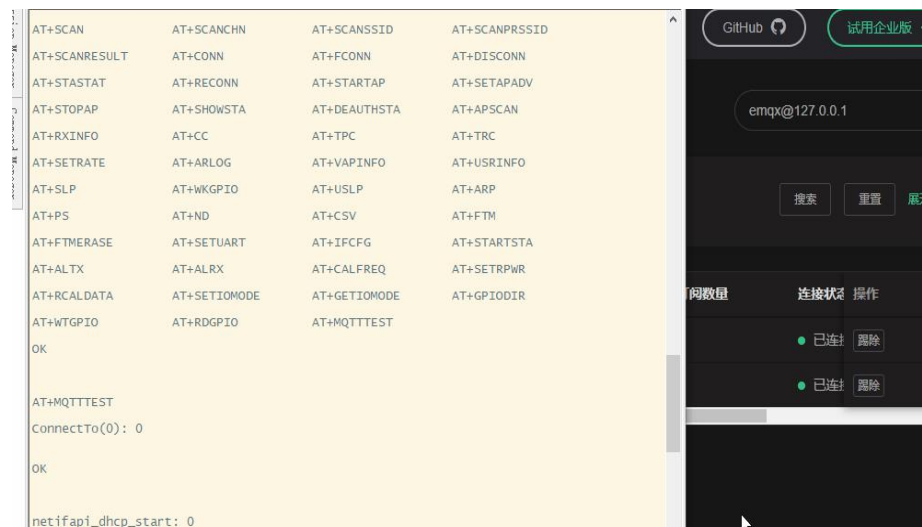


Harmony OS 智能硬件 入门系列课程 <快速上手>

快速掌握Hi3861开发板基础开发技巧

第 8 讲：Hi3861上物联网应用开发



本讲内容

- 第1节: MQTT协议
- 第2节: paho mqtt移植
- 第3节: 搭建MQTT测试环境
- 第4节: 测试MQTT

本讲目标

- 1、了解MQTT协议
- 2、掌握paho mqtt移植方法及步骤
- 3、会写MQTT测试程序

第1节：MQTT协议

- 知识点1：简述
- 知识点2：设计规范
- 知识点3：主要特性
- 知识点4：MQTT协议原理
- 知识点5：MQTT协议数据包结构

知识点1 【简述】

- MQTT (Message Queuing Telemetry Transport, 消息队列遥测传输协议), 是一种基于发布/订阅 (publish/subscribe) 模式的"轻量级"通讯协议, 该协议构建于TCP/IP协议上, 由IBM在1999年发布。MQTT最大优点在于, 可以以极少的代码和有限的带宽, 为连接远程设备提供实时可靠的消息服务。作为一种低开销、低带宽占用的即时通讯协议, 使其在物联网、小型设备、移动应用等方面有较广泛的应用。
- 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。

知识点2 【设计规范】

- 由于物联网的环境是非常特别的，所以MQTT遵循以下设计原则：
 1. 精简，不添加可有可无的功能；
 2. 发布/订阅（Pub/Sub）模式，方便消息在传感器之间传递；
 3. 允许用户动态创建主题，零运维成本；
 4. 把传输量降到最低以提高传输效率；
 5. 把低带宽、高延迟、不稳定的网络等因素考虑在内；
 6. 支持连续的会话控制；
 7. 理解客户端计算能力可能很低；
 8. 提供服务质量管理；
 9. 假设数据不可知，不强求传输数据的类型与格式，保持灵活性。

知识点3 【主要特性】

- MQTT协议工作在低带宽、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"、"至少一次"、"只有一次"
 - 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。
 - 使用Last Will和Testament特性通知有关各方客户端异常中断的机制。
 - ✓ Last Will：即遗言机制，用于通知同一主题下的其他设备发送遗言的设备已经断开了连接。
 - ✓ Testament：遗嘱机制，功能类似于Last Will。

知识点4 【MQTT协议原理】

■ MQTT协议实现方式

- 实现MQTT协议需要客户端和服务端通讯完成，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



■ 网络传输与应用消息

- MQTT会构建底层网络传输：它将建立客户端到服务器的连接，提供两者之间的一个有序的、无损的、基于字节流的双向传输。
- 当应用数据通过MQTT网络发送时，MQTT会把与之相关的服务质量（QoS）和主题名（Topic）相关连。

知识点5 【MQTT协议数据包结构】

- 在MQTT协议中，一个MQTT数据包由：固定头（Fixed header）、可变头（Variable header）、消息体（payload）三部分构成。MQTT数据包结构如下：
 - 固定头（Fixed header）。存在于所有MQTT数据包中，表示数据包类型及数据包的分组类标识。
 - 可变头（Variable header）。存在于部分MQTT数据包中，数据包类型决定了可变头是否存在及其具体内容。
 - 消息体（Payload）。存在于部分MQTT数据包中，表示客户端收到的具体内容。

本节小结

本讲所学知识点有：

- 知识点1：简述
- 知识点2：设计规范
- 知识点3：主要特性
- 知识点4：MQTT协议原理
- 知识点5：MQTT协议数据包结构






第2节：paho mqtt移植

- 知识点1：下载paho mqtt软件包，解压添加到鸿蒙源码中
- 知识点2：在pahomqtt 文件夹下面新建BUILD.gn文件
- 知识点3：配置paho mqtt到sdk中
- 知识点4：移植，修改编译报错

