2021 秋季学期物联网引论课程大作业设计报告

小组序号：14

小组成员及分工

	学号	姓名	分工	工作量百分比（%）
1	19231076	梁兆基	传感器端	23
2	19231007	马文韬	传感器端	23
3	19231258	陈思翰	服务器端	33
4	19373117	孙文佳	客户端	16
5	19182644	李思炀	客户端	5

一、大作业选题、内容及功能说明

我们大作业选择的是题目三：**一个火警报警及应急处理系统**。

主要需要实现四个功能：

1. 感知环境温度，当环境温度超过阈值，自动触发报警：终端 led 以固定频率闪烁，终端上电机转动
2. 基础功能指令实现，可以远程设置阈值
3. 报警状态下，按下按钮可以现场解除报警
4. 通过网页查询温湿度，显示温湿度曲线，远程关闭或打开报警

为了实现上述功能，我们兵分四路，分别处理传感器端、LwM2M 端、Django 服务器端和客户端，通过确定接口，调用 api、订阅信息，完成一系列的工作流程。最终我们完全实现了题目三的要求内容，并颇具亮点。

并且，我们还完成了对于传感器端的完全掌握，客户端、服务器端也都留出了相应的接口方便以后的拓展，对于基础 IOT 开发，其工作流程以及工程方面的拓展性，都有了较深的认识。

二、设计方案与设计思路

架构图

传感器端：

传感器端设计分为平台通信和功能实现两部分，具体设计方案和思路如下：

1. 平台通信
   a) 在电信平台进行实名认证并开通使能服务
   b) 添加产品和物模型
   c) 修改代码服务 ID 与平台一致并将代码加载到开发板上
   d) 接通电源、通信
2. 功能实现
   a) 电机控制，0 表示关闭，1 表示开启
   b) 温湿度上报控制，0 表示关闭上报，1 表示开始上报
   c) 上报周期控制，输入整数改变上报周期
   d) Led 闪烁，0 表示关闭闪烁，1 表示开启闪烁。仿照原有服务新增一个服务，仿照温湿度上传函数实现周期闪烁。
   e) 按键关闭报警，按下 key2 时同时停止电机转动和 led 灯闪烁。可以在服务线程中检查按钮状态，若按下则停止 led 灯闪烁并关闭电机

服务器端：

可以看出服务器端向南连通 LwM2M 服务器和向北连通客户端，起桥梁作用，其主要工作可以分为两块：

向 LwM2M 服务器请求服务：

1. 启动/关闭电机

2. 启动/关闭 LED
3. 启动/关闭数据上报
4. 温湿度数据自动推送
5. 请求最新的温湿度数据
6. 启动/关闭规则
7. 修改规则

为客户端提供服务:

1. 启动/关闭电机
2. 启动/关闭 LED
3. 启动/关闭报警状态（电机、LED 灯同时工作）
4. 启动/关闭数据上报
5. 请求最新的温湿度数据
6. 请求历史温湿度数据
7. 更改规则阈值（包括温湿度）

设计方案：

我们打算通过 Django 框架进行服务器端的开发，向 LwM2M 服务器端请求服务的具体方式为使用天翼 CTWing 提供的 Python SDK，除了温湿度数据自动推送的所有功能依照 API 手册调用相关接口即可，而设备温湿度数据的自动推送则通过 LwM2M 服务器端提供的订阅服务解决，由于没有公网 IP，我们打算采用内网穿透的方法实现订阅功能。

对于客户端需要请求的服务，提供一组后端函数供其调用，当后端函数被调用后，大部分函数会转向 LwM2M 服务器请求服务，因此对客户端提供的服务和向 LwM2M 服务器端请求的服务基本是一一对应和的。

通过文档了解发现，更改规则阈值的的服务需要包含三个操作：停止规则，更改规则，启动规则，否则无法更改规则内容。另外，对于订阅服务，提供一个后端函数供 LwM2M 服务器调用，此函数能接收 LwM2M 服务器推送的温湿度数据，并写入 Django 客户端本地的 MySQL 数据库，当客户端请求相关历史温湿度数据时便予以返回。

服务器端与 LwM2M 服务器端、客户端的之间的通信均基于 http 协议

客户端：

客户端设计的重点主要是 UI 界面设计与客户端、服务端之间的交互。

由于 Vue 框架已经集成了非常完备的功能，我们选择了 vue 作为客户端的主体开发框架。为了提供良好的用户体验，我们选择做一个单页 web，所有功能在用户视角下一览无遗，同时也省去了页面的切换时间。客户端分为五大功能模块，分别为电机控制、LED 控制、警报控制、温湿度管理、阈值设置，每个模块设置若干按钮来实现相应功能。

