MiCO Documentation Working Group (MDWG)
Track Number: 0008CN
Version: 1.2
Category: Technical Manual

Jenny Liu
MXCHIP Co., Ltd
2015.9.7

微信控制 RGB 灯开发实例

摘要（Abstract）

本文档仅介绍如何使用 **MiCOKit 开发套件**开发一个简单的，使用微信控制 RGB LED 灯的应用实例过程。

适用对象（Suitable Readers）

本实例适用于 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。

目 录

1. 概述

本文档仅介绍如何使用 **MiCOKit 开发套件**开发一个简单的使用微信控制 RGB LED 灯开关与亮度，颜色，饱和度的应用实例过程。

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（最新版本下载请至：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 RGB LED 灯的固件；

8. 用 IAR 或 Keil MDK 工具开发 MiCOKit 固件代码（代码注释部分）；

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

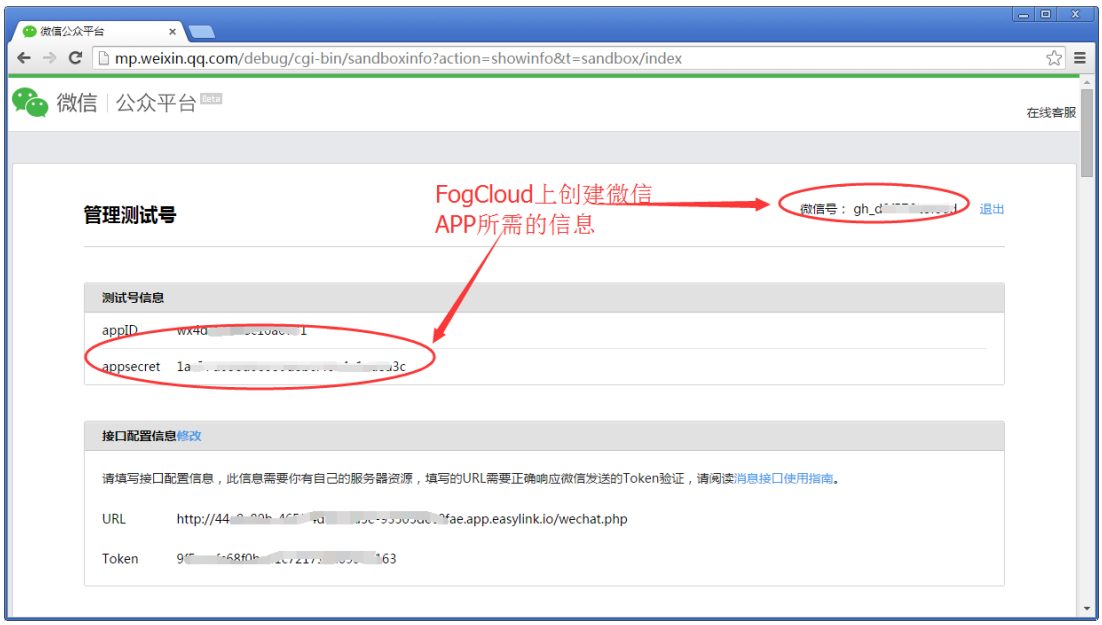
4.1. 注册开发者账号

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。

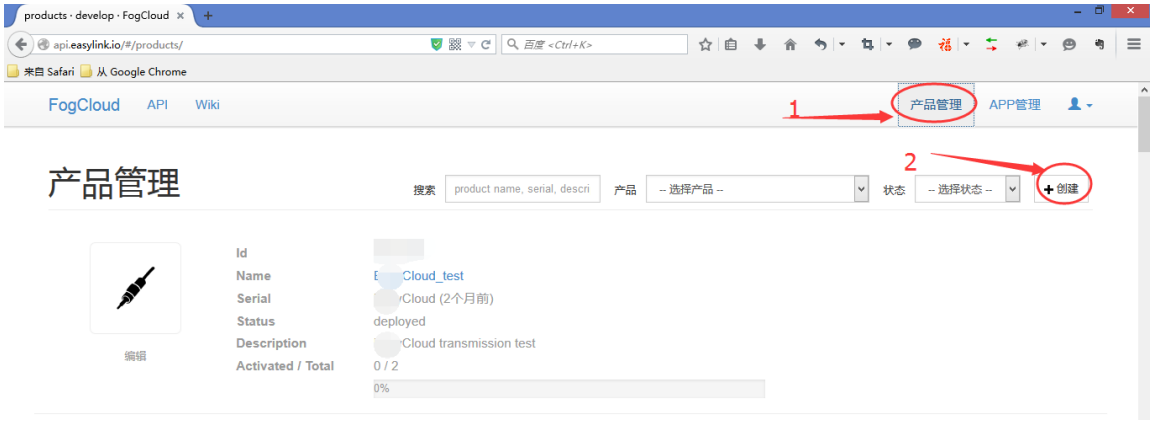
4.2. 使用个人微信号开通测试公众号

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。

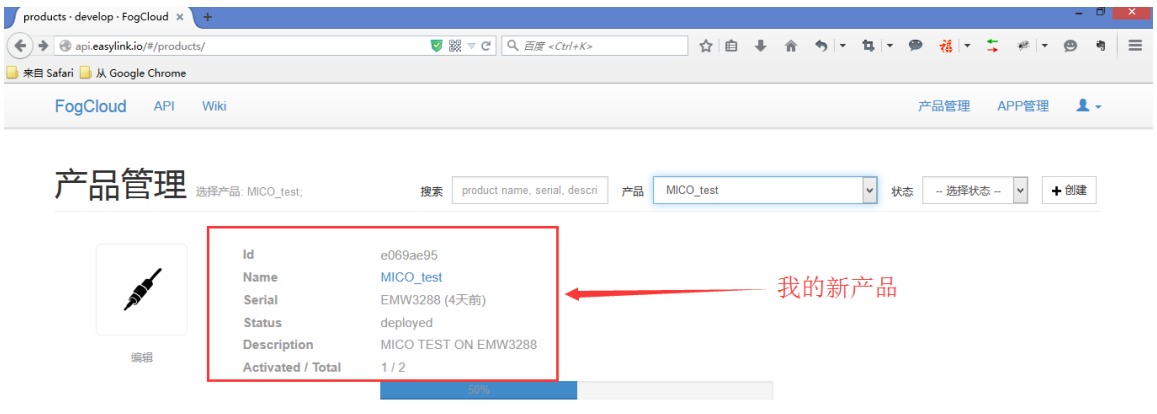
获得微信号、appID、appsecrect,用于后续在 FogCloud 上创建产品对应的微信 APP。



