

# QuecPython MQTT 用户指导

**LTE Standard 模块系列**

版本：1.0.0

日期：2020-11-11

状态：临时文件

上海移远通信技术股份有限公司始终以为客户提供最及时、最全面的服务为宗旨。如需任何帮助，请随时联系我司上海总部，联系方式如下：

上海移远通信技术股份有限公司

上海市闵行区田林路 1016 号科技绿洲 3 期（B 区）5 号楼 邮编：200233

电话：+86 21 51086236 邮箱：[info@quectel.com](mailto:info@quectel.com)

或联系我司当地办事处，详情请登录：<http://www.quectel.com/cn/support/sales.htm>。

如需技术支持或反馈我司技术文档中的问题，可随时登陆如下网址：

<http://www.quectel.com/cn/support/technical.htm> 或发送邮件至：[support@quectel.com](mailto:support@quectel.com)。

## 前言

上海移远通信技术股份有限公司提供该文档内容用以支持其客户的产品设计。客户须按照文档中提供的规范、参数来设计其产品。因未能遵守有关操作或设计规范而造成的损害，上海移远通信技术股份有限公司不承担任何责任。在未声明前，上海移远通信技术股份有限公司有权对该文档进行更新。

## 免责声明

上海移远通信技术股份有限公司尽力确保开发中功能的完整性、准确性、及时性或效用，但不排除上述功能错误或遗漏的可能。除非其他有效协议另有规定，否则上海移远通信技术股份有限公司对开发中功能的使用不做任何暗示或明示的保证。在适用法律允许的最大范围内，上海移远通信技术股份有限公司不对任何因使用开发中功能而遭受的损失或损害承担责任，无论此类损失或损害是否可以预见。

## 保密义务

除非上海移远通信技术股份有限公司特别授权，否则我司所提供文档和信息的接收方须对接收的文档和信息保密，不得将其用于除本项目的实施与开展以外的任何其他目的。未经上海移远通信技术股份有限公司书面同意，不得获取、使用或向第三方泄露我司所提供的文档和信息。对于任何违反保密义务、未经授权使用或以其他非法形式恶意使用所述文档和信息的违法侵权行为，上海移远通信技术股份有限公司有权追究法律责任。

## 版权申明

本文档版权属于上海移远通信技术股份有限公司，任何人未经我司允许而复制转载该文档将承担法律责任。

版权所有 ©上海移远通信技术股份有限公司 2020，保留一切权利。

**Copyright © Quectel Wireless Solutions Co., Ltd. 2020.**

# 文档历史

## 修订记录

版本	日期	作者	变更表述
-	2020-11-11	Kingka/Rivern	文档创建
1.0.0	2020-11-11	Kingka/Rivern	临时版本

## 目录

文档历史 .....	2
目录 .....	3
图片索引 .....	4
<b>1 引言 .....</b>	<b>5</b>
<b>2 MQTT 概述 .....</b>	<b>6</b>
2.1. MQTT 简介 .....	6
2.1.1. MQTT 设计原则 .....	6
2.1.2. MQTT 业务场景 .....	6
2.2. MQTT 发布/订阅模式 .....	7
2.3. MQTT 协议原理 .....	7
2.4. MQTT 中的相关方法 .....	8
2.5. MQTT 主题 .....	8
2.6. MQTT 服务质量 .....	8
<b>3 MQTT API .....</b>	<b>10</b>
3.1. API 介绍 .....	10
3.1.1. MQTTClient .....	10
3.1.2. MQTTClient.set_callback .....	11
3.1.3. MQTTClient.connect .....	11
3.1.4. MQTTClient.disconnect .....	11
3.1.5. MQTTClient.ping .....	12
3.1.6. MQTTClient.publish .....	12
3.1.7. MQTTClient.subscribe .....	13
3.1.8. MQTTClient.check_msg .....	13
3.1.9. MQTTClient.wait_msg .....	13
3.1.10. MQTTClient.set_last_will .....	14
<b>4 MQTT 开发说明 .....</b>	<b>15</b>
4.1. MQTT 服务器搭建和测试 .....	15
4.1.1. 搭建 MQTT 服务器 .....	15
4.1.2. 启动 MQTT 服务器 .....	16
4.1.3. 验证 MQTT 服务器有效性 .....	16
4.1.3.1. 安装 MQTT 客户端 .....	16
4.1.3.2. 配置 MQTT 客户端 .....	17
4.1.3.3. MQTT 交互测试 .....	18
<b>5 使用 MQTT 进行数据发布订阅 .....</b>	<b>20</b>
<b>6 附录 .....</b>	<b>23</b>