客户端与服务端之间的通信采用 vue 中内嵌的 axios 库，客户端向服务端发送一个 HTTP 请求，服务端进行功能调用，将结果返回给客户端，显示在 web 页面上。

三、开发工作介绍

传感器端：

基础指令对接：

可以看到在在下面的核心函数中，每一个指令都会被解析成为一个 CmdStruct 类型的结构体，其中包含关键指令信息。指令则会进一步完成相应的判断、识别、和利用现有函数对传感器的控制。

```
指令解析、生效流程与机理
app_aep_profile.c
        void decode_aep_data(data){
        //解析成为一个指令结构体
                result = decodeCmdDownFromStr(data);
        //依次判断指令内容，完成指令识别、信息获取
                if( result.code == AEP_CMD_SUCCESS )
                if( strcmp(result.serviceIdentifier, "motor_control") == 0 )
                value = *((uint8_t *)result.data);
        //利用指令完成对于led、马达等的控制
                setStatus(MOTOR1 , value); //设置电机状态
                setStatus(LED0_DEVICE_ID,LED_BLUE);//设置led颜色

        }
```

```
typedef struct CmdStruct
{
  char* serviceIdentifier;
  unsigned short taskId;
  void * data;
  int code;
} AepCmdData;
```

温湿度上报：

我们找到了关键的温湿度上报函数和控制其等频率运作的 timer 函数，学习其运作机理对我们实现 led 等频率闪烁作用极大。

```
找到关键频率控制函数
nb_timer_cb(void *args);
        send_msg(char *data, uint32_t len, uint8_t mode)        温湿度上报指令
        soft_timer_start(nb_timer, reportPeriod);               开启等频率循环调用
                soft_timer_t  *nb_timer;                        “循环变量”
                int  reportPeriod = 10;                         频率
        soft_timer_stop(nb_timer);                              关闭等频率循环调用

soft_timer_register(&nb_timer, OS_PERIODIC_MODE, nb_timer_cb, NULL);
```

报警机制处理：

我们在服务器端便把报警功能直接拆分成为了马达转动和 led 闪烁两个部分，分别向传感器端发送指令。识别指令类似于基础指令的识别：

```
指令机理（类比基础指令）：
        //识别指令
        if( strcmp(result.serviceIdentifier, "led_control") == 0 )
                        //拿到控制值
                        value = *((uint8_t *)result.data);
                        //利用led_flag设计简单的状态机
                        if (value == 0)
                                soft_timer_stop(temp_timer);
                                led_flag = 0;
                        else if(value == 1 && led_flag == 0) {
                                soft_timer_start(temp_timer, tempPeriod);
                                led_flag = 1; 指令执行标识
```

而灯的等频率闪烁我们则是效仿了温湿度的等频率上报，设计了自己的 flag 标识，写了与原 timer 的类似的 temp_timer，完成了 led 的等频率闪烁，并且与温湿度上报并不冲突。

```
灯的等频率闪烁（类比温湿度上报）
        仿照*nb_timer创建了* temp_timer
        soft_timer_t  *temp_timer;
        仿照nb_timer_cb创建了temp_timer_cb
        static void temp_timer_cb(void *args)
        {
                if (flag) 灯的闪烁标识
                        setStatus(LED1_DEVICE_ID, LED_RED);
                        flag = 0;
                else
                        setStatus(LED1_DEVICE_ID, LED_OFF);
                        flag = 1;
                soft_timer_start(temp_timer, tempPeriod);
        }
                soft_timer_register(&temp_timer, OS_PERIODIC_MODE,temp_timer_cb, NULL);
```

