**使用微信小程序结合OneNet平台控制麒麟座开发板**

作者：官靖壹 Github：github.com/textGuan

项目地址：https://github.com/textGuan/wechat-onenet

目录

[一 项目简介 2](#_Toc34922529)

[二 开发环境 2](#_Toc34922530)

[三 开发准备 2](#_Toc34922531)

[3.1 软件及工具 2](#_Toc34922532)

[3.2 账号注册 3](#_Toc34922533)

[3.3 设备注册 3](#_Toc34922534)

[3.4 网络准备 4](#_Toc34922535)

[3.5 例程下载 5](#_Toc34922536)

[四 开发板程序 5](#_Toc34922537)

[4.1 为项目添加必要的文件 5](#_Toc34922538)

[4.2 修改代码 7](#_Toc34922539)

[五 小程序配置 13](#_Toc34922540)

[5.0 界面和功能设计 13](#_Toc34922541)

[5.1 项目配置 15](#_Toc34922542)

[5.2 完善文件目录和初始设置 17](#_Toc34922543)

[5.3 页面绘制 20](#_Toc34922544)

[5.4 函数实现 28](#_Toc34922545)

[六 项目总结 48](#_Toc34922546)

[6.1 跨页面的提示功能 48](#_Toc34922547)

[6.2 响应时间问题 48](#_Toc34922548)

[6.3 数据展示只能固定展示最新十条数据 49](#_Toc34922549)

[6.4 数据展示不够直观 49](#_Toc34922550)

[6.5 待补充 49](#_Toc34922551)

[七 感言 49](#_Toc34922552)

[八 参考资料 49](#_Toc34922553)

## 一 项目简介

近日，在四川大学杨彦兵老师的指导和帮助下，学习了中国移动OneNet麒麟座开发板的使用，并成功连接到中移OneNet物联网开放平台，使用微信小程序实现了一个简单的数据更新和设备管理的Demo。主要功能为开发板板载传感器数据的获取和开发板板载蜂鸣器、LED灯的简易控制。

开发此项目主要目的是熟悉微信小程序和STM32开发板的使用，更好地准备就业工作。疫情期间，由于无法按照预期计划前往实训基地实习，因此提前学习了课程，于天猫购入了麒麟座V3.2开发板。简单熟悉了基本操作和官方例程之后，决定使用微信小程序简单模拟智能家居的使用，包括数据的获取（例如家庭内温度获取）、设备的即时控制和设备的定时控制等。

主要实现逻辑为，利用微信小程序使用http协议与OneNet平台通信，使用EDP协议下发指令给开发板。开发板使用EDP协议传输数据到平台，小程序使用http协议获取数据并展示。

## 二 开发环境

Keil 5.27，微信开发者工具v1.02，期间还使用到了串口调试工具sscom33

## 三 开发准备

### 3.1 软件及工具

在刚刚拿到开发板时，首先选用的是Keil 5.0版本，出现了程序卡死的情况，后换成5.27版本，正常。移动官方给出的信息是，“但是如果需要用我们的开发板例程的话，就要去下载keil 5.24及其以上的”，由于本项目基于官方例程，因此版本最好高于5.24。使用keil自带的pack installer安装stm32f1xx的pack，下载较慢的可以单独下载文件安装。

针对本项目而言，还需准备一条具有数据传输功能的micro-usb数据线，用于串口输出，便于及时查看输出信息和调试，如果st-link驱动没能成功自动安装的，还需安装好st-link驱动。

