W5500 如何通过 MQTT 协议连接阿里云

一、 简介：

1、 开发环境与连接平台：

本文主要介绍 W5500 如何通过 MQTT 协议将设备连接到阿里云 IoT，并通过 MQTT 协议实现通信。MQTT 协议是基于 TCP 的协议，所以我们只需要在单片机端实现 TCP 客户端代码之后就很容易移植 MQTT 了， +W5500 实现 TCP 客户端的代码我们以前已经实现过，程序下载地址为（http://www.w5500.com/）

软件环境：Windows

硬件环境：STM32F103+W5500

开发工具：Keil uVision5

调试工具：Wireshark、串口调试助手

连接平台：阿里云-华东 2 节点（https://www.aliyun.com）

2、 MQTT 简介：

MQTT 官网地址：（http://mqtt.org/）

1） MQTT 协议特点

MQTT 是一个基于客户端-服务器的消息发布/订阅传输协议。MQTT 协议是轻量、简单、开放和易于实现的，这些特点使它适用范围非常广泛。在很多情况下，包括受限的环境中，如：机器与机器（M2M）通信和物联网（IoT）。其在，通过卫星链路通信传感器、偶尔拨号的医疗设备、智能家居、及一些小型化设备中已广泛使用。

MQTT 协议当前版本为，2014 年发布的 MQTT v3.1.1。除标准版外，还有一个简化版 MQTT-SN，该协议主要针对嵌入式设备，这些设备一般工作于百 TCP/IP 网络，如：ZigBee。

MQTT 协议运行在 TCP/IP 或其他网络协议，提供有序、无损、双向连接。其特点包括：

使用的发布/订阅消息模式，它提供了一对多消息分发，以实现与应用程序的解耦。

对负载内容屏蔽的消息传输机制。

对传输消息有三种服务质量（QoS）：

- 最多一次，这一级别会发生消息丢失或重复，消息发布依赖于底层 TCP/IP 网络。即：<=1
- 至多一次，这一级别会确保消息到达，但消息可能会重复。即：>=1
- 只有一次，确保消息只有一次到达。即：＝1。在一些要求比较严格的计费系统中，可以使用此级别

数据传输和协议交换的最小化（协议头部只有 2 字节），以减少网络流量

通知机制，异常中断时通知传输双方

2） MQTT 协议原理及实现方式

实现 MQTT 协议需要：客户端和服务器端

MQTT 协议中有三种身份：发布者（Publish）、代理（Broker）（服务器）、订阅者（Subscribe）。其中，消息的发布者和订阅者都是客户端，消息代理是服务器，消息发布者可以同时是订阅者。

MQTT 传输的消息分为：主题（Topic）和消息的内容（payload）两部分

Topic，可以理解为消息的类型，订阅者订阅（Subscribe）后，就会收到该主题的消息内容（payload）

payload，可以理解为消息的内容，是指订阅者具体要使用的内容

二、 连接

1. 阿里云连接步骤：

1) 以 aliyun 账号直接进入 IoT 控制台，如果还没有开通阿里云物联网套件服务，则申请开通

2) 接入引导

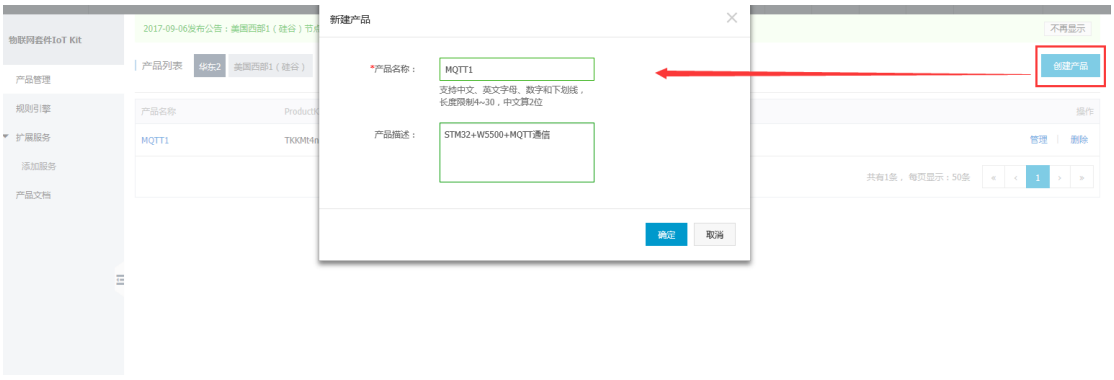
（1）、创建产品

（2）、添加设备

（3）、获取设备的 Topic

- 创建产品

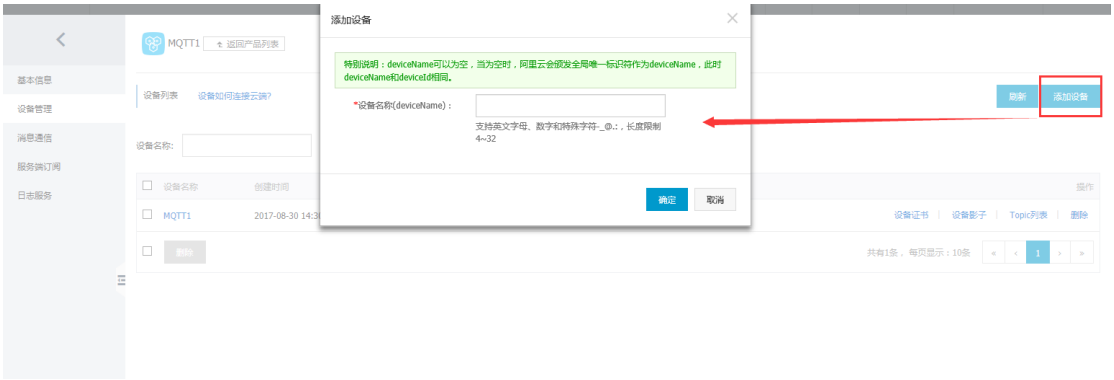
初步进入控制台后，需要创建产品。点击创建产品。产品相当于某一类设备的集合，用户可以根据产品管理其设备等。



