



厦门大学  
XIAMEN UNIVERSITY

# 电子工艺实训

MQTT



## 一、MQTT简介

## 1、MQTT含义

MQTT 的全称是 Message Queuing Telemetry Transport，即消息队列遥感传输。

MQTT 在它的 3.1.1 规范中是这样定义的： MQTT 是用于物联网的 OASIS

（Organization for the Advancement of Structured Information Standards，  
结构化信息标准促进组织）标准消息传递协议，被设计为一种极其轻量级的发布/  
订阅消息传输模型，非常适合连接具有小代码足迹和最小网络带宽的远程设备。

## 2、MQTT主要特点

- (1) MQTT 工作在 TCP/IP 协议栈上。
- (2) 它是为硬件性能低下的远程设备，以及网络状况糟糕的场景而设计的。
- (3) MQTT 现在已经成了物联网的重要组成部分。

MQTT 是一种基于客户端-服务器的消息发布/订阅传输协议，重量轻、开放、简单并且易于实施。这些特性使得它非常适合在许多情况下使用，包括受限制的环境。例如，机器对机器（M2M）和物联网环境中的通信。在这些环境中往往只需要少量的代码，并且网络带宽非常宝贵。

## 3、OpenHarmony轻量系统中为什么选择MQTT

- (1) 开源。
- (2) 轻巧高效。 MQTT 的客户端非常小，占用的资源极少，并且耗电量很小。因此可以在小型微控制器上使用。 MQTT 的消息头也很小，从而可以优化网络带宽。
- (3) 双向通信。 MQTT 允许在“设备到云”和“云到设备”之间进行消息传递，很容易向一组设备广播消息。
- (4) 扩展性强。 MQTT 可以扩展以便连接数百万个物联网设备。

## 3、OpenHarmony轻量系统中为什么选择MQTT

(5) 消息传递可靠。消息传递的可靠性对于物联网项目来说是非常重要的。MQTT定义了 3 个服务质量级别来保障消息传递的可靠性。QoS0 表示最多一次，QoS1 表示最少一次，QoS2 表示恰好一次。

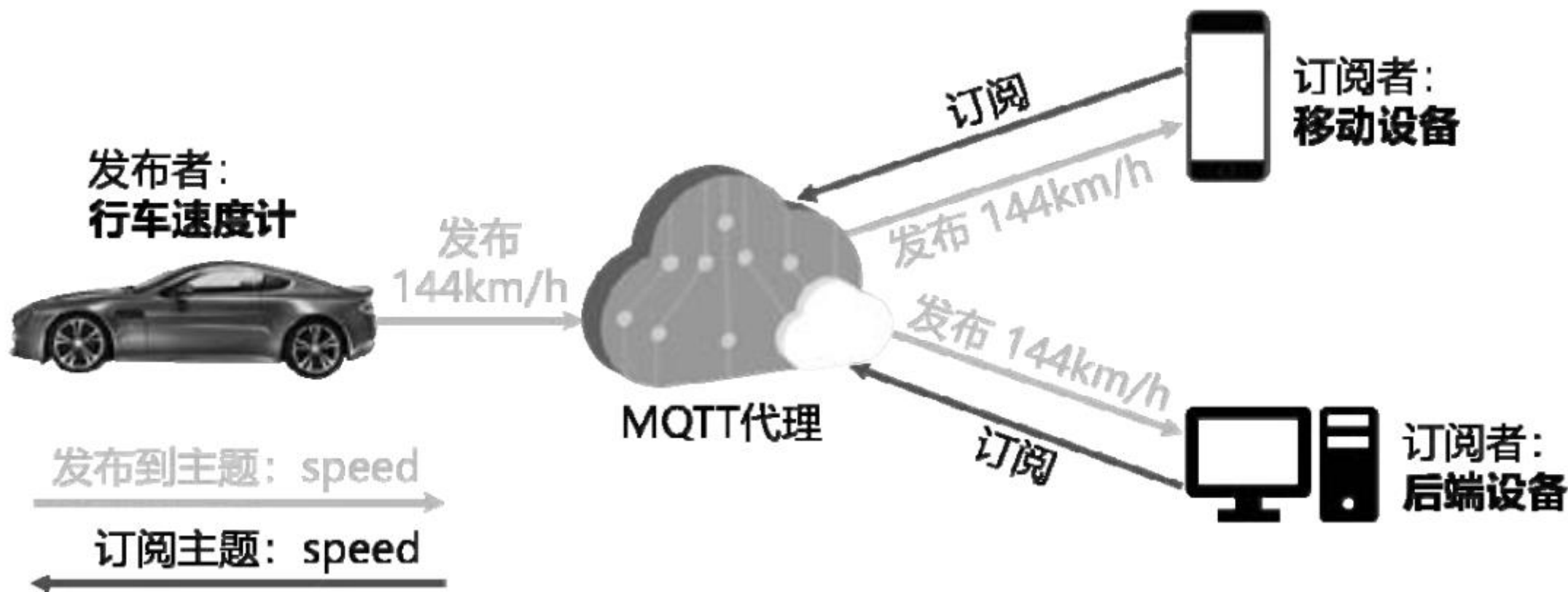
(6) 支持不可靠的网络。许多物联网设备是通过不可靠的移动网络进行连接的，MQTT 对持久会话的支持减少了客户端与代理重新连接的时间。

(7) 具有安全性。MQTT 使用 TLS (Transport Layer Security, 安全传输层协议) 加密消息，并且使用现代身份验证协议 (例如 OAuth)，这样既保证了消息传递的安全性，也很容易对客户端进行身份验证。



## 二、MQTT的发布/订阅模型

## 1、基于主题的过滤机制





## 2、MQTT 客户端

MQTT 客户端（ MQTT Client）指的是运行 MQTT 库，并且通过网络连接到 MQTT 代理的任何设备。

注意：发布者和订阅者都是 MQTT 客户端



## 2、MQTT 客户端

- MQTT 客户端可以是体积小、资源受限的设备，例如 Hi3861 开发板。
- MQTT 客户端也可以是体积更大、资源更丰富的 PC，或者服务器。

## 3、MQTT 代理

MQTT 代理（MQTT Broker）指的是实现发布/订阅功能的服务器。MQTT 代理是任何发布/订阅模式的核心，负责接收所有的消息、过滤消息、确定每条消息的订阅者，并且将消息发送给这些订阅的 MQTT 客户端。

MQTT 代理的规模可大可小，它可以是一台运行 MQTT 代理程序的 PC、服务器，也可以是一个服务器集群。



## 4、MQTT 连接

MQTT 基于 TCP/IP 协议栈，MQTT 客户端和 MQTT 代理都需要有一个 TCP/IP 协议栈。

请注意，MQTT 连接只在 MQTT 客户端和 MQTT 代理之间存在，**客户端之间从不直接连接。**



## 4、MQTT 连接

为了启动连接， MQTT客户端向 MQTT代理发送 CONNECT Packet （发起连接报文），而 MQTT 代理以 CONNACK Packet （连接回执报文）和状态代码进行响应。建立连接之后， MQTT 代理将保持连接状态，直到 MQTT 客户端发送断开连接命令或者连接中断了。

