

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



BÁO CÁO CUỐI KỲ

“Message Queuing Telemetry Transport”

GVHD: ThS. Đặng Lê Bảo Chương

Thành viên nhóm 10:

Trần Thúy Anh – 20521085

Trương Đình Trọng Thanh – 20520766

Lớp: NT131.O11.MMCL

MỤC LỤC

	Trang
I. Giới thiệu.....	1
1.1 Tổng quan	1
1.2 Lịch sử.....	2
1.3 OASIS	3
II. Cơ sở lý thuyết	4
2.1 MQTT là gì?.....	4
2.2 Thành phần.....	4
2.2.1 MQTT Client.....	4
2.2.2 MQTT Broker.....	5
2.3 Kiến trúc.....	5
2.3.1 Publish, Subscribe	7
2.3.2 Topic.....	7
2.3.3 Retain.....	8
2.3.4 QoS	9
2.3.5 Keep-Alive	10
2.4 Cấu trúc gói tin.....	10
2.5 Tính bảo mật	12
2.6 Ưu và nhược điểm của MQTT	13
2.6.1 Ưu điểm.....	13
2.6.2 Nhược điểm	14
III. Thực nghiệm	16
3.1 Ứng dụng thực tế.....	16
3.1.1 Nhà thông minh	17
3.1.2 Nông nghiệp	18
3.1.3 Tự động hóa trong công nghiệp.....	19
3.2 Kịch bản	20
IV. Phân tích	21

4.1 Arduino	21
4.2 MQTT Explorer	22
4.3 ESP8266.....	23
V. Kết quả.....	26
VI. Kết luận	26
Tài liệu tham khảo	28

DANH SÁCH TỪ VIẾT TẮT

MQTT :	Message Queuing Telemetry Transport
QC :	Quality control
IoT :	Internet of Things
OASIS :	Organization for the Advancement of Structured Information Standards
TCP :	Transmission Control Protocol
SSL :	Secure Sockets Layer
TLS :	Transport Layer Security
M2M :	Machine-to-machine
HTTP :	HyperText Transfer Protocol
QoS :	Quality of Service

DANH SÁCH HÌNH ẢNH

	Trang
Hình 1.1: Tổng quan về MQTT	1
Hình 1.2: Lịch sử phát triển MQTT	2
Hình 1.3: Logo tập đoàn OASIS	3
Hình 2.1: Kiến trúc hoạt động của MQTT	5
Hình 2.3: Mô hình kết nối giữa Client-Broker	6
Hình 2.4: Sơ đồ tổng thể Publish - Subscribe	7
Hình 2.5: QoS 0	9
Hình 2.6: QoS 1	9
Hình 2.7: QoS 2	10
Hình 3.1: Ứng dụng của MQTT	16
Hình 3.2: MQTT trong SmartHome	17
Hình 3.3: Mô hình đo độ ẩm đất bằng ESP8266	18
Hình 3.4: Mô hình tổng quát về MQTT trong công nghiệp	19
Hình 3.5: Mô hình thực nghiệm	20
Hình 4.1: Thư viện PubSubClient	21
Hình 4.2: Giao diện thiết lập kết nối	23
Hình 4.3: Mô-đun ESP8266	24
Hình 5.1: Gửi thông điệp sang ESP8266	26

DANH SÁCH BẢNG

Trang

Bảng 1: Cấu trúc gói tin.....11

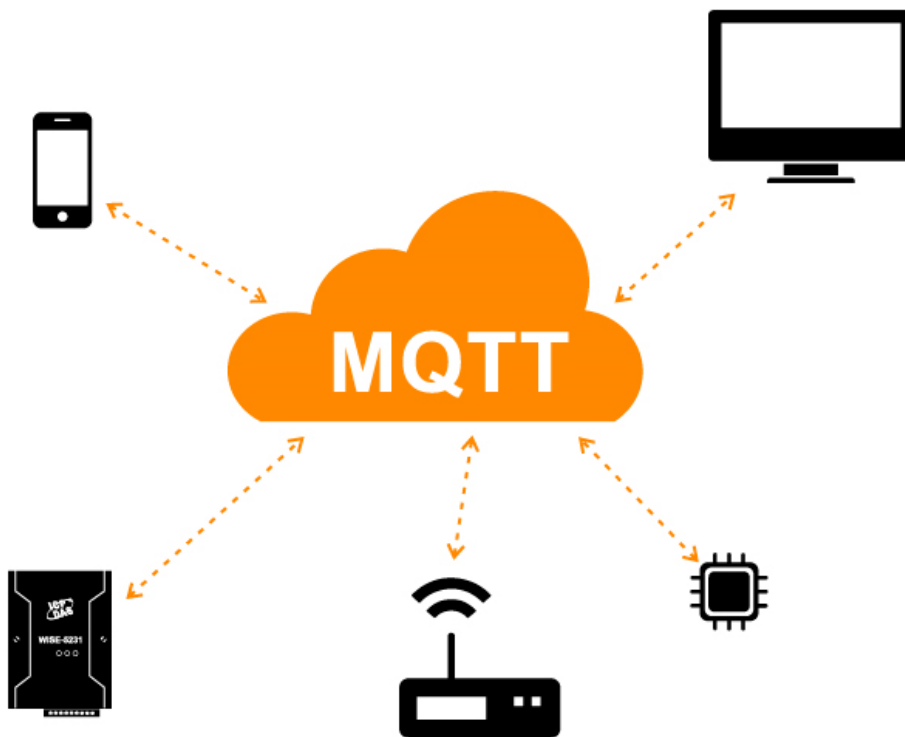
Bảng 2: Các loại gói tin12

I. Giới thiệu

1.1 Tổng quan

Các nhà sản xuất ngày nay phải hoạt động với tốc độ chóng mặt để bắt kịp với nhu cầu luôn thay đổi của khách hàng, xu hướng thị trường và các đối thủ cạnh tranh toàn cầu trong một thị trường phát triển nhanh chóng. Nhiều thách thức truyền thống, chẳng hạn như tăng chi phí, thời gian ngừng hoạt động ngoài kế hoạch, kiểm tra chất lượng (QC), yêu cầu kinh doanh đang phát triển và cơ sở hạ tầng cũ kỹ, tiếp tục cản trở tiến độ. Ngày nay, các tổ chức trong nhiều ngành khác nhau đang sử dụng IoT để hoạt động hiệu quả hơn, hiểu rõ về khách hàng để cung cấp dịch vụ, gia tăng giá trị của doanh nghiệp.

Internet vạn vật hay IoT là mạng lưới các thiết bị có liên quan với nhau, kết nối và trao đổi dữ liệu với các thiết bị IoT khác và đám mây. Các thiết bị IoT thường được tích hợp công nghệ như cảm biến và phần mềm và có thể bao gồm các máy móc cơ khí, kỹ thuật số và các đối tượng tiêu dùng. Ngày càng có nhiều tổ chức trong nhiều ngành công nghiệp sử dụng IoT để hoạt động hiệu quả hơn, cung cấp dịch vụ khách hàng nâng cao, cải thiện việc ra quyết định và tăng giá trị của doanh nghiệp. Với IoT, dữ liệu có thể được truyền qua mạng mà không yêu cầu tương tác giữa người với người hoặc giữa người với máy tính.



Hình 1.1: Tổng quan về MQTT

MQTT là viết tắt của MQ Telemetry Transport. MQTT là một giao thức nhắn tin tiêu chuẩn OASIS cho Internet of Things (IoT), được thiết kế như một phương tiện nhắn tin xuất bản/đăng ký (publish/subscribe) cực kỳ nhẹ, lý tưởng để kết nối các thiết bị từ xa với dấu chân mã nhỏ và băng thông mạng tối thiểu. MQTT ngày nay được sử dụng trong nhiều ngành công nghiệp, chẳng hạn như ô tô, sản xuất, viễn thông, dầu khí, v.v.

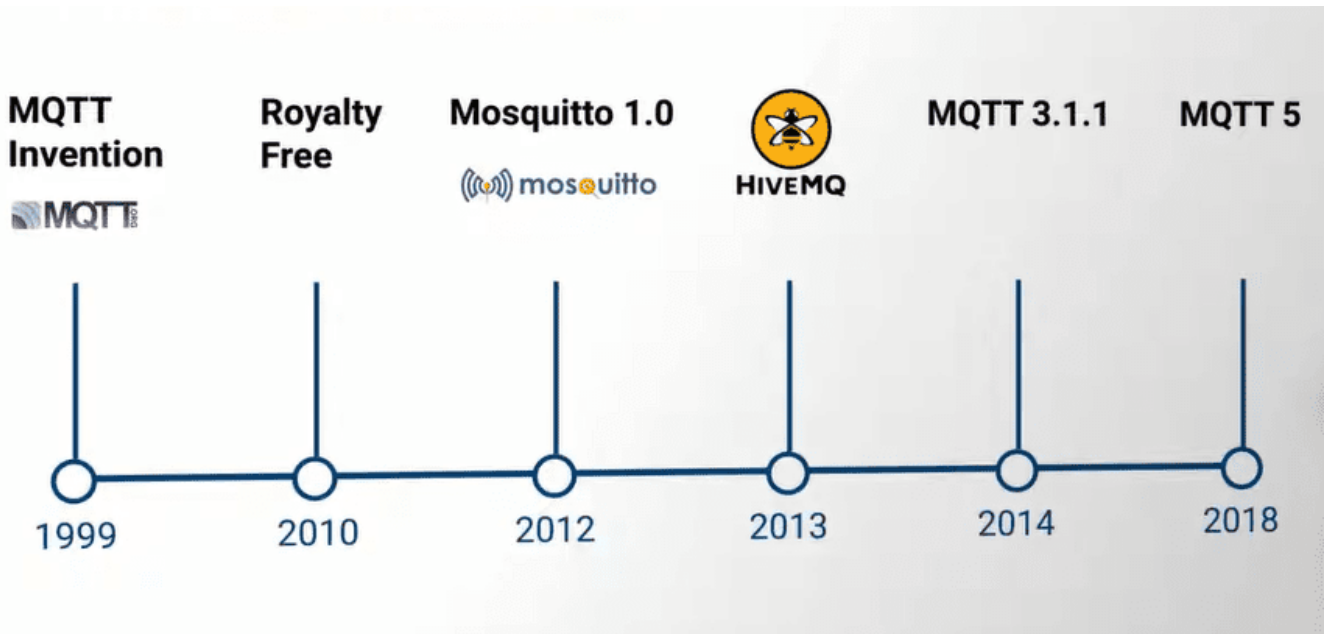
1.2 Lịch sử

MQTT được tạo ra vào năm 1999 bởi Tiến sĩ Andy Stanford-Clark của IBM và Arlen Nipper của Arcom (nay là Eurotech). MQTT được tạo ra với mục đích cung cấp hiệu quả chi phí và độ tin cậy để kết nối thiết bị giám sát được sử dụng trong ngành dầu khí với các máy chủ giám sát từ xa. Khi gặp thách thức với việc tìm cách đẩy dữ liệu từ các cảm biến đường ống trên sa mạc sang các hệ thống điều khiển, giám sát và thu thập dữ liệu bên ngoài (SCADA), họ đã quyết định sử dụng cấu trúc liên kết xuất bản/đăng ký dựa trên TCP/IP theo sự kiện để truyền dữ liệu và giảm chi phí đầu tư.