知识点1 【下载paho mqtt软件包，解压添加到鸿蒙源码中】

■ 下载paho mqtt软件包：

➤ 下载地址：<https://github.com/eclipse/paho.mqtt.embedded-c>

 hihopeorg-HarmonyOS-IoT-Application-Development-master (1).zip	2020/11/23 10:42	360压缩 ZIP 文件	1,113 KB
 hihopeorg-HarmonyOS-IoT-Application-Development-master.zip	2020/10/31 8:15	360压缩 ZIP 文件	1,085 KB
 paho.mqtt.embedded-c-master.zip	2020/11/6 11:13	360压缩 ZIP 文件	242 KB
 wifiiot-1.0.tar.gz	2020/10/19 12:16	360压缩	6,714 KB
 相关源码.zip	2020/11/20 9:37	360压缩 ZIP 文件	1,071 KB

➤ 解压后，将所有文件都拷贝到 pahomqtt 文件夹下

sharefolder (\\192.168.17.139) (Z:) > code-1.0 > third_party > pahomqtt >

名称	修改日期	类型	大小
Debug	2020/11/24 10:24	文件夹	
doc	2020/11/24 10:24	文件夹	
MQTTClient	2020/11/24 10:24	文件夹	
MQTTClient-C	2020/11/24 10:24	文件夹	
MQTTPacket	2020/11/24 10:24	文件夹	
test	2020/11/24 10:24	文件夹	
about.html	2018/3/5 23:56	360 Chrome HT...	2 KB
CMakeLists.txt	2018/3/5 23:56	TXT 文件	2 KB
CONTRIBUTING.md	2018/3/5 23:56	MD 文件	4 KB
edl-v10	2018/3/5 23:56	文件	2 KB
epl-v10	2018/3/5 23:56	文件	11 KB
library.properties	2018/3/5 23:56	PROPERTIES 文件	1 KB
Makefile	2018/3/5 23:56	文件	6 KB
notice.html	2018/3/5 23:56	360 Chrome HT...	10 KB
README.md	2018/3/5 23:56	MD 文件	4 KB
travis-build.sh	2018/3/5 23:56	SH 文件	1 KB
travis-env-vars	2018/3/5 23:56	文件	1 KB
travis-install.sh	2018/3/5 23:56	SH 文件	1 KB

知识点2 【在pahomqtt 文件夹下面新建BUILD.gn文件】

/third_party/pahomqtt/BUILD.gn

<code/>

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

config("pahomqtt_config") {
  include_dirs = [
    "MQTTPacket/src",
    "MQTTPacket/samples",
    "//vendor/hisi/hi3861/hi3861/third_party/lwip_sack/include",
    "//kernel/liteos_m/components/cmsis/2.0",
  ]
}
```

知识点2 【在pahomqtt 文件夹下面新建BUILD.gn文件】

/third_party/pahomqtt/BUILD.gn

<code/>

```
pahomqtt_sources = [  
    "MQTTPacket/samples/transport.c",  
    "MQTTPacket/src/MQTTConnectClient.c",  
    "MQTTPacket/src/MQTTConnectServer.c",  
    "MQTTPacket/src/MQTTDSerializePublish.c",  
    "MQTTPacket/src/MQTTFormat.c",  
    "MQTTPacket/src/MQTTPacket.c",  
    "MQTTPacket/src/MQTTSerializePublish.c",  
    "MQTTPacket/src/MQTTSubscribeClient.c",  
    "MQTTPacket/src/MQTTSubscribeServer.c",  
    "MQTTPacket/src/MQTTUnsubscribeClient.c",  
    "MQTTPacket/src/MQTTUnsubscribeServer.c",  
]
```

知识点2 【在pahomqtt 文件夹下面新建BUILD.gn文件】

/third_party/pahomqtt/BUILD.gn

<code/>

```
lite_library("pahomqtt_static") {  
    target_type = "static_library"  
    sources = pahomqtt_sources  
    public_configs = [ ":pahomqtt_config" ]  
}
```

```
lite_library("pahomqtt_shared") {  
    target_type = "shared_library"  
    sources = pahomqtt_sources  
    public_configs = [ ":pahomqtt_config" ]  
}
```

知识点2 【在pahomqtt 文件夹下面新建BUILD.gn文件】

/third_party/pahomqtt/BUILD.gn

<code/>

```
ndk_lib("pahomqtt_ndk") {  
    if (board_name != "hi3861v100") {  
        lib_extension = ".so"  
        deps = [  
            ":pahomqtt_shared"  
        ]  
    } else {  
        deps = [  
            ":pahomqtt_static"  
        ]  
    }  
    head_files = [  
        "../third_party/pahomqtt"  
    ]  
}
```


知识点3 【配置paho mqtt到sdk中】

- 打开vendor\hisi\hi3861\hi3861\BUILD.gn 文件，在lite_component("sdk") 中增加
"//third_party/pahomqtt:pahomqtt_static",

