实验五：　　课程综合设计实验

第 5 小组：胡琎 王一鸥 马逸行

1. 实验目的

1．本次实验的目的是开发基于 NB-IoT 的物联网应用系统。
2．主要目标是进一步掌握基于 NB-IoT 的物联网系统的整体架构及组成，掌握物联网系统主要组成部分的工作原理、关键技术和开发方法。
3．在端节点、服务器、应用程序等方面的开发拓宽对于物联网系统的理解，并通过实现典型的物联网应用系统，使产业界常见的应用及技术需求与课堂知识相结合，拉近两者的距离。

2. 实验内容及设计功能

2.1 实验内容

设计一个典型的物联网应用系统，包括端节点的设计与实现，服务器的设计与实现（配置），用户端应用程序设计与实现（手机上的 APP）。

2.2 设计功能

本组实验实现了基于温湿度感应的智能空调系统，可以自动上报当前环境的相关信息并以一定条件进行的空调的开关控制。

3. 实验环境介绍

3.1 软件环境

Keil 单片机开发环境;
STM32 单片机烧写程序;
NB-IoT 调试程序;
服务器传感数据显示服务程序;

3.2 硬件环境

基于 Android 平台的智能终端设备;
电信 NB-IoT 开发板（含 BC28 物联网通信模块）;

3.3 网络环境

中国电信物联网开放平台（官网 https://www.ctwing.cn）;
应用服务器;

3.4 其它外设要求

开发使用的计算机若干.

4. 主要系统的组成和原理

4.1 端设备部分

该部分主要由电信 NB-IoT 物联网开发板组成。该开发板上集成了温湿度传感器、光电传感器、电机等等。通过对开发板上的 STM32 单片机进行配置，使其可以执行特定的物联网设备功能，正确解析并上报通讯的信息。

4.2 物联网服务部分

该部分为现有电信 IoT 物联网公共平台服务。电信物联网平台集成了包括但不限于物联网设备管理、物联网应用管理、物联网、边缘计算等若干功能。通过对如上功能的正确配置，使得端设备与平台及客户端与平台之间的通信得以正确实现。

4.3 客户端应用部分（主要部分）

该部分为本组具有特色的重要的部分。本组实现了 SDK 配置、后端服务器配置与前端应用程序配置三大主要部分，使得相关数据能够通过用户进行传输，并使得用户能够直观的获得物联网模块上报的有关信息。

5. 主要开发内容

5.1 端设备部分

（通过前几次实验）建立设备与平台服务器的通讯联系，通过报文设计等内容保证有效信息成功传递并获得解析。

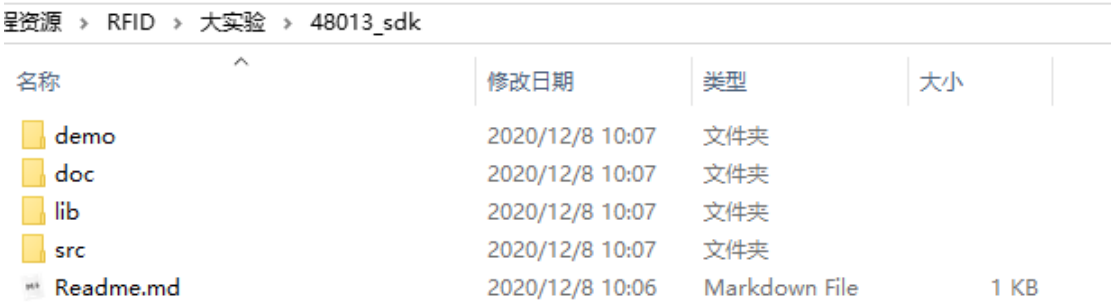
5.2 物联网服务部分

上传应用的 war 部署包，对其中相关配置进行修改（如 application.property）；
根据公共产品创建服务，添加所需服务字段（如 temperature_humidity_data）；
为产品添加订阅，将消息转发至托管云应用地址
添加设备 IMEI 号，与端设备进行有效联系。

