

微信读取温湿度传感器数值开发实例

摘要 (Abstract)

本文介绍如何使用 **MiCOKit 开发套件** 开发一个简单的，通过微信读取温湿度传感器数值的应用实例。

适合读者 (Suitable Readers)

本文适用于所有 MiCOKit 开发套件的开发者，并适合所有 MiCO-物联网 (IoT) 设备开发者参考。

获取更多帮助 (More Help)

MiCO 开发团队向您推荐：MiCO 开发者学习网站：<http://mico.io/> (开发者中心)，获取更多最新资料。

手机微信“扫一扫”关注：“MiCO 总动员”公众号，获取 MiCO 团队小伙伴最新活动信息。



登录上海庆科官方网站：<http://mxchip.com/>，获取公司最新产品信息。

版权声明 (Copyright Notice)

Copyright (c) 2015 MDWG Trust and the persons identified as the document authors. All rights reserved.

目 录

微信读取温湿度传感器数值开发实例	1
1. 概述	2
2. 准备工作	2
3. 开发流程	2
4. 详细步骤	4
4.1. 注册开发者账号	4
4.2. 使用个人微信号开通测试公众号	4
4.3. 在 FogCloud 上创建、定义自己的产品	5
4.4. 在 FogCloud 上创建产品对应的微信 APP	6
4.5. Github 上创建微信 APP 代码托管仓库	8
4.6. 配置微信 APP 以及微信测试公众号	8
4.7. 使用 MiCO SDK 开发固件	12
4.8. 用 IAR 或 MDK 工具开发 MiCOKit 固件代码（代码注释）	18
4.9. 使用 Github 工具托管 APP 代码	21
4.10. FogCloud 上生成设备二维码	24
4.11. 使用手机微信扫码，测试“Airkiss”配网功能以及设备控制功能	28
5. 版本更新	30

1. 概述

本文档仅介绍如何使用 **MiCOKit 开发套件**开发一个简单的 通过微信读取温湿度传感器数值的应用实例过程。

2. 准备工作

注意：开始前请确定射频驱动为最新版本

版本查询及升级方法请参考 MiCO 社区 → wiki 中心 → MiCOKit 板块射频驱动升级

1. 以 MiCOKit-3288 开发套件为例；
2. 开发工具请使用 IAR7.3 版本及以上；
3. FogCloud 开发者账号（Fog 云使用、开发必须）；
4. SDK_MiCOKit_V2.2.0.3（下载请至：http://mico.io/wiki/doku.php?id=micokit_sdk）；
5. 个人微信号（开通测试公众号）；
6. github 个人账号（托管微信 APP 代码）；
7. 网页编辑工具（sublime 等）；
8. 大致了解 MQTT 协议及 json 格式。

3. 开发流程

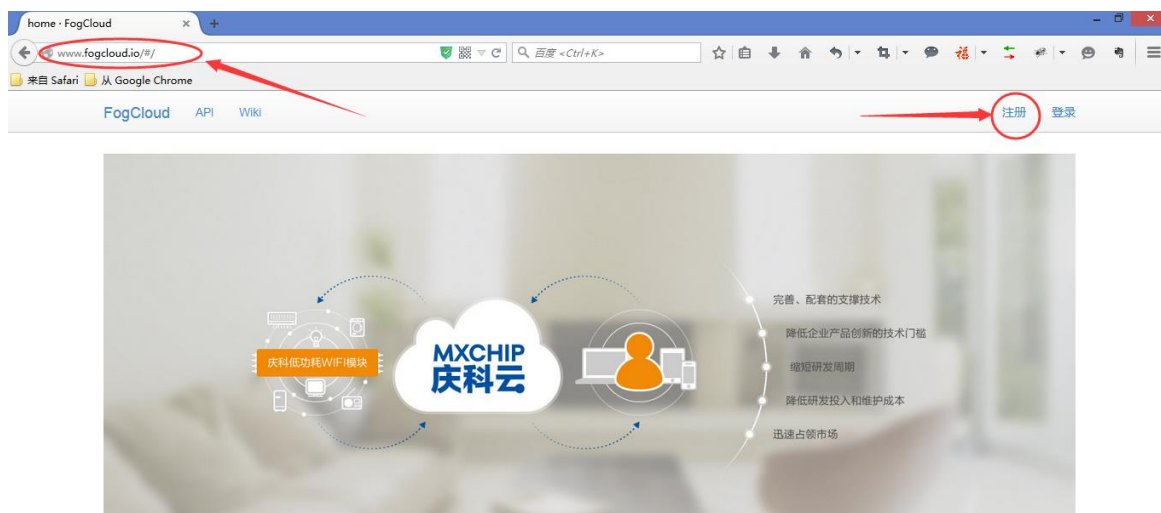
1. 注册 FogCloud 开发者账号；
2. 使用个人微信号开通测试公众号；
3. 在 FogCloud 上创建、定义自己的产品；
4. 在 FogCloud 上创建产品对应的微信 APP；
5. Github 上创建微信 APP 代码托管仓库；
6. 配置微信 APP 和微信测试公众号；
7. 使用 MiCOKit SDK 开发固件；
8. 用 IAR 或 MDK 工具开发 MiCOKit 固件代码（代码注释）；

9. 使用 Github 工具托管 APP 代码；
10. 在 FogCloud 上生成设备微信二维码；
11. 手机微信扫码，测试 Airkiss 配网功能、设备控制功能。

4. 详细步骤

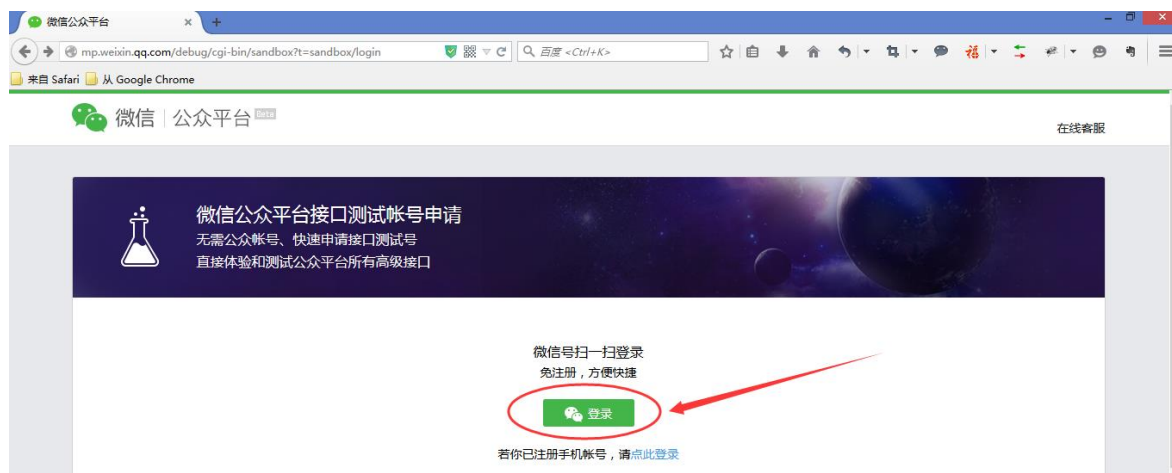
4.1. 注册开发者账号

登录 www.fogcloud.io 直接注册账号即可。该账号将用来管理你的产品及 APP。

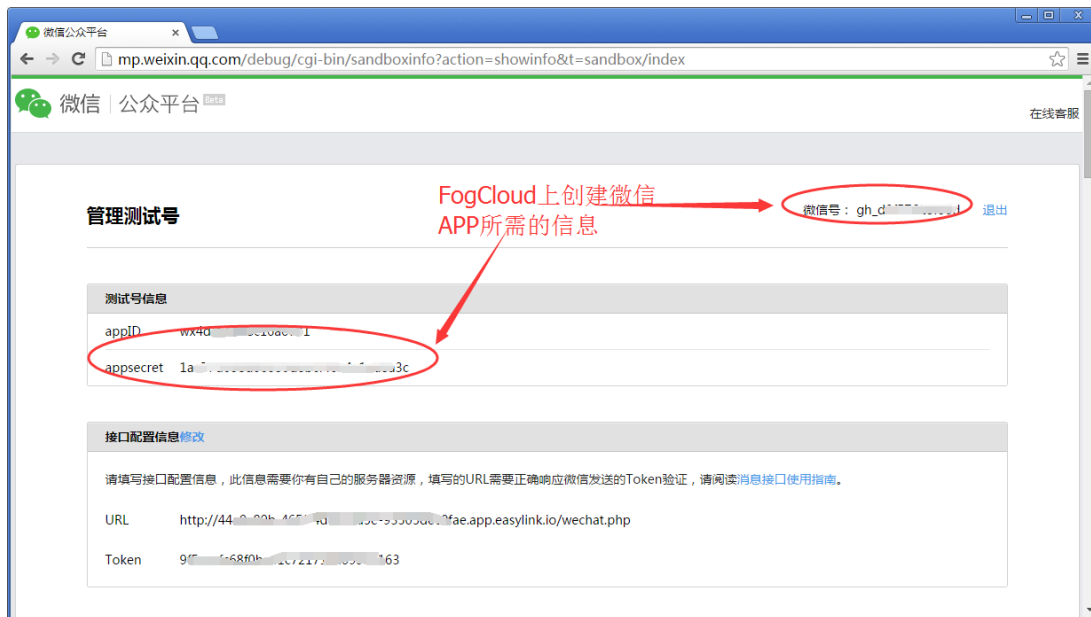


4.2. 使用个人微信号开通测试公众号

浏览器打开 <http://mp.weixin.qq.com/debug/cgi-bin/sandbox?t=sandbox/login> , 点击登录, 使用手机微信扫码, 进入后即开通了测试公众号。