IBM ban đầu đặt tên MQTT là “MQ Telemetry Transport” khi họ sử dụng nó trong nội bộ. Họ lấy tên từ MQSeries của IBM, trong đó “MQ” là viết tắt của “Message Queuing”. “Sau khi được phát hành rộng rãi, việc sử dụng giao thức này nhanh chóng mở rộng ra ngoài phép đo từ xa. Sau này nó được gọi là MQTT - một từ viết tắt không còn ý nghĩa gì nữa. Kể từ đó, tất cả các phiên bản tiếp theo đã gọi giao thức này đơn giản là “MQTT”. Mặc dù MQTT vẫn có mối quan hệ chặt chẽ với IBM, nhưng giờ đây nó là một giao thức mở được quản lý bởi Tổ chức vì sự tiến bộ của các tiêu chuẩn thông tin có cấu trúc (OASIS). MQTT không phải là một phần của IBM MQSeries; Và kể từ phiên bản 7.1, nó có sẵn trong WebSphere MQ. MQTT trước đây được gọi là giao thức SCADA, MQ Integrator SCADA Device Protocol (MQIsdp) và WebSphere MQTT (WMQTT). Tuy nhiên, tất cả các biến thể này đã không còn được sử dụng.

MQTT đã chính thức được phê duyệt như một tiêu chuẩn OASIS vào ngày 28 tháng 10 năm 2015. Vào cuối tháng 1 năm 2016, nó đã được Tổ chức Tiêu chuẩn hóa Quốc tế (ISO) chấp nhận làm tiêu chuẩn. Giao thức này liên tục được cải tiến và hiện hỗ trợ WebSockets, một giao thức khác cho phép giao tiếp hai chiều giữa khách hàng và MQTT Broker trong thời gian thực.

Năm 2013 OASIS phát hành MQTT phiên bản 3.1.1 và năm 2018 - MQTT phiên bản 5.



Hình 1.2: Lịch sử phát triển MQTT

1.3 OASIS



Hình 1.3: Logo tập đoàn OASIS

Được thành lập vào năm 1993 với tư cách là một tổ chức phi lợi nhuận, OASIS (Tổ chức vì sự tiến bộ của các tiêu chuẩn thông tin có cấu trúc) là một tập đoàn quốc tế phát triển các tiêu chuẩn mở cho Internet và các công nghệ liên quan. Nó đã phát triển nhiều tiêu chuẩn quan trọng cho các ngành như điện toán đám mây, bảo mật và IoT, bao gồm AMQP, SAML và DocBook. Quá trình tiêu chuẩn hóa mất khoảng một năm và vào ngày 29 tháng 10 năm 2014, MQTT đã trở thành tiêu chuẩn OASIS được chính thức phê duyệt.

Sự tham gia của OASIS vào MQTT rất quan trọng đối với sự thành công của nó với tư cách là một giao thức IoT được áp dụng rộng rãi. Là một tổ chức bên thứ ba, trung lập, OASIS đảm bảo rằng giao thức được duy trì như một tiêu chuẩn mở mà bất kỳ ai cũng có thể triển khai mà không phải trả phí cấp phép hoặc hạn chế về quyền sở hữu. Ngoài ra, OASIS cung cấp một diễn đàn để cộng đồng cùng nhau và cộng tác cải tiến giao thức, dẫn đến sự phát triển của MQTT 5, phiên bản mới nhất của giao thức với các tính năng mới để cải thiện độ tin cậy và khả năng mở rộng.

OASIS MQTT TC đang tạo ra một tiêu chuẩn cho Giao thức truyền tải từ xa xếp hàng tin nhắn tương thích với MQTT V3.1, cùng với các yêu cầu cải tiến, ví dụ sử dụng được ghi lại, các phương pháp hay nhất và hướng dẫn sử dụng các chủ đề MQTT với các cơ chế khám phá và đăng ký phổ biến. Tiêu chuẩn này hỗ trợ nhắn tin hai chiều để xử lý thống nhất cả tín hiệu và lệnh, gửi tin nhắn xác định, mức QoS cơ bản, các tình huống luôn/đôi khi được kết nối, khớp nối lỏng lẻo và khả năng mở rộng để hỗ trợ số lượng lớn thiết bị. Các ứng cử viên cho các cải tiến bao gồm mức độ ưu tiên và thời hạn sử dụng của tin nhắn, kiểu nhập tin nhắn, yêu cầu/tra lời và hết hạn đăng ký.

II. Cơ sở lý thuyết

2.1 MQTT là gì?

MQTT là viết tắt của Message Queuing Telemetry Transport. Đây là một giao thức nhắn tin nhẹ để sử dụng trong trường hợp khách hàng cần một dấu chân mã nhỏ và được kết nối với các mạng hoặc mạng không đáng tin cậy với tài nguyên băng thông hạn chế. Nó chủ yếu được sử dụng cho giao tiếp giữa máy với máy (M2M) hoặc các loại kết nối Internet of Things. Khi MQTT được thiết kế đơn giản, hiệu quả và không tốn nhiều tài nguyên. MQTT được tạo ra với mục đích gửi một lượng nhỏ dữ liệu qua các mạng không đáng tin cậy với băng thông và kết nối hạn chế. So với các giao thức khác, MQTT có dung lượng mã nhỏ, chi phí vận hành thấp và tiêu thụ điện năng thấp. Do chi phí gói tối thiểu, MQTT vượt trội trong việc truyền dữ liệu qua dây so với các giao thức như HTTP. Điều này cũng làm cho nó trở nên lý tưởng để sử dụng trong các thiết bị có sức mạnh xử lý, bộ nhớ và tuổi thọ pin hạn chế, chẳng hạn như cảm biến và các thiết bị IoT khác. Ví dụ: MQTT đang được sử dụng trong các cảm biến giao tiếp với MQTT Broker thông qua liên kết vệ tinh, SCADA, qua kết nối quay số không thường xuyên với các nhà cung cấp dịch vụ chăm sóc sức khỏe (thiết bị y tế) và trong một loạt các tình huống tự động hóa gia đình và thiết bị nhỏ. MQTT cũng lý tưởng cho các ứng dụng di động vì kích thước nhỏ, gói dữ liệu được giảm thiểu và phân phối thông tin hiệu quả đến một hoặc nhiều người nhận (người đăng ký).

2.2 Thành phần

2.2.1 MQTT Client

MQTT Client(Máy khách) là một ứng dụng hoặc thư viện được triển khai trên các thiết bị để tham gia vào mạng MQTT. MQTT Client có trách nhiệm gửi và nhận tin nhắn thông qua giao thức MQTT, thường thông qua các thủ tục như “publish” và “subscribe” (đăng ký nhận tin nhắn). MQTT Client có thể là thiết bị IoT, máy tính, điện thoại di động hoặc bất kỳ hệ thống nào có khả năng kết nối mạng và hỗ trợ giao thức MQTT. MQTT Client thường được triển khai trên nền tảng phần cứng hoặc phần mềm.

Thư viện MQTT Client là một mô-đun hoặc gói phần mềm triển khai giao thức MQTT và cung cấp giao diện cho các thiết bị hoặc ứng dụng giao tiếp bằng MQTT. Các thư viện này giúp việc thêm hỗ trợ MQTT vào ứng dụng hoặc thiết bị trở nên dễ dàng hơn mà không cần triển khai giao thức từ đầu. MQTT Client có thể là một máy tính thông thường chạy MQTT Client đồ họa được sử dụng cho mục đích thử nghiệm. Bất kỳ thiết bị nào sử dụng giao thức mạng TCP/IP và có phần mềm triển khai chức năng MQTT Client đều có thể được gọi là MQTT Client. MQTT được thiết kế để hoạt động dựa trên giao thức TCP/IP, do đó, bất kỳ thiết bị nào nói TCP/IP và triển khai giao thức MQTT đều có thể là MQTT Client. Việc triển khai giao thức MQTT trên máy khách rất đơn giản và hợp lý, khiến nó trở nên phù hợp lý tưởng cho các thiết bị nhỏ.

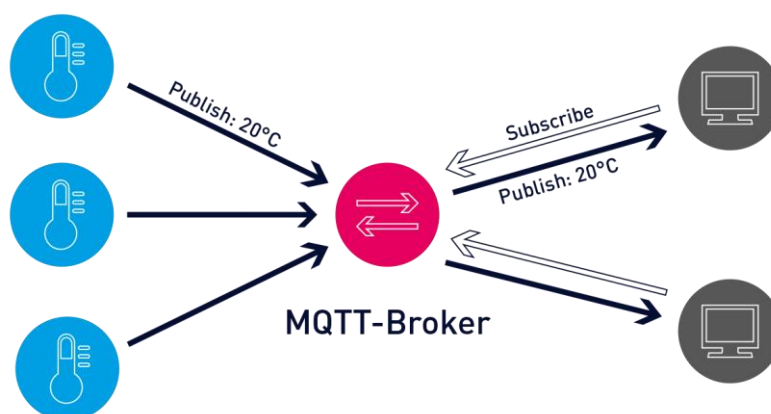
2.2.2 MQTT Broker

MQTT Broker (Nhà môi giới) là một thành phần quan trọng trong hệ thống sử dụng giao thức MQTT. MQTT Broker hoạt động như một máy chủ trung tâm, quản lý việc truyền tin nhắn giữa các Khách hàng MQTT. Nó chịu trách nhiệm định tuyến tin nhắn từ nguồn đến đích, đảm bảo rằng tin nhắn được gửi đến đúng địa chỉ và máy khách.

Chức năng của MQTT Broker bao gồm:

- **Xử lý số lượng lớn kết nối đồng thời:** Tùy thuộc vào việc triển khai, MQTT Broker có thể xử lý hàng triệu khách hàng MQTT được kết nối đồng thời. Nó cho phép liên lạc giữa các thiết bị, mạng và hệ thống phần mềm khác nhau bằng cách thu hẹp khoảng cách giữa chúng.
- **Lọc và định tuyến tin nhắn:** MQTT Broker lọc tin nhắn dựa trên chủ đề đăng ký và xác định (những) khách hàng nào sẽ nhận được tin nhắn.
- **Quản lý phiên:** MQTT Broker duy trì dữ liệu phiên của tất cả các máy khách được kết nối, bao gồm cả đăng ký và tin nhắn bị nhỡ, đối với các máy khách có phiên liên tục.
- **Xác thực và ủy quyền:** MQTT Broker chịu trách nhiệm xác thực và ủy quyền cho khách hàng dựa trên thông tin xác thực do khách hàng cung cấp. MQTT Broker có thể mở rộng, tạo điều kiện cho việc xác thực, ủy quyền và tích hợp tùy chỉnh vào các hệ thống phụ trợ. Ngoài việc xác thực và ủy quyền, các MQTT Broker có thể cung cấp các tính năng bảo mật khác, chẳng hạn như mã hóa tin nhắn trong danh sách kiểm soát truy cập và chuyển tiếp.
- **Khả năng mở rộng, tích hợp và giám sát:** MQTT Broker phải có khả năng mở rộng để xử lý khối lượng lớn tin nhắn và ứng dụng khách, có thể tích hợp vào hệ thống phụ trợ, để giám sát và có khả năng chống lỗi. Để đáp ứng các yêu cầu này, MQTT Broker phải sử dụng công nghệ xử lý mạng theo hướng sự kiện tiên tiến, hệ thống mở rộng mở và các nhà cung cấp giám sát tiêu chuẩn. Các MQTT Broker cũng có thể cung cấp các tính năng nâng cao để quản lý và giám sát hệ thống MQTT, chẳng hạn như lọc tin nhắn, lưu giữ tin nhắn và phân tích thời gian thực.