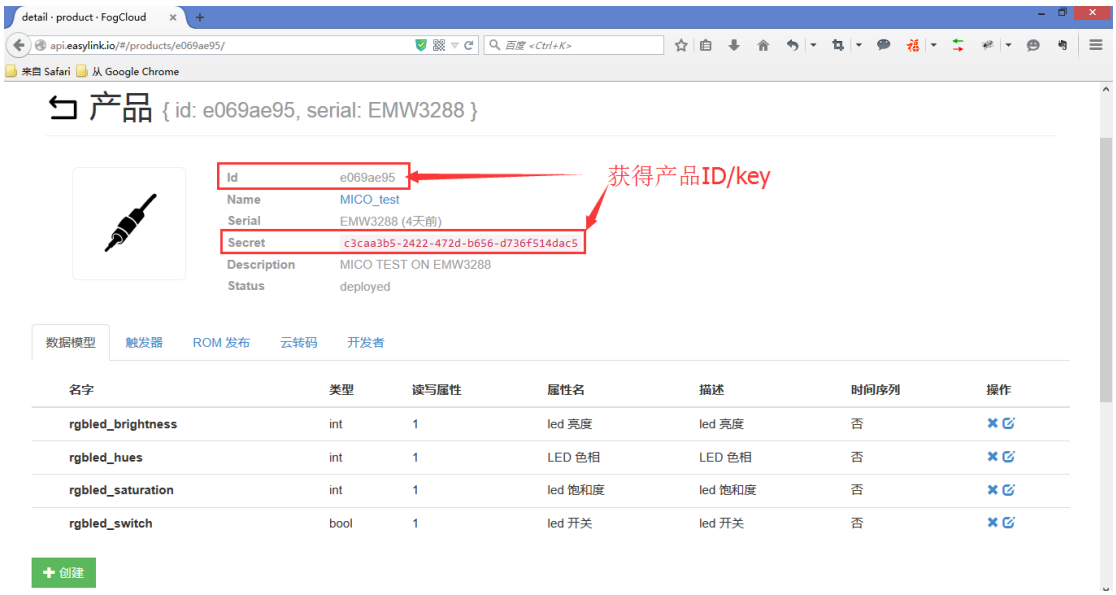
4.3. 在 FogCloud 上创建、定义自己的产品



根据提示填写相关信息，创建完成后，如下

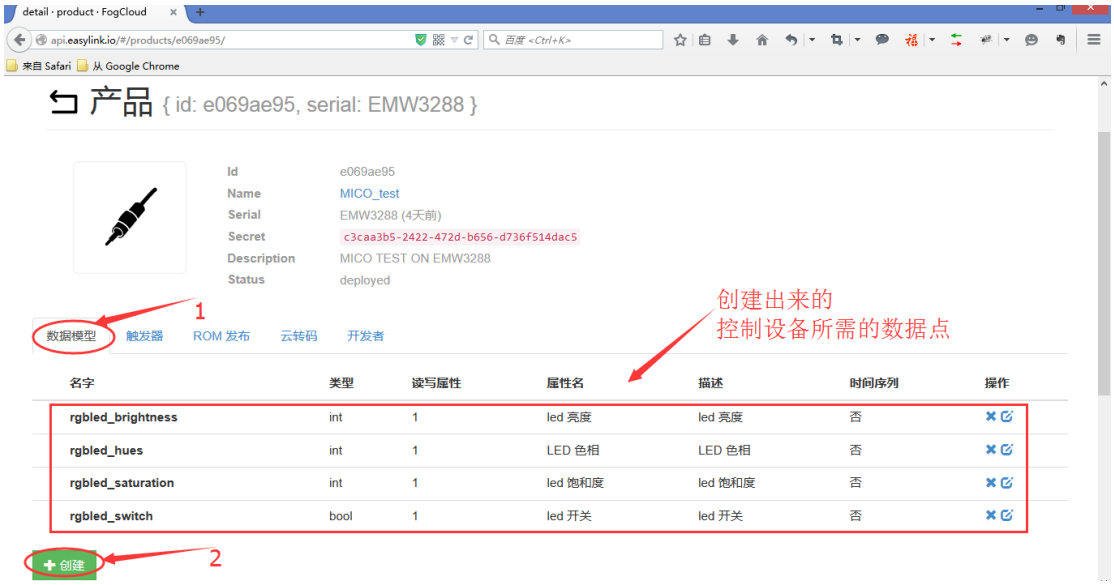


点击产品名称，进入详细信息：



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。



本实例仅控制 MiCOKit-3288 上的一个 RGB LED 灯，所需创建的控制数据点有：

1）开关（rgbled_switch） 读写属性为 1 时间序列为否

2）色相（rgbled_hues） 读写属性为 1 时间序列为否

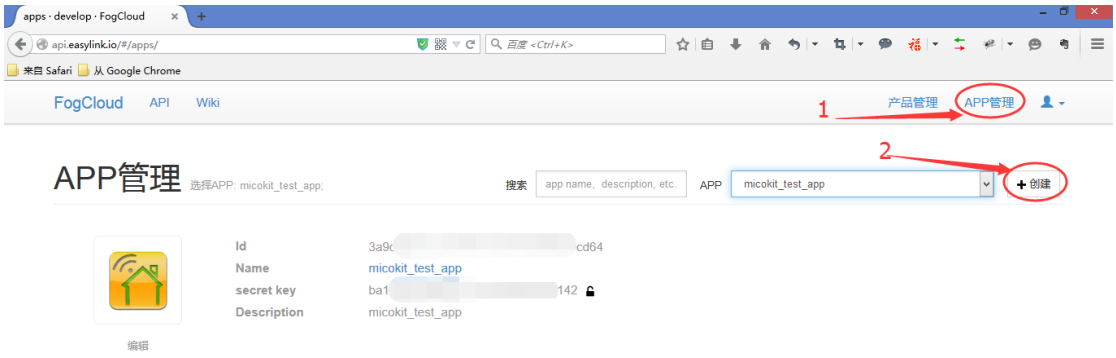
3）饱和度（rgbled_saturation） 读写属性为 1 时间序列为否

4）亮度（rgbled_brightness） 读写属性为 1 时间序列为否

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。

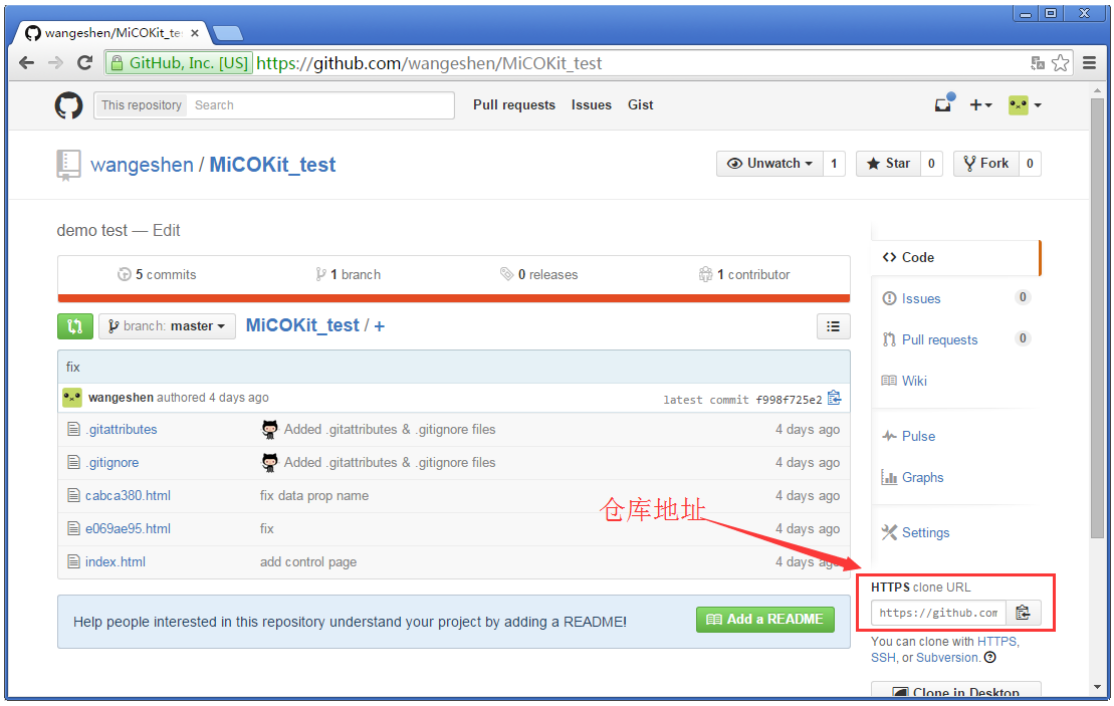
4.4. 在 FogCloud 上创建产品对应的微信 APP



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。

4.5. Github 上创建微信 APP 代码托管仓库

请登录 github.com 自行创建新仓库。并克隆到本地，克隆方法详见《上传文件到 GitHub》。该步骤的目的是获得一个可以在任何地方访问的 git 仓库，后面会使用该仓库托管微信 APP 的代码（其他类似 git 仓库托管工具也可以）。