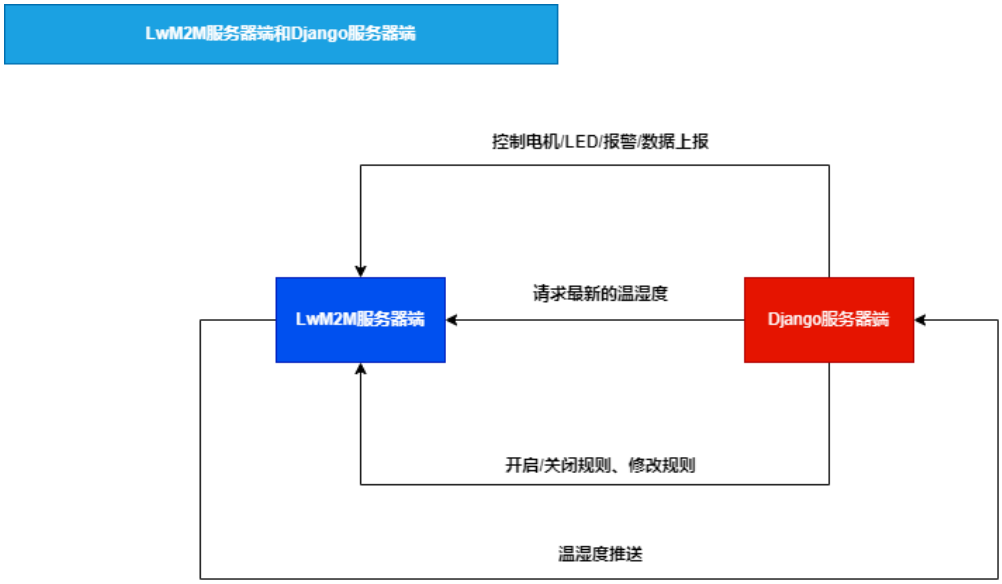
而按钮停止报警的实现则较为简单，只需要我们在主线程函数中进行对于按钮状态的特判，进一步操作原有函数控制 led，马达即可。

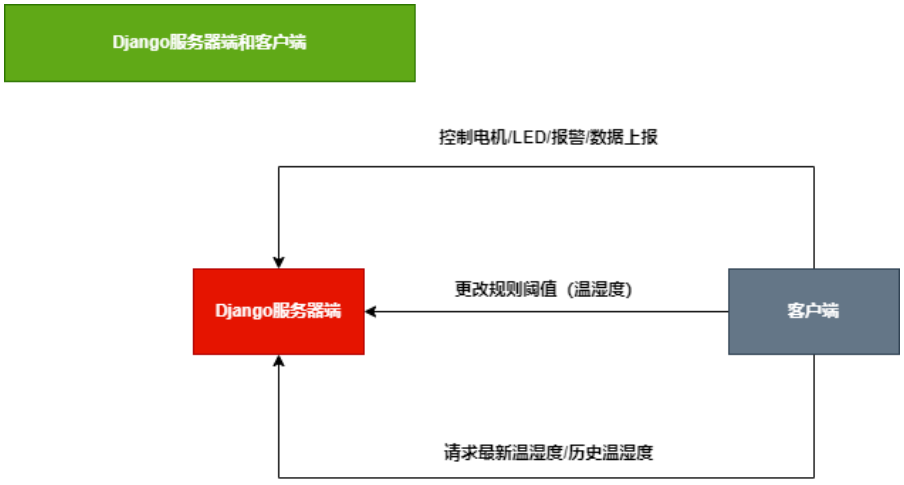
```
在线程循环中检测按钮状态即可

getStatus(KEY2, &BSP_status);
if( BSP_status.key_value == 0 )
        led_flag = 0;
        setStatus(MOTOR1 , 0);
        soft_timer_stop(temp_timer);
        setStatus(LED1_DEVICE_ID, LED_OFF);
```

服务器端：

1. 后端代码详见附录

2. 数据流图





3. 配置表

a) MySQL 配置

i. 新建 iot 数据库，device_data 表

```
CREATE SCHEMA `iot` ;
CREATE TABLE `device_data` (
  `time` varchar(50) NOT NULL,
  `humidity_data` varchar(50) NOT NULL,
  `temperature_data` varchar(50) NOT NULL,
  PRIMARY KEY (`time`)
)
```

time	humidity_data	temperature_data
2021-12-22 09:07:29	23.0	29.399999618530273
2021-12-22 09:07:33	23.0	29.399999618530273
2021-12-22 09:07:35	23.0	29.5
2021-12-22 09:07:37	23.0	29.5
2021-12-22 09:07:40	23.0	29.5
2021-12-22 09:07:42	23.0	29.399999618530273
2021-12-22 09:07:44	23.0	29.399999618530273
2021-12-22 09:07:46	23.0	29.399999618530273

b） Python SDK 只需要导入我们需要使用的接口函数

具体包括四个.py 文件:

aep_device_command.py

aep_device_status.py

aep_rule_engine.py

AepSdkRequestSend.py

c） Django 支持跨域设置

```
# INSTALLED_APPS = [
#       ...
#     'corsheaders',
#       ...
# ]
# MIDDLEWARE = [
#       ...
#     'corsheaders.middleware.CorsMiddleware',
#       ...
# ]
# CORS_ORIGIN_ALLOW_ALL = True
```