```
18 lite_component("sdk") {  
19     features = [ ]  
20  
21     deps = [ "//kernel/liteos_m/components/cmsis",  
22             "//kernel/liteos_m/components/kal",  
23             "//third_party/cJSON:cjson_static",  
24             "//third_party/pahomqtt:pahomqtt_static"  
25     ]  
26 }  
27
```

知识点4 【移植，修改编译报错】

实现步骤：

1. 导入socket相关头文件
2. 修改 transport_sendPacketBuffer 函数
3. 修改 transport_close 函数
4. 未使用的函数参数在函数内添加一条使用代码，如buflen = buflen;

知识点4【移植，修改编译报错】

步骤1：导入socket相关头文件，

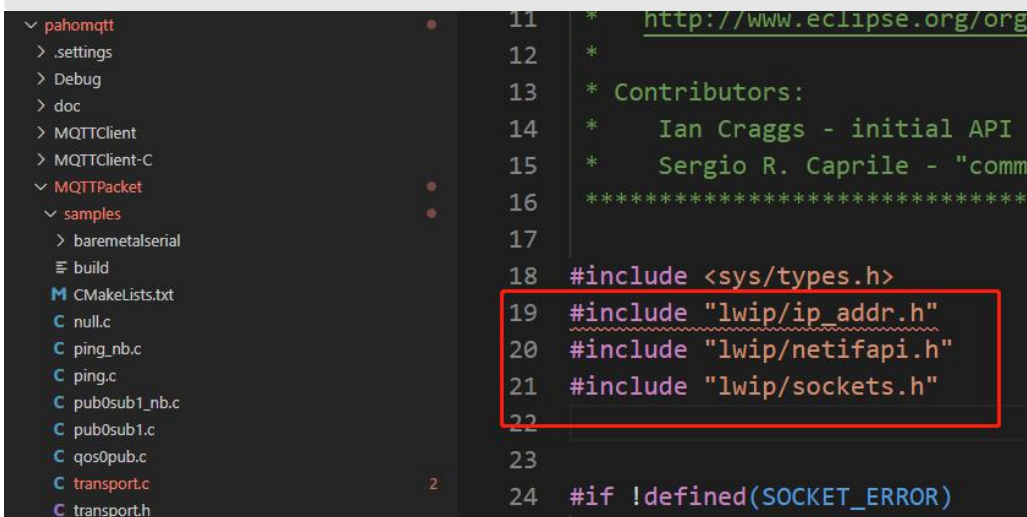
<code/>

```
//third_party\pahomqtt\MQTTPacket\samples\transport.c
```

```
#include "lwip/ip_addr.h"
```

```
#include "lwip/netifapi.h"
```

```
#include "lwip/sockets.h"
```



知识点4 【移植，修改编译报错】

步骤2：修改 transport_sendPacketBuffer 函数

<code/>

```
//third_party\pahomqtt\MQTTPacket\samples\transport.c
int transport_sendPacketBuffer(int sock, unsigned char* buf, int buflen)
{
    int rc = 0;
    // rc = write(sock, buf, buflen); //修改前代码
    rc = send(sock, buf, buflen, 0);
    return rc;
}
```

知识点4 【移植，修改编译报错】

步骤3：修改 transport_close 函数

<code/>

```
//third_party\pahomqtt\MQTTPacket\samples\transport.c
int transport_close(int sock)
{
    int rc;
    rc = shutdown(sock, SHUT_WR);
    rc = recv(sock, NULL, (size_t)0, 0);
    // rc = close(sock); //会报close函数不存在
    rc = lwip_close(sock);
    return rc;
}
```

知识点4 【移植，修改编译报错】

步骤4：未使用的函数参数在函数内添加一条使用代码，如buflen = buflen;

```
../third_party/pahomqtt/MQTTPacket/src/MQTTConnectClient.c:132:111: error: un  
used parameter 'buflen' [-Werror=unused-parameter]  
int MQTTDeserialize_connack(unsigned char* sessionPresent, unsigned char* connack_rc, unsigned char* buf, int buflen)  
^~~~~~
```

```
132 int MQTTDeserialize_connack(unsigned char* sessionPresent, unsigned char* connack_rc, unsigned char* buf, int buflen)  
133 {  
134     buflen = buflen;  
135     MQTTHeader header = {0};
```

```
170 int MQTTSerialize_zero(unsigned char* buf, int buflen, unsigned char packettype)  
171 {  
172     buflen = buflen;
```

知识点4【移植，修改编译报错】

步骤4：未使用的函数参数在函数内添加一条使用代码，如buflen = buflen;

```
../third_party/pahomqtt/MQTTPacket/src/MQTTSubscribeClient.c:100:123: error: unused parameter 'buflen' [-Werror=unused-parameter]
int MQTTDeserialize_suback(unsigned short* packetid, int maxcount, int* count, int grantedQoSs[], unsigned char* buf, int buflen)
```

^~~~~~

```
100 int MQTTDeserialize_suback(unsigned short* packetid, int maxcount, int* count, int
101 {
102     buflen = buflen;
```


知识点4【移植，修改编译报错】

步骤4：未使用的函数参数在函数内添加一条使用代码，如buflen = buflen;

```
../third_party/pahomqtt/MQTTPacket/src/MQTTDeserializePublish.c:82:122: error:
unused parameter 'buflen' [-Werror=unused-parameter]
int MQTTDeserialize_ack(unsigned char* packettype, unsigned char* dup, unsigned
short* packetid, unsigned char* buf, int buflen)
                                ^~~~~~
```

```
82 int MQTTDeserialize_ack(unsigned char* packettype, unsigned char* dup, unsigned sh
83 {
84     buflen = buflen;
```


知识点4【移植，修改编译报错】

步骤4：未使用的函数参数在函数内添加一条使用代码，如buflen = buflen;

```
MQTTDeserialize_subscribe :  
../third_party/pahomqtt/MQTTPacket/src/MQTTSubscribeServer.c:35:81: error: unused parameter 'maxcount' [-Werror=unused-parameter]  
int MQTTDeserialize_subscribe(unsigned char* dup, unsigned short* packetid, int  
maxcount, int* count, MQTTString topicFilters[],  
  
^~~~~~  
../third_party/pahomqtt/MQTTPacket/src/MQTTSubscribeServer.c:36:47: error: unused parameter 'buflen' [-Werror=unused-parameter]  
int requestedQoSs[], unsigned char* buf, int buflen)  
                                         ^~~~~~
```