## 图片索引

图 1: MQTT 业务场景.....	7
图 2: MQTT 协议原理图示 .....	7
图 3: mosquitto 帮助信息 .....	15
图 4: MQTT 服务器启动成功.....	16
图 5: MQTT.fx 启动界面 .....	17
图 6: MQTT 连接配置.....	18
图 7: MQTT 服务器与客户端交互 .....	19
图 8: 发布消息 .....	19
图 9: 订阅主题 .....	19
图 10: 接入开发板 .....	20
图 11: 通信猫在线客户端服务器.....	21

# 1 引言

本文档主要以 EC100Y-CN 模块为例介绍如何使用 QuecPython 类库 API 实现 MQTT 功能,包括 MQTT 介绍、服务器的搭建和测试及数据发布订阅。

本文档适用以下移远通信模块:

- EC100Y-CN
- EC600S-CN

## 2 MQTT 概述

### 2.1. MQTT 简介

MQTT 是基于代理的发布/订阅模式通讯协议，具有开放、简单、轻量和易于实现等特点。MQTT 最大优点在于，可以以极少的代码和有限的网络带宽，为远程设备连接提供实时可靠的消息服务。

由于规范简单，适合对低功耗要求严格和网络带宽有限的物联网场景，例如：遥感数据、汽车、智能家居、智慧城市和医疗医护等。

#### 2.1.1. MQTT 设计原则

MQTT 协议遵循以下设计原则：

1. 功能精简。
2. 采用发布/订阅（Pub/Sub）模式，方便消息传递。
3. 允许用户动态创建主题，降低运营成本。
4. 降低传输量至最低，提高传输效率。
5. 关注低带宽、高延迟、不稳定的网络等因素。
6. 支持连续会话控制。
7. 对客户端计算能力包容性强。
8. 提供服务质量管理。
9. 对传输数据的类型与格式无强制要求，保持灵活性。

#### 2.1.2. MQTT 业务场景

运用 MQTT 协议，设备可以方便地连接到物联网云服务，管理设备并处理数据，最后应用到各种业务场景，如下图所示：

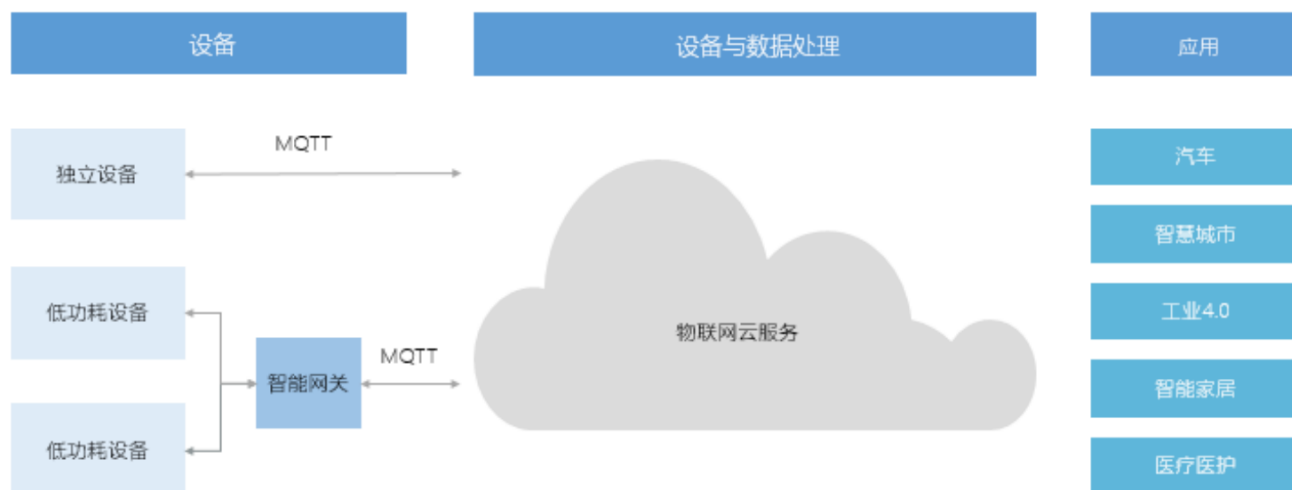


图 1: MQTT 业务场景

## 2.2. MQTT 发布/订阅模式

发布/订阅模式提供传统客户端-服务器体系结构的替代方法。在客户端服务器模型中，客户端直接与端点进行通信。发布/订阅模型解耦了发送消息的客户端（发布者）与接收消息的客户端（订阅者）之间的关系，二者并不直接建立联系。发布者与订阅者之间由第三个组件（代理）进行连接，代理过滤所有传入的消息，并将其正确分发给订阅者。

