第一章 物联网最常用的 MQTT

1. 学习目的及目标

- 掌握 MQTT 原理和工作过程
- 掌握 ESP32 的 MQTT 程序设计

2. MQTT 原理和工作过程讲解

MQTT(消息队列遥测传输)是 ISO 标准(ISO/IEC PRF 20922)下基于发布/订阅范式的消息协议。它工作在 TCP/IP 协议族上，是为硬件性能低下的远程设备以及网络状况糟糕的情况下而设计的发布/订阅型消息协议，为此，它需要一个消息中间件 （服务器）。

通过 MQTT 协议，目前已经扩展出了数十个 MQTT 服务器端程序，可以通过 PHP，JAVA，Python，C，C#等系统语言来向 MQTT 发送相关消息。

此外，国内很多企业都广泛使用 MQTT 作为 Android 手机客户端与服务器端推送消息的协议。MQTT 由于开放源代码，耗电量小等特点。在物联网领域，传感器与服务器的通信，信息的收集，MQTT 都可以作为考虑的方案之一。在未来 MQTT 会进入到我们生活的各各方面。

所以，如果物联网设备想要联网，MQTT 是不二选择。

2.1. MQTT 特点

MQTT 协议是为大量计算能力有限，且工作在低带宽、不可靠的网络的远程传感器和控制设备通讯而设计的协议，它具有以下主要的几项特性：

- 使用发布/订阅消息模式，提供一对多的消息发布，解除应用程序耦合；
- 对负载内容屏蔽的消息传输；
- 使用 TCP/IP 提供网络连接；
- 有三种消息发布服务质量：
  - “至多一次”，消息发布完全依赖底层 TCP/IP 网络。会发生消息丢失或重复。这一级别可用于如下情况，环境传感器数据，丢失一次读记录无所谓，因为不久后还会有第二次发送。
  - “至少一次”，确保消息到达，但消息重复可能会发生。
  - “只有一次”，确保消息到达一次。这一级别可用于如下情况，在计费系统中，消息重复或丢失会导致不正确的结果。
- 小型传输，开销很小（固定长度的头部是 2 字节），协议交换最小化，以降低网络流量；

这就是为什么在介绍里说它非常适合物联网领域，要知道嵌入式设备的运算能力和带宽都相对薄弱，使用这种协议来传递消息再适合不过了。

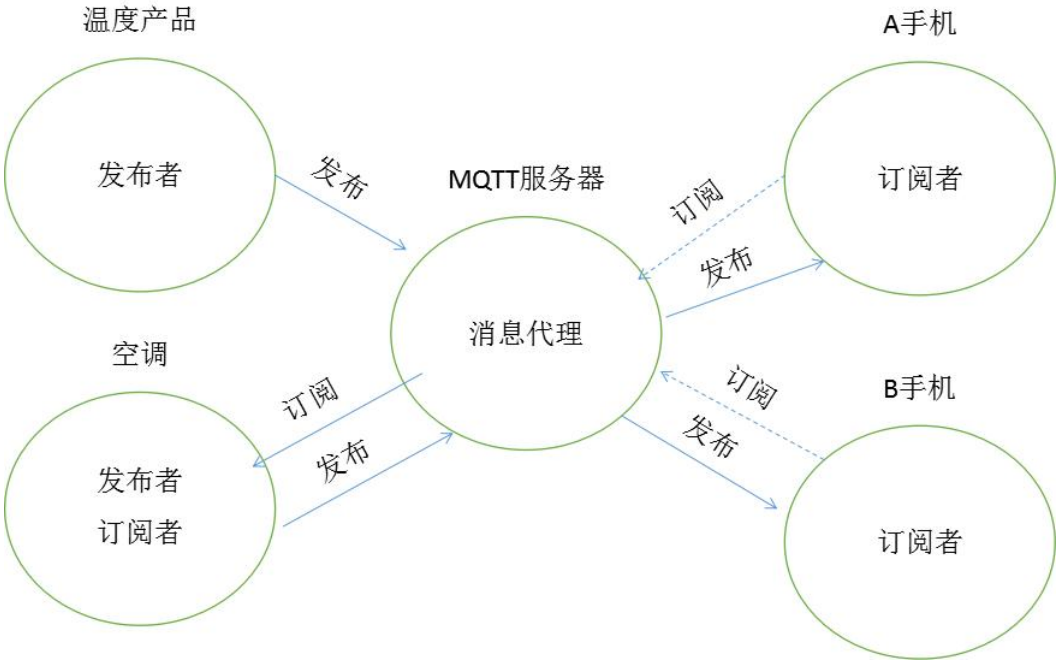
2.2. 实现方式

实现 MQTT 协议需要客户端和服务器端通讯完成，在通讯过程中，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）和负载（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。



