Contents

[Background 1](#_Toc190120782)

[Key Aspects of MQTT 1](#_Toc190120783)

[MQTT topology and the Publish/Subscribe Model: 3](#_Toc190120784)

[Topic Naming scheme: 3](#_Toc190120785)

[Example of MQTT Communication 4](#_Toc190120786)

[The scenario where 445L Lab Temp Changes 6](#_Toc190120787)

[How to approach Lab 4 7](#_Toc190120788)

[Useful MQTT tutorials and projects: 8](#_Toc190120789)

# Background

Wi-Fi is the name given to devices that communicate wirelessly employing the IEEE 802.11 standard. They typically operate in the 2.4 GHz Industrial-Scientific-Medical (ISM) unlicensed band that is shared with ZigBee (IEEE 802.15.4) and Bluetooth (IEEE 802.15.1) communications. The IEEE 802.11 standard describes how information is represented via radio frequency signals, i.e., the physical or PHY layer, and how communications are formatted and controlled, i.e., the media access control or MAC layer. Communication requirements beyond the PHY and MAC, e.g., flow control, error recovery, and routing, are handled at higher levels of the internet stack.

# Key Aspects of MQTT

MQTT ([which originally stood for nothing](https://www.oasis-open.org/committees/download.php/49028/OASIS_MQTT_TC_minutes_25042013.pdf), but was backronymed to mean Message Queuing Telemetry Transport) is message passing mechanism that uses a centralized broker (server) to receive published data and send the data to any number of subscribers as shown below.

A diagram of a diagram

AI-generated content may be incorrect.

Figure 1: MQTT Publish-Subscribe Model

MQTT is a lightweight messaging protocol designed for low-bandwidth, high-latency, and unreliable networks. Developed in 1999 by IBM, MQTT is now a widely used open standard. MQTT is useful as it:

* Organizes data at the protocol level
* Coordinates actions between IOT devices
* Allows easy centralization of data to user-facing dashboards
* Sends connected clients only the data they need
* Can ensure a Quality Of Service (QOS) of data sent by clients
* Allows all clients to publish (broadcast) and subscribe (receive).
* Does not have the overhead of have addresses for clients like in email systems; since messages are not sent directly to clients, the client can be simplified.

Some key points about MQTT are:

* Clients become subscribers to “topic(s)” to get messages published to the broker about the “topic”
* Messages are published to a broker about a particular “topic”. In this Lab all topics MUST begin with your EID. A topic example is rg7677/b2w/hour where rg7677 is the student EID and b2w is a subtopic indicating that the message is from the LaunchPad board to the Web Application and the hour is a sub-subtopic that is current hour on the board.
* The job of an MQTT broker is to collect and sort messages based on the “topic”, so that the broker can distribute messages to subscribers of the “topic”.
* There is no direct connection between a publisher and subscriber.
* MQTT brokers do NOT normally store messages. All messages are sent via TCP which guarantees delivery (vs UDP).
* The publisher dictates the data format. The subscriber must know the format to decipher the data.

MQTT topology and the Publish/Subscribe Model:

A diagram of a broker

AI-generated content may be incorrect.

Figure 2 MQTT Client-Broker topology

MQTT includes ONLY broker to client/client to broker communications. MQTT networks have one broker which facilitates communication between many clients.

MQTT uses a Publish/Subscribe Model. In this model, data movement is controlled by topics. Clients decide what data they receive by deciding what topics they subscribe to. Clients can influence where data is sent by choosing what topic they attach to the data they publish. The messages sent and received always consist of a topic & data.

Although the broker is responsible for relaying data, it has no discretion to decide where data goes. The broker will relay a published message from any client to all clients that are subscribed to the topic of the message.

# Topic Naming scheme:

An MQTT topic is a hierarchical string identifier that categorizes messages being published/subscribed to within the MQTT system. Topics are represented as strings and are organized hierarchically using forward slashes (‘/’). For example, a topic could be

"sensors/temperature/EER/floor1/embeddedlab."

or

"EER/floor1/embeddedlab/temperature"

How topics are defined are up to you. As long as they follow the typical hierarchical format.

# Example of MQTT Communication

Here is an example of a typical MQTT communication between clients and how

the broker routes it.

1. An MQTT client publishes a message to a specific topic on the MQTT broker.
2. The MQTT broker receives the published message and examines the topic.
3. The broker checks its subscription database to determine which MQTT client have subscribed to that topic.
4. The broker forwards the message to all subscribed clients
5. Subscribed MQTT clients receive the message and can take action based on its content.

Now lets imagine a scenario where we have a IoT setup in the EER. We have three sensors which are their very own internet conected devices located in the EER. We have a temperature sensor in the 445L lab room, a thermostat located in the 319K room (let’s imagine the thermostat controls the A.C for the entirety of floor 1), and a faculty member has a mobile app that allows him to control the thermostat or see the readings from the sensors.