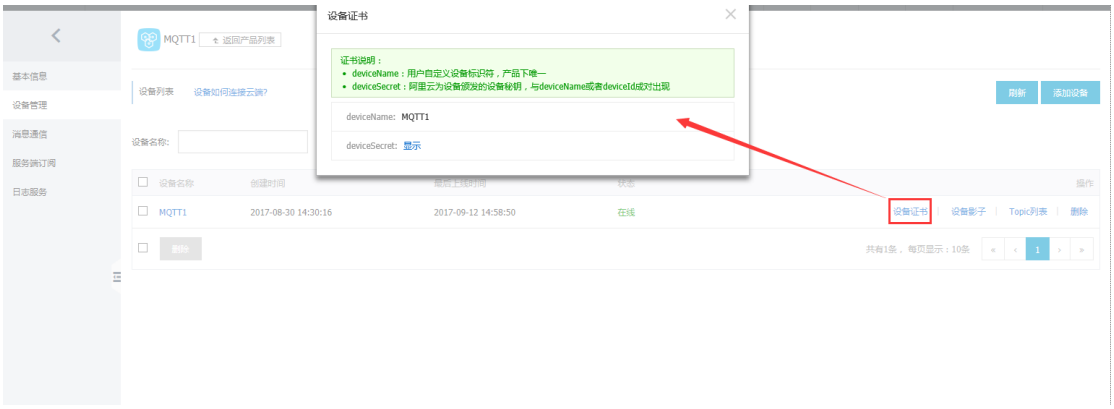
- 产品名称：对产品命名，例如可以填写产品型号。产品名称在账号内保持唯一。
- productKey：阿里云 IoT 为产品颁发的全局唯一标识符

- 添加设备：

创建完产品之后，可以为该产品添加设备。进入产品管理页面下的设备管理，点击添加设备。



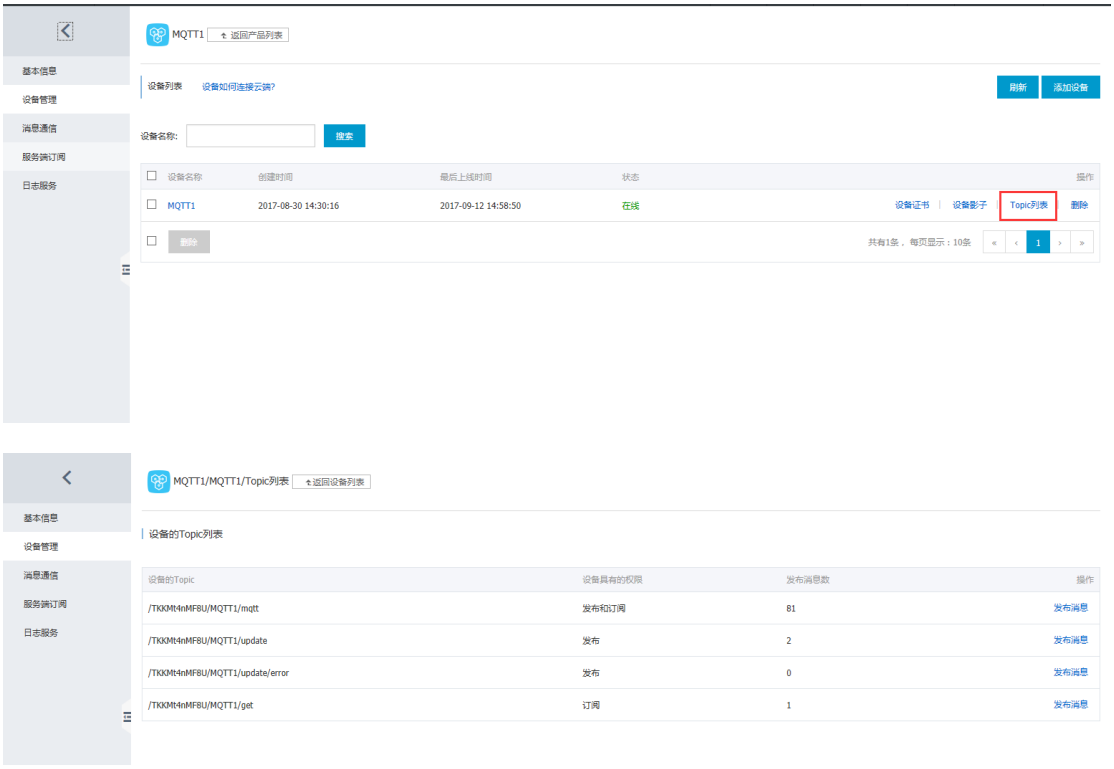
- 说明：用户可以自定义设备名称(即 deviceName)，这个名称即可作为设备唯一标识符，用户可以基于该设备名称与 IoT Hub 进行通信，需要指出的是，用户需要保证 deviceName 产品内唯一。



- 设备证书：添加设备之后，物联网套件为设备颁发的唯一标识符，设备证书用于设备认证以及设备通信，详细的请参考设备接入文档。
- deviceName：用户自定义设备唯一标识符，用于设备认证以及设备通信，用户保证产品维度内唯一。
- deviceSecret：物联网套件为设备颁发的设备秘钥，用于认证加密，与 deviceName 或者 deviceId 成对出现。

● 获取设备的 Topic

添加设备之后，可以获取设备的 Topic。点击 Topic 列表



- 说明：创建产品之后，物联网套件都会为产品默认定义三个 Topic 类。那么，在添加设备之后，每个设备都会默认有三个 Topic，即图中所示。如果想要增加、修改、删除 Topic，请到消息通信重新定义 Topic 类。
- 设备可以基于 Topic 列表中的 Topic 进行 Pub/Sub 通信，例如列表中有/1000118502/test9/update，且设备拥有的权限是发布，这就意味着设备可以往这个 Topic 发布消息；同样，列表中/1000118502/test9/get，权限是订阅，这就意味着设备可以从这个 Topic 订阅消息。

- 设备接入

获得 productKey、设备证书以及设备的 Topic 这些参数，就可以基于 aliyun IoT device SDK for C 将设备连接上 IoT Hub 并进行通信，具体请参考《MQTT 配置》部分

2. MQTT 移植步骤：

MQTT 代码源码下载地址：（http://www.eclipse.org/paho/）

MQTT 的移植非常简单，将 C/C++ MQTT Embedded clients 的代码添加到工程中，然后我们只需要再次封装 4 个函数即可：

int transport_sendPacketBuffer(unsigned char* buf, int buflen);

通过网络以 TCP 的方式发送数据；

int transport_getdata(unsigned char* buf, int count);

TCP 方式从服务器端读取数据，该函数目前属于阻塞函数；

int transport_open(void);

打开一个网络接口，其实就是和服务器建立一个 TCP 连接；

int transport_close(void);