2.3 Kiến trúc

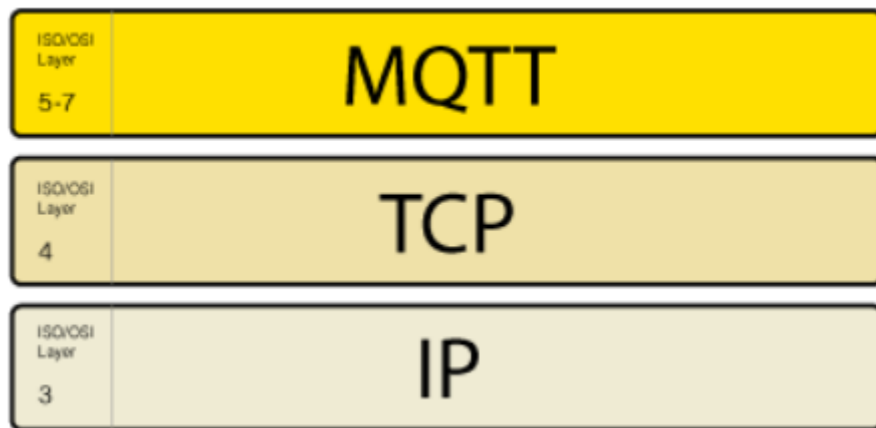


Hình 2.1: Kiến trúc hoạt động của MQTT

Kiến trúc mức cao (high-level) của MQTT gồm 2 phần chính là Broker và Clients.

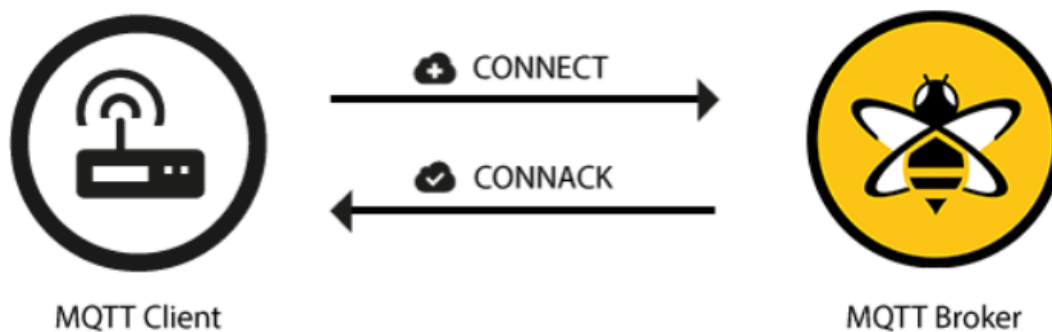
Mô hình nhắn tin của MQTT dựa trên chủ đề và đăng ký, chạy trên TCP/IP bằng cấu trúc liên kết PUSH/SUBSCRIBE. Chủ đề là các chuỗi mà tin nhắn được xuất bản và đăng ký. Trong đó, broker được coi như trung tâm, nó là điểm giao của tất cả các kết nối đến từ client. Nhiệm vụ chính của broker là nhận message từ publisher, xếp các message theo hàng đợi rồi chuyển chúng tới một địa chỉ cụ thể. Nhiệm vụ phụ của broker là nó có thể đảm nhận thêm một vài tính năng liên quan tới quá trình truyền thông như: bảo mật message, lưu trữ message, logs,...

Giao thức MQTT dựa trên TCP/IP, nghĩa là khách hàng và MQTT Broker phải có ngăn xếp TCP/IP.



Hình 2.2: Ngăn xếp TCP/IP

Các kết nối MQTT luôn diễn ra giữa một khách hàng và một MQTT Broker và khách hàng không bao giờ kết nối trực tiếp với các khách hàng khác. Để bắt đầu kết nối, khách hàng sẽ gửi tin nhắn CONNECT tới nhà môi giới, tin nhắn này sẽ phản hồi bằng tin nhắn CONNACK và mã trạng thái. Sau khi kết nối được thiết lập, MQTT Broker sẽ giữ nó mở cho đến khi khách hàng gửi lệnh ngắt kết nối hoặc ngắt kết nối.



Hình 2.3: Mô hình kết nối giữa Client-Broker

Client thì được chia thành 2 nhóm là publisher và subscriber. Client là các software components hoạt động tại edge device nên chúng được thiết kế để có thể hoạt động một cách linh hoạt (lightweight). Client chỉ

làm ít nhất một trong 2 việc là publish các message lên một chủ đề cụ thể hoặc subscribe một chủ đề nào đó để nhận message từ chủ đề này.

2.3.1 Publish, Subscribe

Trong một hệ thống sử dụng giao thức MQTT, nhiều node trạm (gọi là MQTT Client – gọi tắt là Client) kết nối tới một MQTT Server (gọi là broker). Mỗi client sẽ đăng ký một vài kênh (topic), ví dụ như “/client1/channel1”, “/client1/channel2”. Quá trình đăng ký này gọi là “subscribe”, giống như chúng ta đăng ký nhận tin trên một kênh Youtube vậy. Mỗi client sẽ nhận được dữ liệu khi bất kỳ trạm nào khác gửi dữ liệu và kênh đã đăng ký. Khi một client gửi dữ liệu tới kênh đó, gọi là “publish”. Ta sẽ đi sâu hơn về các thuật ngữ trên :

- **Publish:** Gửi một khối dữ liệu chứa tin nhắn sẽ được gửi. Dữ liệu này dành riêng cho từng cách triển khai nhưng có thể đơn giản như chỉ báo bật/tắt hoặc giá trị của một cảm biến nhất định, chẳng hạn như nhiệt độ, áp suất, v.v. Trong trường hợp chủ đề được xuất bản không tồn tại, chủ đề được tạo trên nhà môi giới.
- **Subscribe:** Biến khách hàng thành người đăng ký một chủ đề. Các chủ đề có thể được đăng ký cụ thể hoặc thông qua các ký tự đại diện cho phép đăng ký toàn bộ nhánh chủ đề hoặc một phần của bất kỳ nhánh chủ đề nào. Để đăng ký, khách hàng gửi gói SUBSCRIBE và nhận lại gói SUBACK. Nếu có một tin nhắn được giữ lại cho chủ đề, người đăng ký mới cũng sẽ nhận được tin nhắn đó.



Hình 2.4: Sơ đồ tổng thể Publish - Subscribe

2.3.2 Topic

Tin nhắn trong MQTT được xuất bản dưới dạng chủ đề (Topic). Chủ đề là cấu trúc phân cấp sử dụng ký tự gạch chéo (/) làm dấu phân cách. Cấu trúc này giống với cây thư mục trên hệ thống tệp máy tính. Cấu trúc như sensors/OilandGas/Pressure/ cho phép người đăng ký chỉ định rằng dữ liệu đó chỉ được gửi từ các khách hàng xuất bản về chủ đề OilandGas hoặc để có cái nhìn rộng hơn, có lẽ tất cả dữ liệu từ các khách hàng xuất bản lên bất kỳ chủ đề sensors/OilandGas/. Các chủ đề không được tạo rõ ràng trong MQTT. Nếu MQTT Broker nhận được dữ liệu được xuất bản về một chủ đề hiện không tồn tại thì chủ đề đó sẽ được tạo đơn giản và khách hàng có thể đăng ký chủ đề mới.

MQTT cung cấp hai loại ký tự đại diện để sử dụng với đăng ký chủ đề:

- “+” (dấu cộng) được sử dụng để khớp với một cấp độ duy nhất trong hệ thống phân cấp. Ví dụ: đăng ký “sensors+/livingroom” sẽ khớp với “sensors/temperature/livingroom” và “sensors/humidity/livingroom”, nhưng không khớp với “sensors/temperature/kitchen”.
- “#” (dấu băm) được sử dụng để khớp với nhiều cấp độ trong hệ thống phân cấp. Ví dụ: đăng ký “sensors /#” sẽ khớp với “sensors/temperature/livingroom”, “sensors/humidity/kitchen” và “sensors/power/meter1”.

Dưới đây là một số phương pháp hay nhất để sử dụng Chủ đề MQTT:

- Mỗi chủ đề phải chứa ít nhất một ký tự.
- Chuỗi chủ đề có thể bao gồm các khoảng trống để cho phép các chủ đề mô tả hoặc dễ đọc hơn.
- Các chủ đề có phân biệt chữ hoa chữ thường, có ý nghĩa “myhome/temperature” và “MyHome/Temperature” được coi là hai chủ đề khác nhau.
- Chỉ riêng dấu gạch chéo lên là một chủ đề hợp lệ và có thể được sử dụng để thể hiện một chủ đề rộng hoặc dùng làm ký tự đại diện để đăng ký nhiều chủ đề cùng một lúc.

Đăng ký một số lượng lớn các chủ đề có thể có tác động đáng kể đến hiệu suất của nhà môi giới. Điều này là do mọi tin nhắn được xuất bản theo chủ đề được khách hàng đăng ký phải được gửi đến khách hàng đó. Nếu nhiều khách hàng đăng ký nhiều chủ đề, điều này có thể nhanh chóng trở thành gánh nặng lớn cho nhà môi giới.

Việc sử dụng ký tự đại diện để đăng ký nhiều chủ đề chỉ bằng một lần đăng ký cũng có thể ảnh hưởng đến hiệu suất. Khi khách hàng đăng ký một chủ đề bằng ký tự đại diện, MQTT Broker phải đánh giá mọi tin nhắn được xuất bản đến một chủ đề phù hợp và xác định xem có chuyển tiếp nó cho khách hàng hay không. Nếu số lượng chủ đề phù hợp lớn, điều này có thể gây căng thẳng cho nguồn lực của nhà môi giới.

Để tránh các vấn đề về hiệu suất, điều quan trọng là sử dụng đăng ký chủ đề một cách hiệu quả. Một cách tiếp cận là sử dụng các bộ lọc chủ đề cụ thể hơn bất cứ khi nào có thể, thay vì dựa vào các ký tự đại diện. Một cách tiếp cận khác là sử dụng các đăng ký được chia sẻ, cho phép nhiều khách hàng chia sẻ một đăng ký cho một chủ đề. Điều này có thể giúp giảm số lượng đăng ký và tin nhắn mà MQTT Broker phải xử lý. Cuối cùng, việc giám sát hiệu suất của MQTT Broker và điều chỉnh cấu hình của nó khi cần thiết là điều quan trọng để đảm bảo hiệu suất tối ưu.

