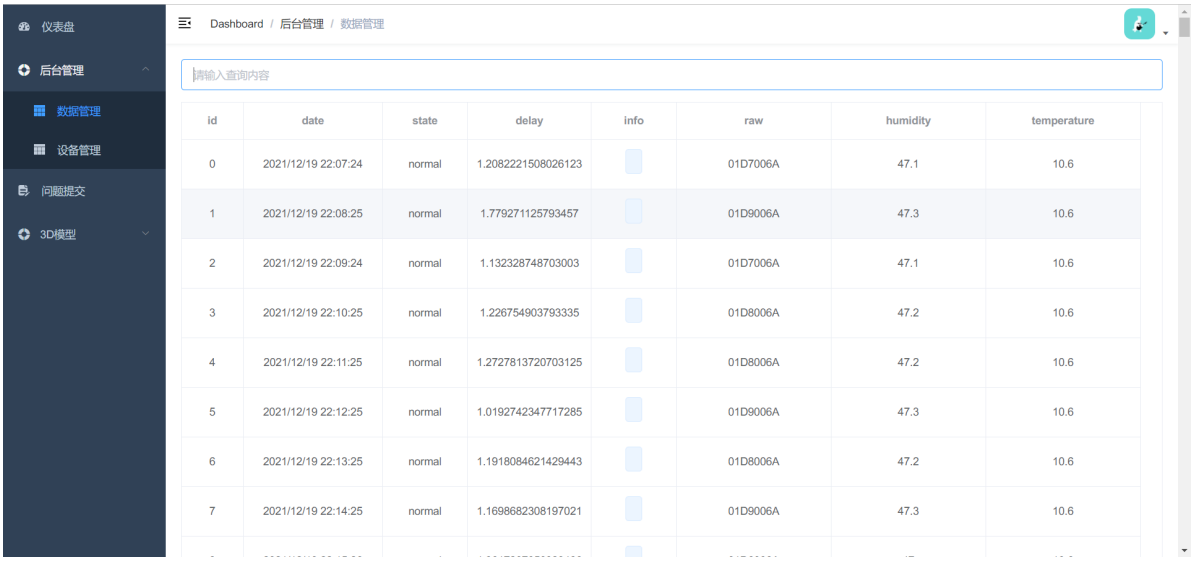


图4-12 后台管理

后台管理中的数据管理页可对传感器数据进行查询。设备管理页支持设备功能、型号查询服务。这两项功能目前不对低权限用户开放。

4.2.4 故障提交

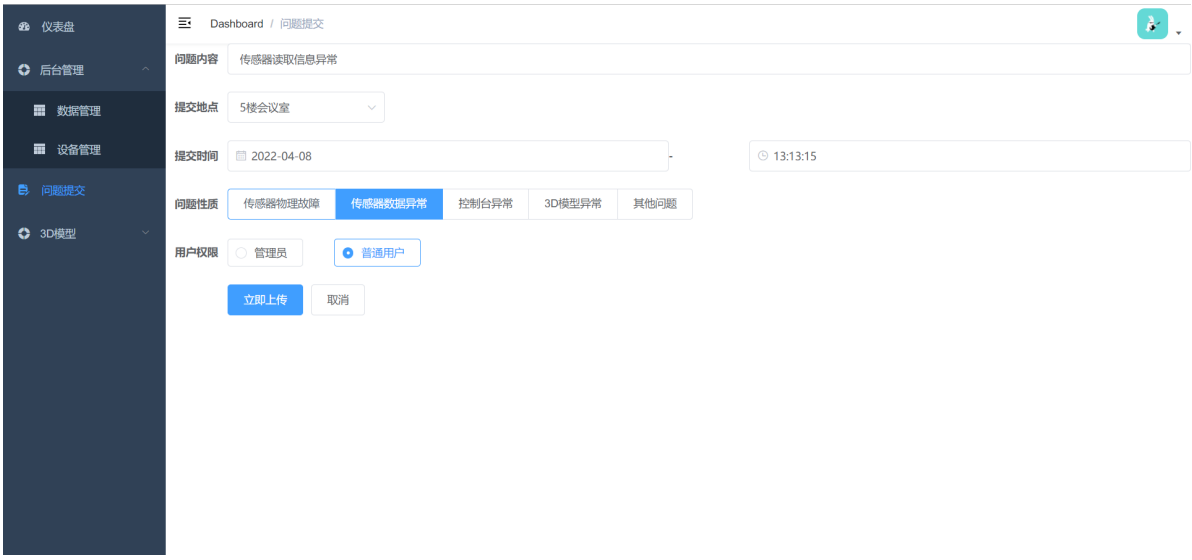


图4-13 故障提交

当用户使用控制台遇到错误或发现异常数据时，可以使用控制台的故障提交功能。故障提交功能将提供一个故障申报表，需要填写具体问题、时间、用户类别、用户类型等信息，提交之后将存储在服务器日志文件内，管理员可对日志出现的问题进行处理。

