

智慧农场

虚谷号和物联网实验

博识教育

目 录

1. 消息发送实验

1.1 实验目的.....	01
1.2 实验描述.....	01
1.3 实验指导.....	01
1.3.1 电子模块连接	
1.3.2 MQTT 简介	
1.3.3. SIIoT 简介	
1.3.4. SIIoT 服务器配置说明	
1.4 实验步骤.....	04
1.4.1 连接 SIIoT 服务器并给“vv/01”发送数据	
1.4.2 登录 SIIoT 服务器 Web 管理界面，查看消息	
1.4.3 定时发送多条数据，并查看消息	
1.5 实验拓展.....	10
1.6 知识链接.....	10

2. 消息订阅实验

2.1 实验目的.....	12
2.2 实验描述.....	12
2.3 实验指导.....	12
2.3.1 MQTT 协议的核心名词	
2.3.2 常见的 MQTT 客户端（PC 端）	
2.4 实验步骤.....	15
2.4.1 连接 SIIoT 服务器并订阅 TopicID	
2.4.2 通过其他客户端发送消息，测试订阅结果	
2.5 实验拓展.....	22
2.6 知识链接.....	22

3. 互动控制实验

3.1 实验目的.....	23
3.2 实验描述.....	23
3.3 实验指导---电子模块连接.....	23
3.4 实验步骤.....	24
3.4.1 订阅消息“vv/01”	
3.4.2 按下按钮发送消息“ok”到“vv/02”	

3.5 实验拓展.....	31
3.6 知识链接.....	32

4. WebAPI 调用实验

4.1 实验目的.....	33
4.2 实验描述.....	33
4.3 实验指导.....	33
4.4 实验步骤.....	34
4.4.1 通过 WebAPI 发送消息	
4.4.2 通过 WebAPI 订阅消息	
4.5 实验拓展.....	39
4.6 知识链接.....	40

5. 物联网数据可视化实验

5.1 实验目的.....	42
5.2 实验描述.....	42
5.3 实验指导.....	42
5.3.1 电子模块连接	
5.3.2 温湿度传感器数据的采集与处理	
5.3.3 Node-RED 简介	
5.4 实验步骤.....	50
5.4.1 订阅数据，并实时画出折线图	
5.4.2 订阅数据，并实时画出仪表盘	
5.5 实验拓展.....	60
5.6 知识链接.....	60

6. 多终端互联实验

6.1 实验目的.....	61
6.2 实验描述.....	61
6.3 实验指导.....	61
6.3.1 电子模块连接	
6.3.2 终端定义	
6.4 实验步骤.....	62
6.5 实验拓展.....	69
6.6 知识链接--- 物联网的十大应用领域.....	70

1. 消息发送实验

1.1 实验目的

- 了解物联网的基本组成、了解 SIoT 软件、MQTT 通讯协议。
- 学会使用 SIoT 完成消息发送。

1.2 实验描述

- 连接 SIoT 服务器并给 “vv/01” 发送数据
- 登录 SIoT 服务器 Web 管理界面，查看消息
- 定时发送多条数据，并查看消息

1.3 实验指导

1.3.1 电子模块连接

硬件：虚谷号、Arduino 扩展板、光线传感器

将光线传感器通过 arduino 扩展板接到虚谷号板载 Arduino 的 A0 口。



1.3.2 MQTT 简介

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, 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 传输的消息分为 Topic 和 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 为消息的内容, 是指订阅者具体要使用的内容。

1.3.3. SIoT 简介

SIoT 为一个为教育定制的跨平台的开源 MQTT 服务器程序,S 指科学 (Science)、简单 (simple) 的意思。SIoT 支持 Win10、Win7、Mac、Linux 等操作系统, 一键启动, 无需用户注册或者系统设置即可使用。

SIoT 为“虚谷物联”项目的核心软件, 是为了帮助中小学生学习理解物联网原理, 并且能够基于物联网技术开发各种创意应用。因为其重点关注物联网数据的收集和导出, 是采集科学数据的最好选择之一。

1.3.4. SIoT 服务器配置说明

SERVER = "127.0.0.1": 设置 MQTT 服务器 IP, 虚谷号用 127.0.01 表示本机。

CLIENT_ID = "" : 在 SIoT 上, CLIENT_ID 可以留空不填写。

IOT_pubTopic = 'vv/01': “topic”为“项目名称/设备名称”, topicid 自己定义, 此处用的是 'vv/01', 表示项目名称为“vv”, 设备名称为“01”。

IOT_UserName 和 IOT_PassWord 分别表示用户名和密码, 统一使用 “scope”。

1.4 实验步骤

1.4.1 连接 SIoT 服务器并给 “vv/01” 发送数据

步骤 1: 在无线模式下打开 jupyter, 新建一个文件夹, 命名为: 物联网实验。

步骤 2: 在该文件目录下, 打开 jupyter, 输入如下代码并执行。

```
"""导入库"""
import siot
from xugu import Pin

"""配置 SIOT 服务器"""
SERVER = "127.0.0.1"
```

```

CLIENT_ID = ""
IOT_pubTopic = 'vv/01'
IOT_UserName = 'scope'
IOT_PassWord = 'scope'

"""连接 SIoT 服务器"""
siot.init(CLIENT_ID, SERVER, user = IOT_UserName, password = IOT_PassWord)
siot.connect()

"""初始化模拟传感器"""
p1 = Pin("A0", Pin.ANALOG) # 初始化 A0 引脚, 设置为输入模式

"""获取传感器数据, 并发送 MQTT 消息"""
value = p1.read_analog()
print(value)
siot.publish(IOT_pubTopic, "%d"%value)

```

【实验结果】

将看到打印出传感器的模拟值数据, 如图所示。

注意: “vv/01” 这个 Topic 不需要事先设置, 只要发过一次消息, SIoT 会根据这一 Topic 自动建立项目名称和设备名称, 方便管理。

1.4.2 登录 SIoT 服务器 Web 管理界面, 查看消息

步骤 1: 打开浏览器, 在网址栏中输入: <http://【虚谷号 IP】:8080>, 打开 SIoT 网页平台管理页面。

SIoT	项目列表	设备列表	发送消息
-------------	------	------	------

帐号

密码

登陆

步骤 2: 输入用户名及密码, 都是 “scope”, 点击登录。

SlIoT

项目列表

设备列表

发送消息

主题 (项目ID/设备名)

项目ID

100条 ▾

查询

项目ID	备注	操作
DFRobot		查看设备列表 添加备注 删除
DIY		查看设备列表 添加备注 删除
xzr		查看设备列表 添加备注 删除
vv		查看设备列表 添加备注 删除
zwl		查看设备列表 添加备注 删除

步骤 3: 点击 “设备列表”, 选择项目 ID “vv”, 点击对应的绿色文字 “查看消息”

【实验结果】

将看到上述程序执行后的数据消息, 如图所示。

SlIoT

项目列表

设备列表

发送消息

当前主题: vv/01

发送消息

消息内容

发送

(为消息加上->前缀代表此消息为纯指令消息, 不会被存入数据库。例如“->off”)

开始时间

结束时间

100条 ▾

查询

导出查询结果

隐藏/显示图表

自动刷新消息 ☐

Topic	消息	时间
vv/01	297	2020-06-08 11:11:49

注意: 客户端发送的消息, 所有订阅了这一主题的客户端都能收到, 包括自己。

1.4.3 定时发送多条数据, 并查看消息

打开 jupyter, 输入如下代码并执行。

```
"""导入库"""
import siot
import time
from xugu import Pin

"""配置 SIOT 服务器"""
SERVER = "127.0.0.1"
CLIENT_ID = ""
IOT_pubTopic = 'vv/01'
IOT_UserName = 'scope'
IOT_PassWord = 'scope'
```

```
"""连接 SIOT 服务器"""
```

```
siot.init(CLIENT_ID, SERVER, user = IOT_UserName, password = IOT_PassWord)
```

```
siot.connect()
```

```
"""初始化模拟传感器"""
```

```
p1 = Pin("A0", Pin.ANALOG)
```

```
"""获取传感器数据，并发送 MQTT 消息"""
```

```
while True:
```

```
    value = p1.read_analog()
```

```
    print(value)
```

```
    siot.publish(IOT_pubTopic, "%d"%value)
```

```
    time.sleep(1)
```

【实验结果】

将看到程序循环打印出一系列传感器的模拟数据值，如图所示。

```
590  
584  
586  
618  
742  
739  
620  
608  
614  
736  
716  
719  
711  
517  
475
```


按照 2 中方法打开 SIIoT 服务器 Web 界面可以查看到相同的数据值，只是排列方式不同，如图。

Topic	消息	时间
vv/01	475	2020-06-09 09:25:50
vv/01	517	2020-06-09 09:25:49
vv/01	711	2020-06-09 09:25:48
vv/01	719	2020-06-09 09:25:47
vv/01	716	2020-06-09 09:25:45
vv/01	736	2020-06-09 09:25:44
vv/01	614	2020-06-09 09:25:43
vv/01	608	2020-06-09 09:25:42
vv/01	620	2020-06-09 09:25:41
vv/01	739	2020-06-09 09:25:40
vv/01	742	2020-06-09 09:25:39
vv/01	618	2020-06-09 09:25:38
vv/01	586	2020-06-09 09:25:37
vv/01	584	2020-06-09 09:25:36
vv/01	590	2020-06-09 09:25:34

1.5 实验拓展

循环发送光线传感器数据，并下载数据绘图。

展示你的实验成果，并且填写评价表。

评价内容	评价结果	备注
连接 SIIoT 服务器并给“vv/01”发送数据	☆☆☆☆☆	
登录 SIIoT 服务器 Web 管理界面，查看消息	☆☆☆☆☆	
定时发送多条数据，并查看消息	☆☆☆☆☆	

1.6 知识链接

物联网（Internet of Things，简称 IoT）是借助互联网、传统电信网等信息承载体，让所有能行使独立功能的普通物体实现互联互通的网络。在物联网上，每个人都可以应用电子标签

将真实的物体上网联结，在物联网上都可以查出它们的具体位置。通过物联网可以用中心计算机对机器、设备、人员进行集中管理、控制，也可以对家庭设备、汽车进行遥控，以及搜索位置、防止物品被盗等，类似自动化操控系统，同时通过收集这些小数据，最后聚集成大数据，包含重新设计道路以减少车祸、都市更新、灾害预测与犯罪防治、流行病控制等等社会的重大改变，实现物和物相联。