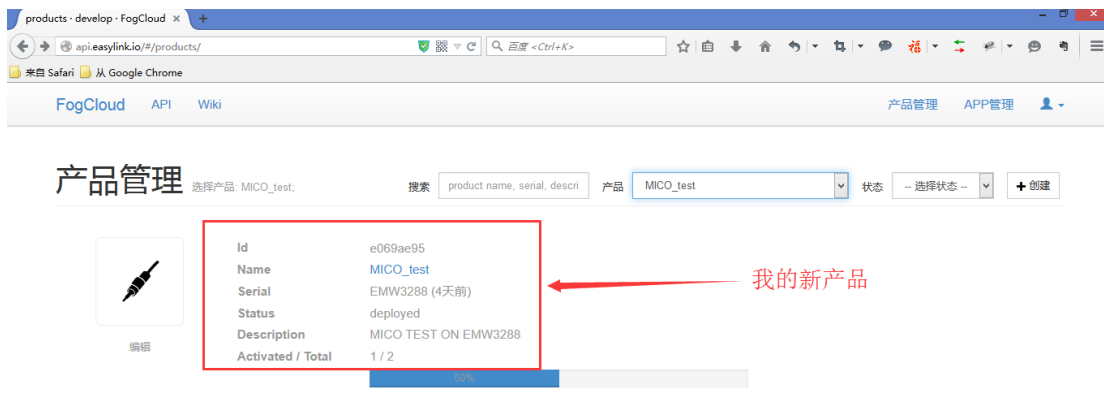
获得微信号、appID、appsecret,用于后续在 FogCloud 上创建产品对应的微信 APP。



4.3. 在 FogCloud 上创建、定义自己的产品



根据提示填写相关信息，创建完成后，如下



点击产品名称，进入详细信息：

产品 ID/KEY 会写到设备固件代码中；

创建产品的数据模型，数据模型是用来在云端定义产品功能、性能等特征数据的标准格式，可以储存设备监控、收集、控制、用户行为等数据，从而对数据进行分析，提升产品服务，开发案例过程中定义数据模型，能够有效帮助开发者将 APP、云端与设备端的关键功能、特征数据等同步，避免开发过程出错。

名字	类型	读写属性	属性名	描述	时间序列	操作
dht11_humidity	int	0	湿度	湿度	是	✕ ✎
dht11_temperature	int	0	温度	温度	是	✕ ✎

本实例读取温度与湿度数据，所需创建的数据点为：

1) 温度 (temp) 读写属性为 0 时间序列为是

2) 湿度 (humi) 读写属性为 0 时间序列为是

属性名和描述可按个人习惯添加

注意：在创建数据点时，“时间序列”选项：选中时云端保存历史数据；不选时云端只保存最新的数据。数据模型的具体作用在后面的进阶教程中详细讲解。

4.4. 在 FogCloud 上创建产品对应的微信 APP

Id	Name	secret key	Description
3a9c...cd64	micokit_test_app	ba1...142	micokit_test_app

根据提示,填写相关信息,其中:微信号、AppID/AppSecret 从步骤 2 中开通的微信测试公众号中获得。

微信

微信号

Enter weixin ID (e.g. gh_xxxx) 填入步骤二中的微信号

AppID

Enter weixin appid 填入步骤二中的AppID

AppSecret

Enter weixin app secret 填入步骤二中的获得的AppSecret

(b)Git 部署 (同步微信 APP 代码到 FogCloud)

The screenshot shows the 'APP' detail page in FogCloud. The APP ID is 3a9dcdb5-c1a8-4037-a126-0c868de0cd64. The name is 'micokit_test_app'. The secret key is 'ba1cfaa1de9bb17a13d2c34fb92a142'. The description is 'micokit_test_app'. The 'Git部署' (Git Deployment) tab is selected. The 'Repo' field contains 'https://github.com/wangeshen/MiCOKit_test.git'. The 'Deploy key' field contains 'ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQADZ+W094ZOAlFoAT2EXFJZ1dlBpD'. The 'Web Hook Url' field contains 'http://api.easylink.io/v1/repo/release?id=3a9dcdb5-c1a8-4037-a126-0c868de0cd64&a'. The '发布' (Publish) button is highlighted. Red arrows point to the 'Repo' field with the label 'github托管仓库地址' (GitHub hosted repository address), to the 'Deploy key' field with the label 'github上代码最新提交记录' (Latest commit record on GitHub), and to the 'Web Hook Url' field with the label '填写到github仓库配置里, 用于自动同步代码' (Fill in the GitHub repository configuration for automatic code synchronization). The '发布' button is labeled '发布微信APP代码' (Publish WeChat APP code).

其中：

Repo 即步骤 5 中创建的 github 仓库地址，填写后保存；

Deploy key 和 Web Hook Url 可添加到 github 仓库的设置中 以自动同步代码到 FogCloud 也可以不添加，但是 github 仓库中代码更新后，需要手动点击“发布”按钮来同步代码，同步后右边可看到最新的代码提交记录。

The screenshot shows the 'Settings' page for the repository 'wangeshen / MiCOKit_test'. The 'Webhooks & Services' tab is selected. The 'Settings' section shows the repository name 'MiCOKit_test' and the default branch 'master'. The 'Features' section shows 'Wikis' and 'Issues' are enabled. Red arrows point to the 'Webhooks & Services' tab with the label 'Webhook和Deploy key 设置' (Webhook and Deploy key settings).

(c) 微信菜单管理

通过 FogCloud 提供的微信公众号首页菜单定制功能，方便的定制手机端微信上的控制界面及功能；至少包含“Airkiss”按钮，打开微信 Airkiss 配网功能，“OAuth”按钮（名称可自定义）进入设备控制。

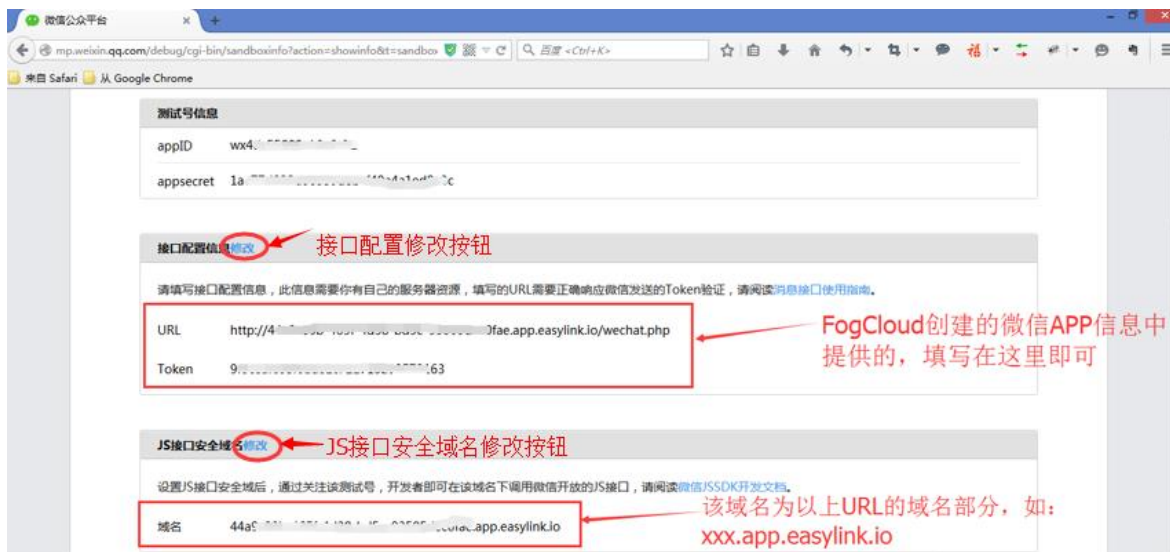


(d) 设置微信测试公众号 URL 与 Token 在创建的 APP 信息中可以找到

修改配置信息：

URL 一般为：<http://4addb71f-1b5c-XXXX-94c5-f3d93795a17e.app.easylink.io/wechat.php>

下划线部分作为 JS 接口安全域名和授权回调页面域名。**域名中不包含 http:// 和 /wechat.php !!!**



开通微信测试号的所有测试功能，如下图 1,2,3：

对话服务	发送消息	接收事件推送	无上限	
		接收语音识别结果	无上限	1 关闭
		自动回复	无上限	
		客服接口	500000	显示“关闭” 实际代表“开启”
		群发接口	详情	
		模板消息（业务通知）	100000	
	用户管理	用户分组管理	详情	
		设置用户备注名	10000	
		获取用户基本信息	500000	
		获取用户列表	500	2
		获取用户地理位置	无上限	关闭
	推广支持	生成带参数二维码	100000	
		长链接转短链接接口	1000	
	界面丰富	自定义菜单	详情	
	素材管理	素材管理接口	详情	
功能服务	智能接口	语义理解接口	1000	3
	设备功能	设备功能接口	无上限	关闭
	多客服	获取客服聊天记录	5000	
		客服管理	详情	
		会话控制	详情	4
	网页账号	网页授权获取用户基本信息	无上限	修改

点击 4 修改，填写网页授权域名，打开如下：

授权回调页面域名： 与JS接口安全域名一致

4adb71f-1b5c-94c5-f3d93795a17e.app.easylink.ic

用户在网页授权页同意授权给公众号后，微信会将授权数据传给一个回调页面，回调页面需在此域名下，以确保安全可靠。沙盒号回调地址支持域名和ip，正式公众号回调地址只支持域名。

确认 取消

同“JS 接口安全域名”，格式为前面提供的 URL 的域名部分，如 xxxx.app.easylink.io，其中“xxxx”为 FogCloud 上创建的微信 APP 的 id。

4.7. 使用 MiCO SDK 开发固件

(a) 登陆 MiCO 开发者网站 mico.io，去 MiCO 社区注册账号，并登陆；



(b) 重新打开 mico.io 首页，开发者中心 ==> Wiki 中心—MiCOKit 发布中心，下载 MiCOKit SDK。

关于MiCOKit

MiCOKit概述

- 获取MiCOKit简介，请参阅: [MiCOKit概述](#)