5.3 客户端应用部分

1. SDK 准备

1.1 下载



demo 中为示例 java 程序

doc 中为 sdk 文档

lib 中为需要导入 maven 的 jar 包和 xml 文件

src 中为 sdk 的源代码

1.2 导入 maven

cd 到 lib 文件夹，通过如下指令导入两个 jar 包到 maven。

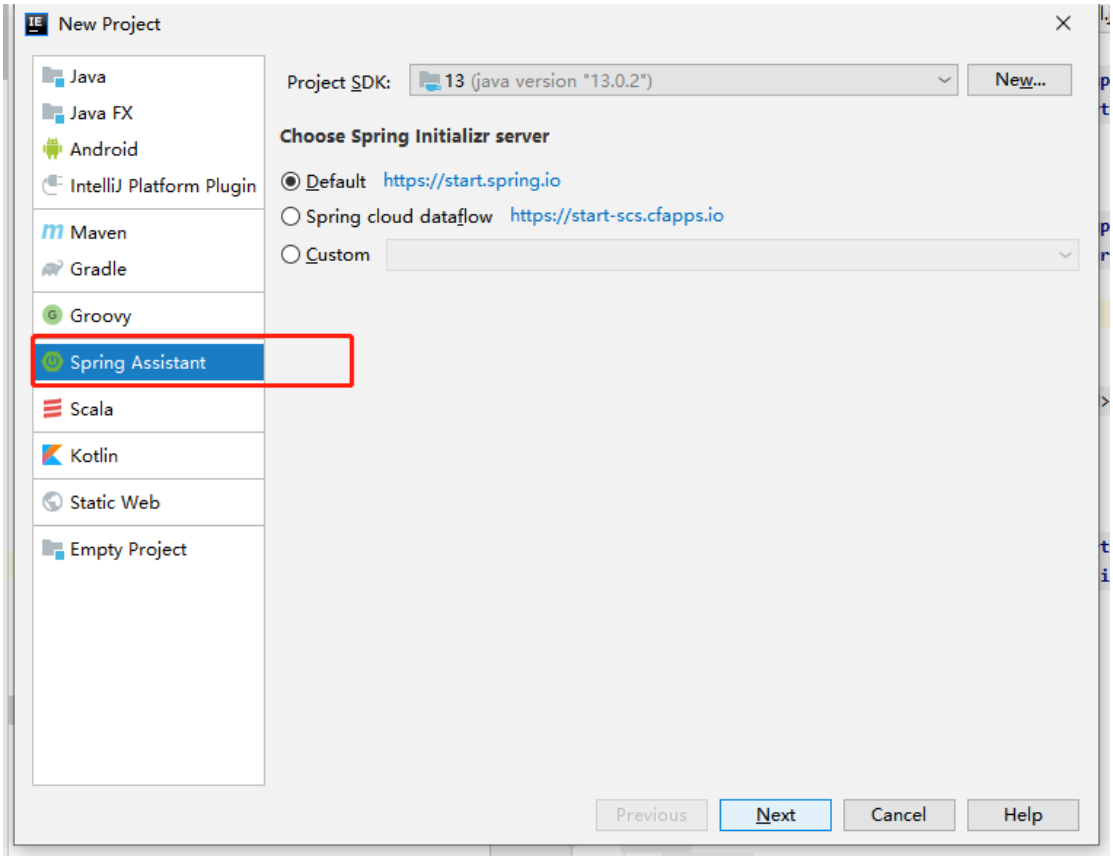
mvn install:install-file -Dfile=aep-ecloud-agsdk-biz-${SDK 版本 }-SNAPSHOT.jar -DpomFile=aep-ecloud-agsdk-biz-${SDK 版本}-SNAPSHOT.pom.xml

