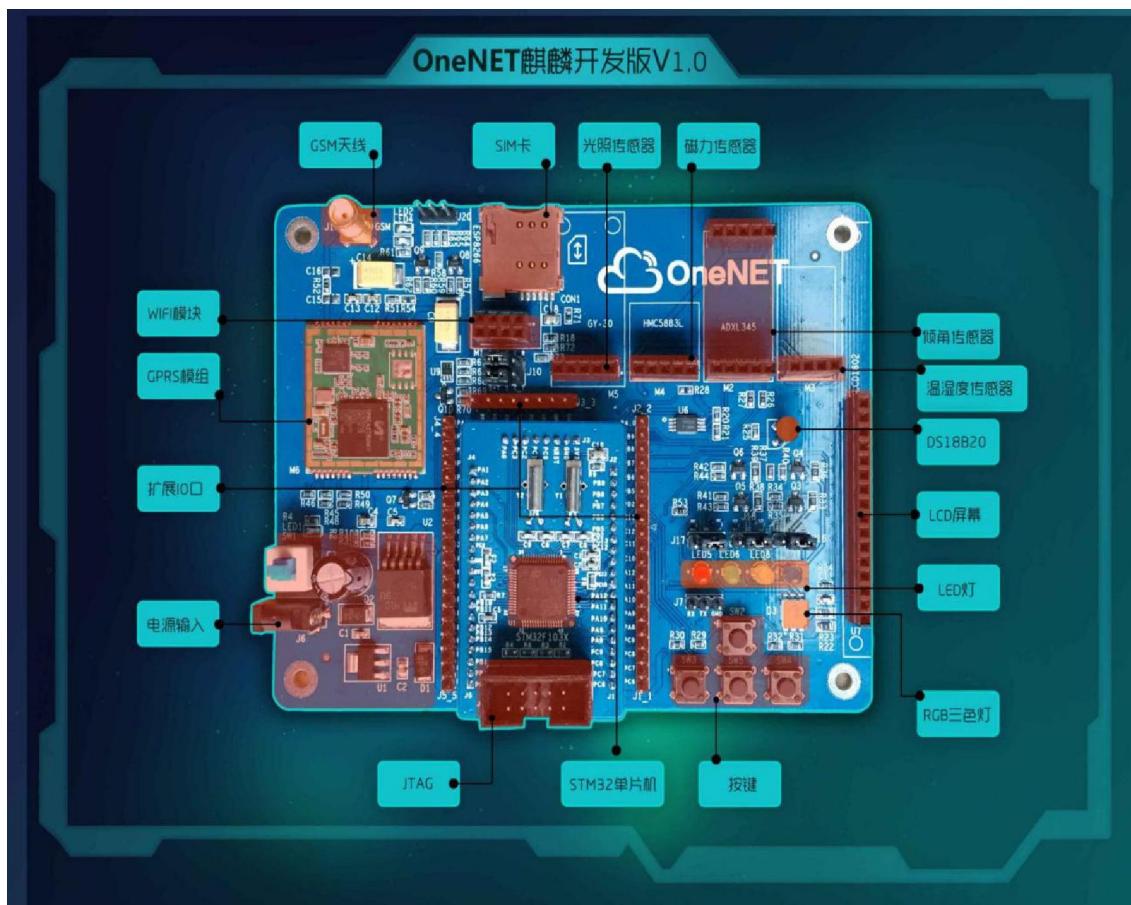


OneNET 平台麒麟开发板

实验例程操作手册



目录

版本信息.....	3
1. 本文档编写目标.....	4
2. OneNET 平台接入流程介绍	4
2.1 接入流程概述.....	4
2.2 资源组织示例.....	4
3. 实验例程操作说明	18
3.1 ESP8266+EDP LED 状态采集及控制实验.....	18
3.2 ESP8266+EDP 上传二进制文件	23
3.3 ESP8266+EDP 多路传感器采集实验	28
3.4 ESP8266+EDP 完成数据转发实验	34
3.5 ESP8266+HTTP 上传温湿度数据实验	40
3.6 GPRS+EDP 采集温湿度上传实验	45
3.7 ESP8266+Modbus 协议接入和数据上传实验	48
3.8 ESP8266+MQTT 协议发布订阅实验	52
3.9 EMW3081+EDP 发送二进制文件（可选）	58
参考资料.....	63

版本信息

日期	修订人	版本	更新内容
2016.4.8	周家绪、敬威、张健、续芳	V1.0	1.初始版本

1. 本文档编写目标

本手册是针对 OneNET 麒麟开发板 v1.0 的例程而编写的，包括每个实验例子的实验原理、参考资料说明、实验步骤等。

2. OneNET 平台接入流程介绍

2.1 接入流程概述

OneNET 的设备接入过程大致分为登录注册、新建项目、新增设备、新增数据流、上传并查看数据、新增应用等几个步骤，如下图所示。



图 2-1 OneNET 的设备接入流程

在接入设备之前必须先在 OneNET 平台 (<http://open.iot.10086.cn>) 注册用户账户，注册登录成功后，可以在用户账户下新增项目，在项目下新增设备，在设备下新增数据流。在设备端编写终端接入代码，主要完成数据采集、协议封装、数据上传等工作，终端数据上传成功后，在平台相应数据流下会生成随时间推移的数据点。最后，为了更直观的呈现数据的变化情况，用户可以运用应用孵化器自定义个性化应用并发布。

2.2 资源组织示例

用户注册

在接入 OneNET 之前，需在平台网站注册用户账户，登陆网址：
<http://open.iot.10086.cn/>，显示如下界面：

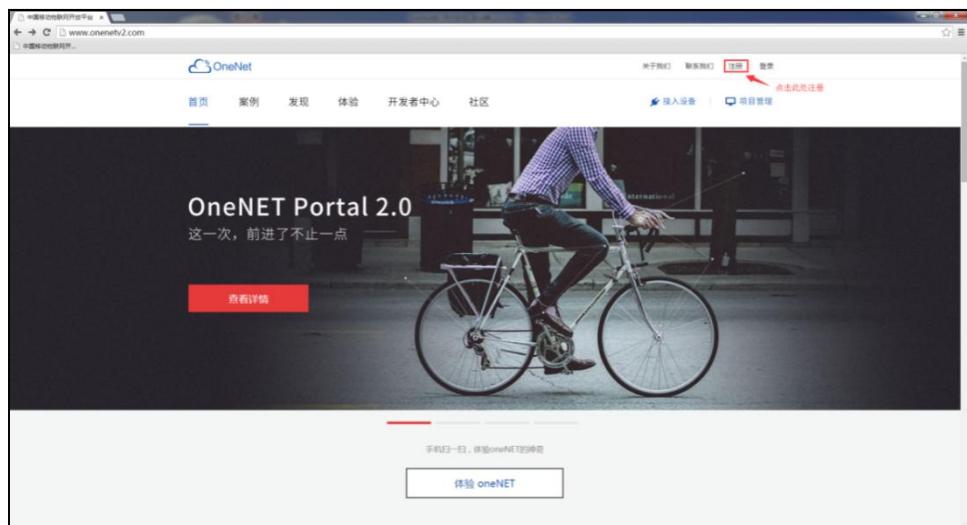


图 2-2 用户注册入口

点击注册入口，注册用户账户：

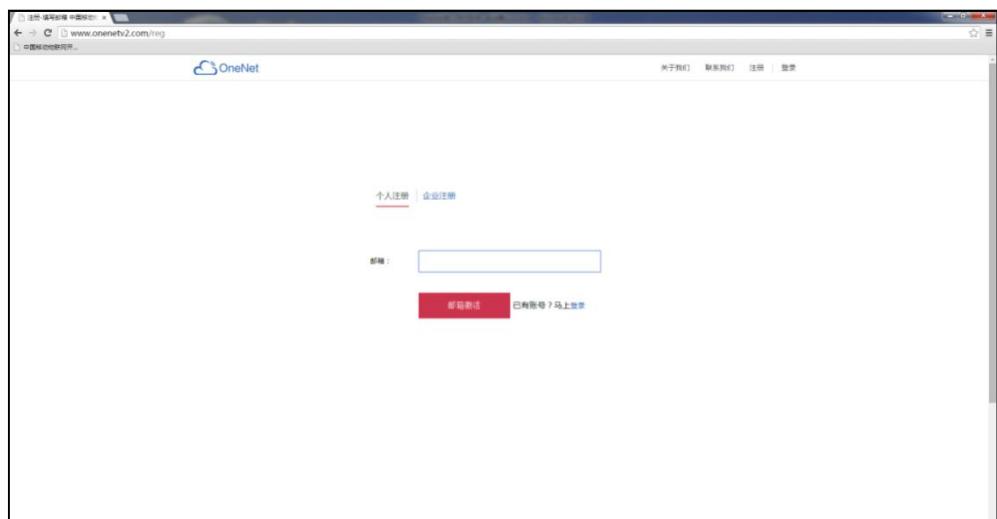


图 0-3 用户注册

填写有效邮箱地址，点击激活邮箱，将注册链接发入邮箱，查看邮箱邮件并点击注册链接：



图 0-4 邮件激活

填写注册信息，完成注册，如下图：

个人注册 | 企业注册

邮箱：

设置密码：
不能是纯数字或者纯字母，长度为6-12位

确认密码：
请再次输入密码

我已阅读并同意网站的服务条款

图 0-5 设置密码，完成注册

用户账户注册成功后，点击登录入口进行登录：

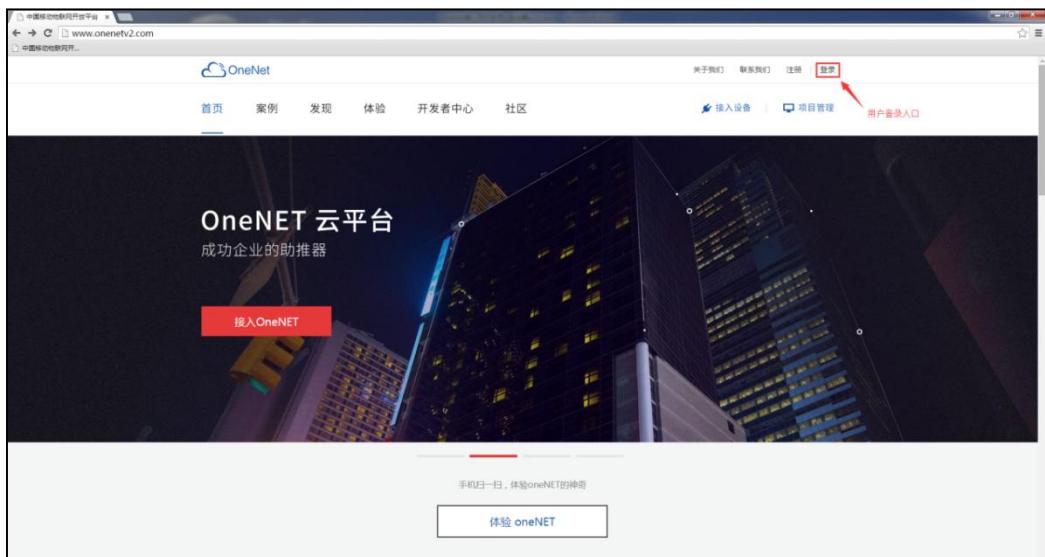


图 0-6 用户登录

添加项目

用户登录 OneNET 平台后，进入 OneNET 主界面，如下图：

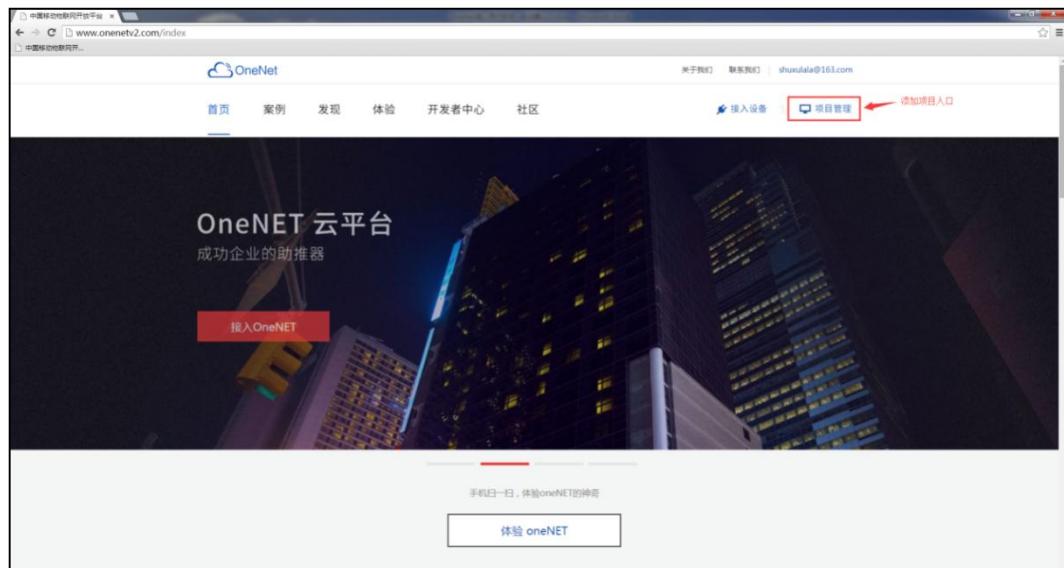


图 0-7 用户登录 OneNET 平台

点击“项目管理”入口，进入项目管理界面：

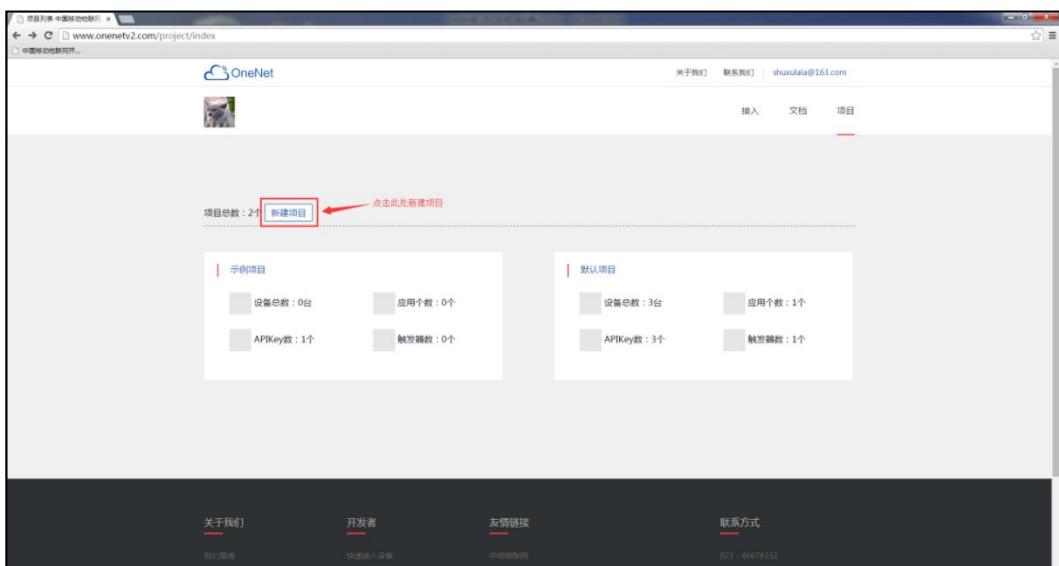


图 0-8 项目管理模块

点击“新建项目”进行新项目创建，如下图：

A screenshot of a 'New Project' creation form. The title '新建项目' is at the top. Below it is a field labeled '项目名称：' (Project Name) with an empty input box. Below that is a larger text area labeled '项目描述：' (Project Description) with an empty input box. At the bottom is a large red button labeled '立即创建项目' (Create Project Now).

