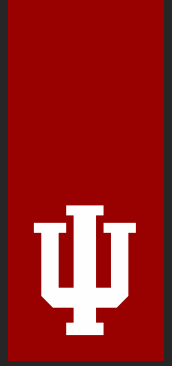


E210 Engineering Cyber-Physical Systems

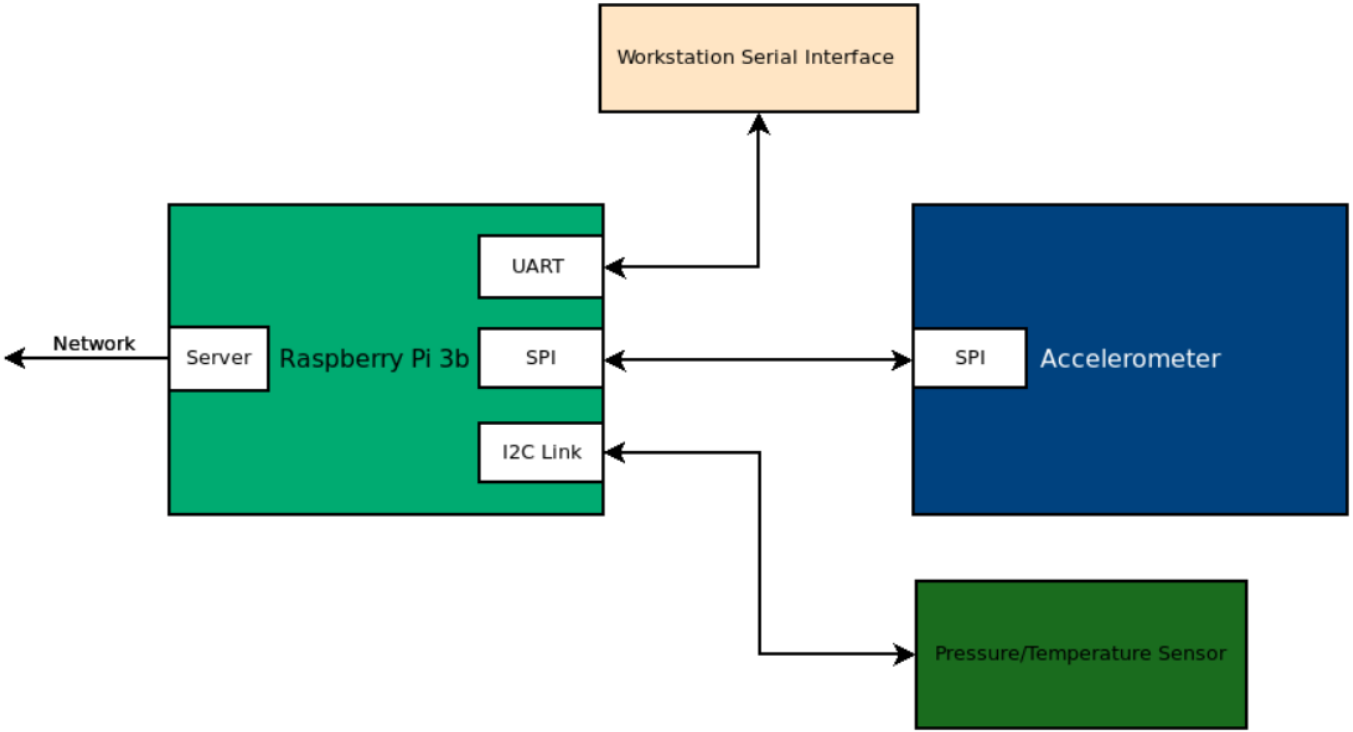
MQTT

Bryce Himebaugh

Weekly Focus	Reading	Monday	Wed	Lab
CPS Intro/UART		**1/10:** CPS Introduction	**1/12:** Pi Intro/UART Bus	**Project 0 Raspberry PI Setup**
I2C Bus		**1/17:** MLK Day	**1/19:** I2C Bus Overview	**Project 1 I2C Pressure/Temperature Sensor**
I2C and SPI Bus		**1/24:** Pressure Sensor	**1/26:** SPI Bus Overview	**Project 2 SPI Accelerometer**
SPI/Networking		**1/31:** Accelerometer	2/2: MQTT	**Project 3 MQTT Sensor Data Server**
Networking		**2/7:** GPIO	**2/9:** TBD	**Project 4 Sensor LED Output**
Web Server		**2/14:** CPS Wrapup	**2/16:** Exam Review	**P5 Demultiplexer**
Evaluation		**2/21:** Exam 1	**2/23:** CE Intro/ Logic	**P6 ALU**