关闭网络接口。

如果已经移植好了 socket 方式的 TCP 客户端的程序，那么这几个函数的封装也是非常简单的，程序代码如下所示：

```
/**
 * @brief  通过 TCP 方式发送数据到 TCP 服务器
 * @param  buf 数据首地址
 * @param  buflen 数据长度
 * @retval 小于 0 表示发送失败
 */

/*订阅消息*/
int Subscribe_sendPacketBuffer(unsigned char* buf, int buflen)
{
    return send(SOCK_TCPS,buf,buflen);
}

/*发布消息*/
int Published_sendPacketBuffer(unsigned char* buf, int buflen)
{
    return send(SOCK_TCPC,buf,buflen);
}

/**
 * @brief  阻塞方式接收 TCP 服务器发送的数据
 * @param  buf 数据存储首地址·
 * @param  count 数据缓冲区长度
 * @retval 小于 0 表示接收数据失败
 */
int Subscribe_getdata(unsigned char* buf, int count)
{

    return recv(SOCK_TCPS,buf,count);

}

int Published_getdata(unsigned char* buf, int count)
{
```

```
35     return recv(SOCK_TCPC,buf,count);
36 
37 }
38 
39 /**
40 * @brief  打开一个 socket 并连接到服务器
41 * @param  无
42 * @retval 小于 0 表示打开失败
43 */
44 int Subscribe_open(void)
45 {
46     int32_t ret;
47     //新建一个 socket 并绑定本地端口 5000
48 ret = socket(SOCK_TCPS,Sn_MR_TCP,50000,0x00);
49     if (ret != 1) {
50         printf("%d:Socket Error\r\n",SOCK_TCPS);
51         while (1);
52     } else {
53         printf("%d:Opened\r\n",SOCK_TCPS);
54     }
55 
56 
57     while (getSn_SR(SOCK_TCPS)!=SOCK_ESTABLISHED) {
58         printf("connecting\r\n");
60       //连接 TCP 服务器÷
61      ret = connect(SOCK_TCPS,server_ip,1883);
62           //端口必须为 1883
63   }
63 if (ret != 1) {
64         printf("%d:Socket Connect Error\r\n",SOCK_TCPS);
65         while (1);
66     } else {
67         printf("%d:Connected\r\n",SOCK_TCPS);
68     }
69     return 0;
70 }
71 
72 int Published_open(void)
73 {
74     int32_t ret;
75 
76     ret = socket(SOCK_TCPC,Sn_MR_TCP,5001,0x00);
77 
78     if (ret != 1) {
79         printf("%d:Socket1 Error1\r\n",SOCK_TCPC);
80         while (1);
81     } else {
82         printf("%d:socket1 Opened\r\n",SOCK_TCPC);
83     }
84 
85 
86     while (getSn_SR(SOCK_TCPC)!=SOCK_ESTABLISHED) {
87         ret = connect(SOCK_TCPC,server_ip,1883);
88           //端口必须为 1883
89 }
90     if (ret != 1) {
91         printf("%d:Socket Connect1 Error\r\n",SOCK_TCPC);
92         while (1);
93     } else {
94         printf("%d:Connected1\r\n",SOCK_TCPC);
95     }
96     return 0;
97 }
98 
99 }
100 
101 /**
```

```
102 * @brief  关闭 socket
103 * @param  无
104 * @retval 小于 0 表示关闭失败
105 */
106 int Subscribe_close(void)
107 {
108     disconnect(SOCK_TCPS);
109     printf("close0\n\r");
110 
111     while (getSn_SR(SOCK_TCPC)!=SOCK_CLOSED) {
112         ;
113     }
114     return 0;
115 }
116 
117 
118 int Published_close(void)
119 {
120     disconnect(SOCK_TCPC);
121     printf("close1\n\r");
122 
123     while (getSn_SR(SOCK_TCPC)!=SOCK_CLOSED) {
124         ;
125     }
126     return 0;
127 }
```

3、 MQTT 配置

1） MQTT 连接参数说明

```
1. 连接域名：${productKey}.iot-as-mqtt.cn-shanghai.aliyuncs.com:1883， ${productKey}请替换为您的产品key，mqtt的Connect
   报文参数如下：
2. mqttClientId: clientId+"|securemode=3,signmethod=hmacsha1,timestamp=132323232|"
3. mqttUsername: deviceName+"&"+productKey
4. mqttPassword: sign_hmac(deviceSecret,content)sign签名需要把以下参数按字典序排序后，再根据signmethod加签。
5. content=提交给服务器的参数（productKey,deviceName,timestamp,clientId），按照字母顺序排序，然后将参数值依次拼接
6. 其中clientId是客户端自表示id，建议mac或sn，64字符内；timestamp当前时间毫秒值，可选；mqttClientId格式中||内为扩展参数，
   signmethod代表签名算法类型，securemode代表目前安全模式，可选值有2 （TLS直连模式）、3（TCP直连模式）,参考以下示例：
```

举例：

```
1. 如果clientId = 12345，deviceName = device， productKey = pk， timestamp = 789，signmethod=hmacsha1，
   deviceSecret=secret，那么使用tcp方式提交给mqtt参数分别如下：
2. mqttclientId=12345|securemode=3,signmethod=hmacsha1,timestamp=789|
3. username=device&pk
4. password=hmacsha1("secret","clientId12345deviceNamedeviceproductKeypktimestamp789").toHexString();  //最后是二进制
   转16制字符串，大小写不敏感。 这个例子结果为 FAFD82A3D602B37FB0FA8B7892F24A477F851A14
5. 注意上面3个参数分别是mqtt Connect登录报文的mqttClientId,mqttUsername,mqttPasswrod
```

2） MQTT 与阿里云连接函数：

参考阿里云内 MQTT 设备接入手册，计算出设备连接的各项参数，例如下列程序中框中的部分为本例程 MQTT 与阿里云连接的参数的配置，详细内容如下：

clientId = 192.168.207.115
deviceName = MQTT1
productKey = TKKMt4nMF8U
timestamp = 789（毫秒值）

signmethod = hmacsha1（算法类型）
deviceSecret = secret
那么使用 tcp 方式提交给 mqtt 参数分别如下：

(1) mqttClientId:clientId+"|securemode=3,signmethod=hmacsha1,timestamp=789|"
clientId=192.168.207.115|securemode=3,signmethod=hmacsha1,timestamp=789|
(2) keepalive 时间需要设置超过 60 秒以上，否则会拒绝连接。
(3) Cleansession 为 1；
(4) mqttUsername: deviceName+"&"+productKey
username = "MQTT1&TKKMt4nMF8U"
(5) password=hmacsha1("secret","clientId192.168.207.115deviceNameMQTT1productKeyTKKMt4nMF8Utimestamp789").toHexString();
最后是二进制转 16 制字符串大小写不敏感。这个例子结果为 9076b0ebc04dba8a8ebba1f0003552dbc862c9b9