- MQTT 服务器的主要工作是数据分发，没有数据保存功能。
- 可以订阅自己发布的主题，服务器就是回发测试。
- MQTT 让逻辑变得更清晰，需要什么订阅什么。
- 走标准化流程，解放了私有协议制定、实现、调试、测试一整套复杂的流程。

3. Win7 搭建本地 MQTT 服务器（参考原文）

目前主流的 Broker 有以下 3 个：

- Mosquitto：https://mosquitto.org/
- VerneMQ：https://vernemq.com/

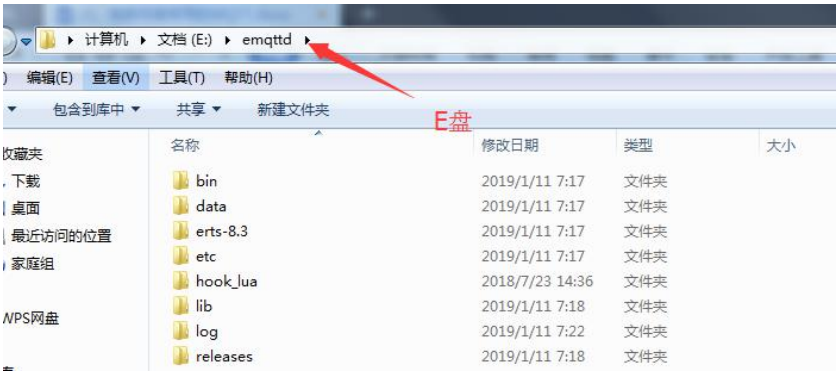