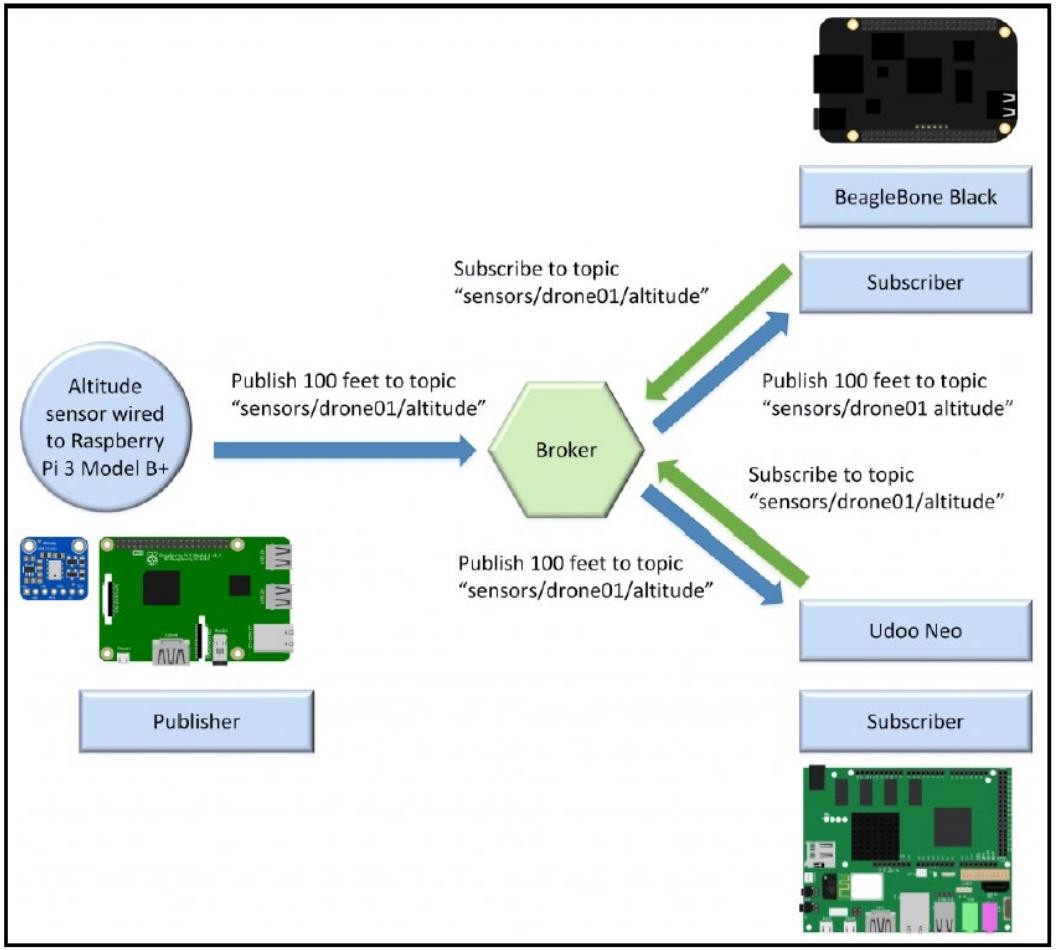
https://engr210.github.io/

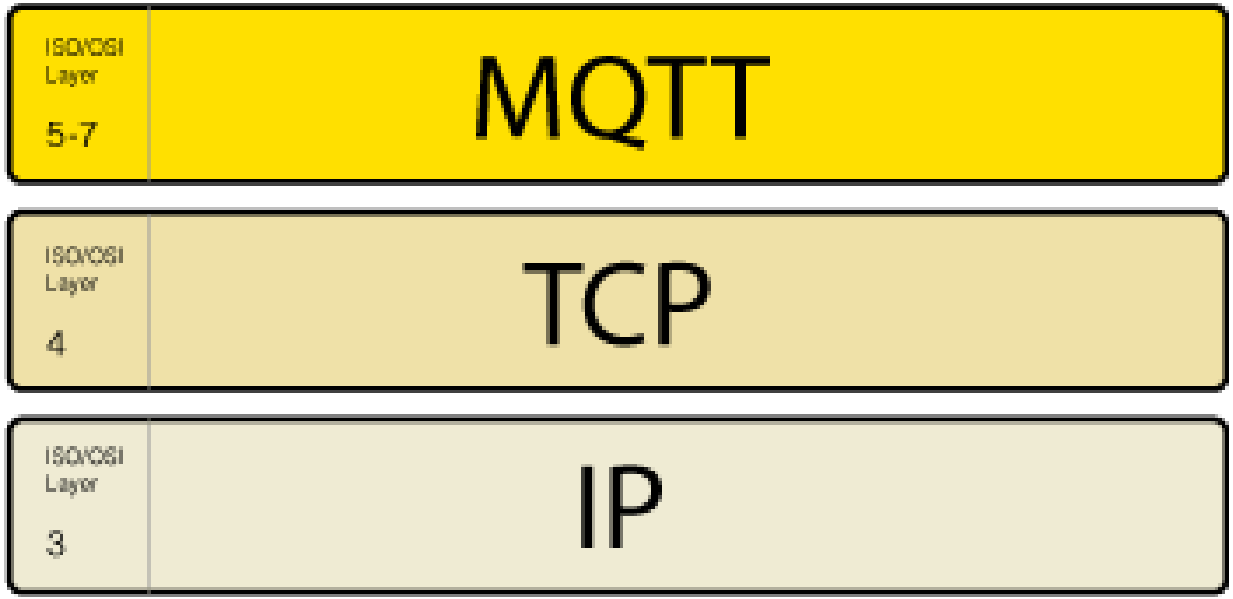
Sensor System



Message Queuing Telemetry Transport

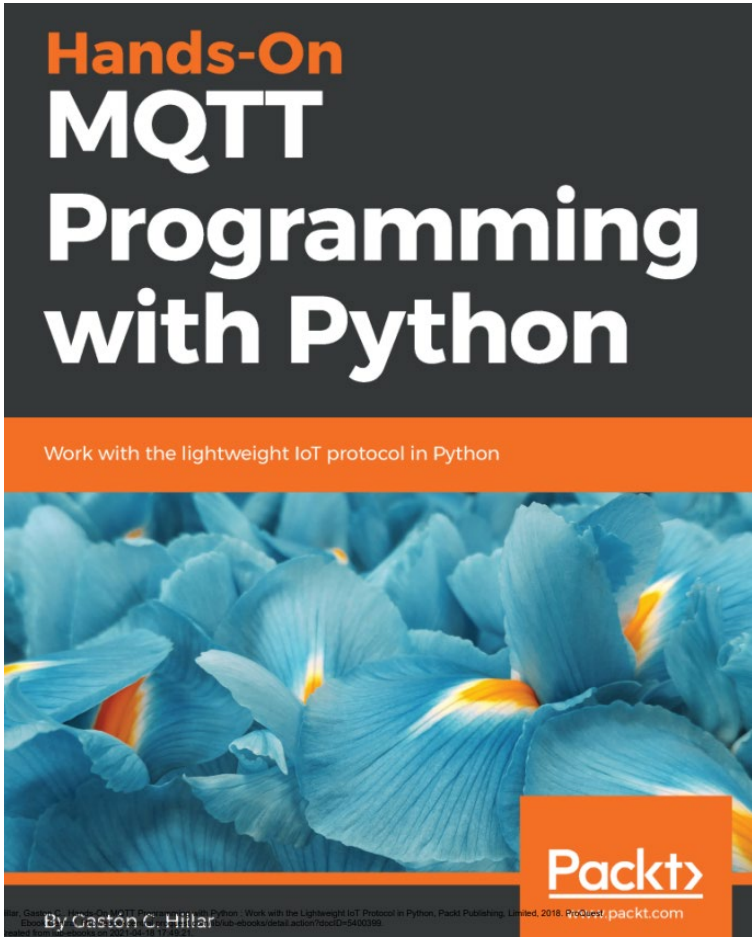
MQTT Overview


BeagleBone Black
Subscribe to topic
“sensors/drone01/altitude”
Subscriber
Altitude sensor wired to Raspberry Pi 3 Model B+
Publish 100 feet to topic
“sensors/drone01/altitude”
Broker
Publish 100 feet to topic
“sensors/drone01 altitude”
Subscribe to topic
“sensors/drone01/altitude”
Publish 100 feet to topic
“sensors/drone01/altitude”
Udoo Neo
Publisher
Subscriber


ISO/OSI Layer
5-7
MQTT
ISO/OSI Layer
4
TCP
ISO/OSI Layer
3
IP

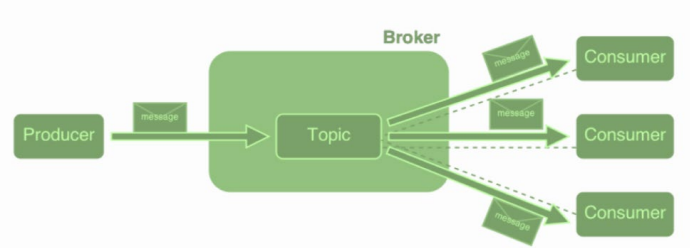
MQTT Reference

1. Electronic Access at IU Library
2. Chapter 1 good MQTT Background
3. Chapter 2-4 good overview of using MQTT
4. Read Chapter 1, skim chapters 2-4.