2.3.3 Retain

Retain là một cờ (flag) được gắn cho một message của giao thức MQTT. Retain chỉ nhận giá trị 0 hoặc 1 (tương ứng 2 giá trị logic false hoặc true). Nếu retain = 1, broker sẽ lưu lại message cuối cùng của 1 chủ đề kèm theo mức QoS tương ứng. Khi client bắt đăng ký chủ đề có message được lưu lại đó, client ngay lập tức nhận được message. Khi một khách hàng mới đăng ký một chủ đề có tin nhắn được giữ lại, MQTT

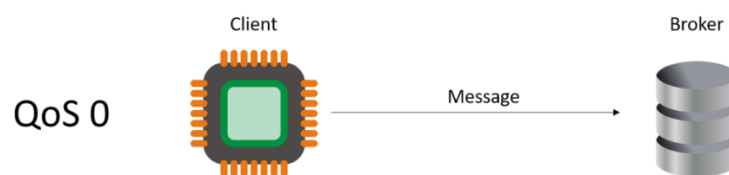
Broker sẽ gửi tin nhắn được giữ lại cuối cùng (về chủ đề đó) cho khách hàng. Điều này cho phép khách hàng nhận được thông tin mới nhất và phù hợp nhất ngay cả khi họ chưa đăng ký chủ đề đó trước đó.

Điều quan trọng cần lưu ý là, giống như nhiều yếu tố khác, việc sử dụng thông báo được giữ lại cũng có thể ảnh hưởng đến hiệu suất của nhà môi giới, đặc biệt nếu có nhiều thông báo được giữ lại. Ngoài ra, nếu một tin nhắn được giữ lại được cập nhật thường xuyên, nó có thể làm tăng lưu lượng truy cập mạng và có khả năng ảnh hưởng đến hiệu suất của mạng.

2.3.4 QoS

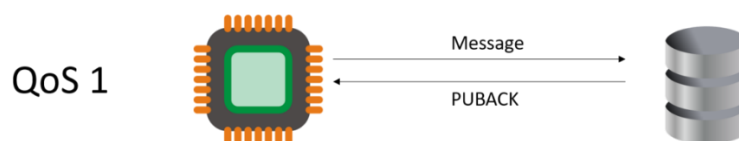
QoS, hay còn gọi là Chất Lượng Dịch Vụ (Quality of Service), là một tính năng quan trọng trong giao thức MQTT (Message Queuing Telemetry Transport). QoS xác định mức độ đảm bảo và đáng tin cậy của quá trình truyền thông giữa các thiết bị MQTT, đặc biệt là trong việc gửi và nhận thông điệp. MQTT hỗ trợ ba cấp độ Chất lượng dịch vụ (QoS): QoS 0, QoS 1 và QoS 2. Dưới đây là bảng phân tích của từng cấp độ:

- **QoS 0:** Cấp độ này cung cấp khả năng gửi “tối đa một lần”, trong đó tin nhắn được gửi mà không cần xác nhận và có thể bị mất. Đây là mức QoS thấp nhất và thường được sử dụng trong các tình huống mất tin nhắn có thể chấp nhận được hoặc khi tin nhắn không quan trọng. Ví dụ: QoS 0 có thể phù hợp để gửi dữ liệu cảm biến trong đó việc mất dữ liệu thường xuyên sẽ không ảnh hưởng đáng kể đến kết quả tổng thể.



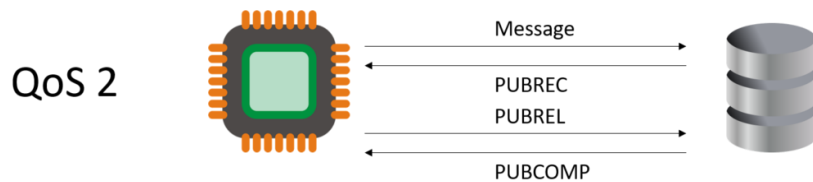
Hình 2.5: QoS 0

- **QoS 1:** Cấp độ này cung cấp khả năng gửi “ít nhất một lần”, trong đó tin nhắn được xác nhận và gửi lại nếu cần. Với QoS 1, nhà xuất bản gửi tin nhắn đến MQTT Broker và chờ xác nhận trước khi tiếp tục. Nếu MQTT Broker không phản hồi trong một khoảng thời gian nhất định, nhà xuất bản sẽ gửi lại tin nhắn. Mức QoS này thường được sử dụng trong các tình huống mà việc mất tin nhắn là không thể chấp nhận được nhưng việc sao chép tin nhắn là có thể chấp nhận được. Ví dụ: QoS 1 có thể phù hợp để gửi thông báo lệnh đến các thiết bị, trong đó một lệnh bị bỏ lỡ có thể gây ra hậu quả nghiêm trọng, nhưng các lệnh trùng lặp thì không.



Hình 2.6: QoS 1

- **QoS 2:** Cấp độ này cung cấp khả năng gửi “chính xác một lần”, trong đó tin nhắn được xác nhận và gửi lại cho đến khi người đăng ký nhận được đúng một lần. QoS 2 là mức QoS cao nhất và thường được sử dụng trong các tình huống hoàn toàn không thể chấp nhận được việc mất hoặc sao chép tin nhắn. Với QoS 2, nhà xuất bản và MQTT Broker tham gia vào quy trình xác nhận hai bước, trong đó MQTT Broker lưu trữ tin nhắn cho đến khi nó được người đăng ký nhận và xác nhận. Mức QoS này thường được sử dụng cho các thông báo quan trọng như giao dịch tài chính hoặc cảnh báo khẩn cấp.



Hình 2.7: QoS 2

2.3.5 Keep-Alive

Trong giao thức MQTT, tính năng "Keep-Alive" là một cơ chế quan trọng để duy trì và kiểm soát kết nối giữa thiết bị và broker. Thông qua tham số "keep-alive", thiết bị client đặt một khoảng thời gian giữa các thông điệp để thông báo cho broker rằng nó vẫn hoạt động và giữ kết nối mạng.

Khi client kết nối đến broker, nó sẽ gửi một giá trị "keep-alive" cùng với thông điệp CONNECT để cấu hình khoảng thời gian mong muốn giữa các thông điệp PINGREQ. Nếu broker không nhận được bất kỳ thông điệp nào từ client trong khoảng thời gian này, nó sẽ gửi một thông điệp PINGREQ đến client. Client phản hồi bằng một thông điệp PINGRESP để xác nhận kết nối vẫn hoạt động.

Tính năng "Keep-Alive" giúp đảm bảo rằng kết nối giữa client và broker không bị đứt, đặc biệt là trong môi trường IoT có thể có những thay đổi đột ngột trong tình trạng mạng. Giá trị "keep-alive" có thể được điều chỉnh tùy thuộc vào yêu cầu cụ thể của ứng dụng và môi trường mạng, từ vài giây đến một số phút. Sử dụng cơ chế "Keep-Alive" trong MQTT không chỉ giúp duy trì kết nối hiệu quả mà còn cung cấp khả năng kiểm soát tình trạng hoạt động của thiết bị trong mạng, tăng tính ổn định và tin cậy của hệ thống IoT.

2.4 Cấu trúc gói tin

Bit	7	6	5	4	3	2	1	0
Byte 1	Message Type				DUP	QoS		Retain
Byte 2	Remaining Length							
Byte 3	Options							
Byte 4	Payload							

Bảng 1: Cấu trúc gói tin

Gói tin MQTT là đơn vị cơ bản của truyền thông trong giao thức MQTT. Header(5 byte):

Byte đầu tiên (Fixed header):

- Bits 7-4 (Message type): xác định loại của gói tin MQTT. Mỗi gói tin MQTT đều có một giá trị trong trường “Message Type” để chỉ định mục đích và nội dung của gói tin. dựa vào bảng dưới đây để kiểm tra loại gói tin

Tên gói	Các loại gói điều khiển MQTT		
	Giá trị	Hướng đi	Mô tả
Reserve	0	Bị cấm	Chủ đề
CONNECT	1	Client -> Server	Client gửi yêu cầu kết nối tới Server
CONNACK	2	Server -> Client	Xác nhận kết nối
PUBLISH	3	Client -> Server hoặc Server -> Client	Xuất bản tin nhắn
PUBACK	4	Client -> Server hoặc Server -> Client	Xuất bản xác nhận
PUBREC	5	Client -> Server hoặc Server -> Client	Đã nhận được xuất bản
PUBREL	6	Client -> Server hoặc Server -> Client	Xuất bản bản phát hành
PUBCOMP	7	Client -> Server hoặc Server -> Client	Xuất bản thành công
SUBSCRIBE	8	Client -> Server	Yêu cầu đăng ký
SUBACK	9	Server -> Client	Xác nhận đăng ký
UNSUBSCRIBE	10	Client -> Server	Yêu cầu hủy đăng ký
UNSUBACK	11	Server -> Client	Xác nhận hủy đăng ký
PINGREQ	12	Client -> Server	Yêu cầu PING
PINGRESP	13	Server -> Client	Phản hồi PING
DISCONNECT	14	Client -> Server	Mất kết nối máy Client

Reserve	15	Bị cấm	Chủ đề
---------	----	--------	--------

Bảng 2: Các loại gói tin

- Bit 3 (DUP – Duplicate): Đánh dấu xem đây có phải là một bản sao của gói tin trước đó không. MQTT DUP cho biết rằng thư bị trùng lặp và đã được gửi lại do người nhận dự định (khách hàng hoặc nhà môi giới) không xác nhận thư gốc. Nó chỉ liên quan đến các tin nhắn có QoS lớn hơn 0. Khi khách hàng hoặc MQTT Broker nhận được tin nhắn có gắn cờ DUP, nó sẽ bỏ qua tin nhắn đó nếu đã nhận được tin nhắn có cùng ID tin nhắn. Khách hàng hoặc MQTT Broker nên xử lý tin nhắn một cách bình thường nếu trước đó họ chưa nhận được nó.
- Bits 2-1 (QoS – Quality of Service): Xác định mức độ đảm bảo chất lượng trong việc gửi và nhận gói tin.
- Bit 0 (Retain): giữ lại thông điệp cuối cùng được gửi đến một chủ đề và gửi lại nó cho mọi client mới kết nối hoặc subscribe vào chủ đề đó.

Byte thứ hai về sau (Remaining Length): là một số nguyên có độ dài thay đổi cho biết số byte còn lại trong gói điều khiển hiện tại, bao gồm cả dữ liệu tiêu đề và payload thay đổi. Kết quả là độ dài còn lại bằng dữ liệu tiêu đề thay đổi cộng với payload. Sử dụng thuật toán Base128 để biểu diễn chiều dài của trường payload và độ dài của trường khác.

2.5 Tính bảo mật

Mục tiêu ban đầu của giao thức MQTT là tạo ra khả năng truyền dữ liệu nhỏ nhất và hiệu quả nhất qua các đường truyền thông đắt tiền, không đáng tin cậy. Do đó, bảo mật không phải là mối quan tâm hàng đầu trong quá trình thiết kế và triển khai MQTT.

Tuy nhiên, có một số tùy chọn bảo mật có sẵn với chi phí truyền dữ liệu nhiều hơn và dung lượng lớn hơn:

- **Bảo mật mạng** – Nếu bản thân mạng có thể được bảo mật thì việc truyền dữ liệu MQTT không an toàn là không liên quan. Trong trường hợp này, các vấn đề bảo mật sẽ phải xảy ra từ bên trong mạng, có thể thông qua một tác nhân xấu hoặc một hình thức xâm nhập mạng khác.
- **Tên người dùng và mật khẩu** – MQTT cho phép khách hàng thiết lập kết nối với MQTT Broker bằng tên người dùng và mật khẩu. Thật không may, để tránh sự chú ý, tên người dùng và mật khẩu được truyền dưới dạng văn bản rõ ràng. Vào năm 1999, điều này là quá đủ vì việc chặn liên lạc vệ tinh đối với những gì về cơ bản là việc đọc cảm biến không quan trọng sẽ cực kỳ khó khăn. Tuy nhiên, ngày nay, việc chặn nhiều loại giao tiếp mạng không dây là chuyện nhỏ, khiến cho việc xác thực như vậy trở nên vô dụng. Nhiều trường hợp sử dụng yêu cầu tên người dùng và mật khẩu không phải để bảo vệ chống lại những kẻ có ý đồ xấu mà là một cách để tránh các kết nối không chủ ý.