```
35 int MQTTDeserialize_subscribe(unsigned char* dup, unsigned short* packetid, int  
36     int requestedQoSs[], unsigned char* buf, int buflen)  
37 {  
38     maxcount = maxcount;  
39     buflen = buflen;
```

知识点4【移植，修改编译报错】

步骤4：未使用的函数参数在函数内添加一条使用代码，如buflen = buflen;

```
../third_party/pahomqtt/MQTTPacket/src/MQTTDeserializePublish.c:37:69: error:
unused parameter 'buflen' [-Werror=unused-parameter]
    unsigned char** payload, int* payloadlen, unsigned char* buf, int buflen)
                                                                    ^~~~~~
```

```
36 int MQTTDeserialize_publish(unsigned char* dup, int* qos, unsigned char* retain,
37     unsigned char** payload, int* payloadlen, unsigned char* buf, int buflen)
38 {
39     buflen = buflen;
```

知识点4【移植，修改编译报错】

步骤4：未使用的函数参数在函数内添加一条使用代码，如buflen = buflen;

```
../third_party/pahomqtt/MQTTPacket/src/MQTTUnsubscribeServer.c:34:83: error:  
unused parameter 'maxcount' [-Werror=unused-parameter]  
  int MQTTDeserialize_unsubscribe(unsigned char* dup, unsigned short* packetid, i  
nt maxcount, int* count, MQTTString topicFilters[],  
  
  ^~~~~~  
../third_party/pahomqtt/MQTTPacket/src/MQTTUnsubscribeServer.c:35:27: error:  
unused parameter 'len' [-Werror=unused-parameter]  
  unsigned char* buf, int len)  
                        ^~~
```

```
34  int MQTTDeserialize_unsubscribe(unsigned char* dup, uns  
35      unsigned char* buf, int len)  
36  {  
37      maxcount = maxcount;  
38      len = len;  
39  }
```



```
gyz@ubuntu: ~/sharefolder/code-1.0
```

```
[common sign][01]=[0x21]
[common sign][30]=[0x2f]
[common sign][31]=[0x82]
[section sign][00]=[0x3]
[section sign][01]=[0x99]
[section sign][30]=[0xe6]
[section sign][31]=[0xec]
[image_id=0x3c78961e][struct_version=0x0]
[hash_alg=0x0][sign_alg=0x3f][sign_param=0x0]
[section_count=0x1]
[section0_compress=0x0][section0_offset=0x3c0][section0_len=0x5f60]
[section1_compress=0x0][section1_offset=0x0][section1_len=0x0]
-----output/bin/Hi3861_wifiot_app_flash_boot_ota.bin image info print e
nd-----

< ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ >
                                BUILD SUCCESS
< ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ >

See build log from: /home/gyz/sharefolder/code-1.0/vendor/hisi/hi3861/hi3861/bui
ld/build_tmp/logs/build_kernel.log
[212/212] STAMP obj/vendor/hisi/hi3861/hi3861/run_wifiot_scons.stamp
ohos wifiot build success!
gyz@ubuntu:~/sharefolder/code-1.0$
```

本节小结

本讲所学知识点有：

- 知识点1：下载paho mqtt软件包，解压添加到鸿蒙源码中
- 知识点2：在pahomqtt 文件夹下面新建BUILD.gn文件
- 知识点3：配置paho mqtt到sdk中
- 知识点4：移植，修改编译报错

第3节：搭建MQTT测试环境

- 知识点1：下载、启动mqtt 服务器
- 知识点2：下载、启动mqtt客户端工具

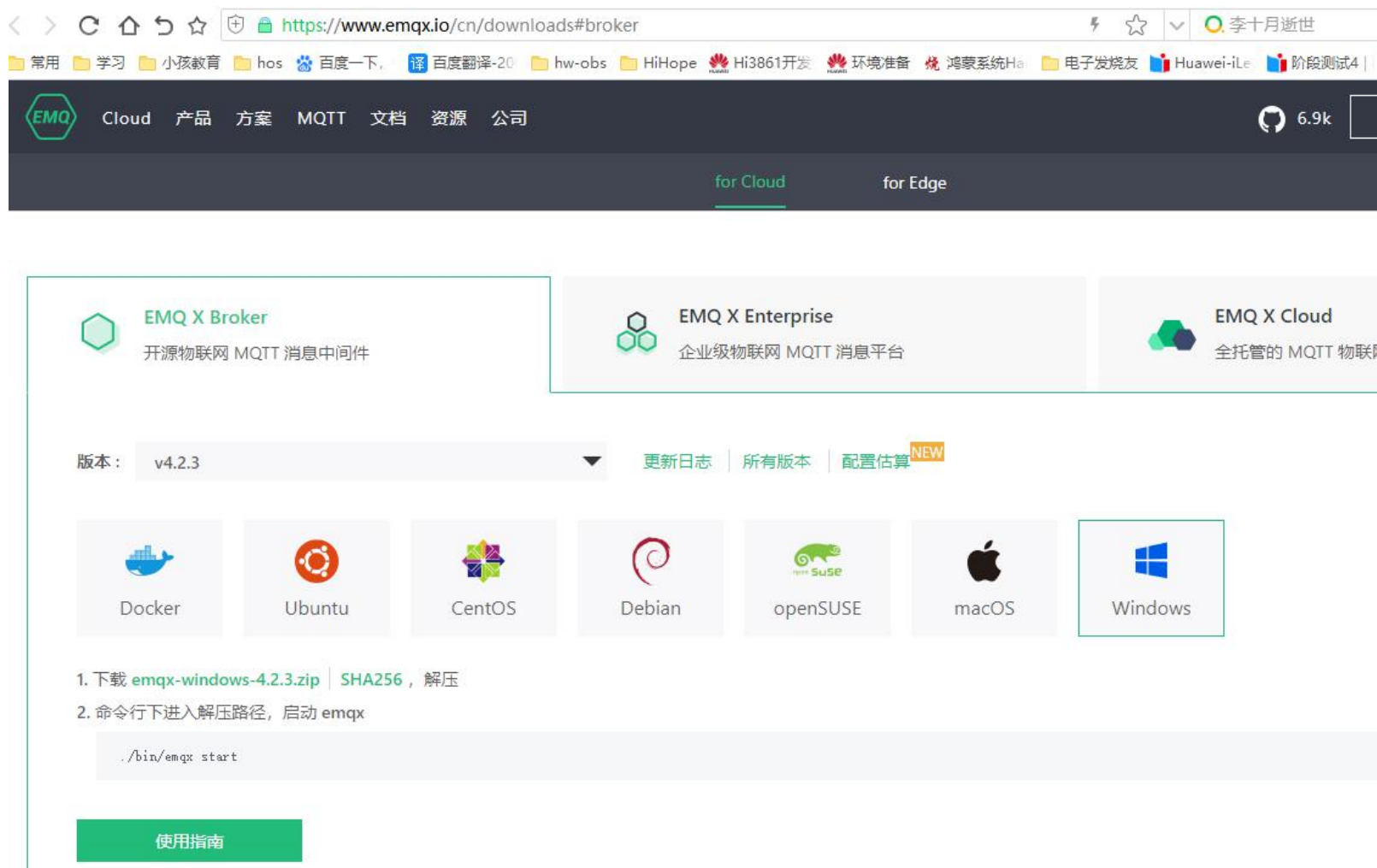
知识点1 【下载、启动mqtt 服务器】

实现步骤:

1. 下载mqtt服务器
2. 启动mqtt服务器
3. 进入服务器后台

知识点1 【下载、启动mqtt 服务器】

步骤1：下载mqtt服务器，官网：<https://www.emqx.io/cn/downloads#broker>



知识点1 【下载、启动mqtt 服务器】

步骤2：启动mqtt服务器，命令行下进入解压路径，运行：./bin/emqx start

