MXCHIP® | 智能硬件解决方案提供商

MiCO Document Working Group (MDWG) Jenny Liu

Track Number: 0010CN MXCHIP Co., Ltd

Version: 1.1 2015.9.7

Category: Technical Manual

MiCOKit 微信控制综合开发实例

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，使用微信控制的综合实例过程。

适合读者（Suitable Readers）

本文适用于初级 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。零基础读者请先参考《“微信控制 RGB LED 灯”开发实例》《“微信读取温湿度传感器数值”开发实例》。

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）

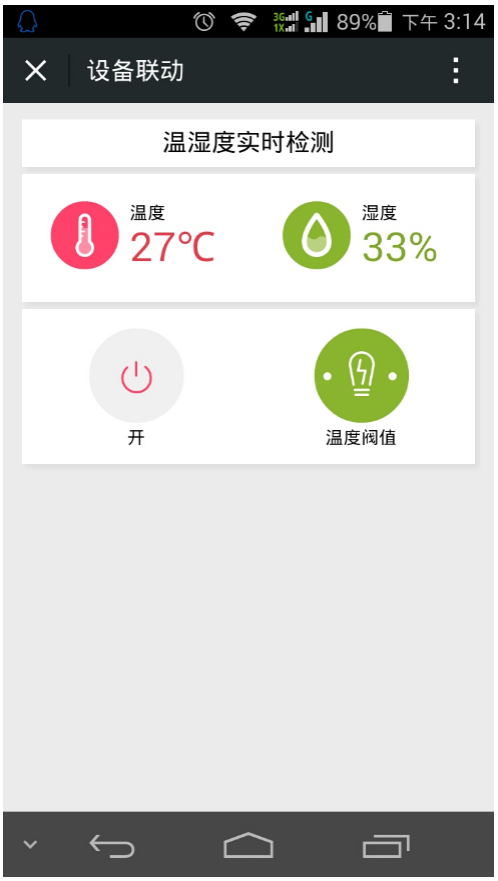
目录

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微信控制综合实例过程。实现功能：

微信端显示 MiCOKit 发上来的温湿度数据，也能控制 LED 灯的开关。在温度为 20 度以下时，LED 灯成黄色；在温度为 20 到 30 度时，LED 灯成绿色；在温度为 20 度以下时，LED 灯成红色。

效果图如下：



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_V2.2.0.4_Beta（最新版本至：http://mico.io/wiki/doku.php?id=micokit_sdk））；
5. 个人微信号（用来开通“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. 使用综合例程的固件；
8. 用 IAR 或 Keil MDK 工具开发 MiCOKit 固件（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

本例程为综合实例，针对一些步骤的详细信息，“ 零基础读者 “请参考《“微信控制 RGB LED 灯” 开发实例教程》或《“微信读取温湿度传感器数值” 开发实例教程》

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

1. 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。
2. 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

1. 点击产品名称，进入详细信息（产品 ID/KEY 将在 4.7 节被写入设备固件代码中）
2. 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
Hum	int	0	湿度	湿度	否	
rgbled_brigtness	int	1	亮度	亮度	否	
rgbled_hues	int	1	色相	色相	否	
rgbled_saturation	int	1	保护度	保护度	否	
rgbled_switch	bool	1	开关	RGBLED开关	否	
Temp	int	0	温度	温度	否	

本实例所需创建的控制数据点有（属性名和描述可按个人习惯添加）：

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

5）温度（Temp） 读写属性为 0 时间序列为否

6）湿度（Hum） 读写属性为 0 时间序列为否

7）设备开关（device_switch） 读写属性为 1 时间序列为否

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 4.2 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）