MQTT 连接函数原型，tcp_client.c 文件中的 MQTT_CON_ALI 函数中调用 make_con_msg 函数并通过阿里云设备的参数，设置 MQTT 连接阿里云函数的参数：

```
void make_con_msg(char* clientID,int keepalive, uint8 cleansession,
                  char*username,char* password,unsigned char*buf,int
                  buflen)
{
    int32_t len,rc;
    MQTTPacket_connectData data = MQTTPacket_connectData_initializer;
    data.clientID.cstring = clientID;
    data.keepAliveInterval = keepalive;
    data.cleansession = cleansession;
    data.username.cstring = username;
    data.password.cstring = password;
    len = MQTTSerialize_connect(buf, buflen, &data);
         //构造链接报文
    return;
}
```

MQTT 连接过程：

```
void MQTT_CON_ALI(void)
{
    int len;
    int type;
    switch (getSn_SR(0)) {
          //获取 socket0 的状态
    case SOCK_INIT:
    //Socket 处于初始化完成（打开）状态
        connect(0, server_ip,server_port);
              //配置 Sn_CR 为 CONNECT，并向 TCP 服务器发出连接请求¢



        break;
    case SOCK_ESTABLISHED:              //
    Socket 处于连接建立状态
        if (getSn_IR(0) & Sn_IR_CON) {
            setSn_IR(0, Sn_IR_CON);       //
                  Sn_IR 的 CON 位置 1，通知 W5500 连接已建立
```

```
19
20          }
21          memset(msgbuf,0,sizeof(msgbuf));
22          if ((len=getSn_RX_RSR(0))==0) {
23              if (1==CONNECT_FLAG) {
24                  printf("send connect\r\n");
25
26                  /*MQTT报 拼接连接报文
27                  *根据阿里云平台 MQTT 设备接入手册配置
28
29                  */
30
31                  //void make_con_msg(char* clientID,int keepalive,
32                                      uint8 cleansession,char*username,
33                                      char* password,unsigned char*buf,
34                                      int buflen)
35                  make_con_msg("192.168.207.115|securemode=3,
36                              signmethod=hmacsha1,timestamp=789|",180,
37                              1,"MQTT1&TKKMt4nMF8U",
38                              "9076b0ebc04dba8a8ebba1f0003552dbc862c9b9"
39                              ,msgbuf,sizeof(msgbuf));
40
41
42                  //printf(" server_ip: %d.%d.%d.%d\r\n", server_ip[0],
43                         server_ip[1],server_ip[2],server_ip[3]);
44                  //printf("connect ALY\r\n");
45                  CONNECT_FLAG = 0;
46                  send(0,msgbuf,sizeof(msgbuf));
47                  Delay_s(2);
48                  while ((len=getSn_RX_RSR(0))==0) {
49                      Delay_s(2);
50                      send(0,msgbuf,sizeof(msgbuf));
51                  };
52                  recv(0,msgbuf,len);
53                  while (mqtt_decode_msg(msgbuf)!=CONNACK) {
54                        //判断是不是 CONNACK
55                      printf("wait ack\r\n");
56                  }
57              } else if (SUB_FLAG == 1) {
58                  memset(msgbuf,0,sizeof(msgbuf));
59                  make_sub_msg(topic,msgbuf,sizeof(msgbuf));
60                  // make_pub_msg(topic,msgbuf,sizeof(msgbuf),"hello");
61                  send(0,msgbuf,sizeof(msgbuf));
62                      // 接收到数据后再回给服务器，完成数据回环
63
64                  SUB_FLAG = 0;
65                  Delay_s(2);
66                  while ((len=getSn_RX_RSR(0))==0) {
67                      Delay_s(2);
68                      send(0,msgbuf,sizeof(msgbuf));
69                  };
70                  recv(0,msgbuf,len);
71                  while (mqtt_decode_msg(msgbuf)!=SUBACK) {
72                        //判断是不是 SUBACK
73                      printf("wait suback\r\n");
74                  }
75                  TIM_Cmd(TIM2, ENABLE);
76                  printf("send sub\r\n");
77
78              }
79  #if 1
80              else {
81                  //count++;
82                  // Delay_s(2);
83                  if (count>10000) {
84                      count = 0;
85                      make_ping_msg(msgbuf,sizeof(msgbuf));
```

```
86                  send(0,msgbuf,sizeof(msgbuf));
87
88                  while ((len=getSn_RX_RSR(0))==0) {
89                      //Delay_s(2);
90                      //send(0,msgbuf,sizeof(msgbuf));
91                      printf("wait pingresponse");
92
93                  };
94                  recv(0,msgbuf,len);
95                  printf("ping len : %d\r\n",len);
96                  if (len>2) {
97                      if (PUBLISH==mqtt_decode_msg(msgbuf+2)) {
98                          printf("publish\r\n");
99                          MQTTDeserialize_publish(&dup, &qos,
100                                                  &retained,
101                                                  &mssageid,
102                                                  &receivedTopic,
103                                                  &payload_in,
104                                                  &payloadlen_in,
105                                                 msgbuf+2, len-2);
106                          // printf("message arrived %d: %s\n\r",
107                                  payloadlen_in, payload_in);
108                          memset(topic,0,sizeof(topic));
109                          memset(ser_cmd,0,sizeof(ser_cmd));
110                          memcpy(topic,receivedTopic.lenstring.data,
111                                receivedTopic.lenstring.len);
112                          replace_string(new_topic,topic , "request",
113                                         "response");
114                          printf("topic:%s\r\n",topic);
115                          strcpy(ser_cmd,(const char *)payload_in);
116                          //parse_topic(ser_cmd);
117                          // printf("message is %s\r\n",ser_cmd);
118                          memset(msgbuf,0,sizeof(msgbuf));
119                          make_pub_msg(new_topic,msgbuf,sizeof(
120                                       msgbuf),"hello");
121                          send(0,msgbuf,sizeof(msgbuf));
122                      }
123                  }
124              }
125          }
126 #endif
127 #if 0
128          if (PUB_FLAG==1) {
129              memset(msgbuf,0,sizeof(msgbuf));
130              // make_sub_msg(topic,msgbuf,sizeof(msgbuf));
131              make_pub_msg(topic,msgbuf,sizeof(msgbuf),"hello");
132              if (count == 10000) {
133 PUB:
134                  send(0,msgbuf,sizeof(msgbuf));  //
135                      接收到数据后再回给服务器，完成数据回环
136                      ÷£¬Íê³ÉÊý¾Ý»Ø»·
137
138                  Delay_s(2);
139                  // while((len=getSn_RX_RSR(0))==0)
140                  //  {
141                  // Delay_s(2);
142                  //send(0,msgbuf,sizeof(msgbuf));
143                  //    printf("puback\r\n");
144                  //  };
145                  // recv(0,msgbuf,len);
146                  //  if(mqtt_decode_msg(msgbuf)!=PUBACK)
147                        //ÅÐ¶ÏÎÊÇ²»ÊÇSUBACK
148                  //  {
149                  //      goto PUB;
150                  //      printf("wait Puback\r\n");
151                  //  }
152                  printf("send Pub\r\n");
153              }
```

```
154           }
155 #endif
156       }
157 #if 1
158       if ((len=getSn_RX_RSR(0))>0) {
159          recv(0,msgbuf,len);
160          if (PUBLISH== mqtt_decode_msg(msgbuf)) {
161             printf("publish\r\n");
162             MQTTDeserialize_publish(&dup, &qos, &retained,
163                                     &mssageid, &receivedTopic,
164                                     &payload_in, &payloadlen_in,
165                                     msgbuf, len);
166             // printf("message arrived %d: %s\n\r", payloadlen_in,
167                     payload_in);
168             memset(topic,0,sizeof(topic));
169             memcpy(topic,receivedTopic.lenstring.data,
170                   receivedTopic.lenstring.len);
171             replace_string(new_topic,topic , "request","response");
172 
173             printf("topic:%s\r\n",topic);
174 
175             memset(ser_cmd,0,sizeof(ser_cmd));
176             memcpy(ser_cmd,(const char *)payload_in,strlen((char*)
177                   payload_in));
178             memset(msgbuf,0,sizeof(msgbuf));
179             make_pub_msg(new_topic,msgbuf,sizeof(msgbuf),rebuf);
180             send(0,msgbuf,sizeof(msgbuf));
181             //printf("%s\n",msgbuf);
182          } else if (PINGRESP== mqtt_decode_msg(msgbuf)) {
183             if (len>2) {
184                if (PUBLISH==mqtt_decode_msg(msgbuf+2)) {
185                   printf("publish\r\n");
186                   MQTTDeserialize_publish(&dup, &qos, &retained,
187                                           &mssageid,
188                                           &receivedTopic,
189                                           &payload_in,
190                                           &payloadlen_in, msgbuf+
191                                           2, len-2);
192                   // printf("message arrived %d: %s\n\r",
193                           payloadlen_in, payload_in);
194                   memset(topic,0,sizeof(topic));
195                   memcpy(topic,receivedTopic.lenstring.data,
196                         receivedTopic.lenstring.len);
197                   replace_string(new_topic,topic,"request",
198                                  "response");
199                   printf("topic:%s\r\n",topic);
200                   memset(ser_cmd,0,sizeof(ser_cmd));
201                   strcpy(ser_cmd,(const char *)payload_in);
202                   // printf("message is %s\r\n",ser_cmd);
203                   //parse_topic(ser_cmd);
204                   memset(msgbuf,0,sizeof(msgbuf));
205                   make_pub_msg(new_topic,msgbuf,sizeof(msgbuf),
206                                "hello");
207                   send(0,msgbuf,sizeof(msgbuf));
208                }
209             }
210 
211          } else {
212             printf("wait publish\r\n");
213          }
214       }
215       //  printf("send ping\r\n");
216 
217 #endif
218       break;
219    case SOCK_CLOSE_WAIT:
220    //Socket 处于等待关闭状态
221       close(0);                           // 关闭 Socket0
```

```
222         break;
223      case SOCK_CLOSED:
224     // Socket 处于关闭状态
225         socket(0,Sn_MR_TCP,local_port,Sn_MR_ND);
226               // 打开 Socket0，并配置为 TCP 无延时模式，打开一个本地端口
227
228         break;
229      }
230
231 }
```

