

ESP8266 使用 EDP 协议连接 ONENET

作者：陆树汉，陆泽宏，黄献游

指导老师：田敬北，谭启锦

疑惑 QQ：3045136580

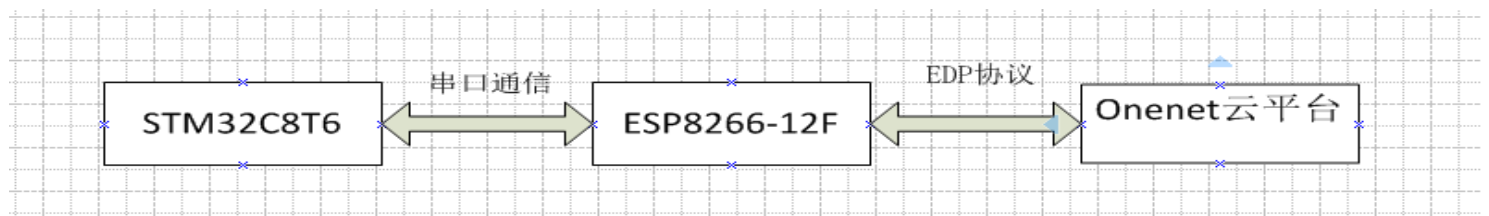
更多资源获取，关注公众号：luxiaoguogege

完整工程源码：<https://www.cnblogs.com/luxiaoguogege/p/10136996.html>

视频教程：<https://v.qq.com/x/page/i0814q78no3.html>



本次教程是使用 STM32C8T6 通过 ESP8266-12F 模块将数据传输到 ONENET 云端去，并且云端能够下发命令给单片机来实现云端控制。本次实验硬件设备：STM32C8T6 最小系统，ESP8266-12F 模块，wifi。下面是简单的设备传输结构图：



实验中只是将定义的数据进行上传，具体的可以自己添加外部数据采集模块来实现数据的上传及控制。本次实验中查看到的资料网址：

OneNET -中国移动物联网开放平台：<https://open.iot.10086.cn/>

基于 WIFI 方式连接 OneNET (ESP8266)：<https://open.iot.10086.cn/doc/art441.html#109>

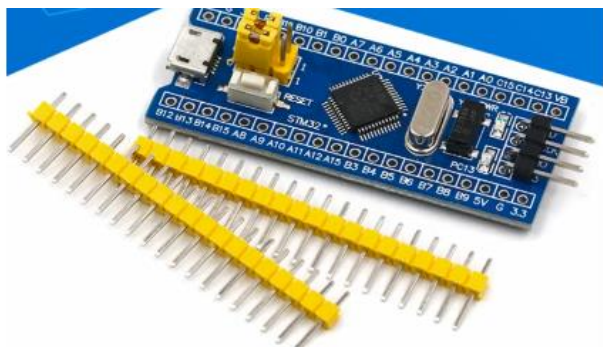
麒麟开发板代码、资料：<https://open.iot.10086.cn/bbs/thread-863-1-1.html>

机智云开发板使用方法：<https://pan.baidu.com/s/1AQIpM4mJ5EimHEYCDM9McQ>

友情提示，在使用麒麟开发板程序时候不知道是他们工程有问题还是软件的问题，在编译下载测试时候工程波特率不对，会出现乱码现象，可以自己新建一个工程然后移植里面的子程序。闲话少说，开始进行教程。

(一) ESP8266 固件烧写

1. 首先去淘宝购买 STM32C8T6 的最小系统（其他的 STM32 也行）和 ESP8266-12F 模块

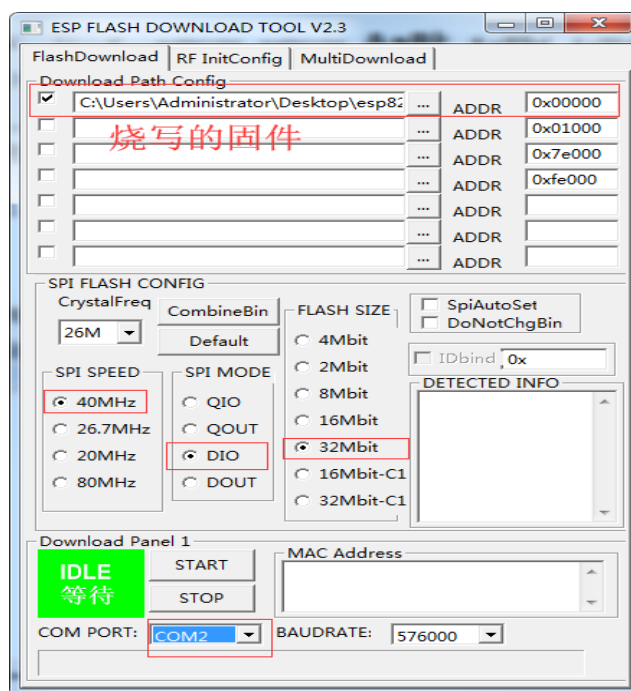
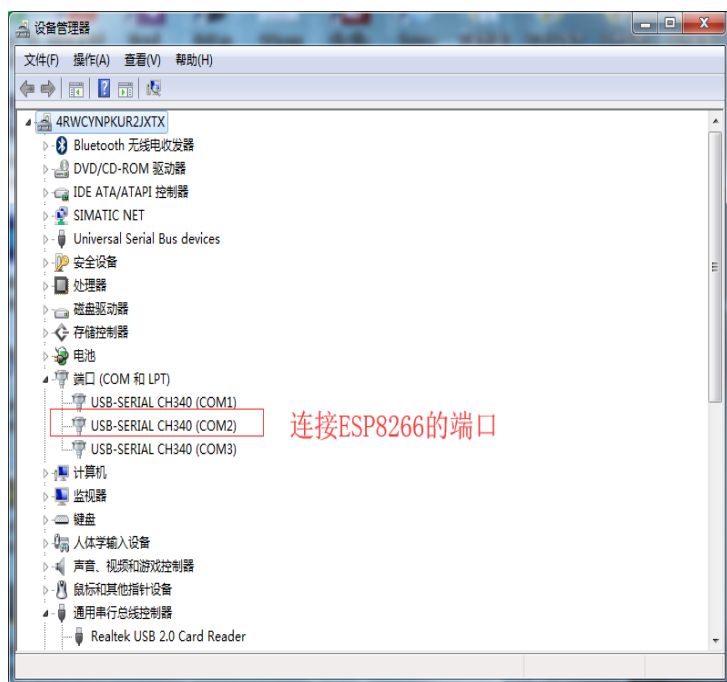


2. 用 USB 线接 ESP8266-12F 模块，接入电脑进行固件的烧写

3. 步骤一：烧写 AT 固件，烧写指导文件目录下的 **v1.3.0.2 AT Firmware**，使用工具 **ESP_DOWNLOAD_TOOL_V2.4** 烧写，

烧写设置如下：

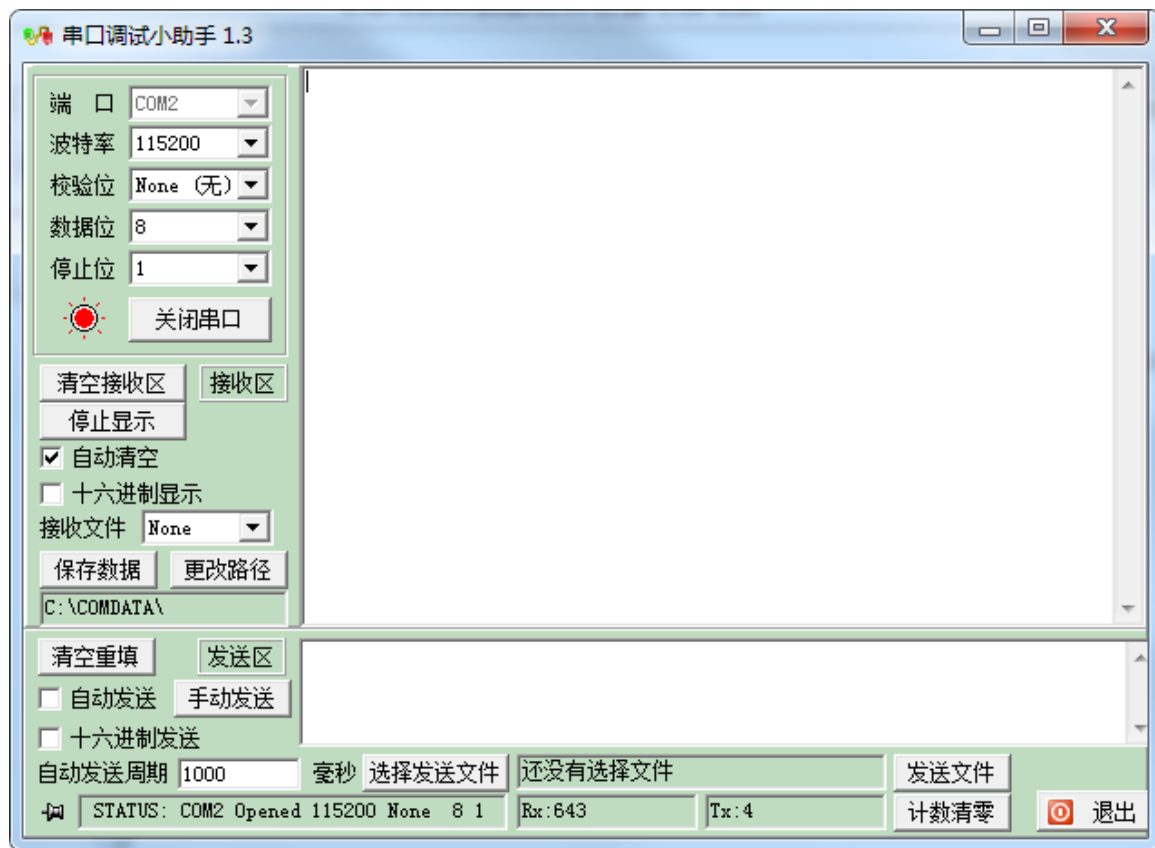
COM PORT 可以通过设备管理器查询，如图：(我的电脑->右键->设备管理器)



4. 我们首先把 GPIO0 接地，然后点击 start，因为 IOT 微信模块下载固件需要上电同步，所以需要 REST 脚轻触接地一下就

可以下载 AT 固件，下载完成再点击 stop。最后把 GPIO0 取消接地。(我们使用的是模块就先按 flash 按钮再按 rec 按钮，多试几次 (或者先 rec 再 flash，在或者同时按下,显示失败需要重新点击 START)，看进度条有变化就行了，下载完成点击 stop，把下载软件关掉)