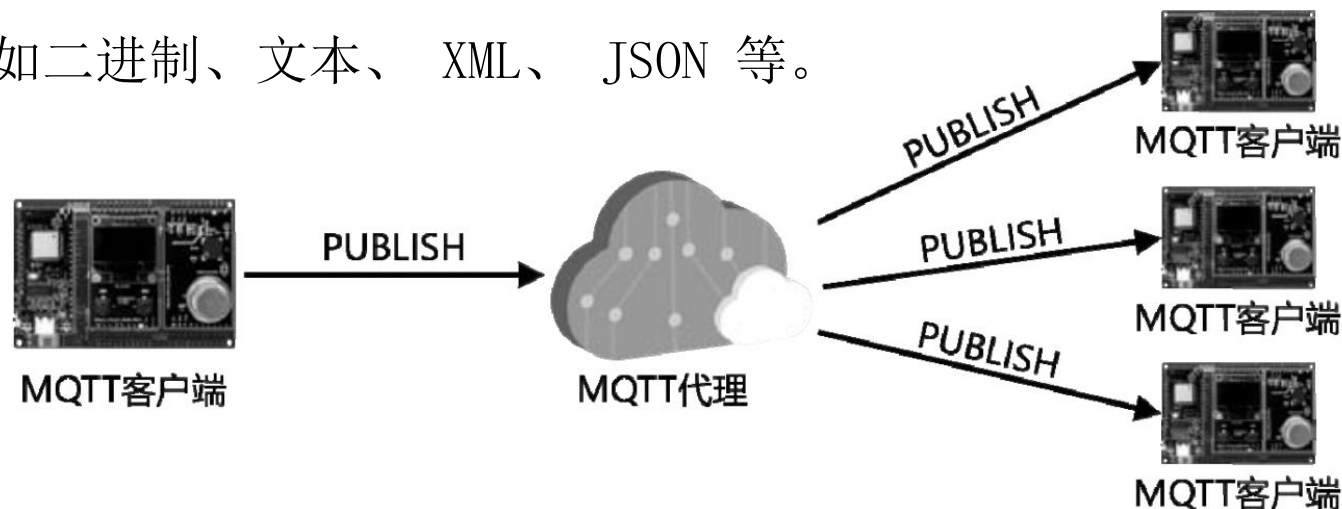
## 5、发布

为了发布消息，左侧的 MQTT 客户端需要向 MQTT 代理发送 PUBLISH Packet（发布消息报文）。

每条消息都必须包含一个主题， MQTT 代理会将消息转发给该主题的订阅者，也就是右侧的 MQTT 客户端。

通常每条消息都有一个有效负载（ payload），其中包含了以字节流方式传输的数据。

消息的发布者决定 payload 的数据结构，比如二进制、文本、 XML、 JSON 等。



## 6、订阅

为了订阅消息，左侧的 MQTT 客户端需要向 MQTT 代理发送 SUBSCRIBE Packet（订阅消息报文），而 MQTT 代理以 SUBACK Packet（订阅回执报文）进行响应。

每次订阅都必须包含一个订阅列表，列表中可以包括多个主题。此后，左侧的 MQTT 客户端将收到已订阅主题的消息，如图中③、④两步所示的那样。这些消息来自包含这些主题的消息的发布者。也就是说，它们发布的消息中包含了这些主题。



## 7、取消订阅

为了取消订阅消息，左侧的 MQTT 客户端需要向 MQTT 代理发送 UNSUBSCRIBE Packet（取消订阅消息报文），而 MQTT 代理以 UNSUBACK Packet（取消订阅回执报文）进行响应。

每一次取消订阅都必须包含一个取消订阅列表，列表中可以包括多个主题。此后，左侧的 MQTT 客户端将不再收到该主题的消息，如图中③、④两步所示的那样。





## 8、主题

在 MQTT 中，主题指的是 MQTT 代理用于为每个连接的 MQTT 客户端过滤消息的 UTF-8 字符串。

主题由一个或多个主题级别组成，每个主题级别由 “ / ” （主题级别分隔符）进行分隔。

请注意，每个主题都必须包含至少一个字符，主题字符串允许包含空格，主题区分大小写。另外，单独的 “ / ” 是一个有效的主题。

比如：myhome/groundfloor/livingroom/temperature

## 8、单级通配符 (+)

当订阅主题的时候， MQTT 客户端可以订阅一个明确的主题，也可以使用通配符同时订阅多个主题。请注意，通配符只能用于订阅主题，而不能用于发布消息。

单级通配符代表且仅代表一个主题级别

**myhome/groundfloor/+/temperature**

- ✓ myhome/groundfloor/livingroom/temperature
- ✓ myhome/groundfloor/kitchen/temperature
- × myhome/groundfloor/kitchen/brightness
- × myhome/firstfloor/kitchen/temperature
- × myhome/groundfloor/kitchen/fridge/temperature

## 9、多级通配符（+）

多级通配符必须作为主题中的最后一个字符放置，并且前面有一个“/”。

**myhome/groundfloor/#**

- ✓ myhome/groundfloor/livingroom/temperature
- ✓ myhome/groundfloor/kitchen/temperature
- ✓ myhome/groundfloor/kitchen/brightness
- × myhome/firstfloor/kitchen/temperature

当 MQTT 客户端使用多级通配符订阅主题时，它会接收到以通配符之前的模式开头的主题的所有消息，无论主题级别有多深。

如果只将多级通配符本身指定为主题， MQTT 客户端就将收到发送到 MQTT 代理的所有消息。



## 三、Paho-MQTT 嵌入式版本

## 1、Paho-MQTT 的含义

Paho-MQTT 是 Eclipse 基金会旗下的一个开源项目，主要实现了 MQTT 客户端功能。

Paho-MQTT 支持多种编程语言，包括 Java、Python、JavaScript、GoLang、C/C++、.Net (C#)、Embedded C/C++等。

本实训课程使用的是它的 Embedded C/C++版本

## 2、Paho-MQTT 的结构

Paho-MQTT 底层的库，最简单、最小，但实际上也是最难以使用的库。

### MQTTPacket 库

只负责处理 MQTT Packet 的序列化和反序列化

支持 MQTT3.1、MQTT3.1.1，但是目前不支持 MQTT5.0。

使用 C++ 语言实现，但是避开了动态内存分配和 STL（Standard Template Library，标准模板库）的使用。

### MQTTClient 库

对于依赖特定操作系统和网络的功能，MQTTClient 库提供了可替换的类。

基于MQTTPacket 库开发的，依赖 MQTTPacket 库，在 MQTTPacket 库的基础上提供了阻塞式 API 的支持。

### MQTTClient-C 库

C 语言版本的 MQTTClient 库，适用于 LiteOS、FreeRTOS 等小型嵌入式操作系统。

同样是基于 MQTTPacket 库开发的，依赖 MQTTPacket 库，也提供了阻塞式 API 的支持。

## 3、Paho-MQTT 源码总体结构

实训中主要用到的是MQTTClient-C 库

- ▼ paho\_mqtt
  - > .settings
  - > Debug
  - > doc
  - > MQTTClient
  - ▼ MQTTClient-C
    - > samples
    - > src
    - > test
  - ≡ BUILD.gn
  - M CMakeLists.txt
- ▼ MQTTPacket
  - > samples
  - > src
  - > test
- ≡ BUILD.gn
- M CMakeLists.txt

Paho-MQTT模块的根目录

MQTTClient-C库目录

MQTTClient-C库目录

示例程序源码目录

库源码目录

测试程序目录

编译脚本

MQTTPacket库目录

示例程序源码目录

库源码目录

测试程序目录

编译脚本

- ▼ MQTTClient-C
  - > samples
  - ▼ src
    - > cc3200
    - > FreeRTOS
    - > linux
    - ▼ ohos
      - C mqtt\_ohos\_cmsis.c
      - C mqtt\_ohos\_posix.c
      - C mqtt\_ohos.h
  - ≡ BUILD.gn
  - M CMakeLists.txt
  - C MQTTClient.c
  - C MQTTClient.h