2. 后端开发

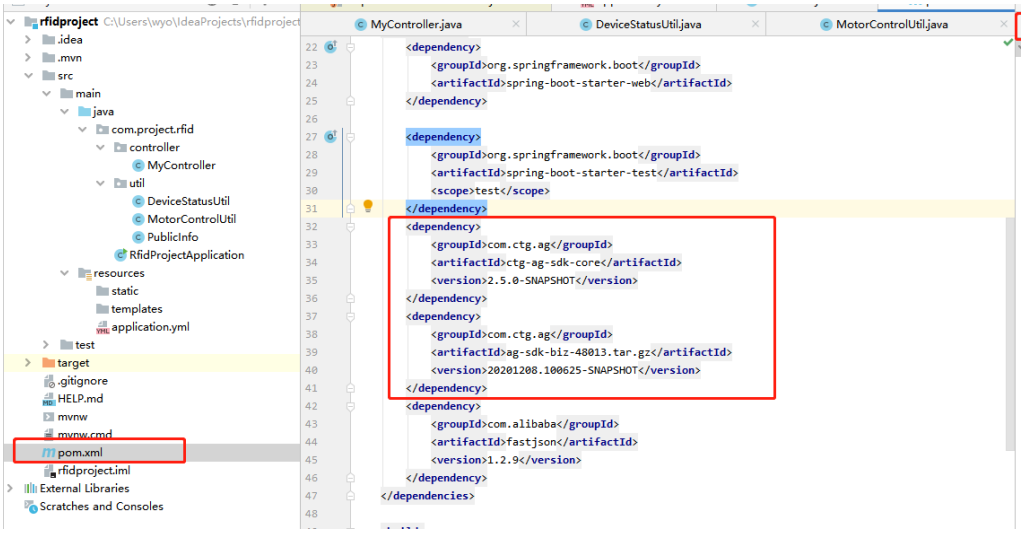
后端使用 Spring Boot 框架，调用 SDK 来与 NB 平台交互，创建了 4 个接口分别用来提供温度、提供湿度、控制电机的开与关。

2.1 Spring Boot 项目创建

在 IDEA 中创建 Spring Boot 项目。



2.2 导入 SDK



在 Spring Boot 项目的 pom.xml 文件中，加入这两个 SDK 的依赖。

2.3 测试 SDK

通过模仿 demo 中的例子，和阅读 doc 文件夹中的文档，进行 SDK 基本功

能的测试。

首先，本次试验使用 SDK 来调用的 API 如下：

API	功能	参数说明
QueryDeviceStatus	获取温度	设置 datasetId 为 temperature_data
	获取湿度	设置 datasetId 为 humidity_data
CreateCommand	打开点击	设置 serviceIdentifier 为“motor_control”，control_int 为 1
	关闭电机	设置 serviceIdentifier 为“motor_control”，control_int 为 0

然后仿照 sdk 给的 demo 来编写代码测试。测试时使用平台的模拟设备进行调试。测试代码和下方封装的代码基本相同。

2.4 封装

将 SDK 的调用、json 的组装与拆解等功能，封装在 Java 类中，方便后续使用。

具体来说，将温度获取和湿度获取，封装在 DeviceStatusUtil 类中；将电机的打开与关闭功能，封装在 MotorControlUtil 类中；将 appKey、appSecret 等参数放在 PublicInfo 类中。