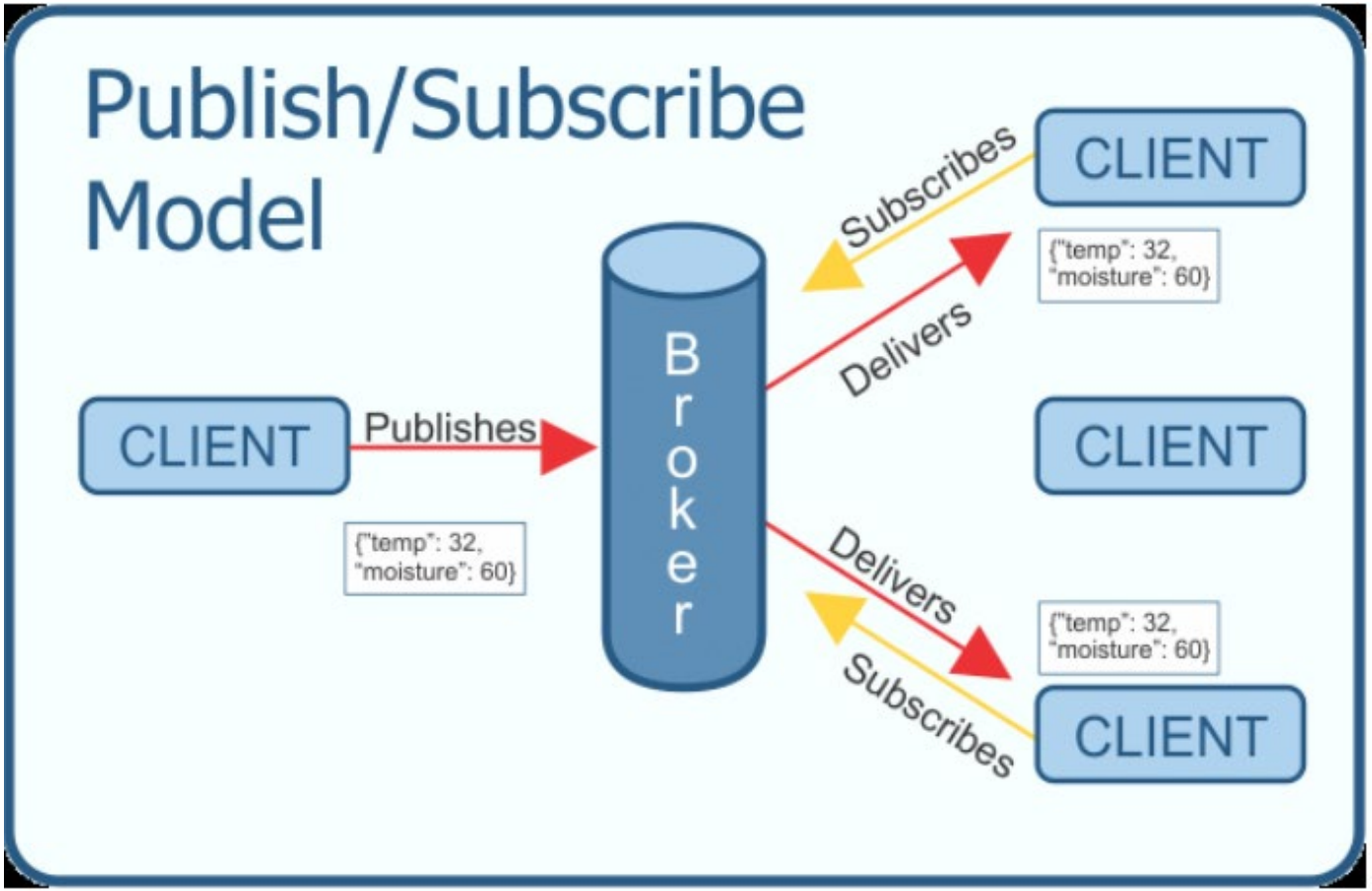


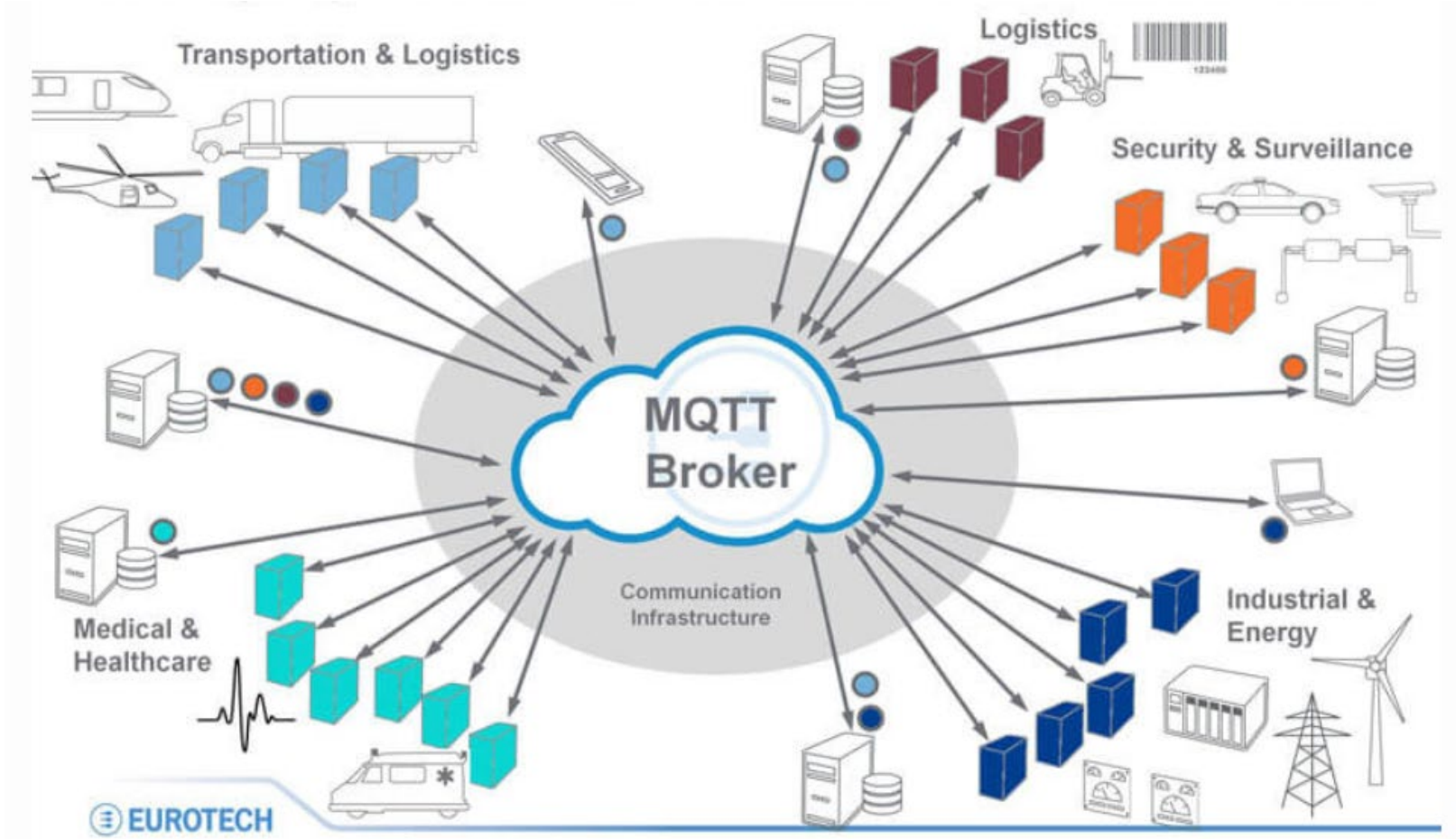
Publish-Subscribe Design Pattern

Publish-Subscribe Pattern



1. Publishers and Subscribers are decoupled.
   - Only need to be able to communicate with the broker/server
   - Publishers do not know about the existence of subscribers.
2. Can be a publisher and subscriber at the same time
3. No guarantee of delivery of messages to intended subscriber