其他配置信息教程文件夹内已阐述，按照说明配置即可。

下载安装微信开发者工具。

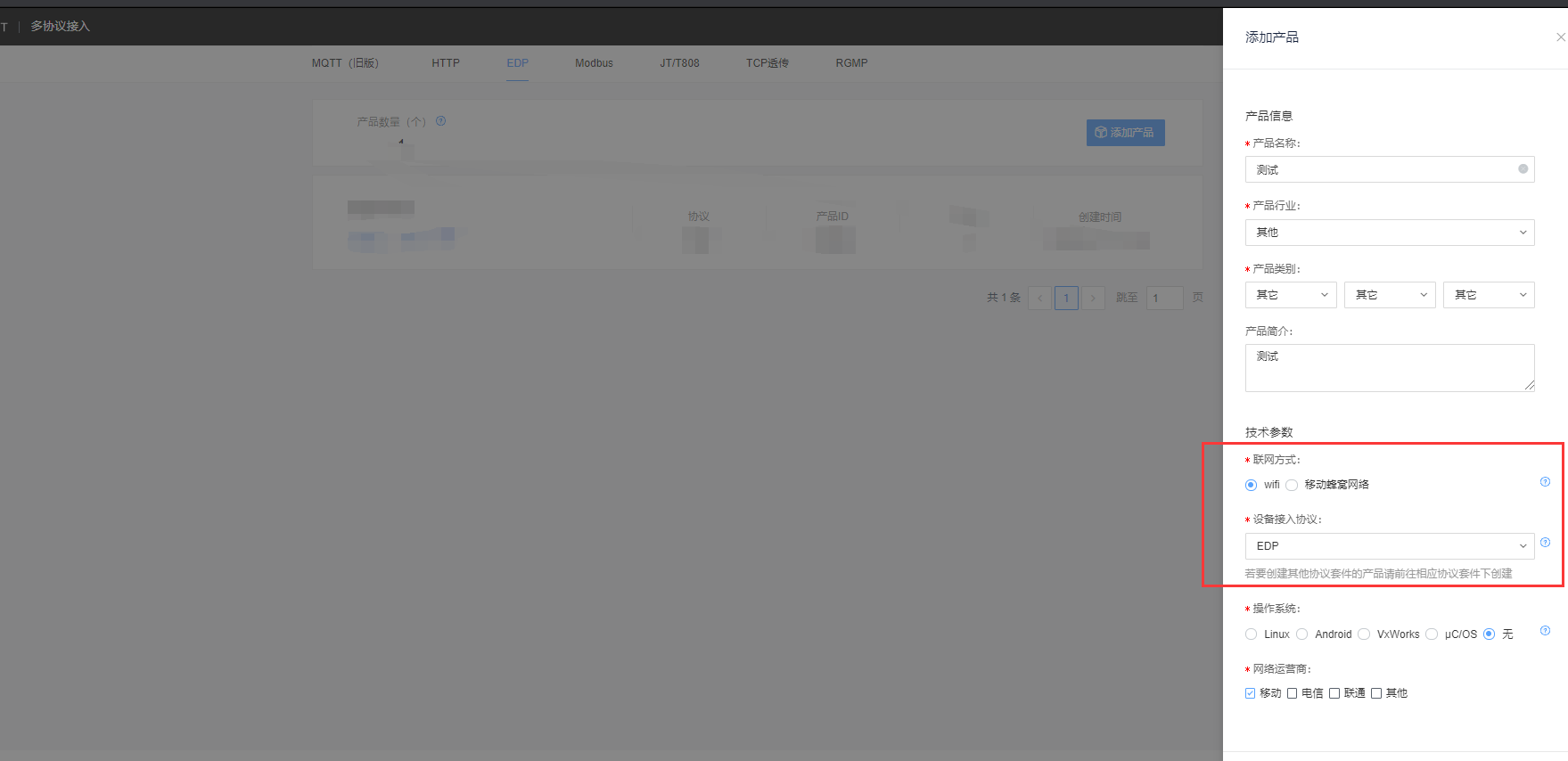
### 3.2 账号注册

项目如需上线的，需要前往微信公众平台注册开发者账号和小程序，此处仅使用测试号调试，因此以上步骤可跳过，直接使用微信扫码登陆即可。

需要前往中移OneNet平台注册账号。

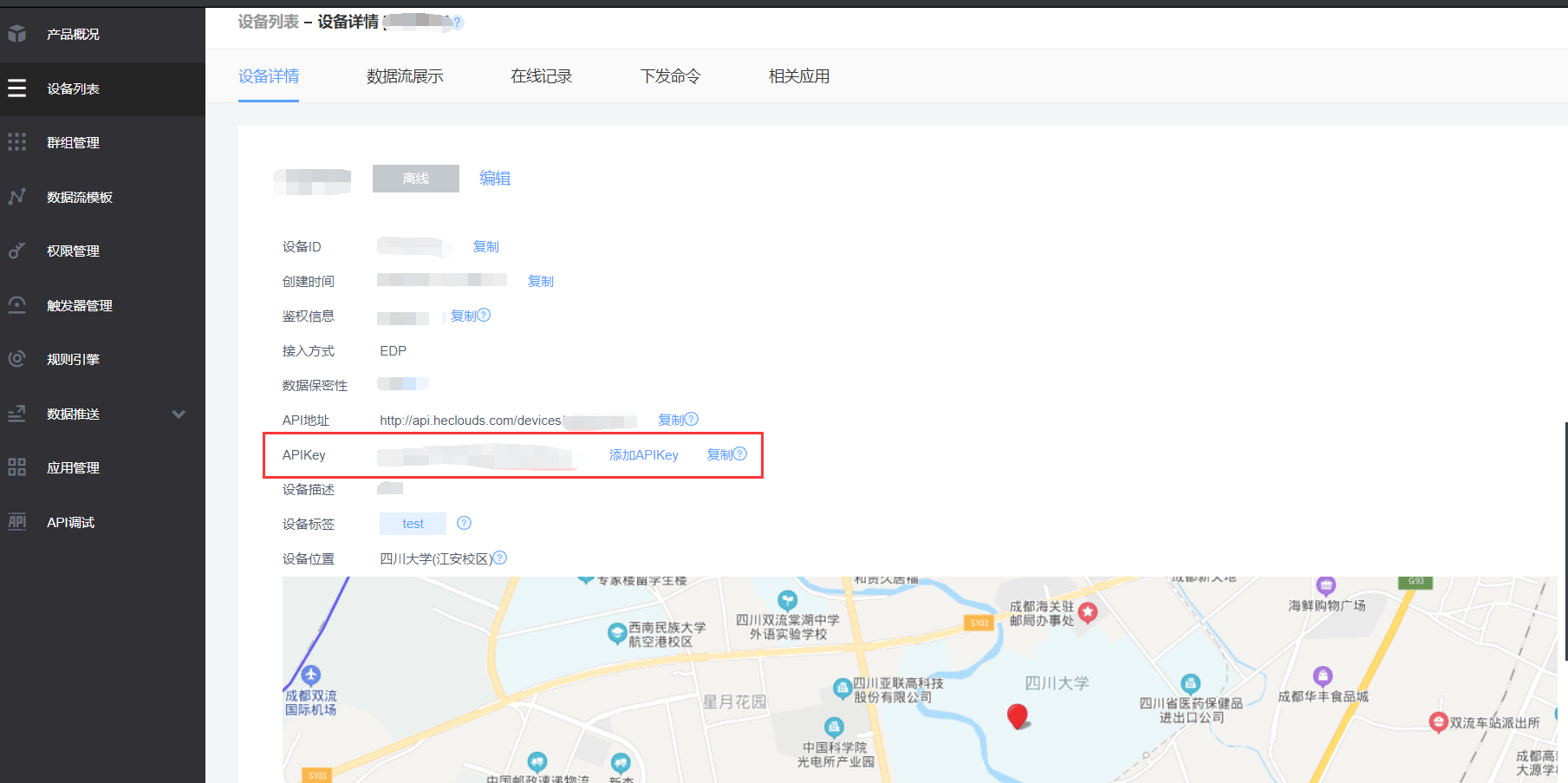
### 3.3 设备注册

任意设备均可免费连接至中移OneNet平台，由于本项目需要使用edp协议实现开发板与OneNet平台通信，因此进入开发者中心-全部产品-多协议接入-EDP-添加产品，如图所示添加设备即可。注意方框内选项。



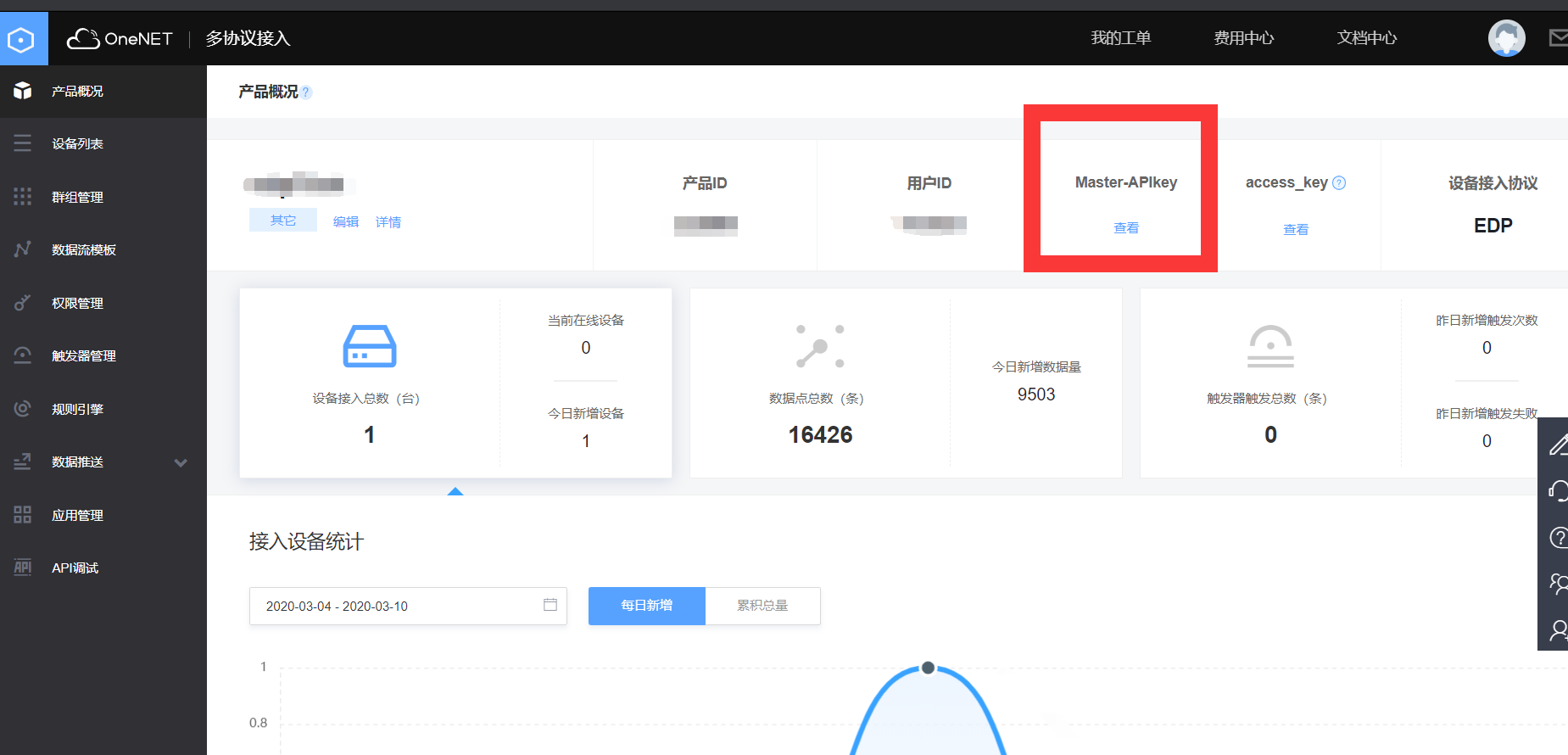
图：设置通信协议

注册完毕后，在设备详情内添加APIKEY。添加完成后，如下图所示。记录APIKey和设备ID信息。



图：APIKey和设备ID

在产品概况下查看Master-APIKey，如下图所示。该操作可能需要手机验证码，所以请使用常用手机账号注册OneNet平台。



图：查看Master-Key

总结：完成以上步骤后，需要记录的数据有：

**产品概况：Master-APIKey**

**设备详情：设备ID，APIKey**

### 3.4 网络准备

开发板似乎不支持5G路由，部分2.4G路由器亦无法接入，因此建议使用手机热点并设置为2.4GHz段，记录热点名称和密码。

### 3.5 例程下载

下载中国移动给出的官方例程中对应版本部分，其他开发板暂无例程下载。

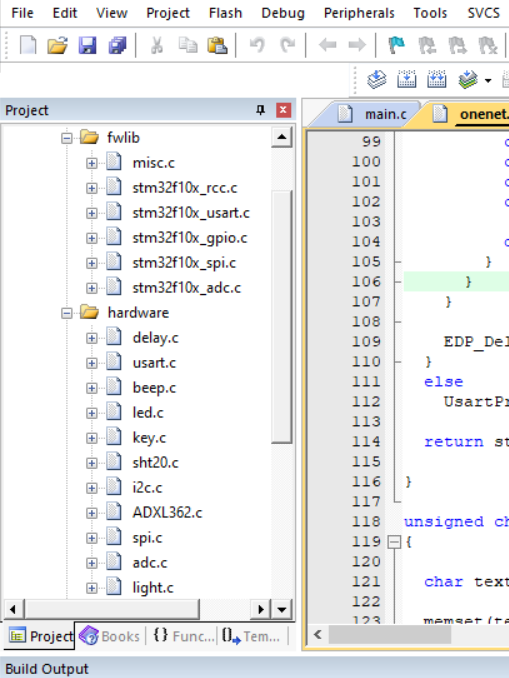
至此，开发准备工作已经完成。

## 四 开发板程序

首先打开ESP8266-EDP\_TYPE3-LED工程

### 4.1 为项目添加必要的文件

首先对照如下目录分别复制必要的.c文件和.h文件至工程文件夹hardware文件夹的src文件夹（.c文件）和inc文件夹（.h），右键Project-STM32F103-Manage Project Items，为hardware添加对应的文件。例如，添加sht20.c文件，首先复制sht20.c文件到hardwar/src，复制sht20.h文件到hardware/inc，右键Project-STM32F103-Manage Project Items，选择hardware，选择add files，选择刚刚复制的hardwa/src/sht20.c文件即可。同理，为fwlib添加stm32f10x\_spi.c、stm32f10x\_adc.c文件，该文件已经在fwlib文件夹下。



图：嵌入式项目结构

添加完毕后在onenet.c和main.c文件内引入如下所示的硬件文件（原文件已有部分）



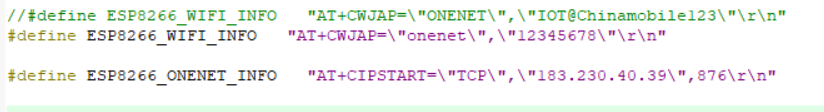
图：onenet.c



图：main.c

### 4.2 修改代码

首先修改onenet.c文件中声明的DEVID为设备ID，修改APIKEY为设备详情内的APIKEY。打开esp8266.c文件，将ESP8266\_WIFI\_INFO中对应的网络名称和密码设置为自己的名称和密码，以本人手机热点为例，本人热点名称为onenet，密码为12345678，则修改为如图所示



图：设置WIFI账号和密码

此时可编译下载测试，若配置正确，设备蜂鸣器将鸣响250ms，并可在onenet平台看到对应的数据流（Red\_Led等）。

#### 4.2.1 硬件初始化

在main.c文件的Hardware\_Init()函数中添加

ADXL362\_Init();

IIC\_Init();

LIGHT\_Init();