发布/订阅模式优点如下：

1. 发布者与订阅者无需互相知悉，只需使用同一个代理即可。
2. 发布者和订阅者无需交互，发布者不必因等待订阅者确认而导致锁定。
3. 发布者和订阅者无需同时在线，可自由选择时间发布/接收消息。

## 2.3. MQTT 协议原理

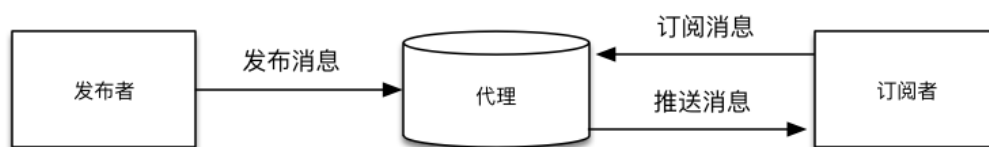


图 2: MQTT 协议原理图示

1. 实现 MQTT 协议需要：客户端和服务端。
2. MQTT 协议中有三种身份：发布者（Publisher）、代理（Broker）、订阅者（Subscriber）。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。

3. 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，订阅者订阅后，就会收到该主题的消息内容；
- Payload，可以理解为消息的内容，是指订阅者具体要使用的内容。

## 2.4. MQTT 中的相关方法

MQTT 中定义了一些方法（或动作），来于表示对确定资源所进行操作。这个资源可以代表预先存在的数据或动态生成数据，这取决于服务器的实现。通常来说，资源是指服务器上的文件或输出：

- Connect，等待与服务器建立连接；
- Disconnect，等待 MQTT 客户端完成所做的工作，并与服务器断开 TCP/IP 会话；
- Subscribe，等待完成订阅；
- UnSubscribe，等待服务器取消客户端的一个或多个 topics 订阅；
- Publish，MQTT 客户端发送消息请求，发送完成后返回应用程序线程。

## 2.5. MQTT 主题

MQTT 通过主题对消息进行分类。主题一般是一个 UTF-8 的字符串，可通过符号“/”表示层级关系。主题可直接使用，无需再次创建。主题还可以通过通配符进行过滤，其中，“+”用于过滤一个层级，“#”一般位于主题后表示过滤任意级别的层级。示例如下：

1. building-b/floor-5：代表 B 楼 5 层的设备。
2. +/floor-5：代表任何一个楼的 5 层的设备。
3. building-b/#：代表 B 楼所有的设备。

### 备注

MQTT 允许使用通配符订阅主题，但并不允许使用通配符广播。

## 2.6. MQTT 服务质量

MQTT 支持三种不同级别的服务质量，为不同场景保证消息传输的可靠性。

- 级别 0：最多一次。  
消息发送者仅发送一次，不会重复发送。一般用于传输不重要数据的场景。

- 级别 1：至少一次。  
消息发送者发送消息以后，若未收到消息接收者的确认信息，则会再次发送，直到接收到消息接收者的确认信息。这种情况可能导致重复消息。一般用于日志处理的场景。
- 级别 2：只发送一次。  
消息发送者发送消息以后等待消息接收者的确认消息，接收到确认消息后消息发送者删除消息并通知消息接收者。如此会减少并发或者增加延时，但是若数据丢失或者重复消息不可接受时，可设置为级别 2。

## 3 MQTT API

### 3.1. API 介绍

#### 3.1.1. MQTTClient

该函数用于构建 MQTT 连接对象。

- 函数原型

```
MQTTClient(client_id, server, port=0, user=None, password=None, keepalive=0, ssl=False,
ssl_params={})
```

- 参数

*client\_id*:

字符串类型。客户端 ID，具有唯一性，也可能是经过加密处理后的 *client\_id*，例如阿里云。

*server*:

字符串类型。服务端地址，可以是 IP 或者域名。

*port*:

（可选）整型。服务器端口，默认为 1883，通过 SSL/TLS 的 MQTT 的默认端口是 8883。

*user*:

（可选）字符串类型。在服务器上注册的用户名，也可能是经过加密处理的用户名，例如阿里云。

*password*:

（可选）字符串类型。在服务器上注册的密码，也可能是经过加密处理的密码，例如阿里云。

*keepalive*:

（可选）整型。客户端的 keepalive 超时值。默认为 60 秒，范围为 60~120 秒。

*ssl*:

（可选）布尔型。是否使能支持 SSL/TLS。

*ssl\_params*:

（可选）字符串类型。SSL/TLS 参数。

- 返回值

MQTT 对象。

### 3.1.2. MQTTClient.set\_callback

该函数用于设置回调函数。

- 函数原型

```
MQTTClient.set_callback(callback)
```

- 参数

*callback:*

消息回调函数。

- 返回值

无。

### 3.1.3. MQTTClient.connect

该函数用于与服务器建立连接。

- 函数原型

```
MQTTClient.connect(clean_session=True)
```

- 参数

*clean\_session:*

布尔型。可选参数，一个决定客户端类型的布尔值。如果为 **True**，那么代理将在其断开连接时删除有关此客户端的所有信息。如果为 **False**，则客户端是持久客户端，当客户端断开连接时，订阅信息和排队消息将被保留。默认为 **False**。

- 返回值

无。

### 3.1.4. MQTTClient.disconnect

该函数用于与服务器断开连接。

- 函数原型

```
MQTTClient.disconnect()
```

- 参数

无。

- 返回值

无。

### 3.1.5. MQTTClient.ping

该函数用于向服务器发送 ping 包，检测保持连通性。

- 函数原型

```
MQTTClient.ping()
```

- 参数

无。

- 返回值

无。

### 3.1.6. MQTTClient.publish

该函数用于发布消息。

- 函数原型

```
MQTTClient.publish(topic,msg)
```

- 参数

*topic:*

字符串类型。消息主题。

*msg:*

字符串类型。需要发送的数据。

- 返回值

无。

### 3.1.7. MQTTClient.subscribe

该函数用于订阅 MQTT 主题。

- 函数原型

```
MQTTClient.subscribe(topic,qos)
```

- 参数

*topic:*

字符串类型。MQTT 主题。

*msg:*

字符串类型。MQTT 消息服务质量，默认为 0，可选择 0 或 1。

- 返回值

无。

### 3.1.8. MQTTClient.check\_msg

该函数用于检查服务器是否有待处理消息。

- 函数原型

```
MQTTClient.check_msg()
```

- 参数

无。

- 返回值

无。

### 3.1.9. MQTTClient.wait\_msg

该函数用于等待服务器消息响应。

- 函数原型

```
MQTTClient.wait_msg()
```

- 参数

无。

- 返回值

无。

### 3.1.10. MQTTClient.set\_last\_will

该函数用于设置要发送给服务器的遗嘱，客户端没有调用 `disconnect()` 异常断开，则发送通知到客户端。

- 函数原型

```
MQTTClient.set_last_will(topic,msg,retain=False,qos=0)
```

- 参数

*topic*:

字符串类型。遗嘱主题。

*msg*:

字符串类型。遗嘱内容。

*retain*:

布尔型。`retain=True` broker 会一直保留消息，默认为 `False`。

*qos*:

整型。消息服务质量，范围为 0~2。

- 返回值

无。

# 4 MQTT 开发说明

## 4.1. MQTT 服务器搭建和测试

上位机安装 MQTT 服务器，即消息代理。本文档以 mosquitto 开源消息代理软件为例。

### 4.1.1. 搭建 MQTT 服务器

1. Linux 系统通过以下命令进行安装。

```
sudo apt install mosquitto
```

2. Windows 系统访问 <https://mosquitto.org/download/> 下载适配版本的.exe 安装包。

以 Windows 系统为例，安装完成后进入 mosquitto 安装目录，启动命令行执行如下命令，查看 mosquitto 的用法：

```
mosquitto -h
```

```
C:\Program Files\mosquitto>mosquitto -h
mosquitto version 1.6.8

mosquitto is an MQTT v3.1.1 broker.

Usage: mosquitto [-c config_file] [-d] [-h] [-p port]

-c : specify the broker config file.
-d : put the broker into the background after starting.
-h : display this help.
-p : start the broker listening on the specified port.
    Not recommended in conjunction with the -c option.
-v : verbose mode - enable all logging types. This overrides
    any logging options given in the config file.

See http://mosquitto.org/ for more information.
```