物联网将现实世界数字化，应用范围十分广泛。物联网拉近分散的信息，统整物与物的数字信息，物联网的应用领域主要包括以下方面：运输和物流领域、工业制造、健康医疗领域范围、智能环境（家庭、办公、工厂）领域、个人和社会领域等，具有十分广阔的市场和应用前景。



2. 消息订阅实验

2.1 实验目的

- 了解 MQTT 的核心内容
- 了解并学会用 MQTT.fx 客户端测试订阅结果
- 学会使用 SIoT 完成消息订阅

2.2 实验描述

- 连接 SIoT 服务器并订阅 TopicTD，通过 Web 页面发送消息，测试订阅结果
- 通过其他客户端发送消息，测试订阅结果

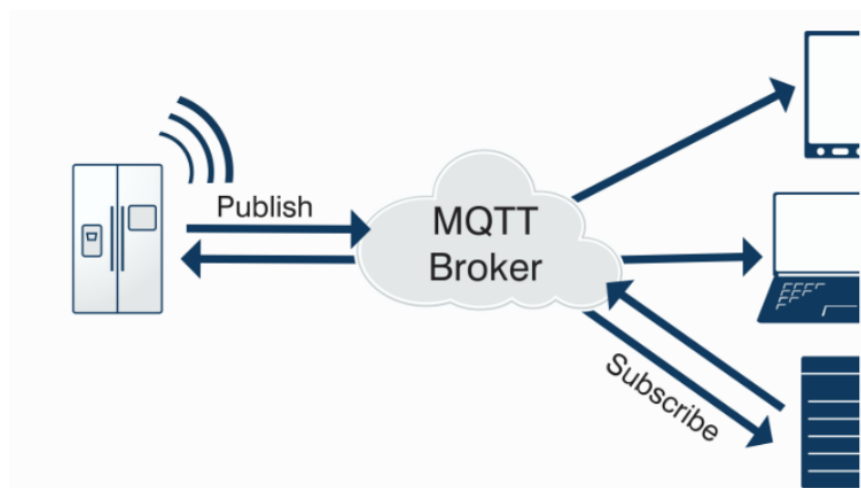
2.3 实验指导

2.3.1 MQTT 协议的核心名词

订阅 (Subscription)：订阅包含主题筛选器 (Topic Filter) 和最大服务质量 (QoS)。订阅会与一个会话 (Session) 关联。一个会话可以包含多个订阅。每一个会话中的每个订阅都有一个不同的主题筛选器。

会话 (Session)：每个客户端与服务器建立连接后就是一个会话，客户端和服务端之间有状态交互。会话存在于一个网络之间，也可能在客户端和服务端之间跨越多个连续的网络连接。

主题名 (Topic Name)：连接到一个应用程序消息的标签，该标签与服务器的订阅相匹配。服务器会将消息发送给订阅所匹配标签的每个客户端。



2.3.2 常见的 MQTT 客户端 (PC 端)

MQTTfx 是一款很好用的 MQTT 客户端调试工具，支持在 Windows、Mac 和 Linux 上面运行。设备将当前所处的状态作为 MQTT 主题发送给 IoT Hub，每个 MQTT 主题 topic 具有不同等级的名称，如“建筑/楼层/温度。” MQTT 代理服务器将接收到的主题 topic 发送给给所有订阅的客户端。MQTT.fx 支持 windows/linux/mac，请选择对应的版本进行安装，这里以 Windows 为例，版本为 1.7.1 。

下载地址：<http://mqttfx.jensd.de/index.php/download>



下载完之后，双击进行安装。

注意：安装完成后，运行时会提示有更新，不要点击 yes，会报错。

2.4 实验步骤

2.4.1 连接 SIoT 服务器并订阅 TopicID，通过 Web 页面发送消息，测试订阅结果

步骤 1: 在无线模式下打开 jupyter, 打开“物联网实验”文件夹。

在该文件目录下, 打开 jupyter, 输入如下代码并执行。

```
""" 导入库 """
import siot

""" 配置 SIOT 服务器 """
SERVER = "127.0.0.1"
CLIENT_ID = ""
IOT_pubTopic = 'vv/02'
IOT_UserName = 'scope'
IOT_PassWord = 'scope'

""" 连接 SIOT 服务器 """
siot.init(CLIENT_ID, SERVER, user = IOT_UserName, password = IOT_PassWord)
siot.connect()

""" 定义收到消息时的提示消息 """
def sub_cb(client, userdata, msg):
    print("\nTopic:" + str(msg.topic) + " Message:" + str(msg.payload))

""" 订阅消息 """
siot.subscribe(IOT_pubTopic, sub_cb)
siot.loop()
```

步骤 2: 打开浏览器, 在网址栏中输入: <http://【虚谷号 IP】:8080>, 进入 SIoT 服务器的 Web 管理界面, 输入用户名和密码“scope”, 点击登录进入。点击“发送消息”。

SIoT	项目列表	设备列表	发送消息
主题 (项目ID/设备名)			
<input type="text" value="项目ID/设备名"/>			
消息			
<input type="text" value="消息"/>			
(为消息加上->前缀代表此消息为纯指令消息, 不会被存入数据库。例如->off)			
<input type="button" value="发送"/>			

步骤 3: 在主题对话框中输入: vv/02, 消息对话框中输入: -> off, 点击发送。

SlOT	项目列表	设备列表	发送消息
主题（项目ID/设备名）			
<input type="text" value="vv/02"/>			
消息			
<input type="text" value="->off"/>			
(为消息加上->前缀代表此消息为纯指令消息，不会被存入数据库。例如"->off")			
<input type="button" value="发送"/>			

【实验结果】

将看到执行程序时，程序输出：“连接结果，连接成功”字样，如图所示。

连接结果：连接成功

发送消息后，被发送的消息被程序接收并显示出来，如图。

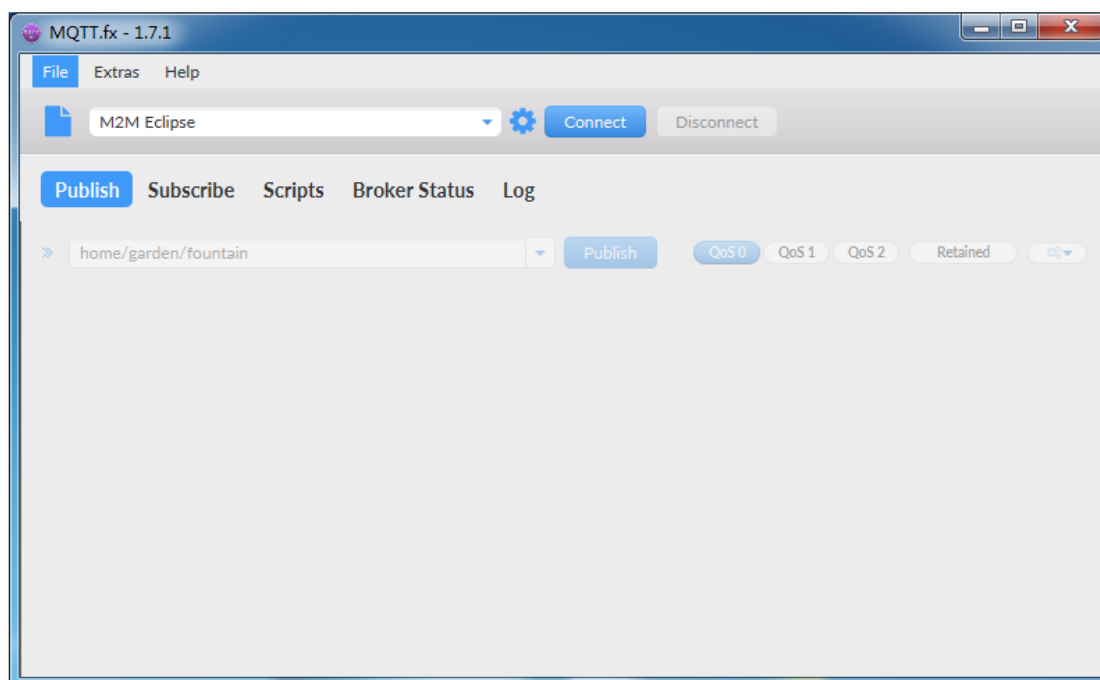
连接结果：连接成功

Topic:vv/02 Message:b'-> off'

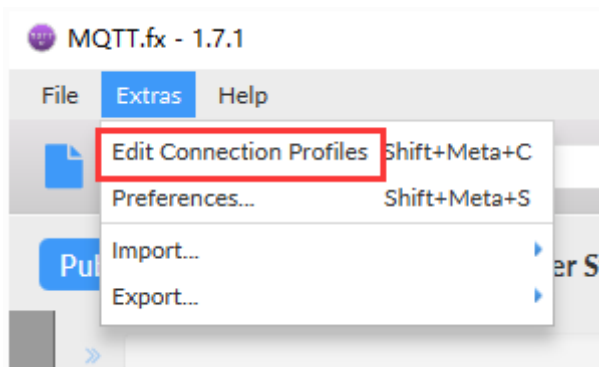
注意：通过 Web 给 Topic 发送消息，如果不想将这一消息记录在数据库中，可以在前面加上“->”的标志。