```
E:\chinashofti\hos\mqtt\mqtt\emqx-windows-4.2.2\emqx>cd bin
E:\chinashofti\hos\mqtt\mqtt\emqx-windows-4.2.2\emqx\bin>emqx stop
ok
E:\chinashofti\hos\mqtt\mqtt\emqx-windows-4.2.2\emqx\bin>emqx start
E:\chinashofti\hos\mqtt\mqtt\emqx-windows-4.2.2\emqx>cd bin
E:\chinashofti\hos\mqtt\mqtt\emqx-windows-4.2.2\emqx\bin>emqx_ctl status
Node 'emqx@127.0.0.1' is started
emqx 4.2.2 is running
E:\chinashofti\hos\mqtt\mqtt\emqx-windows-4.2.2\emqx\bin>■
```

知识点1 【下载、启动mqtt 服务器】

步骤3：进入服务器后台

➤ 浏览器访问：http://127.0.0.1:18083/#/，默认用户名/密码 admin/public



登录

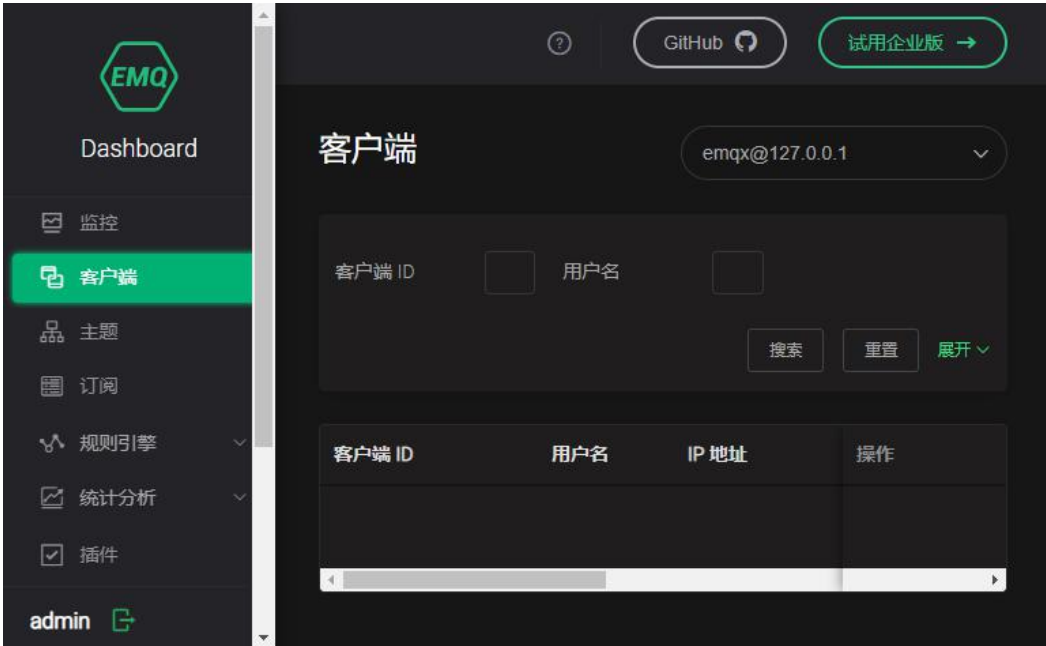
用户名

admin

密码

☒ 记住

登录



EMQ Dashboard

监控

客户端

主题

订阅

规则引擎

统计分析

插件

admin

客户端

emqx@127.0.0.1

客户端 ID 用户名

搜索 重置 展开

客户端 ID	用户名	IP 地址	操作

知识点1 【下载、启动mqtt 服务器】

操作演示



知识点2 【下载、启动mqtt客户端工具】

实现步骤:

1. 下载mqtt客户端工具
2. 启动mqtt客户端工具
3. 连接服务端

知识点2 【下载、启动mqtt客户端工具】

步骤1：下载mqtt客户端工具：<https://repo.eclipse.org/content/repositories/paho-releases/org/eclipse/paho/org.eclipse.paho.ui.app/>

Index of /repositories/paho-releas

Name	Last Modified	Size	D
Parent Directory			
1.0.0/	Thu Jun 26 14:35:00 GMT 2014		
1.0.1/	Fri Nov 28 01:23:11 GMT 2014		
1.0.2/	Wed Feb 18 12:11:54 GMT 2015		
1.1.0/	Wed Jul 20 13:36:39 GMT 2016		
1.1.1/	Mon Mar 20 16:51:19 GMT 2017		
maven-metadata.xml	Wed Dec 04 22:43:39 GMT 2019	491	
maven-metadata.xml.md5	Wed Dec 04 22:43:39 GMT 2019	32	
maven-metadata.xml.sha1	Wed Dec 04 22:43:39 GMT 2019	40	

[org.eclipse.paho.ui.app-1.0.2-org.eclipse.paho.ui.app.executable](#)
[org.eclipse.paho.ui.app-1.0.2-p2artifacts.xml](#)
[org.eclipse.paho.ui.app-1.0.2-p2artifacts.xml.md5](#)
[org.eclipse.paho.ui.app-1.0.2-p2artifacts.xml.sha1](#)
[org.eclipse.paho.ui.app-1.0.2-p2metadata.xml](#)
[org.eclipse.paho.ui.app-1.0.2-p2metadata.xml.md5](#)
[org.eclipse.paho.ui.app-1.0.2-p2metadata.xml.sha1](#)