MiCOKit SDK发布中心

- 获取MiCOKit最新版本SDK，请至: [MiCOKit SDK发布中心](#)

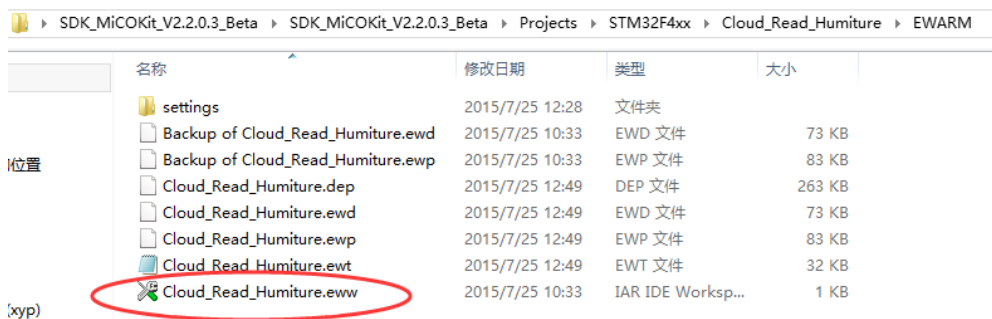
MiCOKit共性资料

MiCOKit SDK 下载

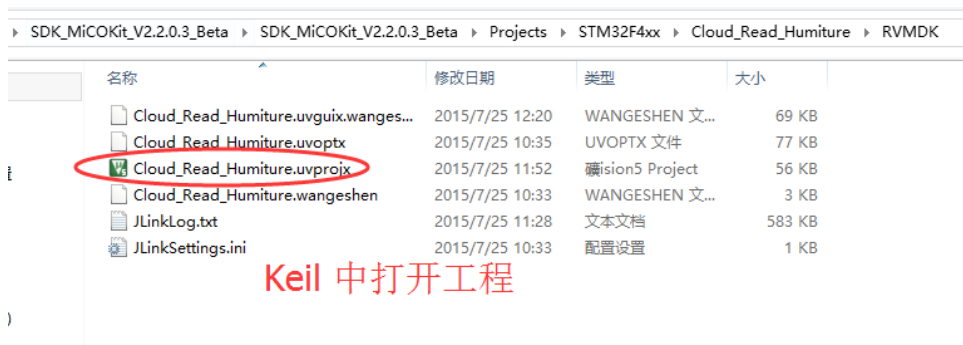
MiCOKit上手玩

- [MiCOKit用户体验](#) -快速体验MiCOKit开发板各应用功能

(c) 打开 MiCOKit SDK 中的微信开发实例工程：

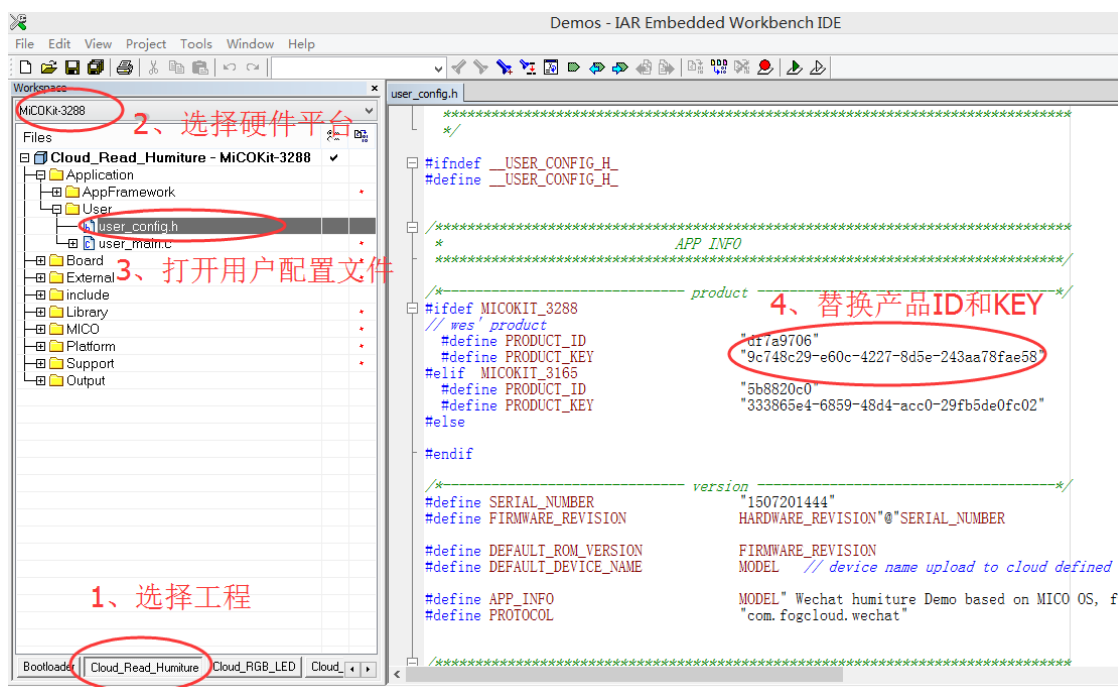


IAR 中打开工程

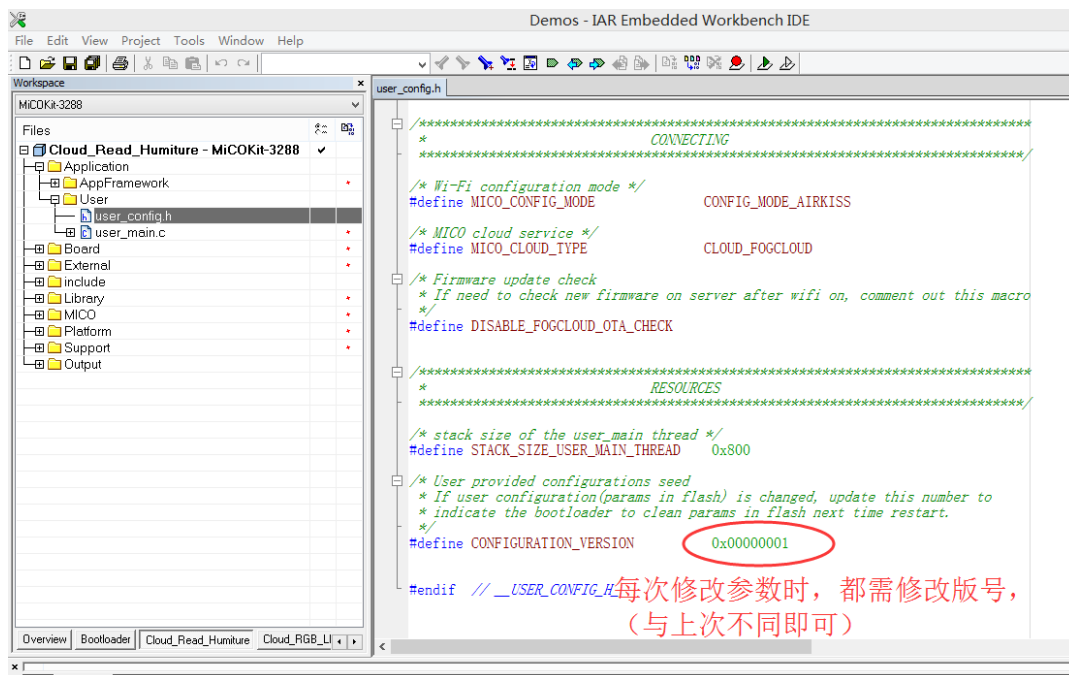


1), 开发者手中拿到的可能是硬件平台可能是 MiCOKit3288 或者 MiCOKit3165 , 在编译工程时要**先选择硬件平台**;

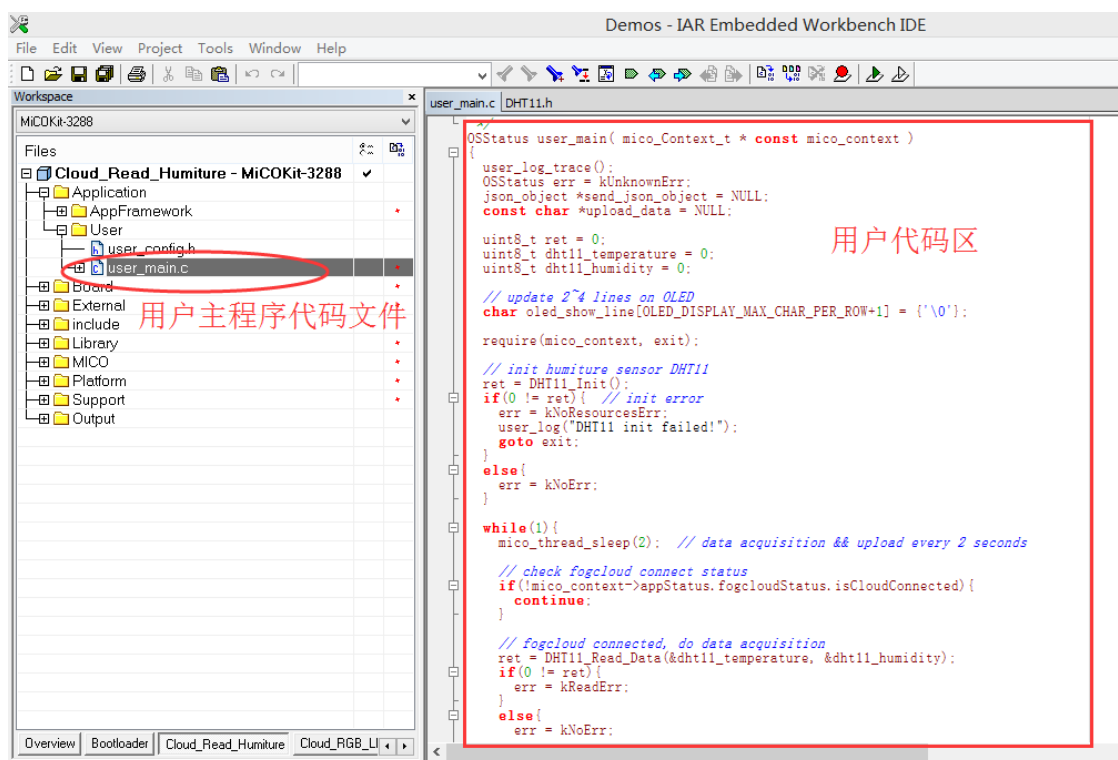
2) 将 FogCloud 上创建的产品 ID/KEY 写入固件 (**必须替换**):