5. 步骤二：打开串口工具。串口号查看方式同上。波特率设置为 115200，具体设置如下：(注意：测试时候有一些串口助手是不能接受也不能发送命令的，这里使用我提供的软件来测试，就是截图那个：ComAssistant.exe)



6. 最后打开串口，然后复位，复位方式，**REST** 轻触 GND 一下（**模块按复位键**）。如果 AT 固件下载成功会打印 ready（**不一定是这个，反正有打印信息**），如下：



7. 发送 AT 测试指令，如下，说明 AT 指令工作正常。（发什么回什么的话是由于没有加“回车换行”，加入即可。**sscom 注意勾选上“发送新行”**），发送 AT 返回 OK 证明模块能够正常使用了。遇到其他问题自行百度解决



8. 打开网站：[基于 WIFI 方式连接 OneNET \(ESP8266\)](#) 里面有说明 EDP 方式上传数据，额不看也行，最好看一下。下面测试 wifi 是否能够连接
9. 逐步执行下面的命令，一条一条的来，记得每个命令都要回车

AT+CWMODE=3 //设置 WIFI 应用模式

AT+RST //重置 WIFI 模块

AT+CIFSR//查询本地 IP

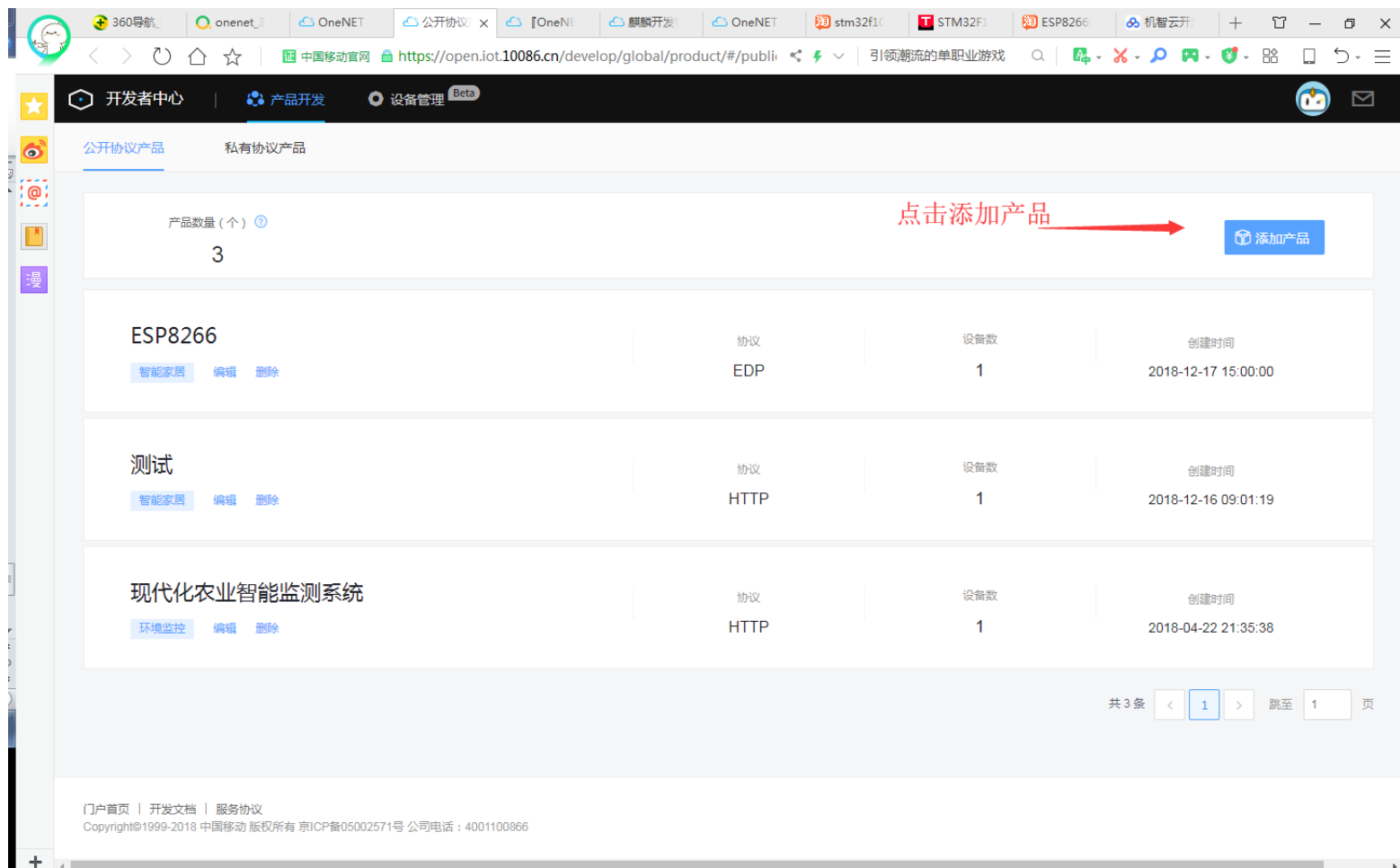
AT+CWJAP="luxiaoguogege","lsh902902"//连接路由器第一个是 wifi 名字，第二个是密码

每次执行完命令返回的都应该是 **OK** 才是正确的，连接好了之后有兴趣的可以看一下那个网址上面的，现在 esp8266 已经能够通过串口来上传数据到云端去了，具体的自己看那个协议，命令使用：[http_协议 esp8266 文档.txt](#) 那里的（这个文档的可看可不看）

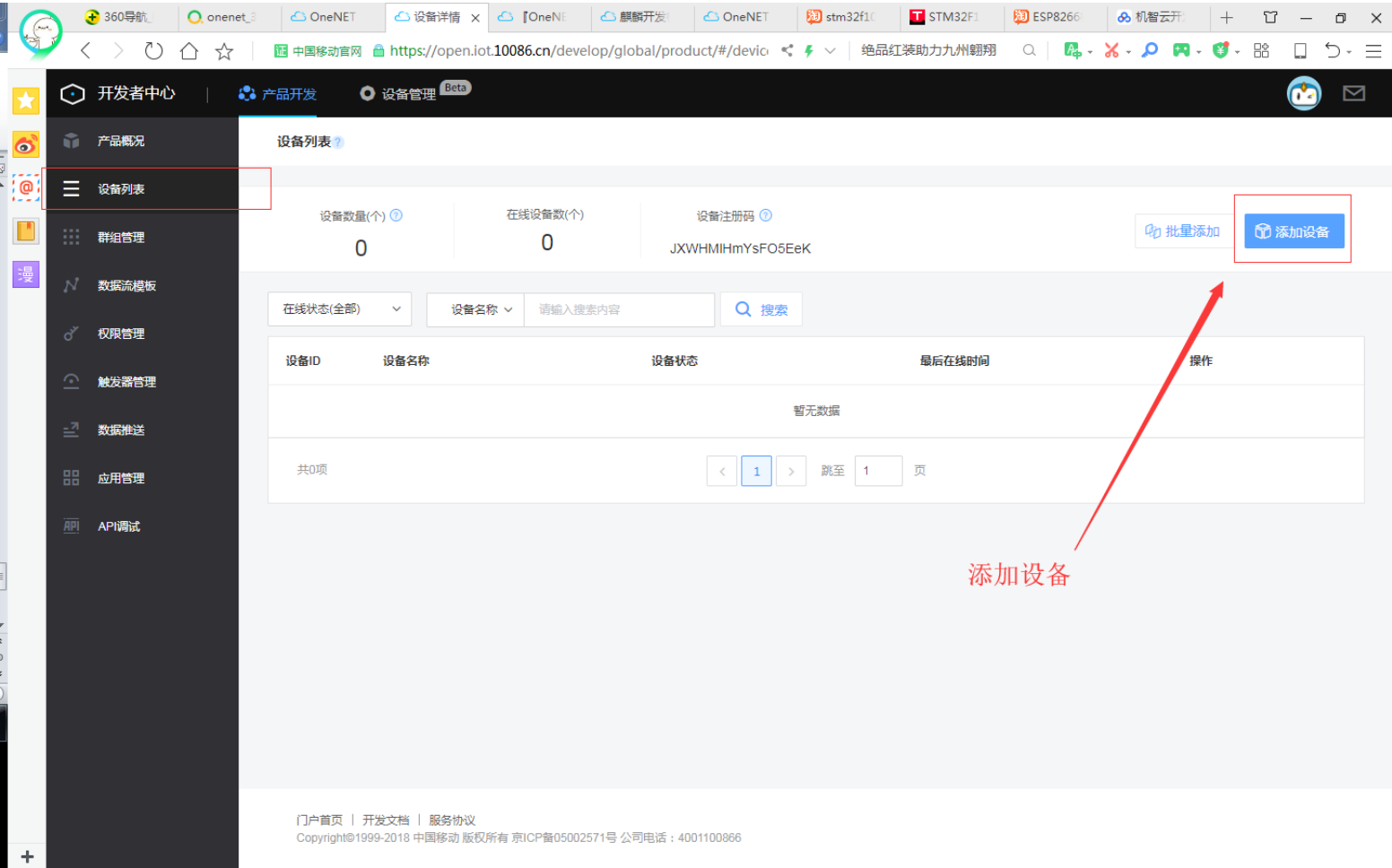
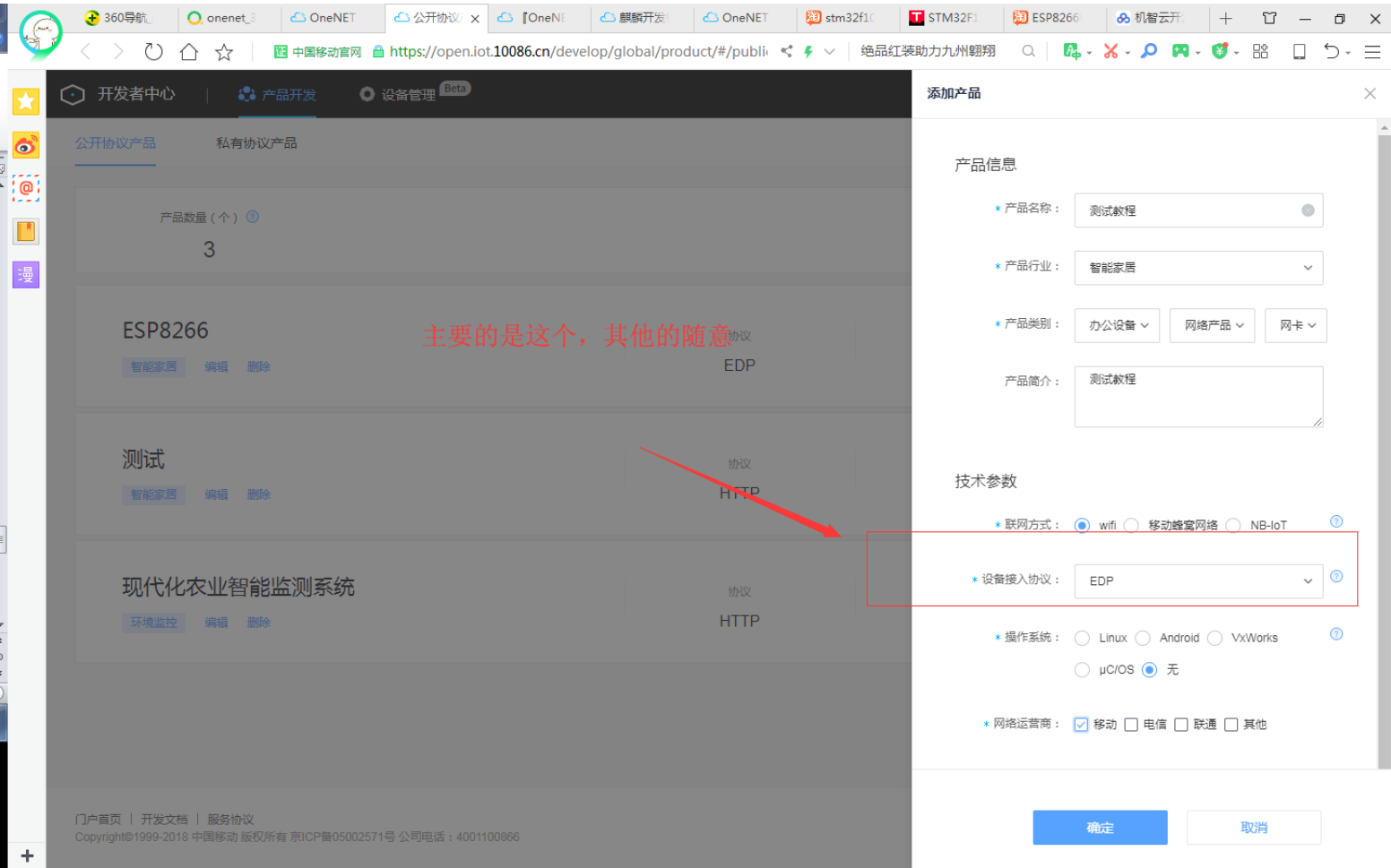