2.4.2 通过其他客户端发送消息，测试订阅结果

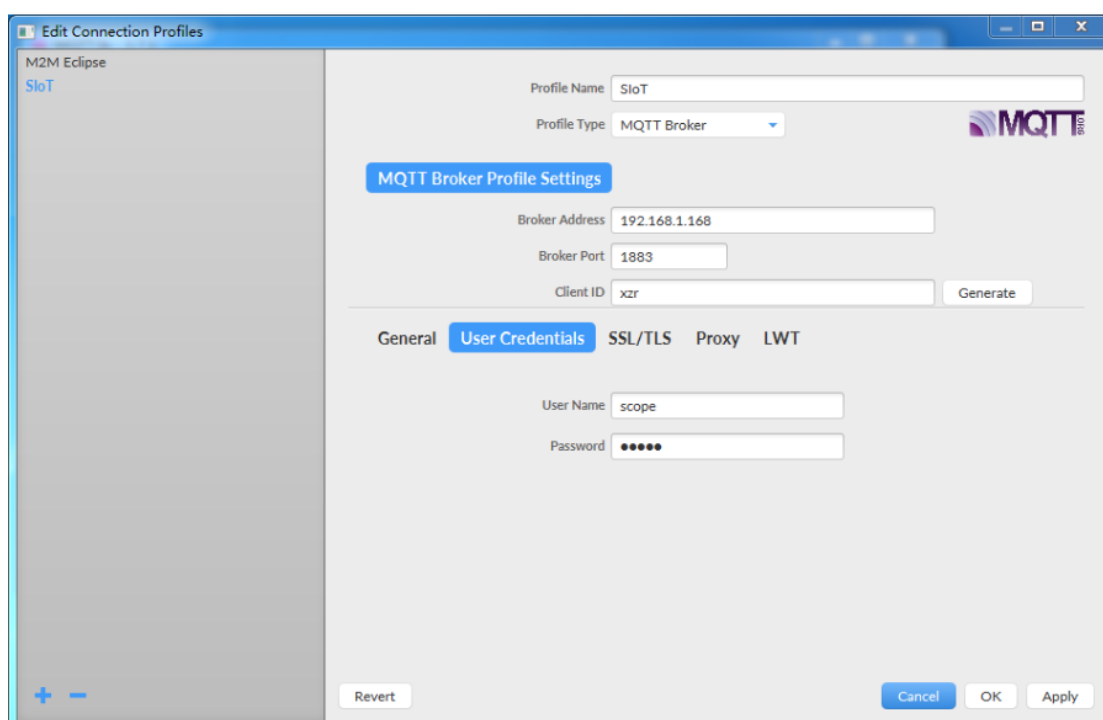
步骤 1：点击电脑“开始”菜单，找到 MQTTfx 打开软件，界面如下。



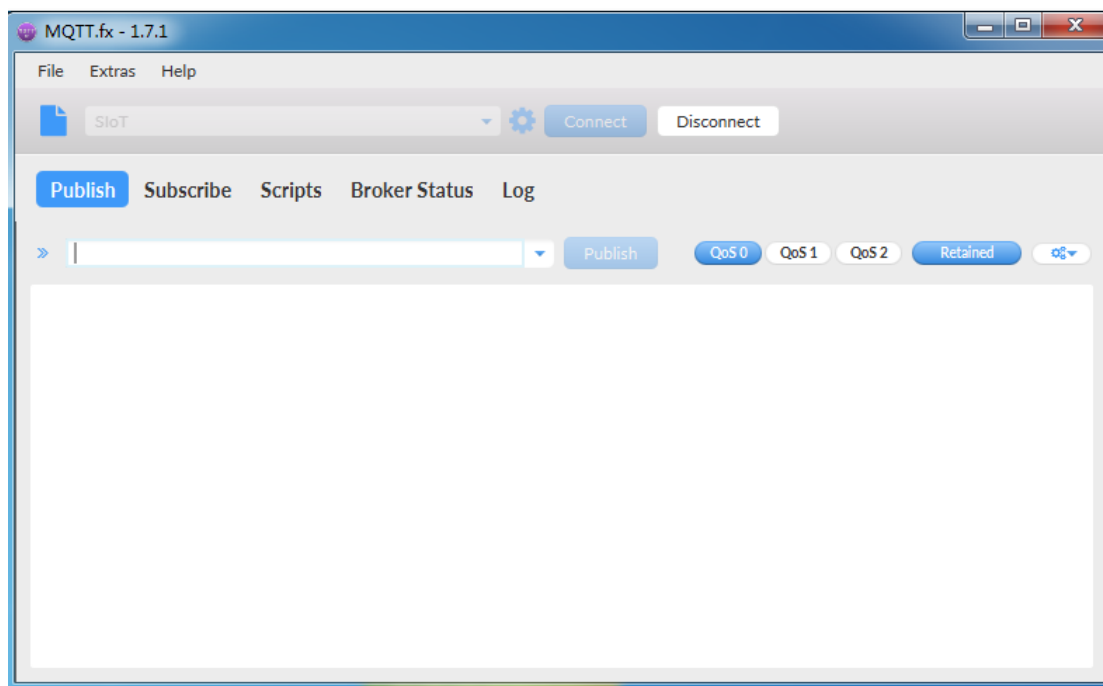
步骤 2: 选择编辑连接, Extras->Edit Connection Profiles, 如下。



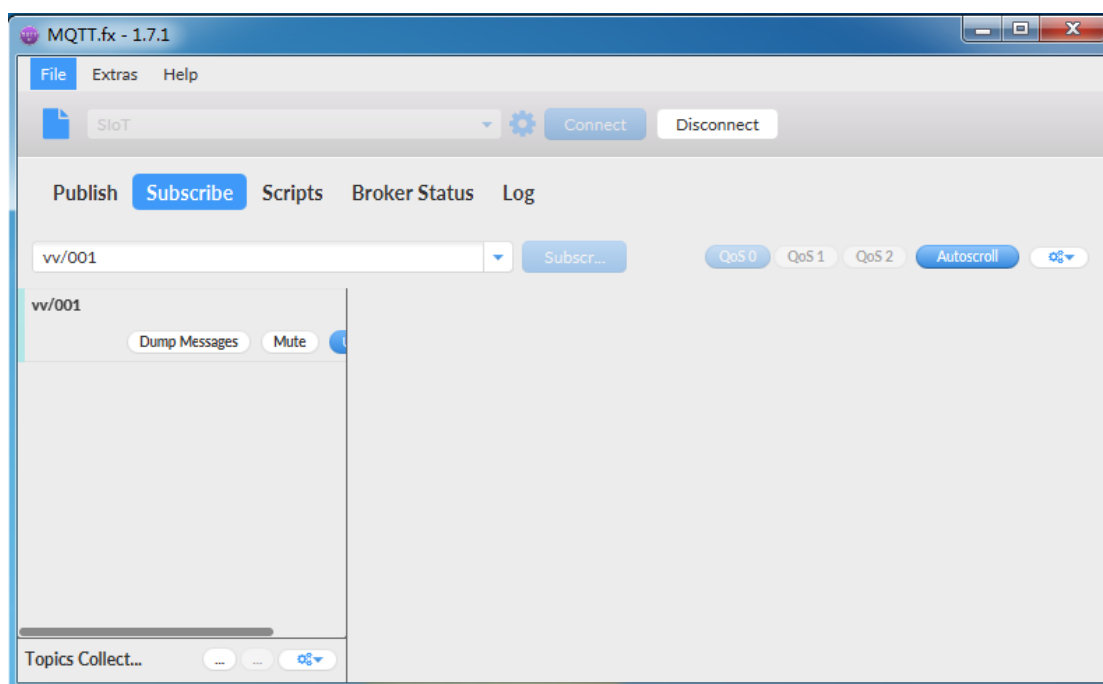
步骤 3: 填写相应的 Profile Name, Broker Address 和 Broker Port (如果修改过的话, 默认是 1883) 等内容, Client ID 可以点击 Generate 按钮自动生成, 用户名和密码都是 “scope”, 如图所示。编辑完之后点击 “OK” 保存退出编辑界面。