若报错，请检查是否正确添加文件到项目，添加后如图所示：



图：添加初始化语句到项目

#### 4.2.2 添加获取数据语句

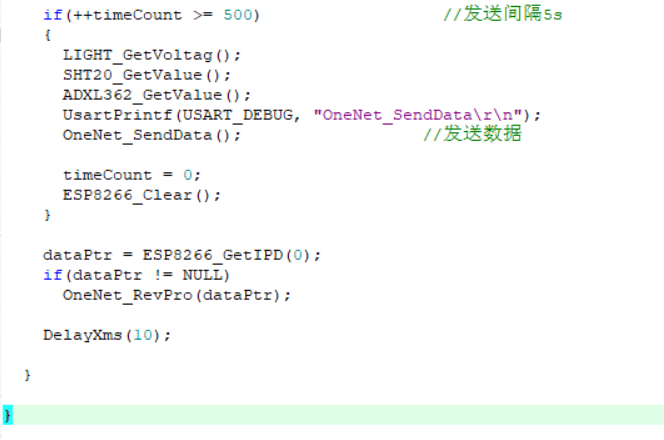
在main.c的main函数的if(++timeCount >= 500)内添加以下语句：

LIGHT\_GetVoltag();

SHT20\_GetValue();

ADXL362\_GetValue();

同理，若报错，请检查是否正确添加文件到项目，下文不再阐述。添加后如图所示：



图：添加获取数据语句到项目

#### 4.2.3 关于main.c文件

官方例程中键与LED灯非按位对等关系，有强迫症的可以修改。

至此，main.c文件修改完成。

#### 4.2.4 数据写入缓存

打开onenet.c文件，在OneNet\_SendData()函数内，修改char buf[128];为char buf[256];此处修改的目的是，避免出现EDP\_NewBuffer.Failed错误导致数据无法正常上传。

在OneNet\_FillBuf()函数内，添加对应的数据。格式如可以参考以下代码或参考原有代码。

memset(text, 0, sizeof(text));

sprintf(text, "\"Red\_Led\":%d,", led\_status.Led4Sta);

strcat(buf, text);

memset(text, 0, sizeof(text));

sprintf(text, "\"Green\_Led\":%d,", led\_status.Led5Sta);

strcat(buf, text);

memset(text, 0, sizeof(text));

sprintf(text, "\"Yellow\_Led\":%d,", led\_status.Led6Sta);

strcat(buf, text);

memset(text, 0, sizeof(text));

sprintf(text, "\"Blue\_Led\":%d,", led\_status.Led7Sta);

strcat(buf, text);

memset(text, 0, sizeof(text));

sprintf(text, "\"Beep\":%d,", beep\_info.Beep\_Status);

strcat(buf, text);

memset(text, 0, sizeof(text));

sprintf(text, "\"Light\":%0.2f,", light\_info.voltag);

strcat(buf, text);

memset(text, 0, sizeof(text));

sprintf(text, "\"Temperature\":%0.2f,", sht20\_info.tempreture);

strcat(buf, text);

memset(text, 0, sizeof(text));

sprintf(text, "\"Humidity\":%0.2f,", sht20\_info.humidity);

strcat(buf, text);

memset(text, 0, sizeof(text));

sprintf(text, "\"Xg\":%0.2f,", adxl362Info.x);

strcat(buf, text);

memset(text, 0, sizeof(text));

sprintf(text, "\"Yg\":%0.2f,", adxl362Info.y);

strcat(buf, text);

memset(text, 0, sizeof(text));

sprintf(text, "\"Zg\":%0.2f", adxl362Info.z);

strcat(buf, text);

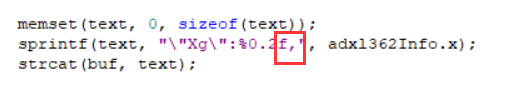
对应的含义如下表所示：

|  |  |
| --- | --- |
| 值 | 含义 |
| led\_status.Led4Sta | 红色LED灯打开情况，1代表开启，0代表关闭 |
| led\_status.Led5Sta | 绿色LED灯打开情况，1代表开启，0代表关闭 |
| led\_status.Led6Sta | 黄色LED灯打开情况，1代表开启，0代表关闭 |
| led\_status.Led7Sta | 蓝色LED灯打开情况，1代表开启，0代表关闭 |
| beep\_info.Beep\_Status | 板载蜂鸣器打开情况，1代表开启，0代表关闭 |
| light\_info.voltag | 光敏电阻获取的电压值（电阻值？） |
| sht20\_info.tempreture | 温度 |
| sht20\_info.humidity | 湿度 |
| adxl362Info.x | X轴加速度 |
| adxl362Info.y | Y轴加速度 |
| adxl362Info.z | Z轴加速度 |

**需要注意的：**

①以sprintf(text, "\"Red\_Led\":%d,", led\_status.Led4Sta);为例，Red\_Led为数据流ID，%d标识数据为整型，led\_status.Led4Sta表示数据值，其中数据流ID和格式可以自定义

②最终上传至平台的数据为json格式，如果出现数据解析错误会导致设备自动离线，因此请注意，除去最后一个数据外，每个数据字符串均以英文逗号结尾，如图所示



图：注意逗号的使用

#### 4.2.5 读取下发的命令

在onenet.c的OneNet\_RevPro()函数中的if else语句段中，增加以下语句：

else if(strstr((char \*)req, "beep"))

{

if(num == 1)

{

Beep\_Set(BEEP\_ON);

DelayXms(250);

Beep\_Set(BEEP\_OFF);

DelayXms(250);

Beep\_Set(BEEP\_ON);

DelayXms(250);

Beep\_Set(BEEP\_OFF);

}

else if(num == 0)

{

Beep\_Set(BEEP\_ON);

DelayXms(250);

Beep\_Set(BEEP\_OFF);

}

}

语句内容可自定义，此处设定的是当指令为打开蜂鸣器，蜂鸣器鸣两次，当指令为关闭蜂鸣器，蜂鸣器鸣一次（因为一直鸣笛我受不了）。

至此，开发板代码修改完成。编译下载到开发板，可以在OneNet\_FillBuf()函数内添加UsartPrintf(USART\_DEBUG, buf);语句，通过串口调试助手查看每次上传的数据。如配置正确且数据无误，此时可以在onenet平台查看到相应的数据流。手动点击按钮改变LED状态后，一段时间后数据同步完成可以查看到对应的LED灯数据流从0变为了1。此时也可以通过onenet平台下发指令控制。

## 五 小程序配置

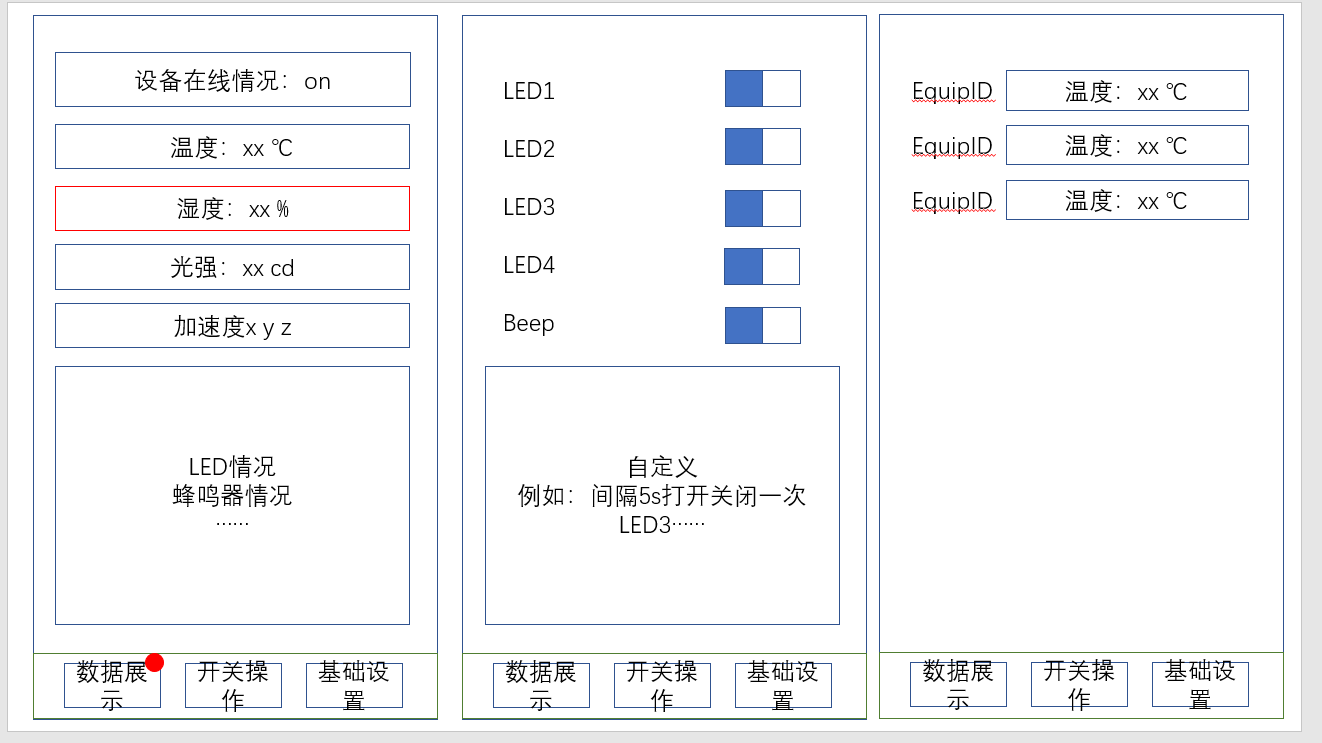
### 5.0 界面和功能设计

个人认为，无论是做什么程序，这一步都必须是最先完成的。尤其是接触新事物时，首先明确好需求，对症下药，能够有效提高开发效率。

**首先是功能，作为一个智能家居系统，那么我们希望他至少有以下几个功能：**

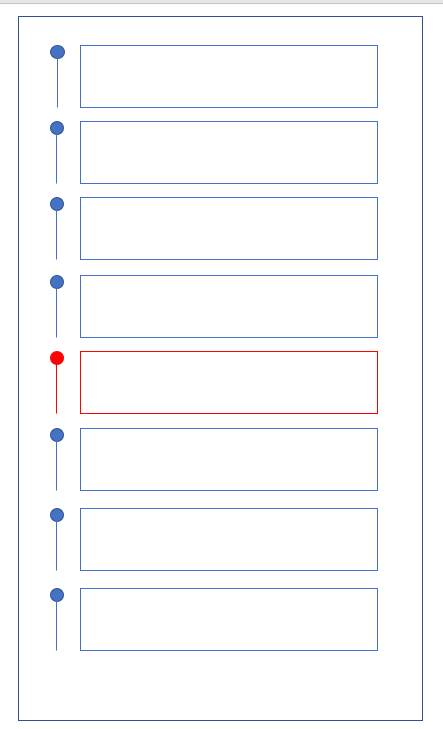
1. 数据获取，定时更新，例如，你想要获取你房间当前的温度。这个数据最好是持续了一段时间的。
2. 设备控制，例如，你希望在床上能够启动远在三间房间以外的热水器。
3. 报警，例如：异常闯入或者温度、烟雾异常（可能有失窃、火灾风险）
4. 定时设备控制，例如，当你外出旅游时，你可能希望每天晚上八点定时开灯营造家里有人的假象。