4.3 3D模拟观测平台

3D模拟观测平台使用WebGL框架Three.js进行建模，在网页端部署。房间模型、桌椅等装饰物可通过外部模型文件进行导入。传感器使用白色方块表示，而执行器使用蓝色方块表示。用户可以对传感器、执行器进行拖拽以调整位置，或在右上角精细调整其坐标。同时也支持通过数据库对传感器/执行器进行动态的添加与删除操作。用户也可直接通过鼠标拖拽调整视角进行移动。

4.3.1 传感器实时监测

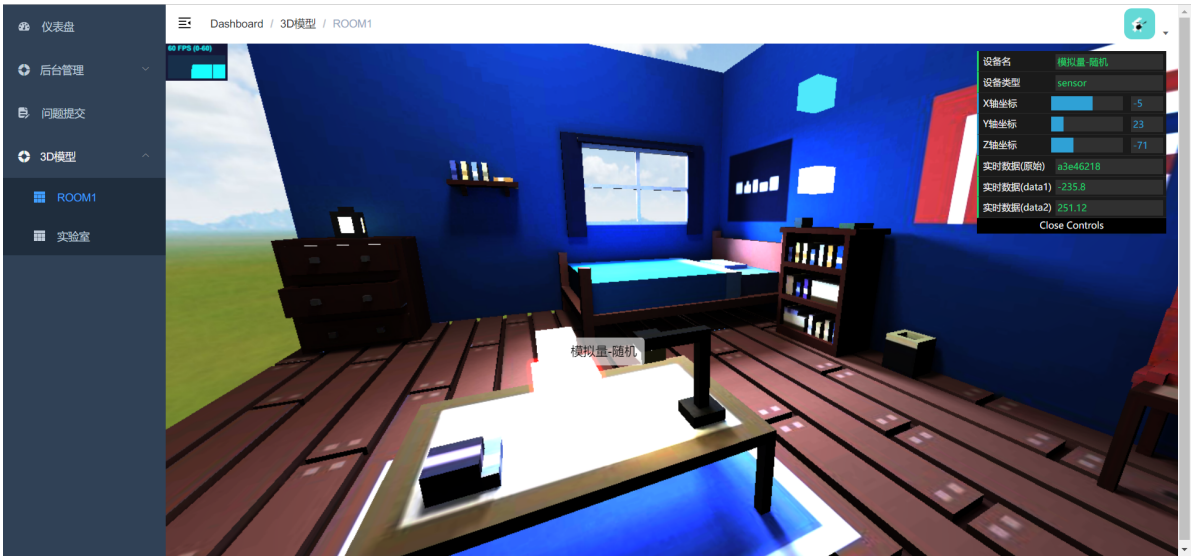
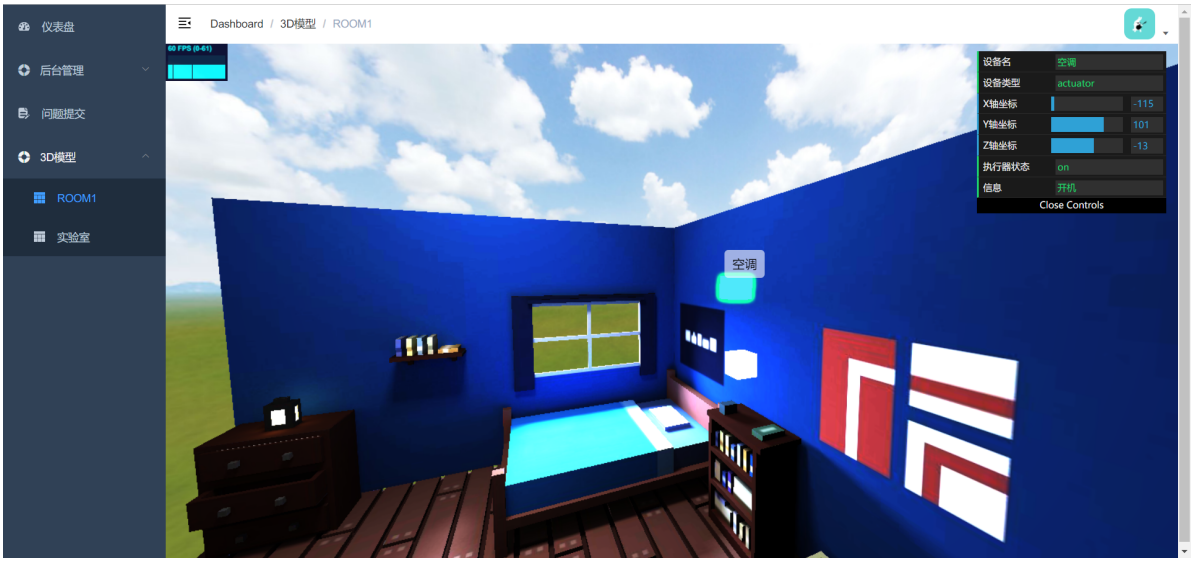