获得仓库地址，例如：https://github.com/wangeshen/MiCOKit_test.git

4.6. 配置微信 APP 以及微信测试公众号

(a) FogCloud 上的微信 APP 信息：

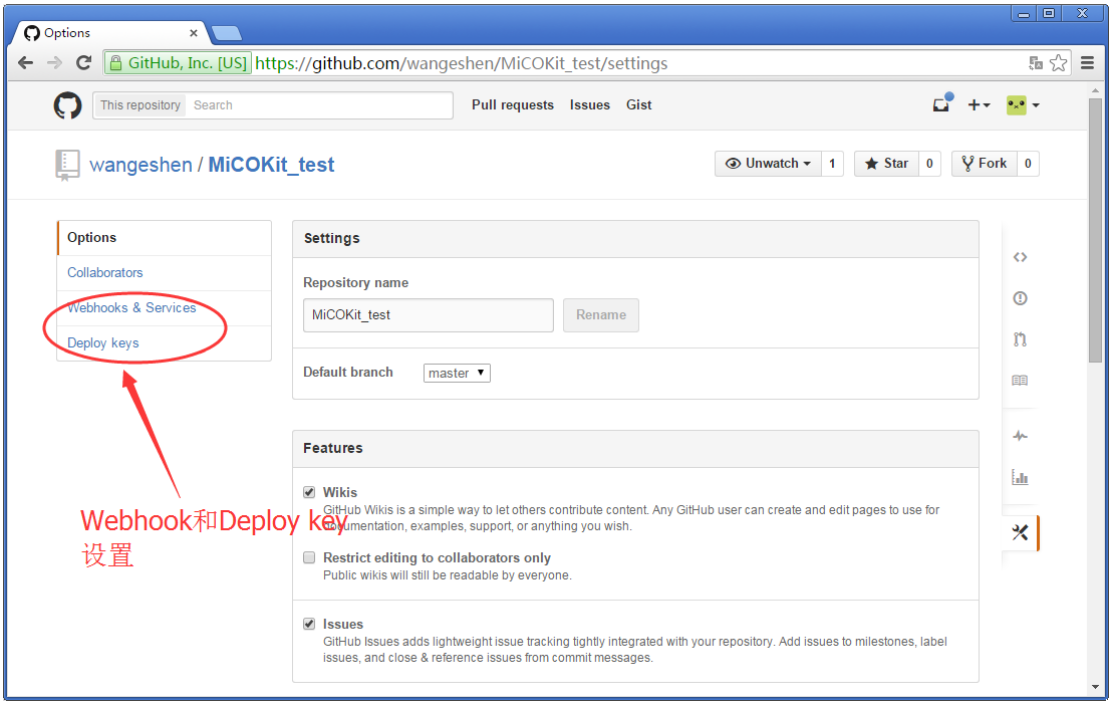
其中 URL 和 Token 会在后续配置微信测试号时用到。

(b) Git 部署（同步微信 APP 代码到 FogCloud）

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。



(c) 微信菜单管理

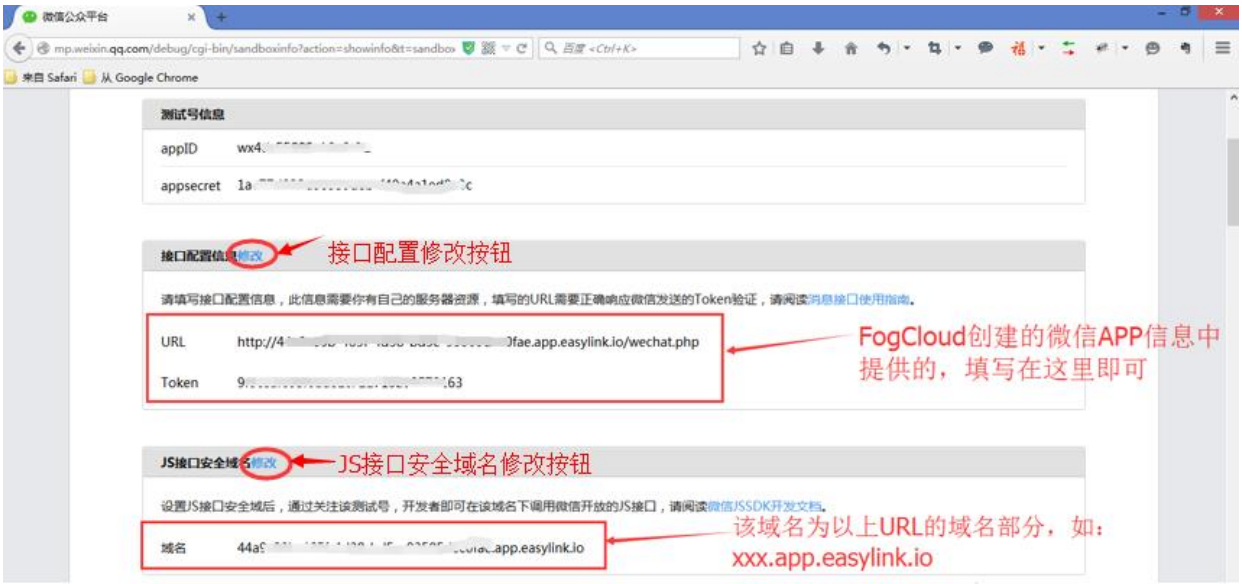
通过 FogCloud 提供的微信公众号首页菜单定制功能，方便的定制手机端微信上的控制界面及功能；至少包含“Airkiss”按钮，打开微信 Airkiss 配网功能，“OAuth”按钮（名称可自定义）进入设备控制。

(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 !!!**



“开启”微信测试号的所有测试功能，如下图 1,2,3：

点击 4，修改“网页账号”，打开如下：

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。

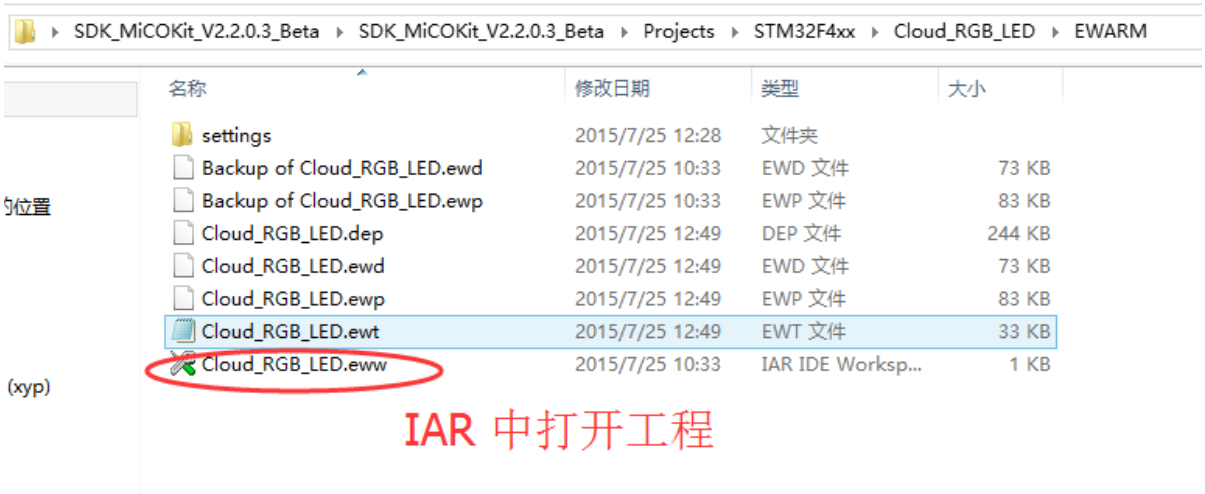
4.7. 使用 MiCO SDK 开发 RGB LED 灯的固件

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

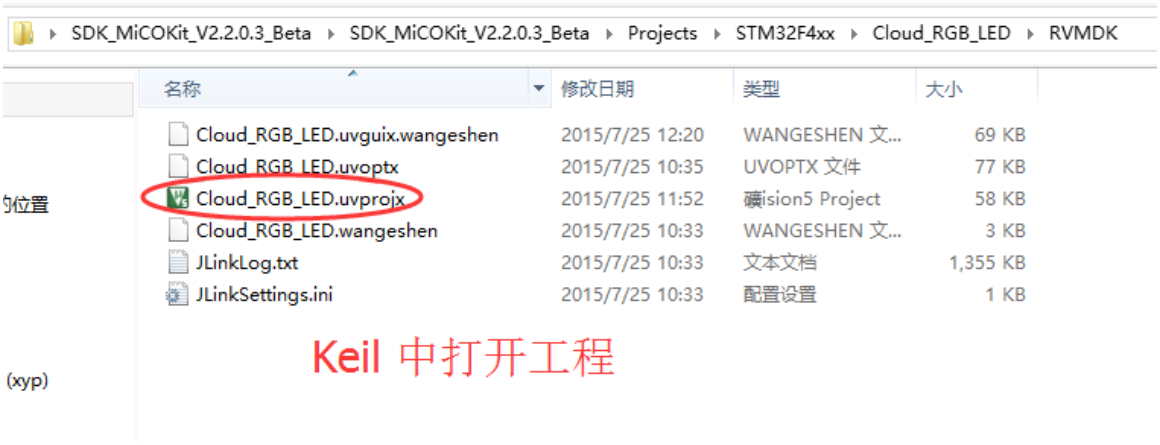
(b) 重新打开 mico.io 首页，开发者中心 ==> Wiki 中心，下载 MiCOKit SDK。