◆ Password 有两种获得方法：

通过网页“在线加密解密”HamcSHA1 获得；(http://encode.chahuo.com/)

通过 hmacsha1 算法解析获得解析步骤如下：

```
1 void hmac_sha1(uint8_t *key, uint16_t key_length, uint8_t *data,
 2              uint16_t data_length, uint8_t *digest)
 3 {
 4     uint8_t b = 64;                              /* blocksize */
 5     uint8_t ipad = 0x36;
 6     uint8_t opad = 0x5c;
 7     uint8_t k0[64];
 8     uint8_t k0xorIpad[64];
 9     uint8_t step7data[64];
10     uint8_t step5data[MAX_MESSAGE_LENGTH+128];
11     uint8_t step8data[64+20];
12     uint16_t i;
13
14     for (i=0; i<64; i++) {
15         k0[i] = 0x00;
16     }
17
18     /* Step 1 */
19     if (key_length != b) {
20         //判断秘钥 K 字节长度是否等于 B
21         /* Step 2 */
22         if (key_length > b) {
23             //如果大于 B，则另 K0=H(K)
24             sha1(key, key_length, digest);
25             for (i=0; i<20; i++) {
26                 k0[i]=digest[i];
27             }
28         }
29         /* Step 3 */
30         else if (key_length < b) {
31                 //如果小于 B，则在末尾添加 B-length(K)
32                 位的 0
33             for (i=0; i<key_length; i++) {
34                 k0[i] = key[i];
35             }
36         }
37     } else {
38         for (i=0; i<b; i++) {
39             k0[i] = key[i];
40         }
41     }
42
43
44 #ifdef HMAC_DEBUG
45     debug_out("k0",k0,64);
46 #endif
47
48     /* Step 4 */
49     for (i=0; i<64; i++) {
50         k0xorIpad[i] = k0[i] ^ ipad;
51                     //将 K0 和 ipad 进行异或运算
```

```
52      }
53
54  #ifdef HMAC_DEBUG
55      debug_out("k0 xor ipad",k0xorIpad,64);
56  #endif
57
58      /* Step 5 */
59      for (i=0; i<64; i++) {
60          step5data[i] = k0xorIpad[i];
61      }
62      for (i=0; i<data_length; i++) {
63          step5data[i+64] = data[i];
64                          //将数据添加在第 4 步生成的字节串
65                          后面
66      }
67
68
69  #ifdef HMAC_DEBUG
70      debug_out("(k0 xor ipad) || text",step5data,data_length+64);
71  #endif
72
73
74      /* Step 6 */
75      sha1(step5data, data_length+b, digest);
76          //将第 5 步的结果运用 H 函数
77
78  #ifdef HMAC_DEBUG
79      debug_out("Hash((k0 xor ipad) || text)",digest,20);
80  #endif
81
82      /* Step 7 */
83      for (i=0; i<64; i++) {
84          step7data[i] = k0[i] ^ opad;
85                       //将 K0 和 opad 进行异或运算
86      }
87
88  #ifdef HMAC_DEBUG
89      debug_out("(k0 xor opad)",step7data,64);
90  #endif
91
92      /* Step 8 */
93      for (i=0; i<64; i++) {
94          step8data[i] = step7data[i];
95      }
96      for (i=0; i<20; i++) {
97          step8data[i+64] = digest[i];
98                          //将第 6 步的结果加在第 7 步生成的��
99                          ⬚节串上
100     }
101
102 #ifdef HMAC_DEBUG
103     debug_out("(k0 xor opad) || Hash((k0 xor ipad) || text)",step8data,
104             20+64);
105 #endif
106
107     /* Step 9 */
108     sha1(step8data, b+20, digest);
109
110 #ifdef HMAC_DEBUG
111     debug_out("HASH((k0 xor opad) || Hash((k0 xor ipad) || text))",
112             digest,20);
113 #endif
114 }
```