图 3: mosquitto 帮助信息



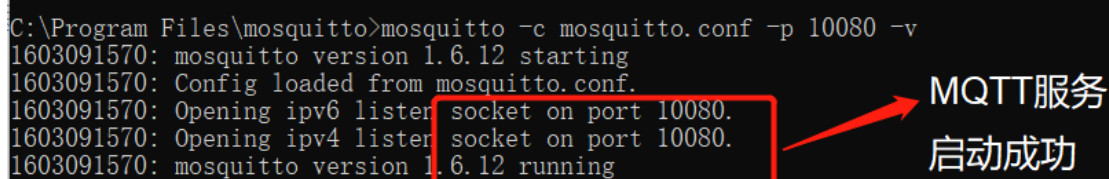
上图中的两个关键参数-c 和-p, -c 用于指定 MQTT 服务的配置, -p 用于指定服务端口。

### 4.1.2. 启动 MQTT 服务器

启动 MQTT 服务器前, 要确保服务器所在网络能够被设备访问。在命令行中键入以下命令, 启动 MQTT 服务:

```
mosquitto -c mosquitto.conf -p 10080 -v
```

测试阶段, 配置文件直接使用安装目录下的默认配置即可; 本测试本地端口使用 10080。



```
C:\Program Files\mosquitto>mosquitto -c mosquitto.conf -p 10080 -v
1603091570: mosquitto version 1.6.12 starting
1603091570: Config loaded from mosquitto.conf.
1603091570: Opening ipv6 listener socket on port 10080.
1603091570: Opening ipv4 listener socket on port 10080.
1603091570: mosquitto version 1.6.12 running
```

MQTT服务  
启动成功

图 4: MQTT 服务器启动成功

### 4.1.3. 验证 MQTT 服务器有效性

#### 4.1.3.1. 安装 MQTT 客户端

本文档以 MQTT.fx 的客户端为例。访问 <http://mqttfx.bceapp.com>, 下载并安装客户端软件。

安装完成后, 打开 MQTT.fx, 默认进入发布界面, 如下图所示:

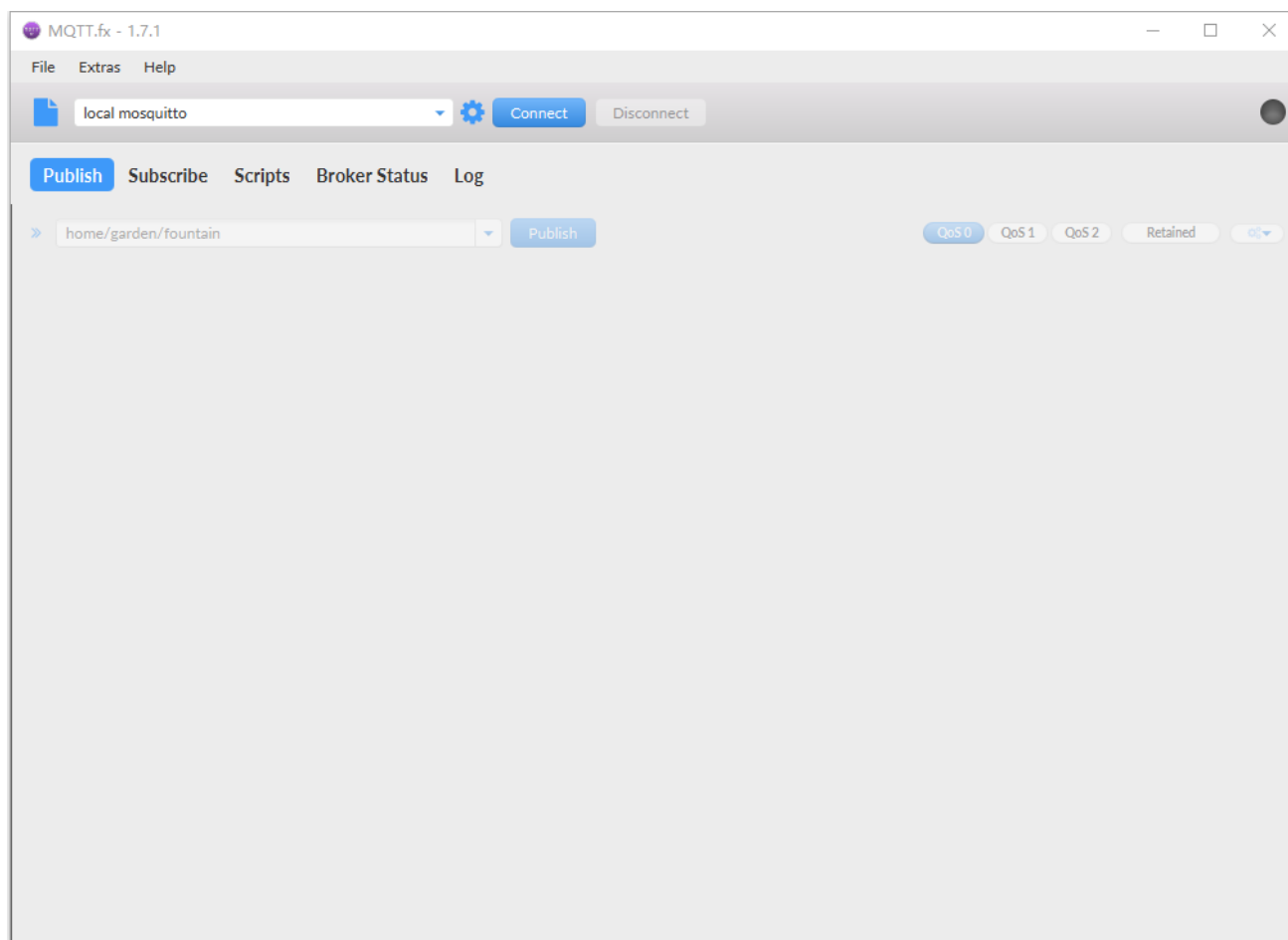


图 5: MQTT.fx 启动界面

#### 4.1.3.2. 配置 MQTT 客户端

点击上图中配置按钮, 添加 MQTT 连接配置。

配置界面默认添加了 m2m.eclipse.org 和 mosquitto 服务的配置。可修改任意配置，主要修改“Broker Address”（代理地址）和“Broker Port”（代理端口）两项。亦可点击配置界面左下角的加号，添加自定义服务连接配置。

配置选项“General”、“User Credentials”、“SSL/TLS”、“Proxy”和“LWT”，可根据服务器的要求进行配置，包括连接超时时间、保活时间间隔、MQTT 协议版本、用户名/密码及 SSL 相关配置。本文中采用默认配置。

