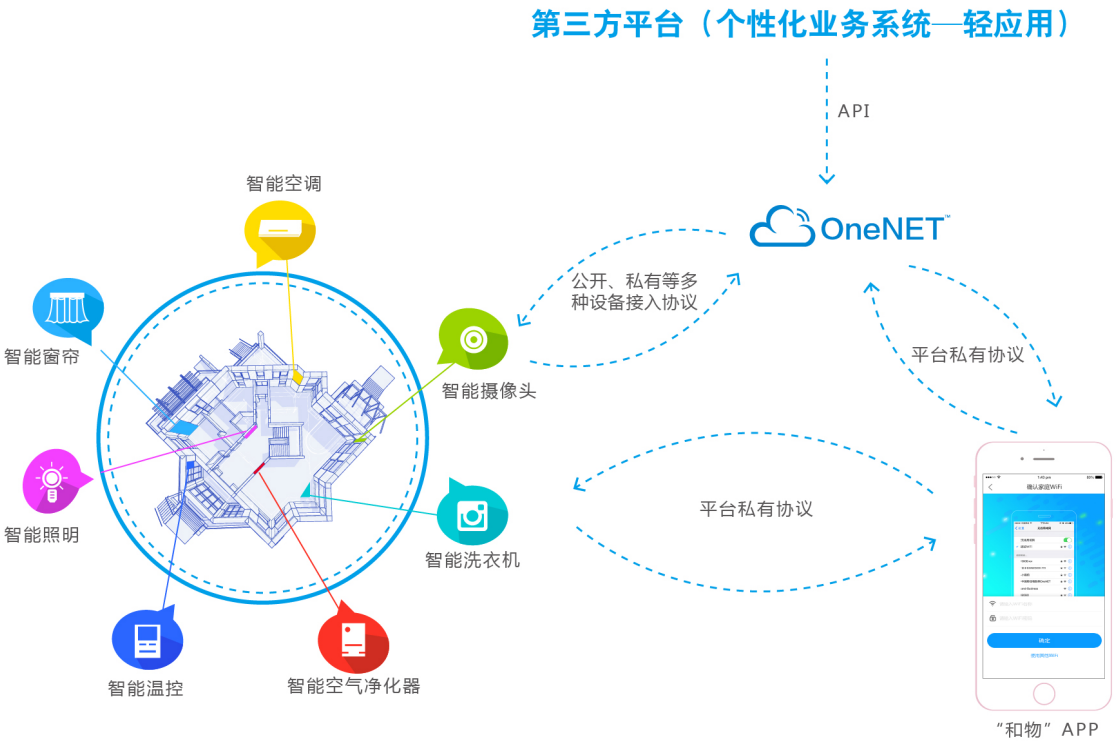
OneNET与MQTT

董峦 新疆农业大学 2018

一、中移动OneNET

OneNET（open.iot.10086.cn）是中国移动物联网有限公司面向公共服务自主研发的开放云平台。该平台为各种跨平台物联网应用、行业解决方案提供简便的海量连接、云端存储、消息分发和大数据分析等优质服务，从而降低物联网企业和个人（创客）的研发、运营和运维成本，使物联网企业和个人（创客）更加专注于应用，共建以OneNET为中心的物联网生态环境。

其使用方式如下图所示



即OneNET作为智能设备、移动APP和第三方平台的中枢，提供一整套通信和存储服务，降低开发物联网应用的门槛，同时提高了系统运行的可靠性。

让我们用一个例子理解。想象这样一个场景：我们用手机APP（中移动的“和物”或自己开发的APP）发现家里的空调没有关，然后我们在手机界面上点了“关闭空调”字样的按钮，该信息以某种协议发送到OneNET平台，OneNET平台随后将该信息下发至相对应的智能空调这个设备，智能空调设备上的程序接收到该信息后关闭了空调，同时将空调已关闭的状态返回OneNET平台，OneNET平台再将该信息返回给手机APP，从而让人确认设备关了。第三方平台指什么呢，比如我们对OneNET提供的数据展示界面不满意，那么可以自己开发Web应用从OneNET平台取得原始数据然后按自己的偏好展示；或者我们不想开发手机APP，而是用微信公众平台访问设备，那么微信公众平台就是图示中的“第三方平台”。