[org.eclipse.paho.ui.app-1.0.2-win32.win32.x86.zip](#)
[org.eclipse.paho.ui.app-1.0.2-win32.win32.x86.zip.md5](#)
[org.eclipse.paho.ui.app-1.0.2-win32.win32.x86.zip.sha1](#)
[org.eclipse.paho.ui.app-1.0.2-win32.win32.x86_64.zip](#)
[org.eclipse.paho.ui.app-1.0.2-win32.win32.x86_64.zip.md5](#)
[org.eclipse.paho.ui.app-1.0.2-win32.win32.x86_64.zip.sha1](#)

知识点2 【下载、启动mqtt客户端工具】

步骤2：启动mqtt客户端工具

卷 (E:) > chinashofti > hos > mqtt > mqtt > org.eclipse.paho.ui.app-1.0.2-win32.win32.x86_64

名称	修改日期	类型	大小
configuration	2020/11/24 14:11	文件夹	
data	2020/11/24 14:11	文件夹	
p2	2020/11/24 14:11	文件夹	
plugins	2020/11/24 14:11	文件夹	
workspace	2020/11/24 14:11	文件夹	
artifacts.xml	2015/2/18 7:11	XML 文件	16 KB
eclipsec.exe	2015/2/18 6:51	应用程序	18 KB
paho.exe	2015/2/18 6:51	应用程序	305 KB
paho.ini	2015/2/18 7:11	配置设置	1 KB

步骤3：连接服务端



知识点2 【下载、启动mqtt客户端工具】

操作演示



本节小结

本讲所学知识点有：

- 知识点1：下载、启动mqtt 服务器
- 知识点2：下载、启动mqtt客户端工具

第4节：测试MQTT

- 知识点1：配置测试程序
- 知识点2：配置PC端客户端

Previous Message

Current Message

AT+SCAN	AT+SCANCHN	AT+SCANSSID	AT+SCANPRSSID
AT+SCANRESULT	AT+CONN	AT+FCNN	AT+DISCONN
AT+STASTAT	AT+RECONN	AT+STARTAP	AT+SETAPADV
AT+STOPAP	AT+SHOWSTA	AT+DEAUTHSTA	AT+APSCAN
AT+RXINFO	AT+CC	AT+TPC	AT+TRC
AT+SETRATE	AT+ARLOG	AT+VAPINFO	AT+USRINFO
AT+SLP	AT+WKGPIO	AT+USLP	AT+ARP
AT+PS	AT+ND	AT+CSV	AT+FTM
AT+FTMERASE	AT+SETUART	AT+IFCFG	AT+STARTSTA
AT+ALTX	AT+ALRX	AT+CALFREQ	AT+SETRPWR
AT+RCALDATA	AT+SETIOMODE	AT+GETIOMODE	AT+GPIODIR
AT+WTGPIO	AT+RDGPIO	AT+MQTTTEST	
OK			
AT+MQTTTEST			
ConnectTo(0): 0			
OK			
netifapi_dhcp_start: 0			

GitHub

试用企业版 →

emqx@127.0.0.1

搜索

重置

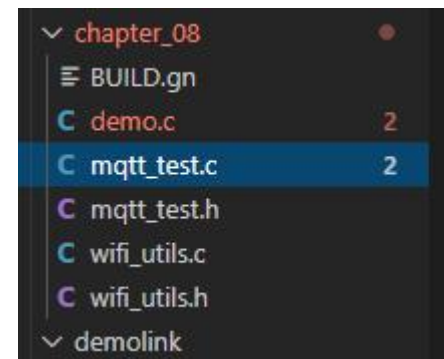
展开

订阅数量	连接状态	操作
	● 已连接	踢除
	● 已连接	踢除

知识点1 【配置测试程序】

实现步骤:

1. 配置mqtt服务器的IP, 端口
2. 配置clientId,
3. 配置订阅主题
4. 配置发布主题
5. 将测试注册为AT命令



知识点1 【配置测试程序】

步骤1：配置mqtt服务器的IP，端口

<code/>