d） 内网穿透

利用 ngrok 进行内网穿透，在命令行中运行：

.\ding -config=.\ding.cfg -subdomain=iot 8080 即可

```
ngrok

Tunnel Status                 online
Version                       1.7/1.7
Forwarding                    http://iot.vaiwan.com -> 127.0.0.1:8080
Forwarding                    https://iot.vaiwan.com -> 127.0.0.1:8080
Web Interface                 127.0.0.1:4040
# Conn                        4
Avg Conn Time                 9323.13ms



HTTP Requests
-------------

POST /get_data                200 OK
POST /get_data                200 OK
POST /get_data                200 OK
POST /get_data                200 OK
POST /get_data                200 OK
POST /get_data                200 OK
POST /get_data                200 OK
POST /get_data                200 OK
POST /get_data                200 OK
```

客户端：

我们以获取最新温湿度这一功能为例，来展示客户端与服务端的通信过程。

```
show_new_data () {
  const that = this
  that.$axios.get( url: 'show_new_data',  config: {}).then(function (response) {
    console.log('1:receive new temperature and humidity data!')
    console.log(response.data)
    that.$notify({
      title: 'Notice',
      message: '当前温度: ' + response.data.temperature_data + '℉<br>当前湿度: ' + response.data.humidity_data + '%',
      dangerouslyUseHTMLString: true,
      duration: 0,
      position: 'bottom-left',
      type: 'success'
    })
  })
},
```

向后端发送Axios请求

将response显示在前端

温湿度曲线表：三个参数分别是时间，湿度和温度

```
###name_list
device_data_name_list = ['time','humidity_data','temperature_data']
```

在 get 里面配置指定图标的数据

```
get() {
  // 基于准备好的dom，初始化echarts实例
  var myChart = echarts.init(document.getElementById("main"));
  // 指定图表的配置项和数据
  var option = {
    title: {
      left: "left",
      text: "温湿度折线统计图",
    },
    dataZoom: [
      {
        type: "slider", //滑动轴
        start: 1, //距离左侧0刻度的距离，1%
        end: 35, //距离左侧0刻度的距离，35%，相当于规定了滑动的范围
      },
    ],
    xAxis: {
      type: "category",
      name: "time",
      data: [
        "00:00",
        "01:00",
        "02:00",
        "03:00",
        "04:00",
        "05:00",
        "06:00",
        "07:00",
        "08:00",
        "09:00",
        "10:00",
```

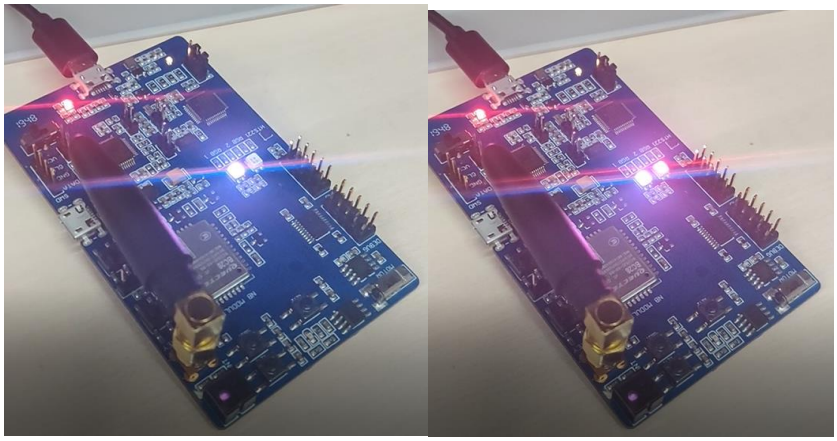
利用 axios 从后端获取数据

```
},
mounted() {
  this.get();
  this.getchart();
},
methods: {
  // 获取数据
  getchart() {
    axios.get("接口", {}).then((response) => {});
  },

  get() {
```