库源码目录

FreeRTOS适配目录

linux适配目录

ohos适配目录 (移植到LiteOS)

cmsis适配源文件

posix适配源文件

ohos适配接口文件

编译脚本

MQTTClient-C库源文件

MQTTClient-C库接口文件

## 3、Paho-MQTT 源码总体结构

MQTTClient-C 库的示例程序源码

MQTTClient-C	
samples	示例程序源码目录
FreeRTOS	FreeRTOS示例程序目录
linux	linux示例程序目录
ohos	LiteOS示例程序目录
BUILD.gn	编译脚本
CMakeLists.txt	
hal_wifiot_at.c	与AT命令相关的API文件(来自OpenHarmony1.0版)
hal_wifiot_at.h	与AT命令相关的API文件(来自OpenHarmony1.0版)
hal_wifiot_errno.h	与AT命令相关的API文件(来自OpenHarmony1.0版)
mqtt_test_cmsis.c	cmsis版示例程序源文件(注册为AT命令)
mqtt_test_posix.c	posix版示例程序源文件(命令行程序)
mqtt_test.c	对MQTTClient-C库二次封装, 源文件
mqtt_test.h	对MQTTClient-C库二次封装, 接口文件
wifiot_at.c	与AT命令相关的API文件(来自OpenHarmony1.0版)
wifiot_at.h	与AT命令相关的API文件(来自OpenHarmony1.0版)





## 四、MQTT 代理 Mosquitto

## 1、Mosquitto简介

只有 MQTT 客户端还不够，我们还需要一个 MQTT 代理才能建立连接，并且进行测试。

Mosquitto 是 Eclipse 基金会旗下的一个开源的消息代理，实现了 MQTT 的 5.0、3.1.1 和 3.1 版本。

Mosquitto 很轻量，适用于从低功耗 IoT 设备到服务器的所有设备。



# 电子工艺实训 MQTT 代理 Mosquitto

## 2、Mosquitto windows版本安装

官网链接 <https://mosquitto.org/download/>

### Windows

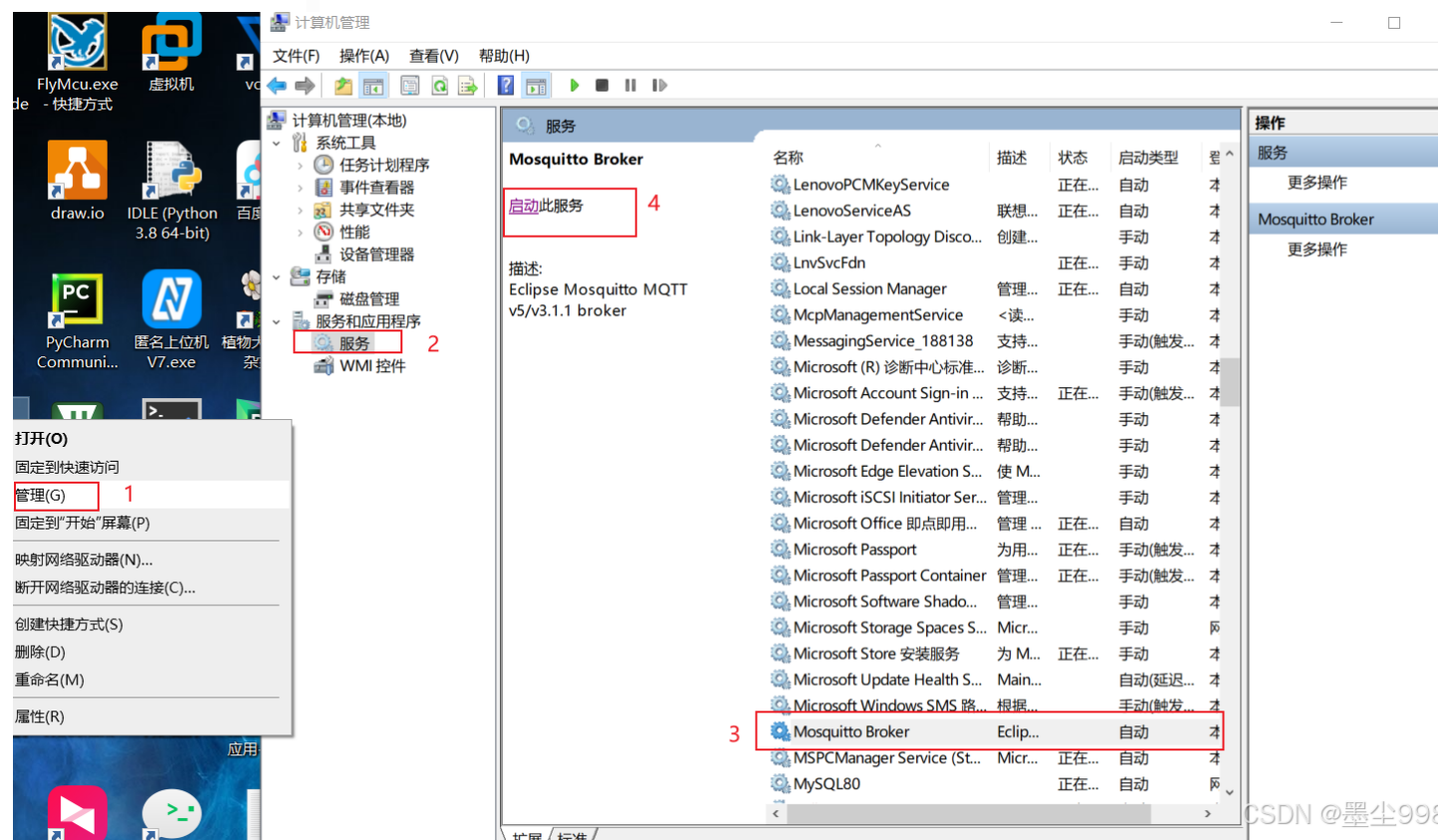
- mosquitto-2.0.21a-install-windows-x64.exe
- mosquitto-2.0.21a-install-windows-x86.exe

Older installers can be found at <https://mosquitto.org/files/binary/>.

See also README-windows.md after installing.