(二) 在云端创建 EDP 协议的设备 [OneNET](https://open.iot.10086.cn/) -中国移动物联网开放平台: <https://open.iot.10086.cn/>

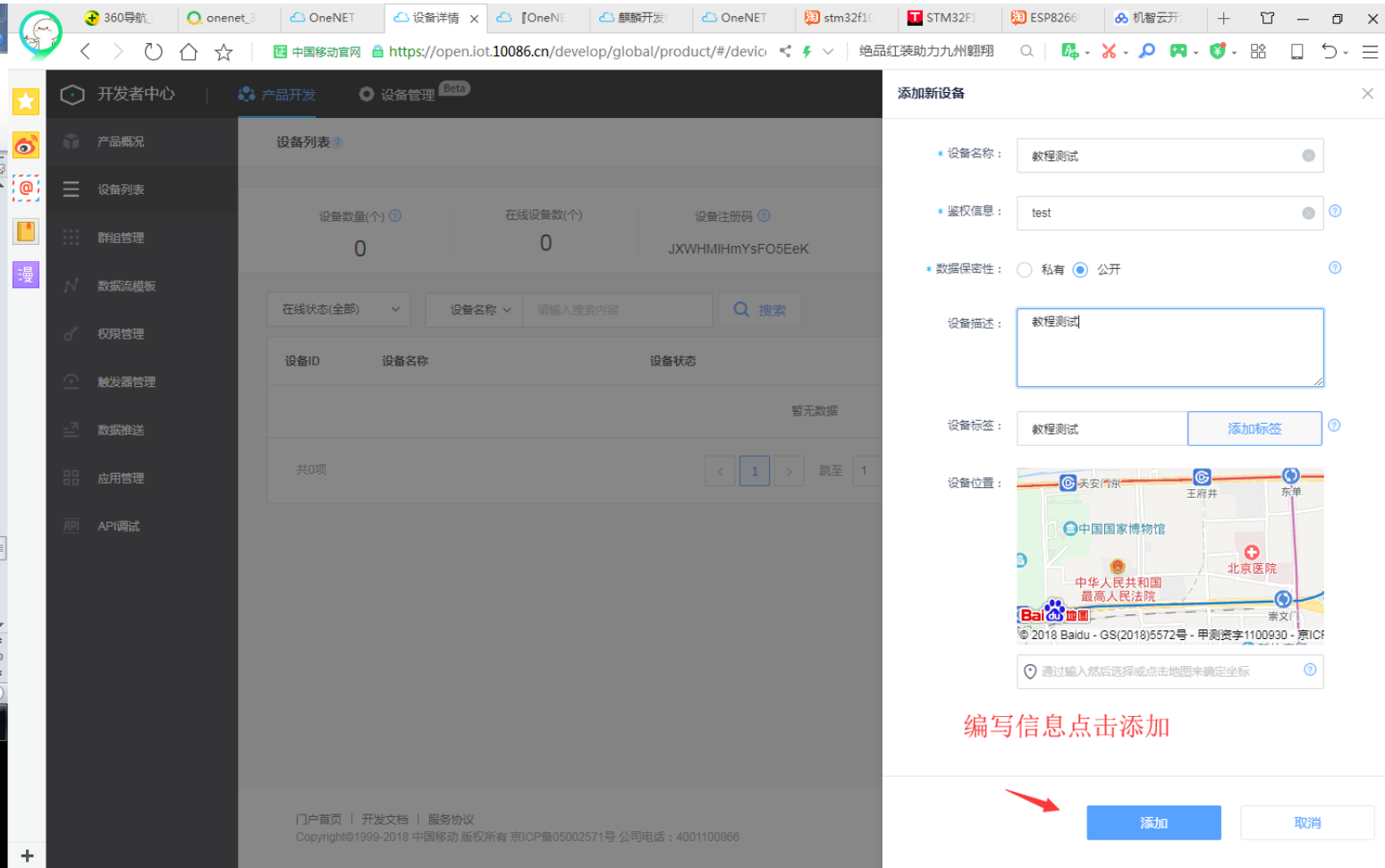
1. 点击添加产品



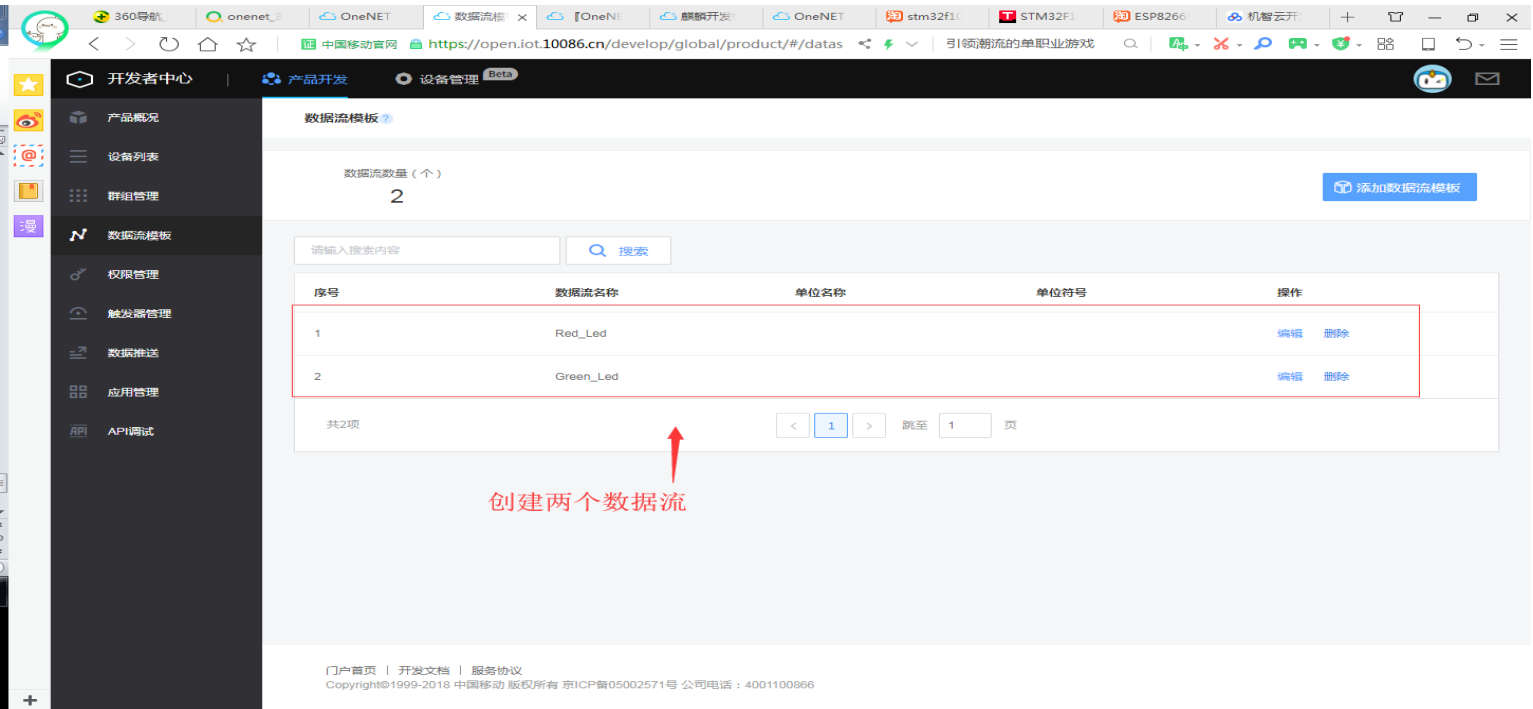
2. 编写产品的设备信息，使用的是 EDP 协议，确定之后然后点击立即添加设备



3. 编写信息点击添加



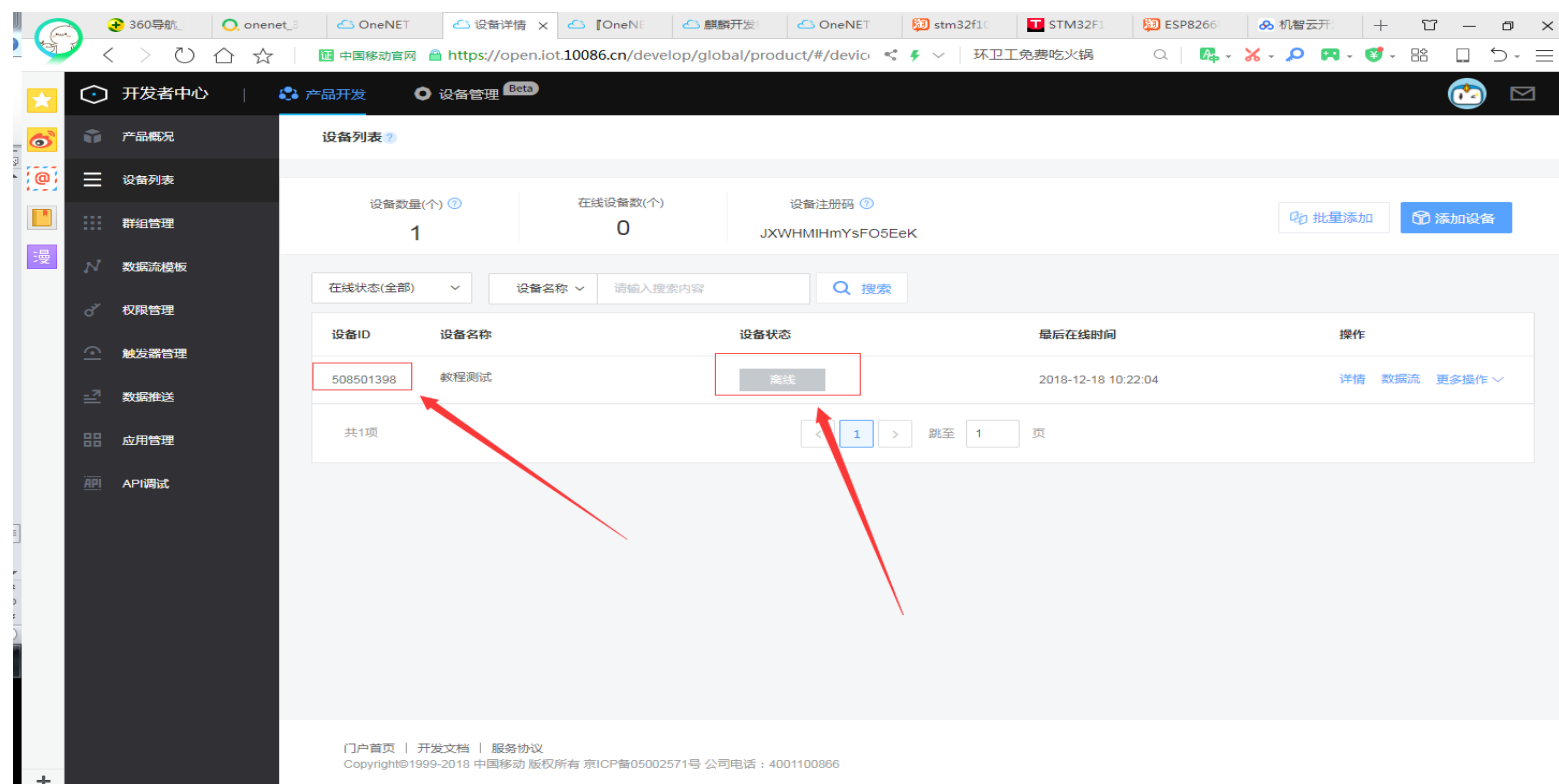