(c) 打开 MiCO SDK 中的微信开发实例工程：

IAR 中打开工程：

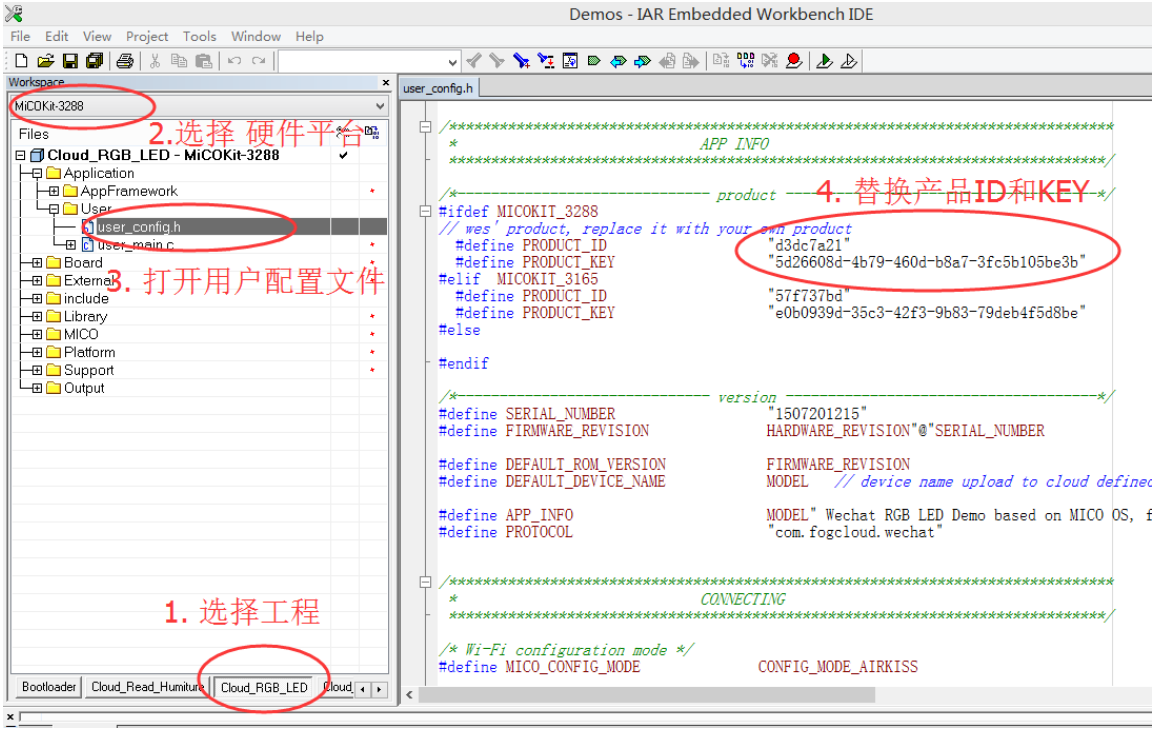


Keil 中打开工程：

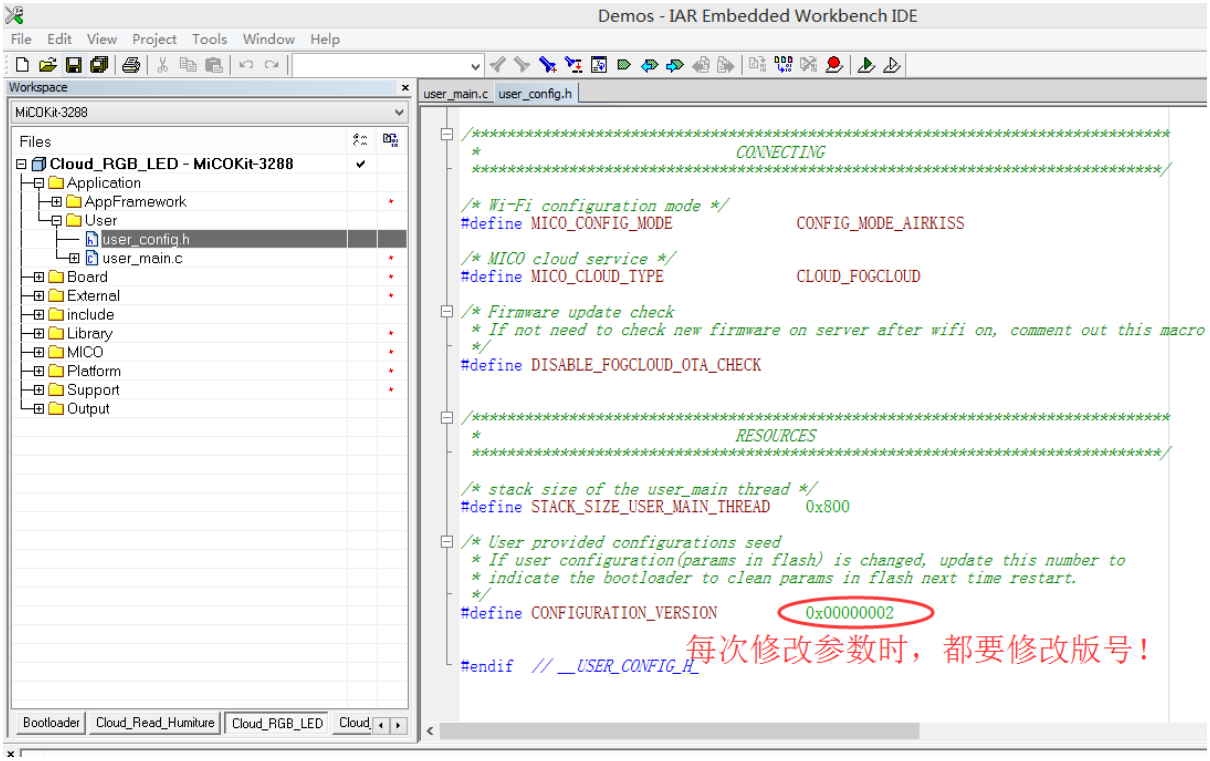


1）开发者手中拿到的可能是硬件平台可能是 MiCOKit3288 或者 MiCOKit3165，在编译工程时要**先选择硬件平台**；

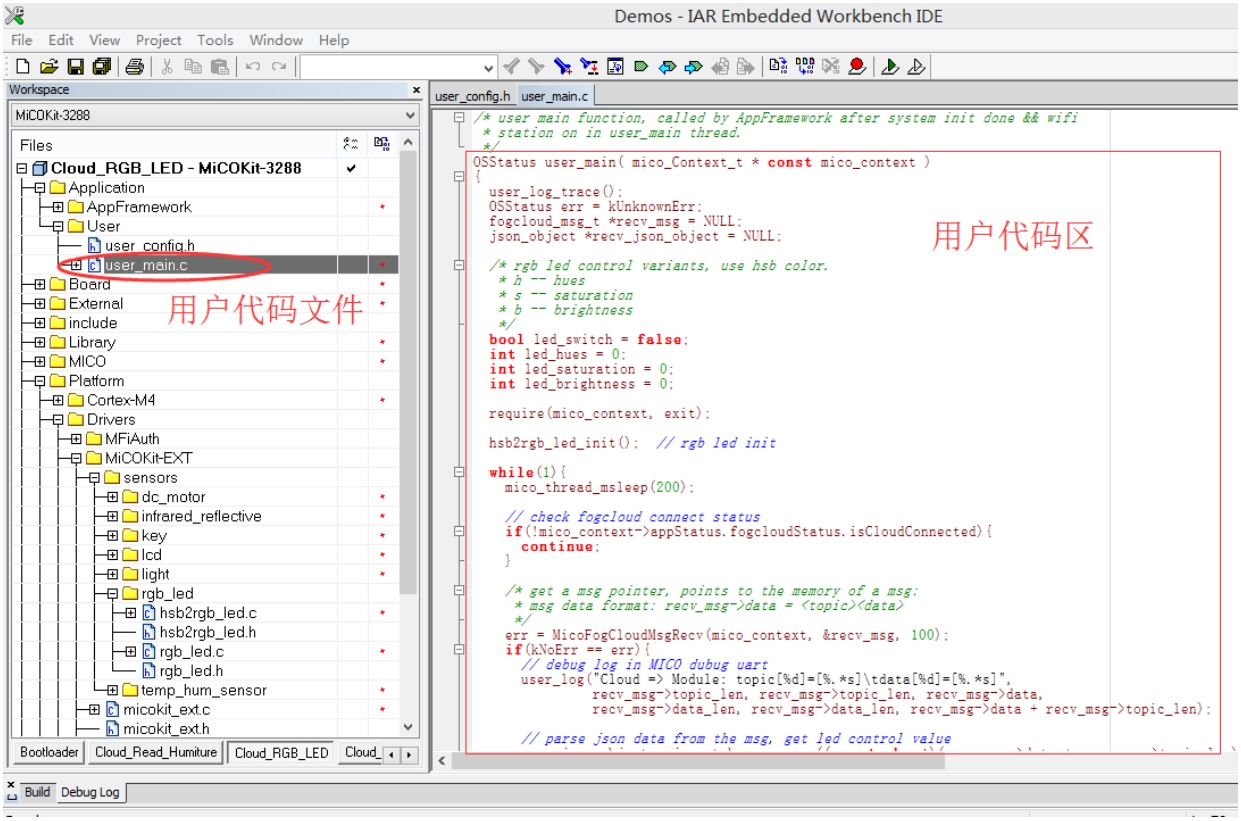
2）将 FogCloud 上创建的产品 ID/KEY 写入固件（**必须替换**）：



