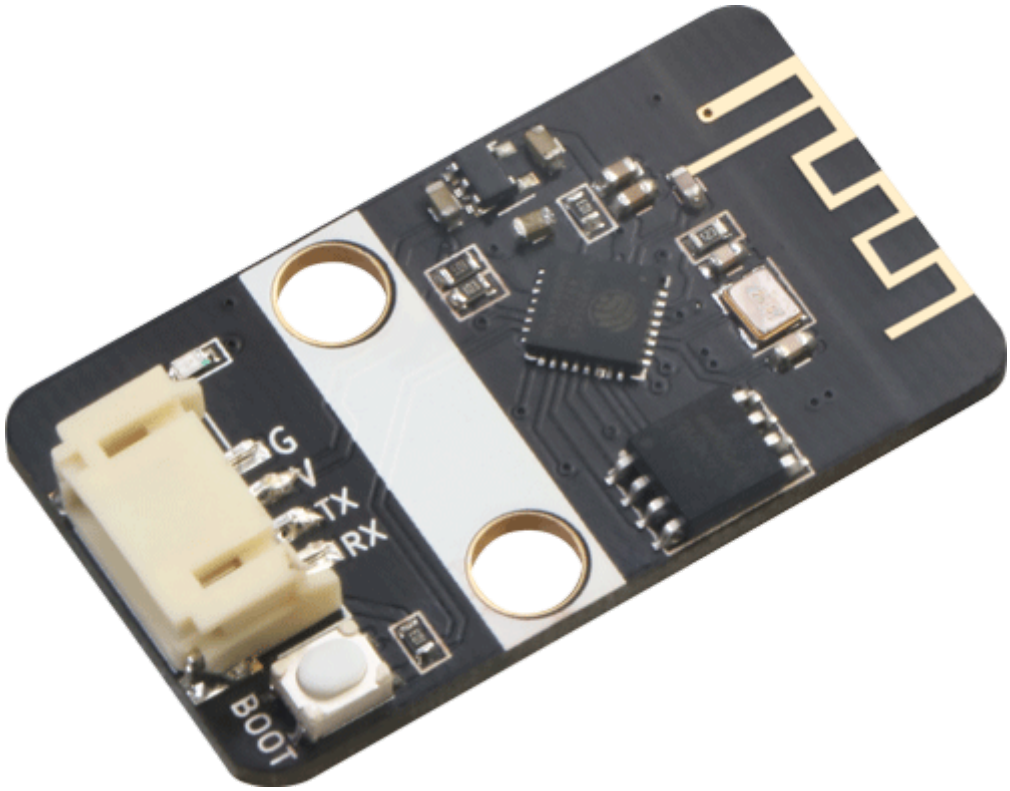
esp8266-mqtt物联网无线模块



概述

esp8266-mqtt无线模块是emakefun公司基于乐鑫科技的wifi芯片ESP8266基础上重新研发的串口转wifi的物联网模块，该模块采用AT配置方式来支持wifi无线通信，AT指令全面兼容乐鑫官方指令库（V3.0.0），在此基础上添加了MQTT指令，并且全部封装成scratch，mixly，Makecode图形化编程块支持arduino，micro:bit。让用户非常容易接收和发送物联网信息，远程物联网控制从未如此简单。

模块特点：

- 内置低功率 32 位 CPU：可以兼作应用处理器
- 内置协议：TCP/IP 协议栈
- 加密类型：WPA WPA2/WPA2–PSK
- 支持乐鑫官方AT标准指令集
- 支持连接标准MQTT协议和TTL串口到无线的应用