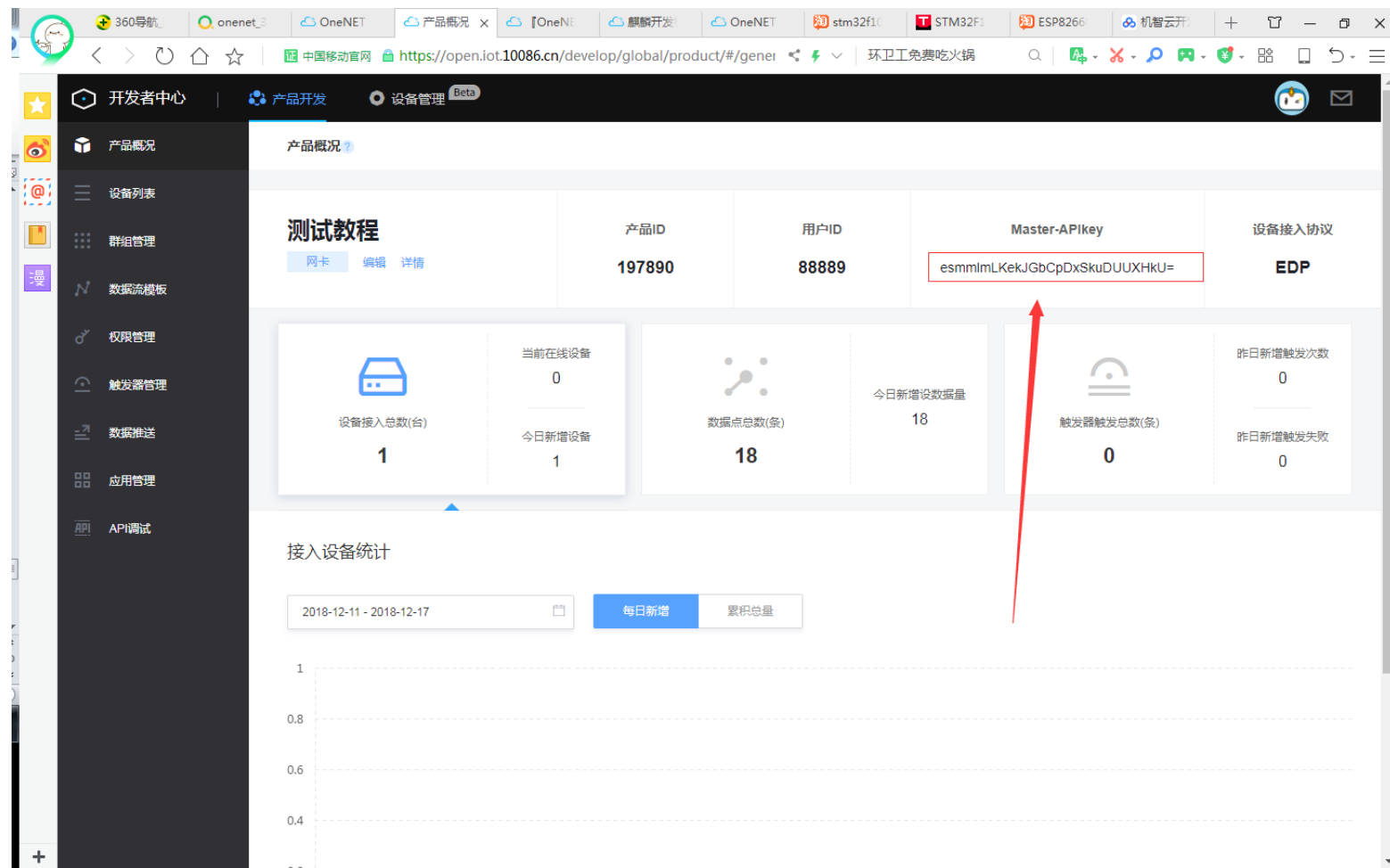
4. 在数据流模板中点击添加数据流模板，这里我们创建两个名字叫 Red_Led 和 Green_Led 名字的数据流（单位名称符号随意）。这样我们程序中上传者两个名字的数据流时候就能够上传到创建的这两个中去。如果程序中上传的数据名字在云端中没有的话，云端会自动创建程序上传上来的数据流名字。



5. 在产品概况那一页有一个 **Master-APIkey** 下面的那个复制记下来，设备列表那一页有个设备 ID 也复制下来。我这里的是。并且目前我们的设备是处于离线状态。