安装过程中，可以修改安装路径，其他默认即可。

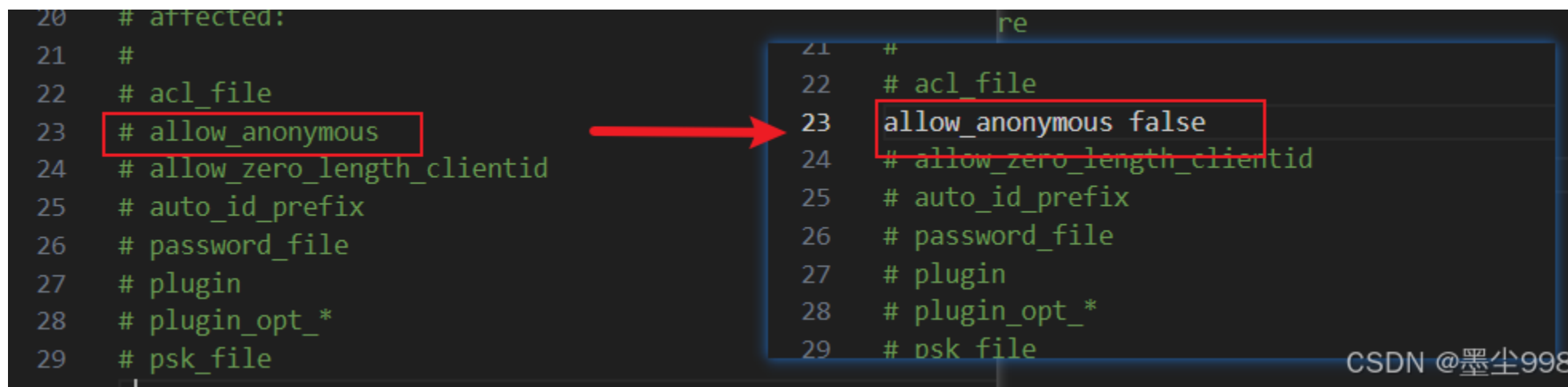
安装好后，我们右键此电脑找到他的服务并启动。



## 3、配置文件mosquitto.conf的修改

在安装路径的mosquitto目录下修改mosquitto.conf配置文件，修改三个地方

第一处修改成不允许匿名登陆，将注释取消，后面加上false



```
20 # affected:
21 #
22 # acl_file
23 # allow_anonymous
24 # allow_zero_length_clientid
25 # auto_id_prefix
26 # password_file
27 # plugin
28 # plugin_opt_*
29 # psk_file
30 # re

21 #
22 # acl_file
23 allow_anonymous false
24 # allow_zero_length_clientid
25 # auto_id_prefix
26 # password_file
27 # plugin
28 # plugin_opt_*
29 # psk_file
```

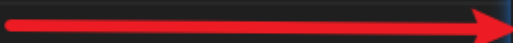
CSDN @墨尘998

## 3、配置文件mosquitto.conf的修改

在安装路径的mosquitto目录下修改mosquitto.conf配置文件，修改三个地方

第二处设置账户密码文件pfile.example位置，这里以我的安装路径展示

```
23 allow_anonymous false
24 # allow_zero_length_clientid
25 # auto_id_prefix
26 #password_file
27 # plugin
28 # plugin_opt_*
29 # psk_file
30 #
```



```
23 allow_anonymous false
24 # allow_zero_length_clientid
25 # auto_id_prefix
26 password_file G:\MQTT_server\mosquitto\pfile.example|
27 # plugin
28 # plugin_opt_*
29 # psk_file
30 #
```

## 3、配置文件mosquitto.conf的修改

在安装路径的mosquitto目录下修改mosquitto.conf配置文件，修改三个地方

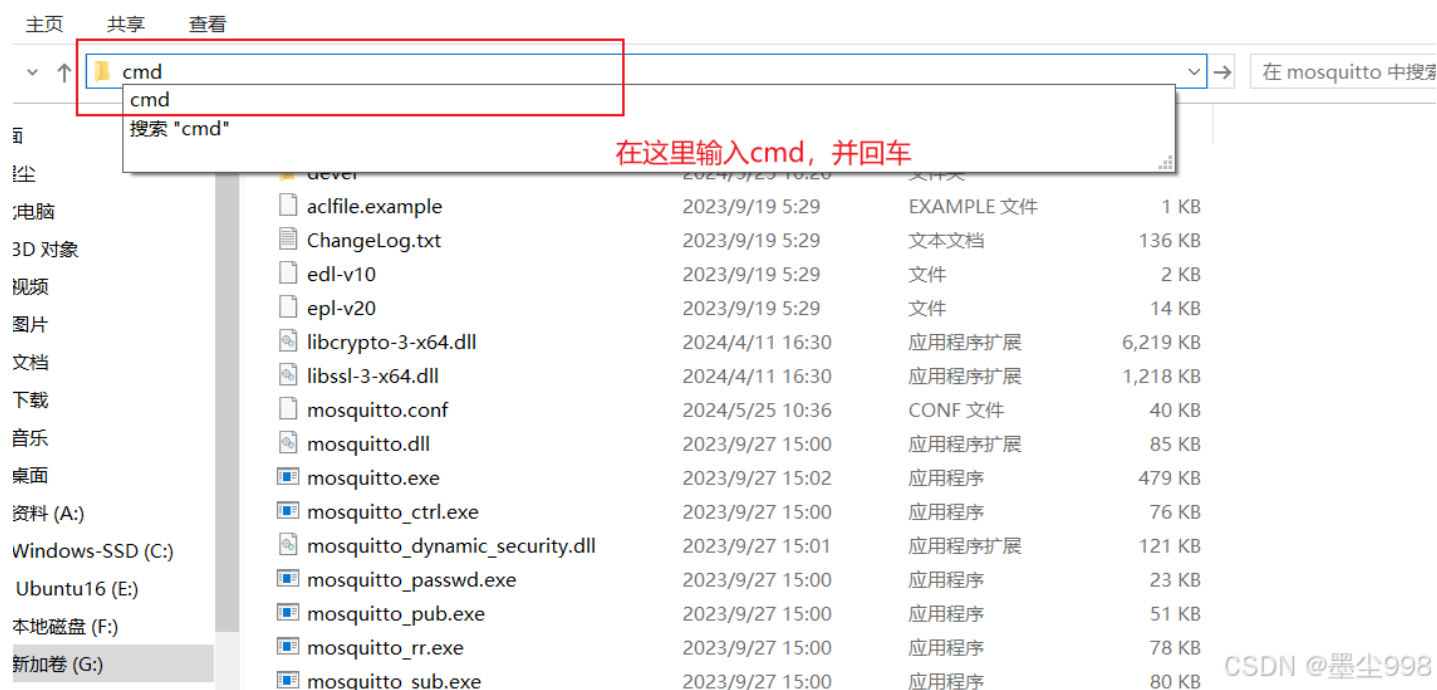
第三处设置监听端口1883

```
231 # listener 0 /tmp/mosquitto.sock
232 #
233 # listener port-number [ip address/host name/unix socket path]
234 #listener
235
236 # By default, a listener will attempt to listen on all supported IP protocol
237 # versions. If you do not have an IPv4 or IPv6 interface you may wish to
```

```
231 # listener 0 /tmp/mosquitto.sock
232 #
233 # listener port-number [ip address/host name/unix socket path]
234 listener 1883
235
236 # By default, a listener will attempt to listen on all supported IP protocol
237 # versions. If you do not have an IPv4 or IPv6 interface you may wish to
238 # disable support for either of these protocol versions. In particular, not
```

## 4、设置Mosquitto用户账号

在安装路径的mosquitto目录下进入cmd命令行



## 4、设置Mosquitto用户账号

```
mosquitto_passwd -c /MosquittoTest/pwfile.example FirstUserName
```

（使用-c 参数会导致清空密码文件，重新插入用户）

FirstUserName（账号名）

C:\Windows\System32\cmd.exe

Microsoft Windows [版本 10.0.19045.4291]  
(c) Microsoft Corporation。保留所有权利。

```
C:\MQTT_server\mosquitto>mosquitto_passwd -c /MQTT_server/mosquitto/pwfile.example admin_
```

该文件的路径

自己起的用户名  
(后面使用mqtt上位机会用到)