步骤 4：到主界面的下拉框选择刚才配置的 Profile Name 名称（SIoT），然后点击 Connect（连接）按钮进行服务连接。

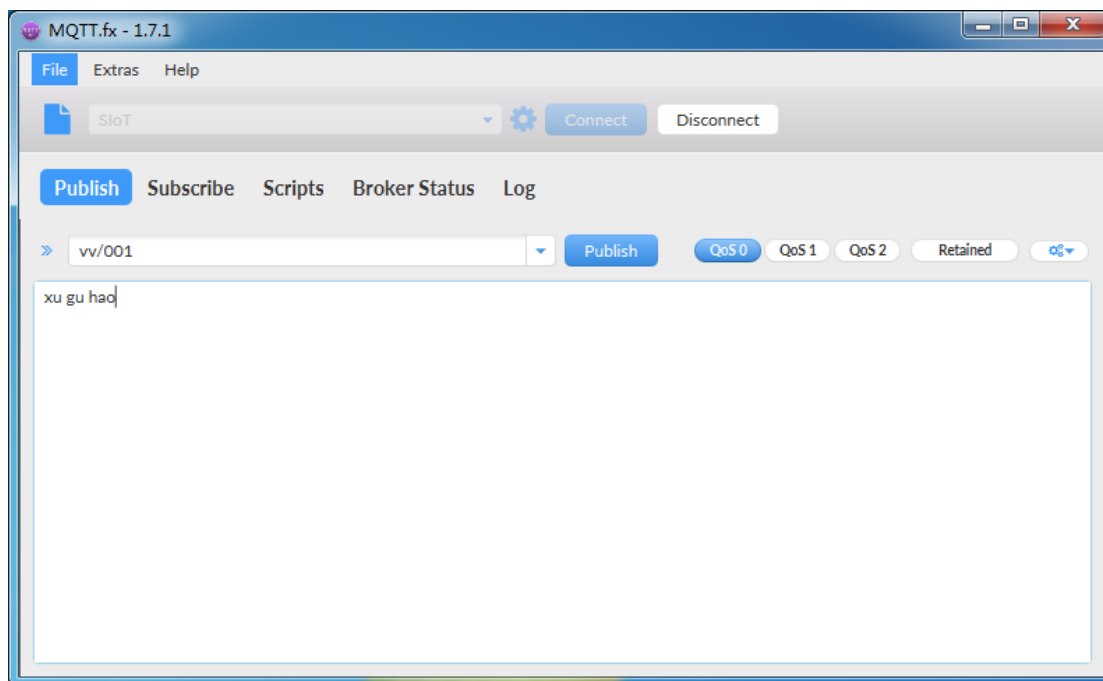


步骤 5：连接成功后，再点击 Subscribe（订阅）选项，在对话框中创建一个主题，如 vv/001，之后点击后面的 Subscribe（订阅）按钮。



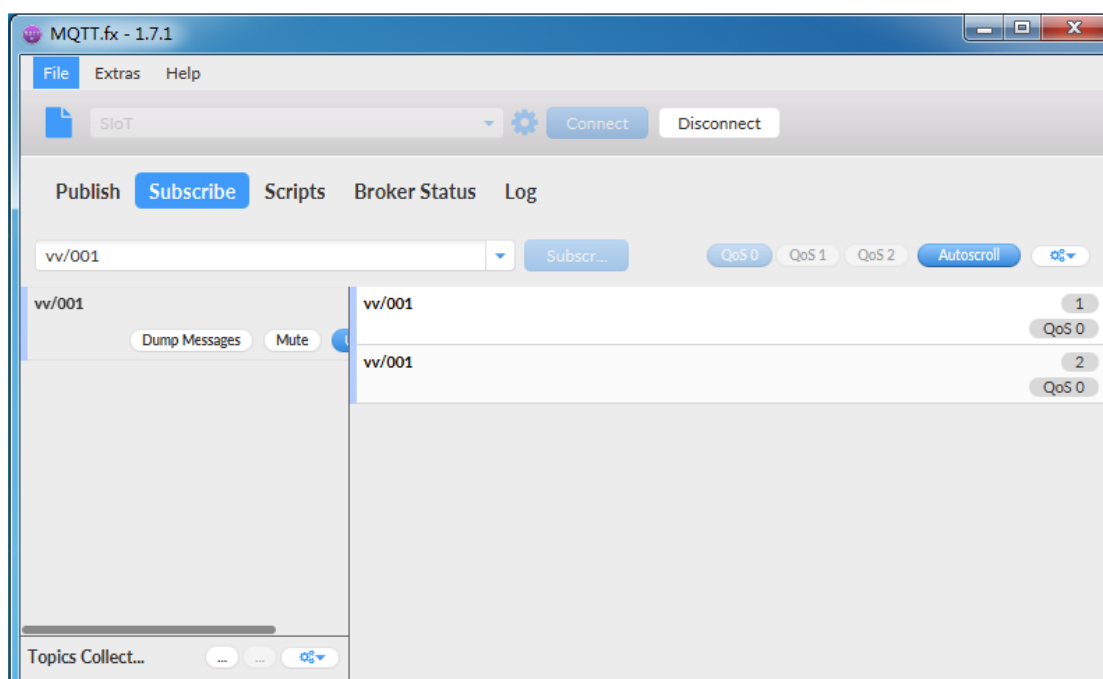
步骤 6：回到 Publish（发布）选项，在下拉框中选择一个主题（或创建一个与 Subscribe（订阅）选项中一样的主题）。在下方的输入区域写上你要发送的消息（如 xu gu hao，中文在订阅

者的消息显示上会乱码），这里的消息支持多种格式，然后点击 Publish（发布）按钮。



【实验结果】

再回来 Subscribe（订阅）选项中查看，已经成功接收到发布者发送的消息（xu gu hao），如图所示。



2.5 实验拓展

用语音输出订阅的消息内容。

展示你的实验成果，并且填写评价表。

评价内容	评价结果	备注
连接 SIoT 服务器并订阅 TopicTD, 通过 Web 页面发送消息, 测试订阅结果	☆☆☆☆	
通过其他客户端发送消息, 测试订阅结果	☆☆☆☆	

2.6 知识链接

SIoT 为标准的 MQTT 服务器, 支持绝大多数的客户端程序连接。常见的 MQTT 的 PC 客户端除了已经介绍过的 MQTTfx 之外还有 MQTTBox, 同样也支持在 Windows、Mac 和 Linux 上运行。还有手机端, 手机作为人们最熟悉的智能终端, 借助一些 MQTT 客户端 App, 手机可以连接 SIoT, 成为一个物联网终端。如, 运行于 iPhone 上的 MQTTTool 和运行在安卓系统上的 MQTT Client。有 App Inventor2 基础的同学, 可以自行开发一个 App 用于控制。

3. 互动控制实验

3.1 实验目的

- 了解简单的物联网场景应用模拟搭建。
- 学会发送 (publish) 消息与订阅 (subscribe) 消息的综合应用。

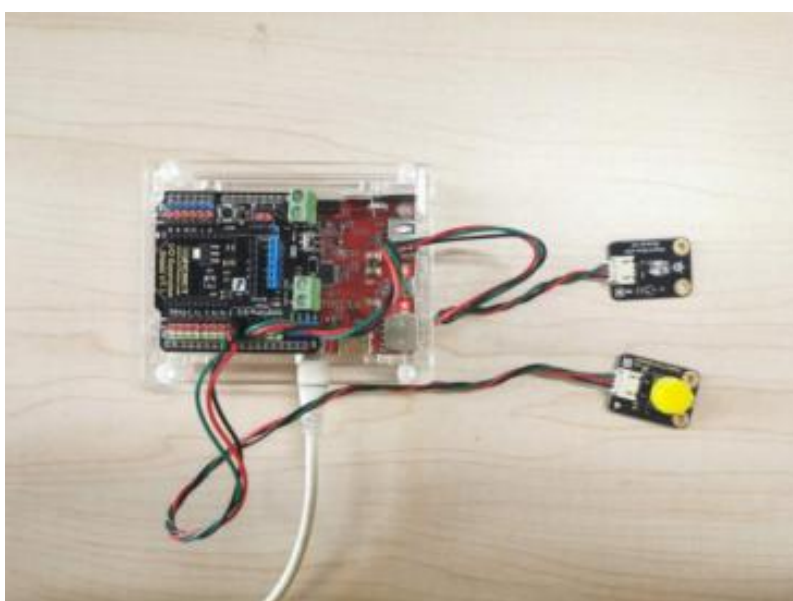
3.2 实验描述

- 订阅消息 “vv/01” ，收到 1 ，发送数据到 “vv/02”
- 按下按钮发送消息 “ok” 到 “vv/02” ，并订阅消息 “vv/01” ，收到 “on” 则点亮 LED ，收到 “off” 则关闭 LED。通过 Web 管理页面测试。

3.3 实验指导 --- 电子模块连接

硬件：虚谷号、Arduino 扩展板、LED 灯、按钮

将按钮、LED 灯通过 arduino 扩展板接到虚谷号板载 Arduino 的上，按钮接 9 号输出口，LED 灯接 11 号口。



3.4 实验步骤

3.4.1 订阅消息“vv/01”，收到1，发送数据到“vv/02”

在无线模式下打开 jupyter，打开“物联网实验”文件夹。在该文件目录下，打开 jupyter，输入如下代码并执行：

```
"""导入库"""
import siot
import time

"""配置 SIOT 服务器"""
SERVER = "127.0.0.1"
CLIENT_ID = ""
IOT_pubTopic = 'vv/01'
IOT_publishTopic = 'vv/02'
IOT_UserName = 'scope'
IOT_PassWord = 'scope'

"""连接 SIOT 服务器"""
siot.init(CLIENT_ID, SERVER, user = IOT_UserName, password = IOT_PassWord)
siot.connect()

"""发送消息到 vv/02"""
def sub_cb(client, userdata, msg):
    print(msg.payload.decode())
    if(str(msg.payload.decode())=="1"):
        #收到消息"1"
        siot.publish(IOT_publishTopic, "收到")

"""订阅消息 vv/01"""
siot.subscribe(IOT_pubTopic, sub_cb)
siot.loop()
```

【实验结果】

将看到执行程序时，程序输出：“连接结果，连接成功”字样，如图所示。

连接结果：连接成功

3.4.2 按下按钮发送消息“ok”到“vv/02”，并订阅消息“vv/01”，收到“on”则点亮LED，收到“off”则关闭LED。

步骤 1：在“物联网实验”文件夹目录下，打开 jupyter，输入如下代码并执行：

```
"""导入库"""
import siot
import time
from xugu import Pin

"""配置 SIOT 服务器"""
SERVER = "127.0.0.1"
CLIENT_ID = ""
IOT_pubTopic = 'vv/01'
IOT_publishTopic = 'vv/02'
IOT_UserName = 'scope'
IOT_PassWord = 'scope'

"""连接 SIOT 服务器"""
siot.init(CLIENT_ID, SERVER, user = IOT_UserName, password = IOT_PassWord)
siot.connect()

"""引脚初始化"""
button = Pin(9, Pin.IN)
led = Pin(11, Pin.OUT)

"""构建订阅信息的回调函数"""
ret = 0
def sub_cb(client, userdata, msg):
    global ret
    print("\nTopic:" + str(msg.topic) + " Message:" + str(msg.payload))
    if msg.payload == b'on':
        ret = 1
    if msg.payload == b'off':
        ret = 0

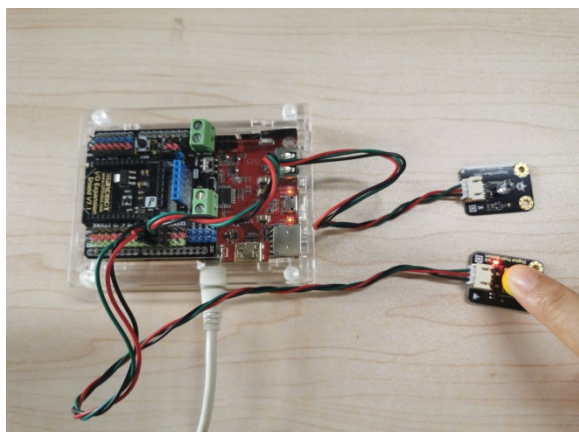
"""实现互动控制"""
while True:
    if ret == 1:
```

```

        led.write_digital(1)
    if ret == 0:
        led.write_digital(0)
    time.sleep(0.5) # 去抖
    value = button.read_digital()
    if(value > 0):
        siot.subscribe(IOT_pubTopic, sub_cb)
        siot.loop()
        siot.publish(IOT_publishTopic, "ok")

```

步骤 2: 用手按压按钮, 打开 SIoT 的 Web 管理页面, 在“设备列表”下查看 Topic “vv/02”。



【实验结果】

将看到程序执行输出输出:“连接结果,连接成功”字样。并在 Web 管理页面设备 Topic“vv/02”下看到消息 “ok”, 如图所示。

当前主题 : vv/02

发送消息

(为消息加上->前缀代表此消息为纯指令消息, 不会被存入数据库。例如"->off")

Topic	消息	时间
vv/02	ok	2020-06-12 16:10:40

点击“发送消息”页面, 在主题对话框中输入: vv/01, 消息对话框中输入: on, 点击发送。

SloT	项目列表	设备列表	发送消息
-------------	------	------	------

主题 (项目ID/设备名)

vv/01

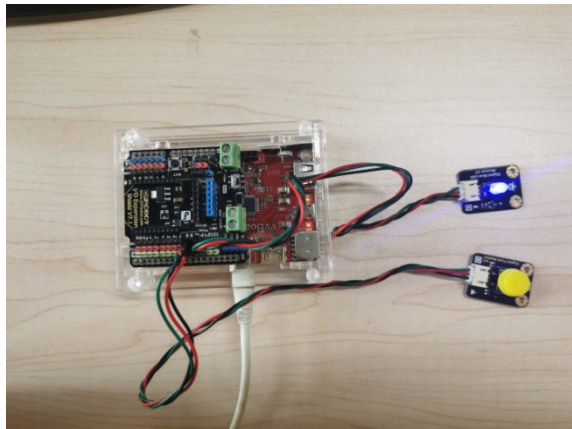
消息

on

(为消息加上->前缀代表此消息为纯指令消息，不会被存入数据库。例如"->off")