Master-APIkey: esmmlmLKekJGbCpDxSkuDUUXHkU=

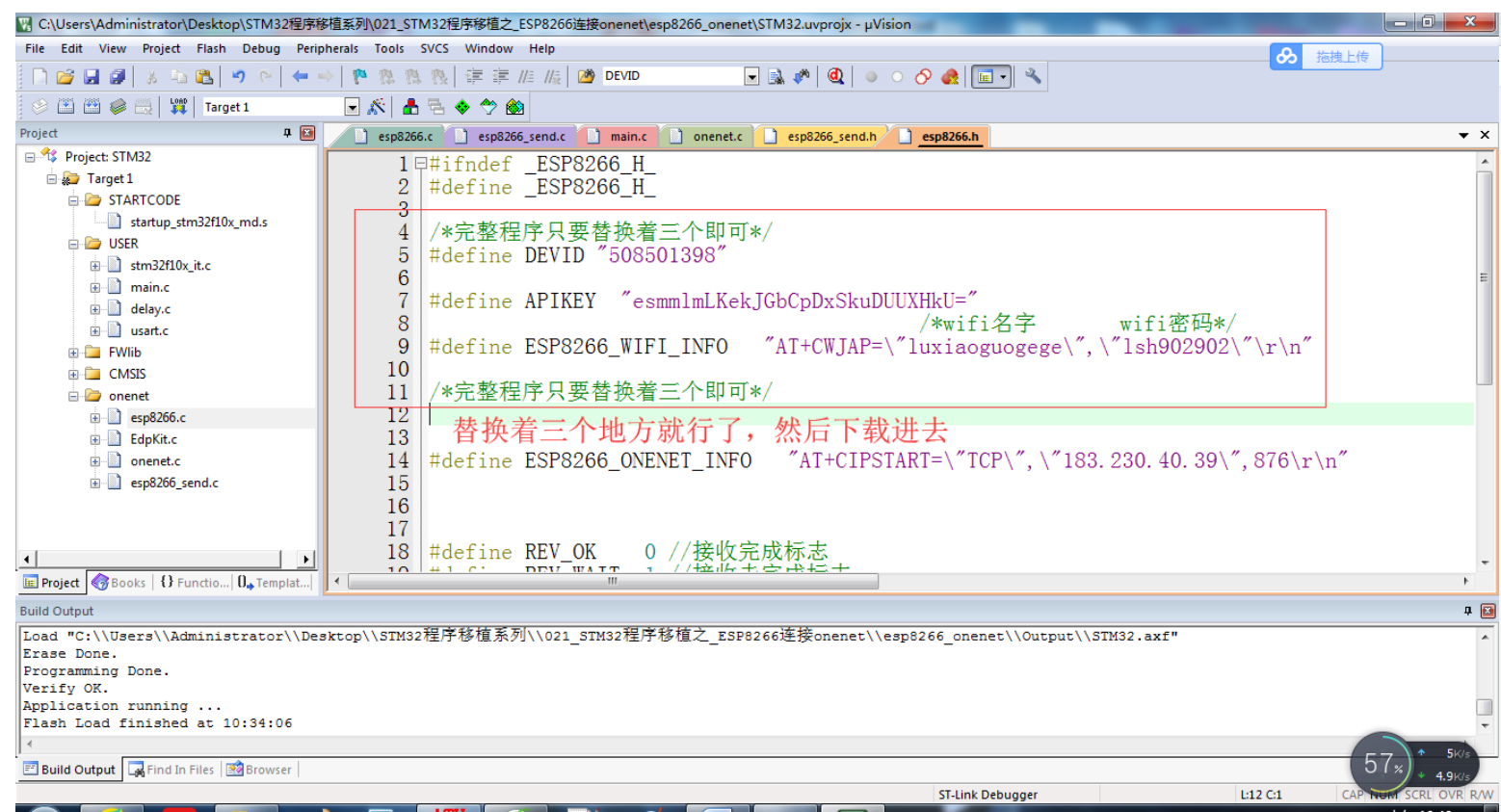
设备 ID: 508501398



6. 那么我们云端的设备先这样先，把 ESP8266-12F 底座拔出来进行连线

STM32 引脚	ESP8266-12F
5V	5V
PA3	TxD
PA2	RxD
REST	PC14
GND	GND
STM32 引脚	CH340 引脚
GND	GND
3.3V	3.3V
PA9	RXD
PA10	TXD

7. 按照上面的进行接线，然后打开 esp8266_onenet 程序 CSDN（或公众号：luxiaoguogege）
下载，在 Esp8266.h 替换三个地方就可以了。第五点那里的设备 ID 和设备密钥还有就是 wifi
名称和密码



```
1 #ifndef _ESP8266_H_
2 #define _ESP8266_H_
3
4 /*完整程序只要替换着三个即可*/
5 #define DEVID "508501398"
6
7 #define APIKEY "esmm1mLKekJGbCpDxSkuDUUXHkU="
8 /*wifi名字 wifi密码*/
9 #define ESP8266_WIFI_INFO "AT+CWJAP=\"luxiaoguogege\", \"lsh902902\"\\r\\n"
10
11 /*完整程序只要替换着三个即可*/
12
13 替换着三个地方就行了，然后下载进去
14 #define ESP8266_ONENET_INFO "AT+CIPSTART=\"TCP\", \"183.230.40.39\", 876\\r\\n"
15
16
17
18 #define REV_OK 0 //接收完成标志
19 #define REV_WAIT 1 //接收未完成标志
```

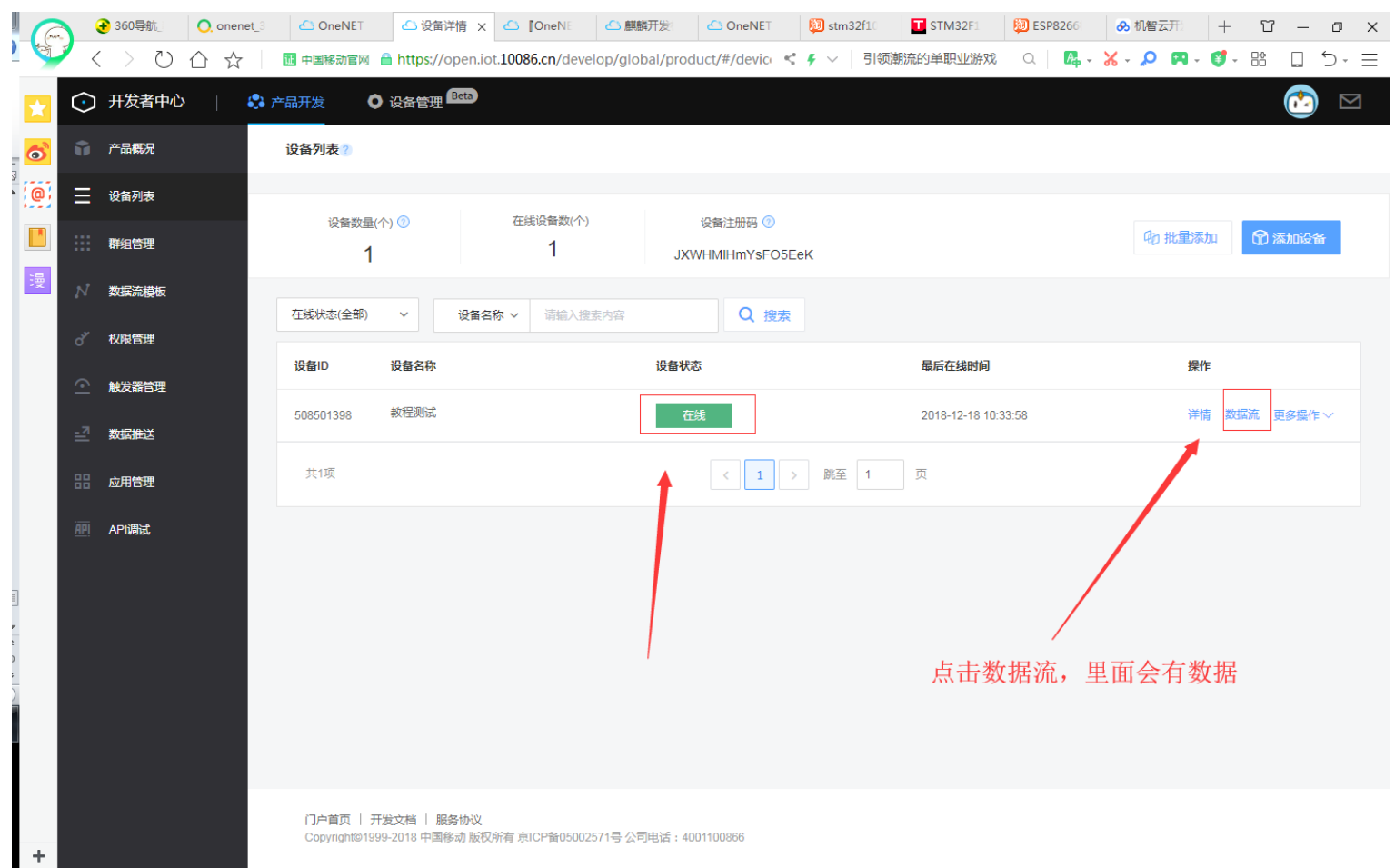
Build Output

```
Load "C:\\Users\\Administrator\\Desktop\\STM32程序移植系列\\021_STM32程序移植之_ESP8266连接onenet\\esp8266_onenet\\Output\\STM32.axf"
Erase Done.
Programming Done.
Verify OK.
Application running ...
Flash Load finished at 10:34:06
```

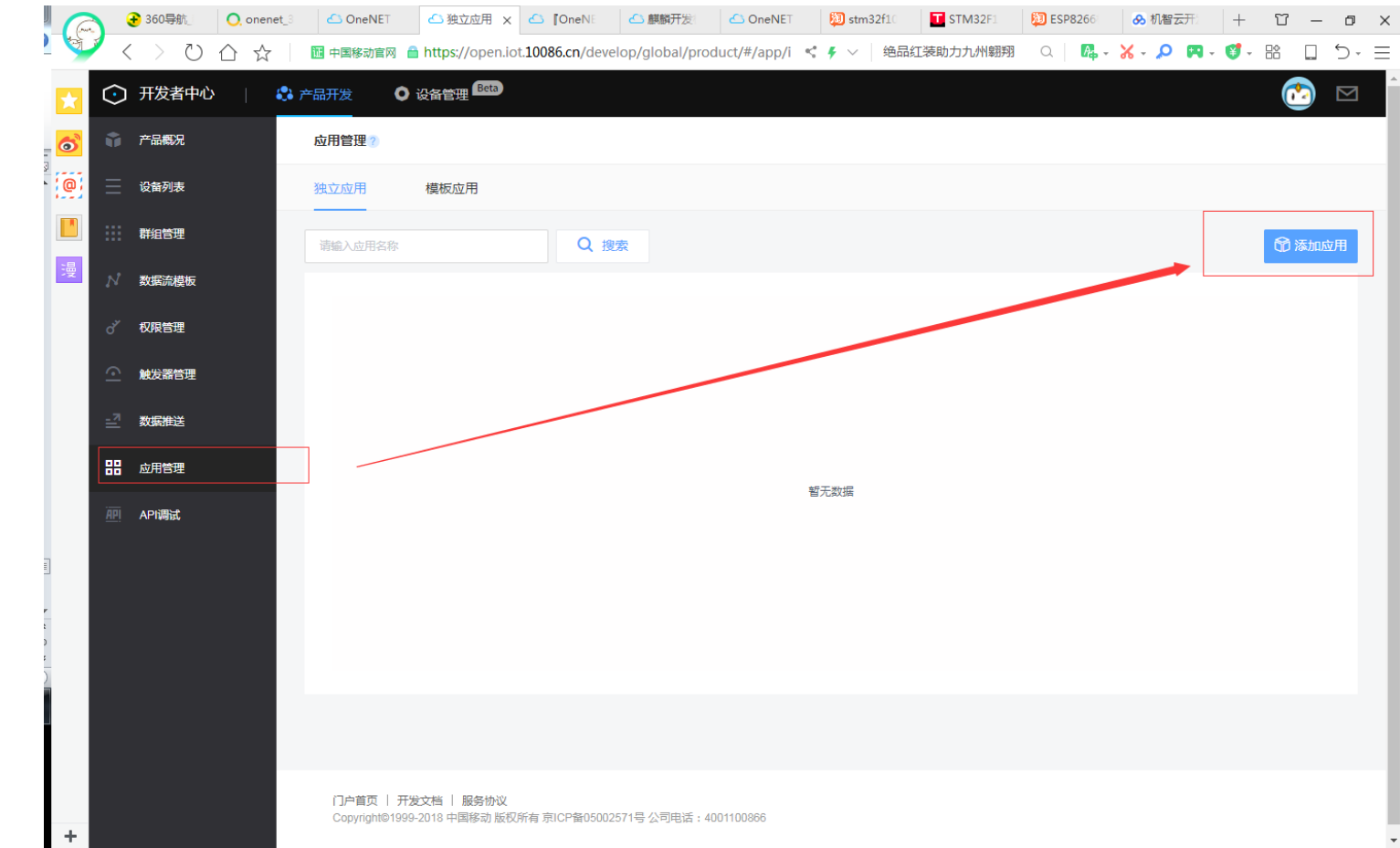
8. 我们下载进去之后通过串口助手开看程序运行中打印的信息，理论上不会有错，具体看串口助手打印什么信息来进一步分析，这里不做说明。如果运行正常的话串口助手会每隔 5S 打印一次上传数据的信息