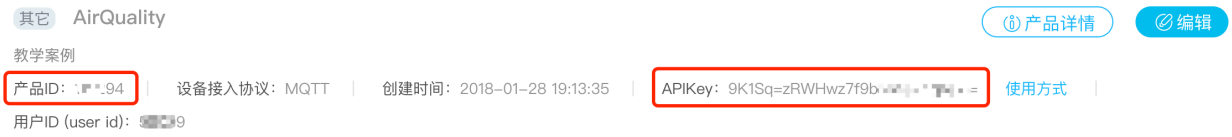
总而言之，当万物互联后，操控硬件便突破地域限制了。

二、使用OneNET

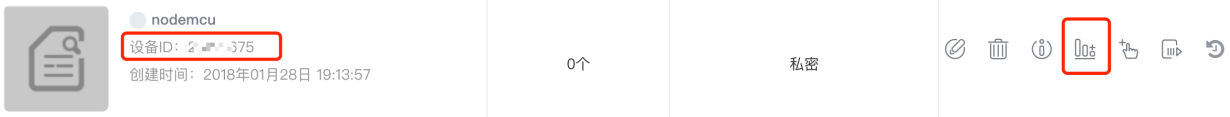
在OneNET文档中心的快速开始页面https://open.iot.10086.cn/doc/art243.html#66 根据介绍完成用户注册和产品创建的工作。在术语介绍页面https://open.iot.10086.cn/doc/art246.html#68 了解该平台所使用术语的含义，比如“产品”指：用户手上的真实设备在OneNET上对应的虚拟名称，是用户在OneNET上最上层的云端资源。创建接入方式是MQTT的一个产品，我的一个名为“AirQuality'”产品如下图所示

创建该产品时的技术参数如下图所示

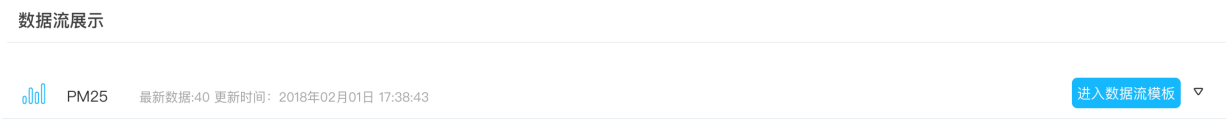
点击进入该产品配置页面，记住其产品ID和APIKey，这两个信息相当于访问产品时的用户名和密码。



接着添加设备，我添加了一个名为“nodemcu”的设备，如下图所示，记住其设备ID，当我们与这个虚拟设备连接时需要该ID。



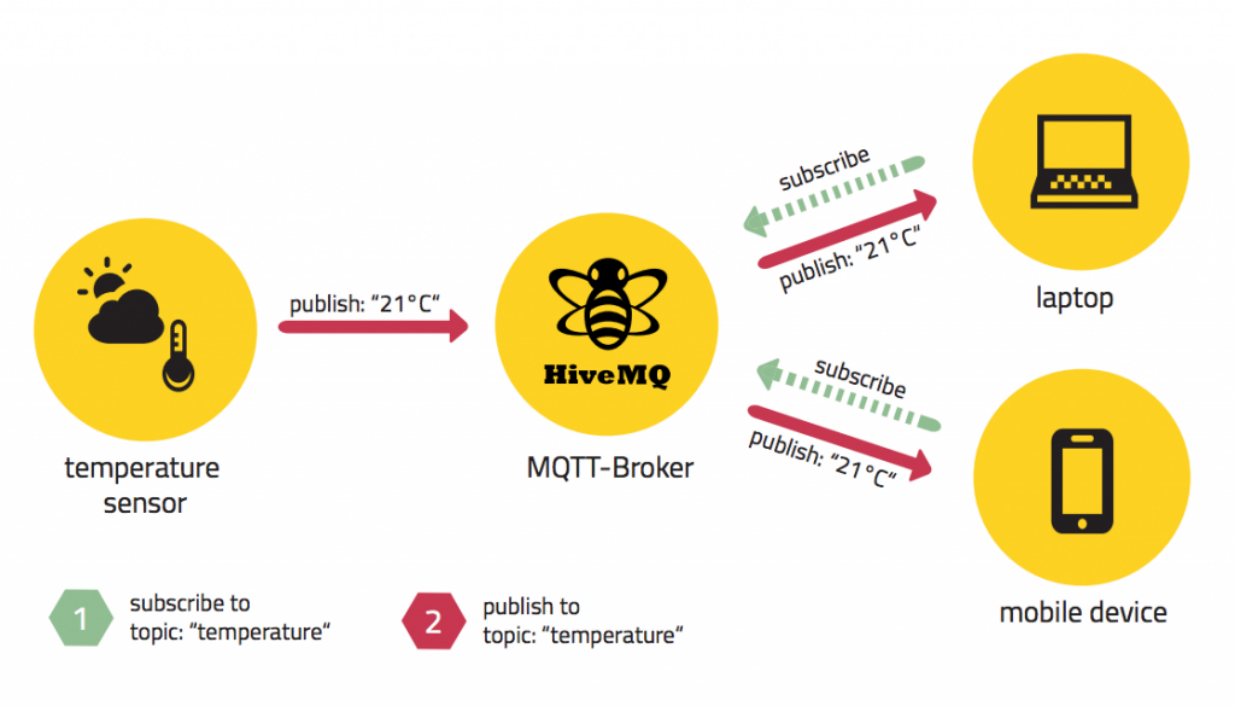
点击上图右侧的“数据流管理”图标，添加一个名为“PM25”的数据流，你也可以取其它名字，但在后面编码时要相应修改。