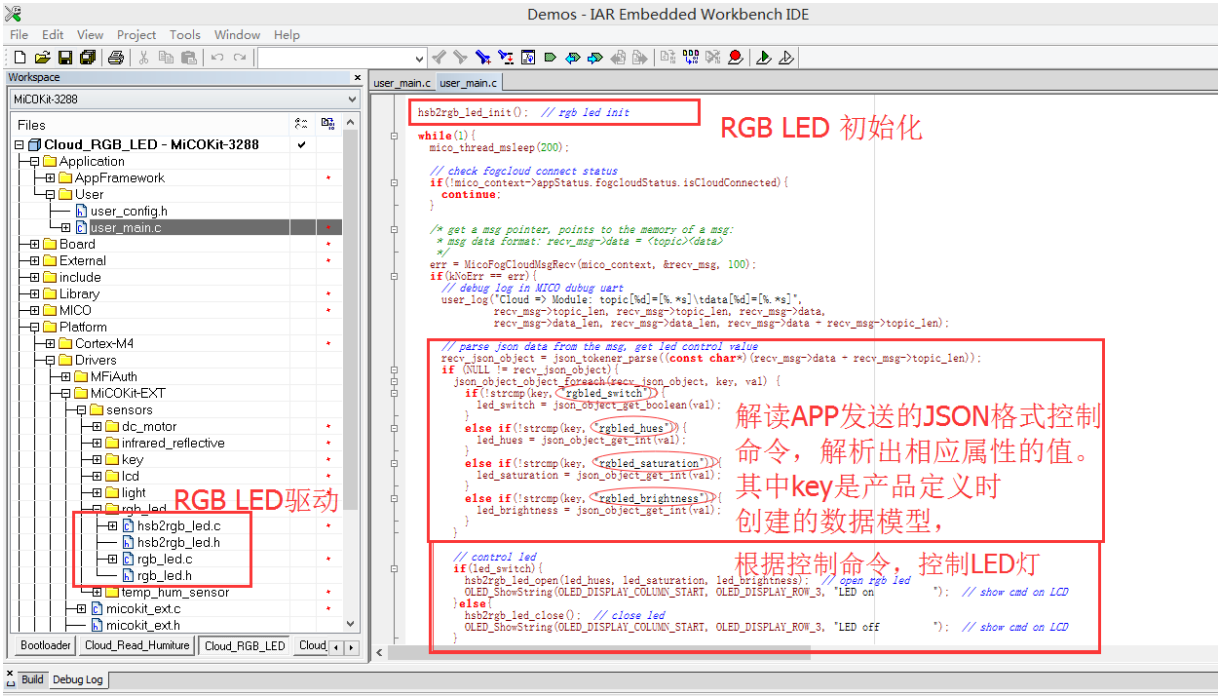
3）修改版本号，这样才能把新的产品 ID/KEY 烧入 FLASH



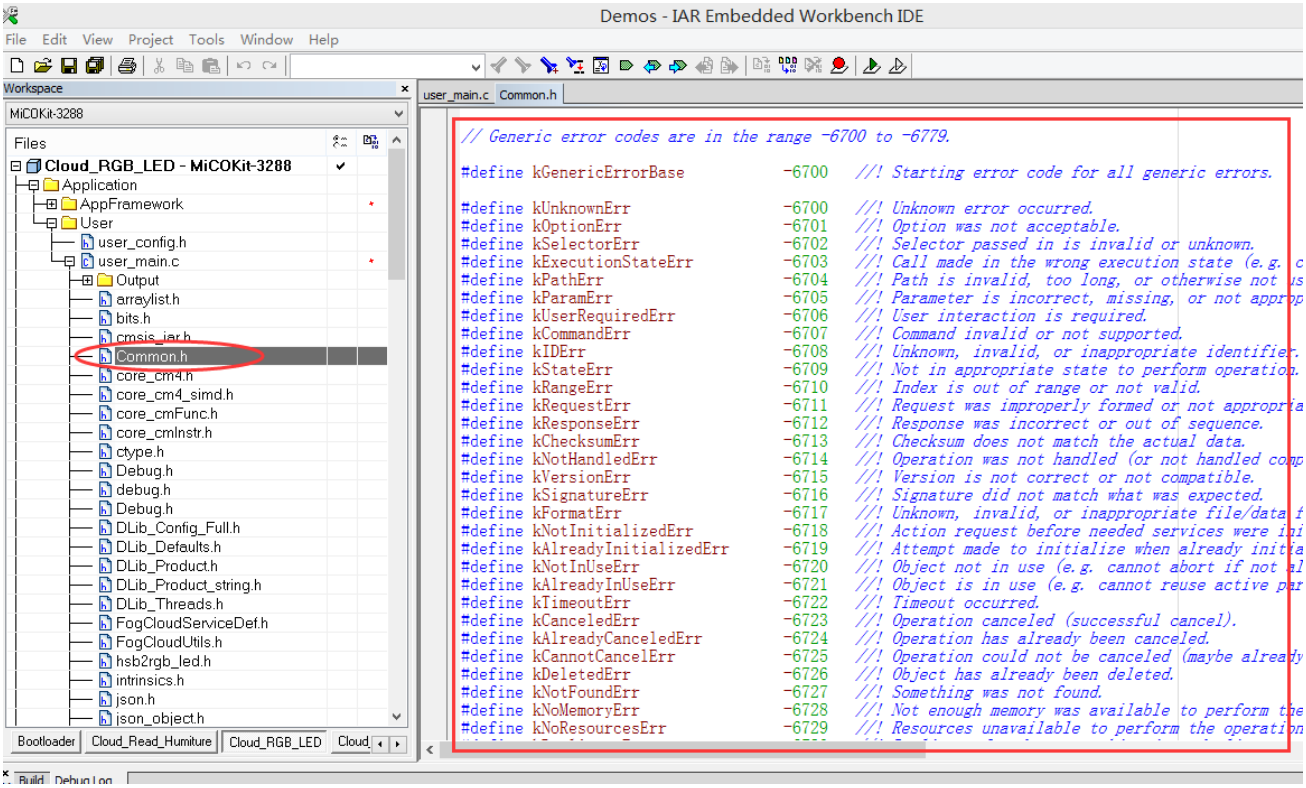
4）添加 LED 灯控制代码：



5）改为收到云端消息后，解析 JSON 数据，并控制 LED。



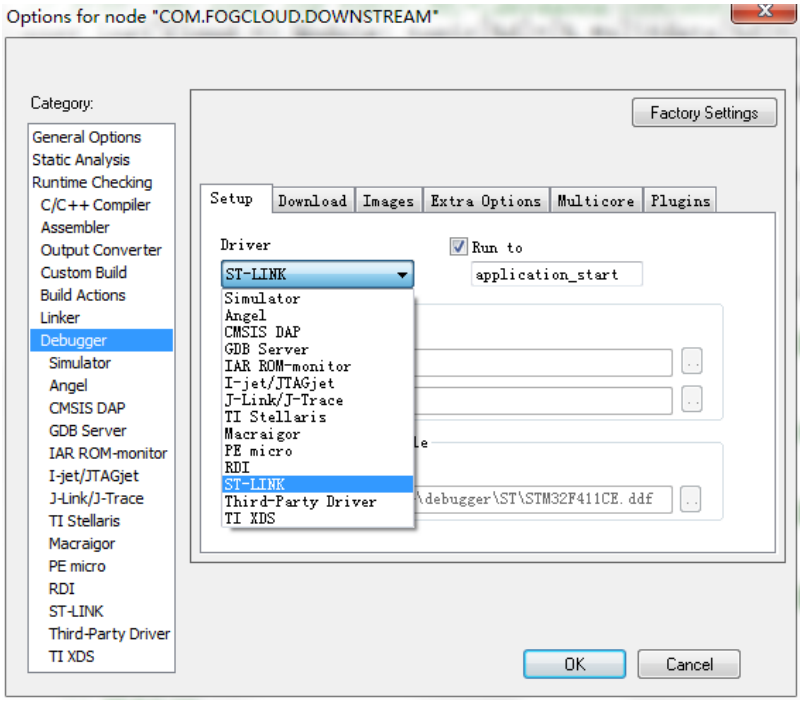
6）错误代码参考：在运行过程中如果出现错误，可在 user_log 中查看错误代码。



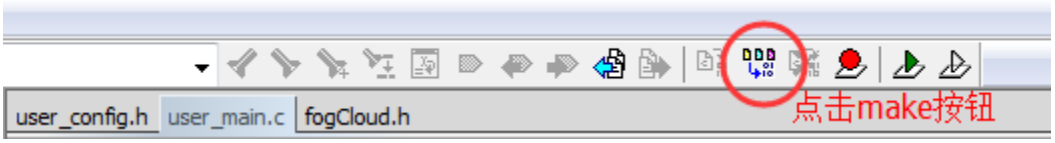
```
-6700   //! Unknown error occurred.
-6701   //! Option was not acceptable.
-6702   //! Selector passed in is invalid or unknown.