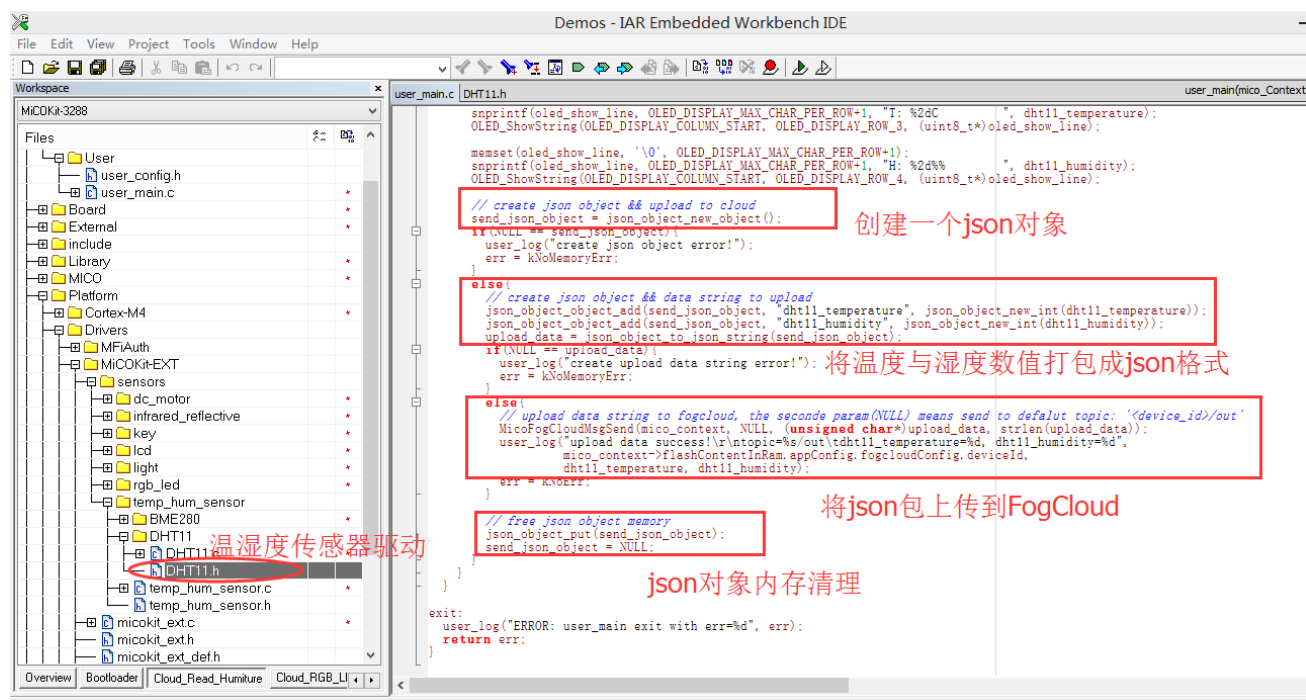
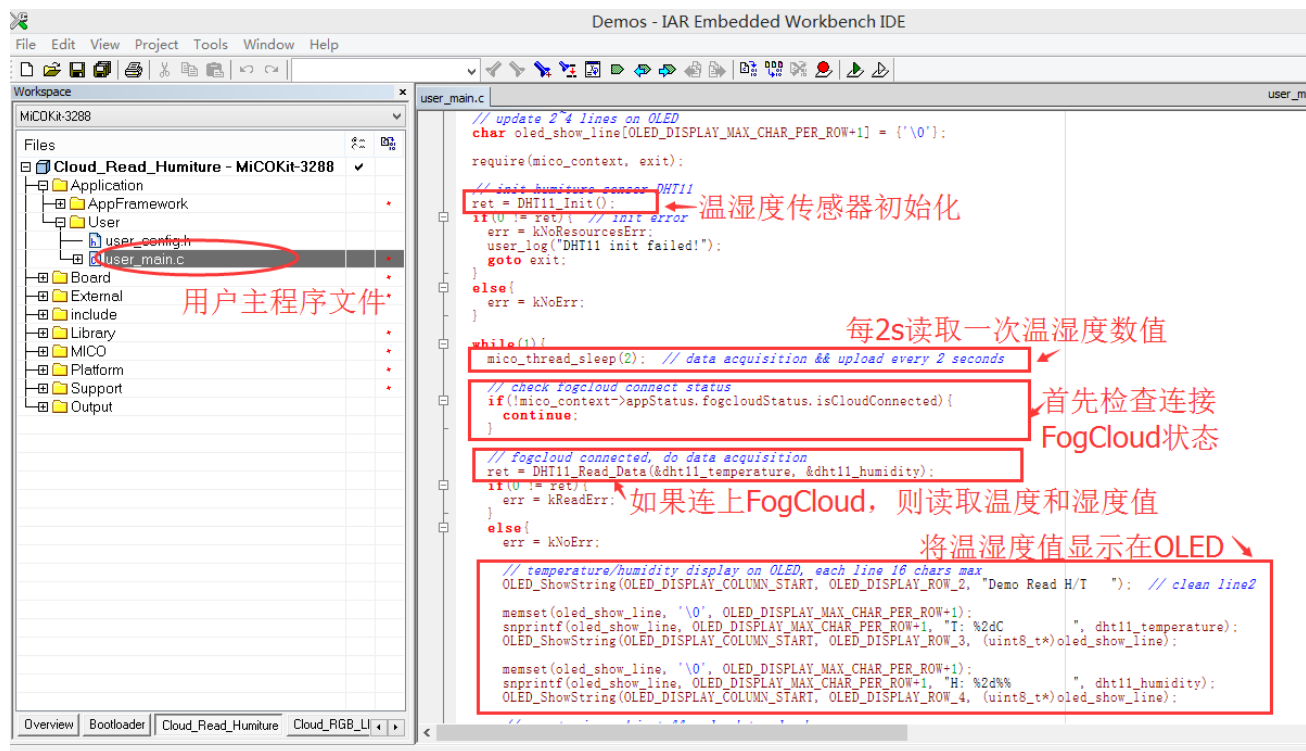
3) **修改版本号** , 这样才能把新的产品 ID/KEY 烧入 FLASH