四、系统成果展示

传感器端：

由于电机、温湿度上传等功能图片不便展示，仅展示 led 闪烁，其余用视频展示。



视频说明（见附录）：

1）电机展示：下发开始指令后电机转动，下发关闭指令后电机停止转动。

2）Led 展示：下发开始指令后 led 开始闪烁，下发关闭指令后 led 停止闪烁。

3）温湿度上报展示：下发开始指令后开始上报，下发关闭指令后上报停止。

服务器端：

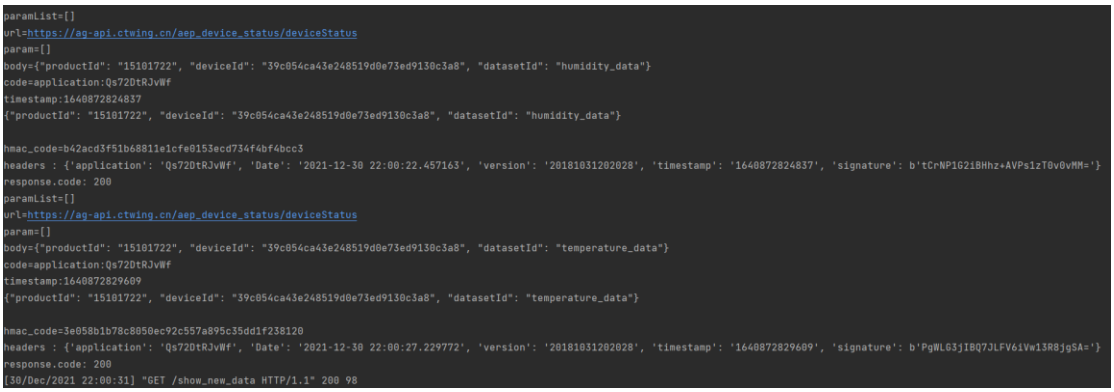
启动电机：

关闭电机：

开启 LED：

关闭 LED：

获取最新温度：



开启/关闭报警：

报警就是同时开启/关闭电机和 LED，因此省略服务器端运行结果的展示

接收温湿度推送：

```
ngrok

Tunnel Status                 online
Version                       1.7/1.7
Forwarding                    http://iot.vaiwan.com -> 127.0.0.1:8080
Forwarding                    https://iot.vaiwan.com -> 127.0.0.1:8080
Web Interface                 127.0.0.1:4040
# Conn                        4
Avg Conn Time                 9323.13ms



HTTP Requests
-------------

POST /get_data                200 OK
POST /get_data                200 OK
POST /get_data                200 OK
POST /get_data                200 OK
POST /get_data                200 OK
POST /get_data                200 OK
POST /get_data                200 OK
POST /get_data                200 OK
POST /get_data                200 OK
```

```
Quit the server with CTRL-BREAK.
####################################
##############成功写入MySQL##############
####################################
[08/Dec/2021 11:00:35] "POST /get_data HTTP/1.1" 200 4
####################################
##############成功写入MySQL##############
####################################
```