图 0-9 新项目创建

填写项目名称和项目描述信息，点击“立即创建项目”完成项目创建，如下图：



图 0-10 项目详情首页

项目创建成功后，点击“去项目详情首页”进入单个项目详情页面，也可以在项目管理页面中点击单个项目进入项目详情首页：

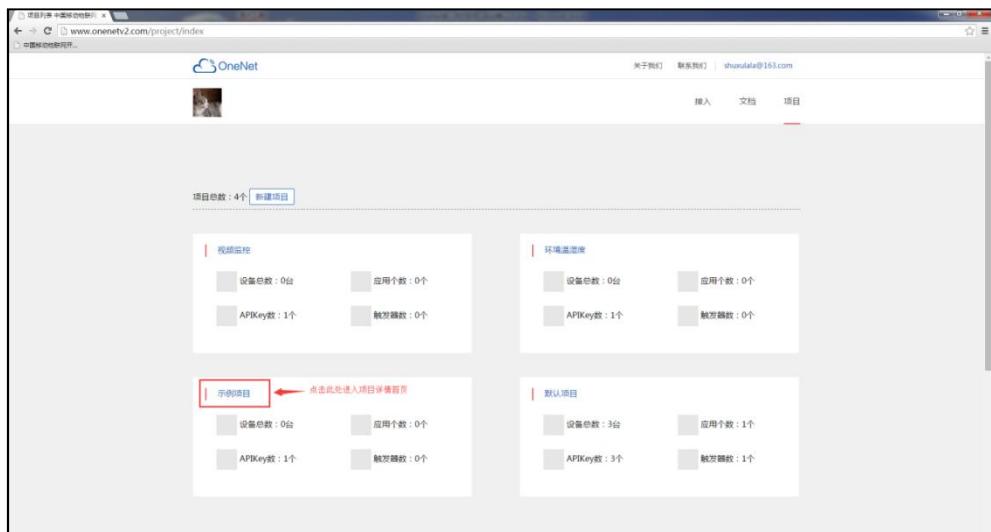


图 0-11 了解单个项目详情

在项目详情首页，用户可以查看该项目的相关概况，包括该项目基本信息以及各项统计数据，除此之外还可以进行该项目下的设备管理、应用管理、ApiKey 管理、触发器管理。项目详情首页如下图所示：

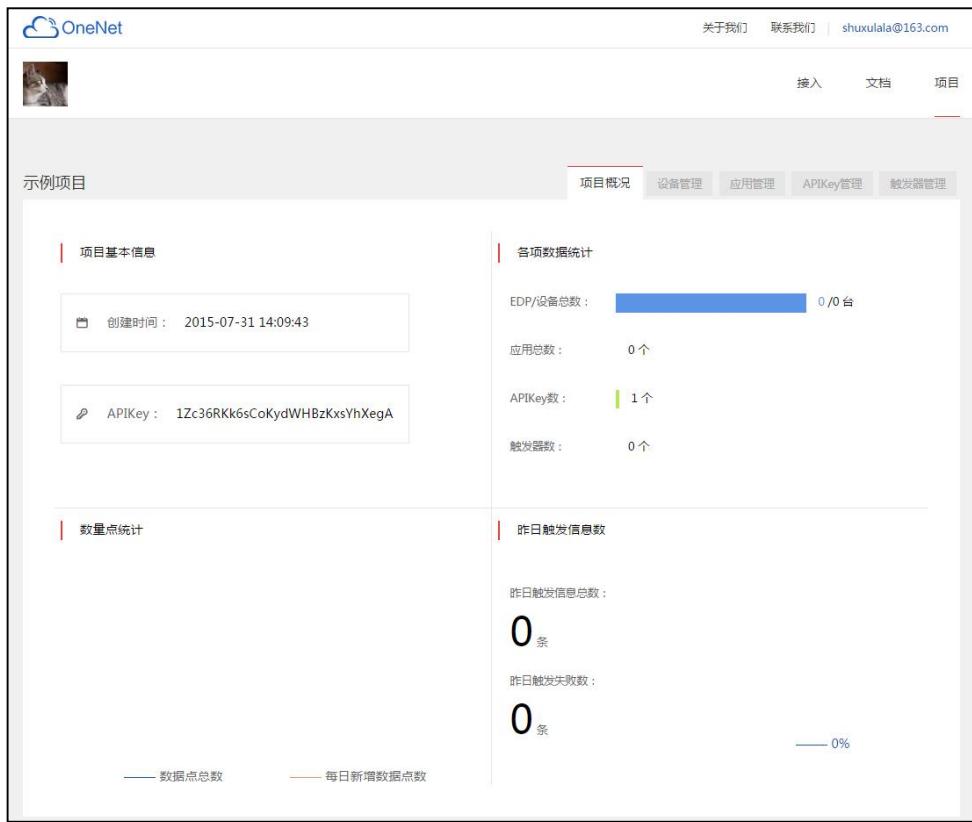


图 0-12 项目详情首页

值得注意的是，在上图中，项目基本信息栏里面显示的 ApiKey 为 MasterKey，它是一个字符串，在项目创建的时候由系统自动创建，与用户创建的 ApiKey 有所不同，MasterKey 权限最大，在该项目下的所有资源操作（协议文档列举的操作）都可以使用 MasterKey 进行鉴权。

添加设备

在项目详情首页，点击“设备管理”菜单，进入设备管理主界面，如下图所示：



图 0-13 设备管理模块

点击“新建设备”进行设备创建页面，如下图：



图 0-14 设备创建

填写设备创建的相关信息，包括设备所属项目、设备名称、设备标签、设备描述、图片上传、接入协议、默认路由、设备权限、设备位置等信息，填写完成后，点击“接入设备”，完成设备创建：



图 0-15 设备创建成功

点击“设备详情”进入设备详情页面，在设备详情页面可以查看单个设备的基本信息，为设备添加数据流、添加设备应用、设置设备相关联的 ApiKey，如下图：

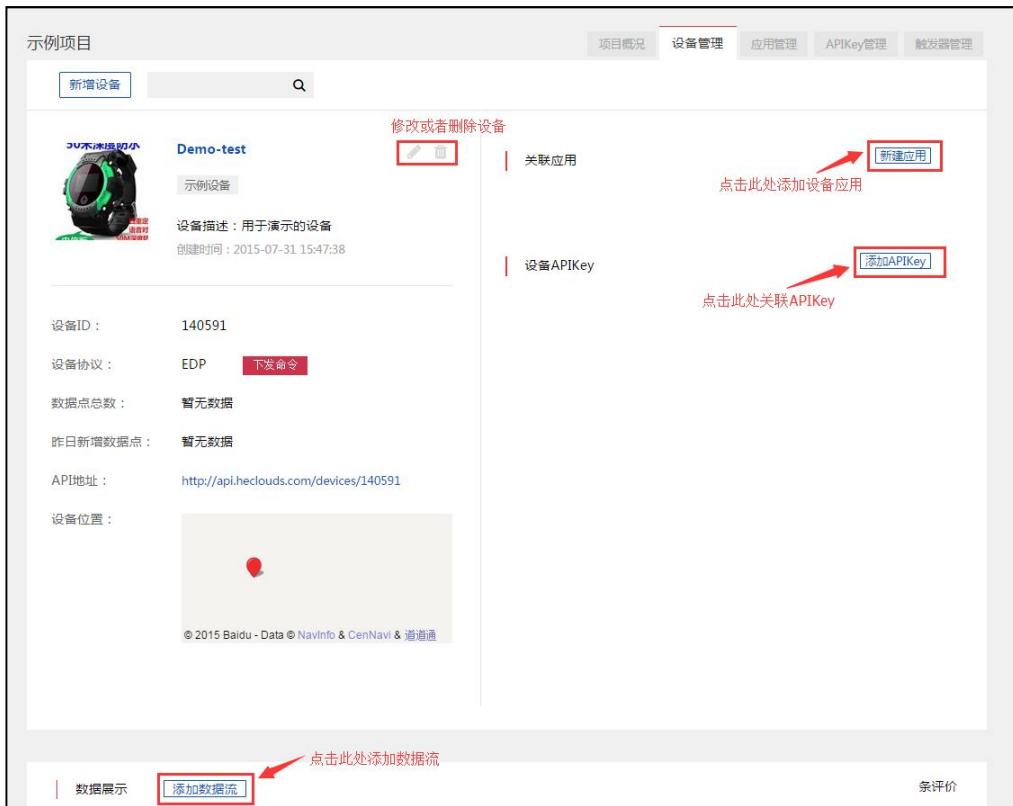


图 0-16 单个设备-数据流添加

点击“添加数据流”，进入数据流添加对话框：



图 0-17 新数据流添加

设置数据流相关的信息，点击“保存数据流”，完成一个数据流添加，在一个设备下面可以连续添加多个数据流，如下图，我们添加了 sys_time、bin_data 两个数据流：



图 0-18 数据流添加成功

除了添加数据流操作以外，用户还可以对已有的数据流进行修改、删除操作。

添加 ApiKey

目前平台有两种 ApiKey，第一种是项目默认 ApiKey，也就是 MasterKey，该 ApiKey 在项目创建的时候自动生成，默认关联本项目所有设备，权限最大，第二种是用户创建的 ApiKey，该 ApiKey 根据用户需要创建，并且由用户指定关联到相应的设备，只有 ApiKey 和相应设备关联才能用于该设备的登录鉴权。用户可以在项目下创建多个 ApiKey，并将这些 ApiKey 分别关联到多个设备，每个设备可以使用和其相关的 ApiKey 进行鉴权。在项目详情页面点击“ApiKey 管理”进入 ApiKey 管理主界面，如下图：

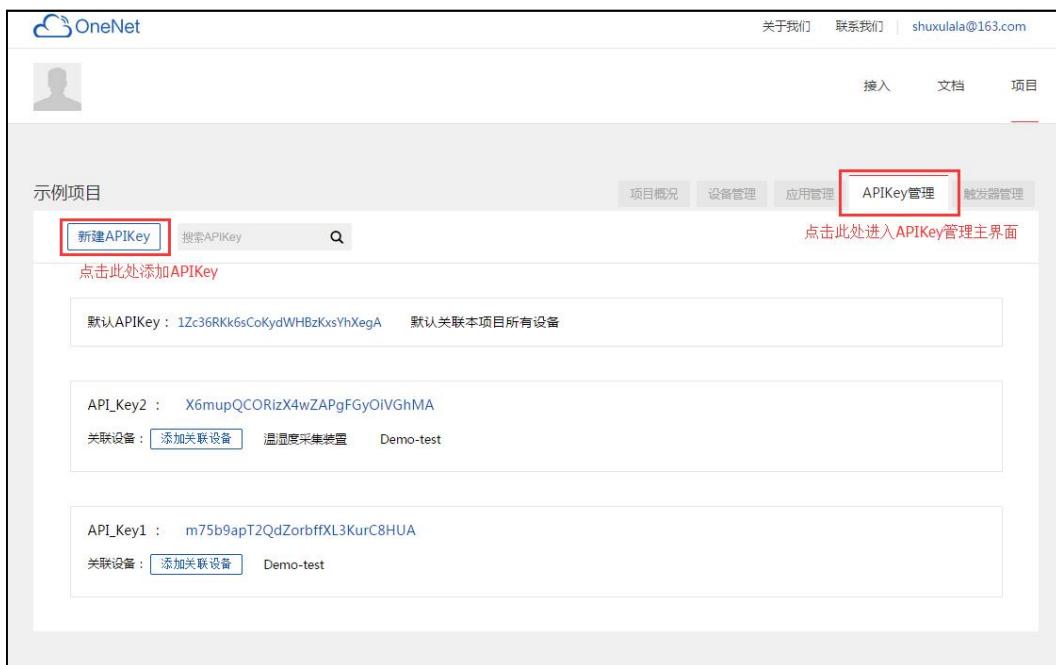


图 0-19 ApiKey 管理模块

点击“新建 ApiKey”进入 ApiKey 新建对话框，如下图：

图 0-20ApiKey 创建

设置 ApiKey 名称，选取与之相关的设备后，点击“添加 ApiKey”完成新 ApiKey 的添加，添加完成后，可以添加多个设备与之相关联，如下图：



图 0-21 为 ApiKey 建立设备、数据流关联

除了添加 ApiKey 操作以外，用户还可以对已有的 ApiKey 进行修改、删除操作。

添加触发器

用户可以用触发器对数据流进行监控，实现特定条件的事件告警，在触发器模块当中添加触发器，对设定条件的事件告警进行监控，触发器的原理是：在触发器中，用户事先设定好所要监控的数据流、事件触发条件、事件请求服务器地址，一旦监控的数据流数据满足设定的条件，触发器就会触发一个 HTTP 请求，将异常事件的内容通过请求发送给设定的服务器，所以用户如果要使用触发器功能，必须先建立自己的 HTTP 服务器，在项目详情页面点击“触发器管理”进入触发器管理主界面，如下图：



图 0-22 触发器管理模块

点击“新建触发器”进入触发器新增页面，分别设置新建触发器的名称、控制范围、数据流名称、触发添加、触发器请求服务器地址等信息后，点击“新增触发器”，完成触发器添加。如下图：

示例项目 / 新增触发器

触发器名称：时间告警

控制范围：Demo-test

数据流名称：sys_time

触发条件：选中数据流值 > 10

URL地址：http://time_test.com

新增触发器 取消

图 0-23 新增触发器

新建触发器完成后，自动转入触发器管理页面，可以为每个触发器关联多个设备，关联到同一触发器的各个设备必须具有相同数据流 ID 的数据流，如下图：



图 0-24 为触发器关联设备、数据流

除了添加触发器操作以外，用户还可以对已有的触发器进行修改、删除操作。

添加应用

用户可以为设备下的数据流创建相关的应用，发布对用户数据的直观展示页面，目前在 OneNET 应用孵化器中提供了曲线图、柱状图、表盘、物体位置、图片和开关等应用，用户添加应用进行数据展示的前提是用户数据已经上传至平台。以已有设备和数据流为例，为设备 Demo-test 下的 sys_time 数据流（设备上电后的时钟信息）添加一个曲线图的应用。

在项目详情页面点击“应用管理”进入应用管理主界面，如下图：



图 0-25 应用管理模块

点击“新建应用”，进入应用编辑器编辑页面，如下图：

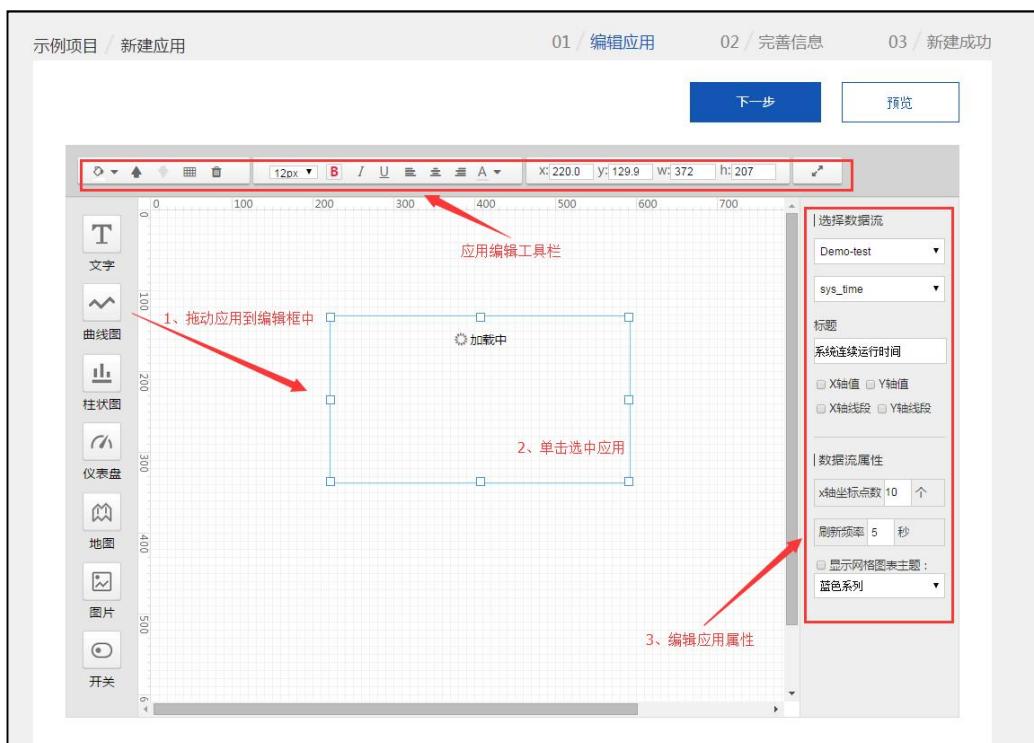


图 0-26 新应用创建

将想要添加的应用拖进编辑区域，比如：我们选择曲线图。点击选择该应用，在编辑框的右边编辑应用属性设置应用相关的参数，包括和应用相关联的设备、数据流、应用标题、刷新数据间隔、X 坐标点数目等，编辑完成后点击“下一步”，进入应用信息完善页面，如下图所示：

所属项目：示例项目

应用名称：XXX系统连续运行时间

应用状态：
私有应用 公开应用

应用描述：展示XXX设备连续运行的时间

保存应用 返回

图 0-27 应用信息完善

补充应用的名称、应用状态、应用描述等相关的应用辅助信息。点击“保存应用”，应用添加完毕，进入该应用的详情页面，如下图：

示例项目

新建应用

XXX系统连续运行时间

创建时间：2015-08-03 09:55:31
发布链接：<http://www.onenetv2.com/appview/p/12a68645c2e23346f52c7624568bd286>
关联设备：Demo-test

展示XXX设备连续运行的时间

编辑

图 0-28 应用详情页

除了添加应用操作以外，用户还可以对已有的应用进行修改、删除操作。

3. 实验例程操作说明

3.1 ESP8266+EDPLED 状态采集及控制实验

代码目录

实验代码\OneNET_Demo_ESP8266_EDP_Led

实验原理

通过采集 PC7/PC8/PC10/PA12 四个 LED 指示灯的 IO 控制引脚的电平值，利用 EDP 协议将采集的数据上传至 OneNET 平台。通过 OneNET 平台的“下发命令”功能实现对 LED 控制（点亮和熄灭），同时也可以通过开发板上的四个按键控制对应的 LED。

实验步骤

Step1 硬件连接

- 1) 按照开发板硬件手册说明选择 LED5、LED6、LED7 的跳线（LED5 为红色指示灯，将 J17 的 R 和 L_R 用跳线帽短接；LED6 为绿色指示灯，将 J18 的 G 和 L_G 用跳线帽短接；LED7 为蓝色指示灯，将 J16 的 B 和 L_B 用跳线帽短接），LED8 不需要跳线选择；
- 2) 连接 UART 调试串口，波特率 115200，8N1；
- 3) 连接 ST-LINK；
- 4) 连接 ESP8266 WIFI 模组，请确保 J10 跳线连接到 ESP8266 一侧；
- 5) 接入电源，烧写程序前先上电。