Here we have three clients:

* 445L Temperature Sensor
  + Can read temperature in the 445L room
* 319K Lab Thermostat
  + Connected via wire to the A.C, can control the A.C for the entirety of floor 1.
  + Has its own temperature sensor that allows it to read the temperature in the room it is in (the 319K lab in this case)
  + Has a temperature the thermostat can be set too. If the temperature
* in any of the rooms dip/rise above this temperature. The A.C is enabled to heat/cool
  + Has a mode for the A.C. The A.C can either be off, set to cool, or set to heat.
  + Can interface with external temperature sensors (such as the 445L temperature sensor)
  + This closely resembles popular IoT thermostats such as the Nest or Ecobee
* Mobile App
  + Allows you to control the thermostat (mode, set temperature)
  + Read the temperature readings across all temperature sensors

All of these clients are connected to one central server (The MQTT Broker) to facilitate communication. This server could be for examples McDermott MQTT server in his office, or a public/cloud based MQTT Server.

A diagram of a business flow

AI-generated content may be incorrect.

Figure 3 Client-Broker setup of the Example

Now lets try to analyze what information each client provides, and what information each client would want from another client.

* 445L Temperature Sensor
  + Reads the 445l lab room temperature, publishes it.
* 319K Lab Thermostat
  + Reads the temperature in the 319K room, can act on this reading (enable the A.C for floor 1). Can also publish it (for the mobile app)
  + Is interested in the temperature in the 445L room, can act on this reading (enable the A.C for floor 1)
  + Has a temperature that can be set to enable cooling/heat (acting on this data) It can publish this temperature (for the mobile app), and it can also have this changed externally (from the mobile app)
  + Has a A.C mode (off/cool/heat). It acts on this mode (enable the A.C for cool/hot if the outside temperature changes). It can publish this data (so the mobile app knows the current A.C mode), it can also have this changed externally (from the mobile app)
* Mobile App
  + Can set the A.C mode from the mobile app. Changes to this can be made from app or thermostat
  + Can set the thermostat temperature. Changes can be made from app or thermostat.
  + Can read all temperature sensor readings.

From this lets define the following topics, and these publishers and subscribers

* /EER/floor1/445Lab/temperature
  + Publishers:
    - 445L Temperature Sensor
  + Subscribers:
    - 319K Thermostat
    - Mobile App
* /EER/floor1/319KLab/temperature
  + Publishers:
    - 319K Thermostat
  + Subscribers:
    - Mobile App
* /EER/floor1/thermostat/set\_temp
  + Publishers:
    - 319K Thermostat
    - Mobile App
  + Subscribers:
    - 319K Thermostat
    - Mobile App
* /EER/floor1/thermostat/mode
  + Publishers:
    - 319K Thermostat
    - Mobile App
  + Subscribers:
    - 319K Thermostat
    - Mobile App

# The scenario where 445L Lab Temp Changes

If the 445L lab room temperature changes. The sensor publishes to its topic.

A diagram of a diagram

AI-generated content may be incorrect.

Figure 4: 445L Temperature Sensor publishes change

The thermostat and mobile app are subscribers, thus the broker forwards the message.

A diagram of a system

AI-generated content may be incorrect.

Figure 5: Subscribers are notified

# How to approach Lab 4

From the perspective of the lab we have a single broker and two clients:

* Your Alarm Clock
* The web dashboard

From the perspective of the free broker service however, it has your alarm clock and web dashboard as a client. As well as the alarm clock and web dashboard of everyone else in the class (and anyone else using the service). So when you define your topics make sure the topic string is unique. For example if everyone used the same topic, lets say “alarm/current\_time”, you would be able to change the time of everyone’s alarm clock. So to make unique topic names, your topic should start with your eid, ex. “eid123/alarm/current\_time”.

Here is how we reccomend approaching this lab to avoid stress and headaches. First things first, don’t try integrating lab3 right away, instead get topics working. Using the starter code, do the following:

1. ESP Bring up, the ESP8266 should be able to connect to a network and to the broker. This should be done in prep. Come to your TA’s if you were unable to do this.
2. Define your topics, and the subscriber/publisher relationships.
3. Have both of you open the web dashboard. Have one person subscribe to a topic, and have the other publish to the topic. The subscriber should be able to receive a message when this happens.
4. Modify your TM4C starter code so you can communicate with the ESP. The TM4C communicates with the ESP over UART. You can observe what is being sent to the ESP via USB-Serial between your PC and the TM4C using a serial terminal.
5. Test, can you publish to a topic from the TM4C, do you see this change reflected on the web dashboard?
6. Test, can you publish to a topic from the web dashboard, do you see this change reflected on the TM4C?
7. Now that you can publish/subscribe to topics. Now integrate your Lab 3 Code.

If you need any help, please email us or come to our office hours.

# Useful MQTT tutorials and projects:

* <http://www.steves-internet-guide.com/mqtt-works/>
* <http://www.steves-internet-guide.com/download/mqtt-study-guide/>
* <http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/>
* <http://www.steves-internet-guide.com/into-mqtt-python-client/>
* <http://www.steves-internet-guide.com/mqtt-websockets/>
* <http://www.steves-internet-guide.com/using-javascript-mqtt-client-websockets/>
* <http://www.steves-internet-guide.com/mosquitto-broker/>
* <http://www.steves-internet-guide.com/install-mosquitto-broker/>
* <http://www.steves-internet-guide.com/mqtt-broker-testing/>
* <http://www.steves-internet-guide.com/monitoring-mqtt-brokers/>