修改规则：

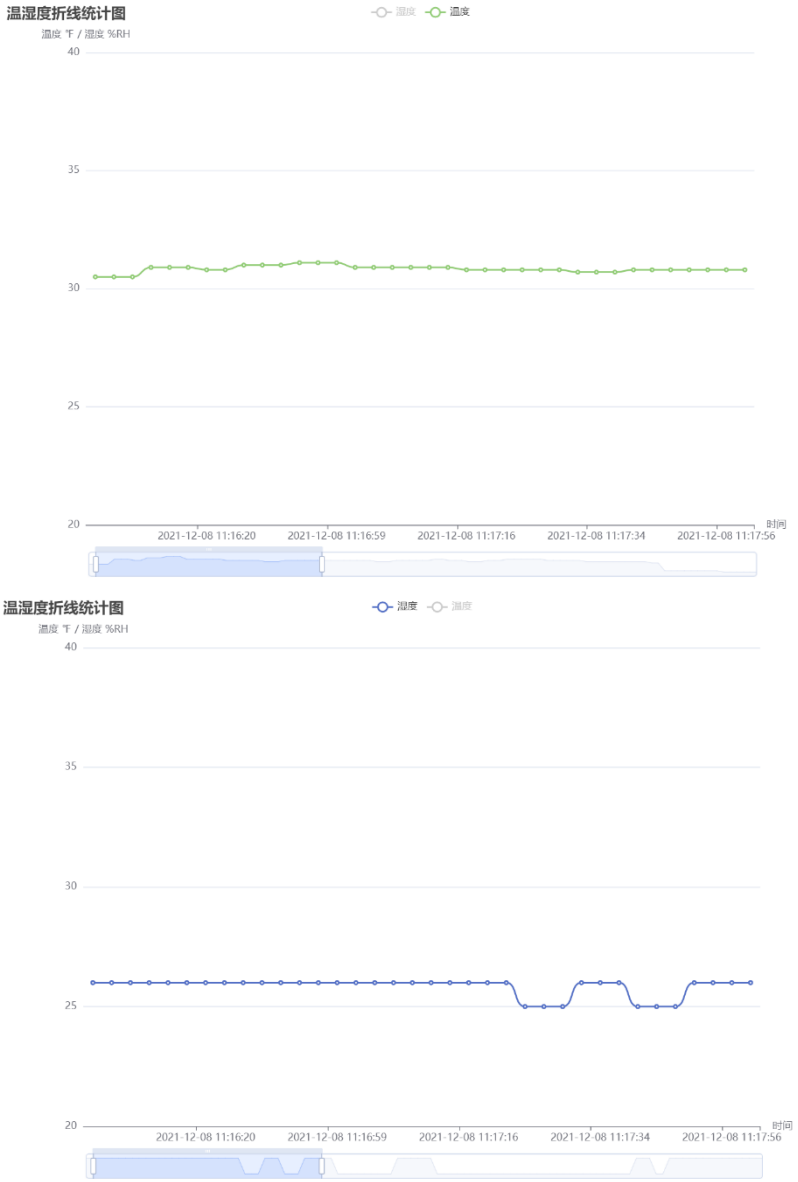
```
paramList=[]
url=https://ag-api.ctwing.cn/aep_rule_engine/v3/rule/modifyRuleRunningStatus
param=[]
body={"ruleId": "99ed150c-944f-c14d-5467-980f3b3cf396", "runningStatus": "1100"}
code=application:Qs72DtRJvWf
timestamp:1640873224245
{"ruleId": "99ed150c-944f-c14d-5467-980f3b3cf396", "runningStatus": "1100"}

hmac_code=f11edac4e4590dd072548259ecffcc8824217f5
headers : {'application': 'Qs72DtRJvWf', 'Date': '2021-12-30 22:07:01.865321', 'version': '20210327062603', 'timestamp': '1640873224245', 'signature': b'8R7axORZDdByVIJZ7P/MiCSCF/U='}
response.code: 200
{'code': '0', 'msg': 'ok', 'result': None}
paramList=[]
url=https://ag-api.ctwing.cn/aep_rule_engine/v3/rule/modifyRuleRunningStatus
param=[]
body={"ruleId": "4b1f2217-f067-dec0-2b12-05906e409cd5", "runningStatus": "1100"}
code=application:Qs72DtRJvWf
timestamp:1640873229018
{"ruleId": "4b1f2217-f067-dec0-2b12-05906e409cd5", "runningStatus": "1100"}

hmac_code=46540ef82e06ab06c4c205463f8937d02b81a839
headers : {'application': 'Qs72DtRJvWf', 'Date': '2021-12-30 22:07:06.637752', 'version': '20210327062603', 'timestamp': '1640873229018', 'signature': b'RlQO+C4GqwbEwgVGP4k30CuBqDk='}
response.code: 200
{'code': '0', 'msg': 'ok', 'result': None}
####################################
#############已关闭规则#############
####################################
```

客户端：

客户端主页面效果如下所示：

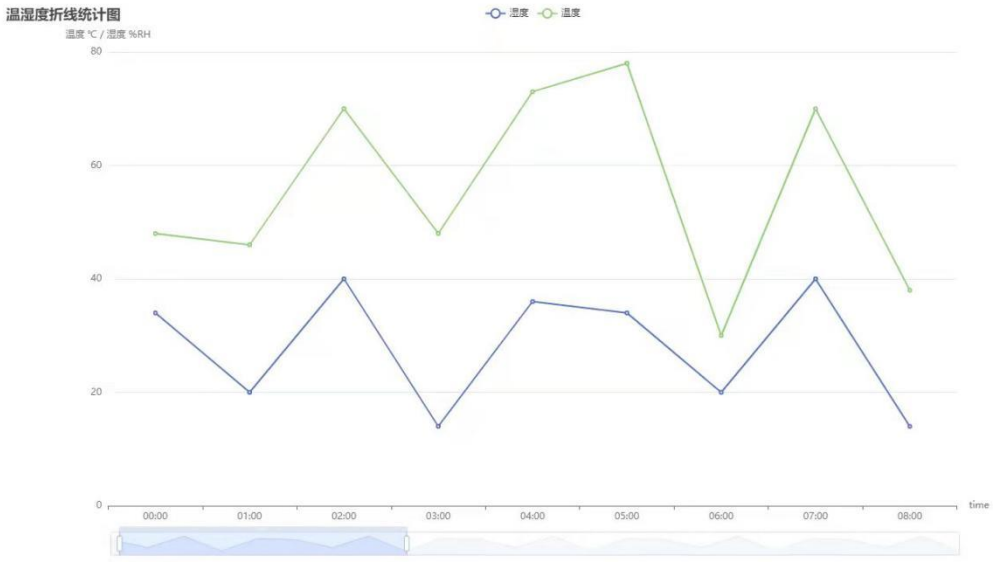
在温湿度管理模块中，点击左下角的获取最新温湿度按钮，客户端会将从服务端获取的温湿度信息以 Notice 窗口的形式显示在页面左下角，该窗口可由用户手动关闭。