下面我们就可以利用MQTT这个协议向OneNET发送数据了。

三、MQTT

OneNET文档中心有一份关于MQTT协议的简要说明https://open.iot.10086.cn/doc/art253.html#68，在该页面后面有一份MQTT OneNET实现的文档。关于MQTT协议的细节，可参考这份文档https://mcxiaoke.gitbooks.io/mqtt-cn/content/。MQTT协议由IBM开发，目前是物联网应用中传递消息的主流协议。下面这个图很好的说明了MQTT协议的特点



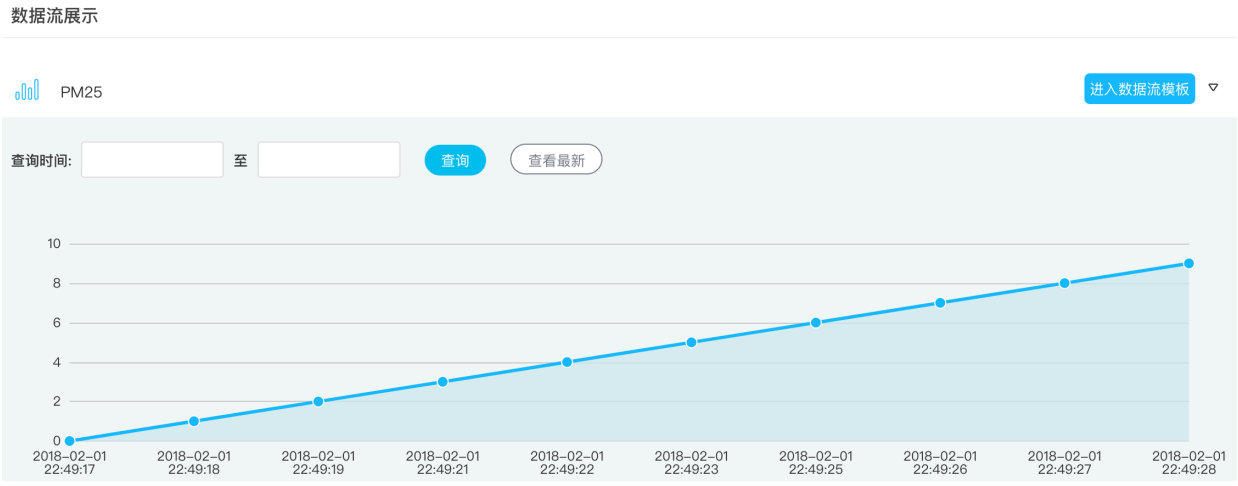
该协议中的“Broker”可以理解为像OneNET或上图中HiveMQ这样的平台，接入平台的设备与其交互有两种方式，分别是“publish”和“subscribe”，即发布和订阅。带有传感器的设备向平台发送信息可以叫“publish”，移动终端如果在平台上“subscribe”了特定主题（topic）的内容，则可以收到相应内容，此时平台向移动终端发信息也可以称为“publish”。上图中传递消息的主题是“temperature”，当传感器把温度信息发送到平台上时，由于移动设备订阅了该主题的消息，则它们可以收到平台下发的关于温度的消息。