Publish/Subscribe Model
CLIENT
Publishes
{"temp": 32, "moisture": 60}
Broker
Subscribes
Delivers
Delivers
Subscribes
CLIENT
{"temp": 32, "moisture": 60}
CLIENT
CLIENT
{"temp": 32, "moisture": 60}

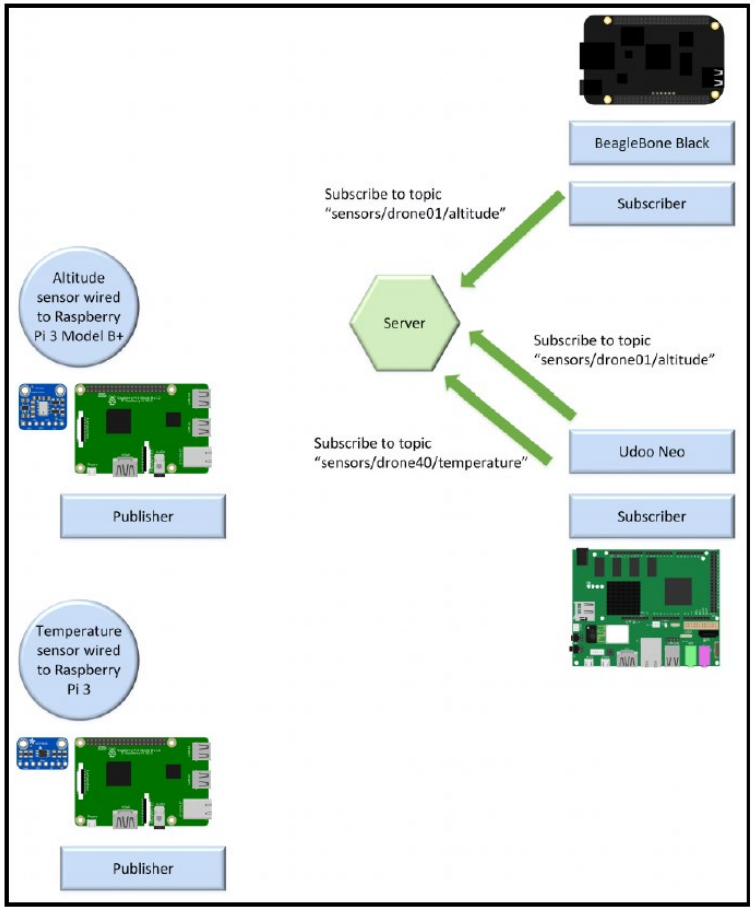

Transportation & Logistics
Logistics
Security & Surveillance
MQTT
Broker
Communication
Infrastructure
Medical &
Healthcare
Industrial &
Energy
EUROTECH