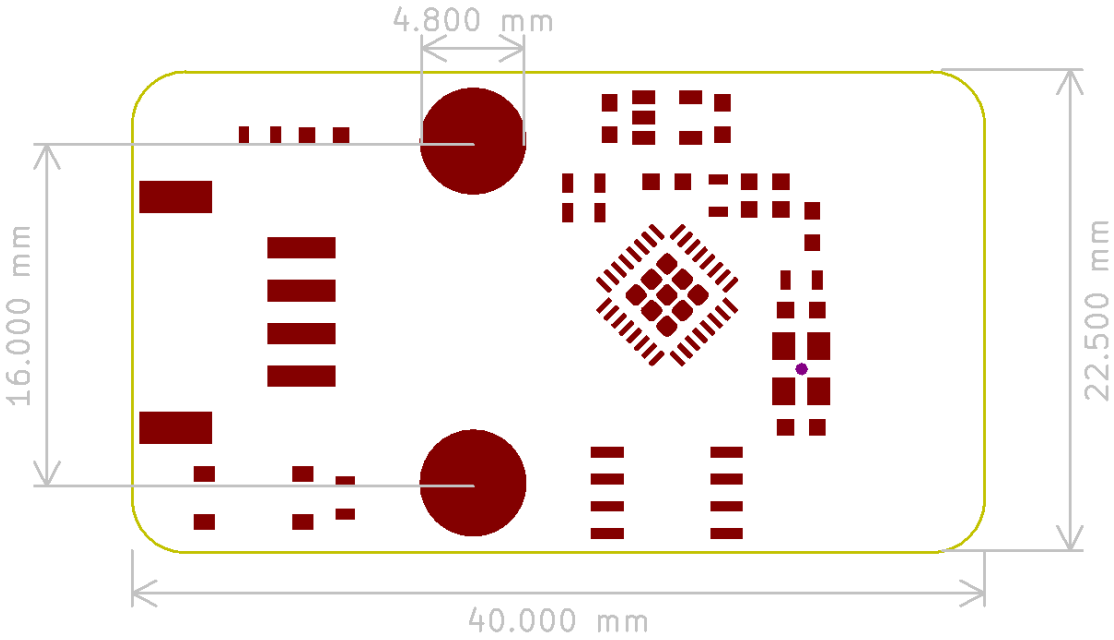
模块参数

- 工作电压：5V
- 接口速率：9600 bps
- 无线频率：2.4GHz
- 接口类型：PH2.0-4Pin (G V TX TX)
- 无线模式：IEEE802.11b/g/n
- SRAM：160KB
- 外置Flash：4MB
- 支持低功耗：<240mA
- 模块尺寸：4 * 2.1cm

- 安装方式：M4螺钉螺母固定

引脚名称	描述
G	GND
V	5V
TX	串口发送端
RX	串口接收端

机械尺寸图



MQTT扩展AT指令

序号	指令	描述	详情
1	AT+MQTTUSERCFG	配置 MQTT 用户属性	查看详情
2	AT+MQTTCONNCFG	配置 MQTT 连接属性	查看详情
3	AT+MQTTCONN	连接指定 MQTT broker	查看详情
4	AT+MQTTCONN?	查询 AT 已连接的 MQTT broker	查看详情
5	AT+ALIYUN_MQTTCONN?	连接指定的阿里云MQTT broker	查看详情
6	AT+MQTTPUB	在 LinkID上通过 topic 发布数据 data，data 为字符串消息	查看详情
7	AT+MQTTPUBRAW	在 LinkID 上通过 topic 发布数据 data，data 为二进制数据	查看详情
8	AT+MQTTSUB	订阅指定连接的 MQTT 主题, 可重复多次订阅不同 topic	查看详情
9	AT+MQTTSUB?	查询 MQTT 所有连接上已订阅的 topic	查看详情

序号	指令	描述	详情
10	AT+MQTTUNSUB	取消订阅指定连接的 MQTT 主题, 可多次取消不同订阅 topic	查看详情
11	AT+MQTTCLEAN	关闭 MQTT Client 为 LinkID 的连接, 并释放内部占用的资源	查看详情

AT+MQTTUSERCFG - 配置 MQTT 用户属性

设置指令:AT+MQTTUSERCFG=,,<"client_id">,<"username">,<"password">,<cert_key_ID>,<CA_ID>,<"path">

功能:设置 MQTT 用户配置

响应:OK或ERROR

参数说明:

- client_id: 对应 MQTT client ID, 用于标志 client 身份, 最长 256 字节
- username: 用于登录 MQTT broker 的 用户名, 最长 64 字节
- password: 用于登录 MQTT broker 的 密码, 最长 64 字节
- cert_key_ID: 证书 ID, 目前支持一套 cert 证书, 参数为 0
- CA_ID: CA ID, 目前支持一套 CA 证书, 参数为 0
- path: 资源路径, 最长 32 字节
- LinkID: 当前只支持 0

  举例： AT+MQTTUSERCFG=0,1,"ESP8266","emakefun","1234567890",0,0,""

  解析： “ESP8266”是对应主板的名称——<"client_id">

  “emakefun”是用户名——<"username">

  “1234567890”是用户密码——<"password">

  “0”默认的——<cert_key_ID>

  “0”默认的——<CA_ID>

  “ ”中间是路径——<"path">

AT+MQTTCONNCFG - 配置 MQTT 连接属性

设置指令:AT+MQTTCONNCFG=,,<disable_clean_session>,<"lwt_topic">,<"lwt_msg">,<lwt_qos>,<lwt_retain>

功能:设置 MQTT 连接配置

响应:OK或ERROR

参数说明:

- LinkID: 当前只支持 0
- keepalive: MQTT PING 超时时间,范围为 [60, 7200], 单位为秒. 默认 120
- disable_clean_session: MQTT 清理会话标志, 参数为 0 或 1, 默认为 0
- lwt_topic: 遗嘱 topic, 最长 64 字节
- lwt_msg: 遗嘱 message, 最长 64 字节