发送

将看到 LED 灯点亮，如图所示。



消息对话框中输入：off，点击发送。

SloT	项目列表	设备列表	发送消息
-------------	------	------	------

主题 (项目ID/设备名)

vv/01

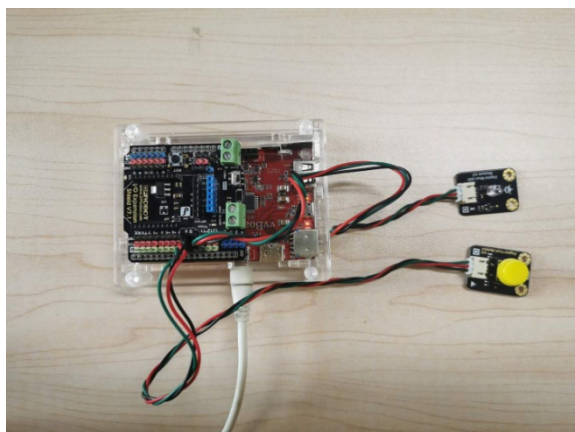
消息

off

(为消息加上->前缀代表此消息为纯指令消息，不会被存入数据库。例如"->off")

发送

将看到 LED 灯熄灭，如图所示。



3.5 实验拓展

展示你的实验成果，并且填写评价表。

评价内容	评价结果	备注
订阅消息“vv/01”，收到 1，发送数据到“vv/02”	☆☆☆☆☆	
按下按钮发送消息“ok”到“vv/02”，并订阅消息“vv/01”，收到“on”则点亮 LED，收到“off”则关闭 LED。通过 Web 管理页面测试。	☆☆☆☆☆	

3.6 知识链接

物联网架构可以分为感知层、网络层、应用层。感知层由各种传感器组成，将物体的数据，通过传感器收集后，由网络层传输出去。网络层包含互联网、云端、运营商网络、各种短距离局域网。应用层是物联网与用户的接口，一般以 UI 界面的形式展现。

感知层用于识别物体、采集信息，与人体结构中皮肤和五官的作用类似。感知层所需要的关键技术包括检测技术、短距离无线通信技术。

网络层用于传递信息和处理信息。网络层包括通信网与互联网的融合网络、网络管理中心、信息中心和智能处理中心等。网络层将感知层获取的信息进行传递和处理，类似于人体结构中的神经中枢和大脑。网络层解决的是传输和预处理感知层所获得数据的问题。网络层所需要的关键技术包括长距离有限和无线通信技术、网络技术等。

网络层中的感知数据管理与处理技术是实现以数据为中心的物联网的核心技术，包括传感网数据的存储、查询、分析、挖掘和理解，以及基于感知数据角儿的理论和技术。

4. WebAPI 调用实验

4.1 实验目的

- 了解 API 的相关概念
- 学会调用 WebAPI 实现消息发送、消息订阅的基本操作。

4.2 实验描述

- 通过 WebAPI 发送消息
- 通过 WebAPI 订阅消息
- 通过 WebAPI 获得历史消息

4.3 实验指导

API 简介

API (Application Programming Interface)，即应用程序接口。是一些预先定义好的函数，或指软件系统不同组成部分衔接的约定。用来提供应用程序与开发人员基于某软件或硬件得以访问的一组例程，使之无需访问原码，或理解内部工作机制的细节。

WebAPI 是专门操作 web 网页的应用程序接口。SIoT 提供了一系列的 WebAPI，供用户调用。一些不支持 MQTT 的编程语言，如 VB、S4A 等，可以通过 HTTP 的形式和 SIoT 进行互动。

需要强调的是，HTTP 的请求/应答方式的会话都是客户端发起的，缺乏服务器通知客户端的机制，如果要获取服务器信息，客户端应用需要不断地轮询服务器。这也就是在物联网方面，MQTT 比 HTTP 更受欢迎的原因。

4.4 实验步骤

4.4.1 通过 WebAPI 发送消息

打开 jupyter，输入如下代码，并执行。

```
"""导入库"""  
import requests
```

```

"""配置 SIOT 服务器"""
SERVER = "127.0.0.1"
PORT="8080"
CLIENT_ID = ""
IOT_pubTopic = 'vv/01'
IOT_UserName = 'scope'
IOT_PassWord = 'scope'

"""生成需要访问的 url"""
def get_url(msg):

url="http://%s:%s/publish?topic=%s&msg=%s&iname=%s&ipwd=%s"%(SERVER, PORT, IOT_pubTopic, msg, IOT_UserName, IOT_PassWord)

    return url

result = requests.get(get_url("欢迎使用虚谷号！"))
print(result.text)

```

【实验结果】

将看到程序返回信息，如图所示。

```
{ "code":1, "msg": "数据已发送" }
```

打开 SIoT 的 web 管理页面，查看主题“vv/01”，也能看到发送的消息。

当前主题：vv/01

发送消息

 消息内容

(为消息加上->前缀代表此消息为纯指令消息，不会被存入数据库。例如"->off")

开始时间	结束时间	100条 ▼	查询	导出查询结果	隐藏/显示图表
Topic	消息	时间			
vv/01	欢迎使用虚谷号！	2020-06-11 18:49:22			

4.4.2 通过 WebAPI 订阅消息

打开 jupyter，输入如下代码，并执行。

```

"""导入库"""
import requests

"""配置 SIOT 服务器"""
SERVER = "127.0.0.1"
PORT="8080"

```

```

CLIENT_ID = ""
IOT_pubTopic = 'vv/01'
IOT_UserName = 'scope'
IOT_PassWord = 'scope'

"""生成需要访问的 url"""
def get_url():

url="http://%s:%s/lastmessage?topic=%s&&iname=%s&ipwd=%s"%(SERVER, PORT, IOT_pubTopic,
IOT_UserName, IOT_PassWord)

    return url

result = requests.get(get_url())
print(result.text)

```

【实验结果】

将看到程序返回信息，如图所示。

```
{'code':1,'data': [{'ID':6234,'Topic':'vv/01','Content':'欢迎使用虚谷号!', 'Created':'2020-06-13 11:41:19'}], 'msg':'成功'}
```

3. 通过

WebAPI 获得历史消息

打开 jupyter，输入如下代码，并执行。

```

"""导入库"""
import requests
import datetime
from dateutil.relativedelta import relativedelta

"""配置 SIOT 服务器"""
SERVER = "127.0.0.1"
PORT = "8080"
CLIENT_ID = ""
IOT_pubTopic = 'x zr/001'
IOT_UserName = 'scope'
IOT_PassWord = 'scope'

"""url 参数设置"""
params = {
    'iname': "scope",
    'ipwd': "scope",
    'pnum': "1",
    'psize': "10",

```

```

        'begin': "2020-05-21",
        'end': "2020-06-30"
    }

    """生成需要访问的 url"""
    def get_url():
        url = "http://%s:%s/messages" % (SERVER, PORT)
        return url

    result = requests.get(get_url(), params = params)
    print(result.text)

```

【实验结果】

将看到程序返回参数 begin 和 end 时间段之间的所有信息，如图所示。

```

{"code":1,"data":[{"ID":4808,"Topic":"vv/01","Content":"on","Created":"2020-06-27 10:31:27"}, {"ID":4809,"Topic":"vv/01","Content":"on","Created":"2020-06-27 10:41:20"}, {"ID":4810,"Topic":"vv/01","Content":"on","Created":"2020-06-27 10:41:31"}, {"ID":4811,"Topic":"vv/01","Content":"on","Created":"2020-06-27 10:41:39"}, {"ID":4812,"Topic":"vv/01","Content":"on","Created":"2020-06-29 10:45:01"}, {"ID":4813,"Topic":"vv/01","Content":"欢迎使用虚谷号!", "Created":"2020-06-29 11:07:16"}], "msg":"成功"}

```

4.5 实验拓展

设计一个网页，控制虚谷号的设备

展示你的实验成果，并且填写评价表。

评价内容	评价结果	备注
通过 WebAPI 发送消息	☆☆☆☆☆	
通过 WebAPI 订阅消息	☆☆☆☆☆	
通过 WebAPI 获得历史消息	☆☆☆☆☆	

4.6 知识链接

返回数据通用格式说明：

code = 1, message = “ ”, data = {}

code 等于 1，表示成功。data 中返回列表信息。

除以上实验中提到的 API，SIoT 还提供了以下这些 API：

(1). 删除消息

Http://[SIoT 的 IP]:8080/clearmsg?topic=%s&iname=%s&ipwd=%s

说明：删除 topicid（主题）的所有消息

(2). 获取项目列表

Http://[SIoT 的 IP]:8080/projects?iname=%s&ipwd=%s

说明：获取所有的项目列表

(3). 更新项目

Http://[SIoT 的 IP]:8080/updateprj?pid=xzr&iname=siot&ipwd=dfrobot&desc=科学测量

说明：将名称为“xzr”的项目的备注修改为“科学测量”，返回数据：{“code”:1,“msg”：“成功”}

(4). 获取设备列表

Http://[SIoT 的 IP]:8080/devices?pid=xzr&iname=siot&ipwd=dfrobot

说明：返回项目名称为“xzr”的设备列表。

(5). 更新设备

Http://[SIoT 的 IP]:8080/updatedev?pid=xzr&dname=001&iname=siot&ipwd=dfrobot&desc=台灯控制

说明：将项目名称为“xzr”，设备名称为“001”的设备，备注修改为“台灯控制”（发现 BUG）

(6). 删除设备

Http://[SIoT 的 IP]:8080/deldev?topic=xzr/001&iname=siot&ipwd=dfrobot

说明：返回数据：{“code”:1,“msg”：“删除成功”}

5. 物联网数据可视化实验

5.1 实验目的

- 了解物联网数据可视化的意义
- 学会基于 Python 编程的物联网数据可视化方法
- 学会基于 Node-RED 的物联网数据可视化方法

5.2 实验描述

- 订阅数据，并实时画出折线图
- 订阅数据，并实时画出仪表盘

5.3 实验指导

5.3.1 电子模块连接

硬件：虚谷号、Arduino 扩展板、温湿度传感器

将温湿度传感器通过 arduino 扩展板接到虚谷号板载 Arduino 的 13 号口。



5.3.2 温湿度传感器数据的采集与处理

使用 dhtc 库中的 DHT 类，对温湿度传感器的数据进行采集和处理。DHT 类中用 “read()” 函数读取传感器的值。

```
from dhtc import DHT
dht = DHT()
data = dht.read()
split()方法通过指定分隔符对字符串进行切片，语法格式 str.split(str="",
num=string.count(str))
```