## 4、设置Mosquitto用户账号

`mosquitto_passwd /MosquittoTest/pwfile.example SecondUserName`

（不使用-c 表示追加用户，不影响旧用户）

SecondUserName（账号名）

```
C:\Windows\System32\cmd.exe
Microsoft Windows [版本 10.0.19045.4291]
(c) Microsoft Corporation。保留所有权利。

G:\MQTT_server\mosquitto>mosquitto_passwd -c /MQTT_server/mosquitto/pwfile.example admin
Password:
Reenter password:
G:\MQTT_server\mosquitto>_
```

回车后会设置密码，再次回车输入密码  
注：这里密码是我们设置，并且密码的输入不会显示

CSDN @墨尘998



# 电子工艺实训 MQTT 代理 Mosquitto

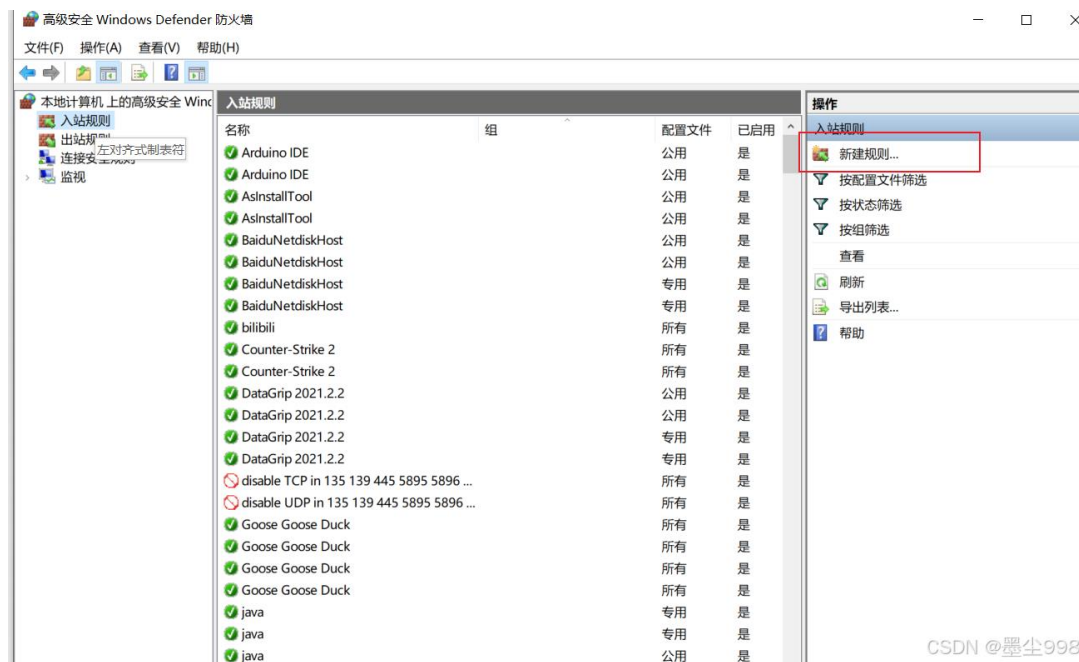
## 5、防火墙设置

打开防火墙的高级设置，添加新的入站规则



## 5、防火墙设置

打开防火墙的高级设置，添加新的入站规则



CSDN @墨尘998



CSDN @墨尘998

## 5、防火墙设置

打开防火墙的高级设置，添加新的入站规则





# 电子工艺实训 MQTT 代理 Mosquitto

## 5、防火墙设置

打开防火墙的高级设置，添加新的入站规则

新建入站规则向导

配置文件

此规则应用的配置文件

何时应用该规则?

规则类型

协议和端口

操作

配置文件

名称

何时应用该规则?

☒ 域(D)

计算机连接到其企业域时应用。

☒ 专用(P)

计算机连接到专用网络位置(例如, 家或工作单位)时应用。

☒ 公用(U)

计算机连接到公用网络位置时应用。

1这里默认即可

2

< 上一步(B)

下一页(N) >

取消

CSDN @ 墨生998

指定此规则的名称和描述。

步骤:

● 规则类型

● 协议和端口

● 操作

● 配置文件

● 名称

1名称设置为MQTT

名称(N):

MQTT

描述(可选)(D):

2

< 上一步(B)

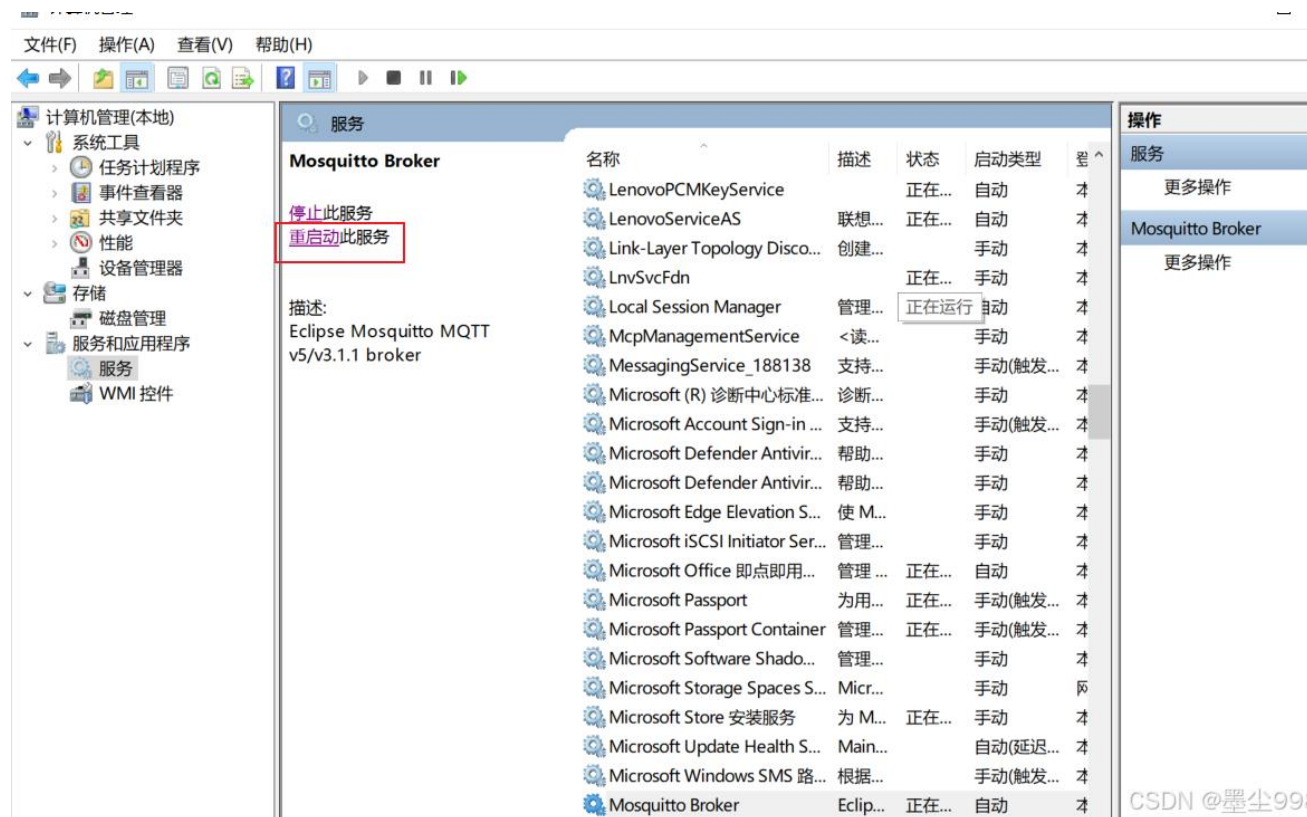
完成(F)