EMQTT：http://emqtt.io/ 此处，我们使用搭建 EMQTT。

3.1. 下载 MQTT 服务器压缩包

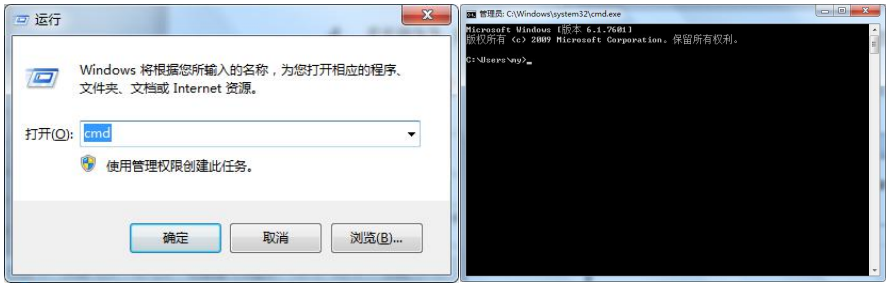
稳定版：emqttd-2.3.11 发布于 2018/07/23

3.2. 解压到电脑

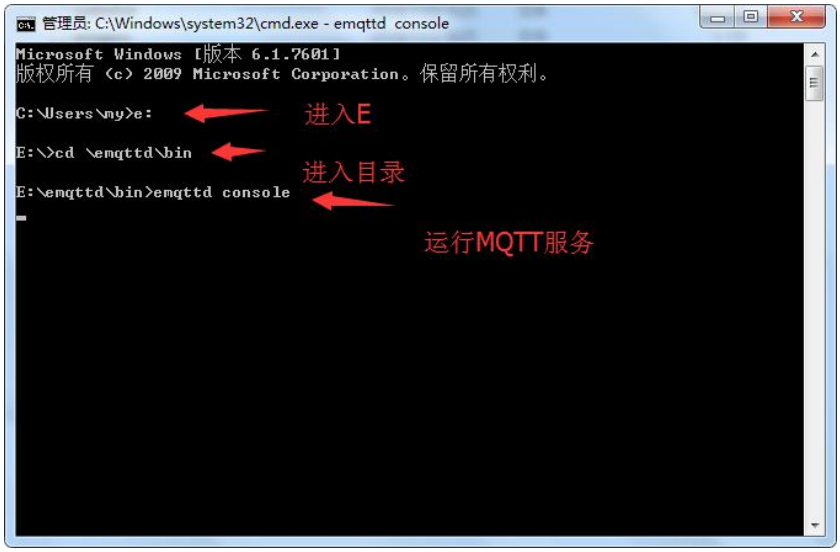
我这里是解压到 E 盘。



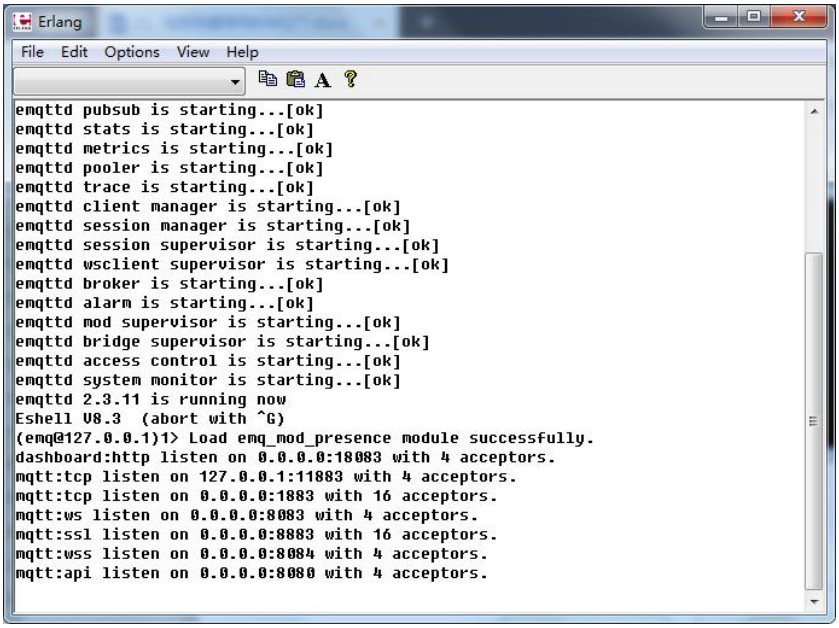
3.3. 打开终端