- lwt_qos: 遗嘱 QoS, 参数可选 0, 1, 2, 默认为 0
- lwt_retain: 遗嘱 retain, 参数可选 0, 1, 默认为 0

  举例：AT+MQTTCONNCFG=0,120,0,topic,msg,0,0

  解析：120 超时时间，范围60-7200——

  0 MQTT 清理会话——<disable_clean_session>

  topic 主题——<"lwt_topic">

  msg 内容——<"lwt_msg">

AT+MQTTCONN - 连接服务器的IP地址

设置指令:AT+MQTTCONN=,<"host">,,

功能:连接指定 MQTT broker

响应:OK或ERROR

举例：AT+MQTTCONN=0,"47.111.117.220",1883,0

解析："47.111.117.220" MQTT域名，IP地址——<"host">

1883 MQTT端口，一般情况下默认1833——

0 代表不重连MQTT，1 会一直重连——

AT+MQTTCONN? - 查询连接状态

功能:查询 AT 已连接的 MQTT broker

设置指令:AT+MQTTCONN=,,<"host">,,<"path">,

参数说明:

- LinkID: 当前只支持 0
- host: 连接 MQTT broker 域名, 最大 128 字节
- port: 连接 MQTT broker 端口, 最大 65535
- path: 资源路径, 最长 32 字节
- reconnect: 是否重连 MQTT, 若设置为 1, 需要消耗较多内存资源
- state: MQTT 当前状态, 状态说明如下:

  0: 连接未初始化

  1: 已设置 MQTTUSERCFG

  2: 已设置 MQTTCONNCFG

  3: 连接已断开

  4: 已建立连接

  5: 已连接, 但未订阅 topic

  6: 已连接, 已订阅过 topic

  scheme:

  1: MQTT over TCP

  2: MQTT over TLS(no certificate verify) 无验证

  3: MQTT over TLS(verify server certificate) 验证服务器证书

4: MQTT over TLS(provide client certificate) 提供客户端证书

5: MQTT over TLS(verify server certificate and provide client certificate) 验证服务器证书，提供客户端证书

6: MQTT over WebSocket(based on TCP) 基于TCP

7: MQTT over WebSocket Secure(based on TLS, no certificate verify) 基于TLS，没有证书验证

8: MQTT over WebSocket Secure(based on TLS, verify server certificate) 基于TLS，验证服务器证书

9: MQTT over WebSocket Secure(based on TLS, provide client certificate) 基于TLS，提供客户端证书

10: MQTT over WebSocket Secure(based on TLS, verify server certificate and provide client certificate) 基于TLS，验证服务器证书并提供客户端证书

举例： AT+MQTTCONN=0,2,2,"47.111.117.220",1883,"",0

AT+ALIYUN_MQTTCONN?

设置指令:AT+ALIYUN_MQTTCONN=<"host">,,<"ProductKey">,<"DeviceName">,<"DeviceSecret">

功能:连接指定的阿里云MQTT broker

参数说明:

- host: 连接阿里云的MQTT broker 域名, 详情请参考**阿里云域名格式**
- port: 连接 MQTT broker 端口, 最大 65535 默认 1883
- ProductKey: 设备所属产品的ProductKey，即物联网平台为产品颁发的全局唯一标识符
- DeviceName: 设备在产品内的唯一标识符。DeviceName与设备所属产品的ProductKey组合，作为设备标识，用来与物联网平台进行连接认证和通信。
- DeviceSecret: 物联网平台为设备颁发的设备密钥，用于认证加密。需与DeviceName成对使用。

响应:OK或ERROR

举例： AT+MQTTCONN=0,"192.168.1.17",1883,0

AT+MQTTPUB

设置指令:AT+MQTTPUB=,<"topic">,<"data">,,

功能:在 LinkID上通过 topic 发布数据 data, 其中 data 为字符串消息, 若要发布二进制,请使用AT+MQTTPUBRAW

响应:OK或ERROR

参数说明:

- LinkID: 当前只支持 0
- topic: 发布主题, 最长 64 字节
- data: 发布消息, data 不能包含 \0, 请确保整条 AT+MQTTPUB 不超过 AT 指令的最大长度限制
- qos: 发布服务质量, 参数可选 0,1,2, 默认为 0
- retain: 发布 retain

  举例： AT+MQTTPUB=0,"topic","test",0,retain

AT+MQTTPUBRAW

设置指令:AT+MQTTPUBRAW=,<"topic">,,,

功能:在 LinkID 上通过 topic 发布数据 data, 其中 data 为二进制数据

响应:OK或ERROR，等待用户输入 length 大小数据, 之后响应如下:+MQTTPUB:FAIL或+MQTTPUB:OK

参数说明:

- LinkID: 当前只支持 0
- topic: 发布主题, 最长 64 字节
- length: 要发布消息长度, 长度受限于当前可用内存
- qos: 发布服务质量, 参数可选 0,1,2, 默认为 0
- retain: 发布 retain
- AT port 未收到指定 length 长度的数据, 将一直等待, 在此期间接收到的数据都会当成普通数据

  举例：AT+MQTTPUBRAW=0,"topic",2,0,retain

AT+MQTTSUB

设置指令:AT+MQTTSUB=,<"topic">,

功能:订阅指定连接的 MQTT 主题, 可重复多次订阅不同 topic

响应:OK或ERROR，当收到对应主题订阅的 MQTT 消息时, 将按照如下格式打印消息内容

+MQTTSUBRECV:,<"topic">,<data_length>,data

如果订阅已订阅过的主题, 仍无条件向 MQTT broker 订阅, Log 口打印 ALREADY SUBSCRIBE

查询指令:

举例：AT+MQTTSUB=0,"topic",0

AT+MQTTSUB?