下面我们以OneNET平台接受的一种数据格式为例说明向该平台发送消息的方式。在“MQTT OneNET实现”这份文档第17页描述了类型3这种消息的格式，该消息第一个字节值为3，表示传递的是类型3消息；第2和第3个字节指示后面要传递的JSON字符串的长度；从第4个字节开始是JSON字符串长度。我们看下面这段代码

```
import time
from umqtt import MQTTClient

SERVER = '183.230.40.39'    # OneNET MQTT服务器
PORT = 6002                 # OneNET MQTT服务端口
CLIENT_ID = 'xxxxxxxx'      # 设备ID，测试该代码时请该为你的设备ID
```

```
TOPIC = b'$dp'              # 数据点上报的固定主题

username = 'xxxxxx'                          # 产品ID，测试该代码时请该为你的产品ID
password = '9K1Sq=zRWHwz7f9b5hejwKxxxxxx' # APIKey，测试该代码时请改为你的APIKey

def construct_msg(raw_msg):
    msglen = len(raw_msg)
    t = [0,0,0]
    t[0] = '\x03'            # 类型3 消息，用16进制数表示
    t[1] = msglen >> 8       # 把 msglen 的高位字节取出来（右移8位来实现），MQTT的消息长度最多是65536，所以这里 msglen 的有效位数是16位
    t[2] = msglen & 0xFF     # 把 msglen 的低位字节取出来（与0xFF相与来实现）
    return "%c%c%c%s" % (t[0], t[1], t[2], raw_msg)  # 按OneNET要求的格式拼出消息，即第一个字节是类型号，
    # 第二和第三个字节是后面JSON字符串的长度，从第四个字节开始是JSON字符串


c = MQTTClient(CLIENT_ID, SERVER, PORT, username, password)

for i in range(10):
    c.connect()
    raw_msg = "{\"PM25\" : %d}" % (i,)   # JSON字符串是键值对格式，在这里键是"PM25"，是消息的topic，
     # 值依次取0到9的整数
    c.publish(TOPIC, construct_msg(raw_msg))
    c.disconnect()
    time.sleep(1)
```