3.4. 进入目录运行 MQTT 服务

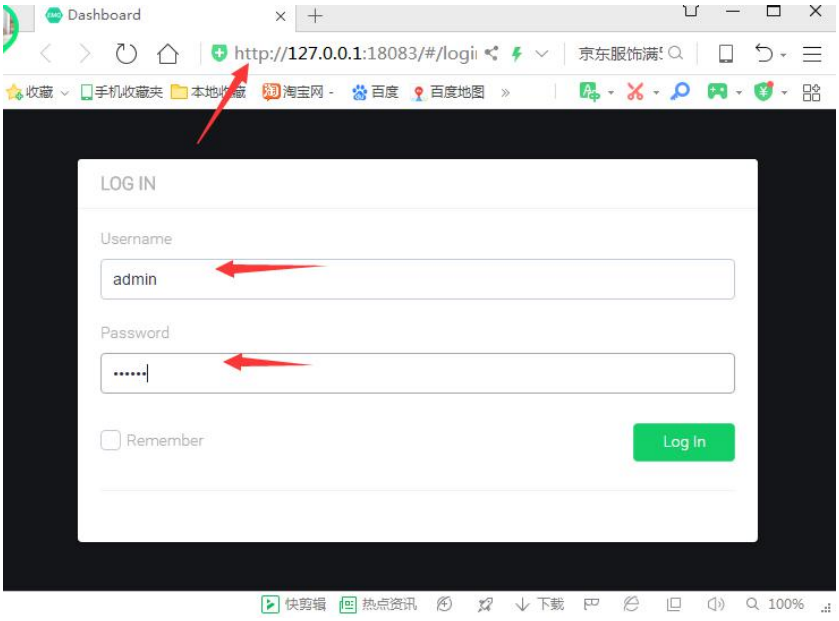


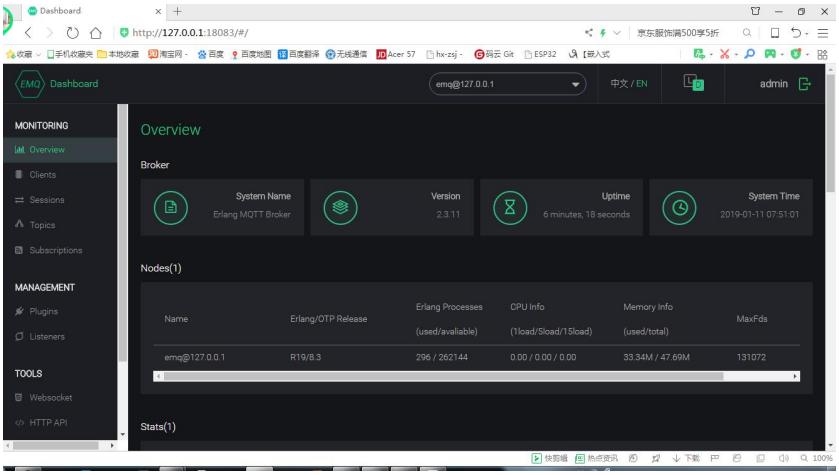
执行成功会弹出下面窗口，不成功就关掉 cmd 重新试下。



3.5. 测试 MQTT 服务器

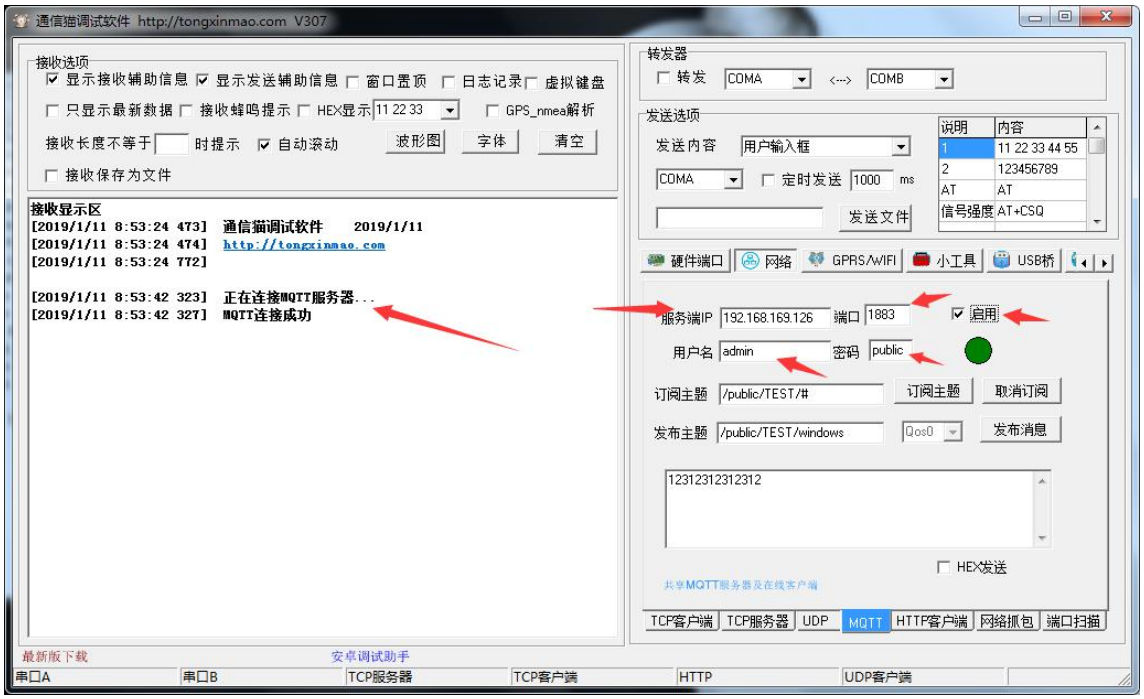
打开浏览器-> 输入 http://127.0.0.1:18083 -> 用户名:admin-> 密码:public-> 进入如下界面