明确了功能之后，我们使用PowerPoint根据功能需求绘制出了一个简陋的界面（最终的效果确实就是这么简陋），如图所示：



图：简单的页面设计

对于左侧的湿度、温度等，可以点击打开数据详情页

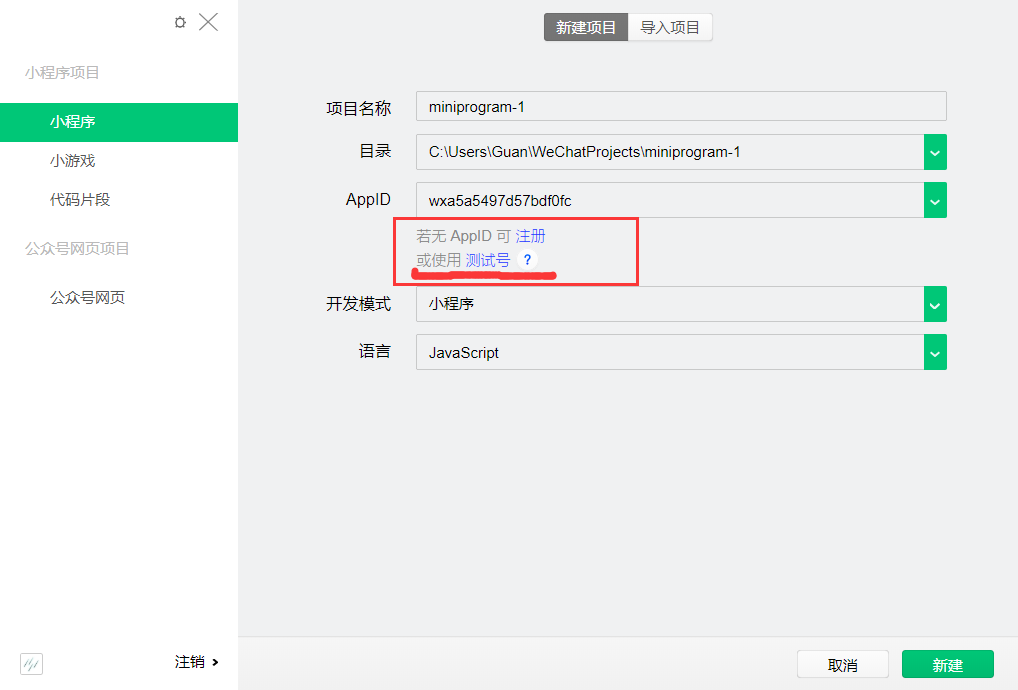


图：简单的数据详情页

明确了功能和大体的设计之后，我们就可以动工了。

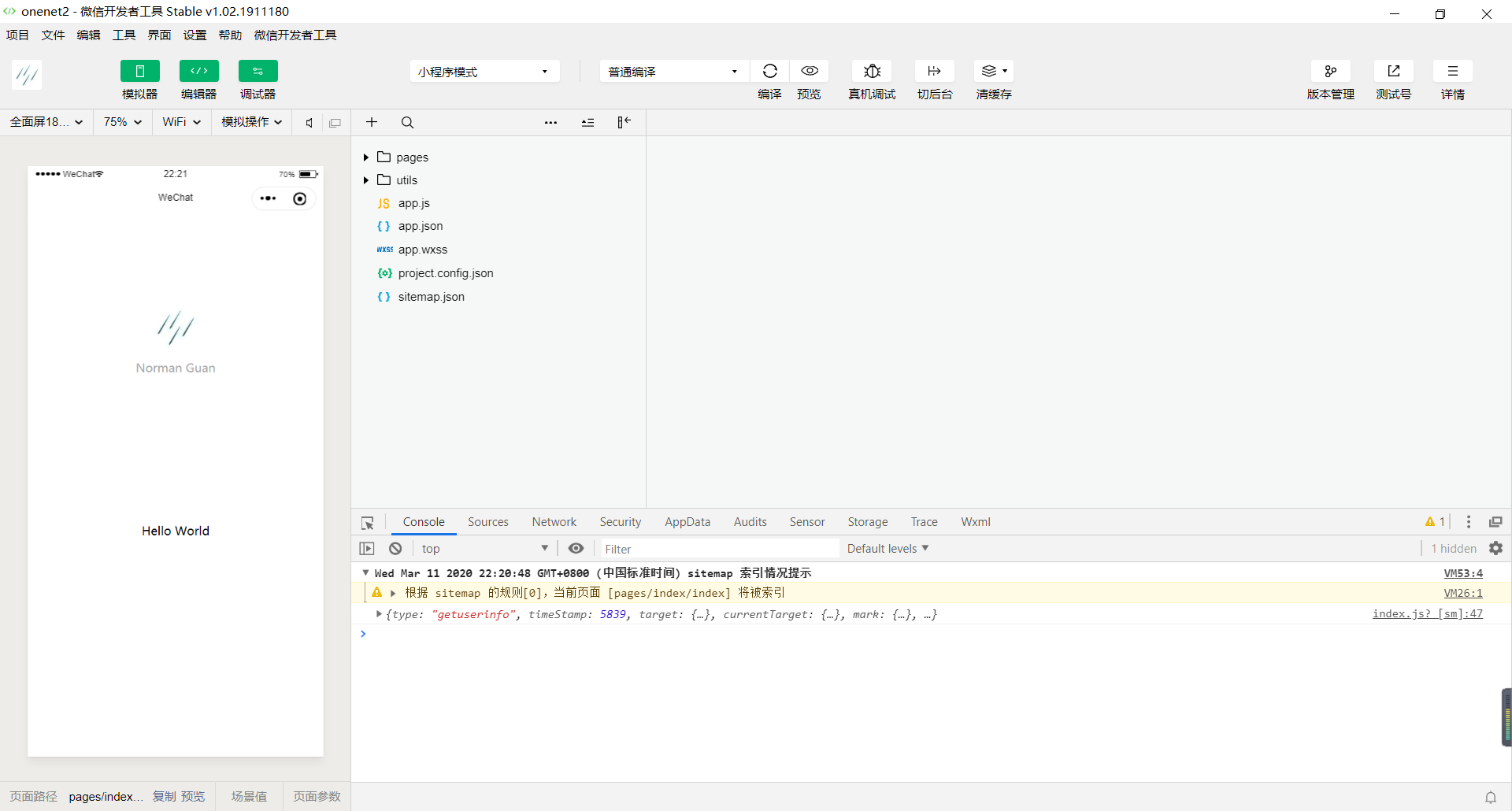
### 5.1 项目配置

首先新建小程序项目，AppID可先选择使用测试号，如需上线再设置为在微信公众平台申请的AppID，项目名称命名为自己喜欢的名字，同理目录，语言此处选择JavaScript，如图所示：



图：创建小程序项目

新建项目成功后可以看到如图所示的界面，左边也可能是“获取头像和昵称”之类的按钮



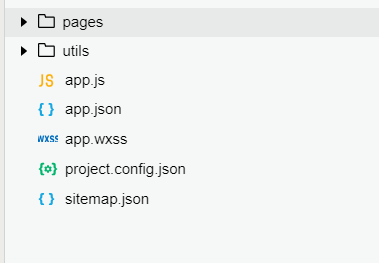
图：程序创建完成

单击菜单栏设置-项目设置-勾选“不校验合法域名、web-view（业务域名）、TLS版本以及HTTPS证书”。（重要：因为我们请求的url是http的，如果不勾选请使用https域名请求，否则会报错）

（囿于时间关系，不再重新写代码，直接以现有的代码讲解，所以可能有些步骤漏掉了。如果以后有时间录视频的时候再重新写……）

### 5.2 完善文件目录和初始设置

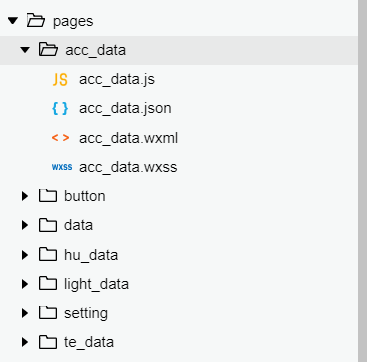
当我们新建好项目时，项目是这样的



图：微信小程序初始项目结构

Pages目录下有index和log两个页面，暂时用不到他们，可以将他们删除，然后打开app.json，在pages中将这两个页面对应的项删去。

根据在5.0中明确的大概设计，我们一共需要7个页面，其中四个页面为数据详情页。所以我们右键pages，首先新建7个目录，再依次点击这7个目录，选择新建页面，目录和页面名称自定义，开发者工具会自动将新添加的页面配置到app.json的pages中，最终的效果如图所示：