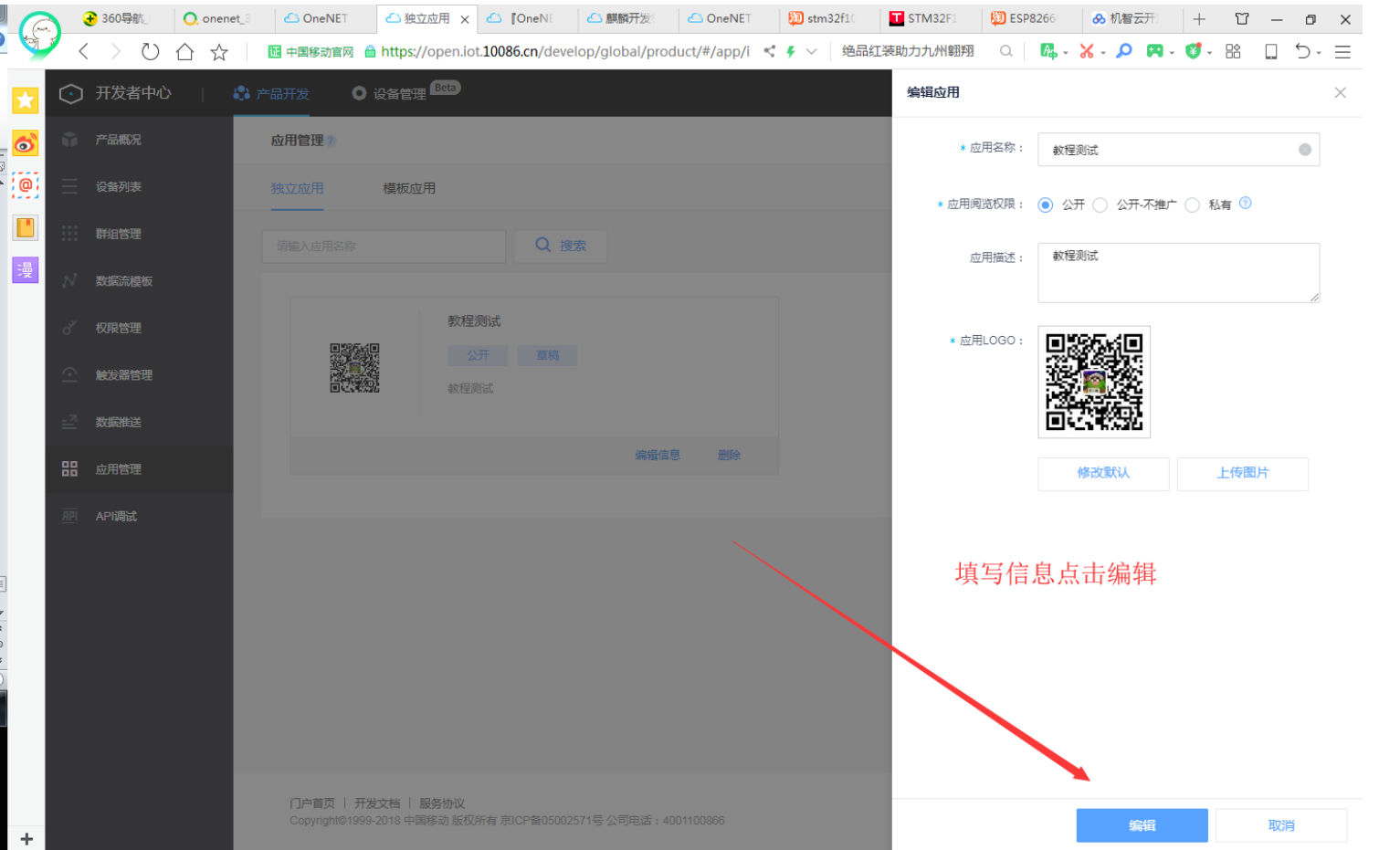
9. 假如数据上传成功之后我们看看云端那里设备列表中设备状态本来离线的就会变成在线的，并且里面的数据流有数据



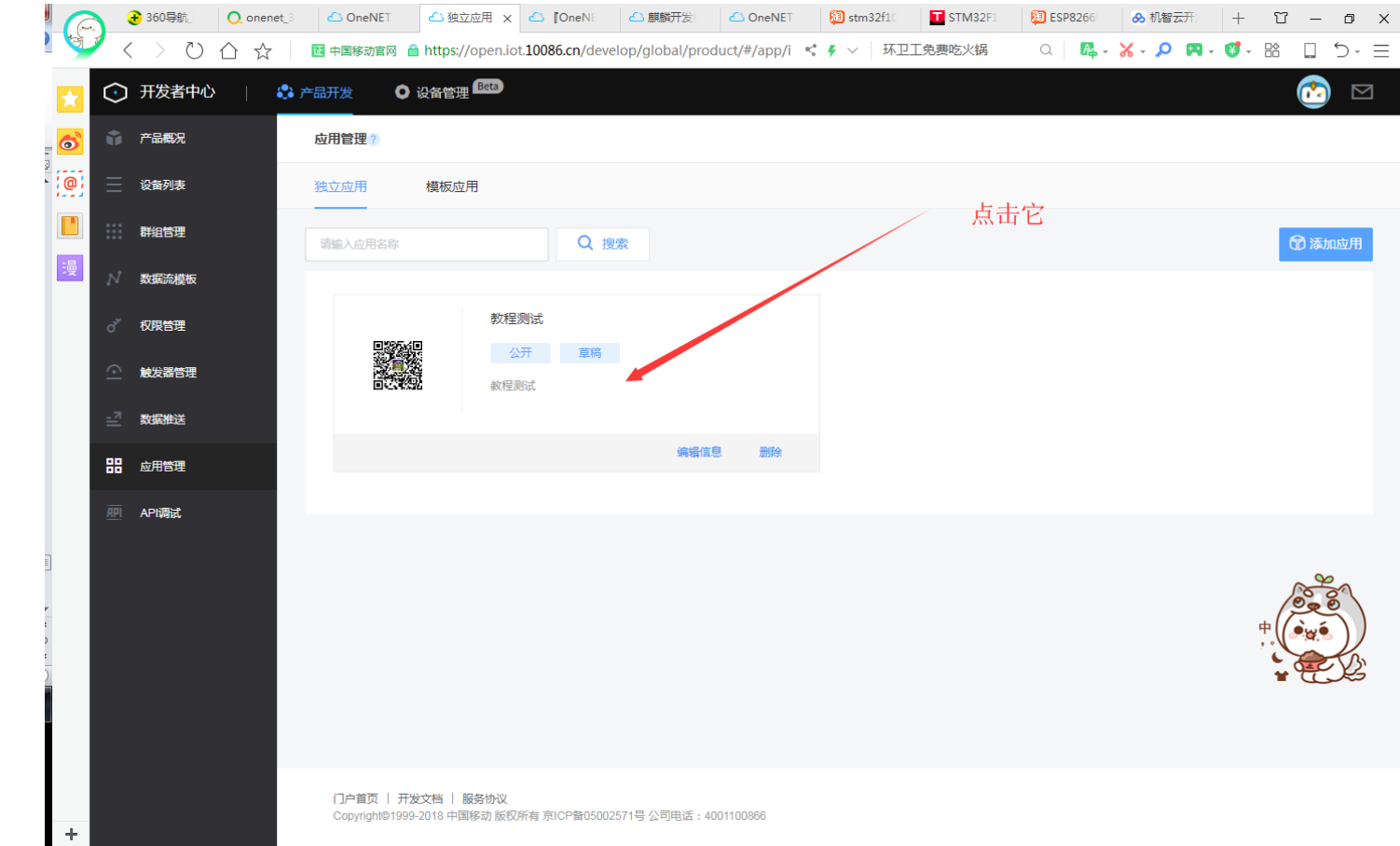
10. 那么从上面的来看我们的数据已经成功传输到 onenet 上面去了。那么就可以开始下一步了，在应用管理中添加应用