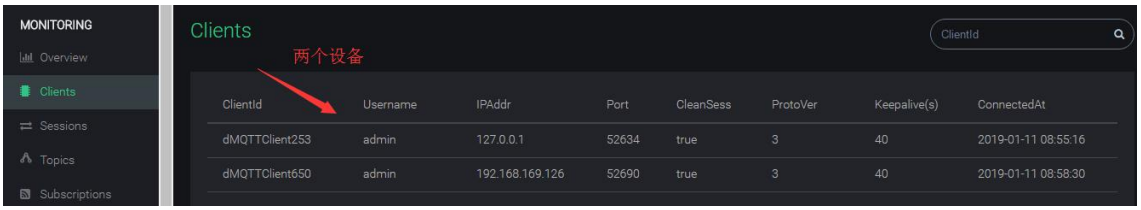


3.6. 连接 MQTT 服务器。

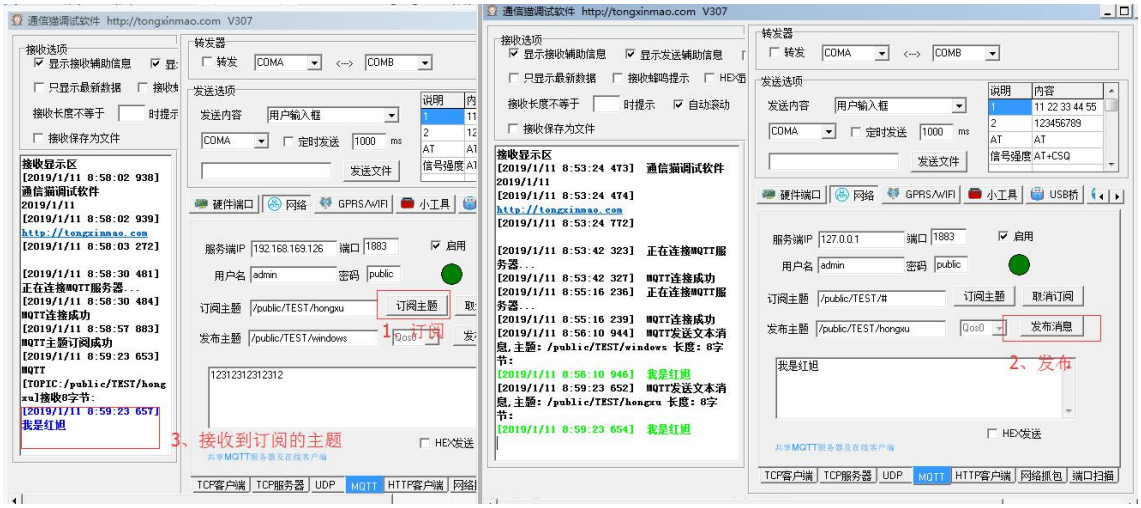
这个软件不怎么好用，连不上，关掉再连。使用 127.0.0.1 也可以。



连上了两个

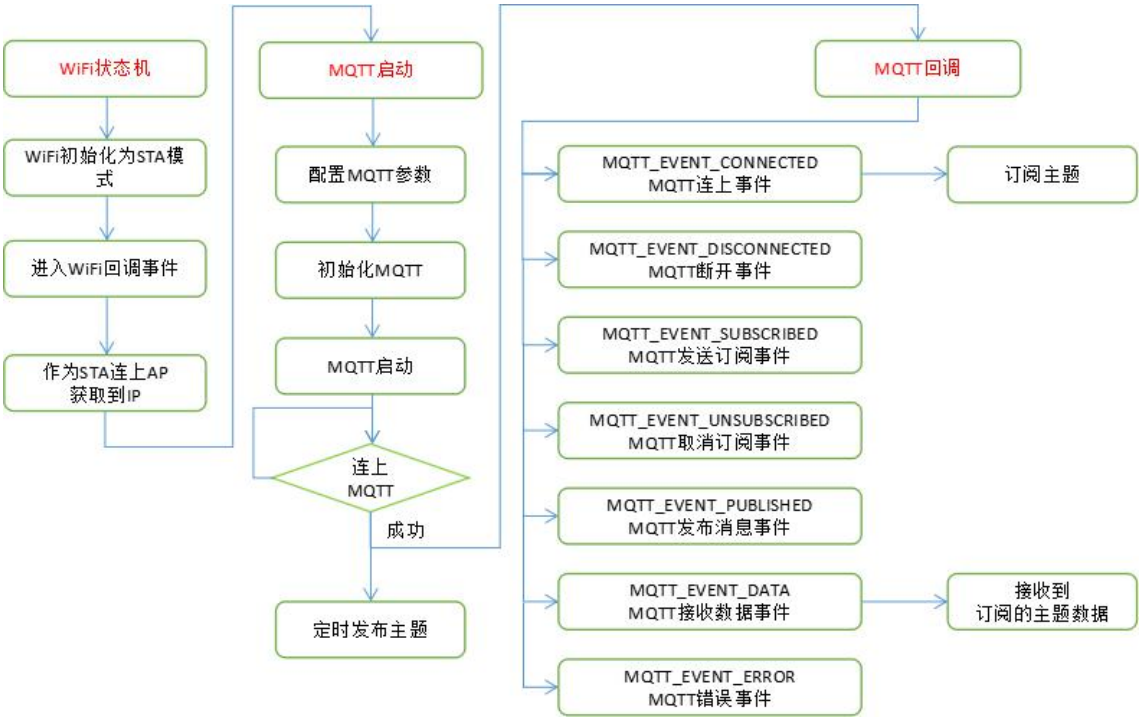


订阅->发布，就可以数据发送了。



4. ESP32 的 MQTT 软件设计

4.1. ESP32 的 MQTT 详细过程逻辑



4.2. MQTT 接口介绍

- MQTT 配置信息 `esp_mqtt_client_config_t;`

参数原型	esp_mqtt_client_config_t
参数功能	MQTT 配置信息
参数	typedef struct { mqtt_event_callback_t event_handle; /*回调*/ const char *host; /*!< MQTT 服务器域名(ipv4 as string)*/ const char *uri; /*!< MQTT 服务器域名 */ uint32_t port; /*!< MQTT 服务器端口*/ const char *client_id; /*MQTT Client 的名字默认是 ESP32_加上 MAC 后 3hex*/ const char *username; /*MQTT 用户名*/ const char *password; /*MQTT 密码*/ const char *lwt_topic; /*!< LWT 主题，默认为空*/ const char *lwt_msg; /*!< LWT 信息，默认为空*/ int lwt_qos; /*!< LWT 消息质量*/ int lwt_retain; /*!< LWT 保留消息标志*/ int lwt_msg_len; /*!< LWT 消息长度*/ int disable_clean_session; /*!< mqtt clean session,默认为真*/ int keepalive; /*MQTT 心跳，默认 120 秒 */ bool disable_auto_reconnect; /*错误，断开后重连，true 不连*/ void *user_context; /*用户信息 */ int task_prio; /*!< MQTT 任务优先级，默认为 5,可以在 make menuconfig 中修改*/ int task_stack; /*!< MQTT 任务堆栈大小,默认 6144 bytes,可以在 make menuconfig 中修改*/ int buffer_size; /*!< MQTT 收发缓存，默认 1024 */ const char *cert_pem; /*指向用于服务器验证（使用 SSL）的 PEM 格式的证书数据的指针，默认值为空，不需要验证服务器 */ const char *client_cert_pem; /*指向用于 SSL 相互身份验证的 PEM 格式的证书数据的指针，默认值为空，如果不需要相互身份验证，则不需要。如果不为空，还必须提供"客户机密钥"。*/ const char *client_key_pem; /*指向用于 SSL 相互身份验证的 PEM 格式的私钥数据的指针，默认值为空，如果不需要相互身份验证，则不需要。如果不为空，还必须提供"client-cert-pem"。*/ esp_mqtt_transport_t transport; /*覆盖 URI 传输*/ } esp_mqtt_client_config_t;

➢ MQTT Client 初始化函数：esp_mqtt_client_init();

函数原型	esp_mqtt_client_handle_t esp_mqtt_client_init (const esp_mqtt_client_config_t *config)
函数功能	MQTT Client 初始化函数
参数	[in] config：MQTT 配置参数，上面介绍了
返回值	MQTT Client 句柄

➢ MQTT Client 启动函数：esp_mqtt_client_start();

函数原型	esp_err_t esp_mqtt_client_start (esp_mqtt_client_handle_t client);