图：为小程序创建Page

这里我将tabbar切换的三个页面命名为data、button和setting，te\_data、hu\_data、light\_data和acc\_data分别代表温度、湿度、光照和加速度数据详情页。或者你也可以手动添加js、wxml、wxss和json文件，但是此时，开发者工具不会帮你自动配置到app.json的pages中，需要手动配置。

对应的文件具体功能不详细阐述（因为我也不熟悉），有兴趣的可以查阅微信官方文档。

以上操作完成后，在app.json中设置"navigationBarTitleText": "OneNet-微信小程序",（不是必须），添加tabBar，我将data、button和setting页面设置在了tabBar中，并且为各自设置了初始为空的badge（小红点，如图所示）。需要注意的是，如果当前展示的页面不在tabBar中，那么tabBar将不会显示。



1. 导航栏标题
2. tabBar
3. 小红点badge

在app.json中还可以针对小程序做进一步的美化和设置，此处做的修改是全局性的，如果需要对某个页面例如（data）做单独设置的（例如将标题设置为data），则需单独在data.json中设置而非在app.json中操作，详情请查看官方文档。

小程序中，首先渲染的页面是app.json中pages配置的第一个页面，所以请按照需求调整页面顺序，注意逗号的使用。

至此，文件目录和初始设置完成。接下来是逐个页面的绘制工作。

### 5.3 页面绘制

#### 5.3.1 data页

以我已经做好的demo为例。页面大体分为三部分，第一部分是对页面的大体介绍，第二部分是最新数据的展示，第三部分是当前设备的状态。如图所示：



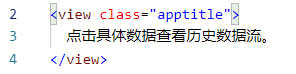
图：data页案例

每一部分又有不同的设计，例如2中的第一行 587632125 离线，这一行向用户展示了设备编号和在线情况，此处没有边框，而第二行温度 20.04 ℃，向用户展示了最新的温度情况。由于温度超过了报警的阈值，因此边框被设置成了红色。

微信小程序与网页极为类似，使用类html语言和类css语言完成界面的绘制（微信官方进行了一定的改动）。

首先在data.wxml文件中设置这些组件：

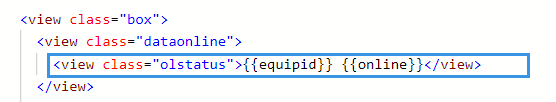
在这个页面中，有静态的部分和动态的部分。例如，“点击具体……”这一行文字可以设置为静态数据，因此我们可以直接按照下图所示的方法用view组件配置。



图：为view组件设置静态数据

注意此处的class，我们需要借助这个class对样式进行设置，因此建议每一个组件都有一个class，不同的组件可以公用同一个class。

而对于动态的部分，例如587632125 离线一行，左侧的587632125数据来自于设备列表，由设备列表选取对应的一个设备，而设备列表应该是动态更新的（本项目中由于只有一台设备所以实际上设置的静态数据）。“离线”表示设备的在线情况，显然这个数据也应该是动态更新的，对于动态变化的数据，微信小程序使用{{变量}}的方式加载，如图所示：



图：为view组件设置动态数据

变量应该在页面对应的js文件中的Page的data内设置，这里我们为online变量设置一个初始值为“检测中”，注意data内，设置变量值的方式为 变量名:值 的形式，注意逗号的使用。



图：在线状态

按照以上思路，温度一行可以设置为

<view class="datashow" bindtap="te\_data" style="border: {{color1}} solid 1px;">

<view class="title">{{name[0].title}}</view>

<view class="num">{{name[0].data}}</view>

<view class="unit">{{name[0].unit}}</view>

</view>

注意此处，我使用了一个数组存储温度、湿度、光照电压等信息（实际上数组的使用个人认为是不友好的，之后会讲到），因此您看到的我是通过name[0].title的形式得到了“温度”这个字符串。bindtap="te\_data"这一句写明了单击这个view组件时触发的事件，事件名称为te\_data。此处将本应该写在wxss文件中的border写在此处，为的是实现动态改变颜色的功能，之后会讲到。熟练使用{{变量}}的方式可以帮助您写出更加精美的程序。

按照此处的逻辑，我完成了其他组件的代码，由于点击温度、湿度、光照电压和加速度会跳转进不同的页面，同时彼此之间报警是独立的，因此我为这些组件配置了独立的单击事件和颜色。

最终的代码您可以在我的github页面查看，您也可以根据自己的喜好设置属于自己的组件。

完成wxml文件之后，我们为这些组件设置样式。以上述的温度组件为例，温度组件wxml代码如下：

<view class="box">

……

<view class="datashow" bindtap="te\_data" style="border: {{color1}} solid 1px;">

<view class="title">{{name[0].title}}</view>

<view class="num">{{name[0].data}}</view>

<view class="unit">{{name[0].unit}}</view>

</view>

……

</view>

可以看出，对于温度数据而言，class为num的组件实际上在datashow的view组件内，而datashow组件又在box组件内，因此按照层次配置相应的样式：

在wxss文件中，使用.classname{ attr1:xxx; attr2:xxx; }的形式配置，这里以温度数据为例：

首先是box，即图示中上半部分的黑色边框：

.box{

margin: 0 auto; //横居中

margin-bottom: 20rpx;//下边距为20rpx

height: 620rpx;//高度为620rpx

width: 80%;//宽度为父组件的80%，若当前组件没有父组件，则表示为窗口宽度的80%

border: black solid 1px;//边框黑色实线，宽度为1px

align-items: center;// flex子项在flex容器的当前行的侧轴（纵轴）方向上的对齐方式

border-radius: 20rpx;//设置圆角边框

}

代码注释表示该行功能，（说到注释，注意.json文件不支持注释）其实我对wxss和css语法不是很熟悉，大部分是边写边查文档边试做出来的，这里面可能有很多冗余的代码。

.box .datashow{

width: 80%;

margin: 0 auto;

display: flex;//flex布局

height: 80rpx;

line-height:80rpx;//文字行高为80rpx，设置为与height相同可以实现居中对齐的效果

margin-top: 20rpx;

border-radius: 10px;

}

对box组件中的datashow组件进行配置，同样地，注释表示功能

.box .datashow .data{

width:40%;

text-align: center;//文字居中

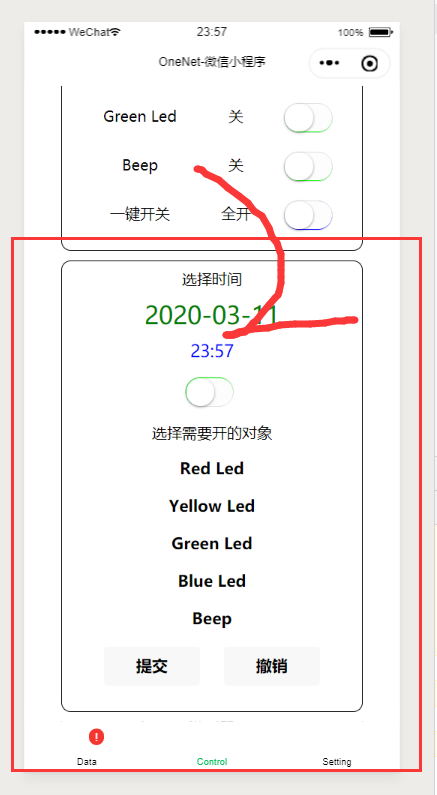
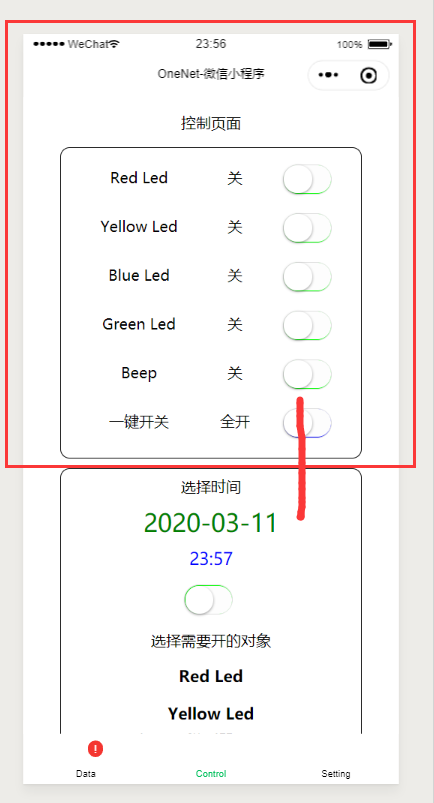
margin: 0 auto;

}

对box组件中的datashow组件中的data组件进行配置。

以上述逻辑完成剩余组件的设置，我的代码您可以在github上获取。关于rpx，您可以在微信官方文档中得到详细的解释。由于仅仅是一个demo，因此本项目中对于很多东西的处理都不可复用。