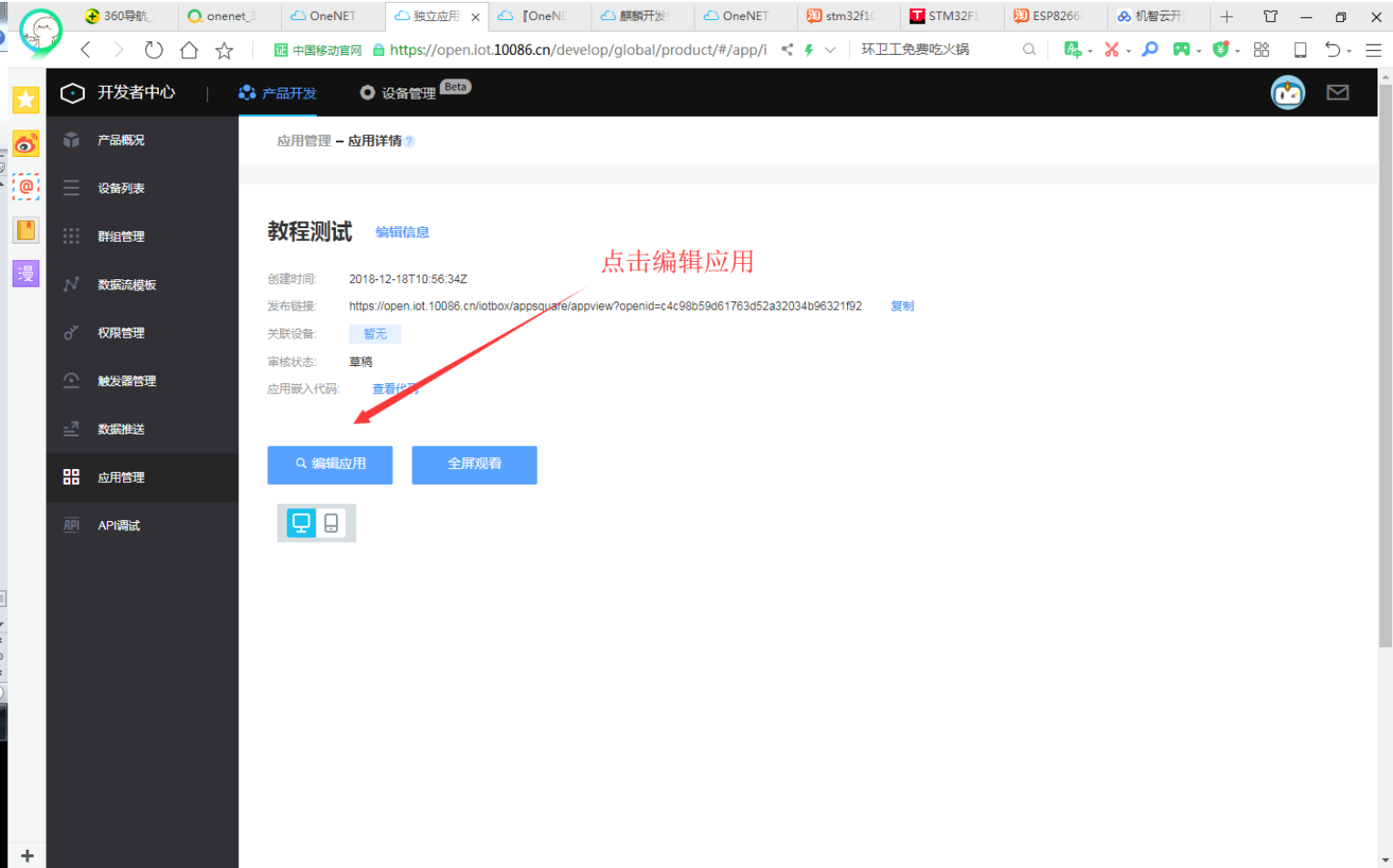
11. 编写应用相关的信息点击确定，截图那里我是重新打开所以是编辑



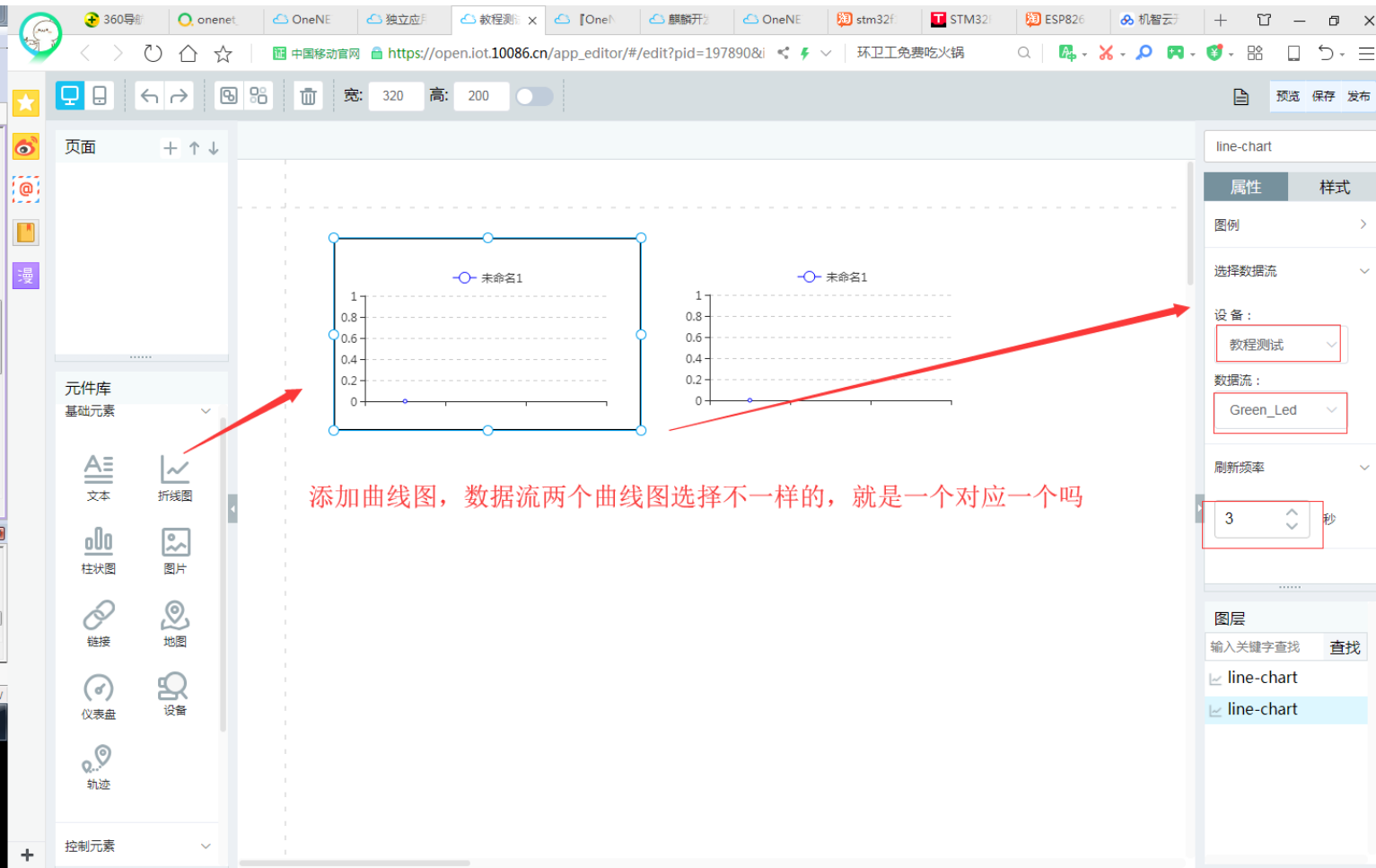
12. 点击那个应用进去



13. 点击编辑应用



14. 添加曲线图，并且修改属性，两个曲线图对应不一样的数据流



15. 添加两个按钮，并且设置开关值, 数据流也是一个对应一个按钮

第一个开关开值: greenled:1
第一个开关关值: greenled:0
第二个开关开值: redled:1
第二个开关关值: redled:0
设置好着两个按钮的值，这些开关开值和关值会发送到 esp8266 上