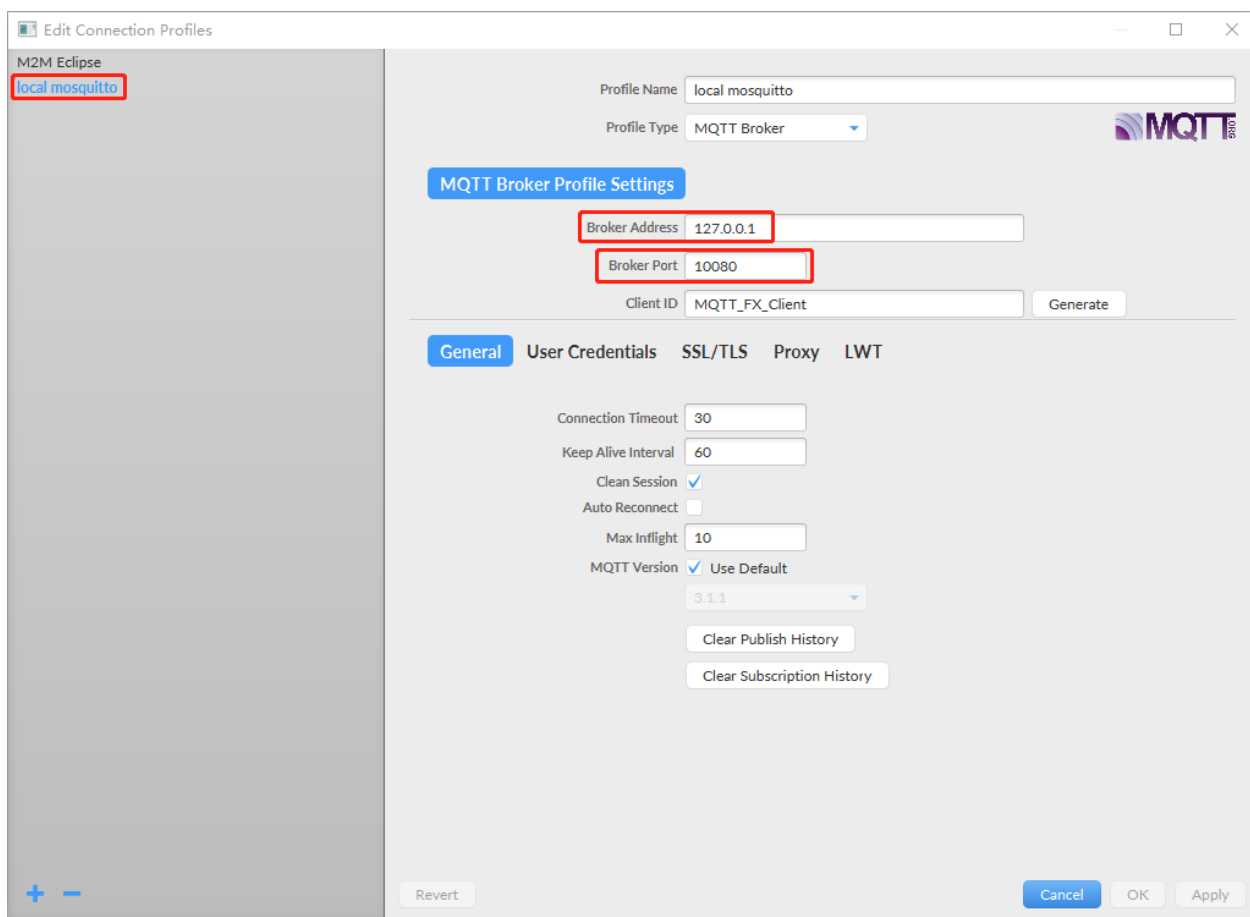


图 6: MQTT 连接配置

#### 4.1.3.3. MQTT 交互测试

##### 1. 建立 MQTT 连接

在 MQTT.fx 启动后进入的默认界面点击“**Connect**”按钮，建立 MQTT 连接。连接成功后，mosquitto 服务器打印的日志请参考 MQTT 服务器与客户端交互。连接成功后即可发布消息、订阅主题，默认已填充待订阅和发布的消息主题。

```

1603093267: New client connected from 127.0.0.1 as 5c705e8e04654934bc3dfacbdffbdlf82 (p2, c1, k60).
1603093267: No will message specified.
1603093267: Sending CONNACK to 5c705e8e04654934bc3dfacbdffbdlf82 (0, 0)
1603093276: Received PUBLISH from 5c705e8e04654934bc3dfacbdffbdlf82 (d0, q1, r0, m1, '/wyc/test', ... (8 bytes))
1603093276: Sending PUBACK to 5c705e8e04654934bc3dfacbdffbdlf82 (m1, rc0)
1603093294: Received PUBLISH from 5c705e8e04654934bc3dfacbdffbdlf82 (d0, q1, r0, m2, '/wyc/test', ... (8 bytes))
1603093294: Sending PUBACK to 5c705e8e04654934bc3dfacbdffbdlf82 (m2, rc0)
1603093323: Received SUBSCRIBE from 5c705e8e04654934bc3dfacbdffbdlf82
1603093323: /wyc/test (QoS 0)
1603093323: 5c705e8e04654934bc3dfacbdffbdlf82 0 /wyc/test
1603093323: Sending SUBACK to 5c705e8e04654934bc3dfacbdffbdlf82
1603093326: Received PUBLISH from 5c705e8e04654934bc3dfacbdffbdlf82 (d0, q1, r0, m4, '/wyc/test', ... (8 bytes))
1603093326: Sending PUBACK to 5c705e8e04654934bc3dfacbdffbdlf82 (m4, rc0)
1603093326: Sending PUBLISH to 5c705e8e04654934bc3dfacbdffbdlf82 (d0, q0, r0, m0, '/wyc/test', ... (8 bytes))
1603093386: Received PINGREQ from 5c705e8e04654934bc3dfacbdffbdlf82
1603093386: Sending PINGRESP to 5c705e8e04654934bc3dfacbdffbdlf82