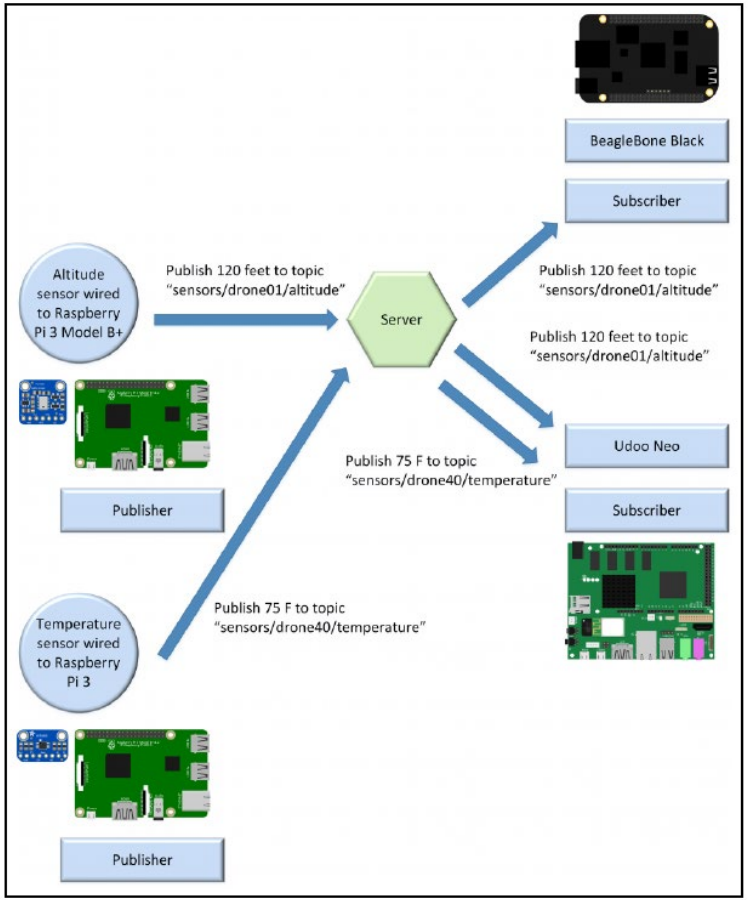
Message Queuing Telemetry Transport

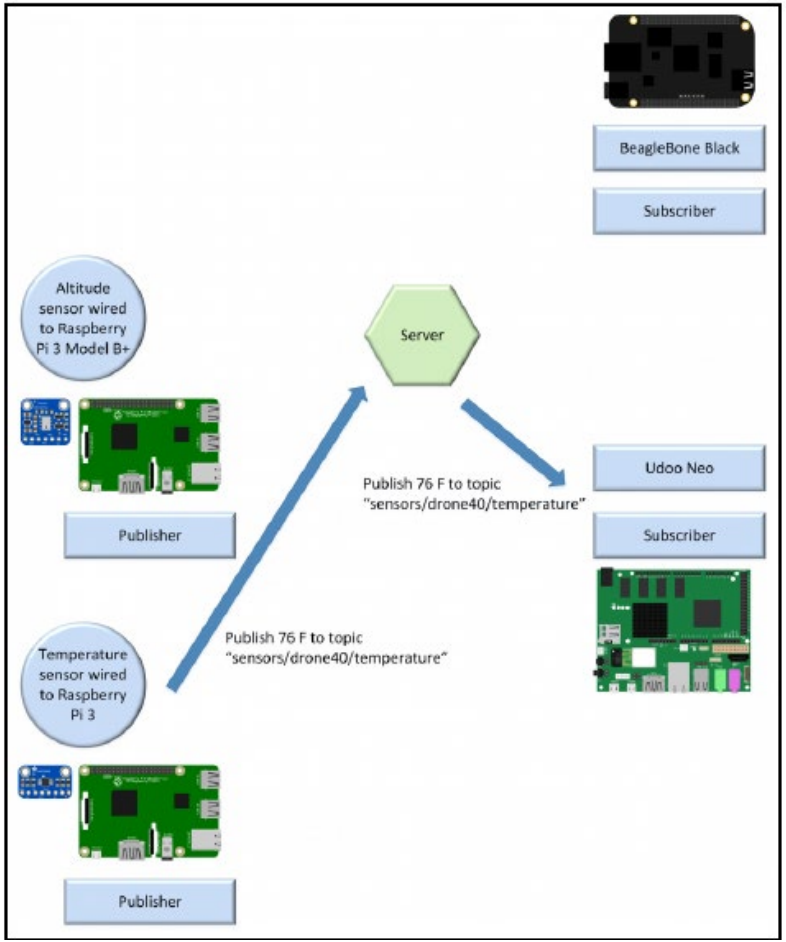
Message Filtering

Message Filtering

1. Topics are used to distribute messages to interested subscribers
   - item1/foo
   - item1/bar
2. Subscribers to item1/foo would not see messages on item1/bar
3. Topic organization is arbitrary and dynamic


BeagleBone Black
Subscriber
Subscribe to topic
"sensors/drone01/altitude"
Server
Subscribe to topic
"sensors/drone01/altitude"
Subscribe to topic
"sensors/drone40/temperature"
Udoo Neo
Subscriber
Altitude sensor wired to Raspberry Pi 3 Model B+
Publisher
Temperature sensor wired to Raspberry Pi 3
Publisher


BeagleBone Black
Subscriber
Altitude sensor wired to Raspberry Pi 3 Model B+
Publish 120 feet to topic "sensors/drone01/altitude"
Server
Publish 120 feet to topic "sensors/drone01/altitude"
Publish 120 feet to topic "sensors/drone01/altitude"
Udoo Neo
Publish 75 F to topic "sensors/drone40/temperature"
Subscriber
Publisher
Temperature sensor wired to Raspberry Pi 3
Publish 75 F to topic "sensors/drone40/temperature"
Publisher


BeagleBone Black
Subscriber
Altitude sensor wired to Raspberry Pi 3 Model B+
Server
Publish 76 F to topic "sensors/drone40/temperature"
Udoo Neo
Subscriber
Publisher
Temperature sensor wired to Raspberry Pi 3
Publish 76 F to topic "sensors/drone40/temperature"
Publisher

MQTT

Topic Wildcards

Single Level: +

1. Replaces one topic level in subscription

Subscription

Message received

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

https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices/

Multi Level:

1. Replaces everything at this level
2. Must be at the end of the topic

Subscription

Message received

myhome / groundfloor / livingroom / temperature
myhome / groundfloor / kitchen / temperature
myhome / groundfloor / kitchen / brightness
myhome / firstfloor / kitchen / temperature

https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices/

Reserved Topics: $

1. Topics beginning with $ are reserved for the broker
2. Not officially standardized. Potential Examples

$SYS/broker/clients/connected

$SYS/broker/clients/disconnected

$SYS/broker/clients/total

$SYS/broker/messages/sent

$SYS/broker/uptime

MQTT

Best Practices for Topics

Topic Best Practices

1. Do not use a leading slash on a topic:
   - /myhome/groundfloor/livingroom (unnecessary topic level at the beginning)
   - myhome/groundfloor/livingroom (preferred)
2. Only ASCII characters and no spaces
3. Specific topics rather than general
   - myhome/livingroom (general)
   - myhome/livingroom/humidity (specific)

https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices/

MQTT

Command Line Tools

Mosquitto

1. Open-source Implementation of MQTT Protocol
2. Broker/Clients
3. Bindings for many languages including Python and C.
4. Managed by Eclipse Foundation
5. Window/Linux/OSX