3） **配置远程服务器 IP 地址和服务器端口**

通过域名解析获取 IP 地址有两种方法：

a、通过在终端下 ping 域名的方法获取 IP 地址

b、通过 DNS 域名解析的方法获取 IP 地址

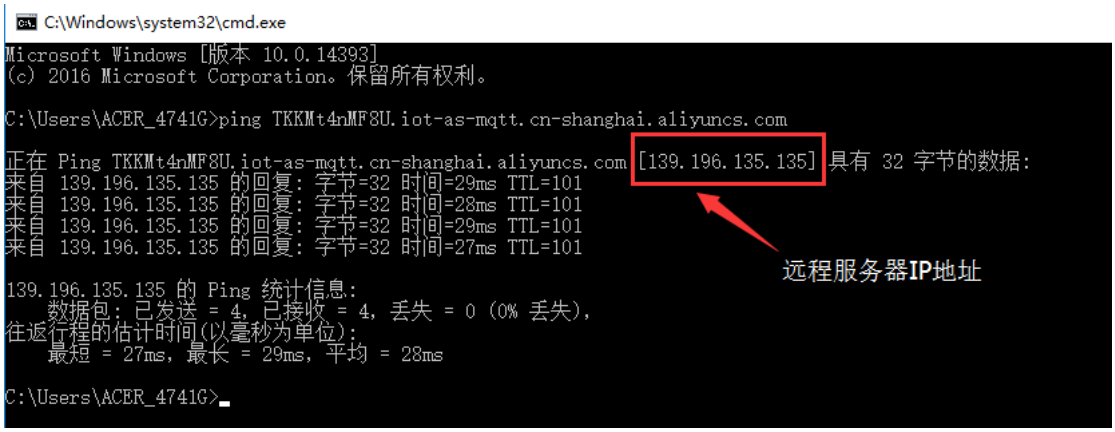
a） **通过在终端下 ping 域名的方法获取 IP 地址**

```
15    //{TKKMt4nMF8U}.iot-as-mqtt.cn-shanghai.aliyuncs.com
16    uint8 server_ip[4]={139,196,135,135};      // 配置远程服务器IP地址
17    uint16 server_port=1883;                   // 配置远程服务器端口
```

把 ${productKey}替换为您的产品 key，并在终端对 MQTT 进行 ping 操作，来获取服务器 IP 地址

```
1. 连接域名：${productKey}.iot-as-mqtt.cn-shanghai.aliyuncs.com:1883， ${productKey}请替换为您的产品key，mqtt的Connect
   报文参数如下：
2. mqttClientId: clientId+"|securemode=3,signmethod=hmacsha1,timestamp=132323232|"
3. mqttUsername: deviceName+"&"+productKey
4. mqttPassword: sign_hmac(deviceSecret,content)sign签名需要把以下参数按字典序排序后，再根据signmethod加签。
5. content=提交给服务器的参数（productKey,deviceName,timestamp,clientId），按照字母顺序排序，然后将参数值依次拼接
6. 其中clientId是客户端自表示id，建议mac或sn，64字符内；timestamp当前时间毫秒值，可选；mqttClientId格式中||内为扩展参数，
   signmethod代表签名算法类型，securemode代表目前安全模式，可选值有2 （TLS直连模式）、3（TCP直连模式）,参考以下示例：
```

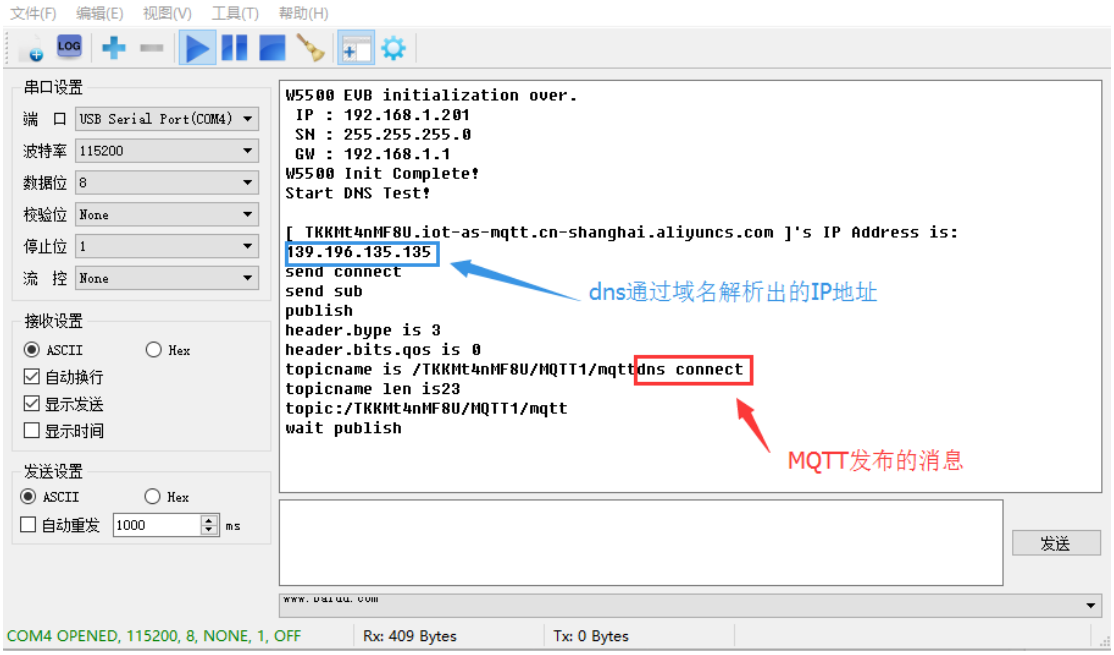
举例：



b） **通过 DNS 域名解析的方法获取 IP 地址**

首先完成 W5500 的 DNS 域名解析例程的移植，把 DNS 相关部分移植到本程序中，再进行相关配置即可完成(DNS 相关例程下载地址 http://www.w5500.com/），DNS 解析域名成功后，把解析出的 IP 地址赋值给 MQTT 的 server_ip，用于 MQTT 与阿里云的连接，完成 MQTT 协议通信