```
//applications/sample/wifi-iot/app/chapter_08/mqtt_test.c
```

```
char *host = "192.168.0.104";
```

```
int port = 1883;
```



知识点1 【配置测试程序】

步骤2: 配置clientID

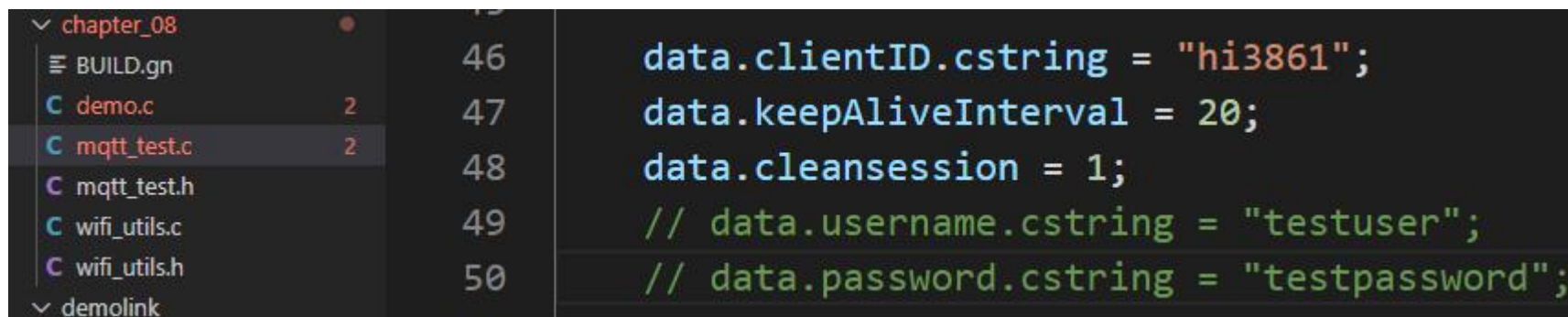
<code/>

```
//applications/sample/wifi-iot/app/chapter_08/mqtt_test.c
```

```
data.clientID.cstring = "hi3861";
```

```
data.keepAliveInterval = 20;
```

```
data.cleansession = 1;
```



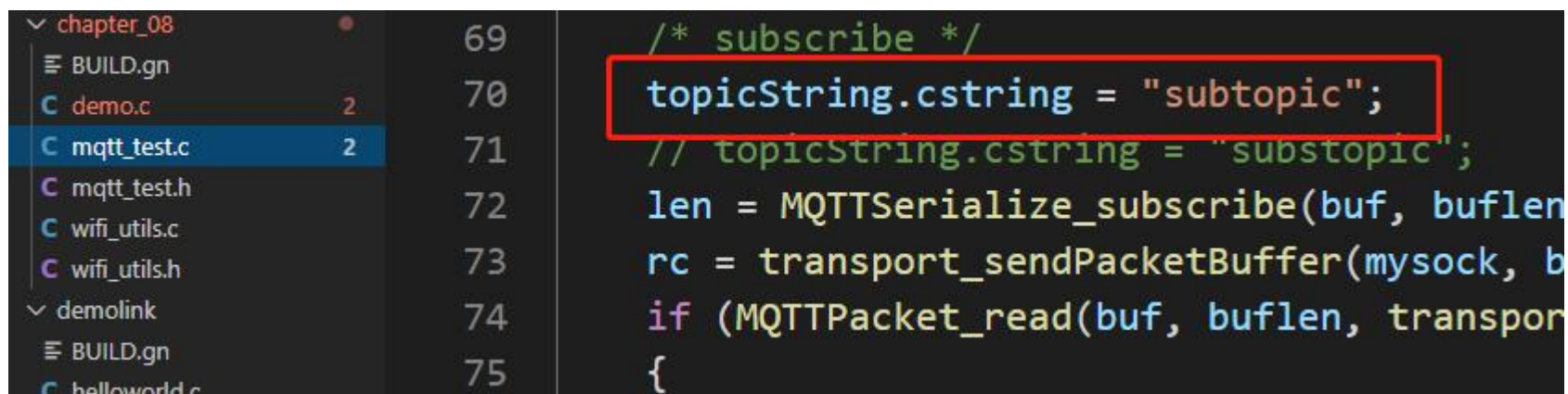
```
46 //applications/sample/wifi-iot/app/chapter_08/mqtt_test.c
47 data.clientID.cstring = "hi3861";
48 data.keepAliveInterval = 20;
49 data.cleansession = 1;
50 // data.username.cstring = "testuser";
51 // data.password.cstring = "testpassword";
```

知识点1 【配置测试程序】

步骤3：配置订阅主题

<code/>