函数功能	MQTT Client 启动函数
参数	[in] client：MQTT Client 句柄

返回值	ESP_OK：成功 other ： 失败

➢ MQTT Client 停止函数：esp_mqtt_client_stop();

函数原型	esp_err_t esp_mqtt_client_stop(esp_mqtt_client_handle_t client);
函数功能	MQTT Client 停止函数
参数	[in] client：MQTT Client 句柄
返回值	ESP_OK：成功 other ： 失败

➢ MQTT Client 订阅主题函数：esp_mqtt_client_subscribe();

函数原型	esp_err_t esp_mqtt_client_subscribe (esp_mqtt_client_handle_t client, const char *topic, int qos)
函数功能	MQTT Client 订阅函数
参数	[in] client：MQTT Client 句柄 [in] topic：MQTT 主题 [in] qos：服务质量
返回值	ESP_OK：成功 other ： 失败

➢ MQTT Client 取消订阅主题函数：esp_mqtt_client_unsubscribe();

函数原型	esp_err_t esp_mqtt_client_unsubscribe (esp_mqtt_client_handle_t client, const char *topic)
函数功能	MQTT Client 取消订阅主题函数
参数	[in] client：MQTT Client 句柄 [in] topic：MQTT 主题
返回值	ESP_OK：成功 other ： 失败

➢ MQTT Client 发布主题函数：esp_mqtt_client_publish();

函数原型	int esp_mqtt_client_publish (esp_mqtt_client_handle_t client, const char *topic, const char *data, int len,

	int qos, int retain)
函数功能	MQTT Client 发布主题函数
参数	[in] client：MQTT Client 句柄 [in] topic：MQTT 主题 [in] data：数据 [in] len：长度，=0 表示 data 长度
返回值	消息 ID

- MQTT Client 注销函数：esp_mqtt_client_publish();

函数原型	esp_err_t esp_mqtt_client_destroy(esp_mqtt_client_handle_t client);
函数功能	MQTT Client 取消订阅主题函数
参数	[in] client：MQTT Client 句柄
返回值	ESP_OK：成功 other ：失败

4.3. 基于 TCP 的 MQTT 源码编写

- MQTT 初始化

```
static void mqtt_app_start(void)
{
    esp_mqtt_client_config_t mqtt_cfg = {
        .host = "192.168.2.104",           //MQTT 服务器 IP
        .event_handle = mqtt_event_handler, //MQTT 事件
        .port=1883,                        //端口
        .username = "admin",               //用户名
        .password = "public",              //密码
        // .user_context = (void *)your_context
    };
    client = esp_mqtt_client_init(&mqtt_cfg);
    esp_mqtt_client_start(client);
    //等 mqtt 连上
    xEventGroupWaitBits(mqtt_event_group, CONNECTED_BIT, false, true, portMAX_DELAY);
}
```