上述代码的注释已经很详细的解释了关键语句的含义，请仔细理解该段代码以便在此次项目中向平台发送实测的PM2.5数据。从平台上看，我们的数据发送上去了。

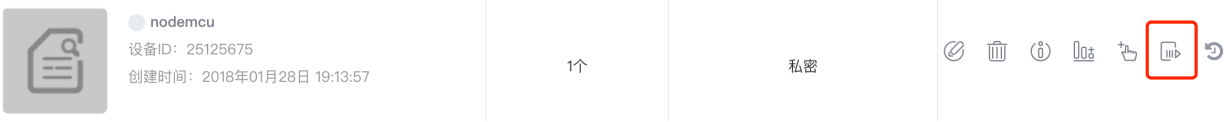


下面我们看看如何从平台给设备下达指令。我们要达成的目标是在平台下达指令（on 或 off）以便控制NodeMCU上蓝色LED的亮或灭。

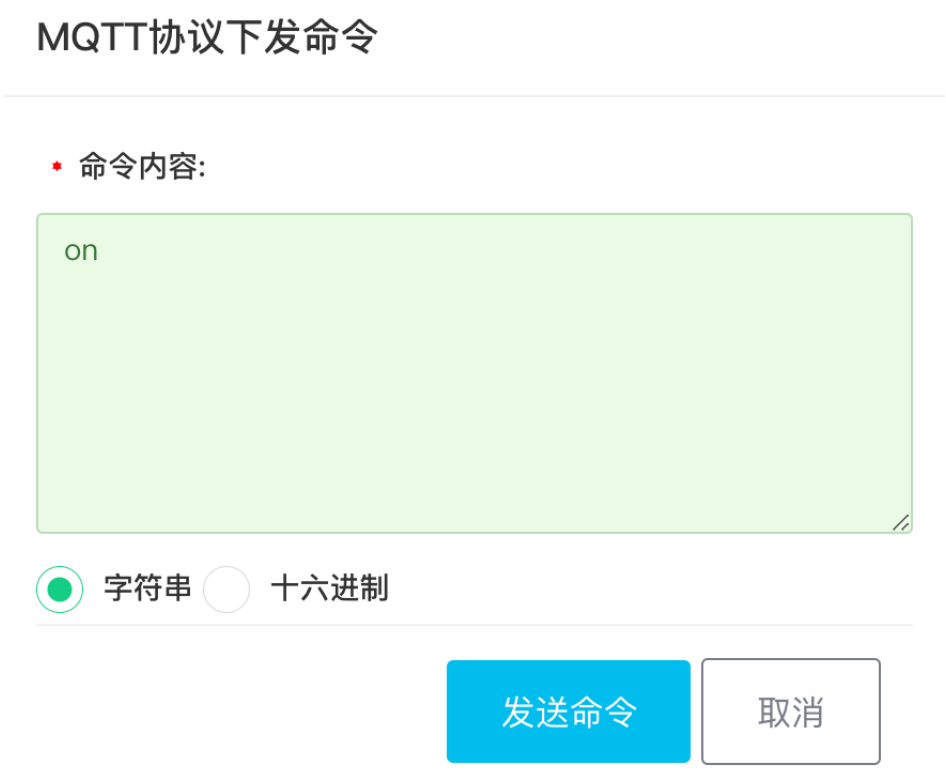
```
import time
from umqtt import MQTTClient
from machine import Pin
```

```
SERVER = '183.230.40.39'    # OneNET MQTT服务器
PORT = 6002                 # OneNET MQTT服务端口
CLIENT_ID = '25xxxxxx'      # 设备ID，需换成实际设备ID
TOPIC = b'led'              # 经测试OneNET平台下发的指令无法自定义主题，故这里可随意设置

username = 'xxxxxx'                          # 产品ID，需换成实际产品ID
password = '9K1Sq=zRWHwz7f9b5hejwKxxxxxx' # APIKey，需换成实际APIKey

led = Pin(16, Pin.OUT)
state = 1

def sub_cb(topic, msg):
    global state
    print(topic, msg)     # 借此观察平台下发指令的主题和内容
    if msg == b"on":
        led.off()         # 因为 NodeMCU 板上的LED阴极接GPIO16，所以这里将端口电平拉低灯才会亮
    elif msg == b"off":
        led.on()
    elif msg == b"end":  # 如果平台下发"end"消息，则结束下面的 while 循环。用修改全局变量 state 实现
        state = 0


c = MQTTClient(CLIENT_ID, SERVER, PORT, username, password)
c.set_callback(sub_cb)  # 指定处理订阅消息的回调函数
c.connect()
c.subscribe(TOPIC)

try:
    while state:
        c.wait_msg()
finally:
    c.disconnect()
```

将上述代码写入NodeMCU的 main.py 文件，然后在OneNET平台下发指令，如下图所示



下达如下指令后，可以观察到NodeMCU上的LED亮了



然后依次发送 off 和 end 指令，可以看到终端上显示的结果

```
b'$creq/171bef35-486d-5f2f-9c73-f19b1c1e3659' b'on'
b'$creq/bca11816-a6a6-5586-8d05-82a23d3f3222' b'off'
b'$creq/5272f7b2-0bb0-5379-ae7d-0657ee477d3e' b'end'

MicroPython v1.9.3-238-g42c4dd09 on 2017-12-30; ESP module with ESP8266
Type "help()" for more information.
>>>
```

在上图中可以看到回调函数打印的信息。平台下发的指令消息其主题是 b'$creq/cmduuid'，cmduuid为该条指令的uuid；内容是我们输入的命令内容。收到 end 指令后，上述代码执行完毕。

这里有必要介绍MQTT中的主题（Topic）概念，主题是一个UTF8字符串，被Broker用来组织/过滤与客户端通信的内容，一个主题由多个层级构成，每一级由符号 '/' 隔开，例如

下面都是有效的主题标识

```
myhome/groundfloor/livingroom/temperature
USA/California/San Francisco/Silicon Valley
5ff4a2ce-e485-40f4-826c-b1a5d81be9b6/status
Germany/Bavaria/car/2382340923453/latitude
```