DeviceStatusUtil.class

```java
public class DeviceStatusUtil {


    public double getTemperature() throws Exception {
        AepDeviceStatusClient client = AepDeviceStatusClient.newClient()
                .appKey(PublicInfo.appKey).appSecret(PublicInfo.appSecret)
                .build();
        QueryDeviceStatusRequest request = new QueryDeviceStatusRequest();

        String json = "{\"productId\":\""+PublicInfo.productId+"\",\"deviceId\":\""+PublicInfo.deviceId+
                "\"datasetId\":\"temperature_data\"} ";
        request.setBody(json.getBytes());

        QueryDeviceStatusResponse rst = client.QueryDeviceStatus(request);
        rst.getMessage();

        String body = new String(rst.getBody(), charsetName: "utf-8");
        JSONObject jsonObject = JSONObject.parseObject(body);
        String value = jsonObject.getJSONObject("deviceStatus").getString( key: "value");

        double temperature = Double.parseDouble(value);

        client.shutdown();
        return temperature;
    }
    public double getHumidity() throws Exception {
        AepDeviceStatusClient client = AepDeviceStatusClient.newClient()
                .appKey(PublicInfo.appKey).appSecret(PublicInfo.appSecret)
                .build();
        QueryDeviceStatusRequest request = new QueryDeviceStatusRequest();

        String json = "{\"productId\":\""+PublicInfo.productId+"\",\"deviceId\":\""+PublicInfo.devi
                "\"datasetId\":\"humidity_data\"} ";
        request.setBody(json.getBytes());

        QueryDeviceStatusResponse rst = client.QueryDeviceStatus(request);
        rst.getMessage();

        String body = new String(rst.getBody(), charsetName: "utf-8");
        JSONObject jsonObject = JSONObject.parseObject(body);
        String value = jsonObject.getJSONObject("deviceStatus").getString( key: "value");

        double temperature = Double.parseDouble(value);

        client.shutdown();
        return temperature;
    }
}
```

MotorControlUtil.java

```java
public class MotorControlUtil {
    public void moveMotor(int open) throws Exception{
        AepDeviceCommandClient client = AepDeviceCommandClient.newClient()
                .appKey(PublicInfo.appKey).appSecret(PublicInfo.appSecret)
                .build();

        CreateCommandRequest request = new CreateCommandRequest();
        request.setParam( name: "MasterKey",PublicInfo.masterKey);

        JSONObject body = new JSONObject();
        body.put("productId",PublicInfo.productId);
        body.put("deviceId",PublicInfo.deviceId);
        body.put("operator","wangyiou");

        JSONObject content = new JSONObject();
        content.put("serviceIdentifier","motor_control");

        JSONObject para = new JSONObject();
        para.put("control_int",open==0?0:1);

        content.put("params",para);
        body.put("content",content);

        request.setBody(body.toJSONString().getBytes());

        CreateCommandResponse rst = client.CreateCommand(request);

        System.out.println(rst);
    }
```


PublicInfo.java

```java
public class PublicInfo {
    public static String appKey="whYLhQvbeN3";
    public static String appSecret="rFt9P89m0z";
    public static long productId=15014432;
    public static String deviceId="a2b3cf0551154ed68377f09aa7396efa";
    public static String masterKey = "297b49bb079b438381651a11110b91b7";
}
```

2.5 编写 Controller

使用 Spring Boot 创建 Controller 来创建 http 接口，供前端 app 进行调用。实例化上面类的对象，调用方法进行 SDK 的调用，并将结果返回。

其中，”/getTemperature”接口返回当前设备的温度，“/getHumidity”接口返回当前设备的湿度；”openMotor”接口打开设备的电机，“/closeMotor”接口关闭设备的电机。