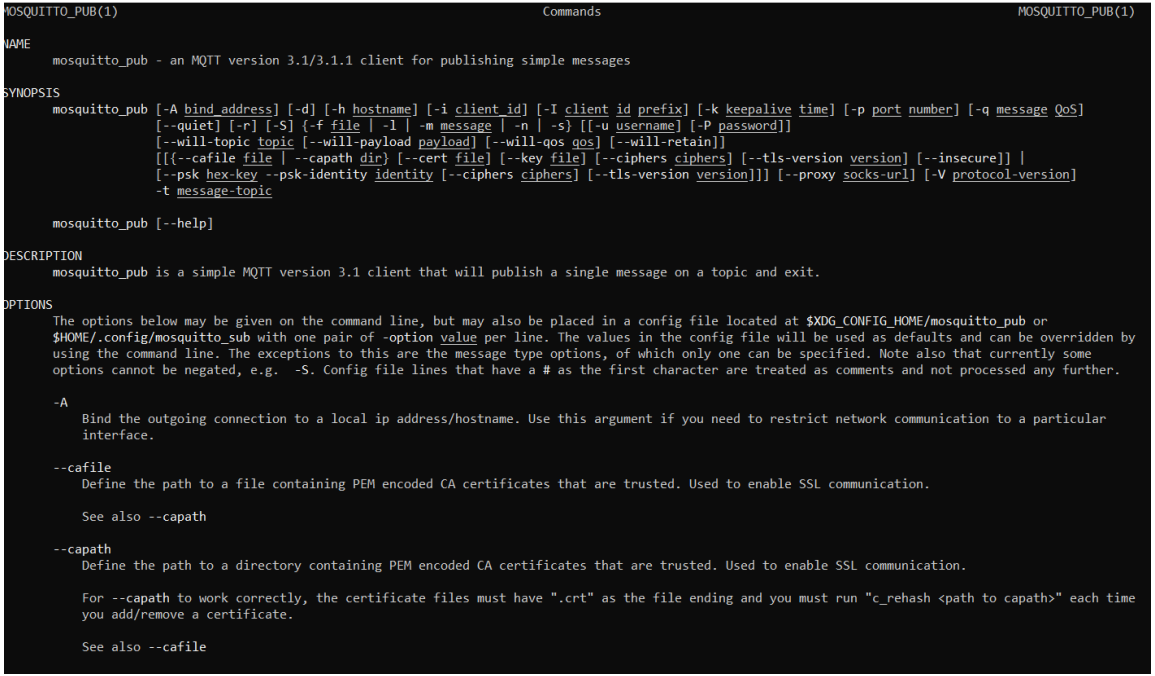
Subscriber

1. mosquitto_sub
   - (-h) pivot.iuiot.org
   - (-t) topic

```
MOSQUITTO_SUB(1)                                Commands                                MOSQUITTO_SUB(1)

NAME
       mosquitto_sub - an MQTT version 3.1 client for subscribing to topics

SYNOPSIS
       mosquitto_sub [-A bind_address] [-c] [-C msg count] [-d] [-h hostname] [-i client_id] [-I client id prefix]
                     [-k keepalive time] [-p port number] [-q message QoS] [-R] [-S] [-N] [--quiet] [-v]
                     [[-u username] [-P password]]
                     [--will-topic topic [--will-payload payload] [--will-qos qos] [--will-retain]]
                     [[{--cafile file | --capath dir} [--cert file] [--key file] [--tls-version version] [--insecure]]
                     | [--psk hex-key --psk-identity identity [--tls-version version]]] [--proxy socks-url]
                     [-V protocol-version] [-T filter-out...] -t message-topic...

       mosquitto_sub [--help]

DESCRIPTION
       mosquitto_sub is a simple MQTT version 3.1 client that will subscribe to a topic and print the messages that
       it receives.

OPTIONS
       The options below may be given on the command line, but may also be placed in a config file located at
       $XDG_CONFIG_HOME/mosquitto_sub or $HOME/.config/mosquitto_sub with one pair of -option value per line. The
       values in the config file will be used as defaults and can be overridden by using the command line. The
       exceptions to this are -t and -T, which if given in the config file will not be overridden. Note also that
       currently some options cannot be negated, e.g.  -S. Config file lines that have a # as the first character are
       treated as comments and not processed any further.

       -A
           Bind the outgoing connection to a local ip address/hostname. Use this argument if you need to restrict
           network communication to a particular interface.

       -c, --disable-clean-session
           Disable the 'clean session' flag. This means that all of the subscriptions for the client will be
           maintained after it disconnects, along with subsequent QoS 1 and QoS 2 messages that arrive. When the
           client reconnects, it will receive all of the queued messages.

           If using this option, it is recommended that the client id is set manually with --id
```

Publisher

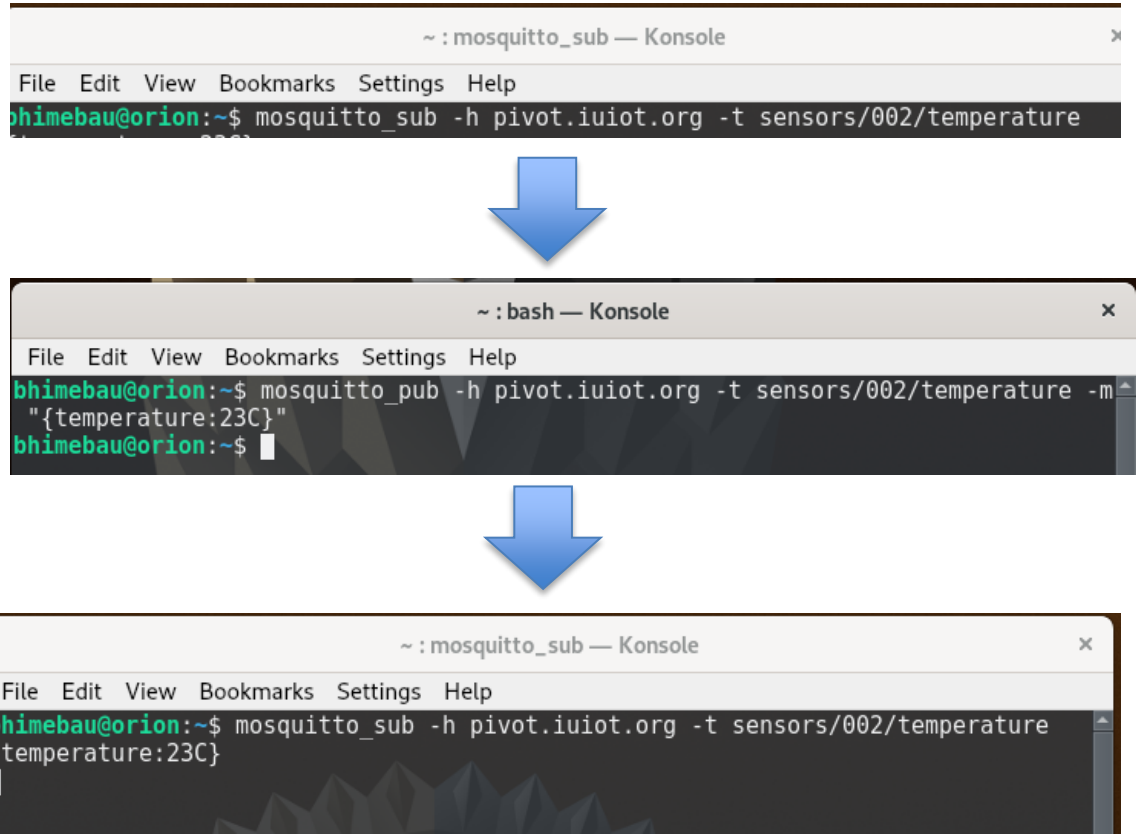
1. mosquitto_pub
   - (-h) pivot.iuiot.org
   - (-t) topic to publish onto
   - (-m) message



```
MOSQUITTO_PUB(1)                                Commands                                MOSQUITTO_PUB(1)

NAME
       mosquitto_pub - an MQTT version 3.1/3.1.1 client for publishing simple messages

SYNOPSIS
       mosquitto_pub [-A bind_address] [-d] [-h hostname] [-i client_id] [-I client id prefix] [-k keepalive time] [-p port number] [-q message QoS]
                     [--quiet] [-r] [-S] {-f file | -l | -m message | -n | -s} [[-u username] [-P password]]
                     [--will-topic topic [--will-payload payload] [--will-qos qos] [--will-retain]]
                     [[{--cafile file | --capath dir} [--cert file] [--key file] [--ciphers ciphers] [--tls-version version] [--insecure]] |
                     [--psk hex-key --psk-identity identity [--ciphers ciphers] [--tls-version version]]] [--proxy socks-url] [-V protocol-version]
                     -t message-topic

       mosquitto_pub [--help]

DESCRIPTION
       mosquitto_pub is a simple MQTT version 3.1 client that will publish a single message on a topic and exit.

OPTIONS
       The options below may be given on the command line, but may also be placed in a config file located at $XDG_CONFIG_HOME/mosquitto_pub or
       $HOME/.config/mosquitto_sub with one pair of -option value per line. The values in the config file will be used as defaults and can be overridden by
       using the command line. The exceptions to this are the message type options, of which only one can be specified. Note also that currently some
       options cannot be negated, e.g.  -S. Config file lines that have a # as the first character are treated as comments and not processed any further.

       -A
           Bind the outgoing connection to a local ip address/hostname. Use this argument if you need to restrict network communication to a particular
           interface.

       --cafile
           Define the path to a file containing PEM encoded CA certificates that are trusted. Used to enable SSL communication.

           See also --capath

       --capath
           Define the path to a directory containing PEM encoded CA certificates that are trusted. Used to enable SSL communication.

           For --capath to work correctly, the certificate files must have ".crt" as the file ending and you must run "c_rehash <path to capath>" each time
           you add/remove a certificate.

           See also --cafile
```