图4-14 传感器实时监测

3D模拟观测平台将实时读取传感器数据。当传感器读取数据时，其模型将闪烁绿光，当数据异常时将闪烁红光，所有展示的传感器将同时进行监测。当选中某一个传感器时，右上角将显示传感器当前读取到的详细信息，并会实时进行动态变化。

4.3.2 执行器状态监控



3D模拟观测平台也将实时读取执行器状态。当执行器在线时，会持续闪烁绿光，发生故障时持续闪烁红光，离线时则不发光。当选中执行器时，右上角将显示执行器的当前状态与状态信息。

4.3.3 多场景监控

用户可以同时添加多个不同的房间模型进行设备监控。例如这里添加了一个实验室模型以模拟实验室安全控制。

第五章 测试报告

5.1 功能测试

5.1.1 设备连接设置

测试内容包括：

1. 确认每种被支持的设备都可以正常连接
2. 测试是否每种设备被支持的操作都可以正常读取数据与执行命令
3. 测试在连接中是否有意外崩溃的情况发生

经测试，所有设备都可以正常连接，并且可以正常读取数据与执行命令，没有意外崩溃出现。

5.1.2 数据处理设置

测试内容包括：

1. 确认每种内置的数据处理模式被正常支持
2. 确认自定义数据处理模式可以正常工作

经测试，所有内置的数据处理模式都被正常支持，较为复杂的自定义数据处理模式也可以正常工作。

5.1.3 数据库连接测试

测试内容包括：

1. 测试数据库是否可以被后端控制平台正常连接
2. 测试传感器读取的数据是否可以实时上传数据库
3. 测试网页端控制台是否可以正常读取数据库数据

经测试，在数据库配置正确的情况下，总是能够将传感器读取的数据与执行器状态实时上传数据库，网页端控制台也可以正常读取数据库数据。

5.1.4 事件系统测试

测试内容包括：

1. 测试事件系统是否能够正常工作
2. 测试事件系统是否能够正常保存读取脚本
3. 测试事件系统是否能够同时运行多个脚本

经测试，事件系统能够正常实现设计功能，并且能够正常保存读取脚本，与支持异步运行多个脚本。

5.1.5 网页链接测试

测试内容包括：

1. 测试所有链接是否按指示的那样确实链接到了该链接的页面
2. 测试所链接的页面是否存在
3. 测试Web应用系统上是否有孤立的页面

经测试，网页端无错误链接，无孤立页面，所有页面均可以通过链接正常访问。

5.1.6 表单测试

对系统中各个功能模块的各项功能逐一测试，包括边界值测试、等价类测试以及异常类测试。整个系统中涉及表单提交的包括了用户权限、参数设置等模块。

经检测，用户提交表单时的信息时，前端程序会判断填写的信息是否出现格式上的错误。一旦发现错误，前端页面会阻止表单提交并提示用户对应的错误信息。

5.1.7 3D图形界面测试

测试内容包括：

1. 测试3D图形界面能够实时显示设备信息
2. 测试在3D场景中移动是否会出现异常
3. 测试读取数据与数据异常时传感器与执行器是否能够正常闪烁

经测试，在3D图形界面中能够流畅移动，并且可以实时读取设备信息，传感器与执行器也能够根据数据交互正常闪烁。

5.2 跨平台测试

我们将后端控制平台于Windows、Linux（包含多个发行版，如Centos和Ubuntu）、MAC OS上分别打包部署，经测试均能正常实现设计功能，并且没有出现因更换平台导致的明显性能下降。