注意，主题中区分大小写，允许使用空格，但实践中最好不要使用空格，并保持尽量简短。

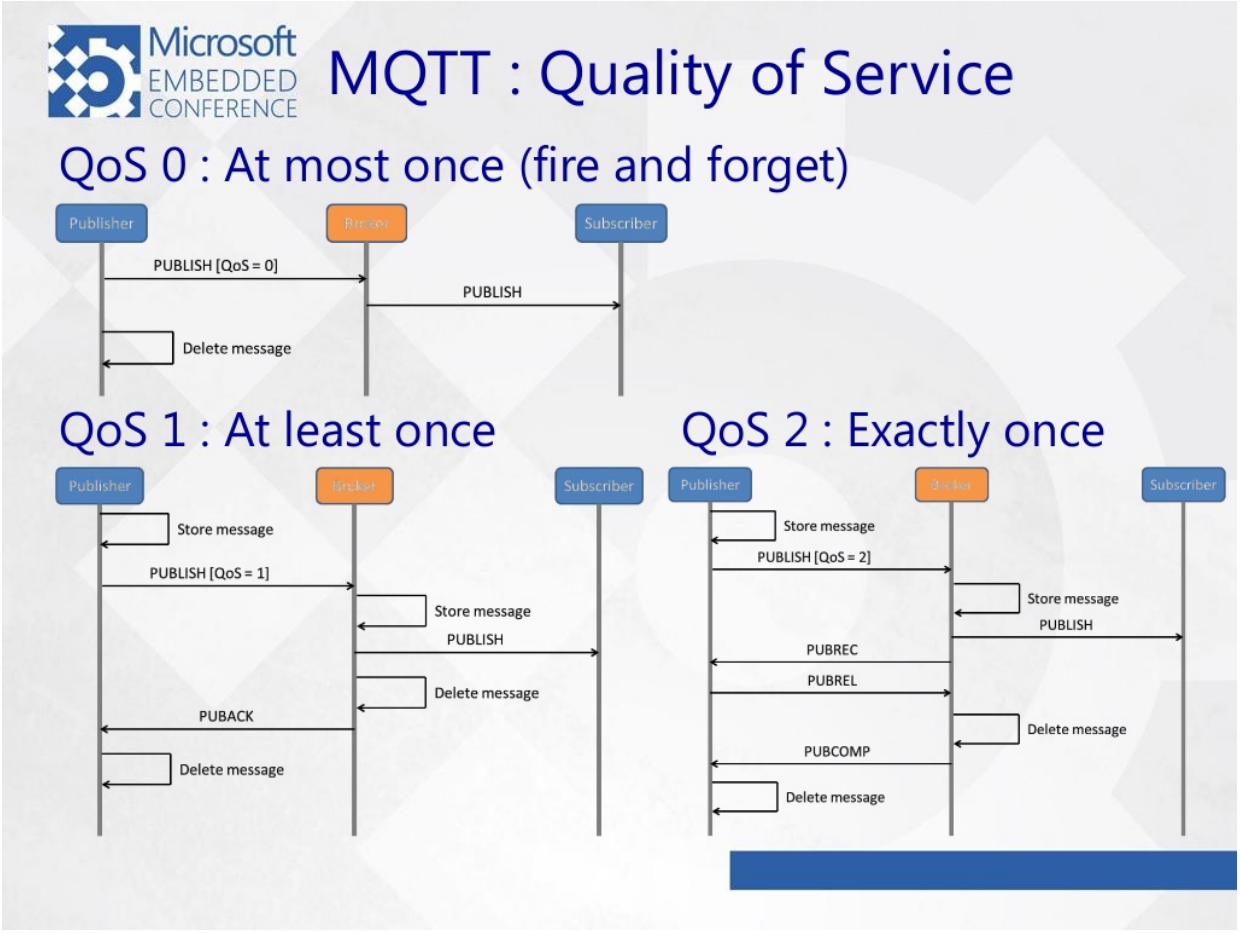
当客户端订阅主题时，可以使用通配符，以便一次订阅多个主题。可以使用的通配符有 '+' 和 '#' ，'+' 号是单级通配符，如下图所示：

'#' 号是多级通配符，如下图所示：

该通配符因为统配任意层级数的主题，所以只能位于主题字符串的最后。如果订阅的主题是'#'，则将收到Broke发送的所有消息。

虽然主题可以任意命名，但以 '$' 符号开头的主题一般是为Broke保留的，比如OneNET实现的MQTT中使用的 $dp、$creq 等主题，OneNET不支持订阅 $ 开头的系统主题。

在上述向OneNET发送数据的例子中，我们在publish函数中并未指定QoS（Quality of Service）等级，根据我们导入的umqtt包的文档，其发布消息的默认QoS是0级。QoS是消息发送方和接收方之间关于信息送达程度的一种约定，MQTT中有三个QoS等级，如下图所描述：



从该示意图不难理解，QoS 0代表着什么了。QoS等级越高，通信的成本就越高（更多的确认过程），所以应根据消息的重要程度选择合适的QoS等级。

MQTT协议通过主题的订阅、发布以及消息推送，可以实现设备间的消息单播以及组播，即实现嵌入式设备间通信，这部分内容与本次项目无关，这里不再介绍，感兴趣的同学可以去进一步了解这方面内容。以上是OneNET平台和MQTT协议的有关内容，相信你通过亲手实践后能够掌握与云平台通信的方法。