#### 5.3.2 button页



图：button页展示

与data页类似的，title不再阐述，此处主要针对图示的两个模块。

首先是第一个模块——即时控制模块，其每一行的组成为：一个view组件显示名称，一个view组件显示状态，一个switch组件控制操作。以其中一行分析：

<view class="one">

<view class="name">Yellow Led</view>

<view class="status" wx:if="{{yellowledstatus}}">开</view>

<view class="status" wx:else>关</view>

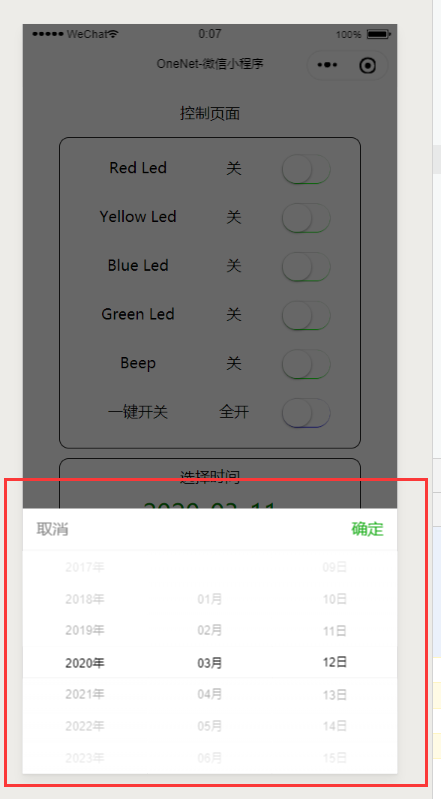
<switch class="switch\_s" type="switch" checked="{{yellowledstatus}}" color="#00ff00" bindchange="yellowledchange"></switch>

</view>

每一行实际是一个view组件，名字可以是固定的，也可以从数组中读取动态变化的。此处使用了wxml中的if else语句，当yellowledstatus为true时渲染“开“，否则渲染”关“。

Switch组件的checked表示根据值决定要不要成为选择状态，color表示颜色，bindchange表示切换事件。其他组件和title不再描述。

第二个模块中，时间使用的picker组件，效果如图所示



图：日期选择

以日期选择为例：

<picker class="datepicker" mode="date" value="{{cur\_date}}" start="{{today}}" end="2099-12-31" bindchange="changeDate">{{cur\_date}}</picker>

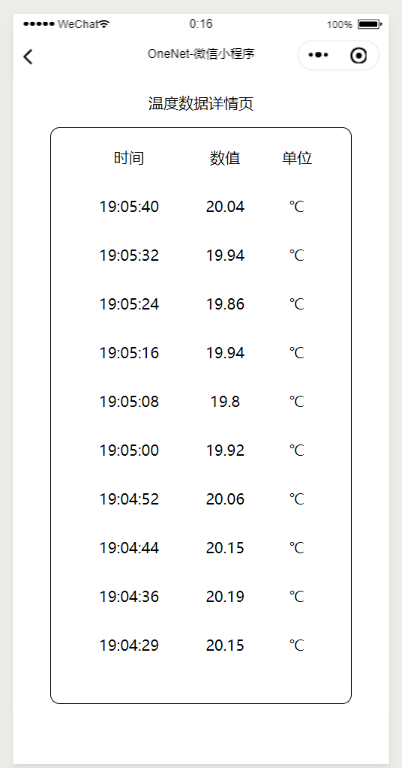
Mode表示模式，此处，日期选择，mode设置为date，时间选择则mode设置为time，value为值，开始结束的区间通过start和end决定，例如，为了合乎逻辑，日期选择的范围应该是今天开始，今天之前的日期不能被选择。

Red Led等和提交、撤销均为button组件，这里将Red Led等的颜色动态设置，方法在data页的绘制中已经提到。样式设置大同小异，因此不再描述。

#### 5.3.3 其他页面

由于有且仅有一台开发板，因此setting页本做有一个设备切换的页面，但是没有配置，因此也不再描述。

数据详情页面如图所示，以温度为例：



图：温度详情页面

此处设定为一页固定展示最新10条记录，其他详细页面同。此处使用了for循环方法渲染，完整wxml代码如下：

<view class="title">温度数据详情页</view>

<view class="box">

<view class="one">

<view class="time">时间</view>

<view class="data">数值</view>

<view class="unit">单位</view>

</view>

<view class="one" wx:for="{{dataList}}">

<view class="time">{{item.time}}</view>

<view class="data">{{item.data}}</view>

<view class="unit">{{item.unit}}</view>

</view>

</view>

其中wx:for="{{dataList}}"表示在dataList数组中循环，通过item的方式逐一访问数组中的每一个成员。

按照以上逻辑完成其他页面的绘制，至此，页面绘制工作完成。

### 5.4 函数实现

**onShow()：页面被展示**

**onLoad()：页面被加载**

**onUnload()：页面被卸载**

**onHide()：页面被隐藏，本项目主要的实现是在onShow()内，onUnload()和onHide()只需要清除定时器的操作。**

#### 5.4.1 data页面

##### 5.4.1.1 页面跳转

点击数据会跳转进展示最近十条数据的页面，示例代码如下，注意括号的使用，下同：

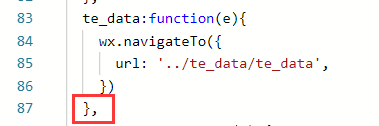
te\_data:function(e){

wx.navigateTo({

url: '../te\_data/te\_data',

})

}



图：注意逗号的使用

##### 5.4.1.2 获取在线数据

查看OneNet开发者文档可知，获取在线数据应当使用GET方法，请求URL为http://api.heclouds.com/devices/device\_id，因此使用wx.request()方法，url为请求url，注意将device\_id替换为自己的设备id，获取设备id的方法在3.3中已经提到。包头中设置请求格式为'application/x-www-form-urlencoded'，并携带master-apikey验证具有权限，此处data为空，若请求成功，则执行success(res){}内的代码，res为请求得到的返回数据。请求示例如下：

wx.request({

url: 'http://api.heclouds.com/devices/'+that.data.equipid,

header:{

'content-type': 'application/x-www-form-urlencoded',

"api-key": that.data.apikey

},

data:{

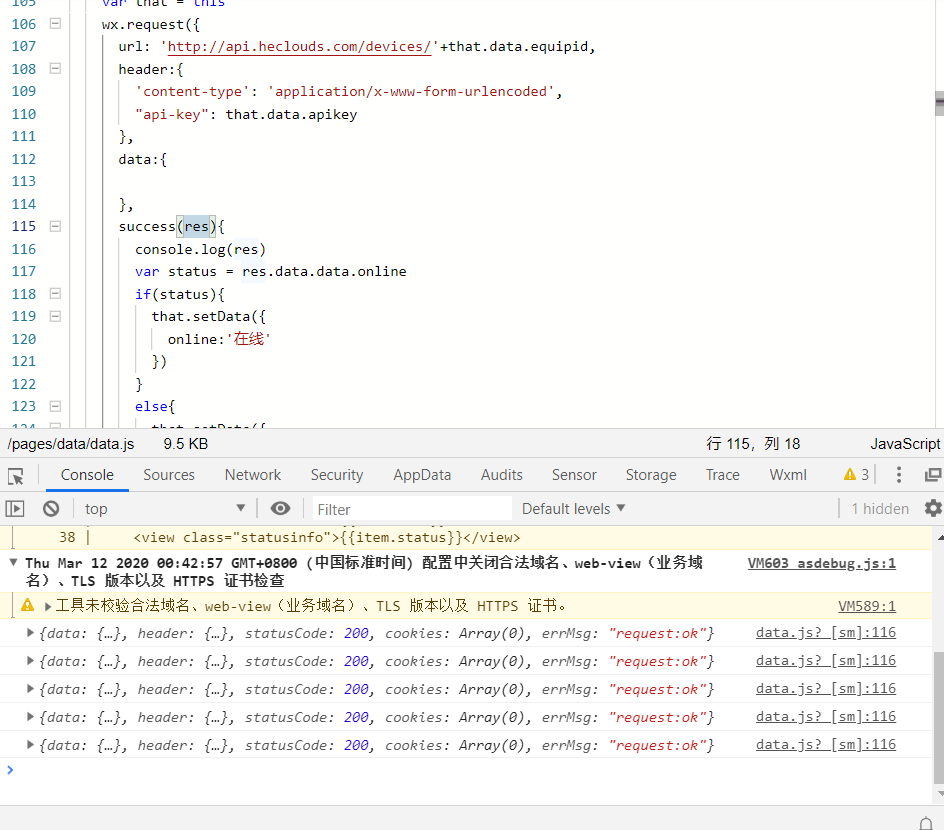
},

success(res){

//action

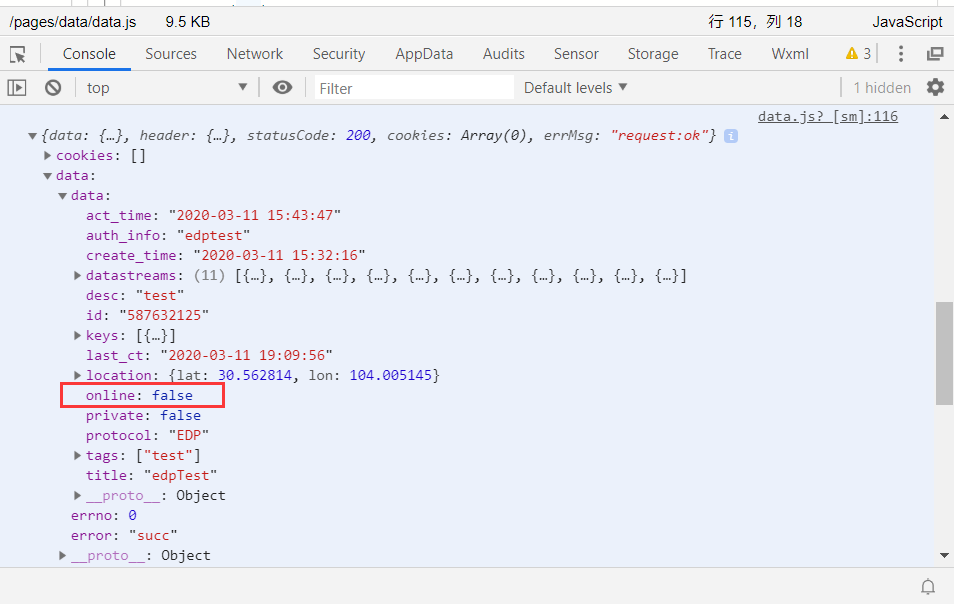
}

})