取消

CSDN @ 墨生998

## 5、防火墙设置

重启Mosquitto Broker服务即可使用





## 五、 Paho-MQTT windows版本客户端



## 1、Paho-MQTT windows版本安装

电脑版的mqtt客户端下载链接:

<https://repo.eclipse.org/content/repositories/paho-releases/org/eclipse/paho/org.eclipse.paho.ui.app/1.1.1/>

[org.eclipse.paho.ui.app-1.1.1-win32.win32.x86.zip](#)  
[org.eclipse.paho.ui.app-1.1.1-win32.win32.x86.zip.md5](#)  
[org.eclipse.paho.ui.app-1.1.1-win32.win32.x86.zip.sha1](#)  
[org.eclipse.paho.ui.app-1.1.1-win32.win32.x86\\_64.zip](#)  
[org.eclipse.paho.ui.app-1.1.1-win32.win32.x86\\_64.zip.md5](#)  
[org.eclipse.paho.ui.app-1.1.1-win32.win32.x86\\_64.zip.sha1](#)

MoI

MoI

MoI

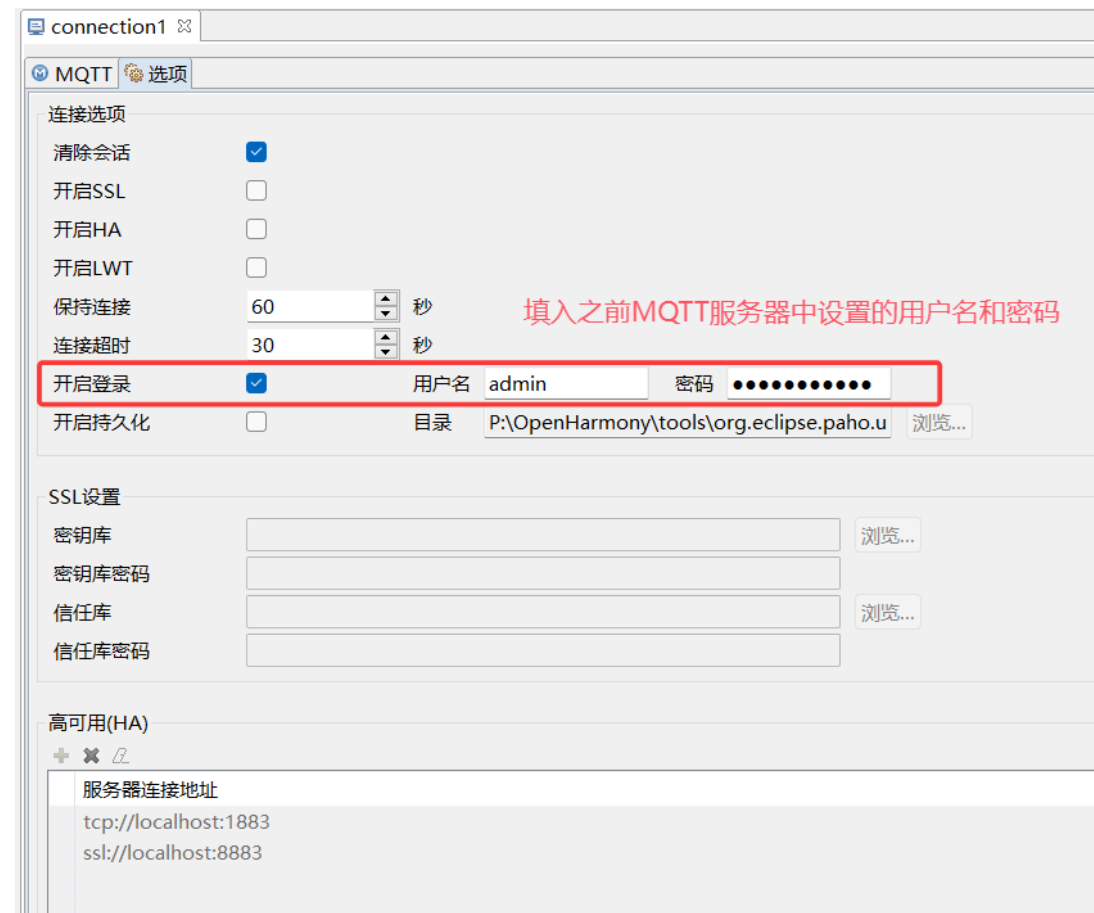
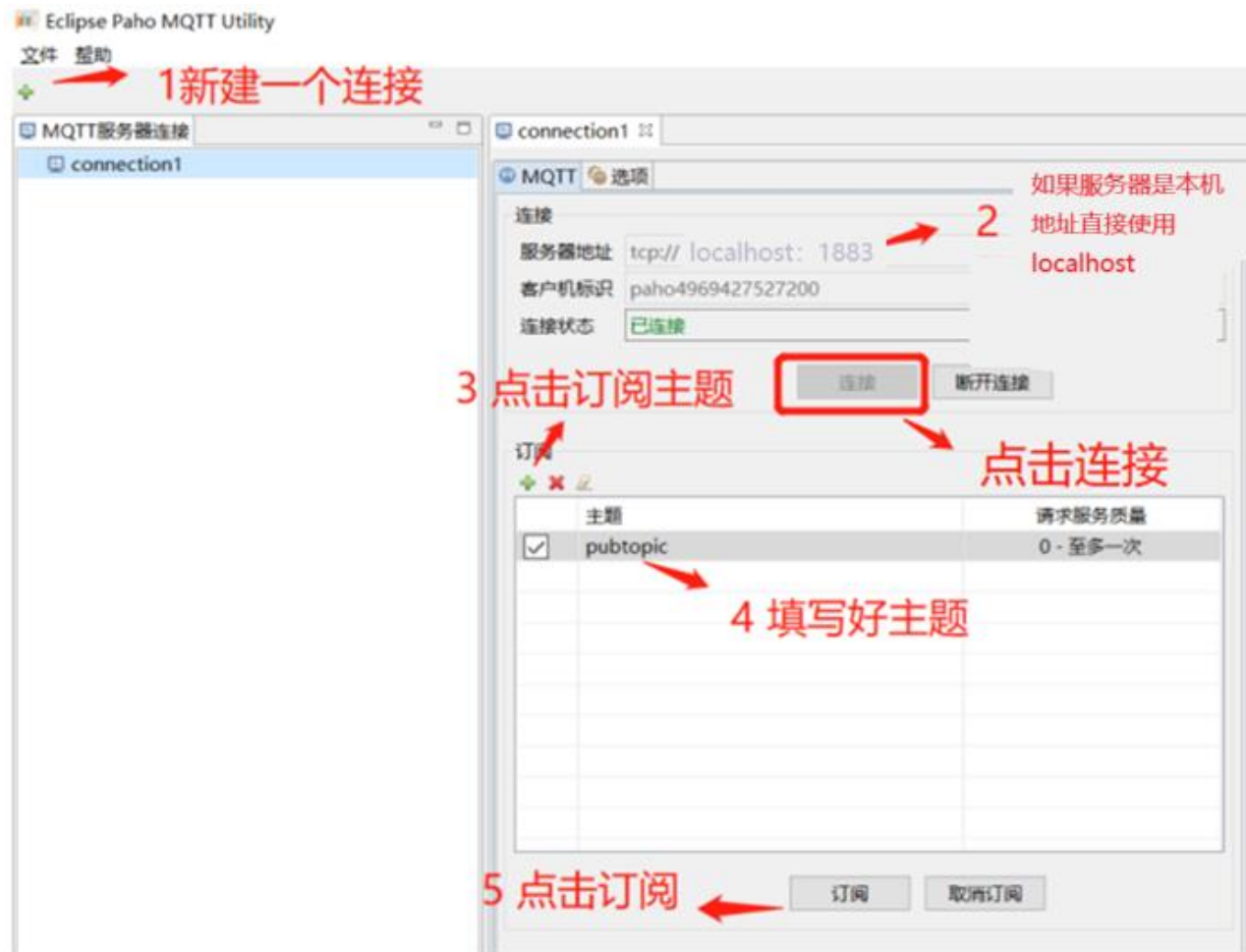
MoI