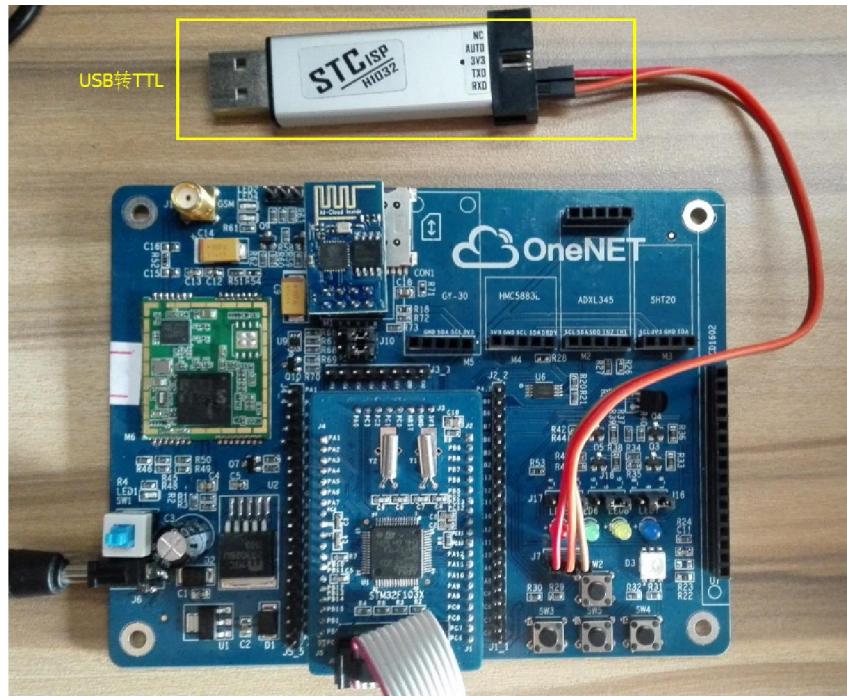


图 3-1 开发板连接

Step2 打开实验工程，目录：

实验代码\OneNET_Demo_ESP8266_EDP_Led\Project\OneNETdemo.uvproj。

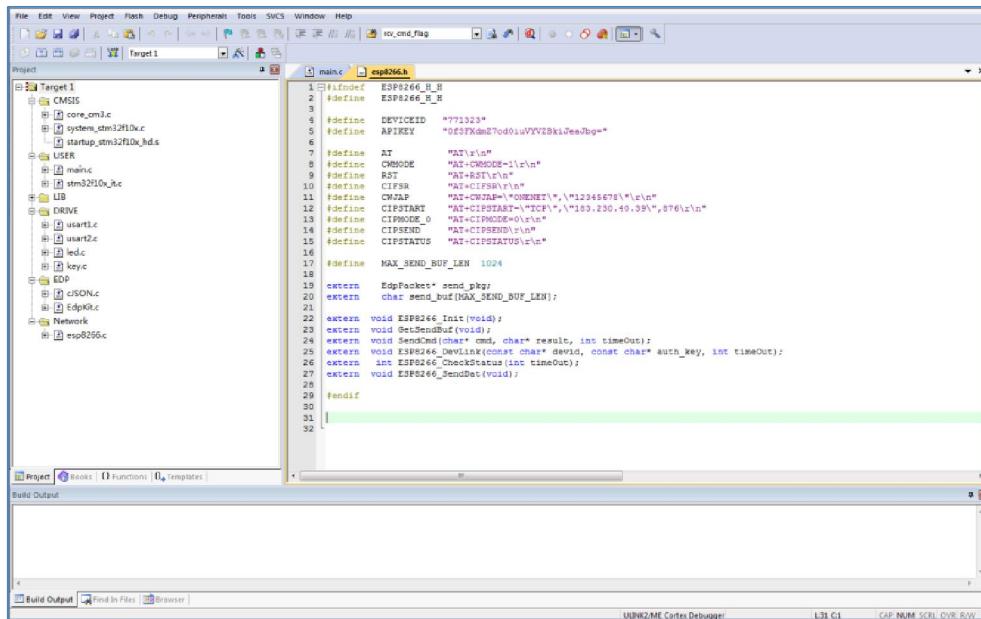


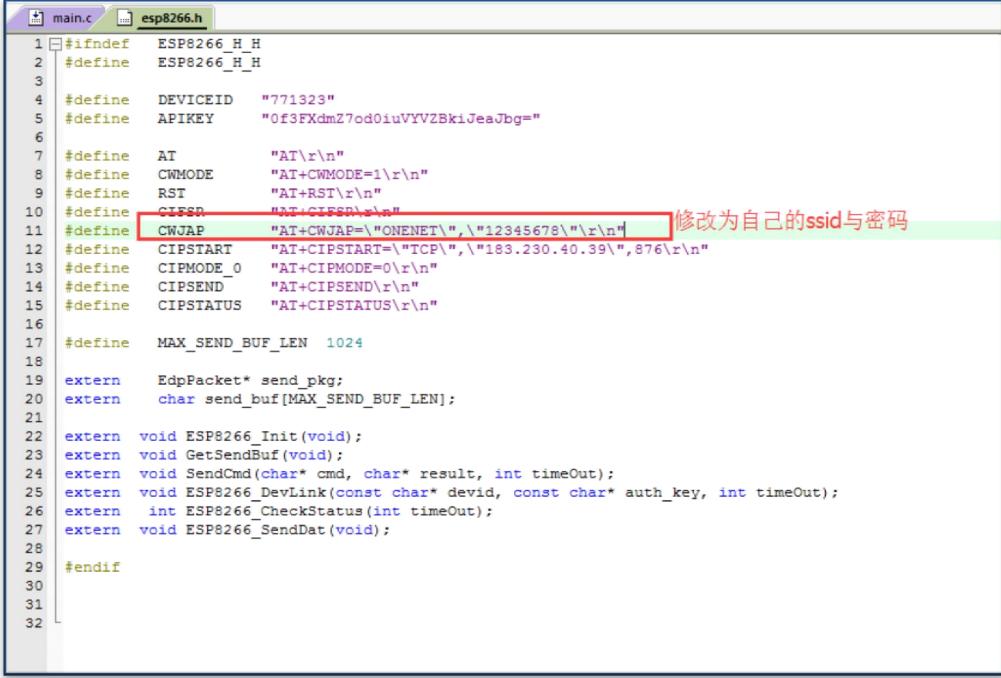
图 3-2 打开 Keil 工程

Step3 修改代码

主要是修改 ESP8266 的接入点配置和设备相关信息。如下图，在 esp8266.h 头文件中修改 CWJAP 宏定义为读者自己路由器的 ssid 和 pwd:

源代码中的 ssid 为 OneNET, pwd 为 12345678。

修改 dev_id 及 ApiKey 信息，在 esp8266.h 头文件中修改宏定义 DEVICEID 为读者在平台创建的设备对应的 dev_id，修改宏定义 APIKEY 为读者在平台创建的设备对应的设备 ApiKey。



```
1 #ifndef ESP8266_H_H
2 #define ESP8266_H_H
3
4 #define DEVICEID "771323"
5 #define APIKEY "0f3FXdmZ7od0iuVYVZBkiJeaJbg="
6
7 #define AT "AT\r\n"
8 #define CNMODE "AT+CNMODE=1\r\n"
9 #define RST "AT+RST\r\n"
10 #define CIPSSD "AT+CIPSSD\r\n"
11 #define CWJAP "AT+CWJAP=\"%ONENET\",\"12345678\"\r\n" 修改为自己的ssid与密码
12 #define CIPSTART "AT+CIPSTART=\"TCP\",\"183.230.40.39\",876\r\n"
13 #define CIPMODE_0 "AT+CIPMODE=0\r\n"
14 #define CIPSEND "AT+CIPSEND\r\n"
15 #define CIPSTATUS "AT+CIPSTATUS\r\n"
16
17 #define MAX_SEND_BUF_LEN 1024
18
19 extern EdpPacket* send_pkg;
20 extern char send_buf[MAX_SEND_BUF_LEN];
21
22 extern void ESP8266_Init(void);
23 extern void GetSendBuf(void);
24 extern void SendCmd(char* cmd, char* result, int timeOut);
25 extern void ESP8266_DevLink(const char* devid, const char* auth_key, int timeOut);
26 extern int ESP8266_CheckStatus(int timeOut);
27 extern void ESP8266_SendDat(void);
28
29
30
31
32
```

图 3-3 修改 dev_id 及设备 ApiKey

Step4 编译整个工程，并下载程序，请保证 ST-LINK 和电源已正确连接。

Step5 烧写程序完后，重新给开发板上电，建议每次烧写程序后都这样做，让 WIFI 模块完全掉电，以免 AT 命令操作失败。

Step6 程序运行后，打印如下：



图 3-4 程序启动打印信息

Step7 OneNET 平台下查看上传的数据

如下图,最新数据为 0 表示对应的 LED 指示灯为熄灭状态, 为 1 表示对应的 LED 指示灯为点亮状态。

参数名	最新数据	更新时间
green_statu	0	2016-04-06 08:56:36
blue_statu	1	2016-04-06 08:56:36
red_statu	1	2016-04-06 08:56:36
yellow_statu	0	2016-04-06 08:56:36

图 3-5 平台数据查看

Step8 打开 OneNET 平台的设备页面, 点击“下发命令”, 弹出下发命令窗口如图所示: 输入命令内容“LED10”, 点击“发送命令”, 观察红色 LED 指示灯是否熄灭。



图 3-6 平台命令下发

程序中对控制命令的内容约定如下表（用户可以自己约定命令内容）：

命令内容	命令响应	命令内容	命令响应
LED10	熄灭红色 LED	LED11	点亮红色 LED
LED20	熄灭绿色 LED	LED21	点亮绿色 LED
LED30	熄灭黄色 LED	LED31	点亮黄色 LED
LED40	熄灭蓝色 LED	LED41	点亮蓝色 LED

Step9 通过按键改变 LED 的状态，在 OneNET 平台下查看上传的数据是否对应改变，程序中按键和 LED 的对应关系如下：

SW2 按键<---->红色 LED

SW3 按键<---->绿色 LED

SW4 按键<---->蓝色 LED

SW5 按键<---->黄色 LED

3.2 ESP8266+EDP 上传二进制文件

代码目录

实验代码\OneNET_Demo_ESP8266_EDP_Picture

实验原理

我们平时用到的图片本质是一个二进制文件，将它转换为字符数组后，利用EDP 协议中支持的二进制文件上传功能，将其上传至平台。

实验步骤

Step1 如果没有 EDP 设备，请先在 OneNET 平台创建 EDP 设备，具体步骤参考《2.1 接入流程概述》，本例中所用到的设备参数如下：



图 3-7 实验设备信息

Step2 硬件连接

- 1) 本例中的图片数据是事先准备好的数组，不需要外接其他传感器；
- 2) 连接 UART 调试串口，波特率 115200，8N1；
- 3) 连接 ST-LINK；
- 4) 连接 ESP8266 WIFI 模组，请确保 J10 跳线连接到 ESP8266 一侧；
- 5) 接入电源，烧写程序前先上电。

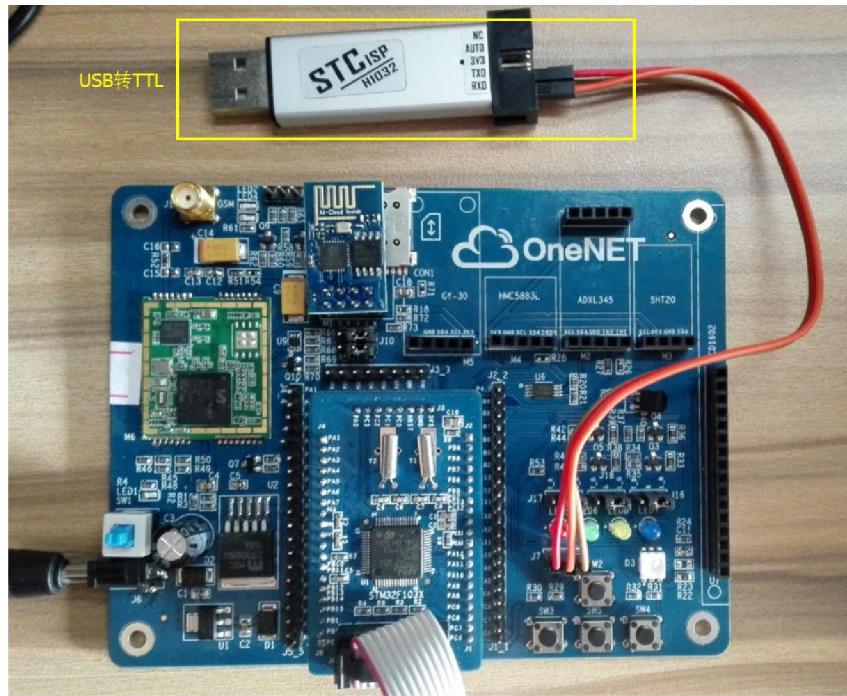


图 3-8 实验设备连接图

Step3 打开实验工程，目录：

实验代码\OneNET_Demo_ESP8266_EDP_Picture\Project\OneNETdemo.uvproj

1) 修改设备鉴权参数

在 main.c 文件中，根据自己的设备信息，修改 ApiKkey 和 dev_id:

```

4  * @author  STMicroelectronics Application Team
5  * @version V3.5.0
6  * @date    08-April-2011
7  * @brief   Main program body
8  ****
9  *
10 /*
11 * THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIDS AT PROVIDING CUSTOMERS
12 * WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE
13 * TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY
14 * DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING
15 * FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS OF THE
16 * CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS.
17 */
18 * <h2><center>&copy; COPYRIGHT 2011 STMicroelectronics</center></h2>
19 ****
20 */
21
22 /* Includes -----
23 //USE_STDEPERIPH_DRIVER, STM32F10X_HD, USE_STM3210B_EVAL
24 */
25 #include "stm32f10x.h"
26 #include "stdio.h"
27 #include "usart1.h"
28 #include "usart2.h"
29 #include "util.h"
30
31 #include "cJSON.h"
32 #include "esp8266.h"
33 #include "Edpkit.h"
34
35
36 #define API_KEY      "DXZccGKxgrpxZJKWFnbMzx1jeITk=" //API KEY 需要修改为用户自己的对应参数
37 #define DEV_ID       "1078702" //设备ID 需要修改为用户自己的对应参数
38 #define PKT_SIZE     200 //图片分包大小，根据模块特性修改，分包过大容易
39
40 #include "image_2k.c" //图片文件二进制数组
41

```

图 3-9 修改 ApiKey 和 dev_id

2) 修改 WIFI 接入点的 ssid 与 pwd

在 esp8266.h 文件中，根据自己的网络环境，将参数修改为：

```
#define CWJAP      "AT+CWJAP=\"your_ssid\", \"your_pwd\"\r\n",
```

本例中 ssid 为 onenet，pwd 为空：

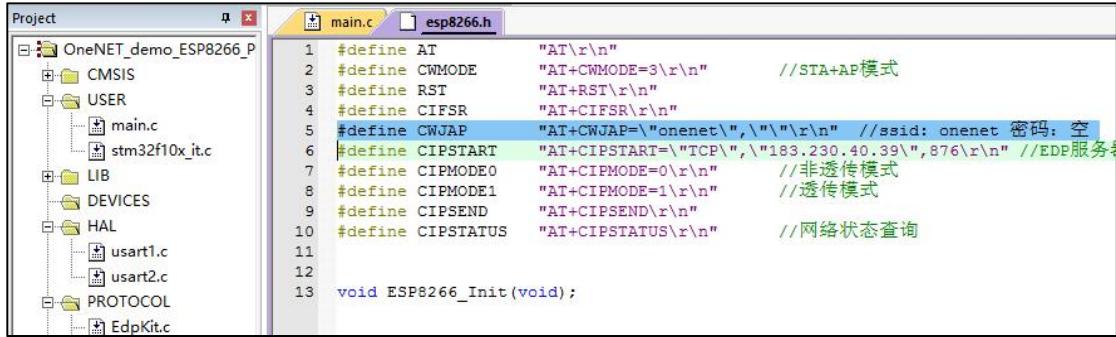


图 3-10 修改 ssid 与 pwd

Step4 编译整个工程，并下载程序。

Step5 烧写程序完后，重新给开发板上电，建议每次烧写程序后都这样做，让 WIFI 模块完全掉电，以免 AT 命令操作 WIFI 失败。

Step6 程序运行后，调试串口会有如下打印信息：

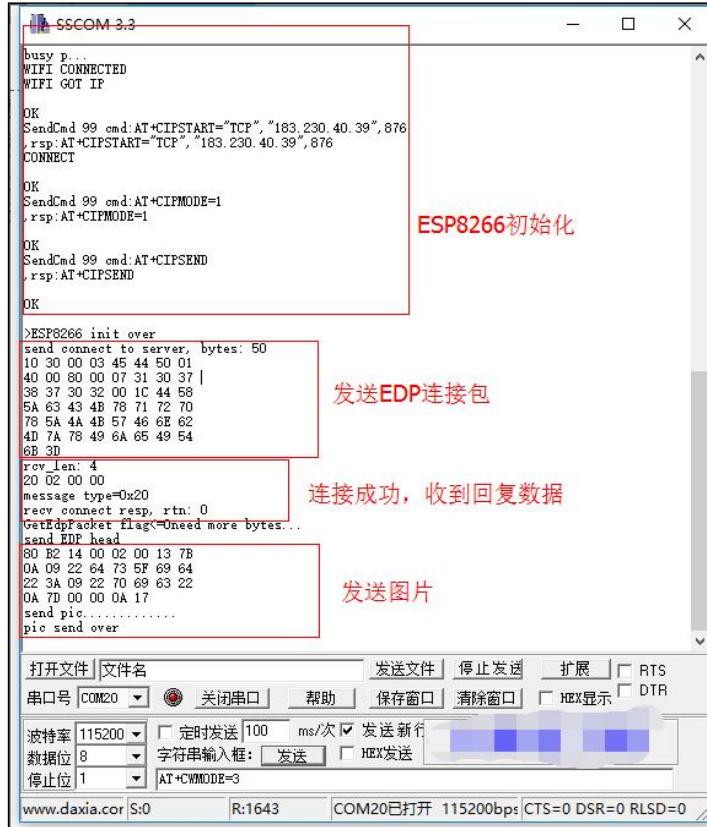


图 3-11 串口打印信息

Step7 OneNET 平台下查看上传的数据。



图 3-12 数据查看

附加说明：

本例中的图片信息是以字符数组的形式保存在 `image_2k.c` 文件中，通过`#include` 包含在 `main.c` 文件中。

```

Project main.c
OneNET_demo_ESP8266_P
  CMSIS
  USER
    main.c
    stm32f10x_it.c
  LIB
  DEVICES
  HAL
    USART1.c
    USART2.c
  PROTOCOL
    EdpKit.c
    EdpDemo.c
  UTILS
    cJSON.c
    utils.c
  NETWORK
    esp8266.c

1  * @brief Main program body
2  ****
3  * @attention
4  *
5  * THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING
6  * WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THE
7  * TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FO
8  * DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLA
9  * FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMER
10 * CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PROD
11 *
12 * <h2><center>&copy; COPYRIGHT 2011 STMicroelectronics</center></h2>
13 ****
14 */
15
16 /* Includes -----
17 //USE_STDPERIPH_DRIVER, STM32F10X_HD, USE_STM3210B_EVAL
18 */
19
20 #include "stm32f10x.h"
21 #include "stdio.h"
22 #include "uart1.h"
23 #include "uart2.h"
24 #include "utils.h"
25
26 #include "cJSON.h"
27 #include "esp8266.h"
28 #include "Edpkit.h"
29
30
31
32
33
34
35
36 #define API_KEY      "DXZcCKxqrpxZJKWFnbMzxIjeITk=" //API KEY 需要修改
37 #define DEV_ID       "1078702" //设备ID 需要修改
38 #define PKT_SIZE     200 //图片分包大小，根据实际情况修改
39
40 #include "image_2k.c" //图片文件二进制数组
41
42 /**
43 * @brief 实现EDP连接，每5s上传一次图片数据到平台