Example

Same window



MQTT

Python Interface

```
#!/usr/bin/env python3

import paho.mqtt.client as mqtt

def on_publish(client,userdata,result):
    print("data published")

client = mqtt.Client()
client.on_publish=on_publish
client.connect("pivot.iuiot.org")
client.publish("sensors/002/temperature","{temperature:23C}")
```

```
#!/usr/bin/env python3

import paho.mqtt.client as mqtt

def on_publish(client,userdata,result):
    print("data published")

client = mqtt.Client()
client.on_publish=on_publish
client.connect("pivot.iuiot.org")
client.publish("sensors/002/temperature","{temperature:23C}")
```

```
#!/usr/bin/env python3

import paho.mqtt.client as mqtt

def on_publish(client,userdata,result):
    print("data published")

client = mqtt.Client()
client.on_publish=on_publish
client.connect("pivot.iuiot.org")
client.publish("sensors/002/temperature","{temperature:23C}")
```

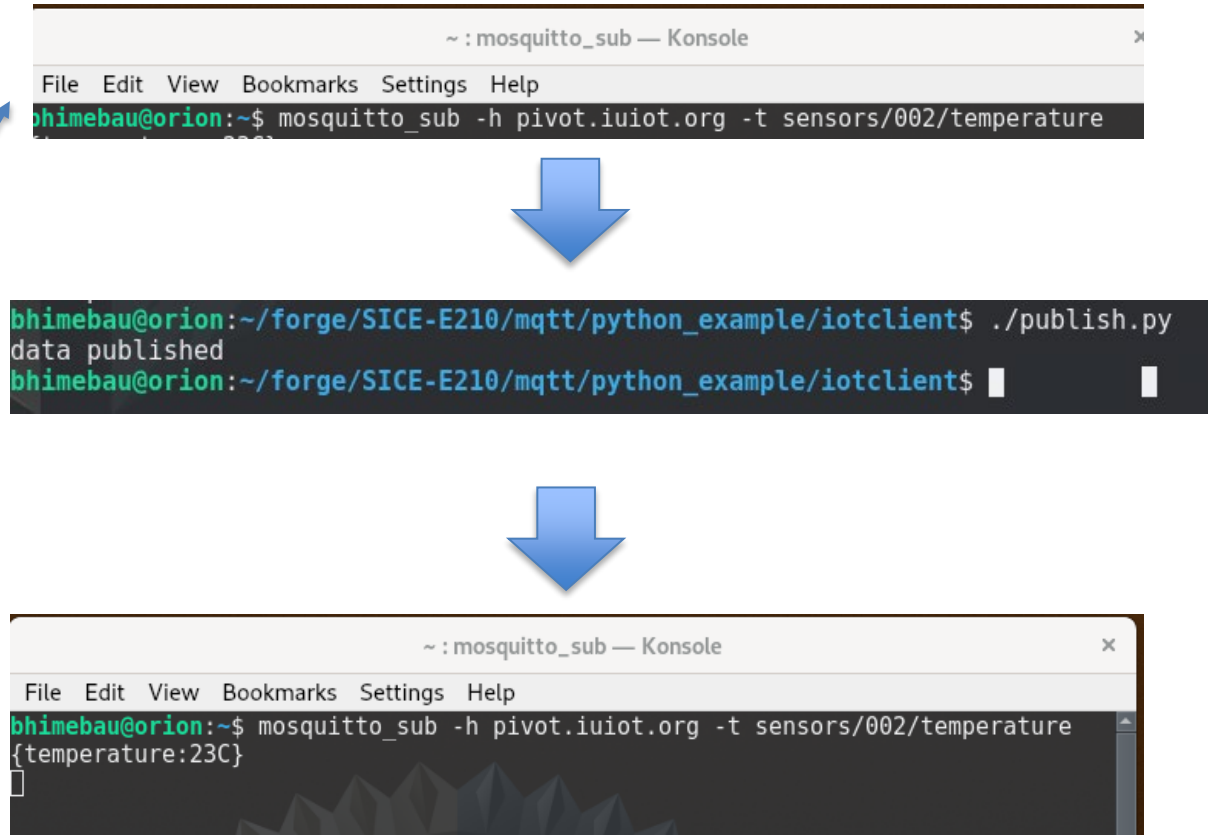
```
#!/usr/bin/env python3

import paho.mqtt.client as mqtt

def on_publish(client,userdata,result):
    print("data published")

client = mqtt.Client()
client.on_publish=on_publish
client.connect("pivot.iuiot.org")
client.publish("sensors/002/temperature","{temperature:23C}")
```

```
#!/usr/bin/env python3

import paho.mqtt.client as mqtt

def on_publish(client,userdata,result):
    print("data published")

client = mqtt.Client()
client.on_publish=on_publish
client.connect("pivot.iuiot.org")
client.publish("sensors/002/temperature","{temperature:23C}")
```

```
#!/usr/bin/env python3

import paho.mqtt.client as mqtt

def on_publish(client,userdata,result):
    print("data published")

client = mqtt.Client()
client.on_publish=on_publish
client.connect("pivot.iuiot.org")
client.publish("sensors/002/temperature","{temperature:23C}")
```