功能:查询 MQTT 所有连接上已订阅的 topic

响应:

- +MQTTSUB:,,<"topic1">,
- +MQTTSUB:,,<"topic2">,
- +MQTTSUB:,,<"topic3">,
- OK
- 1
- 2
- 3
- 4
- 5
- 或ERROR

参数说明:

- LinkID: 当前只支持 0
- state: MQTT 当前状态, 状态说明如下:

- 0: 连接未初始化
- 1: 已设置 MQTTUSERCFG
- 2: 已设置 MQTTCONNCFG
- 3: 连接已断开
- 4: 已建立连接
- 5: 已连接, 但未订阅 topic
- 6: 已连接, 已订阅过 topic
- topic*: 订阅过的主题
- qos: 订阅过的 QoS

AT+MQTTUNSUB

设置指令:AT+MQTTUNSUB=,<"topic">

功能:取消订阅指定连接的 MQTT 主题, 可多次取消不同订阅 topic

响应:OK或ERROR

参数说明:

- LinkID: 当前只支持 0
- topic: 取消订阅主题, 最长 64 字节
  如果取消未订阅的主题, 仍无条件向 MQTT broker 取消订阅, Log 口打印 NO UNSUBSCRIBE

  举例：AT+MQTTUNSUB=0,"topic"

AT+MQTTCLEAN

设置指令:AT+MQTTCLEAN=

功能:关闭 MQTT Client 为 LinkID 的连接, 并释放内部占用的资源

响应:OK或者ERROR

参数说明:

LinkID: 当前只支持 0

举例：AT+MQTTCLEAN=0

arduino 应用场景

AT串口测试

```
#include "Arduino.h"
#include "SoftwareSerial.h"
SoftwareSerial Serial1(5, 6); // RX, TX

void setup()
{
  Serial.begin(115200); // serial port used for debugging
  Serial1.begin(9600);  // your ESP's baud rate might be different
}

void loop()
{
  if(Serial1.available())  // check if the ESP is sending a message
  {
```

```
    while(Serial1.available())
    {
      int c = Serial1.read(); // read the next character
      Serial.write((char)c);  // writes data to the serial monitor
    }
  }

  if(Serial.available())
  {
    // wait to let all the input command in the serial buffer
    delay(10);

    // read the input command in a string
    String cmd = "";
    while(Serial.available())
    {
      cmd += (char)Serial.read();
    }

    // print the command and send it to the ESP
    Serial.println();
    Serial.print(">>>> ");
    Serial.println(cmd);

    // send the read character to the ESP
    Serial1.print(cmd);
  }
}
```

arduino示例程序

[下载最新库程序]

```
#include "WiFiEsp.h"
#include "WifiEspMqtt.h"
#include "SoftwareSerial.h"
SoftwareSerial esp8266_serial(5, 6); // RX, TX
uint32_t _startMillis = 0;

WiFiEspMqtt esp8266;

void setup()
{
  Serial.begin(115200);
  esp8266_serial.begin(9600);
  WiFi.init(&esp8266_serial);
  assertEquals("Firmware version", WiFi.firmwareVersion(), "3.0.2");
  assertEquals("Status is (WL_DISCONNECTED)", WiFi.status(), WL_DISCONNECTED);

  esp8266.mqtt_connect("192.168.2.65", 1883, 0);
  // esp8266.mqtt_connect_cfg(120, 1, "lws_topic", "lws_kill", 0);
  esp8266.mqtt_usercfg("esp8266_client", "emakefun123", "12345678");

  if (WiFi.begin("emakefun", "501416wf") == WL_CONNECTED)
  {
     Serial.println("wifi connected");
     esp8266.mqtt_sub("topic_1", 0); //订阅topic_1主题，qos为0
```

```
  }
  esp8266.mqtt_public("topic_led", "on", 0);
  delay(30000);
}

void loop()
{
    if (esp8266.mqtt_receive())
    {

     Serial.print("topic:");
     Serial.println(esp8266.mqtt_topic);
     Serial.print("message:");
     Serial.println(esp8266.mqtt_message);

    }
}
```