MoI

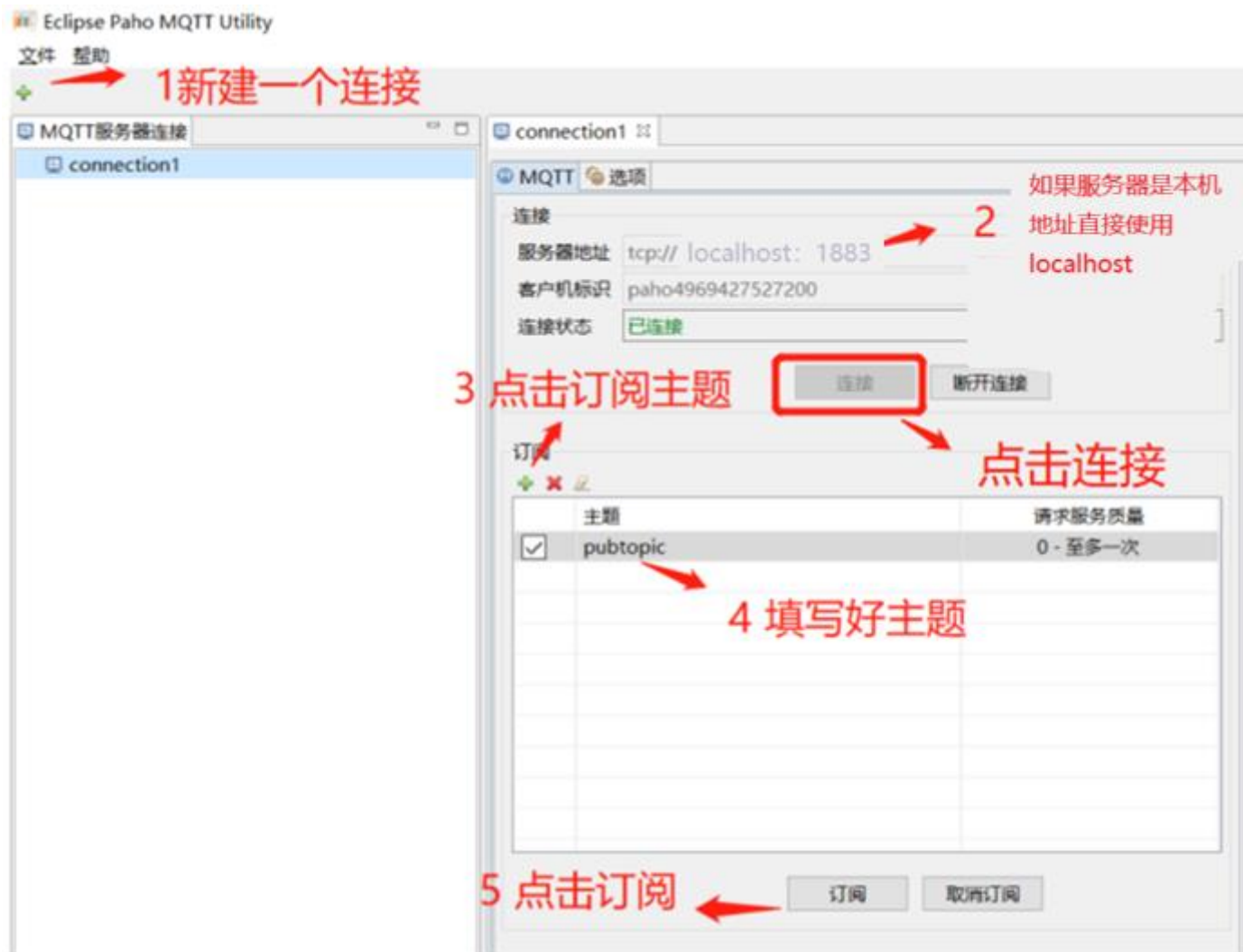
MoI



## 2、连接服务器地址并订阅主题



## 2、连接服务器地址并订阅主题





## 六、 MQTT测试样例

## 1、Paho-MQTT 嵌入式版本的整合

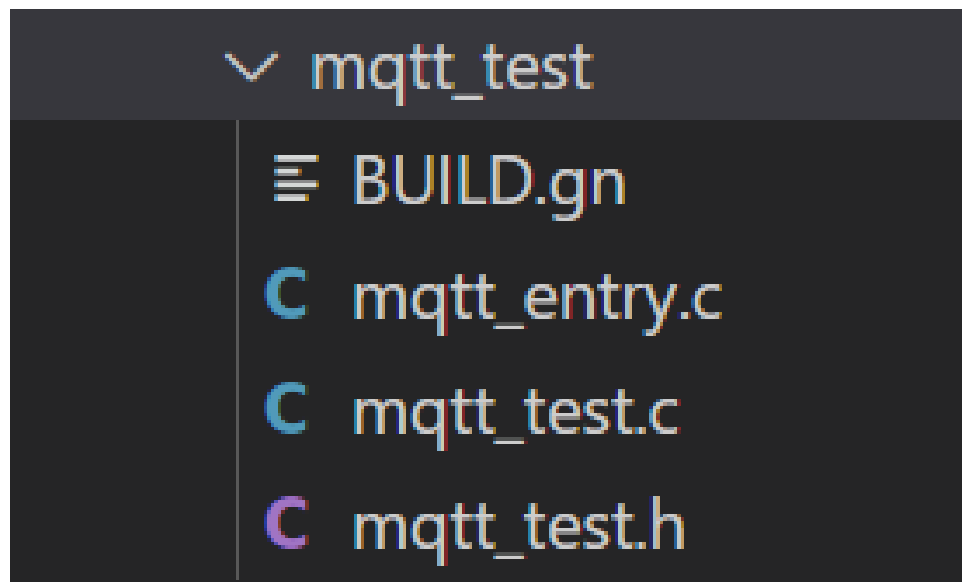
Paho-MQTT 的Embedded C/C++版本库已经整合放入

3.01ts/third\_party/中，为pahomqtt目录

具体整合配置可以查看3.01ts/third\_party/pahomqtt/BUILD.gn文件

## 2、新建mqtt\_test目录

\3.0-lts\src\applications\sample\wifi-iot\app\下新建mqtt\_test目录，包含文件如下：



## 3、mqtt\_test\BUILD.gn文件

```
static_library("mqtt_test") {  
    sources = [  
        "mqtt_test.c",  
        "mqtt_entry.c"  
    ]  
  
    include_dirs = [  
        "//utils/native/lite/include",  
        "//kernel/liteos_m/components/cmsis/2.0",  
        "//base/iot_hardware/interfaces/kits/wifi_iot_lite",  
        "//vendor/hisi/hi3861/hi3861/third_party/lwip_sack/include",  
        "//foundation/communication/interfaces/kits/wifi_lite/wifiservice",  
        "//third_party/pahomqtt/MQTTPacket/src",  
        "//third_party/pahomqtt/MQTTClient-C/src",  
        "//third_party/pahomqtt/MQTTClient-C/src/liteOS",  
    ]  
    #表示需要a_myparty 软件包  
    deps = [  
        "//third_party/pahomqtt:pahomqtt_static",  
    ]  
}
```