参数介绍：str 为分隔符，默认为所有的空字符，包括空格、换行(\n)、制表符(\t)等。num 为分割次数。默认值为-1，即分割所有。如果第二个参数 num 有指定值，则分割为 num+1 个子字符串。

5.3.3 Node-RED 简介

Node-RED 是构建物联网应用程序的一个强大工具，其重点是简化代码块的“连接”以执行任务。它使用可视化编程方法，允许开发人员将预定义的代码块（也叫做“节点（node）”）连接起来执行任务。连接的节点，通常是输入节点、处理节点和输出节点的组合，当它们连接在一起时，构成一个“流(flow)”。

Node-RED 的主界面共有三个部分，从左到右分别为：拥有各种功能的节点栏，放置各种编程节点的流程栏，用于提供节点帮助和调试信息的信息栏。



节点(node)，是流的基本构建块。通过从流中的上一个节点接收消息或通过等待某些外部事件（例如传入的网络上 HTTP 请求、传入硬件设备的信息等）来触发节点，处理该消息或事件，然后将消息发送到流中的下一个节点。一个节点最多可以具有一个输入端口和所需的多个输出端口。

流(flow)，在编辑器工作空间中是组织节点的主要方式。术语“流”还用于非正式地描述一组连接的节点。因此，一个流可以包含多个流（连接的节点集）。

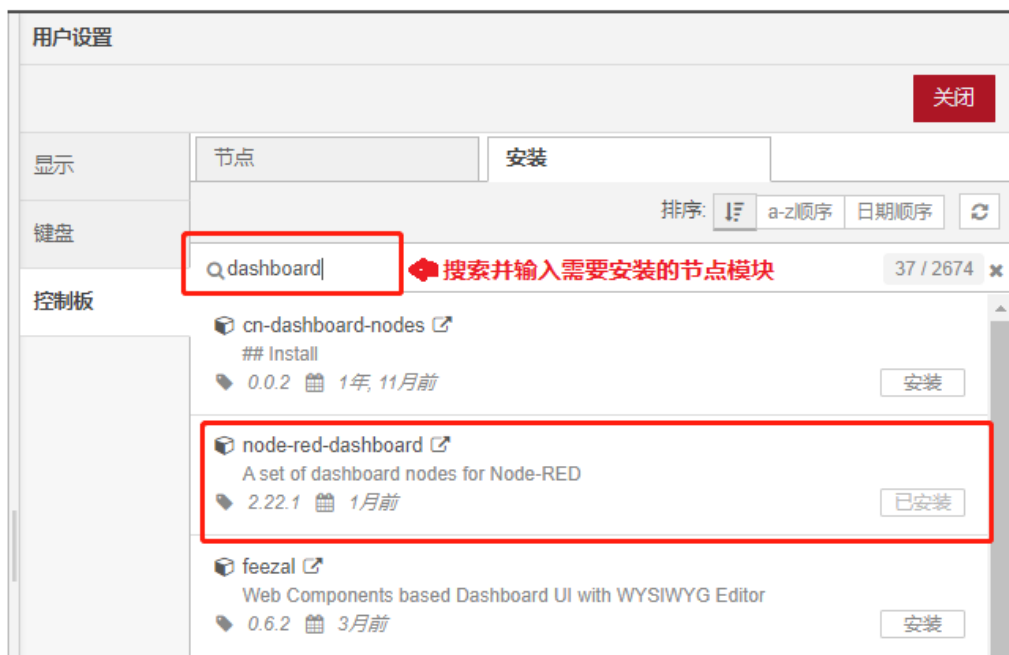
mqtt 节点基于 Mqtt 协议的服务器，能够为传感器提供轻量级的消息订阅和发布。



5.3.4 Dashboard 模块

Dashboard 模块是 Node-RED 中的第三方模块，主要用于快速创建实时数据仪表盘，像 Python 的第三方库一样需要进行安装才能使用。Dashboard 模块提供丰富多样的图形化节点来做出各种各样的图表。

Dashboard 模块的安装：在右上角设置菜单中，选择节点管理，打开对话框，如图所示。



成功安装后会看到左侧的列表中出现了新的可用节点，可以从中选择不同的图表以及各种数据表现形式进行数据可视化。

仪表板的布局依赖于 Tab 和 Group 属性。Tab 可以理解为页面，Group 是分组。Tab 可以包含 Group。

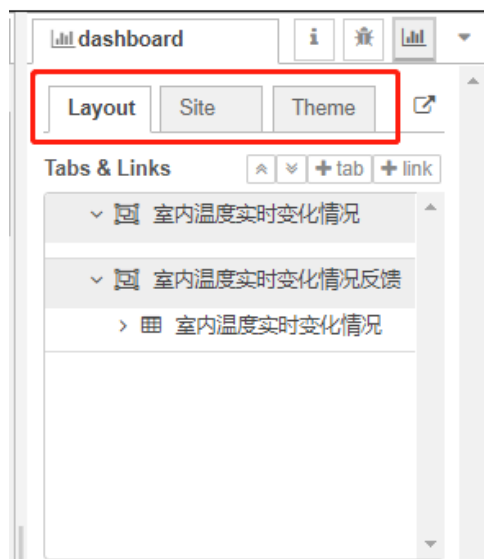


使用 dashboard 节点时，屏幕右侧“调试窗口”的旁边会多一个名为 dashborad 的小标签，下边有 Layout，Theme 和 Site 三个选项。

Layout 是布局，在 Layout 里可以重新排列 Tab，Group 与控件，并编辑其属性。也可以把其它网页添加到 Tab 中。

Theme 是主题。可以选明亮的，或者暗的，或者自定义。

Site 是地址，可以设置标题的 UI，或者选择标题栏。也能够以像素为单位设置网格布局的基本图形，就是刚刚提到默认是 48 像素的那个“单位”，或者单独设置控件，组的大小。



guage 仪表节点，有四种模式，标准，甜甜圈，指南针和波形。可以指定标准仪表或甜甜圈测量仪的颜色范围。

编辑gauge节点

删除 取消 完成

属性

Group [unassigned] Group

Size auto

Type Gauge

Label gauge

Value format {{value}}

Units units

Range min 0 max 10

Colour gradient

Sectors 0 ... optional ... optional ... 10

Name

chart 图表节点，具有折线、条形与饼图模式，可以使用日期格式化程序字符串来自定义 X 轴标签

编辑chart节点

删除 取消 完成

属性

Group [unassigned] Group

Size auto

Label chart

Type Line chart enlarge points

X-axis last 1 hours OR 1000 points

X-axis Label HH:mm:ss as UTC

Y-axis min max

Legend None Interpolate linear

Series Colours

Blank label display this text before valid data arrives

Name

5.4 实验步骤

5.4.1 订阅数据，并实时画出折线图

步骤 1: 打开 jupyter，输入以下代码并执行。

```
""" 导入库 """
import siot

""" 配置 SIOT 服务器 """
SERVER = "127.0.0.1"
CLIENT_ID = ""
IOT_pubTopic = 'vv/100'
IOT_UserName = 'scope'
IOT_PassWord = 'scope'

""" 连接 SIOT 服务器 """
siot.init(CLIENT_ID, SERVER, user = IOT_UserName, password = IOT_PassWord)
siot.connect()

""" 设置绘画函数 """
import matplotlib.pyplot as plt
%matplotlib inline
from IPython import display
x, pl = [], []
i = 0
w = 20 #设置数据的长度
def draw(v1):
    global x, i, pl
    i = i + 1
    x.append(i)
    pl.append(v1)
    # 当数据太多了开始删除，避免图表越来越小
    if len(x) > w:
        x.pop(0)
        pl.pop(0)
    fig = plt.figure()
    plt.plot(x, pl, 'r--')
```

```

display.clear_output(wait = True)
plt.show()

""" 订阅消息 """
def sub_cb(client, userdata, msg):
    print("\nTopic:" + str(msg.topic) + " Message:" + str(msg.payload))
    # msg.payload 是 bytes 类型, 要转换
    s = float(msg.payload)
    draw(s)

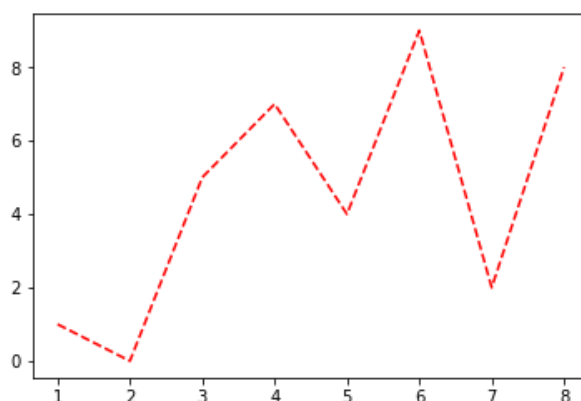
siot.subscribe(IOT_pubTopic, sub_cb)
siot.loop()

```

步骤 2: 打开 SIoT 的 Web 管理界面, 点击“发送消息”, 输入主题: vv/100, 并输入不同的消息点击发送。比如, 输入“1”, 点发送, 观察程序中的绘图结果。不断的输入消息, 并发送。消息示例: 1, 0, 5, 7, 4, 9, 2, 8。