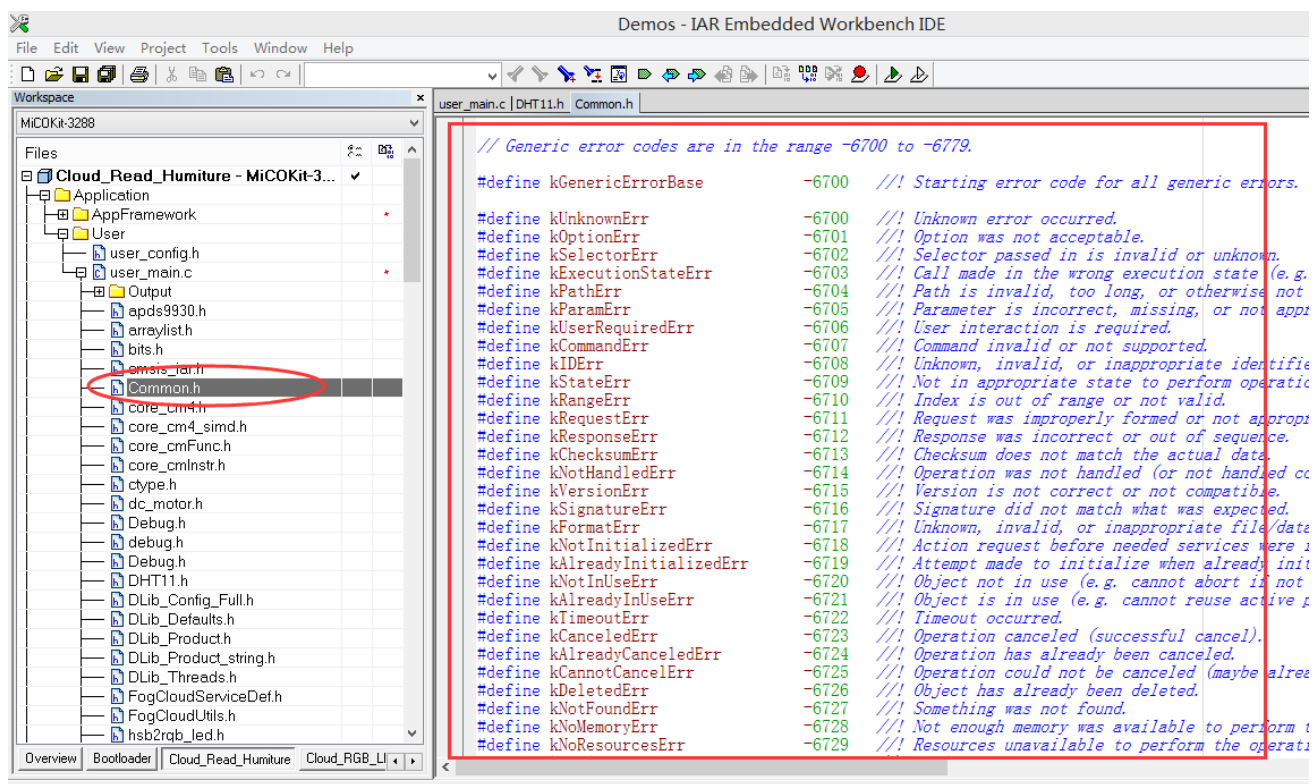
4) 添加读取温湿度代码：



5) 改为从开发板上读取温湿度数据后，打包成 JSON 格式，并上传到 FogCloud。



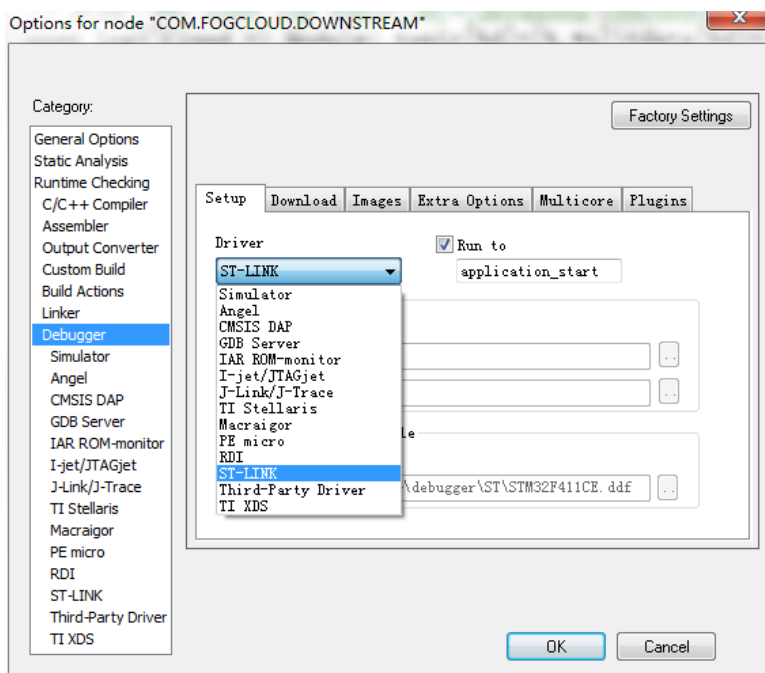
6) 错误代码参考：在运行过程中如果出现错误，可在 user_log 中查看错误代码。



```
-6700 ///! Unknown error occurred.
-6701 ///! Option was not acceptable.
-6702 ///! Selector passed in is invalid or unknown.
-6703 ///! Call made in the wrong execution state (e.g. called at interrupt time).
-6704 ///! Path is invalid, too long, or otherwise not usable.
-6705 ///! Parameter is incorrect, missing, or not appropriate.
-6706 ///! User interaction is required.
-6707 ///! Command invalid or not supported.
-6708 ///! Unknown, invalid, or inappropriate identifier.
-6709 ///! Not in appropriate state to perform operation.
-6710 ///! Index is out of range or not valid.
-6711 ///! Request was improperly formed or not appropriate.
-6712 ///! Response was incorrect or out of sequence.
-6713 ///! Checksum does not match the actual data.
-6714 ///! Operation was not handled (or not handled completely).
-6715 ///! Version is not correct or not compatible.
-6716 ///! Signature did not match what was expected.
-6717 ///! Unknown, invalid, or inappropriate file/data format.
-6718 ///! Action request before needed services were initialized.
-6719 ///! Attempt made to initialize when already initialized.
-6720 ///! Object not in use (e.g. cannot abort if not already in use).
-6721 ///! Object is in use (e.g. cannot reuse active param blocks).
```

7) MiCOKit SDK 固件烧录。(详细固件烧录方法及步骤请参考 <http://mico.io> wiki 中心)

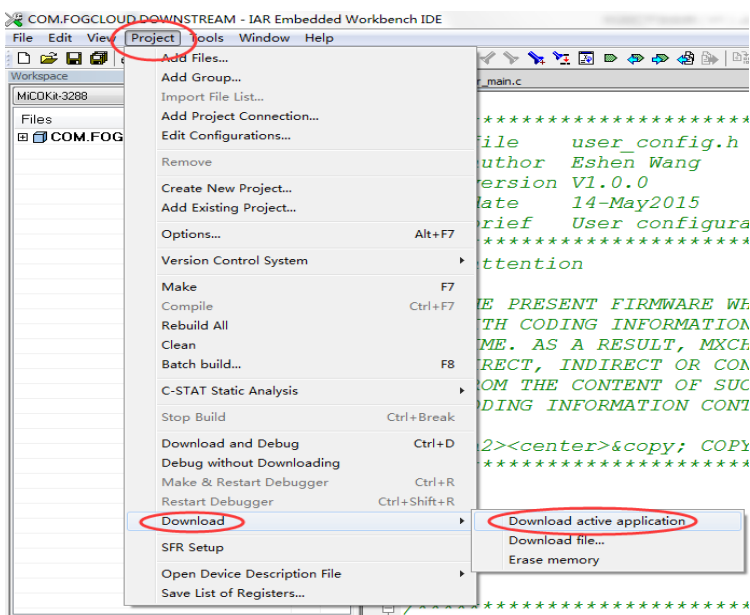
选择使用的烧录工具 J-Link 或者 ST-LINK:



8) 编译/连接:



9) 烧录/下载:



4.8. 用 IAR 或 MDK 工具开发 MiCOKit 固件代码（代码注释）

//应用程序入口在 application_start(void)，一系列动作（如配网、连接云等）以后，用户程序入口在这里。

```
OSStatus user_main( mico_Context_t * const mico_context )  
  
{  
  
    user_log_trace();  
  
    OSStatus err = kUnknownErr;  
  
    json_object *send_json_object = NULL;  
  
    const char *upload_data = NULL;  
  
    uint8_t ret = 0;  
  
    uint8_t dht11_temperature = 0;  
  
    uint8_t dht11_humidity = 0;  
  
    // 将温度与湿度数值同步显示在 OLED 上  
  
    char oled_show_line[OLED_DISPLAY_MAX_CHAR_PER_ROW+1] = {'\0'};  
  
    require(mico_context, exit);  
  
    // 初始化温湿度传感器 DHT11  
  
    ret = DHT11_Init();  
  
    if(0 != ret){  
  
        // 检查初始化时是否有错误  
  
        err = kNoResourcesErr;  
  
        user_log("DHT11 init failed!");  
  
        goto exit;  
    }  
  
    else{
```

```
err = kNoErr;

}

while(1){

mico_thread_sleep(2);  // 每隔 2 秒进行一次数据采集和上报云端

    // 检查与 Fogcloud 的连接状态

    if(!mico_context->appStatus.fogcloudStatus.isCloudConnected){

        continue;

    }

    // 与 Fogcloud 连接后，开始数据采集

    ret = DHT11_Read_Data(&dht11_temperature, &dht11_humidity);

    if(0 != ret){

        err = kReadErr;

    }

    else{

        err = kNoErr;

        // 在 OLED 显示温度与湿度数值，每行最多显示 16 个字母

        OLED_ShowString(OLED_DISPLAY_COLUMN_START, OLED_DISPLAY_ROW_2, "
"); // 第二行不显示

        memset(oled_show_line, '\0', OLED_DISPLAY_MAX_CHAR_PER_ROW+1);

        sprintf(oled_show_line, OLED_DISPLAY_MAX_CHAR_PER_ROW+1, "T: %2dC",
dht11_temperature);

        OLED_ShowString(OLED_DISPLAY_COLUMN_START, OLED_DISPLAY_ROW_3,
(uint8_t*)oled_show_line);
```

```
memset(oled_show_line, '\0', OLED_DISPLAY_MAX_CHAR_PER_ROW+1);

snprintf(oled_show_line, OLED_DISPLAY_MAX_CHAR_PER_ROW+1, "H: %2d%% ",
dht11_humidity);

OLED_ShowString(OLED_DISPLAY_COLUMN_START, OLED_DISPLAY_ROW_4,
(uint8_t*)oled_show_line);
```

```
// 创建一个 json 对象
```

```
send_json_object = json_object_new_object();
```

```
if(NULL == send_json_object){
```

```
    user_log("create json object error!");
```

```
    err = kNoMemoryErr;
```

```
}
```

```
else{
```

```
    // 创建完成后，将温度与湿度数值打包为 json 格式
```

```
    json_object_object_add(send_json_object, "dht11_temperature",
    json_object_new_int(dht11_temperature));
```

```
    json_object_object_add(send_json_object, "dht11_humidity",
    json_object_new_int(dht11_humidity));
```

```
    upload_data = json_object_to_json_string(send_json_object);
```

```
    if(NULL == upload_data){
```

```
        user_log("create upload data string error!");
```

```
        err = kNoMemoryErr;
```

```
}
```

```
else{
```

```
// 将json 包发至 Fogcloud, 第二个参数(NULL)代表发送默认值: '<device_id>/out'
```

```
    MicoFogCloudMsgSend(mico_context, NULL, (unsigned char*)upload_data,
strlen(upload_data));

    user_log("upload data success!\r\ntopic=%s/out\tdht11_temperature=%d,
dht11_humidity=%d",

            mico_context->flashContentInRam.appConfig.fogcloudConfig.deviceId,

            dht11_temperature, dht11_humidity);

    err = kNoErr;

}
```

```
// 释放 json 对象, 这步不能省略, 否则会造成内存溢出
```

```
    json_object_put(send_json_object);

    send_json_object = NULL;

}

}

}
```

exit:

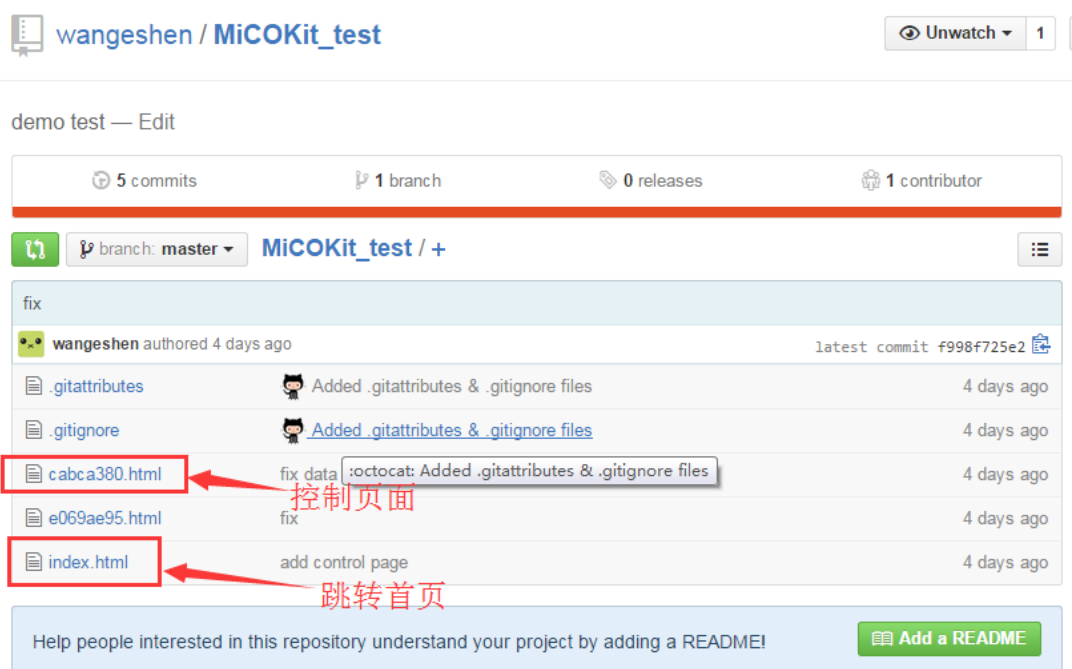
```
    user_log("ERROR: user_main exit with err=%d", err);

    return err;

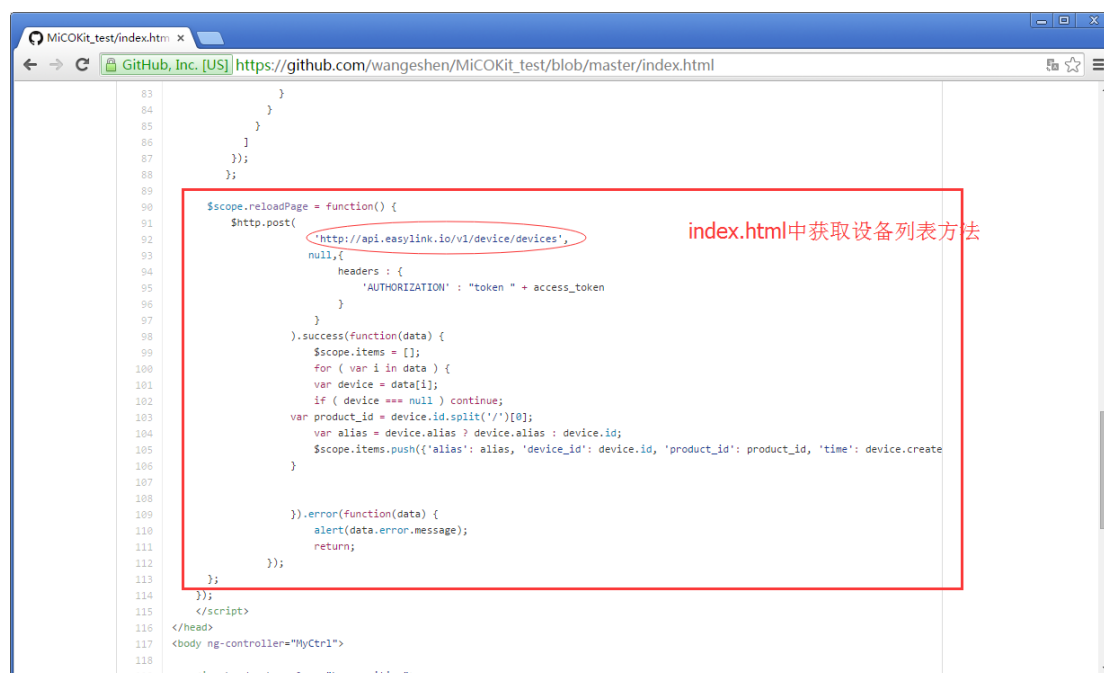
}
```

4.9. 使用 Github 工具托管 APP 代码

在例程包中 (MiCOKit SDK 的 APP 目录下) 找到微信 APP 控制页面代码 (index.html 和 yourID.html), 将 yourID.html.重命名为你的产品 ID, 如 cabca380.html。拷贝这两个文件到本地 git 仓库, 再同步到 git 服务器。同步方法详见《上传文件到 GitHub》。