-6703   //! Call made in the wrong execution state (e.g. called at interrupt time).
-6704   //! Path is invalid, too long, or otherwise not usable.
-6705   //! Parameter 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/doku.php?id=debug ）

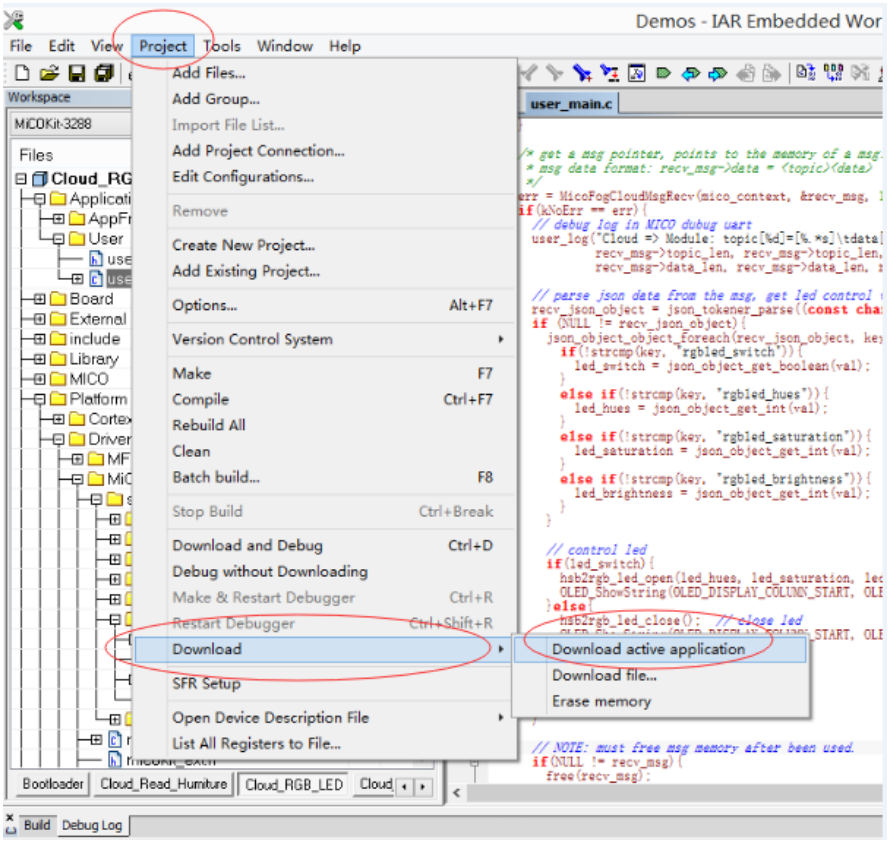
选择使用的烧录工具 J-Link 或者 ST-LINK:



8）编译/连接:



9) 烧录/下载：



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

```
//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;