## 3、mqtt\_test\mqtt\_entry.c文件

WiFi STA (Station) 模式连接，主要包括WiFi初始化、扫描、连接以及网络接口处理

```
int hi_wifi_start_connect(void)
{
    int ret;
    errno_t rc;
    hi_wifi_assoc_request assoc_req = {0};

    /* copy SSID to assoc_req */
    rc = memcpy_s(assoc_req.ssid, HI_WIFI_MAX_SSID_LEN + 1, "RedmiK40", 8); /* 9:ssid length */
    if (rc != EOK) {
        return -1;
    }

    //热点加密方式
    assoc_req.auth = HI_WIFI_SECURITY_WPA2PSK;

    /* 热点密码 */
    memcpy(assoc_req.key, "07686582488", 11);

    ret = hi_wifi_sta_connect(&assoc_req);
    if (ret != HISI_OK) {
        return -1;
    }

    return 0;
}
```

设置wifi账号与密码，  
需要能连上MQTT的  
服务器

## 3、mqtt\_test\mqtt\_test.c文件

此处修改为自己架设的  
MQTT服务器ip地址和  
之前设置的MQTT服务器  
用户名、密码

```
int mqtt_connect(void)
{
    int rc = 0;

    NetworkInit(&n);
    NetworkConnect(&n, "5.196.95.208", 1883);

    buf_size = 4096+1024;
    onenet_mqtt_buf = (unsigned char *) malloc(buf_size);
    onenet_mqtt_readbuf = (unsigned char *) malloc(buf_size);
    if (!(onenet_mqtt_buf && onenet_mqtt_readbuf))
    {
        printf("No memory for MQTT client buffer!");
        return -2;
    }

    MQTTClientInit(&mq_client, &n, 1000, onenet_mqtt_buf, buf_size, onenet_mqtt_readbuf, buf_size);

    MQTTStartTask(&mq_client);

    data.keepAliveInterval = 30;
    data.cleansession = 1;
    data.clientID.cstring = "ohos_hi3861";
    data.username.cstring = "123456";
    data.password.cstring = "222222";
    data.cleansession = 1;
```

MQTT服务器ip地址

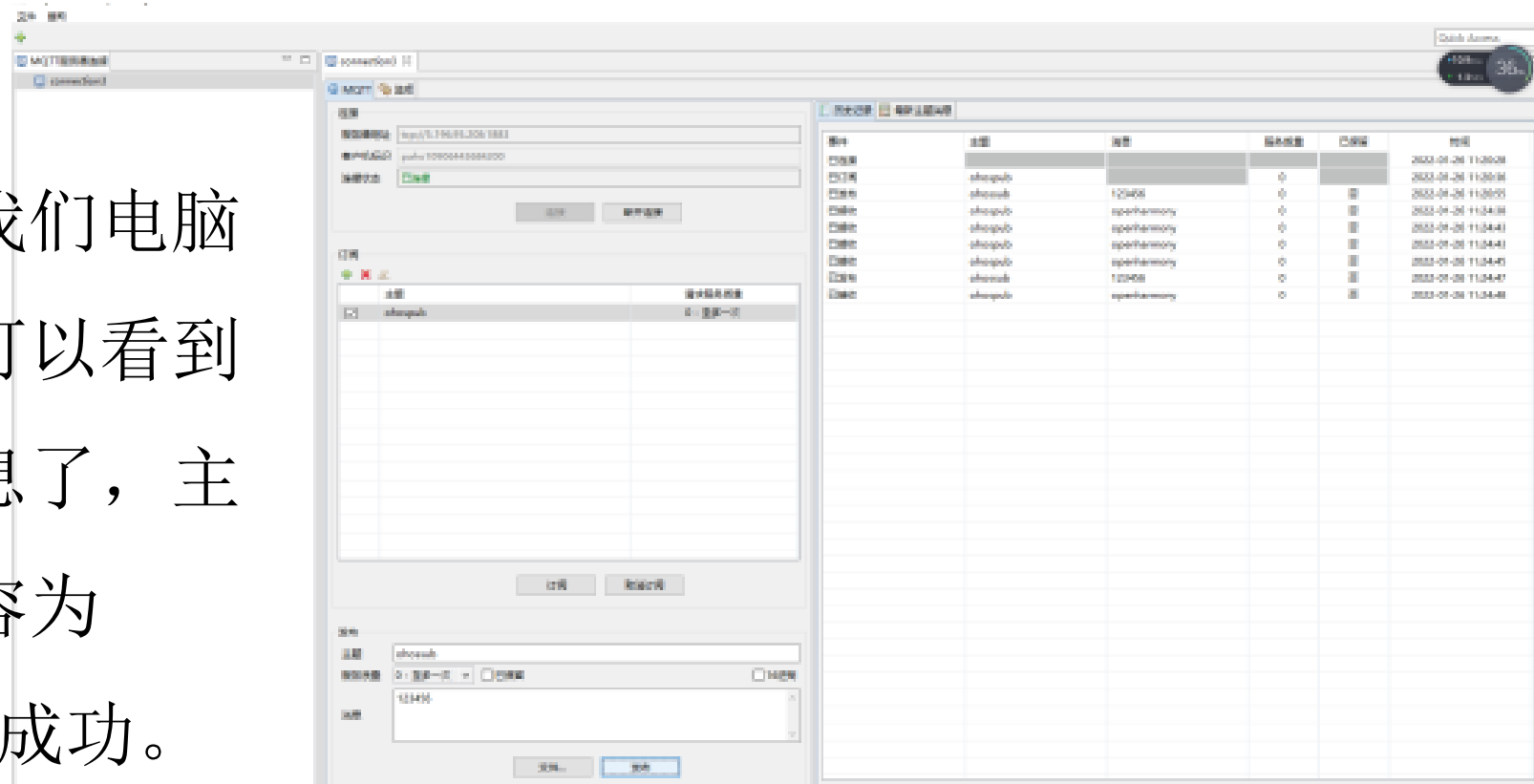
之前设置的MQTT服  
务器用户名和密码



## 4、app\BUILD.gn文件

```
import("//build/lite/config/component/lite_component.gni")

lite_component("app") {
    features = [
        "mqtt_test:mqtt_test",
    ]
}
```



## 5、实验现象

电脑发送主题为ohossub，内容为123456，查看串口监视，可以看到也收到了数据

```
SSID: TP-LINK_06F2
SSID: CMCC-NxSZ
+NOTICE:CONNECTED
WiFi: Connected
topic ohossub receive a message
message is 123456
```

## 六、 MQTT实验:

开发板的MQTT软件，向服务器发送MQTT信息，主题为 ID，消息内容为自己的学号。

电脑的MQTT客户端软件，向服务器发送MQTT信息，主题为 name，消息内容为自己的姓名拼音。