连接成功后，通过串口调试助手验证 DNS 域名解析是否正确，若正确则 MQTT 与阿里云连接成功，并可成功的发布订阅消息：



4) 设置发布订阅的主题：

在 tcp_client.c 文件中设置 MQTT 与阿里云连接参数，并通过调用 mqtt_fun.c 文件中的相关底层函数来完成 MQTT 与阿里云连接：

```
33    MQTTString receivedTopic;
34    char topic[100]={"/TKKMt4nMF8U/MQTT1/mqtt"};//设置发布订阅主题
35    char new_topic[100];
```

底层的订阅发布函数

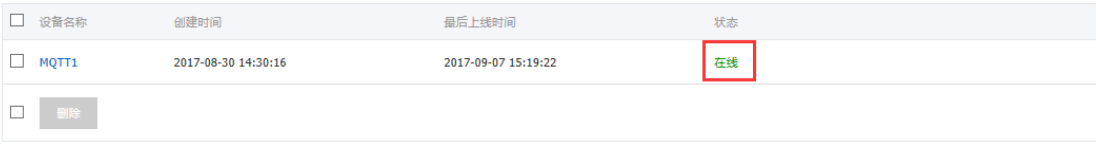
```

/*****************拼接订阅报文*************************************/
void make_sub_msg(char *Topic,unsigned char*msgbuf,int buflen)
{
   int msgid = 1;
   int req_qos = 0;
   unsigned char topic[100];
   MQTTString topicString= MQTTString_initializer;
   memcpy(topic,Topic,strlen(Topic));
   topicString.cstring = (char*)topic;
   //topicString.lenstring.len=4;
   MQTTSerialize_subscribe(msgbuf, buflen, 0, msgid, 1, &topicString,
                           &req_qos);
   return;
}
/*********拼接发布报文*****************/
void make_pub_msg(char *Topic,unsigned char*msgbuf,int buflen,char*msg)
{
   unsigned char topic[100];
   int msglen = strlen(msg);
   MQTTString topicString = MQTTString_initializer;
   memset(topic,0,sizeof(topic));
   memcpy(topic,Topic,strlen(Topic));
   topicString.cstring = (char*)topic;
   MQTTSerialize_publish(msgbuf, buflen, 0, 2, 0, 0, topicString, (
                         unsigned char*)msg, msglen);
   return;
}
```

此发布订阅的主题根据阿里云中设备管理的 Topic 列表设置

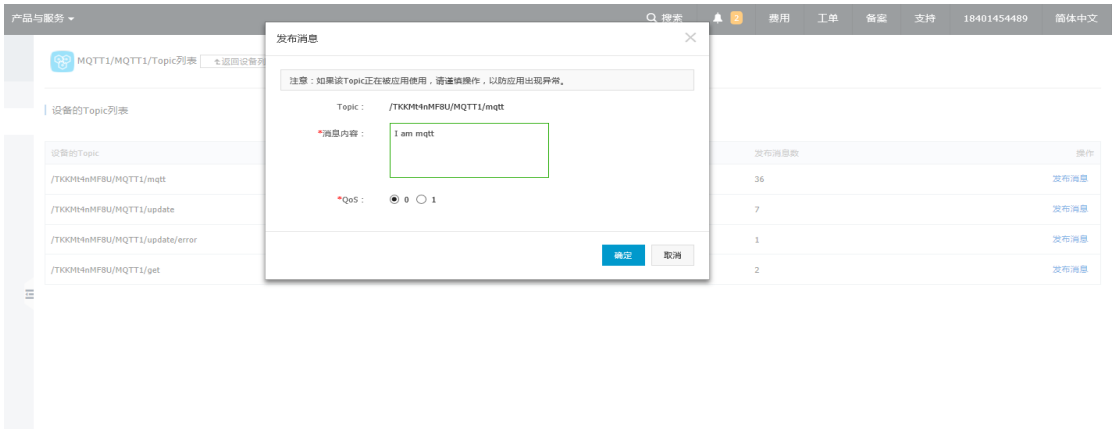
设备可以基于 Topic 列表中的 Topic 进行 Pub/Sub 通信，例如列表中有/TKKMt4nMF8U/MQTT1/mqtt，且设备拥有的权限是发布和订阅，这就意味着设备可以往这个 Topic 发布消息，同样设备可以从这个 Topic 订阅消息。

4、 简单测试：

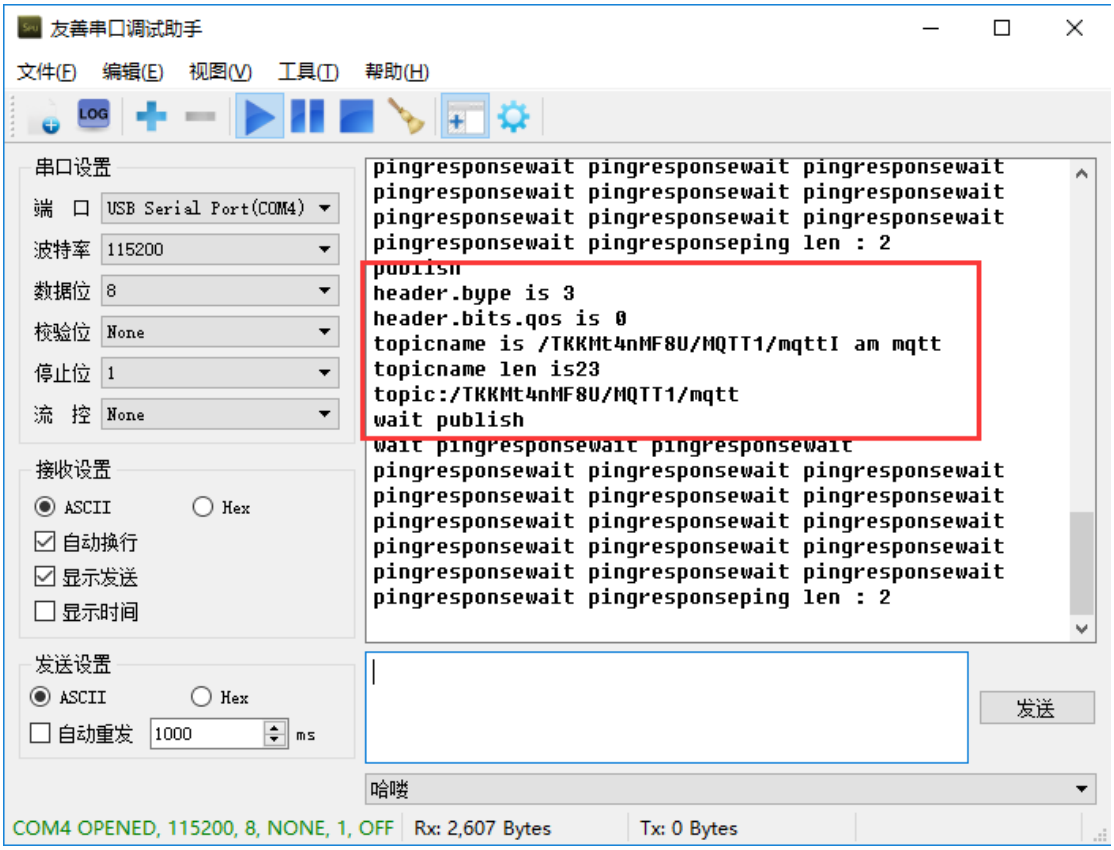
把程序下载到测试板并连接，登陆阿里云，到添加的设备，开启测试板，状态显示在线，说明 MQTT 与阿里云已经初步连接上