- MQTT 回调

```
static esp_err_t mqtt_event_handler(esp_mqtt_event_handle_t event)
{
    esp_mqtt_client_handle_t client = event->client;
    int msg_id;
```

```
    // your_context_t *context = event->context;
    switch (event->event_id) {
        case MQTT_EVENT_CONNECTED://MQTT 连上事件
            ESP_LOGI(TAG, "MQTT_EVENT_CONNECTED");
            xEventGroupSetBits(mqtt_event_group, CONNECTED_BIT);
            //发送订阅
            msg_id = esp_mqtt_client_subscribe(client, "/topic/qos1", 1);
            ESP_LOGI(TAG, "sent subscribe successful, msg_id=%d", msg_id);
            break;
        case MQTT_EVENT_DISCONNECTED://MQTT 断开连接事件
            ESP_LOGI(TAG, "MQTT_EVENT_DISCONNECTED");
            //mqtt 连上事件
            xEventGroupClearBits(mqtt_event_group, CONNECTED_BIT);
            break;
        case MQTT_EVENT_SUBSCRIBED://MQTT 发送订阅事件
            ESP_LOGI(TAG, "MQTT_EVENT_SUBSCRIBED, msg_id=%d", event->msg_id);
            msg_id = esp_mqtt_client_publish(client, "/topic/qos0", "订阅成功", 0, 0, 0);
            ESP_LOGI(TAG, "sent publish successful, msg_id=%d", msg_id);
            break;
        case MQTT_EVENT_UNSUBSCRIBED://MQTT 取消订阅事件
            ESP_LOGI(TAG, "MQTT_EVENT_UNSUBSCRIBED, msg_id=%d", event->msg_id);
            break;
        case MQTT_EVENT_PUBLISHED://MQTT 发布事件
            ESP_LOGI(TAG, "MQTT_EVENT_PUBLISHED, msg_id=%d", event->msg_id);
            break;
        case MQTT_EVENT_DATA://MQTT 接受数据事件
            ESP_LOGI(TAG, "MQTT_EVENT_DATA");
            printf("TOPIC=%.*s\r\n", event->topic_len, event->topic);   //主题
            printf("DATA=%.*s\r\n", event->data_len, event->data);      //内容
            break;
        case MQTT_EVENT_ERROR://MQTT 错误事件
            ESP_LOGI(TAG, "MQTT_EVENT_ERROR");
            xEventGroupClearBits(mqtt_event_group, CONNECTED_BIT);
            break;
    }
    return ESP_OK;
}
```

- 定时发布主题

```
while (1) {
        //发布主题
        //PC 订阅了，会收到这条信息
        esp_mqtt_client_publish(client, "/topic/qos0", "Hello MQTT ,I am HongXu", 0, 0, 0);
```

```
5         vTaskDelay(1000 / portTICK_PERIOD_MS);
6
7     }
```