可以直接查阅文档，得知在线信息可以通过res.data.data.online的方式获取，也可以使用输出res查看的方式得到。使用console.log(res)的方式将res打印到控制台

图：请求信息打印到控制台

可以看到下方的控制台中出现了数据（由于我已经设置了每隔3s执行一次，因此打印出了多次），展开可以得到数据详情：



图：数据展开

可以看到返回的值还包括数据流ID、设备ID、传输协议等等，此处需要的是在线状态，因此只取online数据段，使用setData()函数修改在data中定义的变量。注意，在请求之前，使用var that = this修改了作用域，因此setData应写作that.setData()，在data中定义的变量不能直接通过xxx = yyy 的形式修改，必须通过setData的形式。

完整的获取在线数据的函数代码如下所示：

onGetData:function(){

var that = this

wx.request({

url: 'http://api.heclouds.com/devices/'+that.data.equipid,

header:{

'content-type': 'application/x-www-form-urlencoded',

"api-key": that.data.apikey

},

data:{

},

success(res){

var status = res.data.data.online

if(status){

that.setData({

online:'在线'

})

}

else{

that.setData({

online: '离线'

})

}

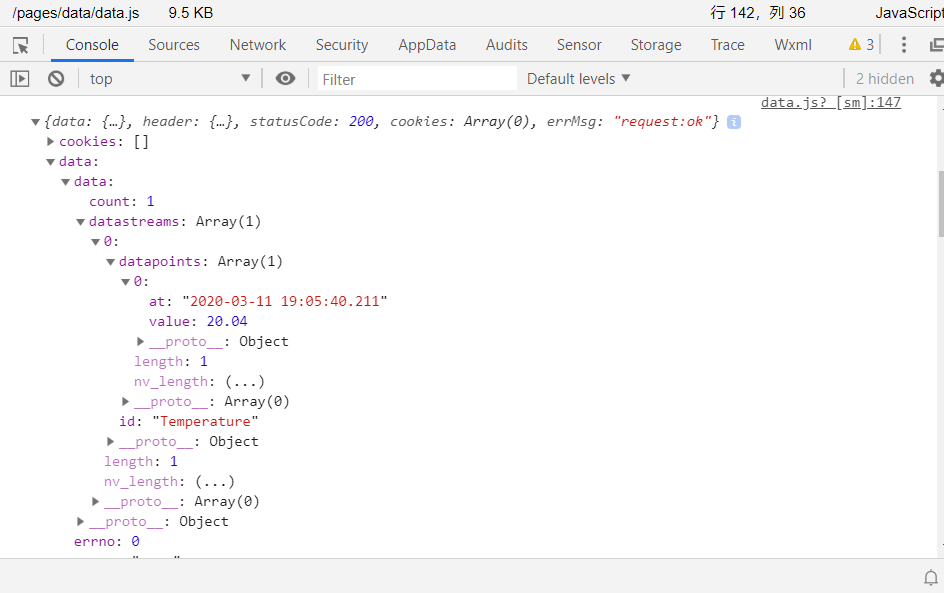
}

})

}

##### 5.4.1.3 获取数据流数据（包括LED灯、传感器和蜂鸣器）

同理，查看开发者文档得知，使用GET的请求方式，请求URL为http(s)://api.heclouds.com/devices/device\_id/datastreams/datastream\_id，其中device\_id和datastream\_id需要替换为设备ID和数据流ID（数据流ID在嵌入式软件代码中定义）。项目中，为了开发的便利性，此处使用的是查询数据流的方式，将请求url替换为http(s)://api.heclouds.com/devices/device\_id/datapoints，根据需要携带不同的URL参数，具体请参考官方文档。同样地，使用console.log(res)语句将返回的数据打印到控制台并展开：



图：打印请求温度的数据

注意其中的Array(1)，此处代表需要以下标的形式访问，最终我们得知温度数据值应通过res.data.data.datastreams[0].datapoints[0].value的形式获取。如果数据流id发生了变化，请求在平台找不到需要的值，同样会得到返回数据，但是res.data.data.count值为0，因此在success中，加上判断语句。由于我之前将温度及其数据的值放在了数组里面，而setdata并不支持直接修改数组中某一项的值，所以需要先将数组复制给一个变量，这个变量的值与原来的数组是完全相同的，再修改这个变量中对应项的值，这个时候可以直接使用=修改了。最后，将修改完毕后的变量赋值给原来的数组。获取数据流（温度）的函数代码如下：

getTe:function(){

var that = this

wx.request({

url: 'http://api.heclouds.com/devices/' + that.data.equipid + "/datapoints?datastream\_id=Temperature",

header: {

'content-type': 'application/x-www-form-urlencoded',

"api-key": that.data.apikey

},

data: {

},

success(res) {

if (res.data.data.count == 1) {

var te = res.data.data.datastreams[0].datapoints[0].value

var temp = that.data.name

temp[0].data = te

that.setData({

name: temp

})

}

}

})

}

注意逗号的使用，温度、LED、蜂鸣器等的数据获取同理，不再累述。在获取加速度数据时，可以使用英文逗号分割数据流ID从而实现批量获取数据流，具体数据如何获取建议使用console.log(res)打印到控制台查看。

##### 5.4.1.4 数据报警

此处实现逻辑是，如果当前值超过报警阈值，则将对应的边框设置为红色，并且在下方添加小红点，起到提示作用。此处仅完成了温度部分，代码如下：

notice:function(){

var tempMax = 16

if(this.data.name[0].data>tempMax){

this.setData({

color1:'#ff0000'

})

wx.setTabBarBadge({

index: 0,

text: '!'

})

}

else{

this.setData({

color1: '#aaaaff'

})

wx.setTabBarBadge({

index: 0,

text: ''

})

}

}

其中，wx.setTabBarBadge()实现的是，为tabBar的第index个对象添加内容为text的小红点，由于小红点尺寸问题，不建议添加过多字符。

##### 5.4.1.5 实现数据的不断刷新

在实际的运用中，我们希望数据能一直得到刷新，因此需要一直更新相应的变量。因此，使用了定时器的方法。

首先在data中定义定时器setInter:’’，然后在onshow函数（即页面被展示）中持续使用setInterval函数实现循环操作。onshow完整代码如下所示：

onShow: function () {

var that = this;

that.data.setInter = setInterval(function () {

that.onGetData()

that.notice()

}, 3000)

}

表示，每隔3000ms执行一次onGetData函数和notice函数。

当我们卸载当前页面（onunload）或隐藏（onhide）当前页面（例如：暂时回到桌面）时，使用clearInterval(this.data.setInter)清空定时器。关于页面生命周期的知识您可以参阅微信官方开发文档。

至此，我们完成了data页的全部功能。对于数据详情页面，只需将5.4.1.3中的url参数按照需要修改，重新根据返回的数据修改数据，并通过定时器循环执行更新操作即可。此处列出温度数据详情页的函数代码。

// pages/te\_data/te\_data.js

var app = getApp()//实现通过app.[attrName]的方式获取定义在app.js中的程序的全局变量

Page({

/\*\*

\* 页面的初始数据

\*/

data: {

setInter: '',

dataList:[

{

time:'null',

data:'null',

unit:'null'

}

]

},

/\*\*

\* 生命周期函数--监听页面加载

\*/

onLoad: function (options) {

},

onGetData:function(){

var that = this

wx.request({

url: 'http://api.heclouds.com/devices/' + app.equipid + "/datapoints?datastream\_id=Temperature&limit=10",

header: {

'content-type': 'application/x-www-form-urlencoded',

"api-key": app.apikey

},

data: {

},

success(res) {

var temp = [

{

time:'none',

data:'none',

unit:'none'

}

]

if (res.data.data.count != 0) {

var n = res.data.data.count

var i

for(i=0;i<n;i++){

var at = res.data.data.datastreams[0].datapoints[i].at

var attemp = String(at).split(' ')

var timetemp = attemp[1].split('.')

at = timetemp[0]

var te = res.data.data.datastreams[0].datapoints[i].value

var input = {

time:at,

data:te,

unit:'℃'

}

temp[i] = input

}

that.setData({

dataList:temp

})

}

}

})

},

/\*\*

\* 生命周期函数--监听页面显示

\*/

onShow: function () {

var that = this;

that.data.setInter = setInterval(function () {

that.onGetData()

// that.getStatus1()

}, 3000)

},

/\*\*

\* 生命周期函数--监听页面隐藏

\*/

onHide: function () {

clearInterval(this.data.setInter)

},

/\*\*

\* 生命周期函数--监听页面卸载

\*/

onUnload: function () {

clearInterval(this.data.setInter)

},

})

#### 5.4.2 button页

##### 5.4.2.1 修改定时器日期changDate

在wxml文件中，我为修改日期的picker绑定了一个changeDate事件，因此在js文件中添加changeData:function(event)，注意，改变的日期是通过event.detail.value的形式得到的，因此括号内必须添加event，使用setData将得到的日期值赋给当前展示的日期（cur\_date）。注意，很容易想到，当日期在今日时，修改时间的范围应该是当前时间到23：59，因此还需要做一个判断，如果当前日期在今天以后，就需要将修改时间的范围修改为00：00-23：59。

changeDate:function(event){

var date = event.detail.value

this.setData({

cur\_date:date

})

if(this.data.cur\_date>this.data.today){

this.setData({

start\_time:'00:00'

})

}

else{

this.setData({

start\_time: this.data.timenow

})

}

}

##### 5.4.2.2 修改定时器时间changeTime

与修改日期的实现相同，通过event.detail.value的形式得到时间，赋值给当前展示的时间即可。

changeTime:function(event){

var time = event.detail.value

this.setData({

cur\_time:time

})