点击历史温湿度曲线按钮，页面会跳转到温湿度曲线图，如下图所示。温湿度曲线通过 echarts 功能组件绘制。用户可通过左右拖动 x 轴游标来自由选择时间范围。



点击历史温湿度曲线按钮，页面会跳转到温湿度曲线图，如下图所示。温湿度曲线通过 echarts 功能组件绘制。用户可通过左右拖动 x 轴游标来自由选择时

间范围。并且两条曲线在同一屏显示，方便用户进行比对。



五、设计亮点

1. 传感器端实现

   led 等频闪烁不与其他任何服务冲突，且很容易调节频率（在源代码中更改一个数字即可）。如有需要还可以增加一个调节频率的服务。

2. 前后端分离

   前端采用了 Vue 进行开发，后端采用 Django 进行开发，采用前后端分离的做法，好处是开发效率较高，整个开发过程中前后端开发可同步进行，且很少遇到 bug。

   此外，前后端可以部署在不同的服务器上，甚至可以使用多个后端服务器，通过 Nginx 进行反向代理，防止某一个后端服务器宕机导致系统整体失效，增加了系统可靠性，是投入生产的保证（虽然实验中不明显）

3. 数据库的使用

   如果不采用数据库进行温湿度的存储，那么系统永远只能知道此时此刻的温湿度数据，将温湿度数据写入 MySQL 能够获取任意开发板历史时间节点的温湿度数据

4. 订阅机制的使用

   如果采用轮询的方法获取温湿度数据，如果开发板某一时间段的温湿度数据没有发生任何变化，将带来大量的冗余数据；如果采用订阅机制，只在开发板温湿度发生变化时对相关数据实现推送，将不会产生此问题

5. 规则的使用

   一般的做法为：在 Django 服务器端进行条件判断，并根据相应逻辑判断，执行相应的动作，但是这种方法耦合性太大，数据链路太长，由架构图可知贯穿三端：设备端，LwM2M 服务器端和 Django 服务器端，并且一旦 Django 服务器端宕机，监视功能将完全失效（本实验为温湿度一旦超过阈值需要开启报警状态）；因此我们选择通过 LwM2M 服务器端实现监视功能，具体通过规则引擎实现，主要原理和通过 Django 服务器端实现基本相同，优点是安全性更高，Django 服务器端的宕机不会直接导致监视功能的失效，耦合性较小，此外数据链路较短，理论上延迟更小（只有两端

进行 http 协议交互）

6. 采用 echarts 组件画曲线图，用户可根据自己的需求选择时间范围，而且温度与湿度可以同时显示在一张图上，用户可以清晰地比较两条曲线的变化趋势。

六、大作业总结

（1）在研发过程中遇到了哪些问题？是如何解决的？

传感器端：

1. 实名认证后平台仍然没有“产品中心”

   需要开通使能服务才能添加设备

2. 按照指导说明添加设备和服务后，进行相关指令下发后无响应。

   自带源代码中服务 ID 不正确，与平台的服务 ID 不吻合，改为相同的服务 ID 可以正常实现功能。

3. 服务 ID 正确后平台没有显示温湿度数据上报

   没有下发指令让温湿度上报打开。下发指令成功实现温湿度数据上报。

4. 想通过主线程的闪烁函数实现 led 闪烁功能，但线程间无法传递变量

   闪烁相关的函数要写在同一个线程里。

5. 创建了 temp_timer_cb 等但是把灯的开启关闭写在了线程循环中，灯的闪烁频率会受温湿度上报影响

   仅使用 Temp_timer_cb 可以完成闪烁任务，不需要将闪烁另写在线程循环里，会受其他循环的服务冲突。

6. 把灯的闪烁写在了温湿度上报的 nb_timer_cb 里，试图让灯和上报频率相等，但实际并不可行。

   需要新写的专门为灯闪烁使用的 Temp_timer_cb 完成闪烁任务。

服务器端：

1. 跨域

前后端分离最大的问题就是会带来跨域问题，我们利用 Django 框架支持跨域，具体方法见配置表

2. 订阅功能需要公网 IP

订阅功能的实现需要公网 IP，但是我们小组没有服务器，没有公网

IP，所以采用 ngrok 进行内网穿透，具体使用方法见配置表

3. 规则的权限问题

最开始手动在网站上配置了规则，但始终无法通过 api 开启/关闭规则和更改规则内容，通过向技术支持人员了解发现，通过 api 创建的规则和手动创建的规则存在操作权限不同的问题，即手动创建的规则只能手动更改、开启/暂停，api 创建的规则同理只能通过 api 操作