```

Project: OneNET_demo_ESP8266_P

```

main.c image_2k.c
1 /**
2  * image_2k.c
3  *
4  * 使用EDP上传二进制文件所用的图片文件的二进制数组
5  *
6  */
7 const char image[] =
8 {
9     0xff, 0xd8, 0xff, 0xe0, 0x00, 0x10, 0x4a, 0x46, 0x49, 0x46, 0x00, 0x01, 0x01, 0x00, 0x60,
10    0x00, 0x60, 0x00, 0x00, 0xff, 0xdb, 0x00, 0x43, 0x00, 0x08, 0x06, 0x06, 0x07, 0x06, 0x05, 0x08,
11    0x07, 0x07, 0x07, 0x09, 0x09, 0x08, 0x0a, 0x0c, 0x14, 0x0d, 0x0c, 0x0b, 0x0b, 0x0c, 0x19, 0x12,
12    0x13, 0x0f, 0x14, 0x1d, 0x1a, 0x1f, 0x1e, 0x1d, 0x1a, 0x1c, 0x20, 0x24, 0x2e, 0x27, 0x20,
13    0x22, 0x2c, 0x23, 0x1c, 0x1c, 0x28, 0x37, 0x29, 0x2c, 0x30, 0x31, 0x34, 0x34, 0x34, 0x1f, 0x27,
14    0x39, 0x3d, 0x38, 0x32, 0x3c, 0x2e, 0x33, 0x34, 0x32, 0xff, 0xdb, 0x00, 0x43, 0x01, 0x09, 0x09,
15    0x09, 0x0c, 0x0b, 0x0c, 0x18, 0x0d, 0x0d, 0x18, 0x32, 0x21, 0x1c, 0x21, 0x32, 0x32, 0x32,
16    0x32, 0x32,
17    0x32, 0x32,
18    0x32, 0x32,
19    0x00, 0x11, 0x08, 0x00, 0x30, 0x00, 0x88, 0x03, 0x01, 0x22, 0x00, 0x02, 0x11, 0x01, 0x03, 0x11,
20    0x01, 0xff, 0xc4, 0x00, 0x1a, 0x00, 0x00, 0x03, 0x01, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00,
21    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x05, 0x03, 0x02, 0x06, 0x01, 0xff, 0xc4, 0x00,
22    0x18, 0x01, 0x00, 0x03, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
23    0x00, 0x00, 0x00, 0x01, 0x02, 0x03, 0x04, 0xff, 0xda, 0x00, 0x0c, 0x03, 0x01, 0x00, 0x02, 0x10,
24    0x03, 0x10, 0x00, 0x01, 0xf7, 0xe1, 0xe0, 0xa2, 0xe1, 0xe3, 0xde, 0xd7, 0x2f, 0x44, 0x73,
25    0x0a, 0x0b, 0x0a, 0x48, 0x73, 0x0a, 0x5a, 0x40, 0x73, 0x0a, 0x8, 0x42, 0x73, 0x0a, 0x10, 0x73
}

```

图 3-13 图片文件数组

3.3 ESP8266+EDP 多路传感器采集实验

代码目录

实验代码\OneNET_Demo_ESP8266_EDP_Sensors

实验原理

使用开发板的 I2C 接口采集配套的 4 路传感器（包括 SHT20 温湿度传感器，ADXL345 三轴加速度传感器，HMC5883L 三轴磁场传感器，BH1750FVI 光照度传感器），并通过开发板上 ESP8266 与 EDP 服务器建立 TCP 连接，利用 EDP 协议将采集的数据保存到 OneNET 平台。

实验步骤

Step1 如果没有 EDP 设备，请先在 OneNET 平台创建 EDP 设备，具体步骤参考《2.1 接入流程概述》，本例中所用到的设备参数如下：



图 3-14 EDP 设备创建

Step2 在上一步中创建的 EDP 设备下，为每个传感器的采集结果创建数据流，如果已经创建请跳过此步；

设备JSON数据

```

{
  "id": "768596",
  "datastreams": [
    {
      "create_time": "2016-04-01 13:41:28",
      "id": "BH1750FVI",
      "uuid": "d7303d3b-1e89-4517-be50-4cddc122acd7"
    },
    {
      "create_time": "2016-04-01 13:08:00",
      "id": "ADXL345_z",
      "uuid": "94b5d036-34f7-468e-80f0-7dac5b256b40"
    },
    {
      "create_time": "2016-04-01 13:08:00",
      "id": "HMC5883L_x",
      "uuid": "15583ec7-d732-4cb4-aab1-3ca955a4288c"
    },
    {
      "create_time": "2016-04-01 13:08:00",
      "id": "ADXL345_y",
      "uuid": "1f5bc12b-b8ef-4820-9e24-d3f6e38eee48"
    },
    {
      "create_time": "2016-04-01 13:08:00",
      "id": "HMC5883L_z",
      "uuid": "e5486532-3481-47fe-9eaf-db675b60cca7"
    }
  ]
}

```

图 3-15 EDP 设备数据流创建

Step3 硬件连接

- 1) 按照开发板硬件手册说明分别连接上 SHT20, ADXL345, HMC5883L, BH1750FVI 这 4 个模块;
- 2) 连接 UART 调试串口, 波特率 115200, 8N1;
- 3) 连接 ST-LINK;
- 4) 连接 ESP8266WIFI 模组, 请确保 J10 跳线连接到 ESP8266 一侧;
- 5) 接入电源, 烧写程序前先上电。

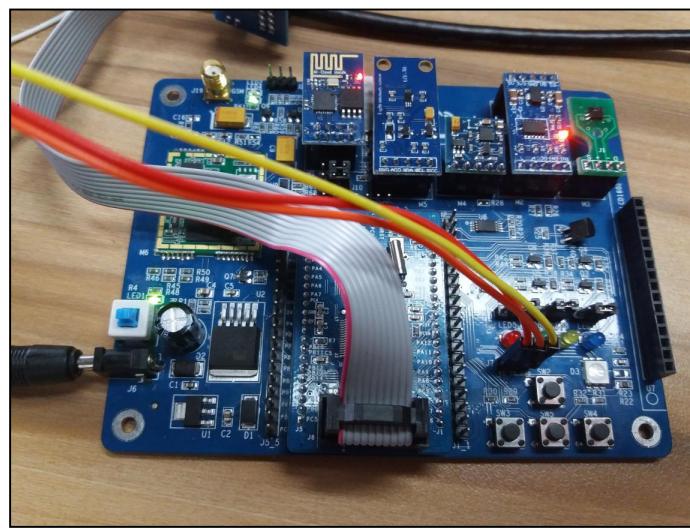


图 3-16 开发板连接

Step4 打开实验工程，目录：

实验代码\OneNET_Demo_ESP8266_EDP_Sensors\Project\OneNET_Demo_EDP_Sensors.uvproj。

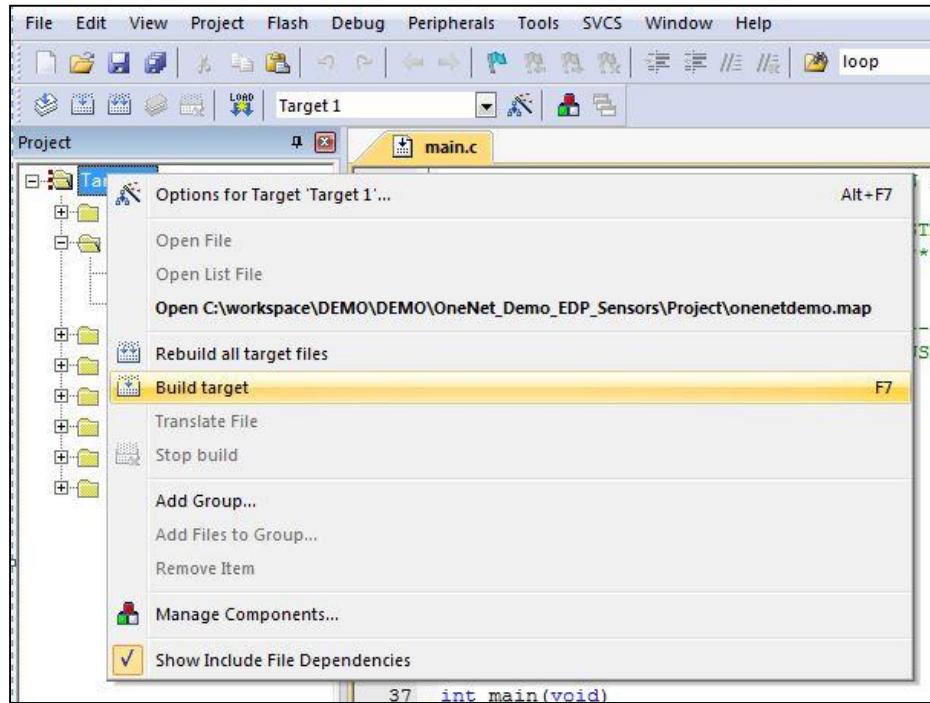


图 3-17 Keil 编译工程

Step5 修改代码，主要是修改 ESP8266 的接入点配置和设备相关信息。在 esp8266.h 头文件中修改 CWJAP 宏定义为读者使用的路由器 ssid 和 pwd，如：

```
#define CWJAP      "AT+CWJAP=\"your_ssid\",\"pwd\"\r\n"
```

源代码的 ssid 为 OneNET，pwd 为空：

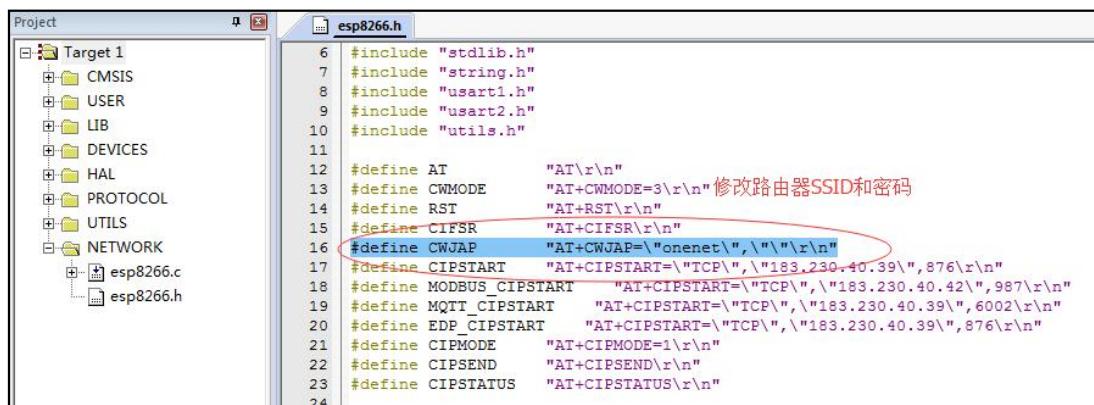


图 3-18 修改 ssid 和 pwd

找到源文件 EdpDemo.c，修改：

```
int8_t src_api_key[] = "your_api_key";      //修改成读者设备对应的 ApiKey
int8_t src_dev[] = "your_device_id";        //修改成读者对应的设备 ID
```

```

19 #include "EdpDemo.h"
20 #include "esp8266.h"
21
22 int8_t src_api_key[] = "nCVNXYCoX68IHG4DgpyNu5aTxY=";
23 int8_t src_dev[] = "768596";
24 /**
25 * @brief EDP数据包发送
26 * @param buffer: 要发送的数据缓冲区地址
27 * @param len: 要发送的数据缓冲区长度
28 * @param sockfd: 兼容linux socket api: STM32下无意义
29 * @retval 发送的数据长度
30 */
31 int32_t DoSend(int32_t sockfd, const uint8_t *buffer, uint32_t len)
32 {
33     USART2_SendData((uint8_t *)buffer), len);
34     /* wululu test print send bytes */

```

图 3-19 修改 ApiKey 和 dev_id

Step6 编译整个工程，并下载程序，请保证 JATG 和电源已正确连接。

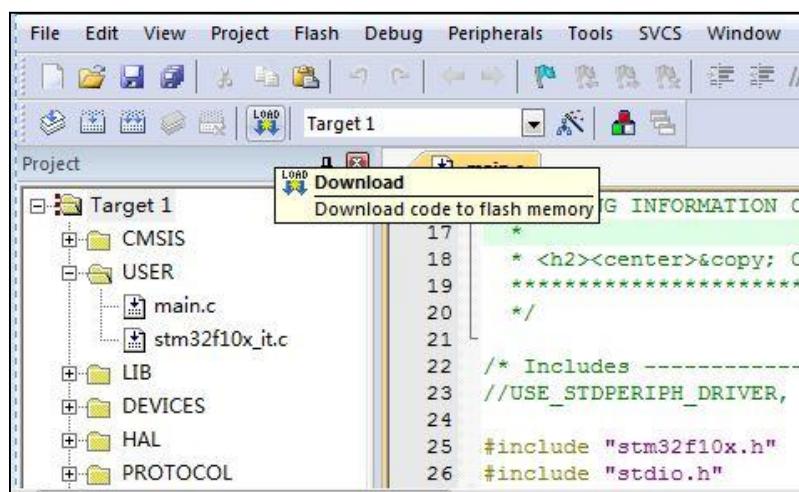


图 3-20 Keil 下载程序

Step7 烧写程序完后，重新给开发板上电，建议每次烧写程序后都这样做，让 WIFI 模块完全掉电，以免 AT 命令操作 WIFI 失败。

Step8 程序运行后，打印如下：



图 3-21 EDP 连接打印

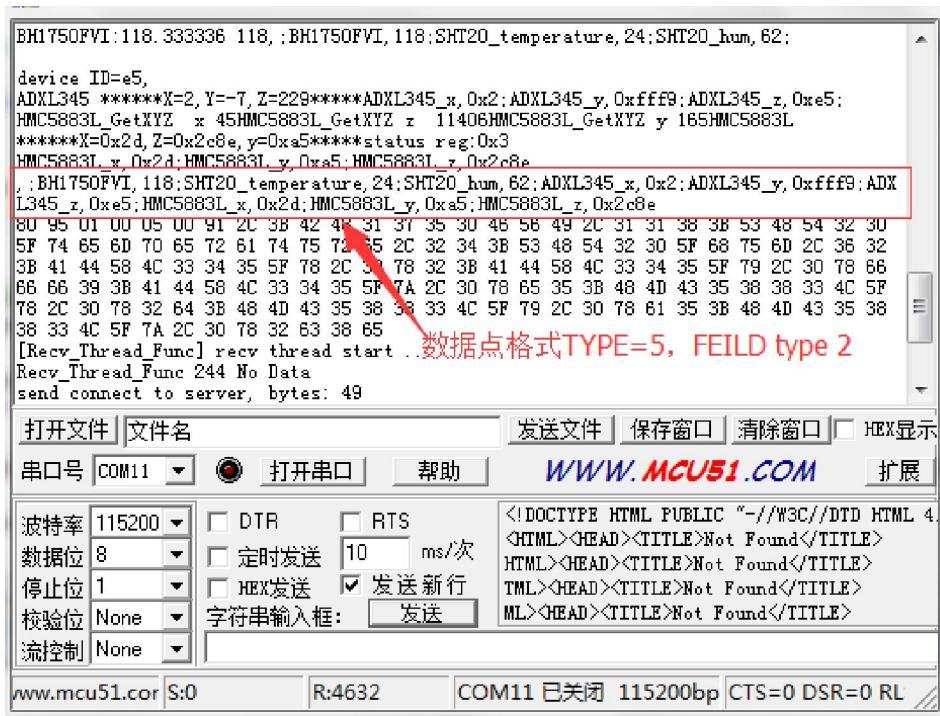


图 3-22 EDP 数据上传打印

Step9 OneNET 平台下查看上传的数据。

	数据展示	添加数据流
 BH1750FVI		最新数据 : 108
 ADXL345_z		最新数据 : 0xe5
 HMC5883L_x		最新数据 : 0x4
 ADXL345_y		最新数据 : 0xffff9
 HMC5883L_z		最新数据 : 0x35e
 SHT20_hum		最新数据 : 58
 HMC5883L_y		最新数据 : 0xe3

图 3-23 前端展示数据流

3.4 ESP8266+EDP 完成数据转发实验

代码目录

实验代码\OneNET_Demo_ESP8266_EDP_Push

实验原理



使用开发板通过 EDP 连接 OneNET 平台，使用 Edp 调试软件模拟同一项目下的另外一个设备，利用 EDP 的转发功能，发消息到开发板，开发板收到消息之后将该消息转发回 EdpProtoDebugger 模拟设备。

实验步骤

Step1 如果没有 EDP 设备，请先在 OneNET 平台创建 EDP 设备，具体步骤参考《2.1 接入流程概述》，本例中所用到的设备参数如下(其中“edp 设备”为开发板使用，“透传测试”为 EdpProtoDebugger 模拟设备)：

Two screenshots of the OneNET platform interface showing device configuration details. Both screens show success messages and configuration parameters for two different devices.