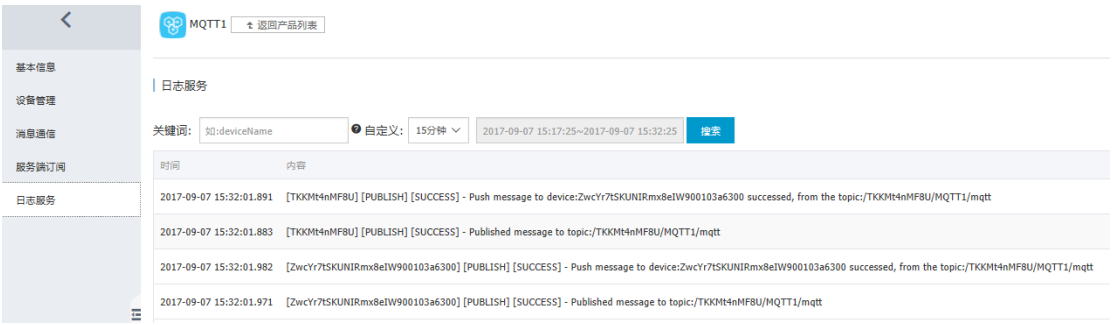
通过设备的 Topic 列表，选择程序中设置的发布订阅的 Topic 进行发布消息的操作：



串口打印接收到的服务器端发送的消息：

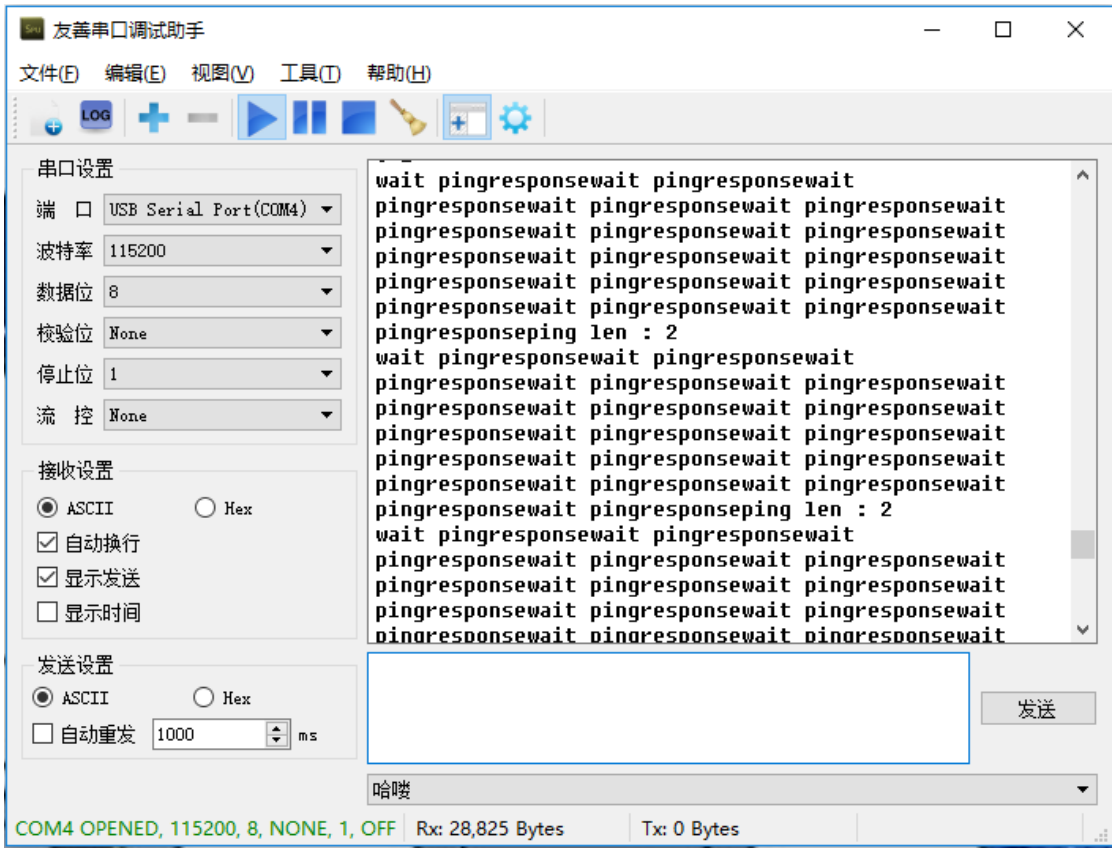


同时可在日志服务中查询相关设备的相关消息：



此时 MQTT 协议通信成功。

说明：在串口通信中会一直打印消息，是因为程序中设置了对 MQTT 的 ping 操作，防止 MQTT 离线。



三、 注意：

1、 在 MQTT 与阿里云连接时，会出现离线的状态，在离线状态时重启测试板并手动刷新阿里云即可。因为状态不是实时的显示，会有一段时间的延迟，可耐心等待。

2、 MQTT CONNECT 协议设置时的注意事项：

```
1. Connect指令中的KeepAlive时间需要设置超过60秒以上，否则会拒绝连接。
2. 如果同一个设备多个连接可能会导致客户端互相上下线，MQTT默认开源SDK会自动重连，您可以通过日志服务看到设备行为。
```

3、 错误码

```
1. 401: request auth error，在这个场景里面通常在签名匹配不通过时返回
2. 460: param error，参数异常
3. 500: unknow error，一些未知异常
4. 5001: meta device not found，指定的设备不存在
5. 6200: auth type mismatch，未授权认证类型错误
```