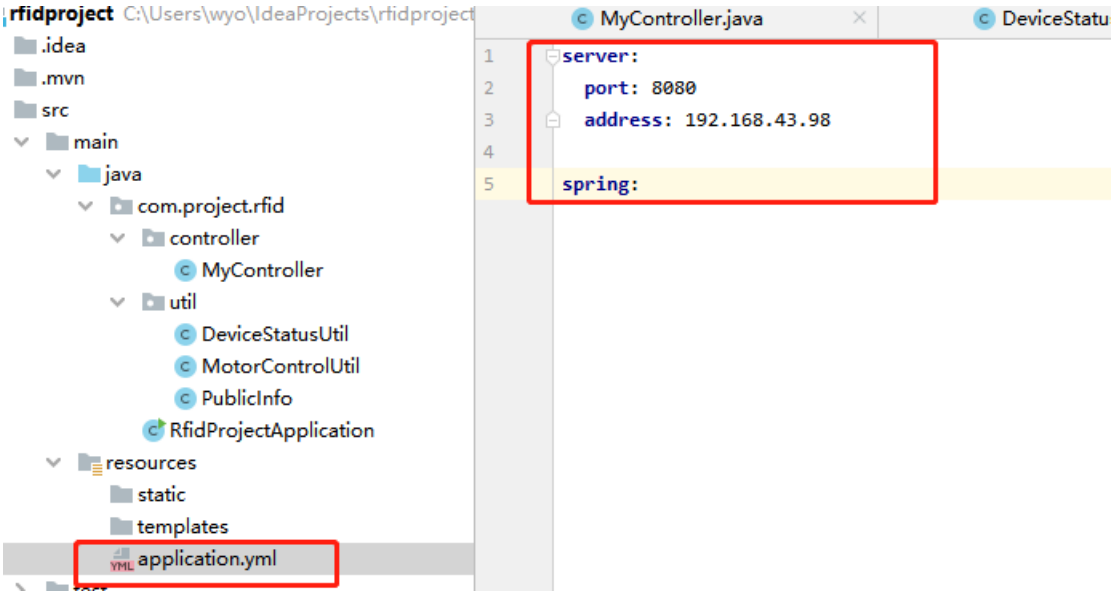
```java
@RestController

public class MyController {
    @CrossOrigin
    @RequestMapping(value="/getTemperature")
    public double getTemperature() throws Exception {
        System.out.println("getTemperature");
        DeviceStatusUtil util = new DeviceStatusUtil();
        return util.getTemperature();
    }
    @CrossOrigin
    @RequestMapping(value="/getHumidity")
    public double getHumidity() throws Exception {
        DeviceStatusUtil util = new DeviceStatusUtil();
        return util.getHumidity();
    }
    @CrossOrigin
    @RequestMapping("/openMotor")
    public int openMotor() throws Exception {
        MotorControlUtil util = new MotorControlUtil();
        util.moveMotor( open: 1);
        return 1;
    }
    @CrossOrigin
    @RequestMapping("/closeMotor")
    public int closeMotor() throws Exception {
        MotorControlUtil util = new MotorControlUtil();
        util.moveMotor( open: 0);
        return 1;
    }
}
```

`

2.6 设置端口和 IP

在 application.yml 中设置端口和 ip。由于我们的作业直接使用的内网，使用自己的笔记本作为服务器，所以这里的 ip 地址为内网的 ip。



3. 前端开发

3.1 app 简介

app 使用 uniapp 技术进行开发，所用语言为 H5。

3.2 温湿度显示

温湿度显示部分使用了“best-guage”插件，可直接根据最小值、最大值、当前值，绘制出仪表盘形状。

```
<template>
    <view class="content">
        <best-gauge :config="gaugeOption" class="ybp"></best-gauge>

        <best-gauge :config="gaugeOption2" class="ybp"></best-gauge>
        <button type="default" @click="refresh">刷新</button>


    </view>
</template>
```

点击“刷新”按钮后，触发相应事件，调取后台接口获得设备温湿度，js 修改 data 中的值，在 HTML 中进行了数据绑定，使 VUE 自动更新界面显示。

```
data() {
    let _width = uni.upx2px(350);
    return {
        gaugeOption: { //定义参数
            id: 'gaugeId0',
            name:"温度",
            status:1,
            unit:"°C",
            value: 0,
            axisTickLength: 2 //该属性与a
        },
        gaugeOption2: { //定义参数
            id: 'gaugeId1',
            name:"湿度",
            status:1,
            unit:"",
            value: 0,
            axisTickLength: 2 //该属性与a
        },
        temperature: 0,
        humidity: 0,
        title: 'Hello',
    }
```

```
methods: {
    refresh(){
        this.refreshTemp();
        this.refreshHumi();
    },
    refreshTemp() {
        console.log("send");
        uni.request({
            url: "http://" + this.Info.server_ip + ":" + this.Info.server_port + "/getTemperature",
            method: "GET",
            data: {

            },
            success: (res) => {
                console.log("success");
                console.log(res.data);
                this.gaugeOption.value = res.data;
            },
            fail: (res) => {
                console.log("fail");
                console.log(res);
            }
        });
    },
    refreshHumi() {
        console.log("send");
        uni.request({
            url: "http://" + this.Info.server_ip + ":" + this.Info.server_port + "/getHumidity",
            method: "GET",
            data: {

            },
            success: (res) => {
                console.log("success");
                console.log(res.data);
```