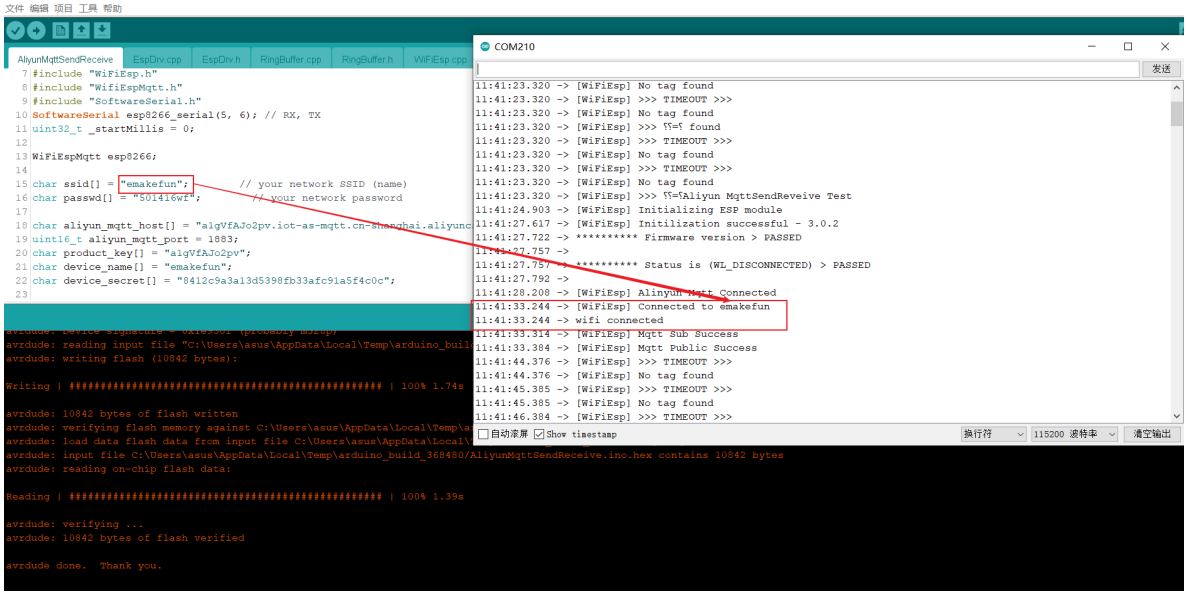
Arduino连接阿里云案例分析

```
/*
 WiFiEsp test: BasicTest

 Performs basic connectivity test and checks.
*/

#include "WiFiEsp.h"
#include "WifiEspMqtt.h"
#include "SoftwareSerial.h"
SoftwareSerial esp8266_serial(5, 6); // RX, TX
uint32_t _startMillis = 0;

WiFiEspMqtt esp8266;

char ssid[] = "emakefun";         // your network SSID (name)
char passwd[] = "501416wf";         // your network password

char aliyun_mqtt_host[] = "a1gVfAJo2pv.iot-as-mqtt.cn-shanghai.aliyuncs.com";
// 阿里云物联网服务器host
uint16_t aliyun_mqtt_port = 1883;     // 阿里云物联网服务器端口
char product_key[] = "a1gVfAJo2pv";  // 设备所属产品的ProductKey，即物联网平台为产品颁发的全局唯一标识符
char device_name[] = "emakefun";      // 设备在产品内的唯一标识符。DeviceName与设备所属产品的ProductKey组合，作为设备标识，用来与物联网平台进行连接认证和通信。
char device_secret[] = "8412c9a3a13d5398fb33afc91a5f4c0c";  // 物联网平台为设备颁发的设备密钥，用于认证加密。需与DeviceName成对使用。

void setup()
{
  Serial.begin(115200);
  esp8266_serial.begin(9600);
  Serial.println("Aliyun MqttSendReveive Test");
  WiFi.init(&esp8266_serial);
  assertEquals("Firmware version", WiFi.firmwareVersion(), "3.0.2");
  assertEquals("Status is (WL_DISCONNECTED)", WiFi.status(), WL_DISCONNECTED);
  esp8266.mqtt_connect_aliyun(aliyun_mqtt_host, aliyun_mqtt_port, product_key,
device_name, device_secret, 0);
```