Left Screen (edp 设备):

- 恭喜，设备 **edp 设备** 修改成功
- 你现在可以为你的设备上传数据
- 上传数据** (button)
- 所属项目：开发板测试
- 设备ID：1078702
- 设备协议：EDP协议
- API地址：<http://api.heclouds.com/devices/1078702>
- 设备APIKey：[DXZcCKxqrpxZJKWFnbMzxlijelTk=](#)
- API文档** (button)
- 设备详情** (button)

Right Screen (透传测试):

- 恭喜，设备 **透传测试** 修改成功
- 你现在可以为你的设备上传数据
- 上传数据** (button)
- 所属项目：开发板测试
- 设备ID：1079750
- 设备协议：EDP协议
- API地址：<http://api.heclouds.com/devices/1079750>
- 设备APIKey：[DXZcCKxqrpxZJKWFnbMzxlijelTk=](#)
- API文档** (button)
- 设备详情** (button)

图 3-24 实验设备信息

Step2 硬件连接

- 1) 连接 UART 调试串口，波特率 115200，8N1；
- 2) 连接 ST-LINK；
- 3) 连接 ESP8266 WIFI 模组，请确保 J10 跳线连接到 ESP8266 一侧；
- 4) 接入电源，烧写程序前先上电。

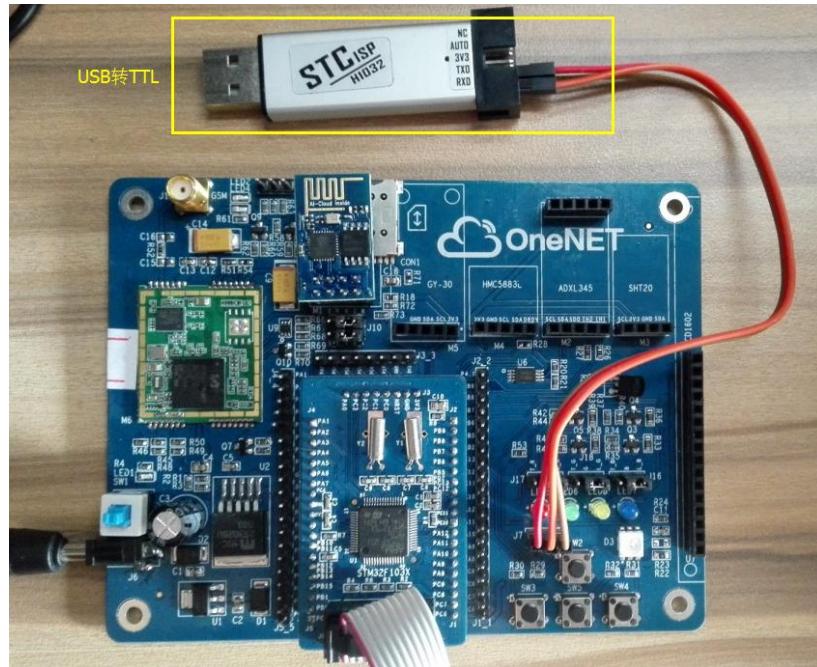


图 3-25 硬件连接

Step3 打开实验工程，目录：

实验代码\OneNET Demo ESP8266 EDP Push\Project\OneNETdemo.uvproj

1) 修改设备鉴权参数

在 main.c 文件中，根据自己的设备信息，修改 ApiKey 和 dev_id：

```
1 /**
2 * *****  
3 * @file      Project/STM32F10x_StdPeriph_Template/main.c  
4 * @author    MCD Application Team  
5 * @version   V3.5.0  
6 * @date     08-April-2011  
7 * @brief    Main program body  
8 * *****  
9 * @attention  
10 *  
11 * THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS  
12 * WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE  
13 * TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY  
14 * DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING  
15 * FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS OF THE  
16 * CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS.  
17 *  
18 * <h2><center>© COPYRIGHT 2011 STMicroelectronics</center></h2>  
19 * *****  
20 */  
21  
22 /* Includes -----*/  
23 //USE_STDPERIPH_DRIVER, STM32F10X_HD, USE_STM3210B_EVAL  
24  
25 #include "stm32f10x.h"  
26 #include "stdio.h"  
27 #include "stdlib.h"  
28 #include "usart1.h"  
29 #include "utils.h"  
30 #include "EdpKit.h"  
31  
32  
33 #define API_KEY      "DX2cCKxgrpxZJKWFnbMzxIjeIfk=" //API KEY 需要修改为用户自己的对应参数  
34 #define DEV_ID       "1078702" //设备ID 需要修改为用户自己的对应参数  
35  
36
```

图 3-26 修改设备信息参数

2) 修改 WIFI 接入点的 ssid 与 pwd

在 esp8266.h 文件中，根据自己的网络环境，将参数修改为

```
#define CWJAP "AT+CWJAP=\"your ssid\", \"your pwd\"\r\n"
```

本例中 ssid 为 onenet, pwd 为空:

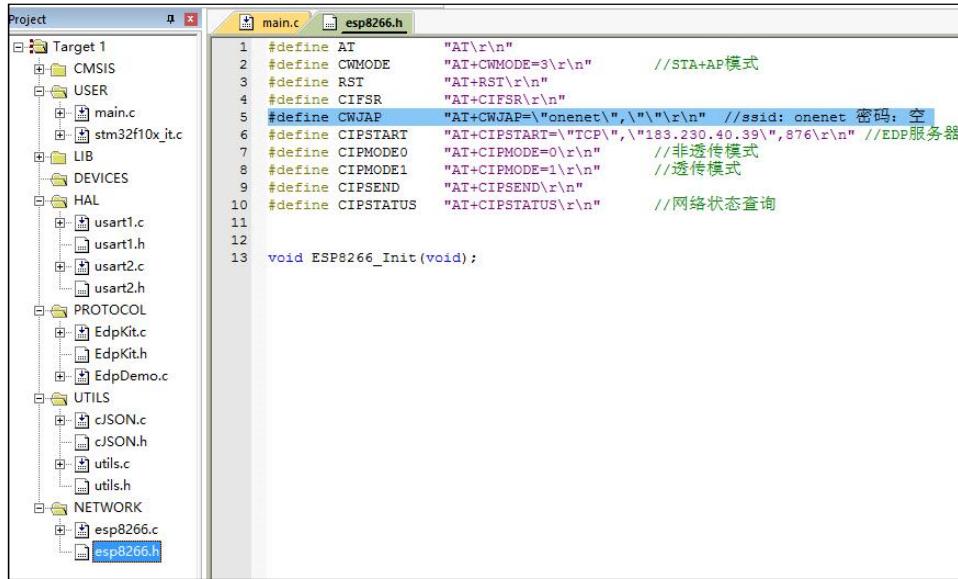


图 3-27 修改路由 ssid 和 pwd

Step4 编译整个工程，并下载程序

Step5 烧写程序完后，重新给开发板上电，建议每次烧写程序后都这样做，让 WIFI 模块完全掉电，以免 AT 命令操作 WIFI 失败。

Step6 程序运行后，打印如下：

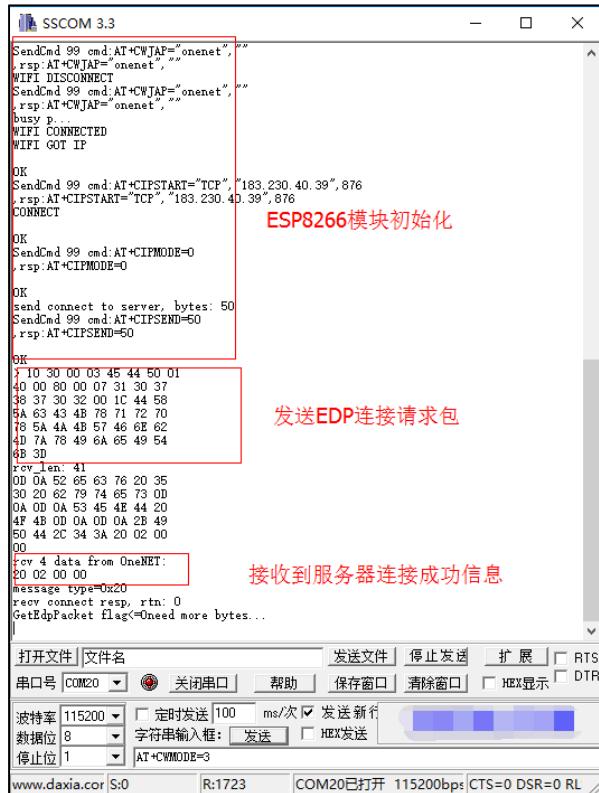


图 3-28 串口打印信息

Step7 软件模拟设备



打开工具目录下的 EdpProtoDebugger 软件，按照 Step1 中“透传测试设备”的参数填写相关配置，依次点击“生成编码”、“发送到设备云”，完成设备连接，界面如下所示：

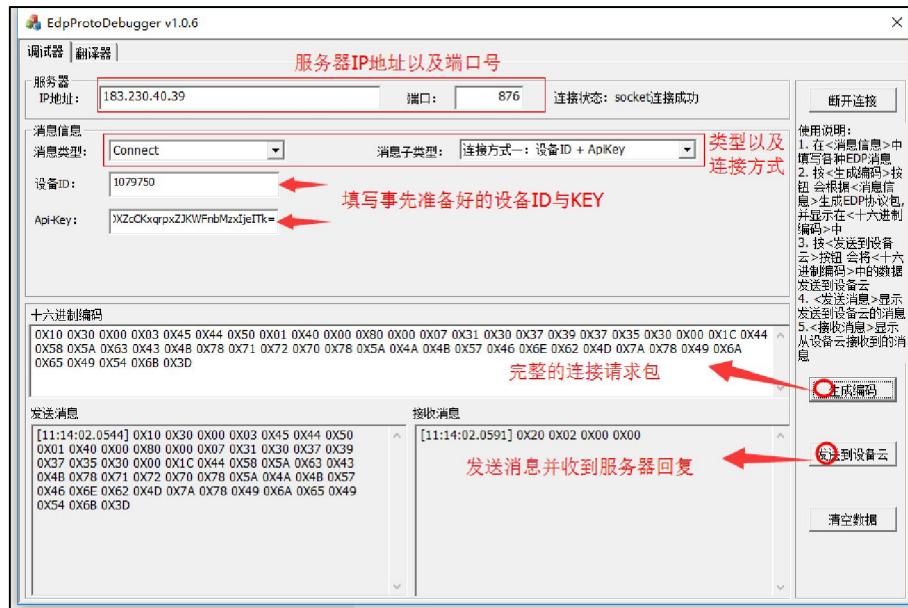


图 3-29 软件操作说明

Step8 数据转发操作

选择“消息类型”为 PushData（转发），目的设备 ID 置为开发板对应的设备 ID（本例中为 1078702），在“数据/路径”中填写需要转发的消息（本例中为 hello），点击“清空数据”清除上一步骤中的数据，依次点击“生成编码”、“发送到设备云”。

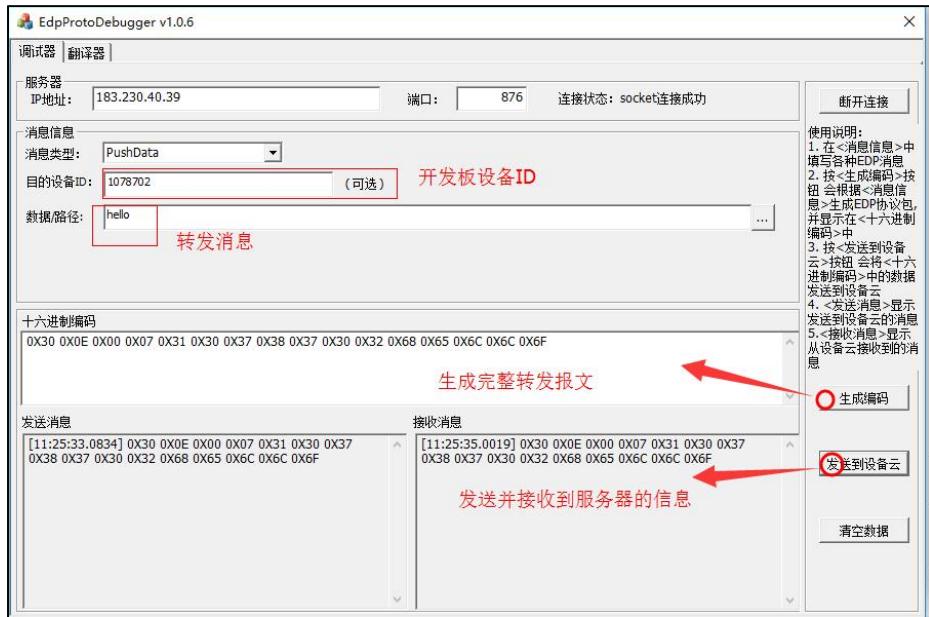


图 3-30 软件发送 push 报文并收到服务器的数据

开发板调试串口可以看到如下打印信息：



图 3-31 串口打印信息

开发板程序中，将收到的所有数据同样以转发的方式回发给源设备，所

以在 EdpProtoDebugger 中，可以看到收到来自服务器的信息。

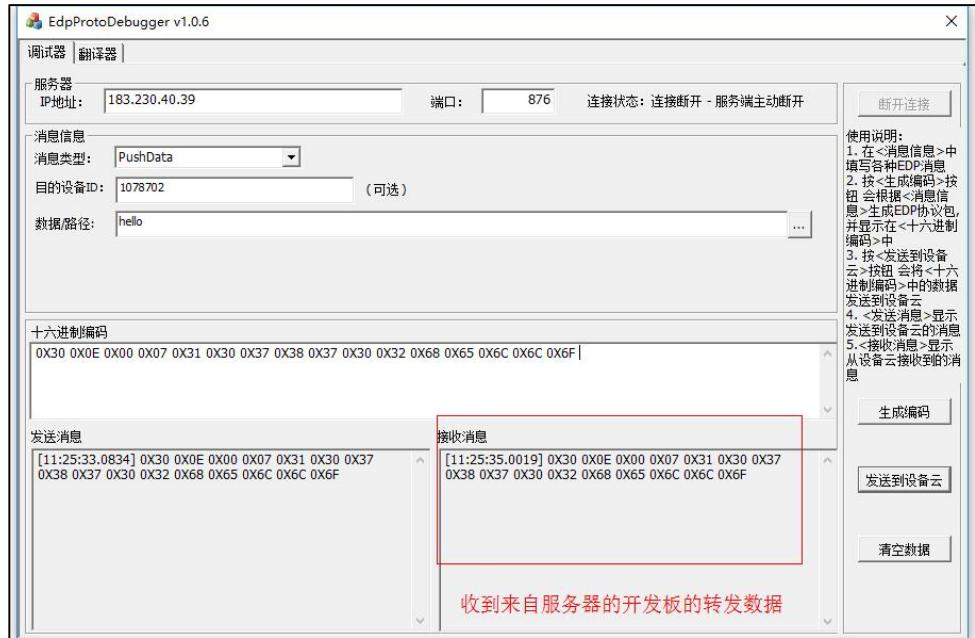


图 3-32 接收数据

3.5 ESP8266+HTTP 上传温湿度数据实验

代码目录

实验代码\OneNET_Demo_ESP8266_HTTP_SHT20

实验原理

使用开发板的 I2C 接口采集配套的 SHT20 温湿度传感器，并通过 ESP8266 与 HTTP 服务器建立 TCP 连接，利用 HTTP 协议将采集的数据传输到 OneNET 平台。

实验步骤

Step1 如果没有 EDP 设备，请先在 OneNET 平台创建 EDP 设备，具体步骤参考《2.1 接入流程概述》，本例中所用到的设备参数如下：



图 3-33 设备信息

Step2 硬件连接

- 1) 按照《麒麟开发板硬件手册》连接 SHT20 模块；
- 2) 连接 UART 调试串口，波特率 115200，8N1；
- 3) 连接 ST-LINK；
- 4) 连接 ESP8266WIFI 模组，请确保 J10 跳线连接到 ESP8266 一侧；
- 5) 接入电源，烧写程序前先上电。