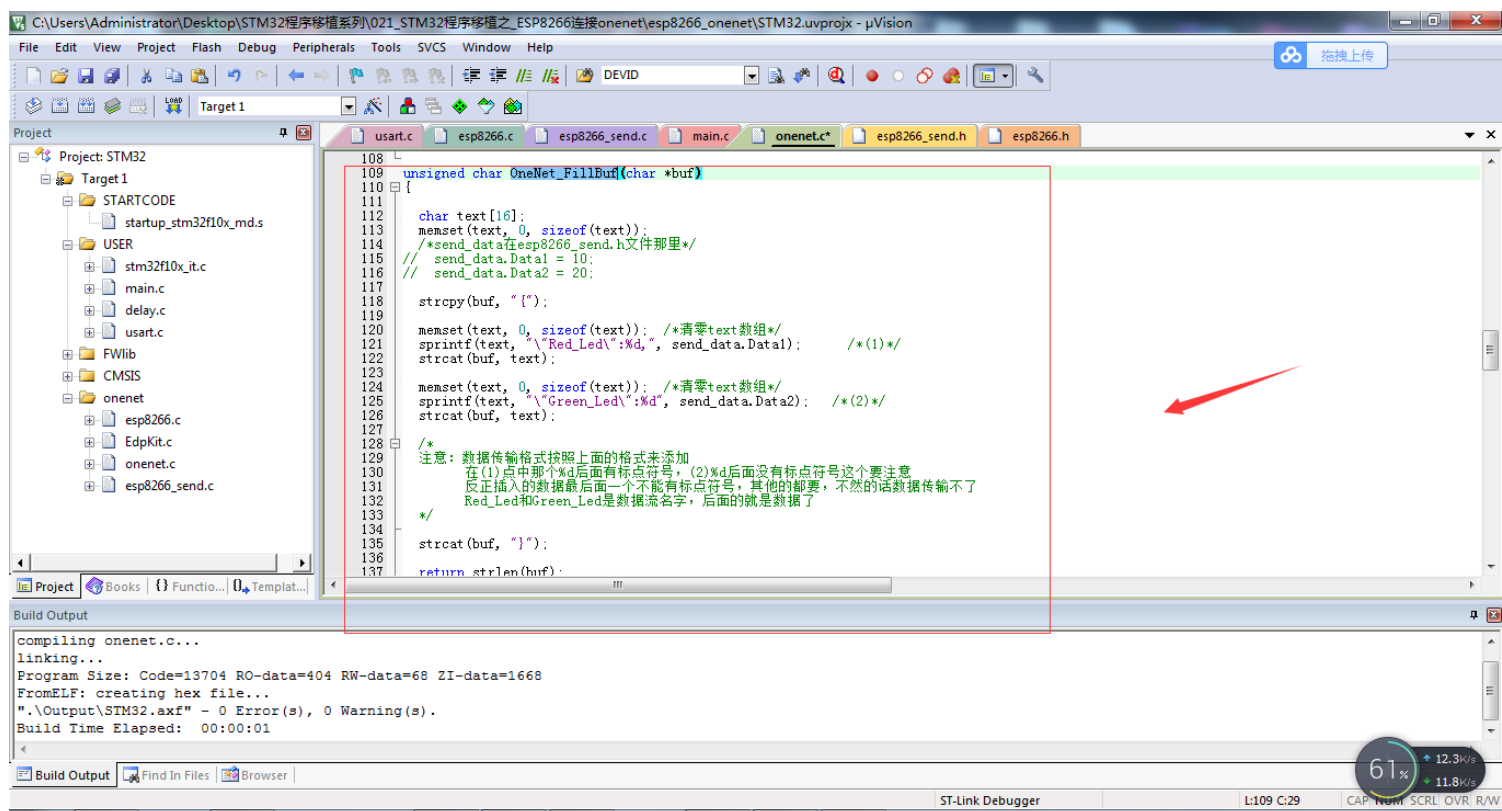


17. 那么实现上面的功能之后我们来分析程序了。这里只分析发送的子函数和接收的子函数处理，其他的自己看了，程序中都有详细注解：

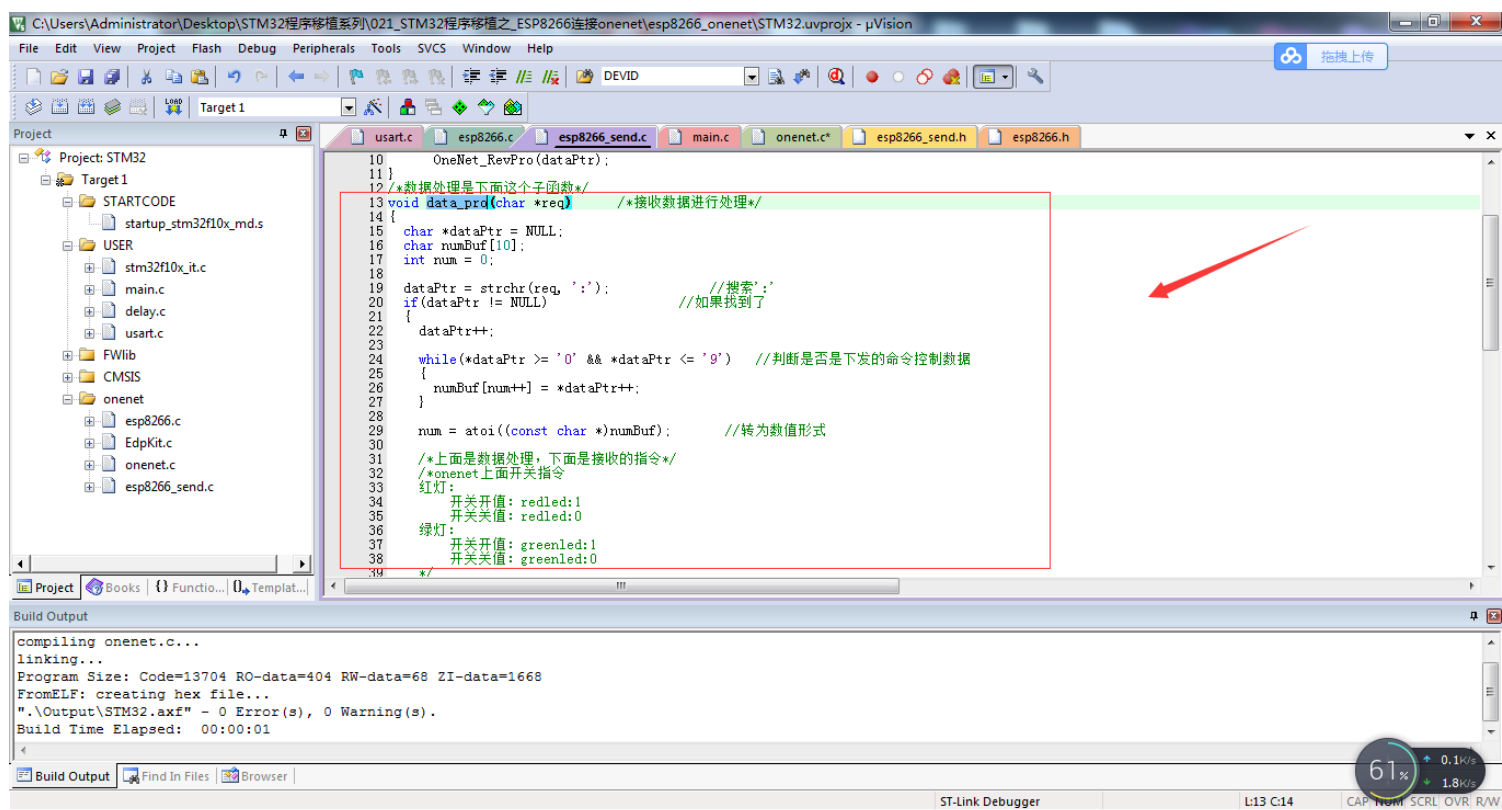
发送子函数请看 onenet.c 中 OneNet_FillBuf 子函数

接收处理子函数请看 esp8266_send.c 中 data_pro 子函数

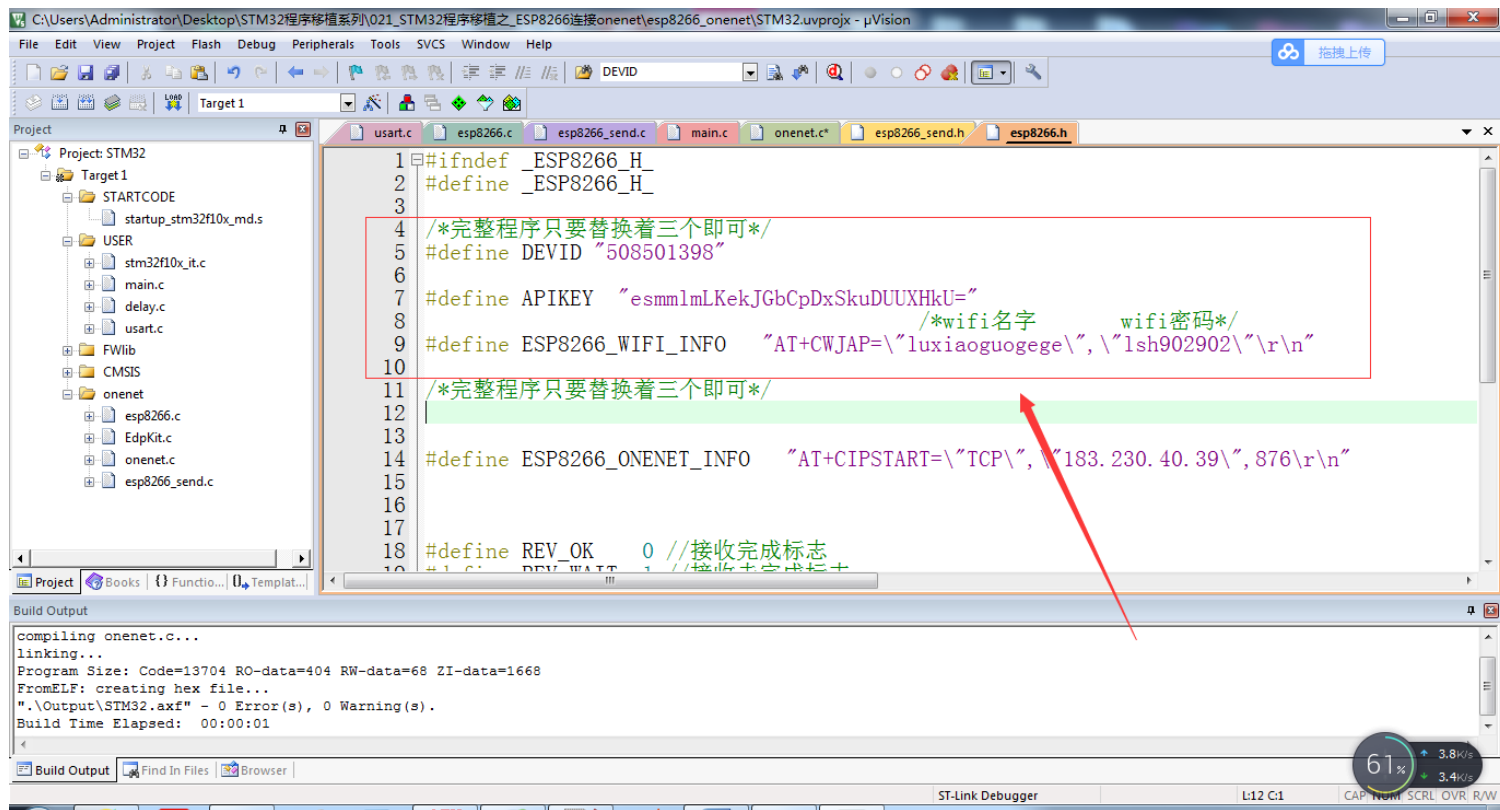
设备 ID 和秘钥还有 wifi 密码请看 esp8266.h 中修改



```
108 unsigned char OneNet_FillBuf(char *buf)
109 {
110     char text[16];
111     memset(text, 0, sizeof(text));
112     /*send_data在esp8266_send.h文件那里*/
113     // send_data.Data1 = 10;
114     // send_data.Data2 = 20;
115
116     strcpy(buf, "(");
117
118     memset(text, 0, sizeof(text)); /*清空text数组*/
119     sprintf(text, "\\Red_Led\\:%d", send_data.Data1); /*(1)*/
120     strcat(buf, text);
121
122     memset(text, 0, sizeof(text)); /*清空text数组*/
123     sprintf(text, "\\Green_Led\\:%d", send_data.Data2); /*(2)*/
124     strcat(buf, text);
125
126     /*
127     注意：数据传输格式按照上面的格式来添加
128     在(1)点中那个%d后面有标点符号，(2)%d后面没有标点符号这个要注意
129     反正插入的数据最后面一个不能有标点符号，其他的都要，不然的话数据传输不了
130     Red_Led和Green_Led是数据流名字，后面的就是数据了
131     */
132
133     strcat(buf, "!");
134
135     return strlen(buf);
136 }
```



```
10 OneNet_RevPro(dataPtr);
11
12 /*数据处理是下面这个子函数*/
13 void data_pro(char *req) /*接收数据进行处理*/
14 {
15     char *dataPtr = NULL;
16     char numBuf[10];
17     int num = 0;
18
19     dataPtr = strchr(req, ':'); /*搜索*/
20     if(dataPtr != NULL) /*如果找到了*/
21     {
22         dataPtr++;
23
24         while(*dataPtr != '0' && *dataPtr != '9') /*判断是否是下发的命令控制数据*/
25         {
26             numBuf[num++] = *dataPtr++;
27         }
28
29         num = atoi((const char *)numBuf); /*转为数值形式*/
30
31         /*上面是数据处理，下面是接收的指令*/
32         /*onenet上面开关指令*/
33         红灯：
34         开关开值：redled:1
35         开关关值：redled:0
36         绿灯：
37         开关开值：greenled:1
38         开关关值：greenled:0
39     }
40 }
```

18. 我这里的数据流开关和曲线都是用一个，现实中不应该出现这样的情况，一般来说我们数据只要传到云端，开关是从云端弄下来。我这里设计是弄下来之后又传上去是为了方便看现象，最好不要这么干。那么本次教程花了半天时间来弄，有什么问题的自己百度解决了，问我我估计也忘了。谢谢大家

19. 欢迎关注公众号：luxiaoguogege