4. request 的返回信息为十六进制表示

调用 api 后返回信息为十六进制，无法阅读

利用 Python 的函数进行解析

```
result = json.loads(apis.aep_device_status.QueryDeviceStatus(appKey=appKey, appSecret=appSecret, body=body))
```

5. 临时突然更新接口

某次检查时发现原本正常的功能，在运行时会出奇怪的 bug，后来发现是 api 进行了更新（查询设备温湿度，之前返回的信息和后来返回的信息的数据结构存在细微区别）

6. 接口调用时无法关闭 Print 开关，调试困难

只要调用 api 就会输出大量的无用信息，且不能关闭，调试困难

自行添加 Print 输出更多更明显的信息辅助调试

（2）请简要地总结一下你们在物联网引论课程中的收获。

序号	姓名	收获
1	梁兆基	在理论课上，我们学习了从初期到现在的各式物联网技术：条码、磁条卡、IC 卡、RFID、EPC 系统，物联网的体系结构：感知层、网络层、应用层，以及各类系统接口：RS232C、RS422、RS48、WIFI、ZigBee、蓝牙等等，以及兴起的窄带通信技术，为实验的进行打下了理论基础。作为负责传感器端的同学，对设备源代码的阅读、理解、运用、调试，使我对设备的理解更加深刻，让我了解到底层工作的特点，加深了我对 NB-IOT 系统的认知，锻炼了团队的协作能力。
2	马文韬	本学期的 IOT 课程对于我来讲意义很大，在小组实验工作中我负责的是硬件功能的实现。6 系的课程环境下，能直接面对硬件并且与软件交通的机会是很少的，这对于同学们日后的方向选择有着很重要的指导意义。通过实验以及课程的学习，我了解到了物联网硬件在当下的重要性，硬件系统在工程实践中的底层性与重要性。
3	陈思翰	经过理论课的学习，我对物联网有了一个大致的了解，包括其发展历史和现状，知道了物联网体系的三层架构：感知层，网络层，应用层，了解了物联网相关技术、数据编码方法、传感器相关的知识等 在实验中，我主要负责服务器端的开发，我大致熟悉了如何使用 SDK 进行开发过程的加速；对物联网的整体架构、端到端的通信协议(LwM2M、http 等)有了总体的了解，学会了在文档说明不清楚的时候寻找技术支持吗，学会了应用已有知识。此外，一学期的小组合作也提高了我的团队合作能力
4	孙文佳	通过理论课的学习，我对万物互联的世界有了基本的认知。 实验作业也很新颖，软硬结合，很有意义。笔者以前从未尝试过前端开发，而通过亲手搭建一个物联网系统的客户端，我对前端开发实现了零的突破。我对 vue 框架的运用更加熟练，对 http 请求的原理以及前后端的交互过程有了更深刻的理解。此外，前后端分离开发是软件工程中很常用的模式，这样的合作过程增强了我与他人沟通的能力，也让我明白写出架构清晰的代码对于自己和他人都是很重要的。笔者还有一个小小的进步，就是之前我遇到难题时都会求助于 CSDN，但是现在我学会了从官方文档中寻找解决办法，这样有助于独立思考。
5	李思炀	通过理论课的学习，我学到了很多。物联网用途广泛，遍及智能交通、物流管理、环境保护、政府工作、公共安全、平安家居、智能消防、工业监测、老人护理、个人健康、花卉栽培、水系监测、食品溯源、敌情侦查和情报搜集等多个领域。在这一学期中，我感受到了物联网的飞速发展，也深刻体会到了计算机技术的更新换代。通过这次学习，我发现原来物联网技术就在我们身边。

		实验作业让我体会到了团队合作的重要性。作为一个可能基础没那么好的学生，其他组员很好的包容并且帮助了我。这是我第一次完成一个前端的部分开发，学会了在 vue 框架下进行编程。和小组同学的交流也让我对前后端的交互有了更深刻的认识。总的来说，这门课程对我的编程生涯可谓意义重大，十分感谢

（3）你们对课程的教学、实践等环节有没有自己的建议？（比如课程知识点安排、实践题目难易等，汇总或分别阐述均可）

1. 实践环节的建议：对于服务器端的指导可以编写一些指导性比较强的参考资料。
2. 希望理论课可以进一步细化，虽然是概论但是多一些实物类型的展示，不然有些抽象。