  fogcloud_msg_t  *recv_msg = NULL;//接收数据结构体

  json_object  *recv_json_object  = NULL;//结构体内包含 json 数据格式

  /* rgb 灯的色彩模式采取 hsb 色彩模式（色相、饱和度、亮度）  */

  bool led_switch  = false;//rgb 灯开关

  int led_hues  = 0;
```

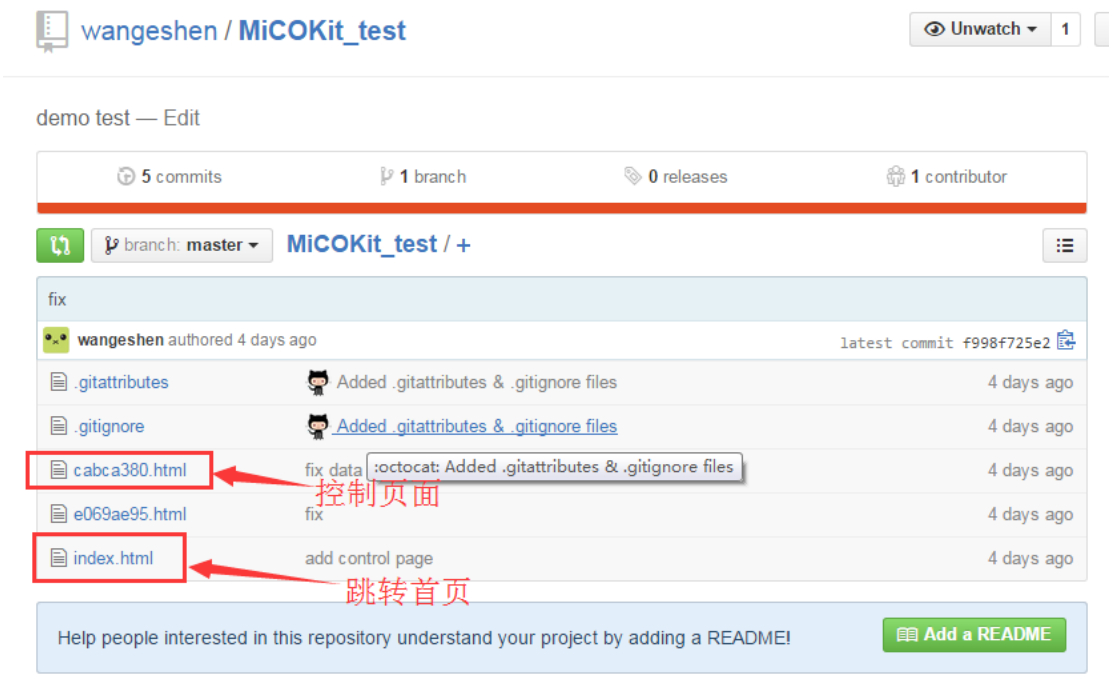
```
    int led_saturation  = 0;
    int led_brightness  = 0;
    require(mico_context,  exit);
    hsb2rgb_led_init();    // RGB_LED 初始化
    while(1){
      mico_thread_msleep(100);  //延时 100ms
      // 检测 fogcloud 连接状态
      if(!mico_context->appStatus.fogcloudStatus.isCloudConnected){
        continue;
      }
      /*
        接收来自云端的数据
        recv_msg->data = <topic>+<data>
        topic 表示模块订阅的 MQTT 通道名，data 表示云端返回的数据
       */
      err = MicoFogCloudMsgRecv(mico_context,  &recv_msg,  100);
      if(kNoErr == err){
// 打印  例：
//user_logtopic[30]=[de54a8ea/c8934691816b/in/write]data[23]=[{"rgbled_switch":false}]
        user_log("Cloud  => Module:  topic[%d]=[%.*s]\tdata[%d]=[%.*s]",
                 recv_msg->topic_len,  recv_msg->topic_len,  recv_msg->data,
                   recv_msg->data_len,  recv_msg->data_len,  recv_msg->data  +
                   recv_msg->topic_len);
// json 格式的字符串数据转成 json 对象
```

```
recv_json_object = json_tokener_parse((const char*)(recv_msg->data + recv_msg->topic_len));
if (NULL != recv_json_object)
{
  //根据键值对遍历
  json_object_object_foreach(recv_json_object, key, val) {
    if(!strcmp(key, "rgbled_switch")){
      //如果键为"rgbled_switch"，提取值“开关值”
      led_switch = json_object_get_boolean(val);
    }
    else if(!strcmp(key, "rgbled_hues")){
      //如果键为"rgbled_hues"，提取值“色相值"
      led_hues = json_object_get_int(val);
    }
    else if(!strcmp(key, "rgbled_saturation")){
      //如果键为"rgbled_saturation"，提取值“饱和度值"
      led_saturation = json_object_get_int(val);
    }
    else if(!strcmp(key, "rgbled_brightness")){
      //如果键为"rgbled_brightness"，提取值“亮度值"
      led_brightness = json_object_get_int(val);
    }
  }
  // 控制 LED
```

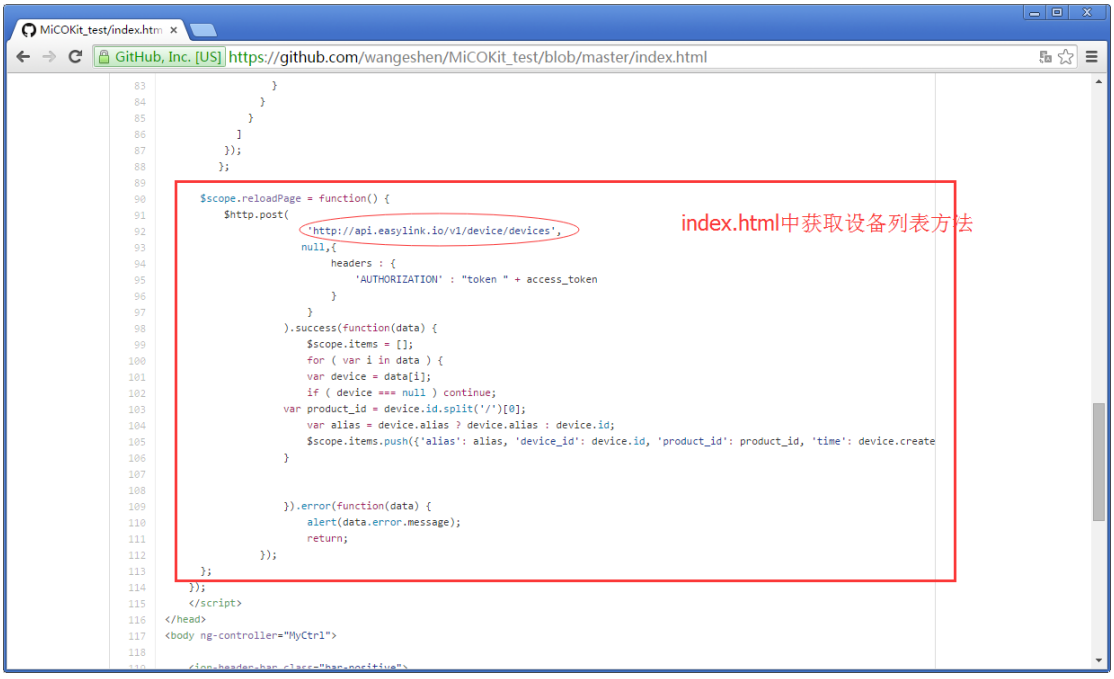
```
            if(led_switch){
                hsb2rgb_led_open(led_hues, led_saturation, led_brightness);
            }else{
                hsb2rgb_led_close();    // 关闭
            }
            // json 对象内存释放
            json_object_put(recv_json_object);
            recv_json_object = NULL;
        }
        // 结构体内存释放
        if(NULL != recv_msg){
            free(recv_msg);
            recv_msg = 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

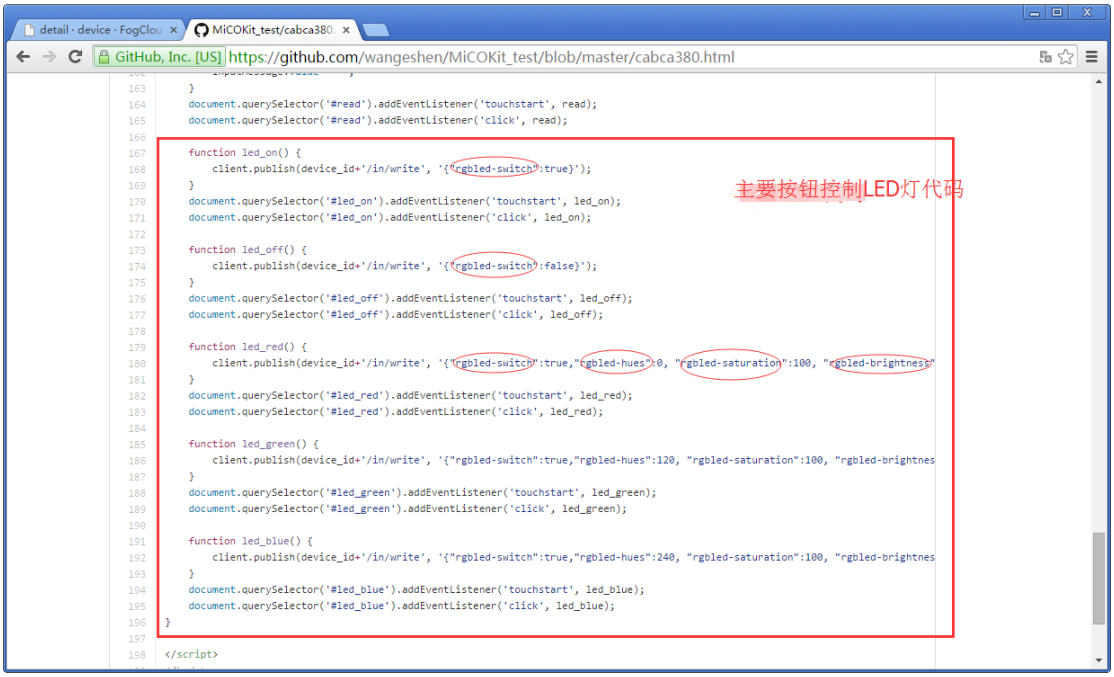
在例程包中(MiCOKit SDK 的 APP 目录下) 找到微信 APP 控制页面代码(index.html 和 yourID.html), 将 yourID.html.重命名为你的产品 ID，如 cabca380.html。拷贝这两个文件到本地 git 仓库，再同步到 git 服务器。同步方法详见《上传文件到 GitHub》。