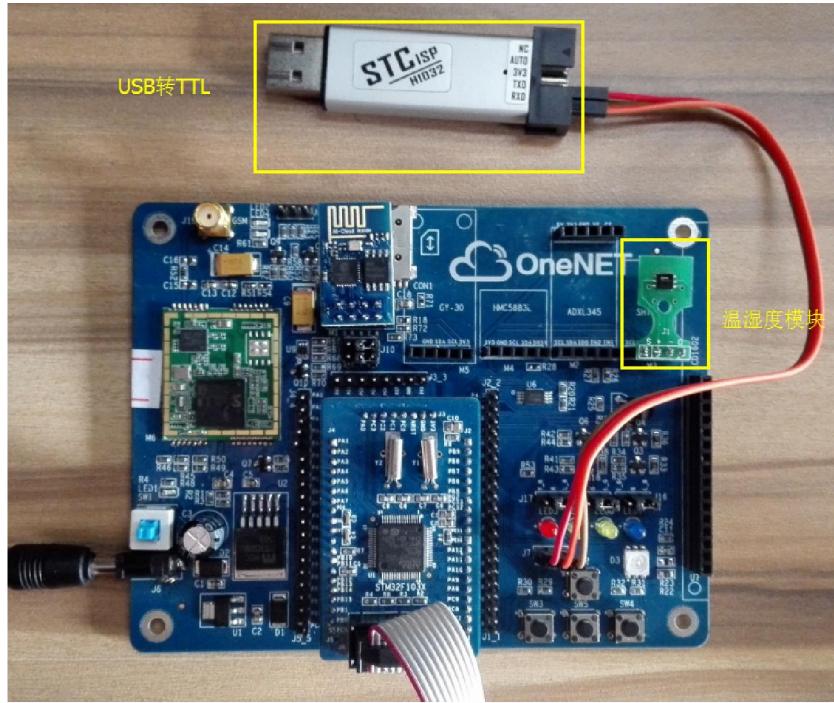


图 3-34 硬件连接

Step3 打开实验工程，目录：

实验代码\OneNET_Demo_ESP8266_HTTP_SHT20\Project\OneNETdemo.uvproj

1) 修改设备鉴权参数

在 main.c 文件中，根据自己的设备信息，修改 ApiKey 和 dev_id:

```

13 * TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY
14 * DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING
15 * FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS OF THE
16 * CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS.
17 *
18 * <h2><center>&copy; COPYRIGHT 2011 STMicroelectronics</center></h2>
19 ****
20 */
21
22 /* Includes -----
23 //USE_STDEPERIPH_DRIVER, STM32F10X_HD, USE_STM3210B_EVAL
24
25 #include "stm32f10x.h"
26 #include "stdio.h"
27 #include "stdlib.h"
28 #include "string.h"
29 #include "usart1.h"
30 #include "usart2.h"
31 #include "utils.h"
32 #include "sht20.h"
33 #include "hal_i2c.h"
34 #include "esp8266.h"
35
36
37 #define API_KEY "DXZccCKxqrpzZJKWFnbMzxIjeITk=" //需要定义为用户自己的参数
38 #define DEV_ID "1078702" //需要定义为用户自己的参数
39
40 /**
41 * @brief 利用I2C接口，采集温湿度传感器的值，使用HTTP协议上传到OneNET
42 * @attention 使用UART2连接ESP8266模块，使用透传模式发送和接收数据
43 * 使用UART1作为调试打印串口，使用printf将从该接口打印消息
44 *
45 */
46 int main(void)
47 {

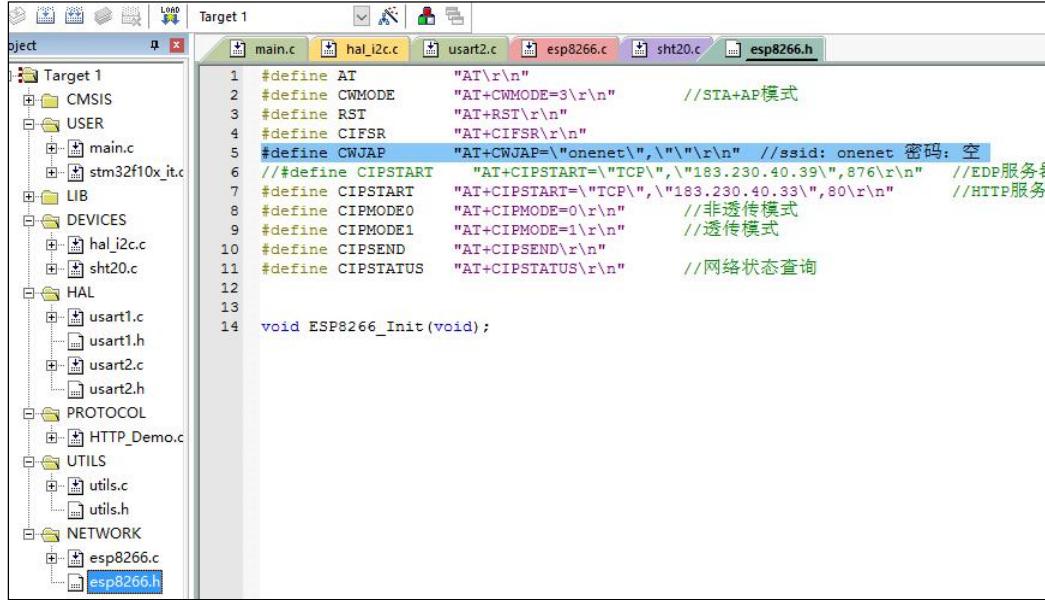
```

图 3-35 修改 ApiKey 和 dev_id

2) 修改 WIFI 接入点的 ssid 与 pwd

在 esp8266.h 文件中，根据自己的网络环境，将参数修改为

```
#define CWJAP      "AT+CWJAP=\"your_ssid\",\"your_pwd\"\r\n"
```



```
1 #define AT          "AT\r\n"
2 #define CWMODE      "AT+CWMODE=3\r\n"           //STA+AP模式
3 #define RST         "AT+RST\r\n"
4 #define CIFSR       "AT+CIFSR\r\n"
5 #define CWJAP        "AT+CWJAP=\"ononet\",\"\"\\r\\n\" //ssid: ononet 密码: 空
6 // #define CIPSTART   "AT+CIPSTART=\"TCP\",\"183.230.40.39\",876\\r\\n" //EDP服务器
7 #define CIPSTART    "AT+CIPSTART=\"TCP\",\"183.230.40.33\",80\\r\\n"      //HTTP服务
8 #define CIPMODE0     "AT+CIPMODE=0\\r\\n"           //非透传模式
9 #define CIPMODE1     "AT+CIPMODE=1\\r\\n"           //透传模式
10 #define CIPSEND      "AT+CIPSEND\\r\\n"
11 #define CIPSTATUS    "AT+CIPSTATUS\\r\\n"          //网络状态查询
12
13
14 void ESP8266_Init(void);
```

图 3-36 修改 ssid 和 pwd

Step4 编译整个工程，并下载程序；

Step5 烧写程序完后，重新给开发板上电，建议每次烧写程序后都这样做，让 WIFI 模块完全掉电，以免 AT 命令操作 WIFI 失败。

Step6 程序运行后，调试串口会有如下打印信息：

SSCOM 3.3

```

WIFI CONNECTED
WIFI GOT IP

OK
SendCmd 99 cmd: AT+CIPSTART="TCP", "183.230.40.33", 80
, rsp: AT+CIPSTART="TCP", "183.230.40.33", 80
CONNECT

OK
SendCmd 99 cmd: AT+CIPMODE=1
, rsp: AT+CIPMODE=1

OK
SendCmd 99 cmd: AT+CIPSEND
, rsp: AT+CIPSEND

OK
>ESP8266 init over
SHT2x_MeasureRH data[0]=101, data[1]=228, checksum=0, t=23.088362
SHT2x_MeasureRH 23
SHT2x_MeasureRH data[0]=140, data[1]=42, checksum=0, rh=62.435719
SHT2x_MeasureRH 62
send HTTP msg
POST /devices/1078702/datapoints?type=5 HTTP/1.1
api-key DKZcKscrpxzJKWFnbMrxIjeITk=
Host: api.heclouds.com
Content-Length: 9
:temp,23
recv response
HTTP/1.1 200 OK
Date: Tue, 05 Apr 2016 12:24:16 GMT
Content-Type: application/json
Content-Length: 26
Connection: keep-alive
Server: Apache-Coyote/1.1
Pragma: no-cache
{"errno":0,"error":"succ"}
send HTTP msg

```

串口号: COM20 波特率: 115200 数据位: 8 停止位: 1

打开文件 | 文件名: www.daxia.cor | 发送文件 | 停止发送 | 扩展 | RTS | DTR | HEX显示 | DTR | HEX显示 | DTR

字符串输入框: 发送 | HEX发送 | HEX发送

AUXL345 Z

ESP8266初始化

读取温湿度值

发送POST, 上传数据

接收服务上传成功应答

图 3-37 串口打印信息

Step7 OneNET 平台下查看上传的数据。

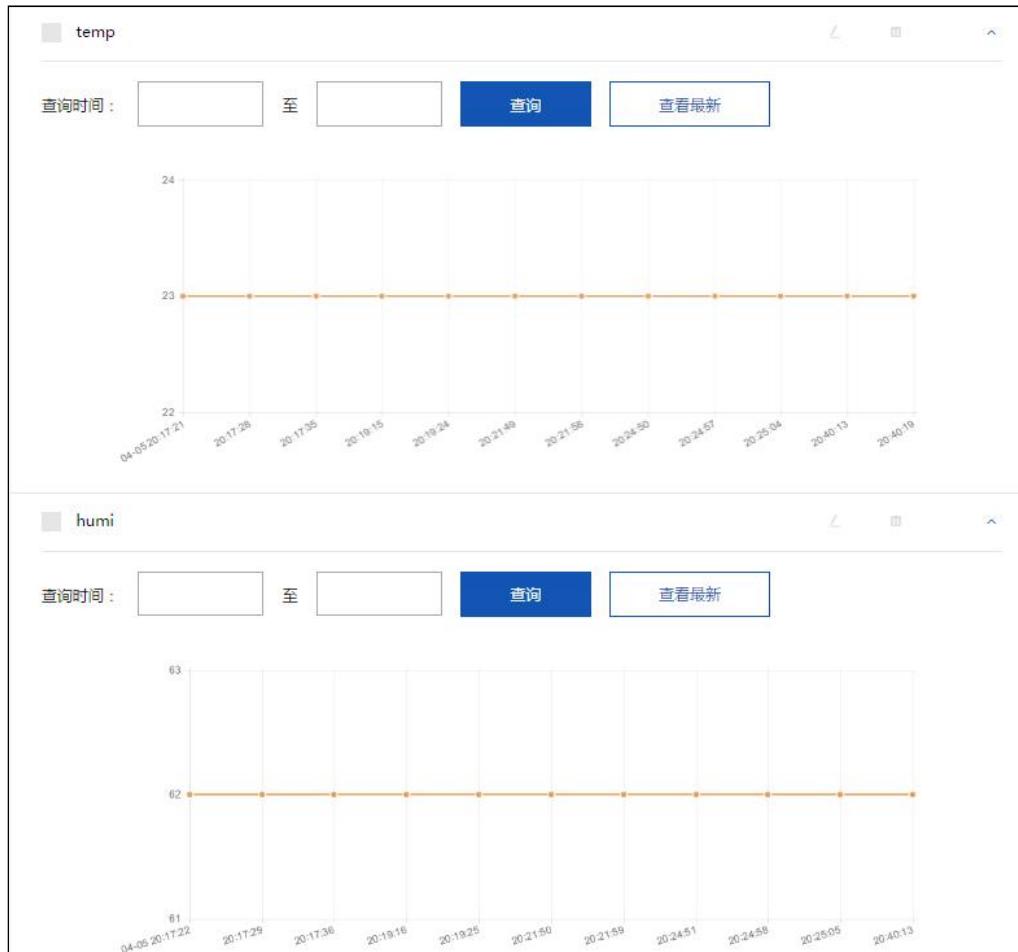


图 3-38 查看结果

3.6 GPRS+EDP 采集温湿度上传实验

代码目录

实验代码\OneNET_Demo_M6311_EDP_SHT20

实验原理

使用开发板的 M6311 GPRS 模块与 EDP 服务器建立 TCP 连接，通过 I2C 接口采集 SHT20 温湿度传感器，使用 EDP 协议将数据上传到 OneNET。

实验步骤

Step1 如果没有 EDP 设备，请先在 OneNET 平台创建 EDP 设备，具体步骤参考《2.1 接入流程概述》，本例中所用到的设备参数如下：



图 3-39 EDP 设备创建

Step2 在上一步中创建的 EDP 设备下，为每个传感器的采集结果创建数据流，如果已经创建请跳过此步；

hum	最新数据： 62	更新时间： 2016-04-06 16:55:43
temperature	最新数据： 24	更新时间： 2016-04-06 16:55:43

图 3-40 EDP 设备数据流创建

Step3 硬件连接

- 1) 按照《麒麟开发板硬件手册》说明连接上 SHT20 模块；
- 2) 连接 UART 调试串口，波特率 115200，8N1；
- 3) 连接 ST-LINK；
- 4) 使用板载 M6311 GPRS 模组（如果有），请确保 J10 跳线连接到 GPRS

一侧；

5) 接入电源，烧写程序前先上电。

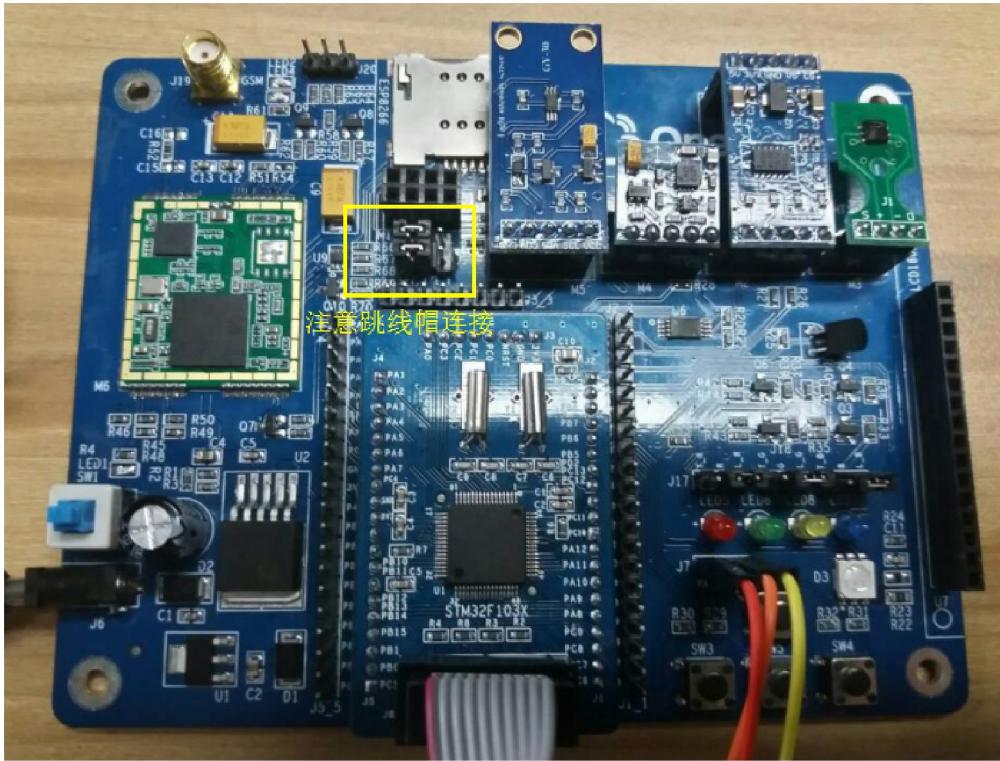


图 3-41 开发板连接

Step4 打开实验工程，目录：

实验代码\OneNET_Demo_M6311_EDP_SHT20\Project\OneNETdemo.uvproj

Step5 找到源文件 EdpDemo.c，修改：

```
int8_t src_api_key[] = "your_api_key"; //修改成读者设备对应的 ApiKey  
int8_t src_dev[] = "your_device_id"; //修改成读者对应的设备 ID
```

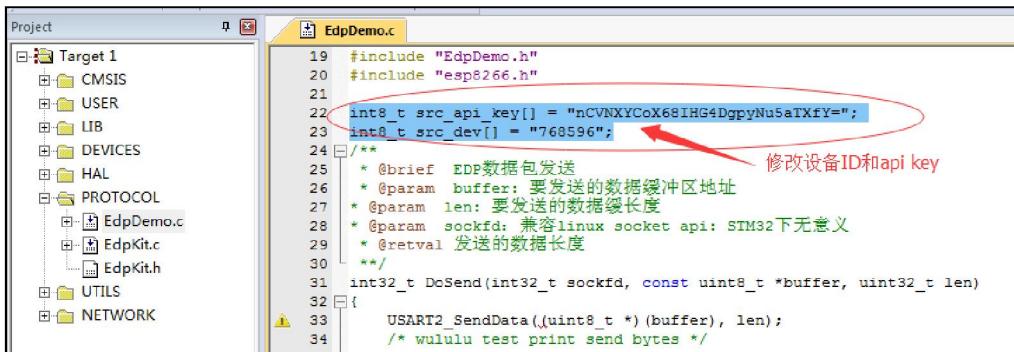


图 3-42 修改 ApiKey 和 dev_id

Step6 编译整个工程，并下载程序，请保证 JATG 和电源已正确连接。

Step7 烧写程序完后，重新给开发板上电，建议每次烧写程序后都这样做，让

GPRS 模块完全掉电，以免 AT 命令操作 GPRS 失败。

Step8 程序运行后， 打印如下：

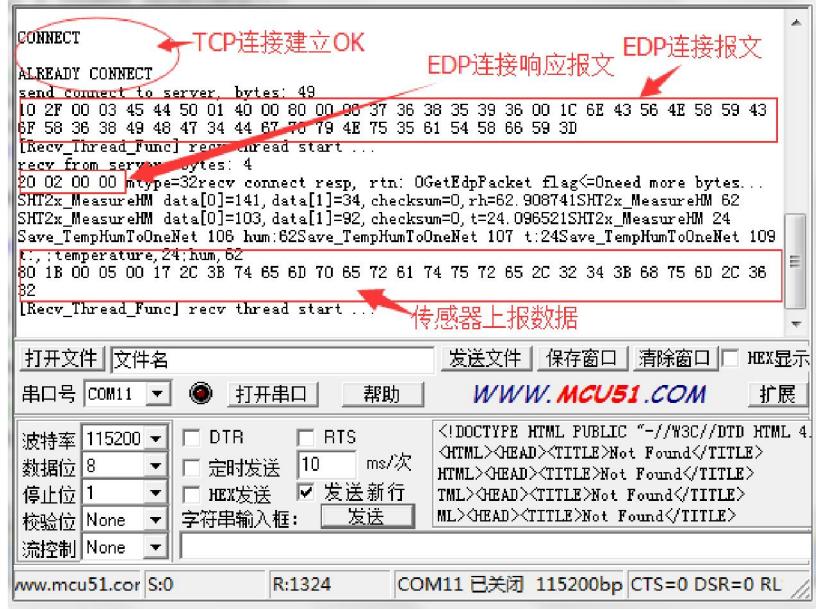


图 3-43 EDP 连接和数据上报打印

Step9 OneNET 平台下查看上传的数据。



图 3-44 EDP 连接和数据上报打印

3.7 ESP8266+Modbus 协议接入和数据上传实验

代码目录

实验代码\OneNET_Demo_ESP8266_MODBUS

实验原理

使用 ESP8266 与 ModBus 服务器建立 TCP 连接，使用开发板实现 ModBus 协议，实验采用 RTU 帧格式，模拟了 ModBus 设备登录和 OneNET 平台周期性发布命令 0x3。