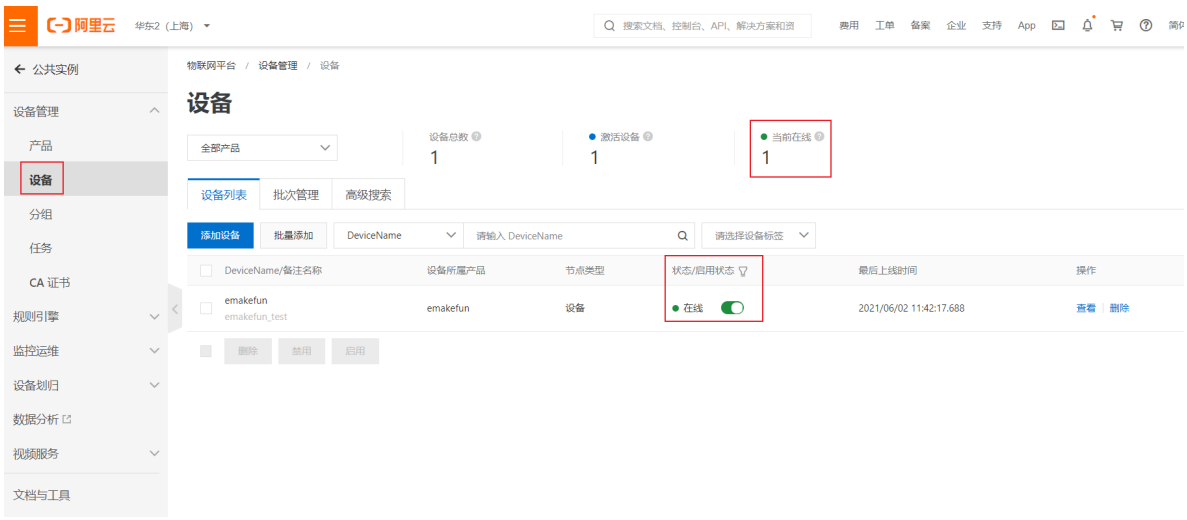
```
  if (WiFi.begin(ssid, passwd) == WL_CONNECTED)
  {
     Serial.println("wifi connected");

     esp8266.mqtt_sub("/a1gVfAJo2pv/emakefun/user/get", 0);   // 订阅topic
  }
  esp8266.mqtt_public("/a1gVfAJo2pv/emakefun/user/add", "on", 0); // 发布topic 数
据为 "on"
  delay(10000);
}

void loop()
{
    if (esp8266.mqtt_receive())
    {

     Serial.print("topic:");
     Serial.println(esp8266.mqtt_topic);    // 打印订阅的topic
     Serial.print("message:");
     Serial.println(esp8266.mqtt_message);  // 打印订阅的topic的数据

    }
}

void assertNotEquals(const char* test, int actual, int expected)
{
  if(actual!=expected)
    pass(test);
  else
    fail(test, actual, expected);
}

void assertEquals(const char* test, int actual, int expected)
{
  if(actual==expected)
    pass(test);
  else
    fail(test, actual, expected);
}

void assertEquals(const char* test, char* actual, char* expected)
{
  if(strcmp(actual, expected) == 0)
    pass(test);
  else
    fail(test, actual, expected);
}


void pass(const char* test)
{
  Serial.print(F("********** "));
  Serial.print(test);
  Serial.println(" > PASSED");
  Serial.println();
}
```

```
void fail(const char* test, char* actual, char* expected)
{
  Serial.print(F("********** "));
  Serial.print(test);
  Serial.print(" > FAILED");
  Serial.print(" (actual=\"");
  Serial.print(actual);
  Serial.print("\", expected=\"");
  Serial.print(expected);
  Serial.println("\")");
  Serial.println();
  delay(10000);
}

void fail(const char* test, int actual, int expected)
{
  Serial.print(F("********** "));
  Serial.print(test);
  Serial.print(" > FAILED");
  Serial.print(" (actual=");
  Serial.print(actual);
  Serial.print(", expected=");
  Serial.print(expected);
  Serial.println(")");
  Serial.println();
  delay(10000);
}
```

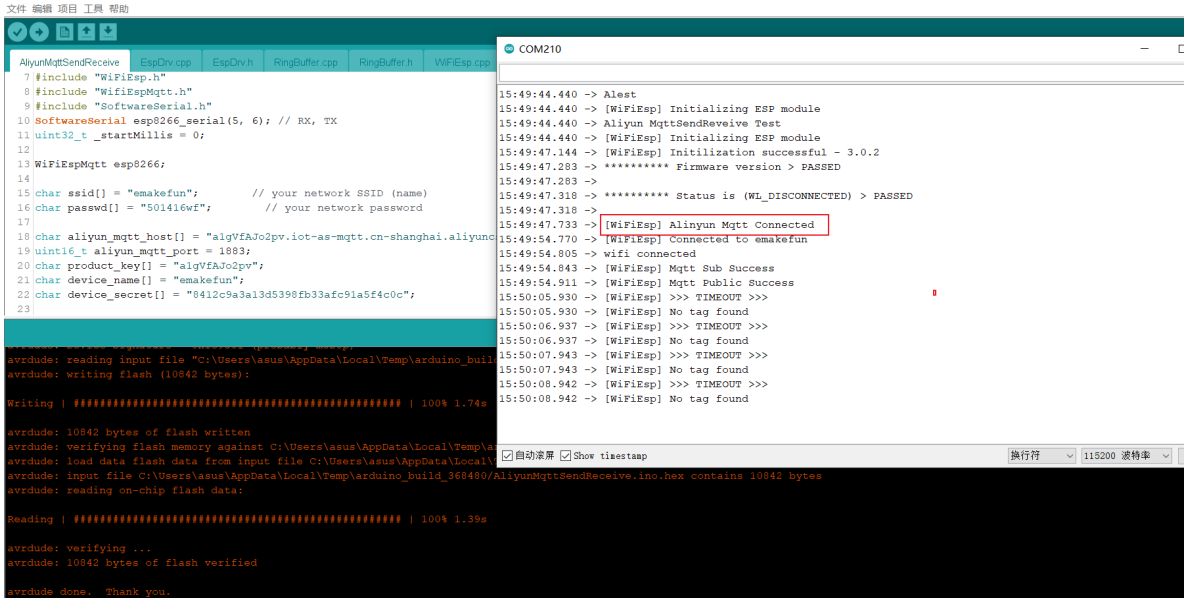
当连接上WIFI时，物联网模块的蓝灯会常亮，否则蓝灯会闪烁，同时串口监视器会显示WiFi connected,如下图所示。