- **SSL/TLS** – Chạy trên TCP/IP, giải pháp rõ ràng để bảo mật đường truyền giữa khách hàng và MQTT Broker là triển khai SSL/TLS. Thật không may, điều này làm tăng thêm chi phí đáng kể cho các hoạt động liên lạc nhẹ nhàng khác.

2.6 Ưu và nhược điểm của MQTT

2.6.1 Ưu điểm

MQTT được đánh giá là một trong những giao thức IoT tốt nhất do các tính năng và khả năng độc đáo của nó phù hợp với nhu cầu cụ thể của các hệ thống IoT. Một số ưu điểm bao gồm:

- **Trọng lượng nhẹ:** Thiết kế của tin nhắn MQTT rất nhỏ (tin nhắn nhỏ nhất có thể nhỏ tới hai byte), giúp chúng dễ dàng truyền qua mạng. Giao thức dựa trên nhị phân giúp tối ưu hóa kích thước của tin nhắn và do đó tiêu thụ băng thông tối thiểu trong quá trình truyền. Điều này làm cho MQTT trở nên lý tưởng cho các ứng dụng thường hoạt động trong phạm vi dung lượng mạng hạn chế.
- **Độ tin cậy:** MQTT cho phép điều chỉnh mức độ tin cậy để gửi tin nhắn đến người nhận dự định. Do đó, MQTT hỗ trợ các mức Chất lượng Dịch vụ (QoS) MQTT khác nhau và cho phép người dùng chỉ định tầm quan trọng của việc gửi tin nhắn bằng cách chọn từ các mức QoS 0, 1 và 2. QoS 0 không đảm bảo tin nhắn được gửi đáng tin cậy, trong khi QoS 2 đảm bảo rằng các thuê bao dự kiến sẽ nhận được tin nhắn chính xác một lần. Đối với các trường hợp khác, khi có vấn đề về ổn định mạng, người ta cũng có thể sử dụng các phiên MQTT liên tục. Phiên MQTT hoạt động liên tục đảm bảo rằng ngay cả khi khách hàng đã thiết lập kết nối với MQTT Broker và đăng ký một chủ đề cụ thể ngoại tuyến, MQTT Broker sẽ giữ lại thông tin mà khách hàng đã không nhận được kể từ phiên hoạt động cuối cùng. Thông tin này sẽ được gửi đến thuê bao khi nó trực tuyến trở lại.
- **Giao tiếp an toàn:** MQTT linh hoạt và cho phép các nhà phát triển thực hiện các biện pháp bảo mật bổ sung như một phần của chức năng môi giới MQTT như xác thực tên người dùng và mật khẩu, xác thực chứng chỉ máy khách và mã hóa bằng Bảo mật lớp vận chuyển (TLS), còn được gọi thông tục là Lớp công bảo mật (SSL). Ta cũng có thể sử dụng danh sách kiểm soát truy cập và các biện pháp khác để bảo vệ thiết lập MQTT của mình khỏi bị truy cập trái phép và đảm bảo mức độ bảo mật cao.
- **Tính hai chiều:** Mô hình đăng ký xuất bản của MQTT cho phép liên lạc hai chiều liên mạch giữa các thiết bị. Khách hàng có thể vừa xuất bản tin nhắn theo chủ đề vừa đăng ký nhận tin nhắn về các chủ đề cụ thể, cho phép trao đổi dữ liệu hiệu quả trong hệ sinh thái IoT đa dạng mà không cần kết nối trực tiếp giữa các thiết bị. Mô hình này cũng đơn giản hóa việc tích hợp các thiết bị mới, đảm bảo khả năng mở rộng dễ dàng
- **Khả năng mở rộng:** Giao thức MQTT hỗ trợ cả phương pháp tiếp cận khả năng mở rộng theo chiều ngang và chiều dọc, cho phép tăng trưởng và mở rộng quy mô dự án IoT của ta một cách suôn sẻ.

- **Tích hợp đơn giản:** Sự đơn giản và sẵn có của giao thức MQTT với các thư viện MQTT khác nhau cho các ngôn ngữ và nền tảng lập trình khác nhau giúp các nhà phát triển IoT mới dễ dàng tích hợp nó vào hệ thống.
- **Hỗ trợ mạnh mẽ trong ngành:** MQTT đã nhận được sự hỗ trợ rộng rãi trong toàn ngành, với các MQTT Broker MQTT và thư viện khách hàng tạo thành một hệ sinh thái mạnh mẽ của các công cụ và tài nguyên. Nhiều nhà cung cấp cũng có các giải pháp MQTT tại chỗ hoặc đám mây thuận tiện và sẵn sàng sử dụng. Họ cung cấp các tính năng phong phú và các công cụ bổ sung giúp MQTT dễ tiếp cận và dễ làm việc hơn.
- **Hỗ trợ ngôn ngữ:** Hệ thống IoT thường bao gồm các thiết bị và ứng dụng được phát triển bằng nhiều ngôn ngữ lập trình khác nhau. Hỗ trợ ngôn ngữ rộng của MQTT cho phép tích hợp dễ dàng với nhiều nền tảng và công nghệ, thúc đẩy khả năng giao tiếp và tương tác liền mạch trong các hệ sinh thái IoT đa dạng.

2.6.2 Nhược điểm

Mặc dù MQTT cung cấp nhiều lợi thế, nhưng nó cũng có một số nhược điểm tiềm ẩn trong một số tình huống nhất định. Một số nhược điểm của MQTT trong IoT đáng chú ý:

- **Các tính năng bảo mật hạn chế:** MQTT được thiết kế để trở thành một giao thức nhẹ và do đó, các tính năng bảo mật của nó bị hạn chế. Mặc dù nó hỗ trợ các cơ chế bảo mật cơ bản như xác thực tên người dùng/mật khẩu và mã hóa SSL/TLS, nhưng nó có thể không cung cấp các tính năng bảo mật nâng cao so với các giao thức khác.
- **Thiếu độ tin cậy tích hợp:** Mặc dù MQTT được thiết kế cho băng thông thấp và mạng không đáng tin cậy, nhưng nó không đảm bảo gửi tin nhắn hoặc cung cấp các cơ chế độ tin cậy tích hợp. Mặc dù có các mức Chất lượng Dịch vụ (QoS) có thể được đặt để gửi tin nhắn, nhưng chúng có thể không đủ cho tất cả các trường hợp sử dụng và có thể cần các biện pháp bổ sung để liên lạc đáng tin cậy.
- **Giao tiếp không trạng thái:** MQTT là một giao thức không trạng thái, có nghĩa là nó không duy trì kết nối liên tục giữa các thiết bị. Đây có thể là một bất lợi trong các tình huống mà việc duy trì kết nối liên tục là rất quan trọng và nó có thể yêu cầu các nỗ lực triển khai bổ sung để quản lý trạng thái của thiết bị.
- **Thách thức về khả năng mở rộng:** Mặc dù MQTT có thể mở rộng và có thể xử lý một số lượng lớn thiết bị, việc triển khai cơ sở hạ tầng MQTT có thể mở rộng và hiệu quả có thể yêu cầu xem xét cẩn thận các yếu tố như cấu hình trình môi giới tin nhắn, hạn chế mạng và cân bằng tải. Trong một số tình huống nhất định, việc mở rộng quy mô có thể trở nên khó khăn.

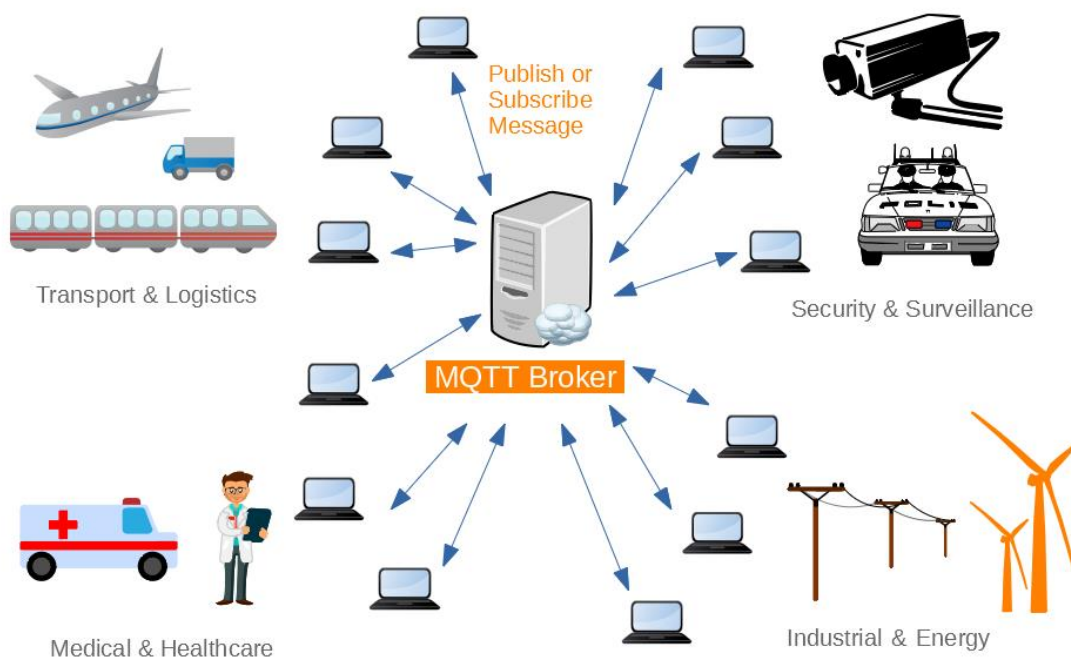
- **Độ phức tạp trong các mạng lớn:** Trong các mạng IoT lớn và phức tạp, việc quản lý các chủ đề, đăng ký và giao tiếp tổng thể có thể trở nên phức tạp. Nếu không có thiết kế và tổ chức các chủ đề phù hợp, có thể khó duy trì cấu trúc giao tiếp rõ ràng và hiệu quả.
- **Tiêu thụ điện năng:** Mặc dù MQTT thường được coi là nhẹ, nhưng trong các tình huống tiêu thụ điện năng thấp là yêu cầu quan trọng (ví dụ: thiết bị chạy bằng pin), kết nối liên tục và giao tiếp định kỳ của MQTT có thể góp phần tiêu thụ điện năng cao hơn so với các giao thức khác được thiết kế dành riêng cho các thiết bị năng lượng thấp.
- **Triển khai dành riêng cho nhà cung cấp:** Mặc dù MQTT là một tiêu chuẩn mở, nhưng có thể có các biến thể và sự khác biệt trong cách các nhà cung cấp khác nhau thực hiện nó. Điều này có thể dẫn đến các vấn đề về khả năng tương tác giữa các thiết bị từ các nhà sản xuất khác nhau, đặc biệt nếu chúng dựa vào các tiện ích mở rộng hoặc tính năng độc quyền.