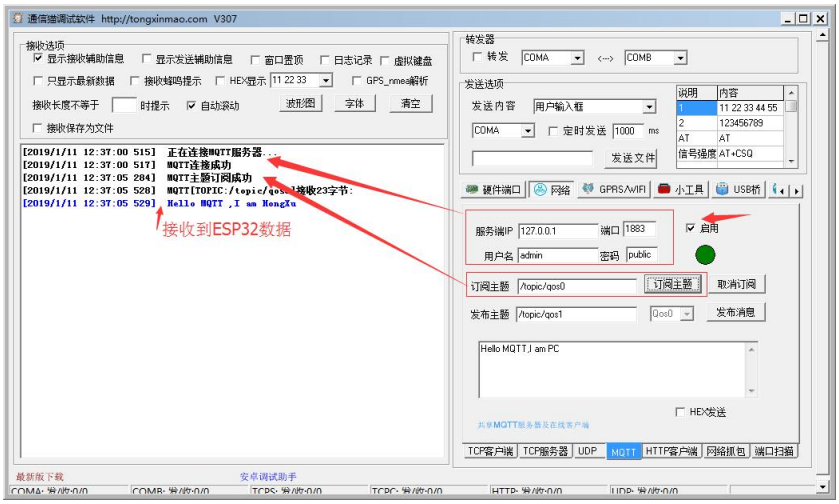
5. 测试效果

5.1. 测试流程

- 启动 MQTT 服务器，上文步骤。
- 下载 MQTT 源码到开发板
  - 修改 WiFi 账号和密码。
  - 修改 MQTT 配置信息

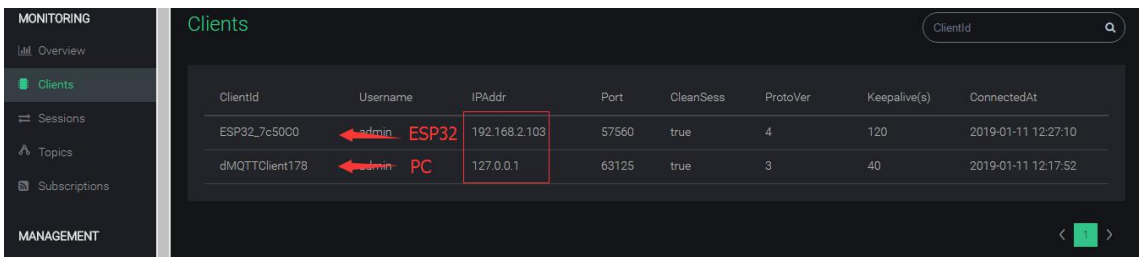
```
esp_mqtt_client_config_t mqtt_cfg = {
    .host = "192.168.2.104",              //MQTT服务器IP
    .event_handle = mqtt_event_handler, //MQTT事件
    .port=1883,                           //端口
    .username = "admin",                  //用户名
    .password = "public",                 //密码
    // .user_context = (void *)your_context
};
```

  - make menuconfig -> make all -> make flash ->make mmonitor
- PC MQTT 助手连上服务器。

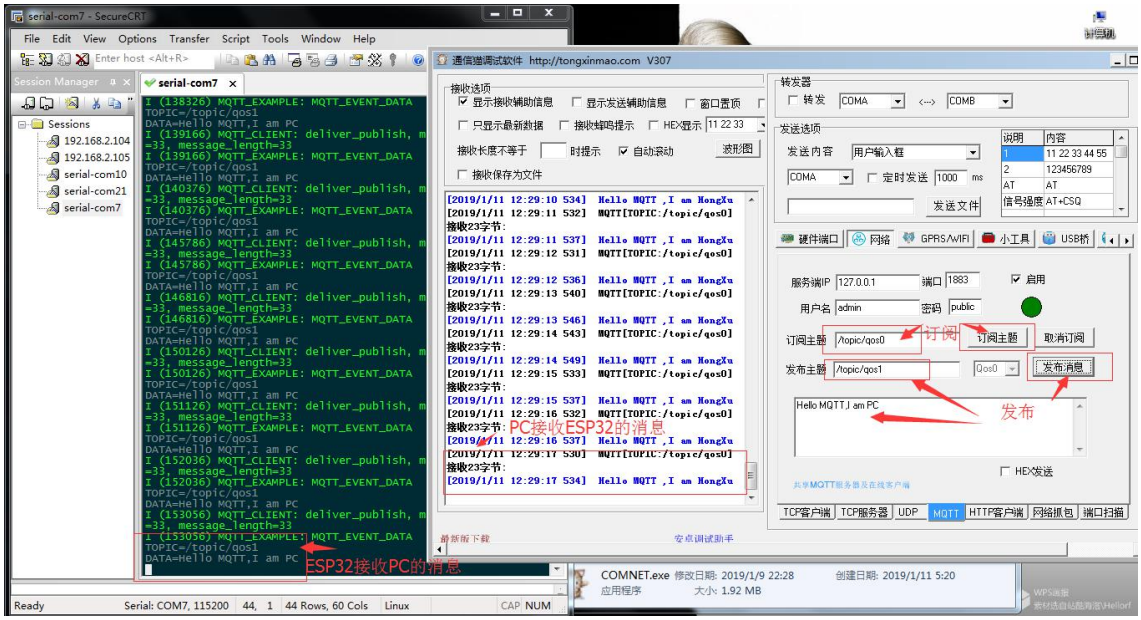


5.2. 效果展示

- 服务器有两个 MQTT 客户端，分别是 ESP32 开发板和 PC 的助手。



- 订阅和发布，可以互通



6. MQTT 总结

- 此处的 MQTT 源码是基于 TCP 的 MQTT 方案
- MQTT 的源码结构和 WiFi 状态机一样，都是初始化+回调的方式。
- MQTT 总结就是：伸手即得
  - 我发布的主题，无论是谁订阅都能收到，包括我自己。
  - 别人发布的主题，只要我订阅就能收到。
- 源码地址：https://github.com/HX-IoT/ESP32-Developer-Guide