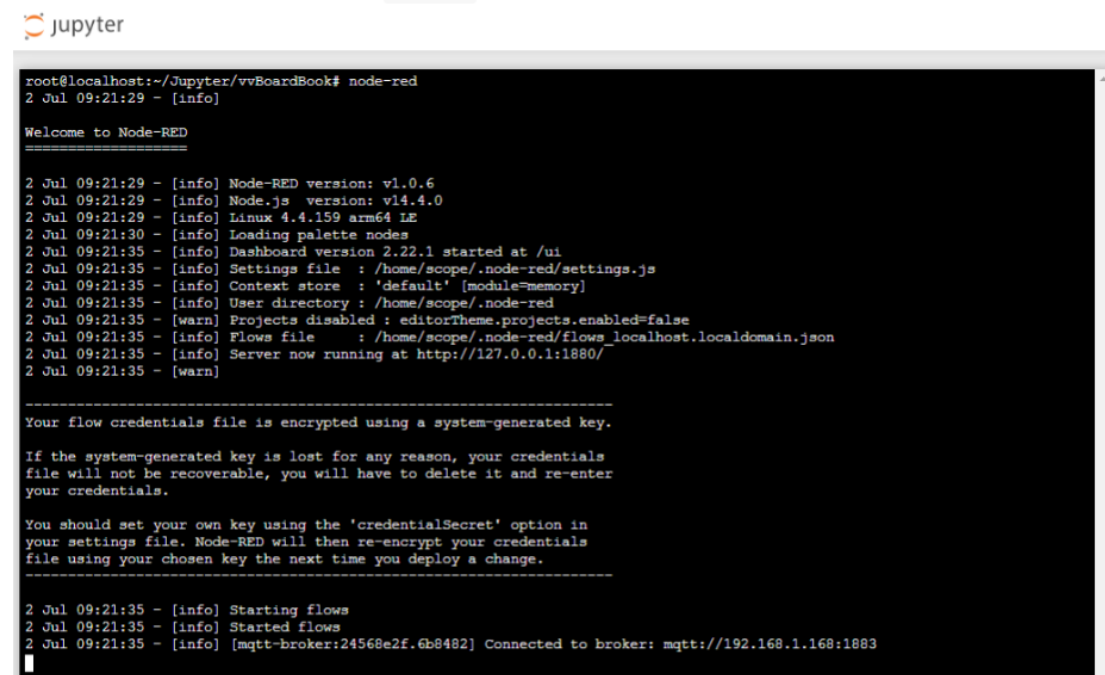
【实验结果】

将看到随着消息的不断发出, 将动态的绘制出一张折线图, 如图所示。



5.4.2 订阅数据, 并实时画出仪表盘

步骤 1: 打开 jupyter, 新建终端, 在终端输入: node-red 命令, 点击回车, 启动 Node-RED。



```
jupyter

root@localhost:~/Jupyter/vvBoardBook# node-red
2 Jul 09:21:29 - [info]

Welcome to Node-RED

=====

2 Jul 09:21:29 - [info] Node-RED version: v1.0.6
2 Jul 09:21:29 - [info] Node.js version: v14.4.0
2 Jul 09:21:29 - [info] Linux 4.4.159 arm64 LE
2 Jul 09:21:30 - [info] Loading palette nodes
2 Jul 09:21:35 - [info] Dashboard version 2.22.1 started at /ui
2 Jul 09:21:35 - [info] Settings file : /home/scope/.node-red/settings.js
2 Jul 09:21:35 - [info] Context store : 'default' [module=memory]
2 Jul 09:21:35 - [info] User directory : /home/scope/.node-red
2 Jul 09:21:35 - [warn] Projects disabled : editorTheme.projects.enabled=false
2 Jul 09:21:35 - [info] Flows file : /home/scope/.node-red/flows_localhost.localdomain.json
2 Jul 09:21:35 - [info] Server now running at http://127.0.0.1:1880/
2 Jul 09:21:35 - [warn]

-----

Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.

-----

2 Jul 09:21:35 - [info] Starting flows
2 Jul 09:21:35 - [info] Started flows
2 Jul 09:21:35 - [info] [mqtt-broker:24568e2f.6b8482] Connected to broker: mqtt://192.168.1.168:1883
```

注意：服务器需要在终端中持续运行，可以在终端看到数据流的开始和终止，关闭终端，浏览器会提示丢失数据。

步骤 2：打开浏览器，在网址栏中输入：<http://【虚谷号 IP】:1880>，进入 Node-RED 网页编程页面。将 mqtt in 节点选中并拖出，放进流程栏。双击 mqtt in 节点，进入节点编辑对话框，如图。



编辑mqtt in节点

删除 取消 完成

属性

服务端 192.168.1.168:1883 在这里添加服务器IP地址和账户密码

主题 vv/01

QoS 1

输出 自动检测 (字符串或buffer)

名称 名称

将服务端端口设置为 1883，账户密码都是“scope”。完成后点击“更新”、“完成”。

编辑mqtt in节点 > 编辑mqtt-broker节点

删除 取消 更新

属性

名称 名称

连接 安全 消息

服务端 192.168.1.168 端口 1883

☐ 使用安全连接 (SSL/TLS)

客户端ID 留白则自动生成

Keepalive计时(秒) 60 ☒ 使用新的会话

☐ 使用旧式MQTT 3.1支持

依次拖出 debug 节点、chart 节点、gauge 节点，设置节点信息如图所示。

编辑debug节点

删除 取消 完成

属性

输出 完整信息

目标 ☒ 调试窗口 ☐ Console

名称 室内温度值

编辑chart节点

删除 取消 完成

属性

Group [室内气温变化实时情况] 室内温度

Size auto

Label 室内温度变化情况

Type Line chart ☐ enlarge points

X-axis last 1 minute OR 1000 points

X-axis Label HH:mm ☐ as UTC

Y-axis min 5 max 35

Legend Show Interpolate cubic

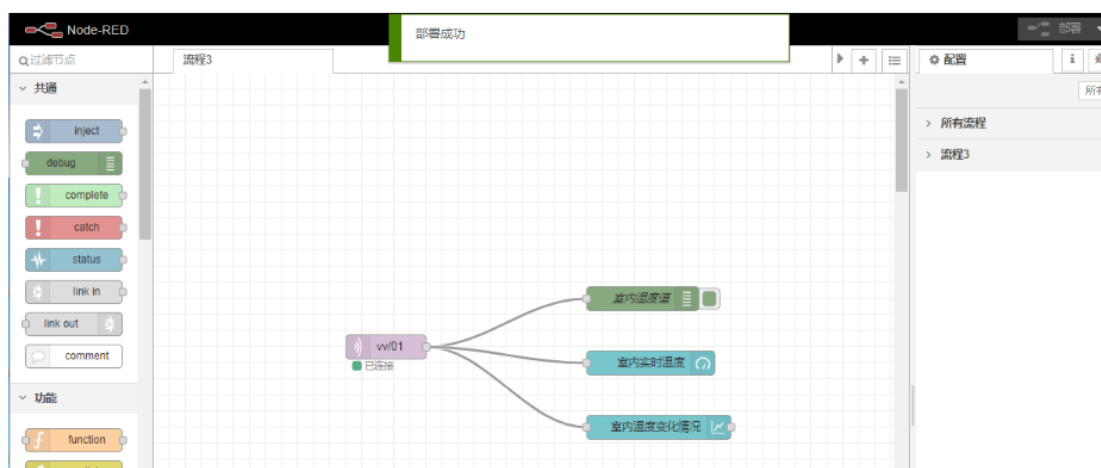
Series Colours

Blank label display this text before valid data arrives

Name



步骤 3: 完成所有设置, 点击节点上灰色小点, 将各个节点与 mqtt in 相连, 连好之后, 点击右上角**部署**, 显示“部署成功”, 如图所示。



步骤 4: 打开浏览器, 在网址栏中输入: `http://【虚谷号 IP】:1880/ui`, 如:
`http://192.168.1.168:1880/ui`

步骤 5: 打开 jupyter, 输入以下代码并执行。

"""导入库"""

`import siot`

`import time`

`from dhc import DHT`

"""配置 SIOT 服务器"""

`SERVER = "127.0.0.1"`

`CLIENT_ID = ""`

`IOT_pubTopic = 'vv/02'`


```
IOT_publishTopic = 'vv/01'
IOT_UserName = 'scope'
IOT_PassWord = 'scope'

"""连接 SIOT 服务器"""
siot.init(CLIENT_ID, SERVER, user = IOT_UserName, password = IOT_PassWord)
siot.connect()
siot.loop()

"""构建订阅信息的回调函数"""
ret = 0
def sub_cb(client, userdata, msg):
    global ret
    print("\nTopic:" + str(msg.topic) + " Message:" + str(msg.payload))

siot.subscribe(IOT_pubTopic, sub_cb)
dht = DHT()
while True:
    failcount = 3
    count = 0
    while count <= failcount:
        time.sleep(10)
        data = dht.read()
        if(type(data) == str):
            break
        count += 1
    if(type(data) == int):
        pass
    else:
        print(data.split('.')[1])
        siot.publish(IOT_publishTopic ,data.split('.')[1])
```

【实验结果】

将看到程序打印出温度传感器返回的值，同时查看 Node-RED 的调试信息，也可以看到温度值信息，如图所示。

```
连接结果: 连接成功
not found firmata protocol, burn it.
burn complete
27
27
27
28
28
28
28
29
30
31
31
31
31
30
```



点击查看步骤 4 打开的网页，可以看到实时的温度仪表盘图和折线图。



5.5 实验拓展

实时分析数据，用仪表盘页面显示并自动刷新。
展示你的实验成果，并且填写评价表。

评价内容	评价结果	备注
订阅数据，并实时画出折线图	☆☆☆☆☆	
订阅数据，并实时画出仪表盘	☆☆☆☆☆	

5.6 知识链接

修改“plt.plot”中的参数“r—”可以得到不同颜色和类型的线条，其中第一个字母表示颜色，如“r”表示红色，“g”表示绿色等。后面的字符表示线型。如：“r--”表示红色的虚线，“bs”表示蓝色的方块，“g^”表示绿色的三角形等等。

6. 多终端互联实验

6.1 实验目的

- 掌握消息发送与消息订阅的综合应用
- 学会建立简单的物联网应用

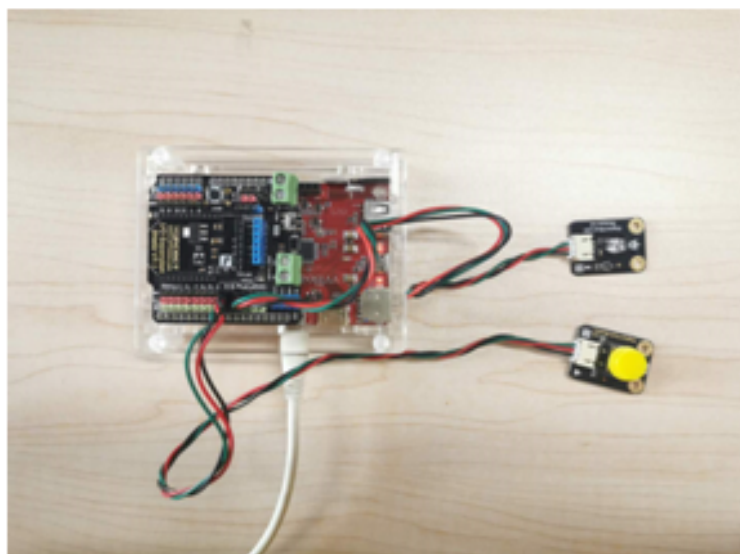
6.2 实验描述

- 终端 1:按下按钮发送消息 “on” 到 “vv/02” , 并订阅消息 “vv/01” , 收到 “on” 则点亮 LED ;
- 终端 2:按下按钮发送消息 “on” 到 “vv/01” , 并订阅消息 “vv/02” , 收到 “on” 则点亮 LED。