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 !!!**

开通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， 如下图：

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。

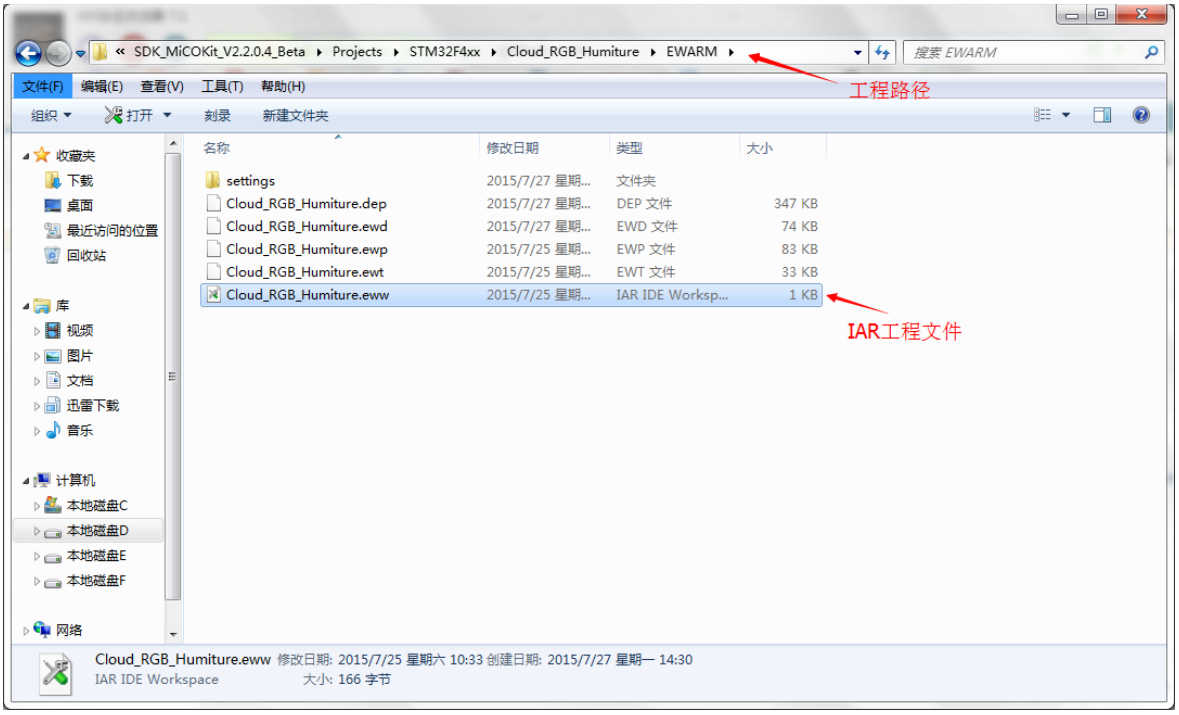
4.7. 使用 MiCOKit SDK 开发综合例程的固件

(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 MiCOKit SDK 中的综合例程开发实例工程：

...\SDK_MiCOKit_V2.2.0.4_Beta\Projects\STM32F4xx\Cloud_RGB_Humiture\EWARM\Cloud_RGB_Humiture.eww



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

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）添加固件代码；

5）MiCOKit 与云端上下通信，以 JSON 格式解析消息，并执行相应操作；

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；

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→编译/连接→烧录/下载。

4.8. 用 IAR 或 Keil MDK 工具开发 MiCOKit 固件（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;

  char oled_show_line[OLED_DISPLAY_MAX_CHAR_PER_ROW+1] = {'\0'};

  require(mico_context, exit);


  hsb2rgb_led_init();    // 初始化 RBG_LED 灯

  hsb2rgb_led_close();  // 关闭 RBG_LED 灯

  // 开启 downstream 线程用来处理用户指令

  Err = mico_rtos_create_thread(&user_downstrem_thread_handle，MICO_APPLICATION_PRIORITY,
  "user_downstream",user_downstream_thread,STACK_SIZE_USER_DOWNSTREAM_THREAD,mico_context );

  //参数检查

  require_noerr_action( err, exit, user_log("ERROR: create user_downstream thread failed!") );

  // 开启 upstream 线程用来发送温湿度数据
  Err=mico_rtos_create_thread(&user_upstream_thread_handle,MICO_APPLICATION_PRIORITY,"user_upstream",
  user_upstream_thread,STACK_SIZE_USER_UPSTREAM_THREAD,mico_context );

  //参数检查

  require_noerr_action( err, exit, user_log("ERROR: create user_uptream thread failed!") );