III. Thực nghiệm

3.1 Ứng dụng thực tế

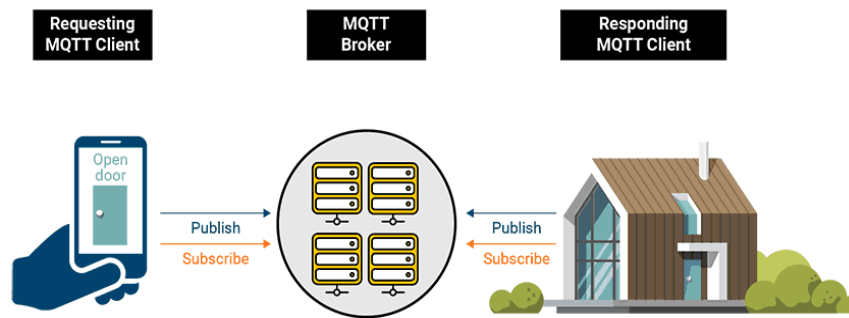
MQTT được sử dụng rộng rãi trong các ứng dụng IoT, IoT công nghiệp (IIoT) và M2M. Dưới đây là một vài ví dụ:

- **Nhà thông minh:** MQTT được sử dụng để kết nối nhiều thiết bị khác nhau trong một ngôi nhà thông minh, bao gồm bộ điều nhiệt thông minh, bóng đèn, camera an ninh và các thiết bị khác. Điều này cho phép người dùng điều khiển các thiết bị trong nhà của họ từ xa bằng ứng dụng di động.
- **Tự động hóa công nghiệp:** MQTT được sử dụng để kết nối máy móc và cảm biến trong nhà máy và các cơ sở công nghiệp khác. Điều này cho phép giám sát và kiểm soát các quy trình theo thời gian thực, có thể cải thiện hiệu quả và giảm thời gian ngừng hoạt động.
- **Nông nghiệp:** MQTT được sử dụng trong nông nghiệp chính xác để theo dõi độ ẩm của đất, điều kiện thời tiết và sự phát triển của cây trồng. Điều này giúp nông dân tối ưu hóa việc tưới tiêu và các biện pháp quản lý cây trồng khác.
- **Chăm sóc sức khỏe:** MQTT được sử dụng để kết nối các thiết bị và cảm biến y tế, chẳng hạn như máy đo đường huyết và máy đo nhịp tim, với các nhà cung cấp dịch vụ chăm sóc sức khỏe. Điều này cho phép theo dõi bệnh nhân từ xa, có thể cải thiện kết quả của bệnh nhân và giảm chi phí chăm sóc sức khỏe.
- **Vận tải:** MQTT được sử dụng trong ô tô được kết nối và các hệ thống giao thông khác để cho phép theo dõi và giám sát phương tiện theo thời gian thực. Điều này có thể cải thiện sự an toàn và giúp tối ưu hóa lưu lượng giao thông.



Hình 3.1: Ứng dụng của MQTT

3.1.1 Nhà thông minh



Hình 3.2: MQTT trong SmartHome

MQTT khi kết hợp với các cảm biến thông minh trong ngôi nhà để đo lường, thu thập dữ liệu về môi trường xung quanh có thể dễ dàng truyền dữ liệu này đến các thiết bị khác hoặc lưu trữ trên nền tảng đám mây gửi tới người dùng. Ví dụ dữ liệu từ cảm biến nhiệt độ và độ ẩm khi được gửi qua MQTT giúp giám sát và điều khiển hệ thống điều hòa nhiệt độ, cảnh báo khi điều kiện vượt quá giới hạn. Khi kết hợp với cảm biến ánh sáng, nó giải quyết được việc tự động bật/tắt đèn hoặc điều chỉnh độ sáng dựa trên mức ánh sáng trong nhà. Ứng dụng chung với cảm biến chuyển động giúp các cánh cửa tự động, việc bật tắt đèn tự hoạt động khi phát hiện chuyển động, đồng thời kích hoạt được hệ thống an ninh khi có người khả nghi đột nhập. Đối với cảm biến khí gas, dữ liệu thông qua MQTT kích hoạt hệ thống báo động và thông báo với người dùng về tình trạng nguy hiểm trong nhà. Ứng dụng cảm biến mở/cửa qua MQTT để giám sát an ninh và có thể kích hoạt các hành động như tắt điều hòa nhiệt độ khi cửa sổ được mở.

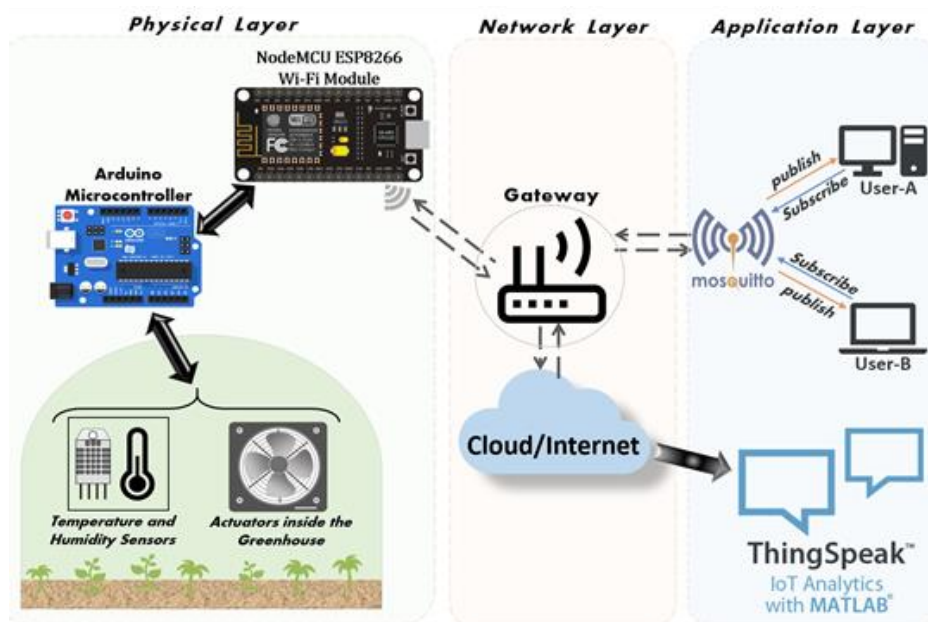
Đã có một số dự án được thực hiện với giao thức MQTT trong SmartHome:

- **OpenHAB:** là một nền tảng tự động hóa nhà thông minh mã nguồn mở và có khả năng tích hợp chặt chẽ với giao thức MQTT (Message Queuing Telemetry Transport). OpenHAB giúp tạo ra một môi trường quản lý nhà thông minh linh hoạt và mở rộng, trong đó MQTT chơi một vai trò quan trọng. Giao thức MQTT được tích hợp sâu rộng trong OpenHAB để kết nối và giao tiếp với các thiết bị trong hệ thống nhà thông minh. Bằng cách sử dụng mô hình xuất bản - đăng ký (publish-subscribe), OpenHAB có thể theo dõi trạng thái của các thiết bị thông qua các chủ đề MQTT và thực hiện các hành động tự động dựa trên sự thay đổi trong trạng thái này. Trong OpenHAB, các thiết bị và chủ đề MQTT được quản lý thông qua định nghĩa Items trong file cấu hình. Điều này cho phép người dùng liên kết các thiết bị vật lý với chủ đề MQTT tương ứng, giúp tạo ra sự tương tác mạnh mẽ giữa các thành phần khác nhau trong hệ thống. Ngoài ra, OpenHAB cung cấp tích hợp đầy đủ cho MQTT, cho phép người dùng thiết lập các broker MQTT, cấu hình chất lượng dịch vụ (QoS), và xác thực tài khoản. Điều này giúp đảm bảo an toàn và bảo mật trong quá trình truyền thông giữa các thiết bị và OpenHAB. Tính tích hợp sâu rộng giữa OpenHAB và MQTT mang lại sự linh hoạt cho người dùng để xây dựng và quản lý hệ thống nhà thông minh của mình. Sự kết hợp

giữa hai công nghệ này cung cấp một giải pháp mạnh mẽ cho việc kiểm soát và tự động hóa các thiết bị và dịch vụ trong ngôi nhà thông minh.

- **Home Assistant:** là một nền tảng tự động hóa nhà thông minh mã nguồn mở và đa nền tảng, được tích hợp chặt chẽ với giao thức MQTT (Message Queuing Telemetry Transport). MQTT chơi một vai trò quan trọng trong việc kết nối và tương tác giữa các thiết bị trong hệ thống nhà thông minh do tích hợp sâu rộng của nó trong Home Assistant. Home Assistant hỗ trợ các tính năng chính của MQTT, bao gồm khả năng kết nối với broker MQTT, định nghĩa các chủ đề để liên kết thiết bị, và đồng bộ dữ liệu giữa các thiết bị thông minh. Người dùng có thể thiết lập broker MQTT nội bộ hoặc sử dụng các dịch vụ MQTT công cộng, tùy thuộc vào yêu cầu của họ. Mô hình xuất bản - đăng ký (publish-subscribe) của MQTT được sử dụng trong Home Assistant, cho phép các thiết bị xuất bản thông điệp và các thiết bị khác đăng ký để nhận thông điệp từ các chủ đề tương ứng. Điều này tạo ra một môi trường tương tác linh hoạt, nơi các thiết bị có thể truyền thông và thực hiện các hành động tự động dựa trên sự thay đổi trong trạng thái. Home Assistant sử dụng cấp độ chất lượng dịch vụ (QoS) của MQTT để đảm bảo tính đáng tin cậy trong truyền thông, và retained messages giúp duy trì trạng thái cuối cùng của các chủ đề, cung cấp thông tin liên quan khi các thiết bị mới tham gia vào mạng. Tích hợp mạnh mẽ giữa Home Assistant và MQTT mang lại sự linh hoạt và khả năng tương tác lớn trong quản lý và kiểm soát các thiết bị thông minh trong ngôi nhà thông minh. Sự kết hợp giữa hai công nghệ này tạo nên một giải pháp mạnh mẽ và linh hoạt, đáp ứng đa dạng nhu cầu của người dùng trong lĩnh vực tự động hóa nhà thông minh.