3.3 电机控制

使用 motor 变量的取值 0 或 1 来判断当前状态，并在按钮上显示不同内容。

```
<template>
    <view class="content">
        <button v-show="motor==0" type="default" @click="openMotor">打开电机</button>

        <button v-show="motor==1" type="default" @click="closeMotor">关闭电机</button>
    </view>
</template>
```

点击按钮后，调取后端接口进行点击的打开与关闭的操作。

```
methods: {
    openMotor(){
        this.motor=1;
        uni.request({
            url: "http://"+this.Info.server_ip+":"+this.Info.server_port+"/openMotor",
            method: "GET",
            data: {},
            success: (res) =>{
                console.log(res.data);
            }
        })
    },
    closeMotor(){
        this.motor=0;
        uni.request({
            url: "http://"+this.Info.server_ip+":"+this.Info.server_port+"/closeMotor",
            method: "GET",
            data: {},
            success: (res) =>{
                console.log(res.data);
            }
        })
    }
}
```

3.4 ip 修改

当服务器 ip 地址修改后，不用重新打包 app，可以直接在此页面修改参数。

```
<template>
    <view class="content">
        <text class="text">服务器ip:</text>
        <input class="input" type="text" :value="ip" @input="inputIp" />
        <button class="btn" type="default" @click="saveIp">保存</button>

        <text class="text">服务器端口: </text>
        <input class="input" type="text" :value="port" @input="inputPort" />
        <button class="btn" type="default" @click="savePort">保存</button>
    </view>
</template>
```

```
methods: {
    inputIp(e){
        this.ip = e.detail.value;
    },
    inputPort(e){
        this.port = e.detail.value;
    },
    saveIp(){
        this.Info.server_ip = this.ip;
        uni.showModal({
            title:"",
            content:"修改成功",
            showCancel:false,
            success:function(){
                console.log("success");
            },
            fail:function(){
                console.log("fail");
            }
        });
    },
    savePort(){
        this.Info.server_port = this.port;
        uni.showModal({
            title:"",
            content:"修改成功",
            showCancel:false,
            success:function(){
                console.log("success");
            },
            fail:function(){
                console.log("fail");
            }
        });
    }
}
```