   // 用户主循环，每秒更新一次 oled 显示

  while(1){

    mico_thread_msleep(500);//小睡 500ms

    // 设备开关变化

    if(device_switch_changed){
```

```
    device_switch_changed = false;
    //设备关
    if(!device_switch){
      // 挂机 upstream 线程
      mico_rtos_suspend_thread(&user_upstream_thread_handle);
      hsb2rgb_led_close();//关闭 LED
      OLED_Display_Off();//关闭 OLED
    }
    //设备开
    else{
      // 继续 upstream 线程
      mico_rtos_resume_thread(&user_upstream_thread_handle);
      // 开启 OLED
      OLED_Display_On();
    }
}
// 设备开, 更新 OLED 和 rgbled 状态
if(device_switch){
// 更新 OLED 2~4 行
OLED_ShowString(OLED_DISPLAY_COLUMN_START, OLED_DISPLAY_ROW_2, "Demo H/T &&
LED");memset(oled_show_line,'\0',OLED_DISPLAY_MAX_CHAR_PER_ROW+1);
 snprintf(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);
        // 控制 RBG_LED 灯
        if(rgbled_changed){
        rgbled_changed = false;
        if(rgbled_switch){
        // 打开 RBG_LED 灯
        hsb2rgb_led_open(rgbled_hues, rgbled_saturation, rgbled_brightness);
            }else{
              // 关闭 RBG_LED 灯
              hsb2rgb_led_close();
            }
          }
       }
    }
exit:
    if(kNoErr != err){
       user_log("ERROR: user_main thread exit with err=%d", err);
    }
    mico_rtos_delete_thread(NULL);   // 删除当前线程
    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 xxxxx.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（**将 xxxx 替换成 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')
        // 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
    // var msgindex = 1;
    var msgjson;
    function onMessageArrived(message) {
        msgjson = $.parseJSON(message.payloadString);
        var tempval = msgjson.dht11_temperature;
        if("undefined" != typeof(tempval)){
            // 显示温度
            $("#tempid").text(tempval+"℃");
            // 显示湿度
            $("#humiid").text(msgjson.dht11_humidity+'%');
            // 温度超过 29 则红灯亮，20-29 之间则绿灯亮，低于 20 黄灯亮
            if(tempval > "29"){
```

```
                led_red();
                $("#rgbimgid").attr("src", "./image/02-red.svg");
            }else if(tempval < "30" && (tempval > "19")){
                led_green();
                $("#rgbimgid").attr("src", "./image/02-green.svg");
            }else if(tempval < "20"){
                led_yellow();
                $("#rgbimgid").attr("src", "./image/02-yellow.svg");
            }
        }
    }
    // 控制按钮显示的事件
    $("#switchbtn").click(function() {
        var topic = device_id+'/in';
        var commond;
        var swtimg = $("#switchbtn").attr("src");
        if("./image/03-turn-on.svg" == swtimg){
            commond = '{"device_switch":false}';
            $("#switchbtn").attr("src", "./image/03-turn-off.svg");
            $("#switchspan").text("关");
        }else{
            commond = '{"device_switch":true}';
            $("#switchbtn").attr("src", "./image/03-turn-on.svg");
```