当连上阿里云服务器时，选择阿里云设备菜单时，会显示当前在线的设备数量，并且所连接的设备的状态为在线状态，如下图所示。



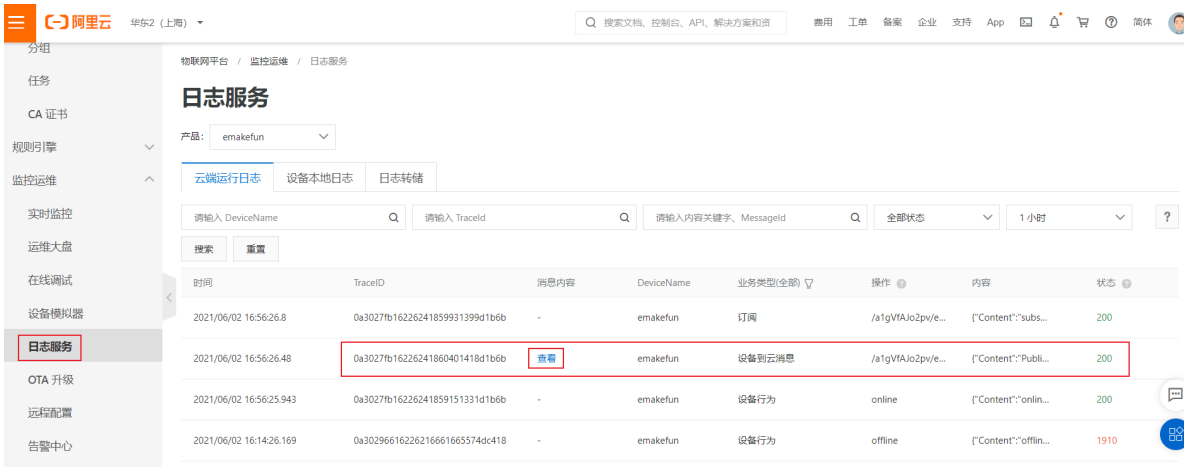
同时在串口监视器里面可以看到是否连接成功，如下图所示。

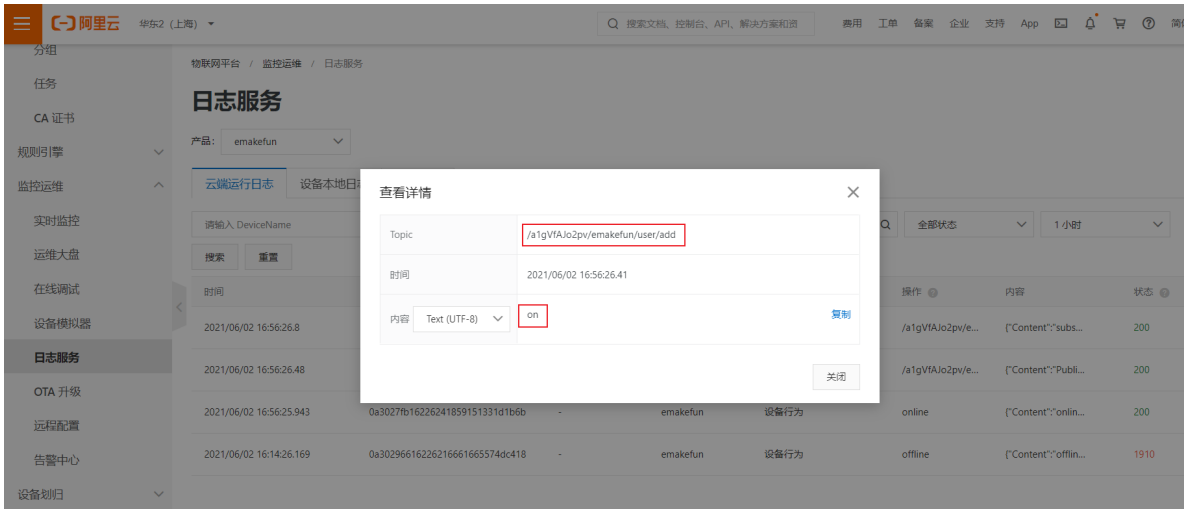


当前程序会在连接阿里云服务器成功之后，会向服务器发布和订阅相关的主题(topic)。

发布: 向云端发送数据。比如温度、湿度、气压值、停车位.....