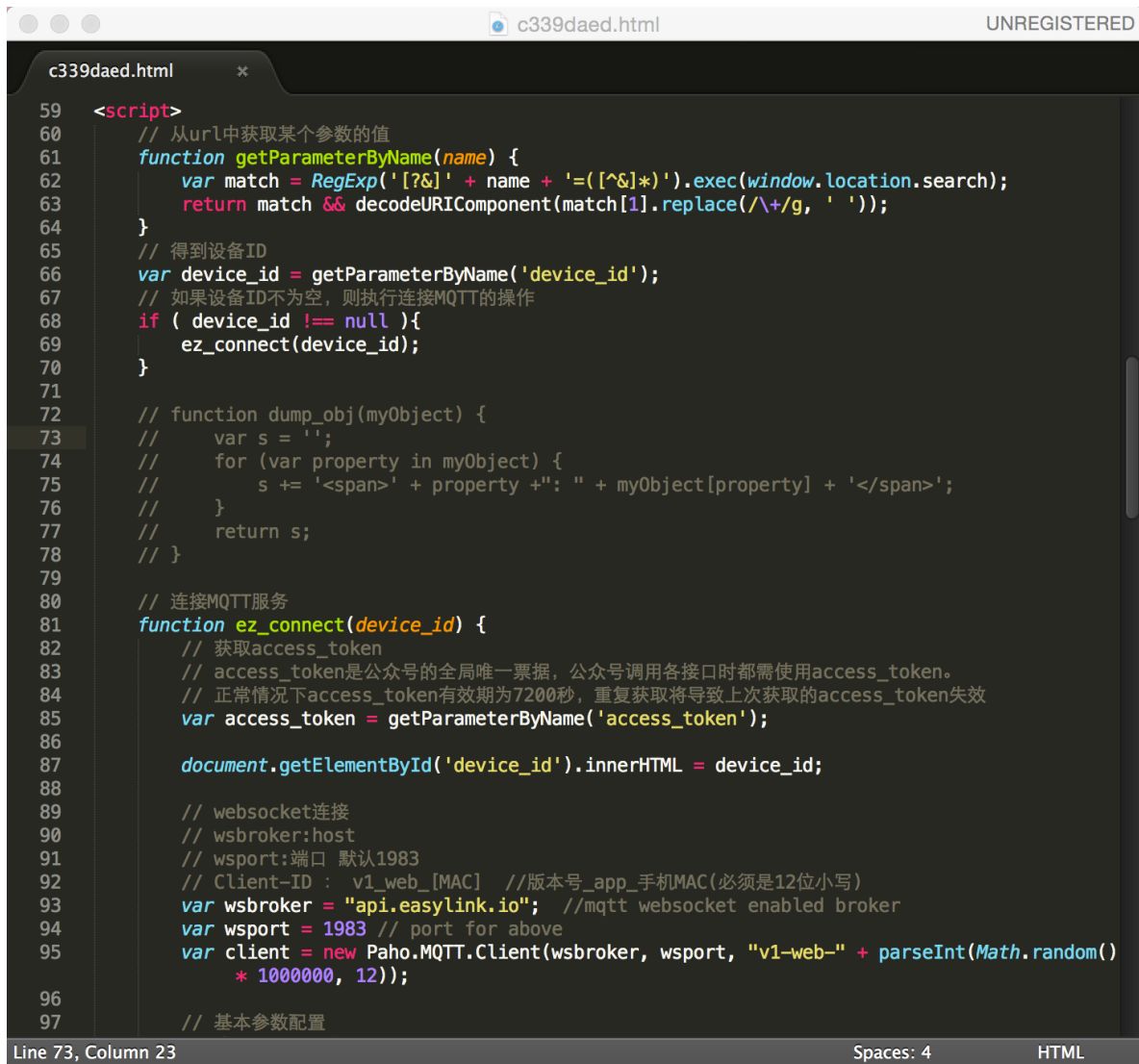


index.html —— OAuth 按钮跳转到该页面（一般为设备列表页面，可不作改动）



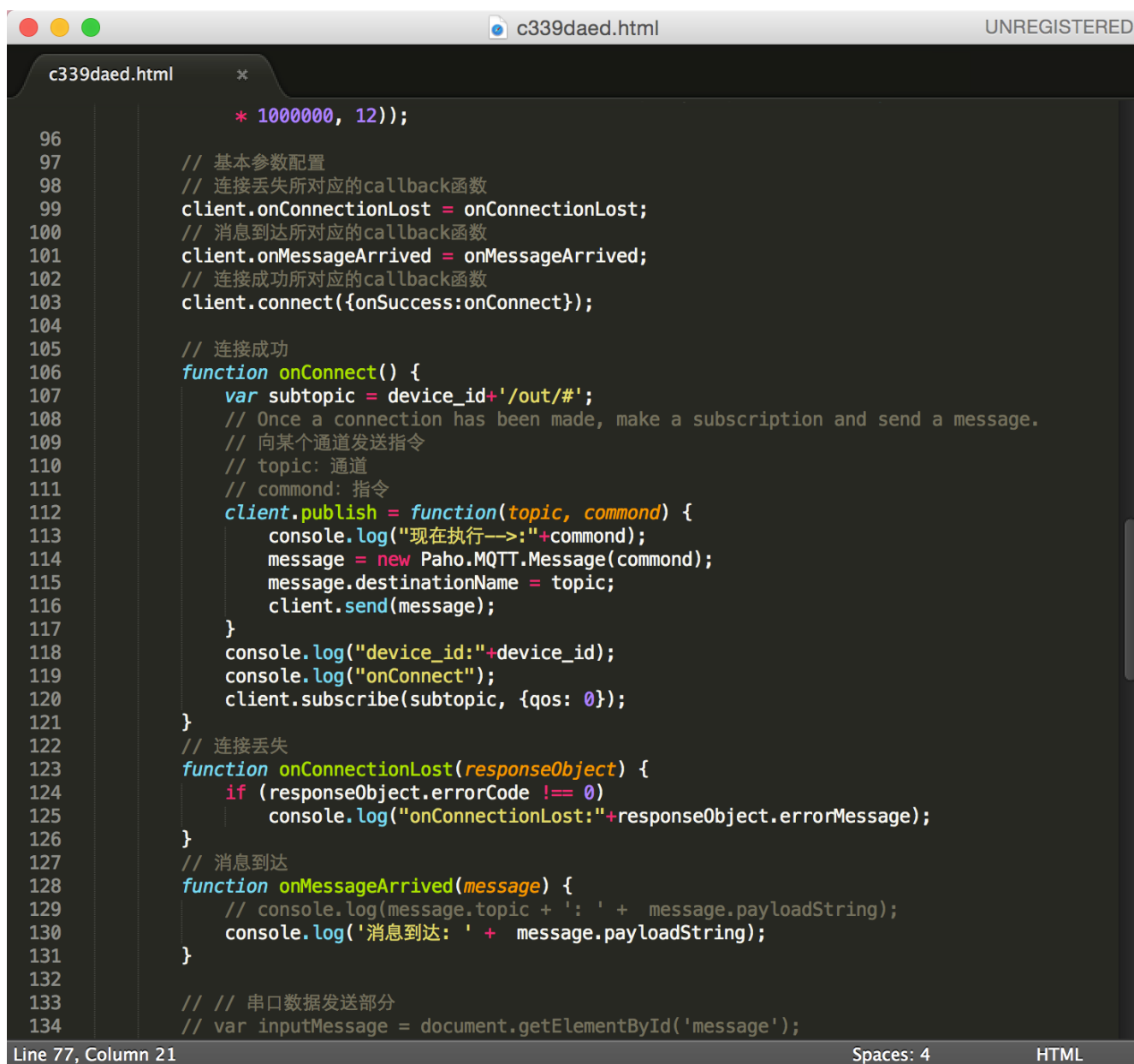
xxxx.html —— 设备控制页面（其中 xxxx 为 FogCloud 上创建的产品 id）

详细代码见附件代码包，部分代码解释如下：



```
59 <script>
60 // 从url中获取某个参数的值
61 function getParameterByName(name) {
62     var match = RegExp('[?&]' + name + '=(^&)*').exec(window.location.search);
63     return match && decodeURIComponent(match[1].replace(/\+/g, ' '));
64 }
65 // 得到设备ID
66 var device_id = getParameterByName('device_id');
67 // 如果设备ID不为空, 则执行连接MQTT的操作
68 if ( device_id !== null ){
69     ez_connect(device_id);
70 }
71
72 // function dump_obj(myObject) {
73 //     var s = '';
74 //     for (var property in myObject) {
75 //         s += '<span>' + property + ': ' + myObject[property] + '</span>';
76 //     }
77 //     return s;
78 // }
79
80 // 连接MQTT服务
81 function ez_connect(device_id) {
82     // 获取access_token
83     // access_token是公众号的全局唯一票据, 公众号调用各接口时都需使用access_token。
84     // 正常情况下access_token有效期为7200秒, 重复获取将导致上次获取的access_token失效
85     var access_token = getParameterByName('access_token');
86
87     document.getElementById('device_id').innerHTML = device_id;
88
89     // websocket连接
90     // wsbroker: host
91     // wsport: 端口 默认1983
92     // Client-ID : v1_web_[MAC] //版本号_app_手机MAC(必须是12位小写)
93     var wsbroker = "api.easylink.io"; //mqtt websocket enabled broker
94     var wsport = 1983 // port for above
95     var client = new Paho.MQTT.Client(wsbroker, wsport, "v1-web-" + parseInt(Math.random()
96         * 1000000, 12));
97     // 基本参数配置
```

Line 73, Column 23 Spaces: 4 HTML



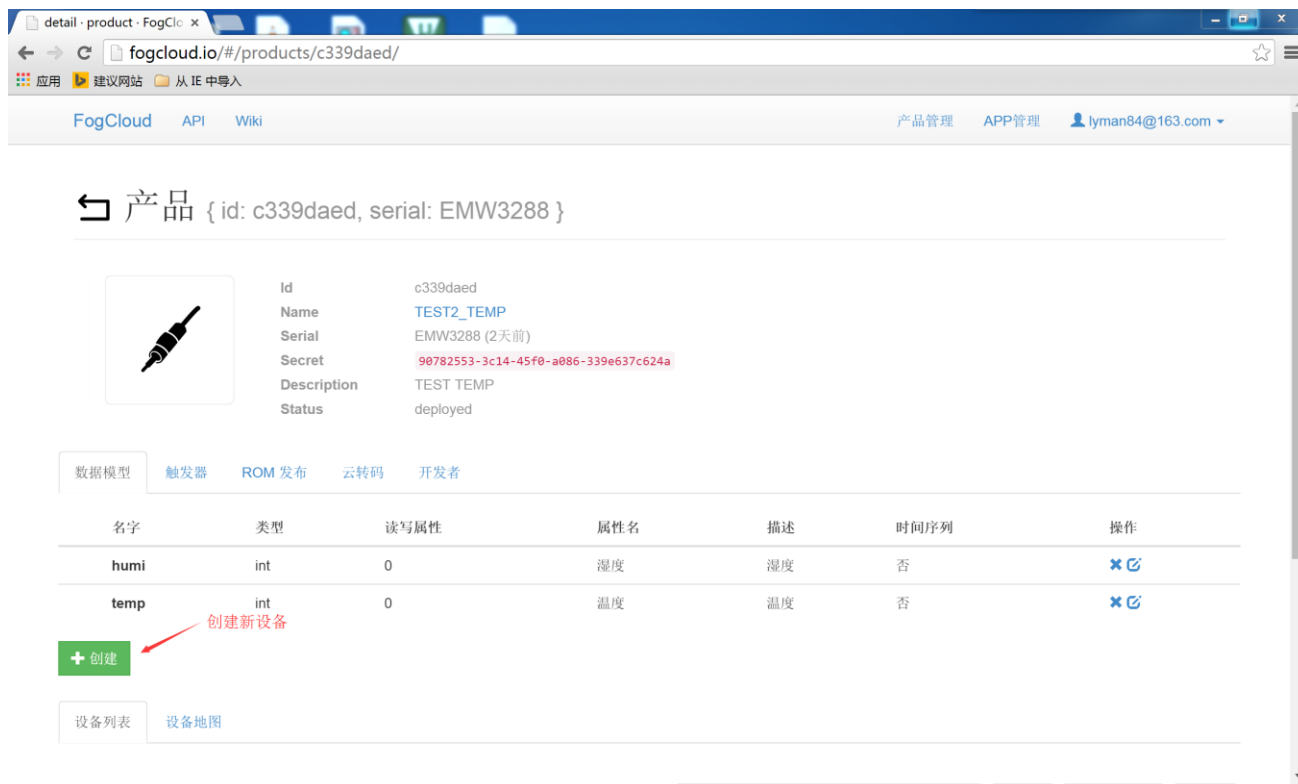
```

    * 1000000, 12));
96
97 // 基本参数配置
98 // 连接丢失所对应的callback函数
99 client.onConnectionLost = onConnectionLost;
100 // 消息到达所对应的callback函数
101 client.onMessageArrived = onMessageArrived;
102 // 连接成功所对应的callback函数
103 client.connect({onSuccess:onConnect});
104
105 // 连接成功
106 function onConnect() {
107     var subtopic = device_id+'/out/#';
108     // Once a connection has been made, make a subscription and send a message.
109     // 向某个通道发送指令
110     // topic: 通道
111     // command: 指令
112     client.publish = function(topic, command) {
113         console.log("现在执行-->:"+command);
114         message = new Paho.MQTT.Message(command);
115         message.destinationName = topic;
116         client.send(message);
117     }
118     console.log("device_id:"+device_id);
119     console.log("onConnect");
120     client.subscribe(subtopic, {qos: 0});
121 }
122 // 连接丢失
123 function onConnectionLost(responseObject) {
124     if (responseObject.errorCode !== 0)
125         console.log("onConnectionLost:"+responseObject.errorMessage);
126 }
127 // 消息到达
128 function onMessageArrived(message) {
129     // console.log(message.topic + ': ' + message.payloadString);
130     console.log('消息到达: ' + message.payloadString);
131 }
132
133 // // 串口数据发送部分
134 // var inputMessage = document.getElementById('message');
```