6.3 实验指导

6.3.1 电子模块连接

硬件：虚谷号、Arduino 扩展板、LED 灯、按钮
连接方式与 5.3 相同，具体参见 5.3。



6.3.2 终端定义

准备两个虚谷号,按照上述要求连接好,将其中一个虚谷号作为终端 1,另外一个作为终端 2。在实际实验中,终端 1 同时作为服务器使用。因此在终端 1 输入有终端 1 注释的代码,终端 2 输入另外一段代码。

本实验为两人一组合作完成的实验,请同学们两人一组合作完成,在一个组内,一个同学负责终端 1 的所有操作,另外一个同学负责终端 2 的操作。

6.4 实验步骤

终端 1: 按下按钮发送消息“on”到“vv/02”,并订阅消息“vv/01”,收到“on”则点亮 LED; 终端 2: 按下按钮发送消息“on”到“vv/01”,并订阅消息“vv/02”,收到“on”则点亮 LED。

步骤 1: 将虚谷号连接好电脑,打开 jupyter,在“物联网实验”文件夹目录下,新建 jupyter,输入如下代码,并执行:

```
# 终端 1
"""导入库"""
import siot
import time
from xugu import Pin

"""配置 SIOT 服务器"""
SERVER = "127.0.0.1"
CLIENT_ID = ""
IOT_pubTopic = 'vv/01'
IOT_publishTopic = 'vv/02'
IOT_UserName = 'scope'
IOT_PassWord = 'scope'

"""连接 SIOT 服务器"""
siot.init(CLIENT_ID, SERVER, user = IOT_UserName, password = IOT_PassWord)
siot.connect()
siot.loop()

"""引脚初始化"""
button = Pin(9, Pin.IN)
led = Pin(11, Pin.OUT)

"""构建订阅信息的回调函数"""
```

```
ret = 0
def sub_cb(client, userdata, msg):
    global ret
    print("\nTopic:" + str(msg.topic) + " Message:" + str(msg.payload))
    if msg.payload == b'on':
        ret = 1
    if msg.payload == b'off':
        ret = 0

siot.subscribe(IOT_pubTopic, sub_cb)

"""实现互动控制"""
on = False
while True:
    if ret == 1:
        led.write_digital(1)
    if ret == 0:
        led.write_digital(0)
    time.sleep(0.2) # 去抖
    value = button.read_digital()
    print(value)
    if(value > 0):
        if(not on):
            siot.publish(IOT_publishTopic, "on")
        else:
            siot.publish(IOT_publishTopic, "off")
        on = not on

# 终端 2
"""导入库"""
import siot
import time
from xugu import Pin

"""配置 SIOT 服务器"""
SERVER = "192.168.1.168"
CLIENT_ID = ""
IOT_pubTopic = 'vv/02'
IOT_publishTopic = 'vv/01'
IOT_UserName = 'scope'
```

```
IOT_PassWord = 'scope'
```

```
"""连接 SIOT 服务器"""
```

```
siot.init(CLIENT_ID, SERVER, user = IOT_UserName, password = IOT_PassWord)
```

```
siot.connect()
```

```
siot.loop()
```

```
"""引脚初始化"""
```

```
button = Pin(9, Pin.IN)
```

```
led = Pin(11, Pin.OUT)
```

```
"""构建订阅信息的回调函数"""
```

```
ret = 0
```

```
def sub_cb(client, userdata, msg):
```

```
    global ret
```

```
    print("\nTopic:" + str(msg.topic) + " Message:" + str(msg.payload))
```

```
    if msg.payload == b'on':
```

```
        ret = 1
```

```
    if msg.payload == b'off':
```

```
        ret = 0
```

```
siot.subscribe(IOT_pubTopic, sub_cb)
```

```
"""实现互动控制"""
```

```
on = False
```

```
while True:
```

```
    if ret == 1:
```

```
        led.write_digital(1)
```

```
    if ret == 0:
```

```
        led.write_digital(0)
```

```
    time.sleep(0.5) # 去抖
```

```
    value = button.read_digital()
```

```
    print(value)
```

```
    if(value == 0):
```

```
        if(not on):
```

```
            siot.publish(IOT_publishTopic, "on")
```

```
        else:
```

```
            siot.publish(IOT_publishTopic, "off")
```

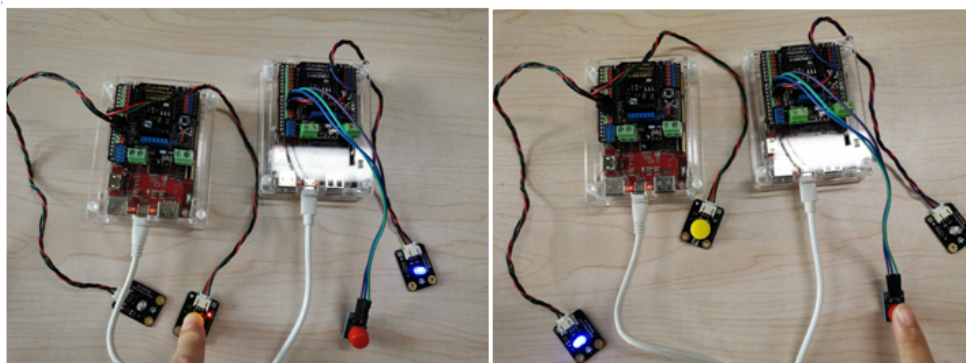
```
on = not on
```

注意：两段代码分别对应两个终端，一个终端只需要输入一段即可，请根据组内终端分配情况分别输入。

步骤 2：用手按压按钮，观察 LED 灯状态。

【实验结果】

将看到当终端 1 按下按钮，终端 2 的 LED 灯被点亮；当终端 2 按下按钮，终端 1 的 LED 灯被点亮。



6.5 实验拓展

用终端 1 的旋钮控制终端 2 的舵机

展示你的实验成果，并且填写评价表。

评价内容	评价结果	备注
终端 1: 按下按钮发送消息 “on” 到 “vv/02”，并订阅消息 “vv/01”，收到 “on” 则点亮 LED；	☆☆☆☆	
终端 2: 按下按钮发送消息 “on” 到 “vv/01”，并订阅消息 “vv/02”，收到 “on” 则点亮 LED。	☆☆☆☆	

6.6 知识链接

物联网的十大应用领域：

- (1) 智慧物流：新技术应用于物流行业的统称，通过物联网、大数据、人工智能等信息技术，实

现物流各个环境内的系统感知、全面分析及处理等功能。物联网技术应用于物流领域分为三个方面，即仓储管理、运输监测以及智能快递柜。目前，该行业的几大应用已全部实现了物联网数字化，未来应加强物流数字化水平，利用大数据、人工智能等算法实现物流数据化，满足客户的个性化需求。

(2) 智能交通：是物联网的重要应用场景之一。智能交通利用先进的信息技术，使人、车和路能够紧密的配合，改善交通运输环境、提高资源利用率等。智能交通的未来发展在于增强数据采集多样性，提高系统协同性，降低行业成本，培育更加适合地域与行业的新模式。

(3) 智能安防：安防应用于物联网技术，主要是通过摄像头进行物体监控。其最核心的部分在于智能安防系统，可分为门禁、报警和监控三大部分，行业中主要以视频监控为主。智能安防的未来发展在于提高识别精准度，深挖垂直行业的解决方案，发展民用市场，实现从数字化向智能化方向转变。

(4) 智能医疗：物联网技术应用于医疗领域，能有效地帮助医院实现对人和医疗物品的智能化管理。在智能医疗领域，我国刚处于起步阶段，未来应设计多场景应用传感器，挖掘更多的以人为主的医疗场景，同时加快提高医院医疗数字化水平。

(5) 智慧能源：物联网技术应用在能源领域，主要用于水，电，燃气等表计以及室外路灯的远程控制等，基于环境和设备进行物体感知，通过监测，提升利用效率，减少能源损耗。智慧能源现阶段还在探索商业模式阶段，未来政府应提高政策保障，企业应解决能源设备互联互通问题，同时加快节能设备的更换速度。

(6) 智慧农业：智慧农业指的是利用新技术实现农业可视化诊断、远程控制以及灾害预警等功能。当前，已经能简单的实现农作物、水果类以及畜牧产品的监测，未来发展应降低系统解决成本，着重获取农业数据，培育市场，提高农业数字化水平。

(7) 智能家居：物联网应用于智能家居领域，能够对家居产品的位置、状态、变化进行监测，同时根据人的需要，在一定的程度上进行反馈。智能家居行业发展主要分为三个阶段，单品连接、物物联动和平台集成。其未来发展方向是自单品向物物联动发展，同时制定行业标准，根据客户需要，个性化定制智能家居产品，打造多个智能家居入口。

(8) 智慧建筑：当前的智慧建筑主要体现在用电照明、消防监测以及楼宇控制等，实现设备感知并远程监控等。未来应从建筑内单纯的设备节能，向设备间的子系统协同发展，进而可向不同建筑系统的协同发展。

(9) 智能制造：制造领域应用于物联网技术，主要体现在数字化以及智能化的工厂改造上，包括工厂机械设备监控和工厂的环境监控。目前，工厂数字化水平还未实现，未来应提高工业设备的数字化水平，挖掘原有设备数据的价值，提高设备间的协同能力。

(10) 智能零售：物联网技术应用于零售领域，主要应用于近场零售，即无人便利店和自动（无人）售货机。通过将传统的售货机和便利店进行数字化升级、改造，打造无人零售模式。未来应着重利用获取的数据，通过人、场景等定位，对数据分析后进行用户画像描述，实现对用户的精准推荐。