Example

Same window

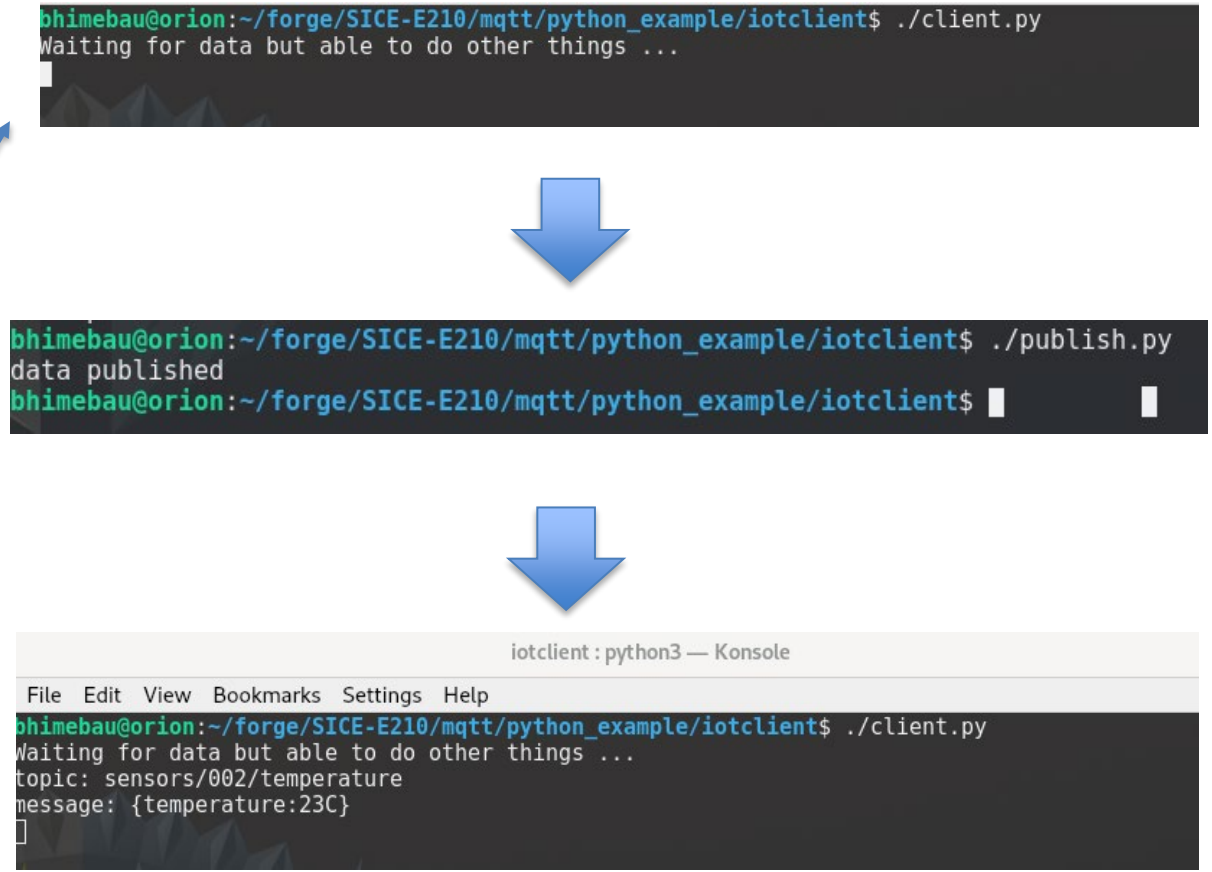


```
#!/usr/bin/env python3

import paho.mqtt.client as mqtt

def on_message(client, userdata, message):
    print("topic:", message.topic)
    print("message:", message.payload.decode('UTF-8'))

def on_connect(client,userdata,flags,rc):
    client.subscribe("sensors/002/temperature")

client = mqtt.Client()
client.on_message=on_message
client.on_connect=on_connect
client.connect("pivot.iuiot.org")
client.loop_start()
print("Waiting for data but able to do other things ... ")
while(1):
    pass
```

```
#!/usr/bin/env python3

import paho.mqtt.client as mqtt

def on_message(client, userdata, message):
    print("topic:", message.topic)
    print("message:", message.payload.decode('UTF-8'))

def on_connect(client,userdata,flags,rc):
    client.subscribe("sensors/002/temperature")

client = mqtt.Client()
client.on_message=on_message
client.on_connect=on_connect
client.connect("pivot.iuiot.org")
client.loop_start()
print("Waiting for data but able to do other things ... ")
while(1):
    pass
```

```
#!/usr/bin/env python3

import paho.mqtt.client as mqtt

def on_message(client, userdata, message):
    print("topic:", message.topic)
    print("message:", message.payload.decode('UTF-8'))

def on_connect(client,userdata,flags,rc):
    client.subscribe("sensors/002/temperature")

client = mqtt.Client()
client.on_message=on_message
client.on_connect=on_connect
client.connect("pivot.iuiot.org")
client.loop_start()
print("Waiting for data but able to do other things ... ")
while(1):
    pass
```

```
#!/usr/bin/env python3

import paho.mqtt.client as mqtt

def on_message(client, userdata, message):
    print("topic:", message.topic)
    print("message:", message.payload.decode('UTF-8'))

def on_connect(client,userdata,flags,rc):
    client.subscribe("sensors/002/temperature")

client = mqtt.Client()
client.on_message=on_message
client.on_connect=on_connect
client.connect("pivot.iuiot.org")
client.loop_start()
print("Waiting for data but able to do other things ... ")
while(1):
    pass
```

Example

Same window