3.1.2 Nông nghiệp

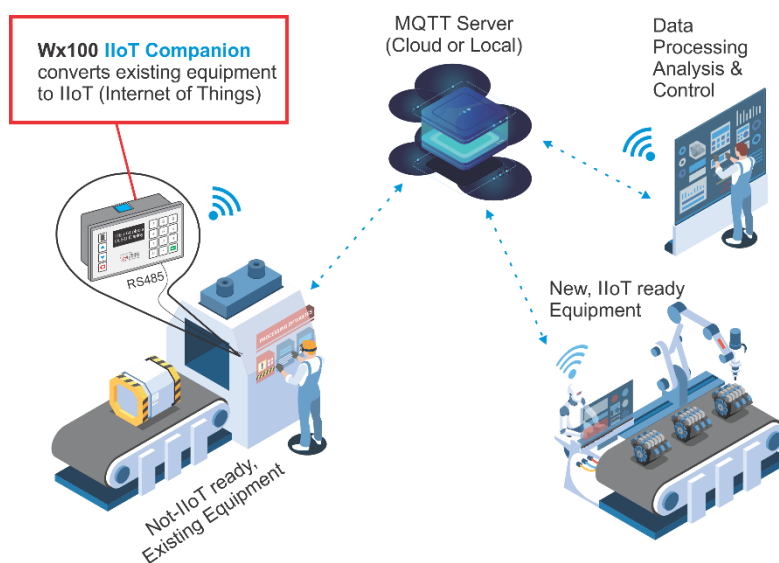


Hình 3.3: Mô hình đo độ ẩm đất bằng ESP8266

MQTT có thể được sử dụng một cách hiệu quả trong nông nghiệp để giám sát và điều khiển các thiết bị, thu thập dữ liệu từ cảm biến và tối ưu hóa việc quản lý trong nông nghiệp. MQTT giúp giám sát điều kiện

môi trường bằng cách sử dụng các cảm biến như cảm biến độ ẩm đất, cảm biến nhiệt độ, cảm biến ánh sáng và gửi dữ liệu từ các cảm biến này đến một Broker MQTT. Nó cung cấp thông tin về điều kiện môi trường trong nông trại giúp người nông dân quyết định về thời điểm tưới nước, sử dụng năng lượng và quản lý đất đai. MQTT giúp giám sát năng lượng và tự động hóa tưới nước, các dữ liệu về hệ thống tưới nước tự động, thiết bị kiểm soát năng lượng sẽ được gửi về Broker MQTT. Người nông dân có thể giám sát và điều khiển hệ thống từ xa và tối ưu hóa sử dụng nước và năng lượng. MQTT giúp theo dõi vật nuôi và thú y bằng các lấy dữ liệu từ cảm biến về sức khỏe, nhiệt độ và các yếu tố môi trường khác, thông báo có thể được gửi khi có dấu hiệu bất thường, giúp nông dân can thiệp kịp thời. Các dữ liệu về mức nước, tình trạng cây trồng và dữ liệu sản lượng được gửi đến Broker MQTT giúp người nông dân có thể quản lý kết tủa, việc tưới nước và dinh dưỡng. Kết hợp với cảm biến theo dõi sức khỏe của động vật, cảm biến đo lường chất lượng thức ăn, dữ liệu đến Broker MQTT có thể thông báo đến nông dân kịp thời để can thiệp, bảo vệ sức khỏe của gia súc và gia cầm. Các cảm biến đo lường hiệu suất máy nông nghiệp, cảm biến theo dõi dầu nhớt thông qua MQTT giúp người nông dân theo dõi được hiệu suất máy móc, tình trạng bảo dưỡng. Bên cạnh đó, nông dân có thể lên lịch bảo dưỡng hoặc sửa chữa để giảm thiểu thời gian chết máy và tăng hiệu suất.

3.1.3 Tự động hóa trong công nghiệp



Hình 3.4: Mô hình tổng quát về MQTT trong công nghiệp

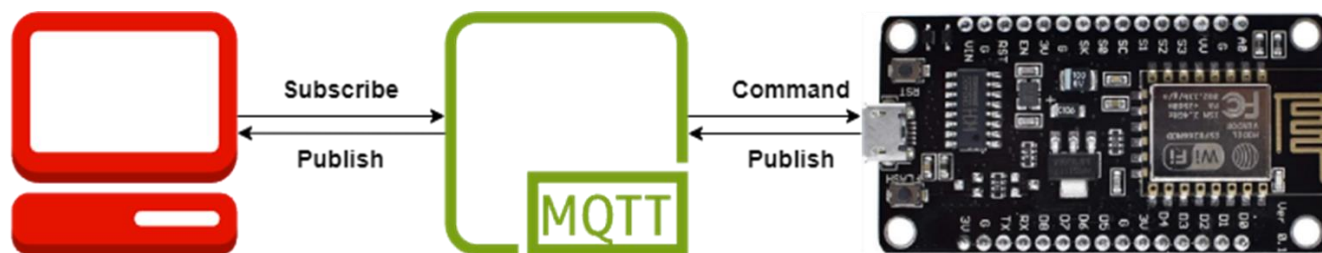
Trong lĩnh vực công nghiệp, MQTT được sử dụng rộng rãi để giám sát và điều khiển các thiết bị, máy móc và hệ thống khác nhau. MQTT nhận dữ liệu về máy sản xuất, robot công nghiệp về trạng thái hoạt động, hiệu suất và tình trạng của máy móc. Hệ thống giám sát có thể theo dõi và quản lý hiệu suất máy móc, đồng thời cung cấp khả năng điều khiển từ xa. Khi kết hợp với hệ thống năng lượng, thiết bị đo lường tiêu thụ năng lượng, dữ liệu về tiêu thụ năng lượng và tình trạng hoạt động của các thiết bị năng lượng được gửi qua MQTT giúp hệ thống giám sát có thể tối ưu hóa việc sử dụng năng lượng và quản lý tài nguyên một cách thông minh. Dữ liệu về quy trình sản xuất, chất lượng sản phẩm và thông tin từ các cảm biến được gửi qua MQTT giúp tổ chức theo dõi và cải thiện quy trình sản xuất. Kết hợp với cảm biến theo dõi tình trạng

của máy móc, dữ liệu gửi đến Broker MQTT giúp hệ thống bảo dưỡng có thể lên lịch bảo dưỡng hoặc sửa chữa dựa trên thông tin này, giảm thiểu thời gian dừng máy và tăng tuổi thọ của thiết bị. Giúp quản lý kho và vận chuyển có thể theo dõi và tối ưu hóa các quy trình liên quan khi kết hợp với cảm biến vị trí, hệ thống theo dõi hàng hóa. Kết hợp MQTT với Hệ thống kiểm soát và giám sát tự động (SCADA) cung cấp khả năng mở rộng và tích hợp với các thiết bị khác. Quản lý được chất lượng và an toàn sản phẩm, dữ liệu về chất lượng sản phẩm và thông tin an toàn từ cảm biến theo dõi chất lượng sản phẩm, hệ thống an toàn được gửi qua MQTT. Hệ thống quản lý có thể tự động phát hiện và ứng phó với chất lượng hoặc an toàn.

3.2 Kịch bản

Kịch bản ở đây là sử dụng MQTT để điều khiển ESP8266 bật/tắt đèn. Để thực hiện, ta cần có Arduino IDE để lập trình và sử dụng một số thư viện hỗ trợ MQTT cho ESP8266. Tiếp theo, ta sẽ sử dụng MQTT Explorer đóng vai trò là trình mô phỏng để tạo kết nối MQTT broker và từ đó truyền lệnh bật/tắt đèn cho ESP8266.

Môi trường thực nghiệm: Windows 10, CPU 8 nhân AMD Ryzen 7, Ram 16GB



Hình 3.5: Mô hình thực nghiệm

IV. Phân tích

4.1 Arduino

Arduino là một nền tảng mã nguồn mở được sử dụng để xây dựng các dự án điện tử. Arduino bao gồm cả bảng mạch lập trình vật lý (thường được gọi là vi điều khiển) và một phần mềm hoặc IDE (Môi trường phát triển tích hợp) chạy trên máy tính của bạn, được sử dụng để viết và tải mã máy tính lên bảng vật lý.

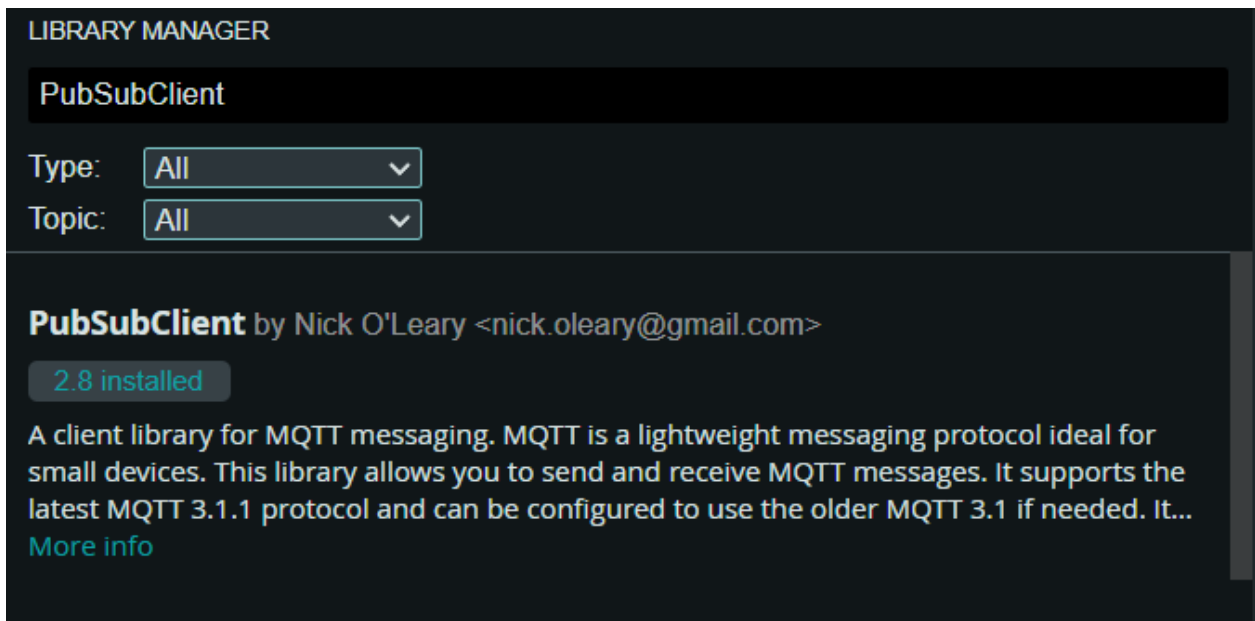
Nền tảng Arduino đã trở nên khá phổ biến với những người mới bắt đầu làm quen với thiết bị điện tử và vì lý do chính đáng. Không giống như hầu hết các bảng mạch lập trình trước đây, Arduino không cần một phần cứng riêng biệt (được gọi là bộ lập trình) để tải mã mới lên bảng - bạn chỉ cần sử dụng cáp USB. Ngoài ra, Arduino IDE sử dụng phiên bản C++ đơn giản hóa, giúp học lập trình dễ dàng hơn. Cuối cùng, Arduino cung cấp một hệ số dạng tiêu chuẩn để chia nhỏ các chức năng của bộ điều khiển vi mô thành một gói dễ tiếp cận hơn.

Đối với dự án này, ta sẽ lập trình khả năng kết nối tới ESP8266 thông qua MQTT và sẽ trải qua các bước sau:

Bước 1: Cài Đặt Thư Viện

Mở Arduino IDE và chọn “Sketch” -> “Include Library” -> “Manage Libraries...”

Tìm kiếm “PubSubClient” và cài đặt thư viện.