客户端用户界面展示：


11:34
电机控制
打开电机
实时数据
电机控制
设置


11:34
实时数据
温度
20.0℃
在线
湿度
40.0
在线
刷新
实时数据
电机控制
设置


11:34
设置
服务器ip:
192.168.43.98
保存
服务器端口:
8080
保存
实时数据
电机控制
设置

6. 问题及解决方案

1．物联网平台与端设备无法进行有效通信

首先确认物联网设备是否连接上平台。一般可以通过物联网平台的状态查看实现，或者通过端设备上相关指示信息（本实验中为物联网开发板上的指示灯显示）进行判断。

其次，需要对物联网平台部署的信息进行设定。对于不同的端设备，其具有不同的 IMEI 号码。通过将平台与该设备 IMEI 号进行绑定，从而实现有效通信的关联。

最后，在平台部署的时候，要为产品添加相关的服务和订阅，随时对消息进行监控，从而使得平台正确部署解析端设备信息。

2．模拟设备可行后真实设备无法成功部署

首先需要明确，模拟设备中端设备-物联网平台关系是虚拟的，故原因可能在于真实设备未与平台建立有效通讯联系（也有可能是没有收到电信信号，网络环境较差）。

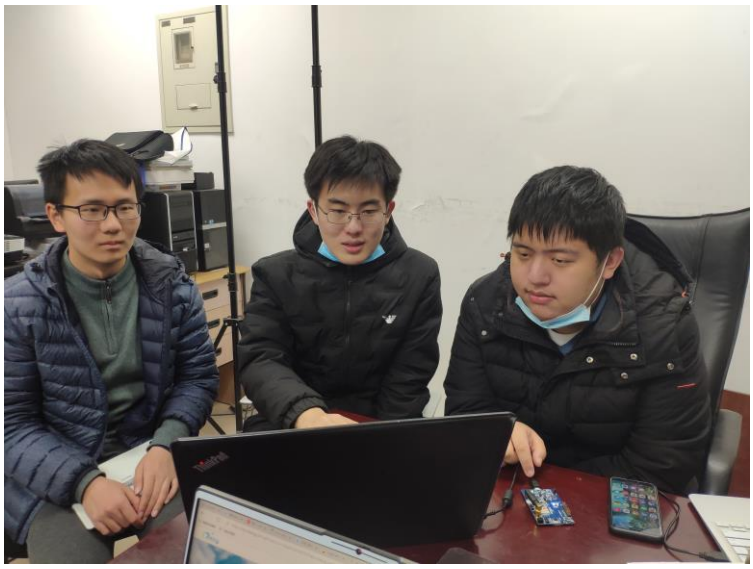
其次，真实设备中的部分配置是通过 STM32 单片机进行的，不排除由于 STM32 单片机中配置程序等问题导致无法建立有效通讯联系。这时可以通过网络调试助手等方式进行错误处理。

3．数据下放到端设备过程无法保证实时

NB-IoT 设备默认具有一定的省电模式，这也是物联网设备的共同的特点。对于任何一种省电模式，都具有相对高功耗的通信时段和相对无功耗的休眠时段，且休眠时段相比通信时段会更长。在设备处于休眠时段时，设备无法接收下行消息，此时消息被保存在平台的消息队列中等待下放。故下放到端设备过程中无法保证实时。但是可以通过修改休眠策略等方式增加通讯时段，满足高时延要求的业务需求，但随之设备功耗会上升，需要酌情考虑。

7. 收获及总结

通过本次实验，我们组在一定的分工与合作下完成了一个有一定实现难度的“智能空调”物联网系统。这不仅提高了我们对物联网基本概念的理解和应用能力，同时也锻炼和培养了我们的团队意识与合作精神。



首先，我们进一步的认识到了物联网的基本概念与特点。作为实验课的主线，NB-IoT 技术贯穿整个实验过程。我们在实验中通过实际情况认识到了该技术的省电方式带来的若干特性等知识，并学习到了若干种实际应用中常见的通讯方式与协议，例如 UDP 协议。

其次，我们也锻炼了客户端应用的开发能力。我们在本学期开设的 Android 课程中学习到了相关的应用开发知识，在本实验中我们结合这些知识，最终打通了物联网平台与客户端应用程序的构建，并以较好的客户端界面和稳定的应用进行了展示。

同时，我们也熟悉了现在常用的物联网平台之一—电信物联网平台。通过依托该平台进行相关设计与实现，我们实现了信息的通讯与交互处理的内容，并借助其提供的接口使得开发更加规范与便利。

开发过程是很艰难的。我们经历了网络通信问题、端设备配置问题等等，最终通过解决上文提到的若干问题，我们实现了既定目标。

最终，当物联网模块成功执行下放的电机指令，应用程序正常获得物联网端设备上报的温湿度传感器信息的时候，我们是非常激动的。这不仅意味着我们达成了预定的目标任务，同时也代表我们以自己的力量成功实现了一个具有一定复杂度的物联网应用系统，实现了物联网端设备-物联网平台-客户端应用程序三者之间的通信目标。