实验步骤

Step1 如果没有 MODBUS 设备，请先在 OneNET 平台创建 ModBus 设备和对应数据流，具体步骤参考本章 2.1 节，ModBus 设备创建和普通设备有一定差异，具体参考 RESTful API 和 ModBus 接入协议文档，注意数据流要包含 interval 和 cmd；本例 cmd 将以 10 秒为间隔向设备发送 0x3 命令，命令要求设备返回从 0x006b 开始的 3 个寄存器值。



图 3-45 ModBus 设备详情

Step2 硬件连接

- 1) 连接 UART 调试串口，波特率 115200，8N1；

- 2) 连接 ST-LINK;
- 3) 连接 ESP8266 WIFI 模组, 请确保 J10 跳线连接到 ESP8266 一侧;
- 4) 接入电源, 烧写程序前先上电。

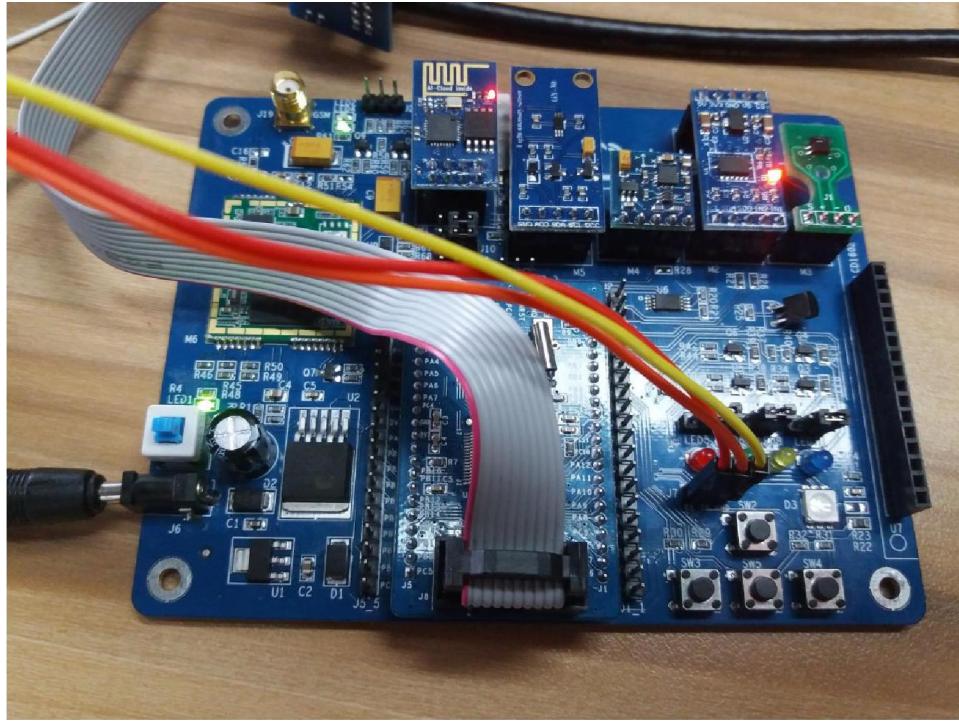


图 3-46 开发板连接

Step3 打开实验工程, 目录:

实验代码\OneNET_Demo_ESP8266_MODBUS\Project\OneNETdemo.uvproj

Step4 修改代码, 主要是修改 ESP8266 的热点配置和相关设备信息。如下图在 esp8266.h 头文件中修改 CWJAP 宏定义为读者使用的路由器 ssid 和 pwd。

```
#define CWJAP      "AT+CWJAP=\"your_ssid\";\"pwd\"\r\n"
```

源代码的 ssid 为 onenet, pwd 为空。

```

Project
  Target 1
    CMSIS
    USER
    LIB
    DEVICES
    HAL
    PROTOCOL
    UTILS
    NETWORK
      esp8266.c
      esp8266.h

esp8266.h
6 #include "stdlib.h"
7 #include "string.h"
8 #include "uart1.h"
9 #include "uart2.h"
10 #include "utils.h"
11
12 #define AT      "AT\r\n"
13 #define CWMODE  "AT+CWMODE=3\r\n" 修改路由器SSID和密码
14 #define RST    "AT+RST\r\n"
15 #define CIFSR  "AT+CIFSR\r\n"
16 #define CWJAP  "AT+CWJAP=\"onet\";\"\"\r\n\""
17 #define CIPSTART "AT+CIPSTART=\"TCP\", \"183.230.40.39\", 876\r\n"
18 #define MODBUS_CIPSTART "AT+CIPSTART=\"TCP\", \"183.230.40.42\", 987\r\n"
19 #define MQTT_CIPSTART  "AT+CIPSTART=\"TCP\", \"183.230.40.39\", 6002\r\n"
20 #define EDP_CIPSTART  "AT+CIPSTART=\"TCP\", \"183.230.40.39\", 876\r\n"
21 #define CIPMODE   "AT+CIPMODE=1\r\n"
22 #define CIPSEND   "AT+CIPSEND\r\n"
23 #define CIPSTATUS "AT+CIPSTATUS\r\n"
24

```

图 3-47 路由器 ssid 和 pwd 修改

找到源文件 modbus.c 中的 MODBUS_Init_Login_Data 函数，这些信息要和创建 ModBus 设备时填写的内容一致，修改：

```
int8_t phone[] = "your_phone_id";      //卡号信息
int8_t pwd[] = "your_pwd";            //密码信息
int8_t p_id[] = "your_project_id";    //项目 ID
```

```
Project: [modbus]
Target 1:
  CMSIS
  USER
  LIB
  HAL
  PROTOCOL
    modbus.c
    modbus.h
  UTILS
  NETWORK

modbus.c
25 */
26 void MODBUS_Init_Login_Data(void)
27 {
28     int32_t i = 0;
29     int8_t type[11] = "type"; /*暂时不支持，服务器不处理*/
30     int8_t name[9] = "name"; /*暂时不支持，服务器不处理*/
31     int8_t phone[12] = "12345670"; /*创建MODBUS设备时对应的卡号信息，鉴权信息*/
32     int8_t pwd[9] = "12345670"; /*创建MODBUS设备时对应的密码信息，鉴权信息*/
33     int8_t p_id[11] = "39484"; /*设备所在项目ID*/
34     memset(req, 0, 52);
35     memcpy(req, type, 11);
36     memcpy(req + 11, name, 9);
37     memcpy(req + 11 + 9, phone, 12);
38     memcpy(req + 11 + 9 + 12, pwd, 9);
39     memcpy(req + 11 + 9 + 12 + 9, p_id, 11);
```

图 3-48 MODBUS 鉴权信息修改

Step5 编译整个工程，并下载程序，请保证 JATG 和电源已正确连接。

Step6 烧写程序完后，重新给开发板上电，建议每次烧写程序后都这样做，让 WIFI 模块完全掉电，以免 AT 命令操作 WIFI 失败。

Step7 程序运行后，打印如下：



图 3-49 ModBus 登录和命令处理打印

Step8 OneNET 平台下查看上传的数据，开发板随机回复 6 个字节的数据。

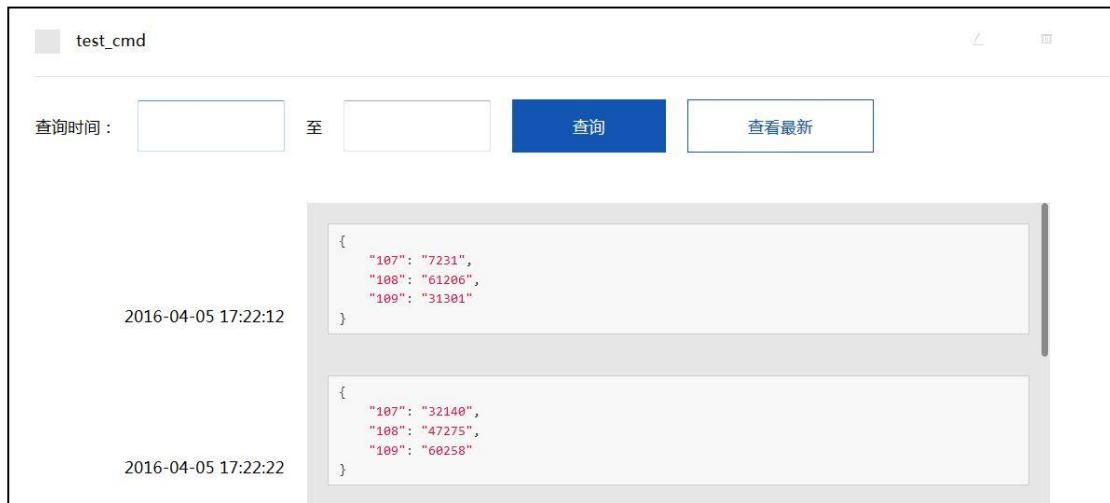


图 3-50 ModBus 上传的随机数据

3.8 ESP8266+MQTT 协议发布订阅实验

代码目录

实验代码\OneNET_Demo_ESP8266_MQTT

实验原理

使用 2 个 MQTT 设备演示消息发布和订阅功能，其中一个设备是移植了 MQTT 协议的开发板，作为消息发布者，共发布三个数据流，另一个是 ubuntu（或者虚拟机），运行 mqttc 命令作为消息订阅者，订阅主题仅为开发板设备的数据流 mqtt_msg。开发板上的 ESP8266 与 MQTT 服务器建立 TCP 连接，利开发板向 MQTT 服务器发布三个数据流的数据点后，在 ubuntu 下观察接收打印，仅收到订阅过的 mqtt_msg 消息。

实验步骤

Step1 为了测试发布和订阅功能，读者需要 2 个 MQTT 设备，创建方法和 EDP 设备类似，一个用于 ubuntu 登录作为订阅者(ID 773123)，一个用于开发板登录作为发布者(ID 769832)。



图 3-51 MQTT 设备信息



图 3-52 MQTT 设备数据流信息

Step2 硬件连接

- 1) 按照《麒麟开发板硬件手册》说明连接上 SHT20 模块；
- 2) 连接 UART 调试串口，波特率 115200，8N1；
- 3) 连接 ST-LINK；
- 4) 连接 ESP8266 WIFI 模组，请确保 J10 跳线连接到 ESP8266 一侧；
- 5) 接入电源，烧写程序前先上电。

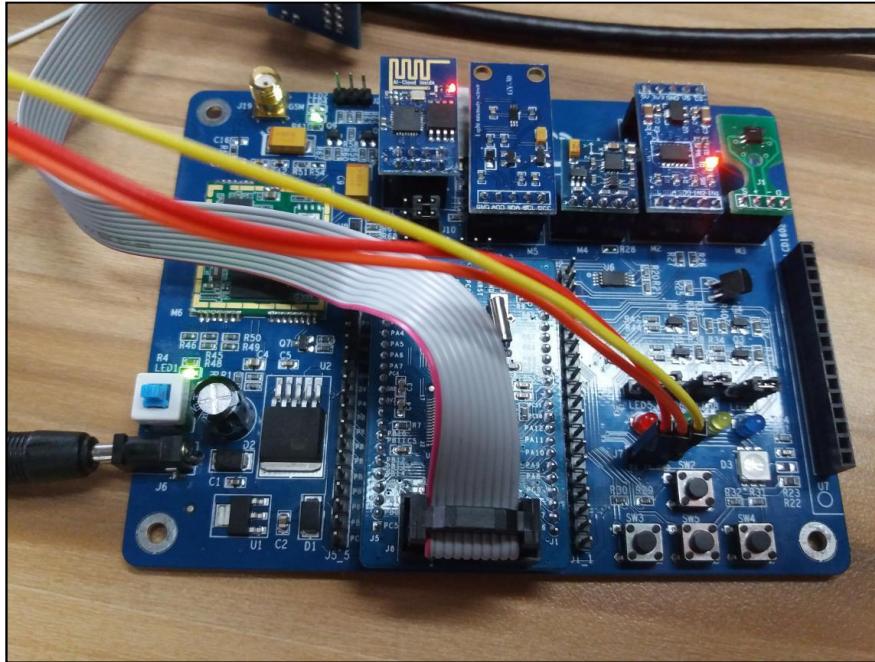


图 3-53 开发板连接

Step3 打开实验工程，目录：

实验代码\OneNET_Demo_ESP8266_MQTT\Project\OneNETdemo.uvproj

1) 首先开始开发板端的代码修改，修改 ESP8266 WIFI 联网配置，如下图在 esp8266.h 头文件中修改 CWJAP 宏定义为读者使用的路由器 ssid 和 pwd，如：

```
#define CWJAP      "AT+CWJAP=\"your_ssid\",\"pwd\"\r\n"
```

源代码的 ssid 为 onenet，pwd 为空。

```

Project Target 1
  CMSIS
  USER
  LIB
  DEVICES
  HAL
  PROTOCOL
  UTILS
  NETWORK
    esp8266.c
    esp8266.h

esp8266.h
6 #include "stdlib.h"
7 #include "string.h"
8 #include "usart1.h"
9 #include "uart2.h"
10 #include "utils.h"
11
12 #define AT      "AT\\r\\n"
13 #define CWMODE  "AT+CWMODE=3\\r\\n" 修改路由器SSID和密码
14 #define RST    "AT+RST\\r\\n"
15 #define CIFSR  "AT+CIFSR\\r\\n"
16 #define CWJAP  "AT+CWJAP=\"ononet\",\"\"\\r\\n\""
17 #define CIPSTART "AT+CIPSTART=\"TCP\",\\\"183.230.40.39\\\",876\\r\\n"
18 #define MODBUS_CIPSTART "AT+CIPSTART=\"TCP\",\\\"183.230.40.42\\\",987\\r\\n"
19 #define MQTT_CIPSTART "AT+CIPSTART=\"TCP\",\\\"183.230.40.39\\\",6002\\r\\n"
20 #define EDP_CIPSTART "AT+CIPSTART=\"TCP\",\\\"183.230.40.39\\\",876\\r\\n"
21 #define CIPMODE  "AT+CIPMODE=i\\r\\n"
22 #define CIPSEND  "AT+CIPSEND\\r\\n"
23 #define CIPSTATUS "AT+CIPSTATUS\\r\\n"
24

```

图 3-54 路由器 ssid 和 pwd 修改

(2) 修改 MQTT 设备信息，读者替换对应的项目 ID，设备 ID 和 ApiKey。

```

#define MQTT_DEVICE_PROJ_ID "39484" //项目 ID
#define MQTT_DEVICE_ID "769832"      //设备 ID
#define MQTT_DEVICE_API_KEY "nCVNXYCoX68IHG4DgpyNu5aTXfY="      //ApiKey

```

```

Project mqtt_loop.c mqtt_loop.h
  USER
  LIB
  HAL
  PROTOCOL
    config.h
    mqtt.c
    mqtt.h
    mqtt_buffer.c
    mqtt_buffer.h
    mqtt_loop.c
    mqtt_loop.h
  UTILS
  NETWORK
  DEVICES

mqtt_loop.c
23 #define DS_TO_PUBLISH_T "mqtt_msg_t"
24 #define DS_TO_PUBLISH_RH "mqtt_msg_rh"
25
26 #define TOPIC_TO_SUB "39484/nCVNXYCoX68IHG4DgpyNu5aTXfY=/769243/da_test_a"
27 #define PACK_FALG_UNSUB 11
28 #define TOPIC_TO_UNSUB "39484/nCVNXYCoX68IHG4DgpyNu5aTXfY=/769243/da_test_a"
29
30 #define TIME_OUT 1
31 #define EVENT_2
32
33 #define MQTT_DEVICE_PROJ_ID "39484"
34 #define MQTT_DEVICE_ID "769832"
35 #define MQTT_DEVICE_API_KEY "nCVNXYCoX68IHG4DgpyNu5aTXfY="
36
37 void MQTT_Loop(void);
38
39
40

```

图 3-55 MQTT 设备信息修改

(3) 修改数据流 ID（如有需要），开发板发布数据用的数据流 id 如下：

```

Project mqtt_loop.c mqtt_loop.h
  USER
  LIB
  HAL
  PROTOCOL
    config.h
    mqtt.c
    mqtt.h
    mqtt_buffer.c
    mqtt_buffer.h
    mqtt_loop.c
    mqtt_loop.h
  UTILS
  NETWORK
  DEVICES

mqtt_loop.c
14 #include <stdio.h>
15 #include <stdlib.h>
16 #include <string.h>
17 #include <assert.h>
18 #include <time.h>
19
20 #include "uart2.h"
21
22 #define DS_TO_PUBLISH "mqtt_msg"
23
24 #define DS_TO_PUBLISH_T "mqtt_msg_t"
25
26 #define DS_TO_PUBLISH_RH "mqtt_msg_rh"
27
28 #define TOPIC_TO_SUB "39484/nCVNXYCoX68IHG4DgpyNu5aTXfY=/769243/da_test_a"
29 #define PACK_FALG_UNSUB 11
30 #define TOPIC_TO_UNSUB "39484/nCVNXYCoX68IHG4DgpyNu5aTXfY=/769243/da_test_a"
31