```
//applications/sample/wifi-iot/app/chapter_08/mqtt_test.c  
topicString.cstring = "subtopic";
```



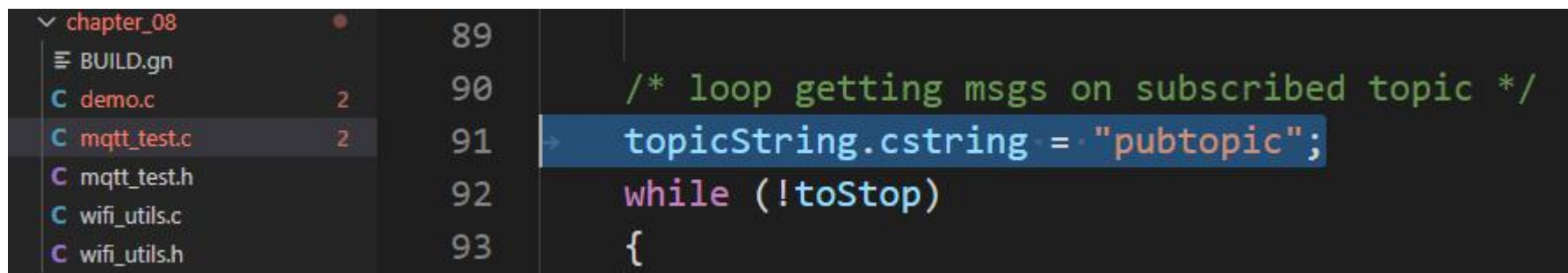
```
69  
70 /* subscribe */  
71 topicString.cstring = "subtopic";  
72 // topicString.cstring = "subtopic";  
73 len = MQTTSerialize_subscribe(buf, buflen,  
74 rc = transport_sendPacketBuffer(mysock, b  
75 if (MQTTPacket_read(buf, buflen, transpor  
{
```

知识点1 【配置测试程序】

步骤4：配置发布主题

<code/>

```
//applications/sample/wifi-iot/app/chapter_08/mqtt_test.c  
topicString.cstring = "pubtopic";
```



```
89  
90 /* loop getting msgs on subscribed topic */  
91 topicString.cstring = "pubtopic";  
92 while (!toStop)  
93 {
```

知识点1 【配置测试程序】

步骤5：将测试每种注册为AT命令

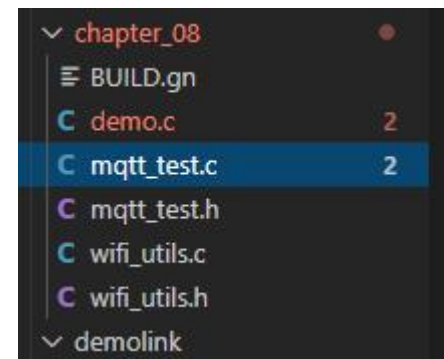
<code/>

```
//applications/sample/wifi-iot/app/chapter_08/demo.c
const at_cmd_func g_at_mqtt_func_tbl[] = {
    {"+MQTTTEST", 9, HI_NULL, HI_NULL, HI_NULL, (at_call_back_func)at_exe_mqtt_test_cmd},
};
void AtMqttTestEntry(void){
    hi_at_register_cmd(g_at_mqtt_func_tbl, sizeof(g_at_mqtt_func_tbl) / sizeof(g_at_mqtt_func_tbl[0]));
}
SYS_RUN(AtMqttTestEntry);
```


知识点2 【配置PC端客户端】

实现步骤:

1. 配置订阅主题
2. 发布消息



步骤1：配置订阅主题：pubtopic

[illegible]

知识点2 【配置PC端客户端】

步骤2：发布消息->发布主题为subtopic的消息

发布

主题

subtopic

服务质量

0 - 至多一次 ▼

☐ 已保留

☐ 16进制

消息

HI I am from pc 3333
1111

文件...

发布

第4节：测试MQTT

操作演示



本节小结

本讲所学知识点有：

- 知识点1：配置测试程序
- 知识点2：配置PC端客户端

本章总结

本章所学内容有：

- 第1节：MQTT协议
- 第2节：paho mqtt移植
- 第3节：搭建MQTT测试环境
- 第4节：测试MQTT

任务挑战

挑战任务1：参考本章知识，完成MQTT控制led的亮与灭

AT+SCAN

AT+SCANCHN

AT+SCANSSID

AT+SCANPRSSID

AT+SCANRESULT

AT+CONN

AT+FCONN

AT+DISCONN

AT+STASTAT

AT+RECONN

AT+STARTAP

AT+SETAPADV

AT+STOPAP

AT+SHOWSTA

AT+DEAUTHSTA

AT+APSCAN

AT+RXINFO

AT+CC

AT+TPC

AT+TRC

AT+SETRATE

AT+ARLOG

AT+VAPINFO

AT+USRINFO

AT+SLP

AT+WKGPIIO

AT+USLP

AT+ARP

AT+PS

AT+ND

AT+CSV

AT+FTM

AT+FTMERASE

AT+SETUART

AT+IFCFG

AT+STARTSTA

AT+ALTX

AT+ALRX

AT+CALFREQ

AT+SETRPWR

AT+RCALDATA

AT+SETIOMODE

AT+GETIOMODE

AT+GPIODIR

AT+WTGPIIO

AT+RDGPIIO

AT+MQTTTEST

OK

AT+MQTTTEST

ConnectTo(0): 0

OK

netifapi_dhcp_start: 0

GitHub

试用企业版 →

emqx@127.0.0.1

搜索

重置

展开

订阅数量	连接状态	操作
	● 已连接	踢除
	● 已连接	踢除

THANKS

更多学习视频，关注宅客学院.....