5.3 性能测试

我们将后端控制平台部署于服务器上，进行了为期三个月的传感器监听测试。在测试过程中，所有数据均能够正常实时上传至数据库，并且没有出现异常状况。这证明我们的平台具有较高的稳定性。

5.4 可用性测试

5.4.1 整体界面测试

随机抽取10名用户，对后端控制平台、网页端控制台与3D模拟观测平台进行测试。

反馈结果显示整个系统保持了较高的易用性，并且能够支持较为复杂的设备交互操作。对于界面风格的涉及，整体反馈也较为满意，用户能够很快找到所需的功能，在简单了解如何使用后便能快速上手。

5.4.2 图形测试

经测试排查，系统页面中图形均有明确用途，无杂图或尺寸过大图。包括后端控制平台、网页端控制台与3D模拟观测平台在内的页面字体风格均一致，背景颜色与字体颜色和前景颜色搭配和谐。

第六章 安装与使用

6.1 网页端控制台与3D模拟观测平台使用

网页端控制台与3D模拟观测平台无需安装即可通过网址http://101.132.165.23/直接访问。

如需在本地使用，可分别解压“Web Server.zip”和“Web Client.zip”，在Node.js已安装并正确配置环境变量的条件下通过命令行在相应目录下分别执行“npm install”安装依赖，然后各自执行“npm run dev”分别启动网页服务端与客户端，即可在本地启动网页端控制台。

如需部署至服务器，需要对客户端运行“npm run build:prod”，打包后的静态文件会输出到dist/文件夹，可通过nginx等服务器工具进行代理；对于服务端，仅需将整个文件夹复制到服务器，在服务器上安装Node.js和pm2，直接运行pm2 start src/index.js即可，或者也可以通过nginx反向代理以达到更好的稳定性。

6.2 后端控制平台部署

后端控制平台已经打包成了单独的exe可执行文件，在Windows平台下双击“IoT Platform.exe”即可运行。

对于Linux服务器与MAC OS服务器，附件中并没有直接提供完成打包的文件，需要在相应平台中安装Python开发环境（需Python 3.10及以上版本）并正确配置环境变量。解压“IoT Platform.zip”后，进入源码根目录通过终端运行“pip install -r requirements.txt”安装必要的依赖，然后运行python index.py启动应用即可。

如需打包部署，则需要安装Pyintsaller打包工具，推荐直接安装基于Pyinstaller的Python图形界面打包工具auto-py-to-exe，通过“pip install auto-py-to-exe”即可安装。安装完成后，在终端中运行“auto-py-to-exe”即会弹出图形界面，先设置为“单文件”与“无终端”打包，将目标文件选择为解压后根目录下的“index.py”，图标选择为“icon.ico”，并将icon/文件夹添加至附加文件，然后点击生成可执行文件即开始生成可执行文件。生成完成后，可执行文件将存放至output/文件夹中，双击即可打开运行。生成的可执行文件不再依赖于Python环境，可随意移动并在任意位置打开使用。

第七章 项目总结

当前来说，我们基于WebGL技术的3D物联网智能监测-控制平台已经基本完成，为了方便之后的开发，我们对该系统进行总结。整个系统的后端控制平台使用PyQt进行UI开发，并可脱离PyQt直接通过命令行操作（方便没有安装图形界面的服务器）；网页端控制台后端使用Node.js框架Express.js进行开发，前端使用基于Vue框架的Vue-Element-Admin模板；3D模拟观测平台则使用WebGL框架Three.js进行开发，最终构建了一个较为完善的涵盖前后端的物联网平台。

通过组件开发模式、合理的设备抽象以及选择合适的模型、视图以及控制器设计，开发过程较为流畅，并且各组件之间的耦合度较低。通过重写大量与物理设备进行交互的API与多线程优化，后端控制平台可以在实现易操作、高度可自定义的情况下实现长期稳定运行；通过应用各种异步技术，例如RxJS框架，前端（包括3D平台）可以实现流畅的数据读取与实时监控。

在整个开发过程中，服务器的运行状况十分稳定，后端控制平台的操作可以实时反馈至前端。相应的MySQL数据库也并没有发生异常。

当前该平台已经实现了一套较为完整的物联网解决方案，可以替代一部分中小规模物联网应用中所使用的物联网软件平台。在未来该平台将进一步完善，通过继续为事件系统与3D模拟观测平台添加功能以实现更为完整的智能物联网解决方案。