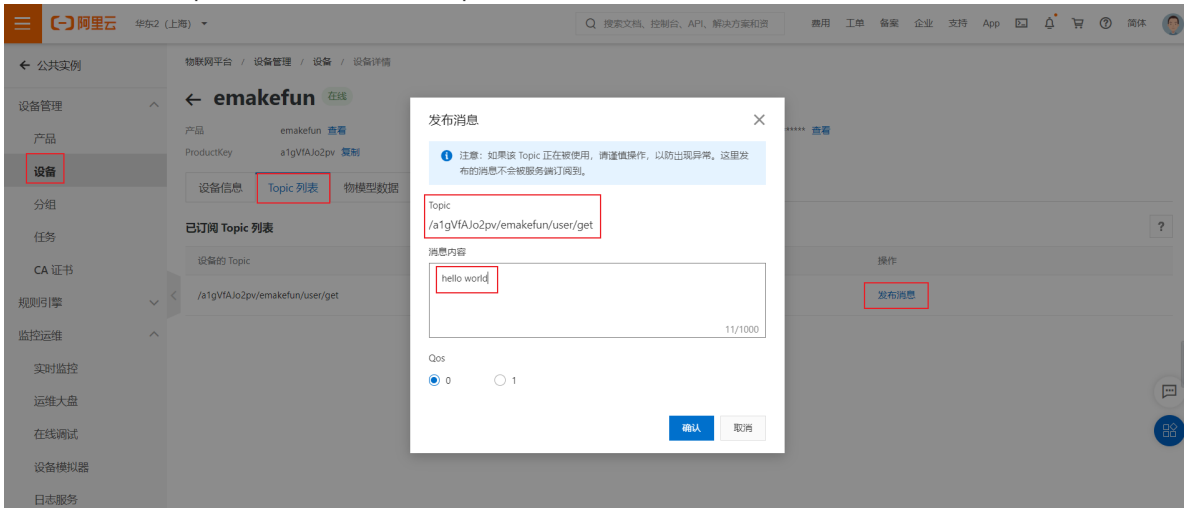
可以在阿里云的监控运维->日志服务里面可以看到当前发布的记录，并且点击查看可以看到发送的数据。



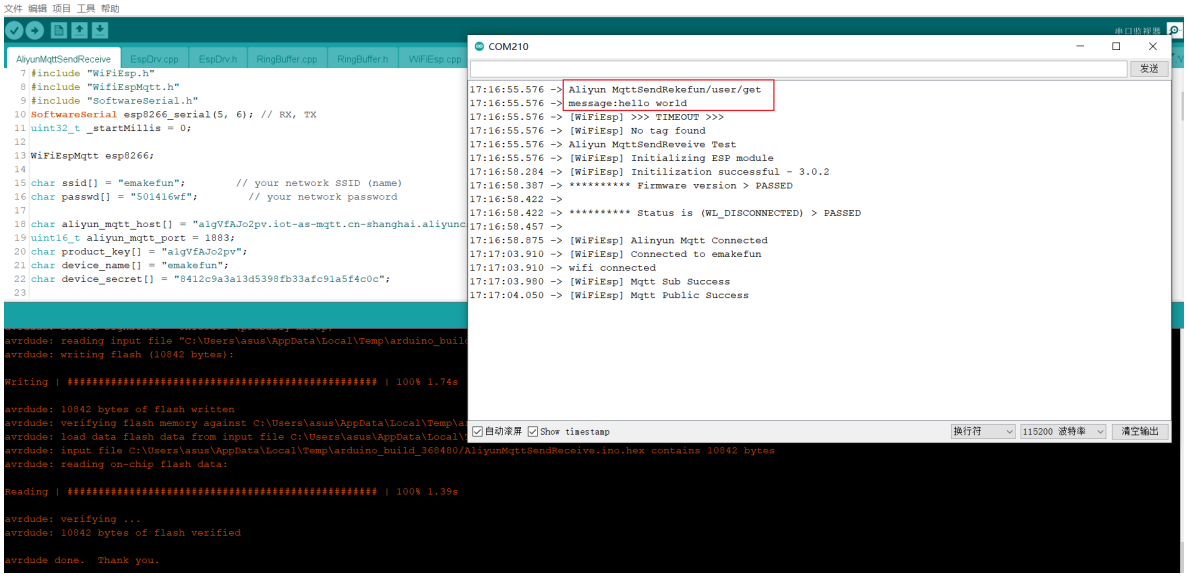


订阅: 获取云端的数据。比如天气预报.....

可在设备的topic列表里找到相应的topic，并且点击发布消息，发布想要发布的数据。



同时在串口监视器里面我们可以看到订阅的数据。



[下载最新示例程序]

MQTT物联网需要先导入或者更新最新PH2.0的Mixly图形化库具体请看

[物联网模块的mixly参考链接]

[microbit使用参考示例]