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



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：

```
<script>
    // 从url中获取某个参数的值
    function getParameterByName(name) {
        var match = RegExp('[?&]' + name + '=([^&]*)').exec(window.location.search);
        return match && decodeURIComponent(match[1].replace(/\+/g, ' '));
    }
    // 得到设备ID
    var device_id = getParameterByName('device_id');
    // 如果设备ID不为空，则执行连接MQTT的操作
    if ( device_id !== null ){
        ez_connect(device_id);
    }
    // 连接MQTT服务
    function ez_connect(device_id) {
        // 获取access_token
        // 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。
        // 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
        var access_token = getParameterByName('access_token');

        document.getElementById('device_id').innerHTML = device_id;

        // websocket连接
        // wsbroker:host
        // wsport:端口 默认1983
        // Client-ID ：  v1_web_[MAC]  //版本号_app_手机MAC(必须是12位小写)
        var wsbroker = "api.easylink.io";  //mqtt websocket enabled broker
        var wsport = 1983 // port for above
        var client = new Paho.MQTT.Client(wsbroker, wsport, "v1-web-" + parseInt(Math.random() * 1000000, 12));

        // 基本参数配置
        // 连接丢失所对应的callback函数
        client.onConnectionLost = onConnectionLost;
        // 消息到达所对应的callback函数
        client.onMessageArrived = onMessageArrived;
        // 连接成功所对应的callback函数
        client.connect({onSuccess:onConnect});
```

```
// 连接成功
function onConnect() {
    var subtopic = device_id+'/out/#';
    // Once a connection has been made, make a subscription and send a message.
    // 向某个通道发送指令
    // topic：通道
    // commond：指令
    client.publish = function(topic, commond) {
        console.log("现在执行-->:"+commond);
        message = new Paho.MQTT.Message(commond);
        message.destinationName = topic;
        client.send(message);
    }
    console.log("device_id:"+device_id);
    console.log("onConnect");
    client.subscribe(subtopic, {qos: 0});
}
// 连接丢失
function onConnectionLost(responseObject) {
    if (responseObject.errorCode !== 0)
        console.log("onConnectionLost:"+responseObject.errorMessage);
}
// 消息到达
function onMessageArrived(message) {
    console.log(message.topic + ': ' +  message.payloadString);
}

// 串口数据发送部分
var inputMessage = document.getElementById('message');
// 将消息发送到指定的通道
function send2uart() {
    if ( inputMessage.value.length == 0 ) return;
    var topic = device_id+'/in/write';
    var commond = '{"11":"' + inputMessage.value + '"}';
    client.publish(topic, commond);
    console.log(inputMessage.value);
    inputMessage.value = '';
}
```

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

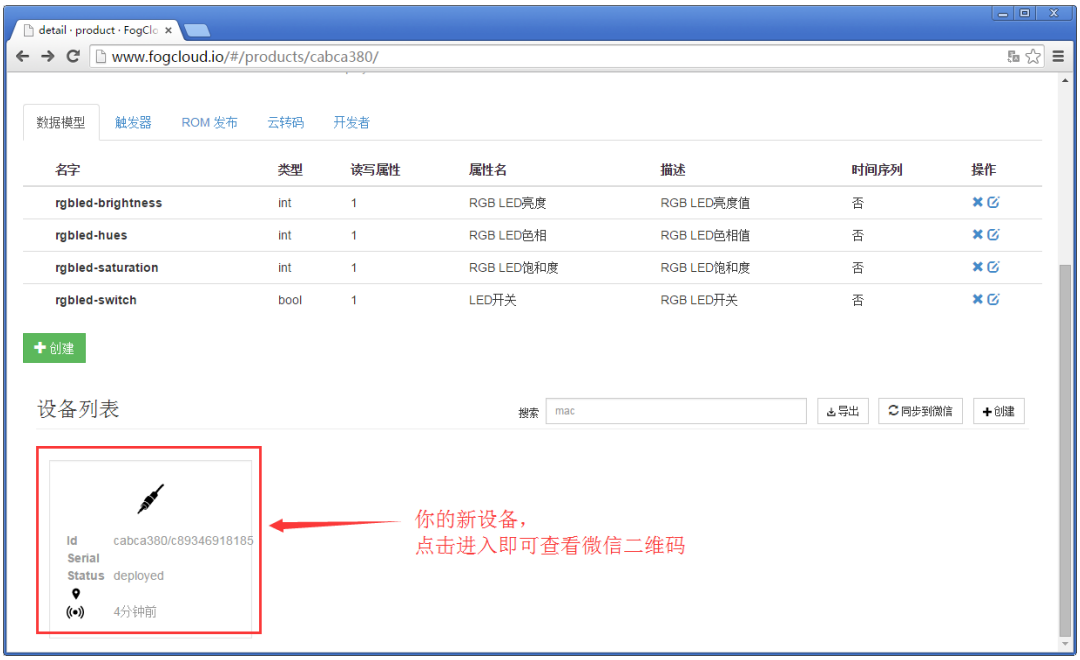
{"result":200,"message":"success"}

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 · device · FogClou
www.fogcloud.io/#/products/cabca380/devices/cabca380_c89346918185/
Device Secret
Master Device Key
Device Status
Bssid c89346918185
设备二维码
设备二维码
MiCOKit-test
micokit_test_app
WES_MICOKIT_WECHAT_APP
/in
ut
message
message
发送
发送
请求日志
请求日志

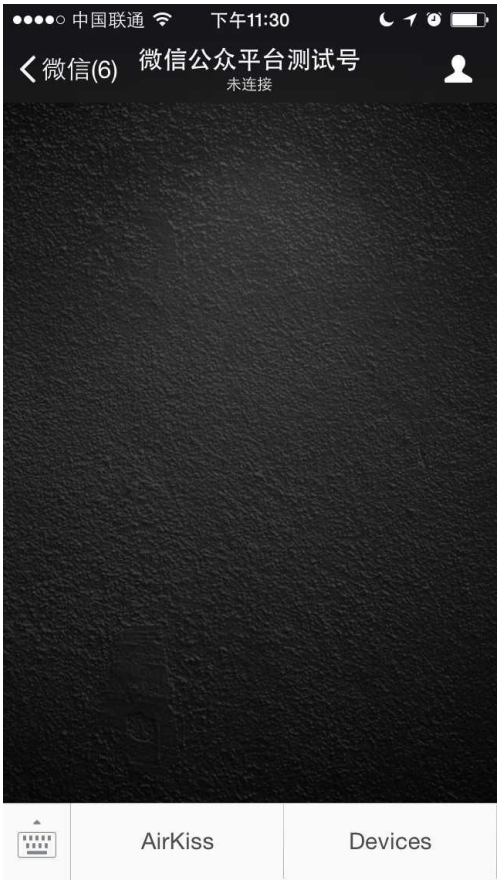
4.11. 使用手机微信扫码，测试“Airkiss”配网功能以及设备控制功能

(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。

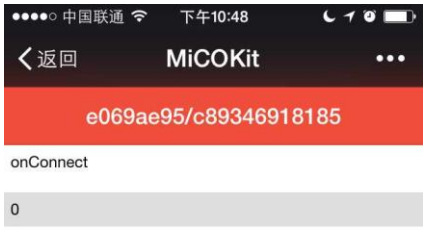
注意：

（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。

点击列表进入设备控制界面，点击控制按钮，控制 MiCOKit-3288 扩展板上的 RGB LED 灯。

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，点按按钮，控制 MiCOKit-3288 扩展板上的 RGB LED 灯。



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. 版本更新说明

日期	版本	更新内容
2015-7-22	V1.0	1. 初始版本
2015-7-29	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”说明。
2015-9-7	V1.2	1. 删除对适用型号的指定内容，本文适用于所有型号 MiCOKit