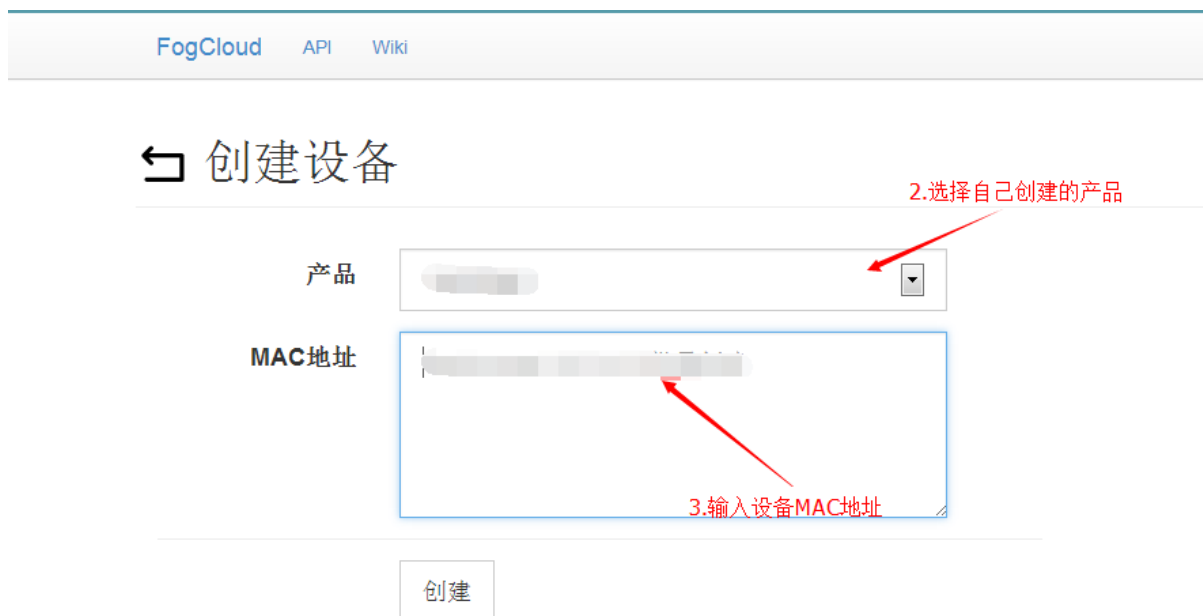
修改、并提交代码后，如果没有配置 WebHook 让 FogCloud 自动更新代码，则需要到 FogCloud 上手动点击“发布”按钮更新代码，并通过 git 提交记录确认是否更新成功。

4.10. FogCloud 上生成设备二维码

(a) 创建新设备



(b) 填写设备 MAC 地址(由小写字母和数字构成,设备上电后会向串口发送 MAC 地址及其他数据,可用串口工具查看)接口创建设备。



(b) 为设备生成微信二维码

点击“同步到微信”按钮：



选择创建的产品和要使用的 APP 后点击同步按钮。



成功后提示：{"result":200,"message":"success"}

同步设备到微信

```
{"result":200,"message":"success"}
```

极少数情况会出现 result : 500 问题。解决方法：

- (1) 确认步骤 4.6 节中微信测试号已开通所有测试功能。
- (2) 因为微信公共平台具有“延时”的问题,等待几分钟后重试。

返回到产品界面，可看到你的新设备。

detail · product · FogCloud x

fogcloud.io/#/products/c339daed/

应用 建议网站 从 IE 中导入

数据模型 触发器 ROM 发布 云转码 开发者

名字	类型	读写属性	属性名	描述	时间序列	操作
humi	int	0	湿度	湿度	否	✕ ↻
temp	int	0	温度	温度	否	✕ ↻

+ 创建

设备列表 设备地图

设备列表

点击进入获得设备二维码 搜索 mac

导出 同步到微信 + 创建

MICOKit-3288

Id c339daed/c89346918175

Serial

Status deployed

1天前

MICOKit-3288

Id c339daed/c8934691816c

Serial

Status deployed

2天前

detail · device · FogCloud x

fogcloud.io/#/products/c339daed/devices/c339daed_c8934691816c/

应用 建议网站 从 IE 中导入

FogCloud API Wiki

产品管理 APP管理 lyman84@163.com

设备 { id: c339daed/c8934691816c, serial: }

Product { id: c339daed, name: TEST2_TEMP, serial: }

Product Secret 90782553-3c14-45f0-a086-339e637c624a

Device Secret

Master Device Key

Last Active

Device Status

Bssid c8934691816c

设备二维码

TESE2_TEMP

c339daed/c8934691816c/in

c339daed/c8934691816c/out

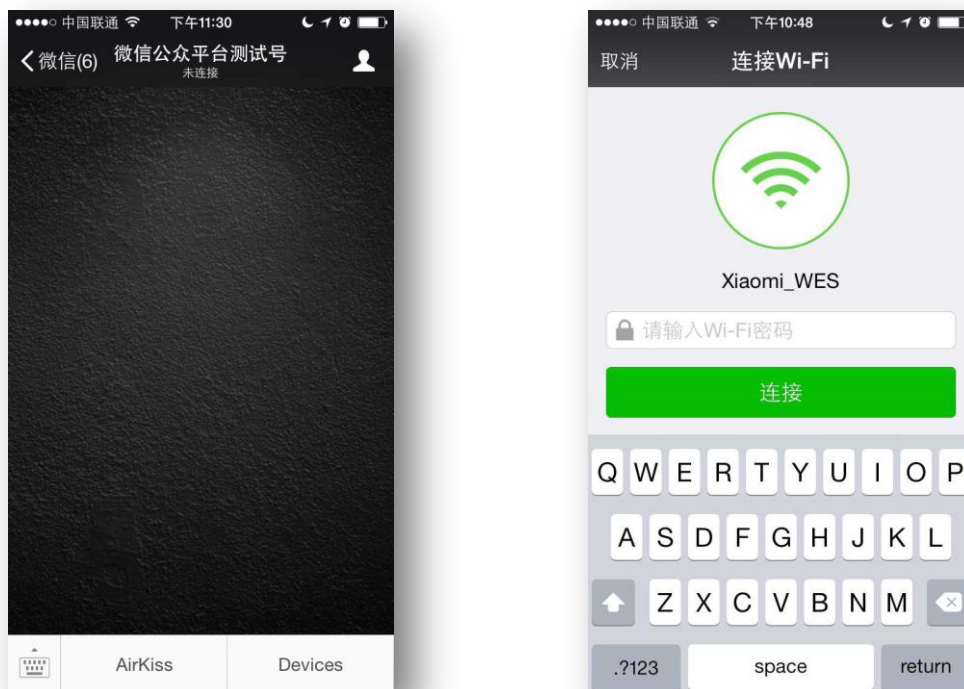
4.11. 使用手机微信扫码，测试“Airkiss”配网功能以及设备控制功能

(a) Airkiss 配网

Airkiss 技术可以帮助你的设备在没有人机交互的情况下智能配置当前 Wi-Fi 环境的 SSID 及密码。(假如你的智能设备是一颗灯泡，总没有屏幕和按键让你输入 SSID 及密码吧)

按设备上的 Easylink 按钮进入配网模式，底板上的 LED(D1)灯快速闪烁；

手机输入当前所在环境的 wifi 密码，点击连接，成功或超时会自动跳出该页面。



(b) 设备控制

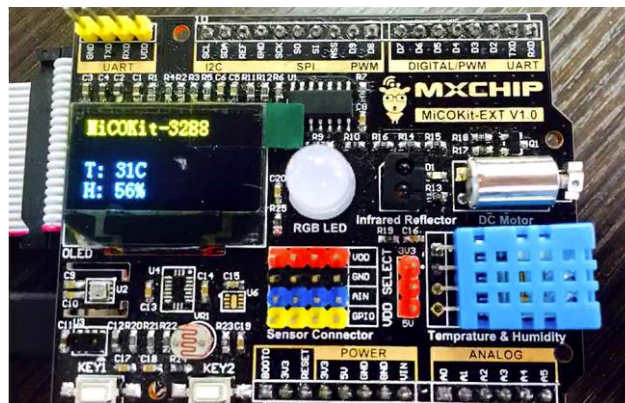
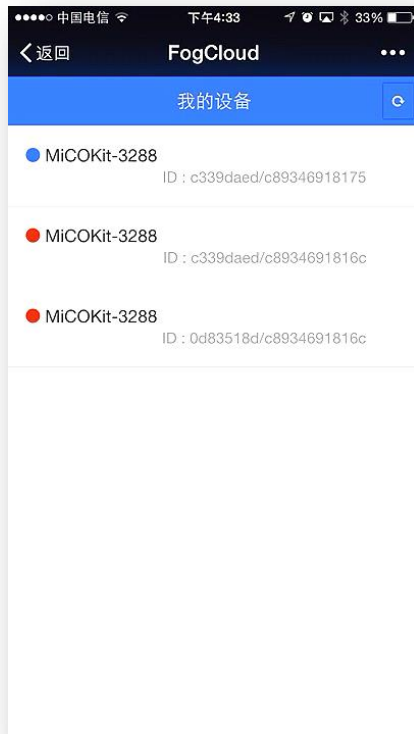
Airkiss 配网成功后会跳转到设备列表，红色圆点表示设备不在线，蓝色圆点表示设备在线。

注意：

- (1) 若配网成功后，没有自动跳转至设备列表页面，请确认步骤 4.9 节中设备页面 html 文件名已改为 FogCloud 中的产品 ID。
- (2) 若配网成功后，设备依然显示设备不在线，可点击右上角刷新按钮。

若配网成功后设备依然显示设备不在线，可点击右上角刷新按钮。点击列表进入设备控制界面，点击控制按钮，控制 MiCOKit-3288。

已经配网成功的设备不需要再次进行 Airkiss ,只需点击测试公众号中的 “Devices” 按钮 ,进入设备列表 , 点击列表进入设备控制界面。



如果您完成到此步骤，那么恭喜您通关啦！！

最简单的物联网设备已经被您开发出来了！！

没有完成也不要灰心，仔细参照本文检查之前的步骤，如果还有问题，请移步至 MiCO 社区 <http://mico.io>

Good Luck !

5. 版本更新

日期	修改人	版本	更新内容
2015-7-23	Bruce Li	V1.0	1. 初始版本
2015-7-29	Jenny Liu	V1.1	1. 增加 4.6 节中，“开通微信测试号的所有测试功能”说明 2. 增加 4.10 节中，“同步微信”时 result : 500 说明 3. 增加 4.11 节，“设备列表页面无跳转”注意事项
2015-9-7	Jenny Liu	V1.2	1. 更换 4.3 节中，“数据模型创建”图片内容 2. 删除对 MiCOKit 适用型号的指定 适用于所有型号 MiCOKit