}

##### 5.4.2.3 设置定时器操作pickerchange

该函数的功能为，决定是定时打开还是定时关闭，使用switch组件完成，因此获取value，并赋值给对应的变量

pickerchange:function(event){

var status=event.detail.value

this.setData({

pickerctl:status

})

}

##### 5.4.2.4 获取定时器按钮点击状态（LED、Beep）redbtn等

该函数实际上是按钮的单击事件，判断当前按钮是否已经被点击过，如果没有被点击过，则点击后，将颜色设定为点击后的颜色，并且将状态设定为选中，反之同理。

redbtn:function(event){

var uncheck = '#ffffff'

var colornow = this.data.rbtncolor

if(colornow == uncheck){

var clr = '#eeffee'

this.setData({

rbtncolor: clr,

rbtnstatus:true

})

}

else{

var clr = '#ffffff'

this.setData({

rbtncolor: clr,

rbtnstatus: false

})

}

}

##### 5.4.2.5 操作函数operate和operateAll

实际上operateAll的实现是通过遍历输出设备（LED、Beep）列表和各自的状态，获取当前操作的信息（开启/关闭），编辑对应的命令，并逐个调用operate函数实现。操作函数operate的实现是将传递的command使用POST方法传递到onenet平台，onenet平台会将对应的命令下发至开发板。

operateAll:function(id){

var i = 0

for(i=0;i<this.data.equiplist.length;i++){

var command = this.data.equiplist[i]+id

this.operate(command)

this.sleep(500)

// console.log(command)

}

},

operate: function (comm) {

wx.request({

url: 'http://api.heclouds.com/cmds?device\_id='+this.data.equipid,

method: 'POST',

header: {

'content-type': 'application/x-www-form-urlencoded',

"api-key": this.data.apikey

},

data: comm,

success(res) {

// console.log(res)

}

})

}

##### 5.4.2.6 自定义延时函数sleep

由于微信的js中不支持sleep、delay等常用的延时函数，因此自定义sleep函数，通过起始时间和当前时间的方式实现自定义延时。

sleep: function (numberMillis){

var now = new Date();

var exitTime = now.getTime() + numberMillis;

while (true) {

now = new Date();

if (now.getTime() > exitTime)

return;

}

}

##### 5.4.2.7 一键开关状态的检查 allcheck

获取所有输出设备的状态，如果全为打开状态则将一键开关的switch组件通过相应的状态变量设定为切换实现一键关闭全部的状态，反之同理。

allcheck:function(){

if (this.redledstatus && this.yellowledstatus && this.blueledstatus && this.greenledstatus && this.beepstatus){

this.setData({

allstatus:true

})

}

}

##### 5.4.2.8即时的一键开关 allchange

根据选定的状态，选择打开或关闭，调用operateAll函数实现一键打开/关闭全部设备

allchange:function(event){

var status = event.detail.value

this.setData({

redledstatus: status,

yellowledstatus: status,

greenledstatus: status,

blueledstatus: status,

beepstatus: status,

allstatus:status

})

if(status){

this.operateAll('1')

}

else{

this.operateAll('0')

}

}

##### 5.4.2.9 即时的单个输出设备的开关（LED、Beep） redledchange

获取switch组件的状态（event.detail.value），根据状态选择命令，调用operate函数实现，调用完毕后调用一键开关状态检查allcheck，刷新一键开关状态

redledchange:function(event){

var status = event.detail.value

this.setData({

redledstatus:status

})

if (this.data.redledstatus){

var command = '{redled}:1'

this.operate(command)

}

else{

var command = '{redled}:0'

this.operate(command)

}

this.allcheck()

}

##### 5.4.2.10 获取系统时间和刷新当前时间 gettime renew\_time

使用时间戳的方式获取当前时间（系统时间），具体实现参见代码。注意的是，getMonth的以数组下标的形式返回，其取值范围为0-11而非1-12，所以需要加1处理；另外，注意修改日期和修改时间的操作返回的日期格式，需要使用a?b:c（if a then b else c）的形式格式化日期

刷新当前时间：通过不断调用获取系统时间的方式刷新当前的时间

gettime:function(event){

var timestamp = Date.parse(new Date());

var date = new Date(timestamp)

var year = date.getFullYear()

var mouth = (date.getMonth() + 1) < 10 ? '0' + (date.getMonth() + 1) : (date.getMonth() + 1)

var day = date.getDate() < 10 ? '0'+ date.getDate(): date.getDate()

var hour = date.getHours() < 10 ? '0' + date.getHours() : date.getHours()

var minute = date.getMinutes() < 10 ? '0' + date.getMinutes() : date.getMinutes()

var datenow = year +'-'+ mouth +'-'+ day

var timenow = hour+':'+minute

var temp = [datenow,timenow]

return temp

},

renew\_time:function(){

var temp = this.gettime()

var d = temp[0]

var t = temp[1]

// console.log(t, d)

this.setData({

today: d,

timenow: t

})

}

##### 5.4.2.11 提交和撤销定时操作 submit undo

提交操作首先获取按钮选择清空和开关switch组件的属性，获取定时器的时间，设置对应的变量，并通过设置flag的形式打开定时器开关。反向操作通过undo实现，具体实现参看代码。

submit:function(){

var cmd\_list = [this.data.rbtnstatus,this.data.ybtnstatus,this.data.bbtnstatus,this.data.gbtnstatus,this.data.beepbtnstatus]

this.setData({

opt\_time:this.data.cur\_time,

opt\_date:this.data.cur\_date,

picklist:cmd\_list,

pickflag:true

})

if(this.data.pickerctl){

this.setData({

pickcmd:'1'

})

}

else{

this.setData({

pickcmd: '0'

})

}

}

##### 5.4.2.12 定时开关实现 timecheck

在提交操作时，已经将预约时间修改，则首先需要判断当前时间是否为预约时间。如果是，再确定是否已经提交，如果是，则根据提交的参数，发送相应的指令到平台。全部发送完成后，为了避免重复发送指令，应当将提交flag设置为false。具体实现参看代码。

timecheck:function(){

// console.log('checking', this.data.opt\_time, this.data.opt\_date, this.data.today)

var i

var len

var flag1 = (this.data.opt\_date == this.data.today)

var flag2 = (this.data.opt\_time == this.data.timenow)

// console.log(flag1,flag2)

if(flag1 && flag2 && this.data.pickflag){

for (i = 0, len = this.data.picklist.length;i<len;i++){

if(this.data.picklist[i]){

var command = this.data.equiplist[i]+this.data.pickcmd

this.sleep(1000)

this.operate(command)

this.setData({

pickflag:false

})

}

}

this.undo()

}

}

##### 5.4.2.13 数据更新 onGetData

实现原理同data页面，定时刷新即可，此处不再贴出代码

##### 5.4.2.14 页面的生命周期

同data页，此处主要变动的是onshow

分析页面，展示页面后，首先应该展示当前的时间，因此首先应该调用gettime函数更新定时器展示的时间。之后，应该借助定时器实现定时更新的功能。在定时更新中，显然数据流应当及时得到更新，系统当前的时间也应得到更新。另外，此处应该实现持续检查能否执行定时开关，因此，定时开关实现timecheck也被写进定期执行的语句内。

onShow: function(){

var temp = this.gettime()

var d = temp[0]

var t = temp[1]

var statusTemp

this.setData({

cur\_date:d,

cur\_time:t,

start\_time:t

})

var that = this;

that.data.setInter = setInterval(function () {

that.renew\_time()

that.timecheck()

that.onGetData()

}, 1000)

}

至此，全部函数编写完毕。此时可以点击不同的组件测试，如果没有反应的，确认连接无误的情况下，可以进入对应的函数下使用log调试。您也可以在我的github页面下载我已经测试通过的项目，将project.config.json文件中的appid、projectname等信息换为自己的信息，将app.js、data.js、button.js的equipid、apikey替换为自己的对应值即可。

## 六 项目总结

项目完成的第一时间就着手写下了本文档，总结之后本项目还是存在一定的需要改进的地方。现在列部分我想到的如下。

### 6.1 跨页面的提示功能

目前的数据警报功能只能在data页面内实现，在实际运用中，应该能做到在任意页面都能收到预警消息。解决方案是，使用额外的js文件专用于从onenet平台获取数据，再在每个页面对应调用，这样也能实现数据的持续更新。

### 6.2 响应时间问题

目前的数据由于更新时间并非瞬时，而是间隔一定时间获取一次数据，因此在控制页使用开关、一键开关时，会出现：打开开关后，开关自动关闭，过一段时间后又自动开启的现象。解决方案是，优化代码，在bindchange事件中更新status值，并根据实际应用场景修改。

### 6.3 数据展示只能固定展示最新十条数据

由于本项目实际上没有服务器，因此所有的数据都不会被保存，如果需要查看以前的数据，除了在查询页面设置对应的url参数以外，还可以通过第三方服务器的方式存储数据和读取。

### 6.4 数据展示不够直观

建议使用折线图等方式展示数据，推荐使用百度echarts组件。

### 6.5 待补充

## 七 感言

通过本次学习，更加熟悉了OneNet平台、STM32F10x系列开发板、Keil软件和微信小程序的操作和使用，感谢四川大学杨彦兵老师在本人学习期间不遗余力的指导，感谢中国移动OneNet交流群内技术人员的帮助。感谢互联网作者提供的部分教程。

## 八 参考资料

* 中移OneNet平台开发者文档
* 微信官方文档 小程序部分
* https://blog.csdn.net/weixin\_45006076/article/details/102762293微信小程序连接onenet控制RGB三色灯
* https://www.cnblogs.com/zl0372/p/11011304.html python做上位机
* https://www.cnblogs.com/i-douya/p/8807454.html 微信小程序 设置计时器（setInterval）、清除计时器（clearInterval）
* https://www.jb51.net/article/134097.htm 微信小程序switch开关选择器使用详解
* （另有部分参考资料没有列出）