```

图 3-56 MQTT 设备数据流修改

Step4 编译整个工程，并下载程序，请保证 JATG 和电源已正确连接。

Step5 烧写程序完后，重新给开发板上电，建议每次烧写程序后都这样做，让 WIFI 模块完全掉电，以免 AT 命令操作 WIFI 失败。

Step6 程序运行后，打印如下：

```
device id:769832  
we publish topic:mqtt_msg mqtt_msg_t mqtt_msg_rh  
connect to mqtt server  
MqttSample_CmdConnect 270  
*****bytes=87*****  
0x10 0x55 0x0 0x4 0x4d 0x51 0x54 0x54 0x4 0x6 0x0 0x0 0x6 0x37 0x36 0x39 0x36  
0x33 0x32 0x0 0x9 0x57 0x89 0x6c 0x6c 0x54 0x6f 0x70 0x69 0x63 0x0 0x11 0x77 0x69  
0x6c 0x6c 0x20 0x6d 0x65 0x73 0x73 0x61 0x67 0x65 0x24 0x74 0x65 0x73 0x74 0x0 0x5  
0x33 0x39 0x34 0x38 0x34 0x0 0x1c 0x6e 0x43 0x56 0x4e 0x58 0x59 0x43 0x6f 0x58 0x36  
0x38 0x49 0x48 0x47 0x34 0x44 0x67 0x70 0x79 0x4e 0x75 0x35 0x61 0x54 0x58 0x66 0x59  
0x3d  
*****  
bytes=4  
0x20 0x2 0x0 0x0  
Success to connect to the server, flags(0), code(0).
```

打开文件 | 文件名 | 发送文件 | 保存窗口 | 清除窗口 | HEX显示
串口号 | COM11 | 打开串口 | 帮助 | WWW.MCU51.COM | 扩展
波特率 115200 | DTR | RTS |
数据位 8 | 定时发送 10 ms/次 |
停止位 1 | HEX发送 | 发送新行 |
校验位 None | 字符串输入框: |
流控制 None |
www.mcu51.com S:0 R:1726 COM11 已关闭 115200bp CTS=0 DSR=0 RL

图 3-57 MQTT 设备登录连接打印

发布数据打印：

```
*****bytes=27*****  
0x34 0x19 0x0 0x8 0x6d 0x71 0x74 0x74 0x5f 0x6d 0x73 0x67 0x0 0x1 0x84 0xe8 0xae  
0xbe 0xe5 0xa4 0x87 0xe7 0xa6 0xbb 0xe4 0xbd 0x8d  
*****  
bytes=4  
0x50 0x2 0x0 0x1  
Recv the publish rec, packet id is 1. Mqtt_PackPubRelPkt 1278  
*****bytes=4*****  
0x62 0x2 0x0 0x1  
*****  
bytes=4  
0x70 0x2 0x0 0x1  
Recv the publish comp, packet id is 1. MqttSample_HandlePubComp 186!!!!!!!!!!!!!!  
打开文件 | 文件名 | 发送文件 | 保存窗口 | 清除窗口 |  HEX显示  
串口号 | COM11 | 打开串口 | 帮助 | WWW.MCU51.COM | 扩展  
波特率 115200 |  DTR |  RTS |  
数据位 8 |  定时发送 10 ms/次 |  
停止位 1 |  HEX发送 |  发送新行 |  
校验位 None | 字符串输入框: |  
流控制 None |  
www.mcu51.com S:0 R:2635 COM11 已关闭 115200bp CTS=0 DSR=0 RL
```

图 3-58 MQTT 设备发布数据打印

Step7 OneNET 平台下查看上传的数据。



图 3-59 MQTT 设备发布数据流信息

Step8 修改 ubuntu 端的代码，在 ubuntu 上解压 mqtt_app.tar.gz，修改设备信息和订阅主题。

- 1) Mqtt.c 中修改 ctx->proid 为读者设备对应的项目 ID;
- 2) Mqtt.c 修改 ctx->devid 为读者设备 ID;
- 3) Mqtt.c 修改 ctx->apikey 为读者设备的 ApiKey;

```
static int MqttSample_Init(struct MqttSampleContext *ctx)
{
    struct epoll_event event;
    int err;

    ctx->sendedbytes = -1;
    ctx->mqttd = -1;
    ctx->host = "183.230.40.39";
    ctx->port = 6002;
    ctx->proid = "39484";
    ctx->devid = "773123";
    ctx->apikey = "nCVNXYCoX68IHG4DgpyNu5aTXY=";

    err = Mqtt_InitContext(ctx->mqttd, 1 << 20);
    if(MQTTERR_NOERROR != err) {
        printf("Failed to init the MQTT context errcode is %d", err);
        return -1;
    }
}
```

图 3-60 MQTT 设备登录信息修改

- 4) 在 Mqtt.c 中 MqttSample_CmdSubscribe 函数内替换读者对应的项目 ID/APIKEY/设备 ID/订阅数据流 ID。

```
static int MqttSample_CmdSubscribe(struct MqttSampleContext *ctx)
{
    int err;
    err = Mqtt_PackSubscribePkt(ctx->mqttd, 1, "39484/nCVNXYCoX68IHG4DgpyNu5aTXY=/769832/mqtt_msg", MQTT_QOS_LEVEL1);
    if(err != MQTTERR_NOERROR) {
        printf("Critical bug: failed to pack the subscribe packet.\n");
        return -1;
    }

    return 0;
}
```

图 3-61 MQTT 设备订阅主题修改

Step9 进入 MQTT-master/mqtt_app 输入 make 编译生成 mqttc，运行./mqttc 效果如下：

```
Commands:
  connect      Establish the connection.
  ping         Send ping packet.
  publish      Send data points.
  subscribe    Subscribe the data streams.
  unsubscribe  Unsubscribe the data streams.
  disconnect   Close the connection.
  cmdret       Sed the command return information.
  exit         Exit the sample.
  help         Print the usage of the commands.
```

图 3-62 ubuntu MQTT 设备交互命令

依次输入 connect 登录和 subscribe 进行订阅。

Step10 由于麒麟开发板上的 MQTT 发布者已经运行了，当 ubuntu 登录和订阅成功后，将收到麒麟开发板发布的数据。如下，可见消息的 topic 是 769832/mqtt_msg，与开发板上配置一致：

```
-----
0x32 0x20 0x0 0xf 0x37 0x36 0x39 0x38 0x33 0x32 0x2f 0x6d 0x71 0x74 0x74 0x5f 0x
6d 0x73 0x67 0x0 0x2 0x84 0xe8 0xae 0xbe 0xe5 0xa4 0x87 0xe5 0x9c 0xa8 0xe4 0xbd
0x8d
-----
dup=0, qos=1, id=2
topic: 769832/mqtt_msg
payloadsize=13

*****
0x40 0x2 0x0 0x2
*****
send bytes:4
-----
-----
```

图 3-63 ubuntu MQTT 设备接收到麒麟开发板发布的 mqtt_msg 信息

Step11 输入 unsubscribe 可取消订阅，不再接收开发板发布的消息。

3.9 EMW3081+EDP 发送二进制文件（可选）

代码目录

实验代码\OneNET_Demo_EMW3081_EDP_Bin

实验原理

WIFI 模块 EMW3081（需购买）集成了 OneNET 的接入协议并提供 AT 指令接口，用户可以在不了解平台接入协议细节的情况下，只需要使用 AT 指令（指令具体使用方法，请参考 OneNET-AT 指令使用手册），就可以实现数据的上传，本实验利用 AT 指令实现二进制文件的上传。

实验步骤

Step1 硬件连接

- 1) 连接 UART 调试串口，波特率 115200，8N1；
- 2) 连接 ST-LINK；
- 3) 连接 EMW3081 模组；
- 4) 接入电源，烧写程序前先上电。

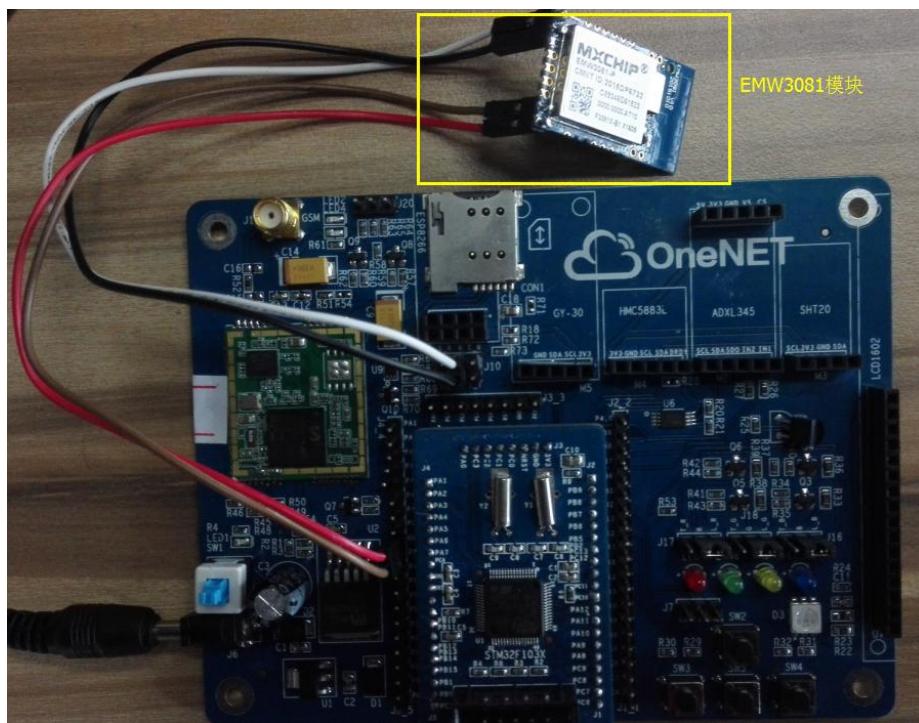


图 3-64 开发板连接

Step2 打开实验工程，目录：

实验代码\OneNET_Demo_EMW3081_EDP_Bin\Project\OneNETdemo.uvproj。

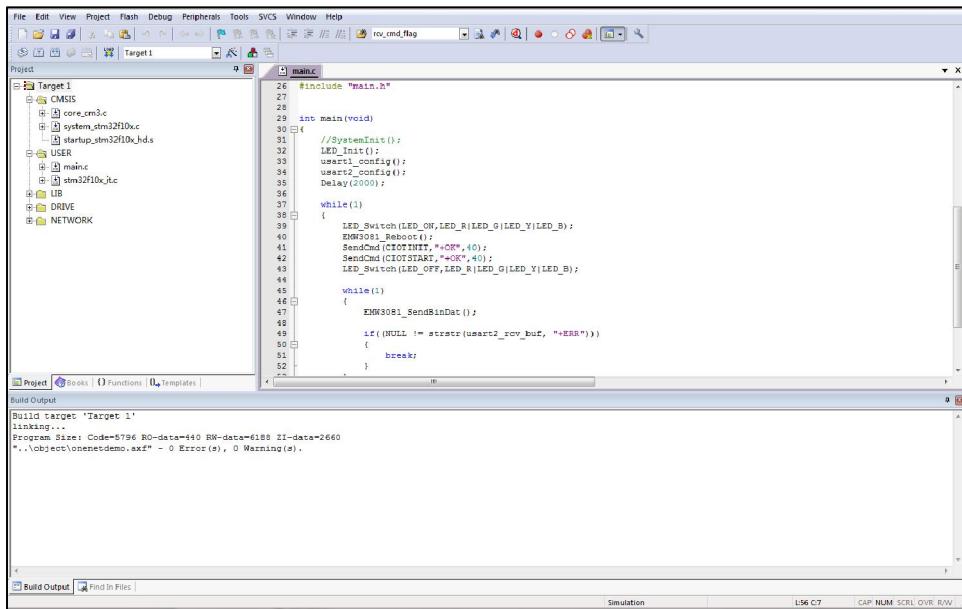


图 3-65 打开 Keil 工程

Step3 修改代码

主要是修改 EMW3081 的热点配置和项目 ApiKey 信息。

如下图，在 emw3081.h 头文件中修改 CIOTMKEY 宏定义为读者使用的项目 ApiKey：

```
#define CIOTMKEYA "AT+CIOTMKEYA=Sezt7lZFiwVUcYZeWLZM91kOMBw=\r" //修改 ApiKey
main.c emw3081.h
1 #ifndef EMW3081_H_H
2 #define EMW3081_H_H
3
4
5 #define REBOOT      "AT+REBOOT\r"
6 #define NSTATUS     "AT+WSTATUS\r"
7 #define CIOT        "AT+CIOT?\r"
8 #define CIOTMKEY    "AT+CIOTMKEY?\r"
9 #define CIOTMKEYA   "AT+CIOTMKEYA=Sezt7lZFiwVUcYZeWLZM91kOMBw=\r"
10 #define CIOTID     "AT+CIOTID?\r"
11 #define CIOTIDC    "AT+CIOTIDC?\r"
12 #define CIOTINIT   "AT+CIOTINIT?\r"
13 #define CIOKEY     "AT+CIOTKEY?\r"
14 #define CIOKEYC    "AT+CIOTKEYC?\r"
15 #define CIOTSTATUS "AT+CIOTSTATUS?\r"
16 #define CIOTSTART  "AT+CIOTSTART=\r"
17 #define CIOTDAT   "AT+CIOTDAT=TEMP3,,20\r"
18 #define CIOTSTATUS "AT+CIOTSTATUS?\r"
19
20 #define CIOTBINSET "AT+CIOTBINSET=PIC4,5636\r"
21
22 #define CIOTBINEND "AT+CIOTBINEND\r"
23
24 extern unsigned char dat1[1024];
25 extern unsigned char dat2[1024];
26 extern unsigned char dat3[1024];
27 extern unsigned char dat4[1024];
28 extern unsigned char dat5[1024];
29 extern unsigned char dat6[1024];
30
31 extern void SendCmd(char* cmd, char* result, int timeOut);
32 extern void EMW3081_Reboot(void);
33 extern void EMW3081_SendBinData(void);
34
35 #endif
36
```

图 3-66 修改主 ApiKey



Step4 修改 EMW3081 的热点配置，双击 **SecureCRT Portable.exe - 快捷方式**，打开串口工具软件，将电脑和 EMW3081 进行连接，发送 AT 指令“AT+WMODE=AP”使模块进入 AP 模式，如下图所示，然后使用“AT+SAVE”进行保存设备。

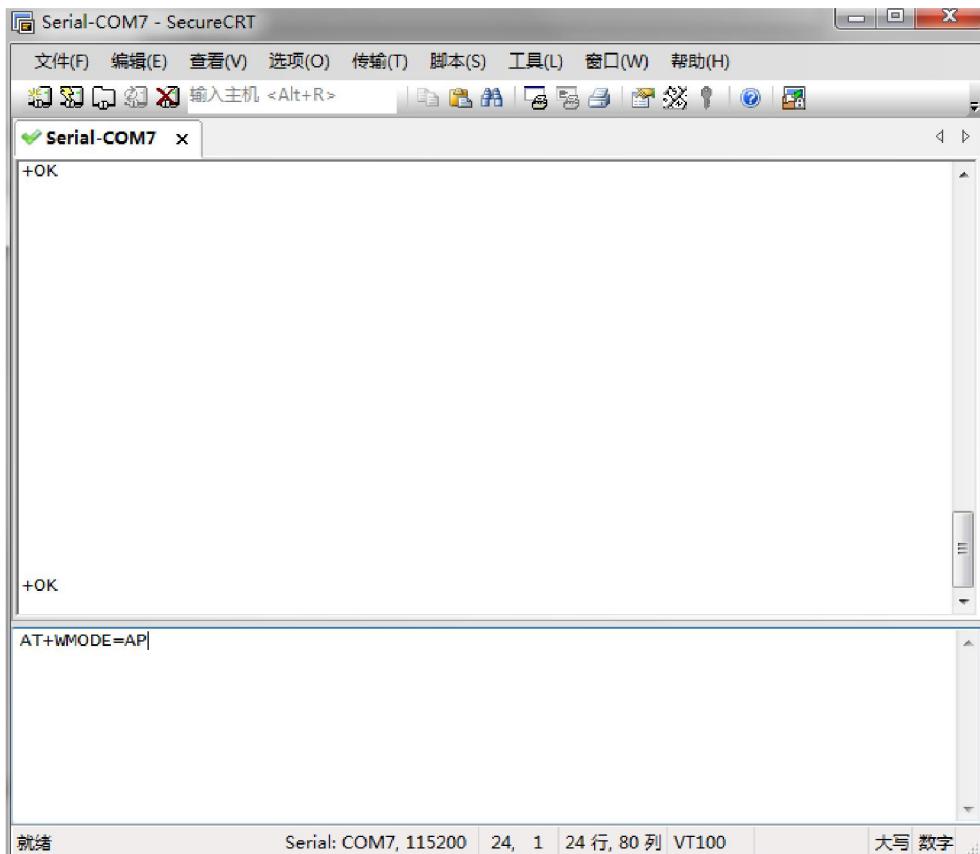


图 3-67 修改模块的 AP 模式

使用电脑搜索名称为 MXCHIP_xxxx（xxxx 为模块 MAC 地址的后五位）的热点，并进行连接。连接成功后，打开电脑浏览器，输入模块的 IP 地址 <http://10.10.10.1> 进行访问，弹出用户登录窗口如下图所示，用户名和密码 s 均为 admin。



图 3-68 模块登录界面

在“无线终端设置”中设置 EMW3081 的 AP 信息，如下图所示：设置 ssid 为 OneNET，pwd 为 12345678。



图 3-69 配置模块的 ssid 及 pwd

Step5 编译整个工程，并下载程序，请保证 JATG 和电源已正确连接。

Step6 烧写程序完后，重新给开发板上电，建议每次烧写程序后都这样做，让 WIFI 模块完全掉电，以免 AT 命令操作失败。

Step7 程序运行后，打印如下：



图 3-70 程序启动打印信息

Step8OneNET 平台下查看上传的数据。如下图所示。



图 3-71 平台数据查看

参考资料

- (1) 开发板硬件原理图 V1.0.pdf
- (2) 设备终端接入协议 2-EDP.docx
- (3) ESP8266 AT 命令手册
- (4) M6311 AT 命令用户使用手册.pdf
- (5) DS0013C_EMW3081_V1.0.pdf
- (6) 设备终端接入协议 4-MQTT
- (7) MODBUS 协议.pdf
- (8) MQTT-3.1.1-CN.pdf
- (9) 设备终端接入协议 3-MODBUS.docx