```

图 7: MQTT 服务器与客户端交互

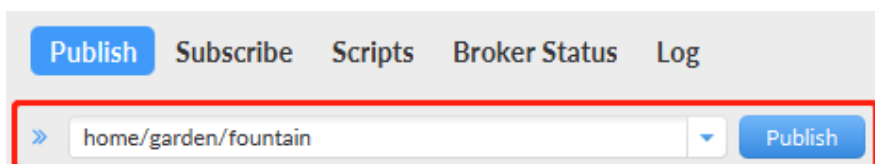


图 8: 发布消息

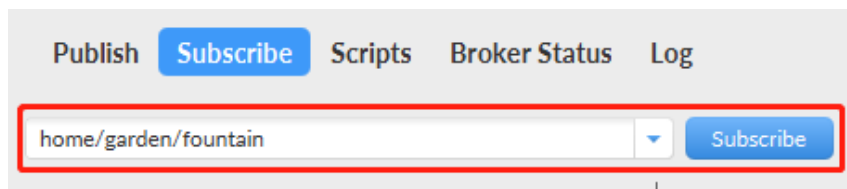


图 9: 订阅主题

MQTT 允许用户动态创建主题，也可动态修改主题。将待订阅和发布的消息主题配置一致，如此可实现回显功能。客户端向服务器发布的消息可立即分发至到客户端。

## 5 使用 MQTT 进行数据发布订阅

本测试采用 echo 功能作为测试示例。

**步骤1:** 开发板接入电脑，接入后详细操作方法，请参考《Quectel\_QuecPython\_基础操作说明》。

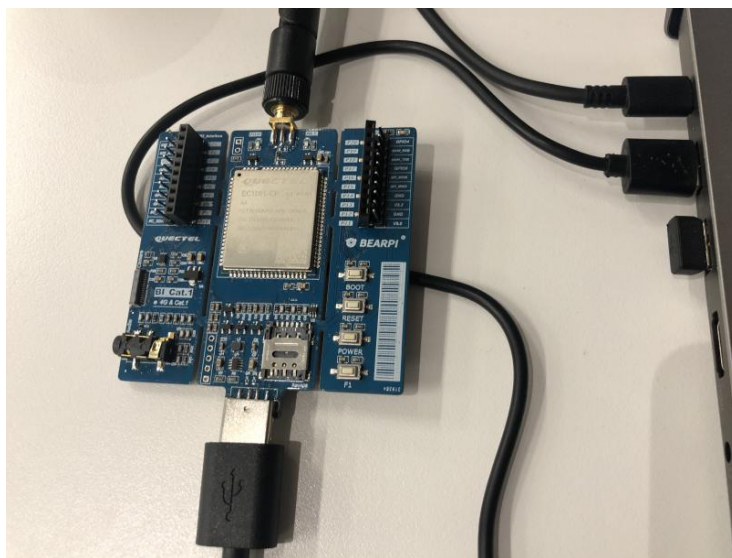


图 10: 接入开发板

**步骤2:** 建立 `test.py` 文件，导入 `umqtt` 模块，我们以连接 [mq.tongxinmao.com](http://mq.tongxinmao.com) 网址为例创建以下代码：

```
from umqtt import MQTTClient
state = 0
def sub_cb(topic, msg):
    global state
    print("subscribe recv:")
    print(topic, msg)
    state = 1

#创建一个 mqtt 实例
c = MQTTClient("umqtt_client", "mq.tongxinmao.com", '18830')
#设置消息回调
c.set_callback(sub_cb)
#建立连接
```

```
c.connect()
#订阅主题
c.subscribe(b"/public/TEST/quecpython")
print("Connected to mq.tongxinmao.com, subscribed to /public/TEST/quecpython topic" )
#发布消息
c.publish(b"/public/TEST/quecpython", b"my name is Kingka!")
print("Publish topic: /public/TEST/quecpython, msg: my name is Quecpython")

while True:
    c.wait_msg() #阻塞函数，监听消息
    if state == 1:
        break

#关闭连接
c.disconnect()
```

## 备注

也可以通过访问通信猫在线客户端服务器进行 MQTT 验证：  
<http://www.tongxinmao.com/txm/webmqtt.php>。

图 11：通信猫在线客户端服务器

**步骤3：** 将 `test.py` 文件上传至 EC100Y-CN QuecPython 开发板内，详细上传方法请参考《Quectel\_QuecPython\_基础操作说明》。

**步骤4：** 在开发板中运行 `test.py` 文件，即可以看到模块执行结果，如下图所示：

```
>>> import example
>>> example.exec('test.py')
Connected to mq.tongxinmao.com, subscribed to /public/TEST/quecpython topic
Publish topic: /public/TEST/quecpython, msg: my name is Quecpython
subscribe recv:
b'/public/TEST/quecpython' b'my name is Kingka!'
>>>
```

# 6 附录

表 1：术语缩写

缩写	英文全称	中文全称
API	Application Programming Interface	应用程序编程接口
IP	Internet Protocol	网际互连协议
MQTT	Message Queuing Telemetry Transport	消息队列遥测传输
SSL	Secure Sockets Layer	安全套接层
TLS	Transport Layer Security	传输层安全（协议）