```
                $("#switchspan").text("开");
            }
            client.publish(topic, commond);
        });
        function led_red() {
            var topic = device_id+'/in';
            var commond = '{"rgbled_switch":true,"rgbled_hues":0, "rgbled_saturation":100, "rgbled_brightness":100}';
            client.publish(topic, commond);
        }
        function led_green() {
            var topic = device_id+'/in';
            var commond = '{"rgbled_switch":true,"rgbled_hues":120, "rgbled_saturation":100, "rgbled_brightness":100}';
            client.publish(topic, commond);
        }
        function led_yellow() {
            var topic = device_id+'/in';
            var commond = '{"rgbled_switch":true,"rgbled_hues":60, "rgbled_saturation":100, "rgbled_brightness":100}';
            client.publish(topic, commond);
        }
    }
    </script>
```

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

（1）创建新设备

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。

（2）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"}

返回到产品界面，可看到你的新设备。

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。

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。

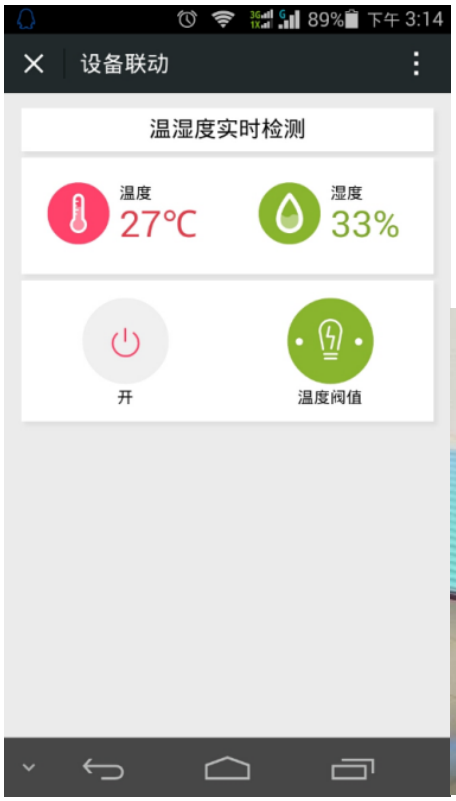
注意：

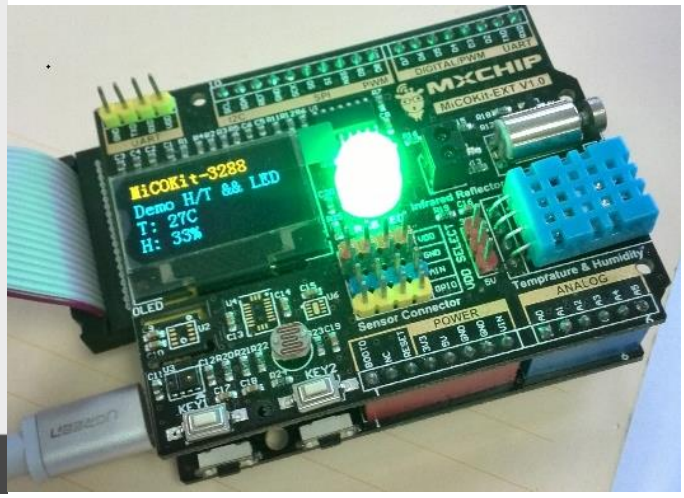
（1）若配网成功后，没有自动跳转至设备列表页面，请确认步骤 4.9 节中设备页面 html 文件名已改为 FogCloud 中的产品 ID。

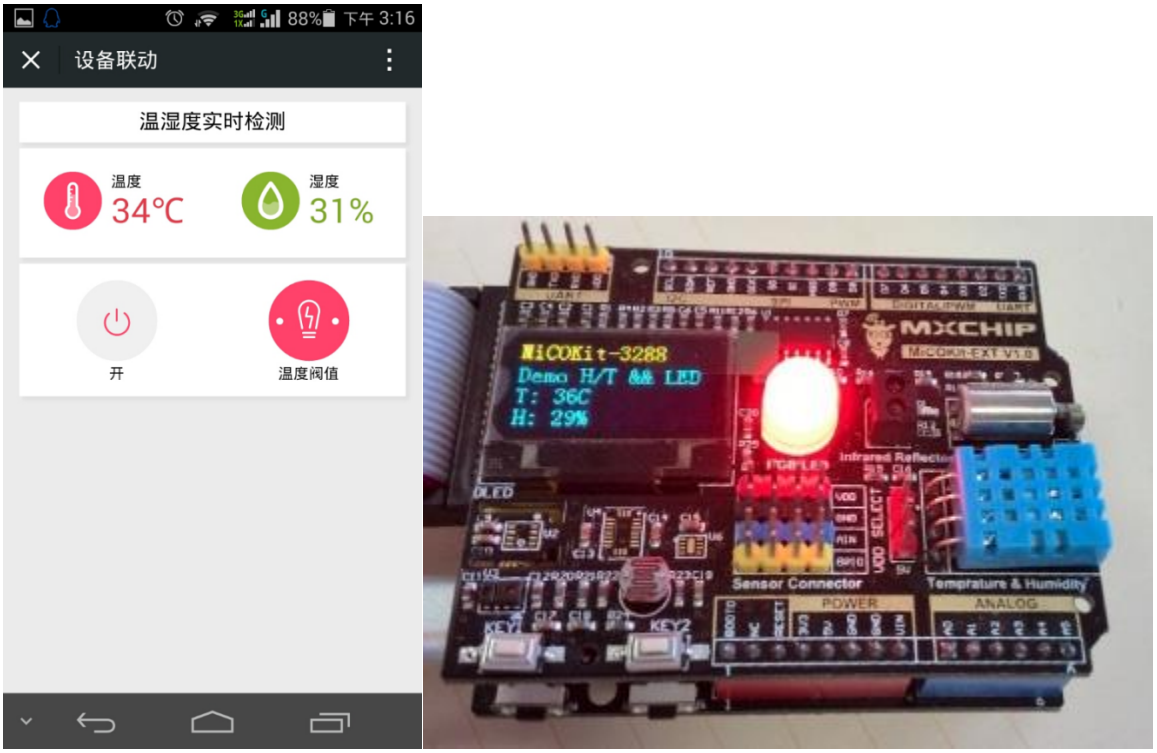
（2）若配网成功后，设备依然显示设备不在线，可点击右上角刷新按钮。

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，点击设备进入设备控制界面。此时能收到 MiCOKit 发上来的温湿度数据，也能控制 LED 的开关。在温度为 20 度以下时，LED 灯成黄色；在温度为 20 到 30 度时，LED 灯成绿色；在温度为 20 度以下时，LED 灯成红色。







如果您完成到此步骤，那么恭喜您又通关啦 !!

通过本案例主要向大家展示了 MiCOKit 上下行通信的功能。

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	修改人	版本	更新内容
2015-7-28	Tom Hong	V1.0	1. 初始版本
2015-9-7	Jenny Liu	V1.1	1. 删除对适用型号的指定内容 ,适用于所有型号 MiCOKit