Hình 4.1: Thư viện PubSubClient

Bước 2: thêm thư viện yêu cầu và thay đổi giá trị của ssid, password, mqttServer, mqttUser, và mqttPassword theo thông tin của ta.

```
#include <ESP8266WiFi.h>           // Thư viện dùng để kết nối WiFi của ESP8266
#include <PubSubClient.h>          // Thư viện dùng để connect, publish/subscribe MQTT

const char* ssid = "TrThanh.69";   // Tên của mạng WiFi mà bạn muốn kết nối đến
const char* password = "donknow?"; // Mật khẩu của mạng WiFi

const char* mqttServer = "test.mosquitto.org";
const int   mqttPort = 1883;
const char* mqttUser = "";
const char* mqttPassword = "";
```

Bước 3: khởi tạo kết nối MQTT và tạo 1 chủ đề(Topic)

```
void setup() {
    Serial.begin(9600);           // Khởi tạo kết nối Serial để truyền dữ liệu đến máy tính

    pinMode(led, OUTPUT);

    startWiFi();

    connectBroker();

    client.subscribe("blockynode/led"); // Đăng ký nhận tin từ topic = "blockynode/led"
    client.publish("blockynode/led", "Hello from Blocky Node"); // Gửi message = "Hello from Blocky Node" đến topic = "blockynode/led"
```

Bước 4: thiết lập yêu cầu điều khiển bật/tắt đèn

```
void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    int i;
    for (i = 0; i < length; i++) {
        messageBuff[i] = (char)payload[i];
    }
    messageBuff[i] = '\0';

    String message = String(messageBuff);
    Serial.println(message);

    if (message == "on") {
        digitalWrite(led, LOW);           // Bật đèn LED khi nhận được tín hiệu "on"
        client.publish("blockynode/led", "LED is turned ON"); // Thông báo trạng thái bật LED cho tất cả client đăng ký topic = "blockynode/led"
    } else if (message == "off") {
        digitalWrite(led, HIGH);          // Tắt đèn LED khi nhận được tín hiệu "off"
        client.publish("blockynode/led", "LED is turned OFF"); // Thông báo trạng thái tắt LED cho tất cả client đăng ký topic = "blockynode/led"
    }
}
```

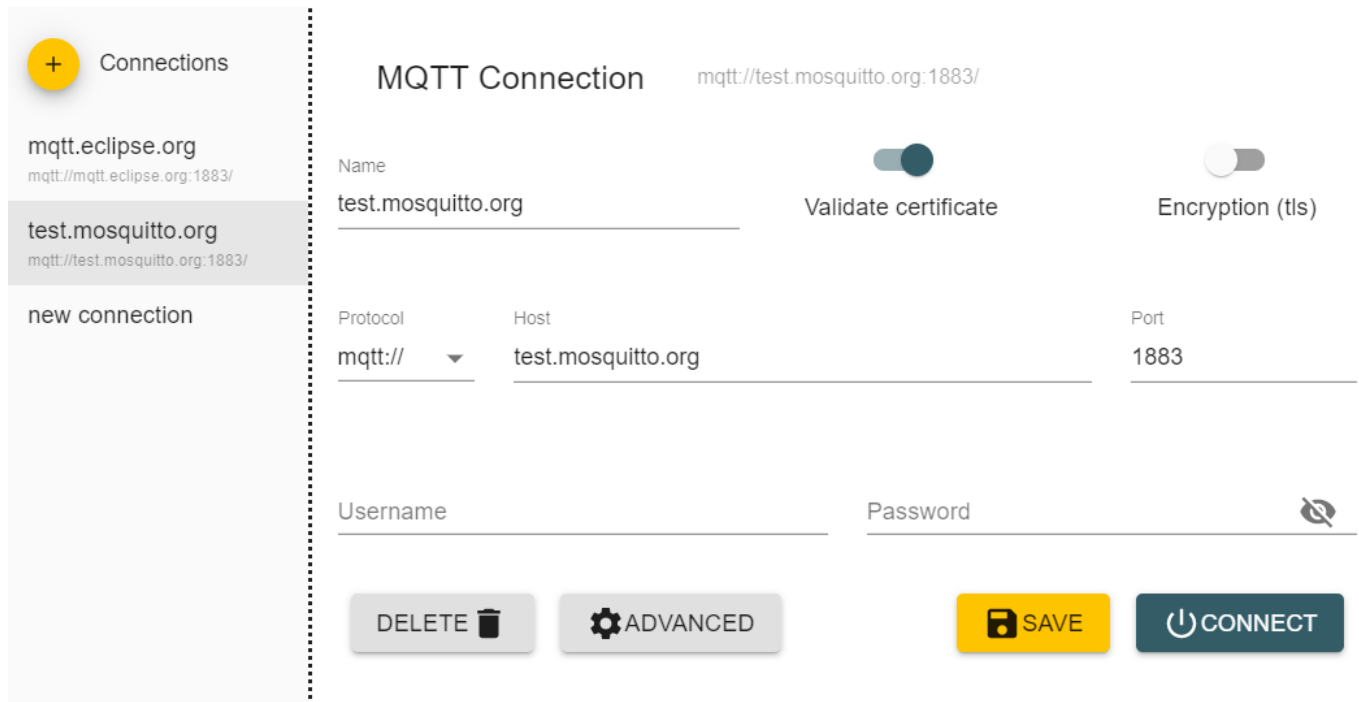
Sau khi hoàn thành lập trình, cắm trực tiếp mô-đun ESP8266 để truyền code vào.

4.2 MQTT Explorer

MQTT Explorer là một công cụ MQTT Client mã nguồn mở cung cấp giao diện người dùng đồ họa (GUI) dễ sử dụng với tổng quan về chủ đề có cấu trúc. Nó sử dụng chế độ xem chính phân cấp và hỗ trợ hiển thị biểu đồ trực quan về các tin nhắn tải trọng(payload) đã nhận.

MQTT Explorer hỗ trợ các giao thức MQTT 5.0 và 3.1.1 và cho phép các nhà phát triển tạo đồng thời một kết nối MQTT/MQTTs. MQTT Explorer được viết bằng Typescript và được phát triển bởi Thomas Nordquist. Nó đa nền tảng và có thể chạy trên Windows, macOS và Linux.

Sau khi hoàn thành lập trình trên Arduino, ta sẽ cho khởi chạy MQTT Explorer, sau đó thiết lập kết nối như bên dưới để tiến hành tạo liên kết với máy chủ. Ở đây ta sẽ mượn tên miền test.mosquitto.org đóng vai trò là MQTT broker để điều khiển đèn của ESP8266



Hình 4.2: Giao diện thiết lập kết nối

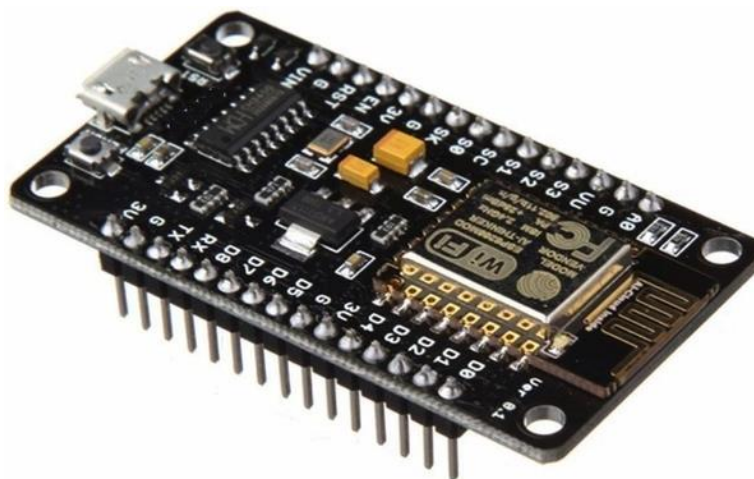
4.3 ESP8266

ESP8266 là một mô-đun Wi-Fi có kết nối Internet và được tích hợp vào một số bo mạch nhúng như NodeMCU, Wemos và ESP-01. ESP8266 có thể hoạt động như một điểm truy cập, một máy khách kết nối với một điểm truy cập khác hoặc cả hai. Nó được sử dụng rộng rãi trong các ứng dụng IoT (Internet of Things) như cảm biến thông minh, hệ thống điều khiển thiết bị hoặc ứng dụng điều khiển từ xa. Mô-đun này rẻ và rất dễ sử dụng, cùng với khả năng tương thích với nhiều loại vi điều khiển khác nhau. Dưới đây là một số tính năng đáng chú ý của ESP8266:

- **Vi điều khiển với Wi-Fi tích hợp:** ESP8266 không chỉ là một mô-đun Wi-Fi; nó cũng bao gồm một đơn vị vi điều khiển (MCU) với các chân GPIO (Đầu vào / Đầu ra Mục đích chung). Điều này làm cho nó trở thành một nền tảng linh hoạt để xây dựng các thiết bị IoT, vì nó có thể xử lý cả các tác vụ xử lý và giao tiếp qua Wi-Fi.
- **Chi phí thấp:** Một trong những lợi thế đáng kể của ESP8266 là khả năng chi trả của nó. Điều này làm cho nó trở thành một lựa chọn hấp dẫn cho những người có sở thích, nhà sản xuất và các dự án IoT quy mô nhỏ, nơi chi phí là một yếu tố quan trọng.
- **Hỗ trợ cộng đồng:** ESP8266 có một cộng đồng lớn và tích cực gồm các nhà phát triển và những người đam mê. Hỗ trợ cộng đồng này cung cấp quyền truy cập vào vô số tài nguyên, hướng dẫn,

thư viện và diễn đàn nơi người dùng có thể chia sẻ kinh nghiệm và giải pháp của họ cho những thách thức chung.

- **Ngôn ngữ lập trình:** ESP8266 có thể được lập trình bằng nhiều ngôn ngữ khác nhau, với Arduino và MicroPython là những lựa chọn phổ biến. Arduino IDE (Môi trường phát triển tích hợp) cung cấp một môi trường quen thuộc và thân thiện với người mới bắt đầu để lập trình ESP8266, trong khi MicroPython cho phép các nhà phát triển sử dụng Python cho các dự án IoT.
- **Giao thức IoT:** ESP8266 có thể giao tiếp bằng các giao thức IoT khác nhau và nó thường được sử dụng với MQTT để nhắn tin nhẹ và hiệu quả giữa các thiết bị IoT. Tính linh hoạt này cho phép các nhà phát triển tích hợp ESP8266 vào hệ sinh thái IoT hiện có một cách dễ dàng.
- **Tích hợp với các nền tảng IoT:** Nhiều nền tảng IoT, chẳng hạn như AWS IoT, Google Cloud IoT và Microsoft Azure IoT, cung cấp hỗ trợ cho ESP8266. Điều này cho phép tích hợp liền mạch với các dịch vụ đám mây, cho phép lưu trữ dữ liệu, phân tích và điều khiển từ xa các thiết bị IoT.
- **Tích hợp cảm biến:** ESP8266 có thể giao tiếp với các cảm biến và bộ truyền động khác nhau thông qua các chân GPIO của nó. Điều này làm cho nó phù hợp để thu thập dữ liệu từ các cảm biến (ví dụ: nhiệt độ, độ ẩm, chuyển động) và các thiết bị điều khiển trong các ứng dụng IoT.
- **Cập nhật OTA (Over-the-Air):** Công ESP8266 hỗ trợ cập nhật OTA, cho phép cập nhật firmware không dây. Tính năng này rất quan trọng đối với các thiết bị IoT được triển khai tại hiện trường, vì nó cho phép cập nhật từ xa mà không cần truy cập vật lý vào thiết bị.
- **Cấu hình Wi-Fi:** ESP8266 có thể được cấu hình để kết nối với mạng Wi-Fi, giúp dễ dàng tích hợp các thiết bị IoT vào các mạng hiện có. Nó hỗ trợ cả chế độ trạm (kết nối với mạng Wi-Fi hiện có) và chế độ điểm truy cập (hoạt động như một điểm phát sóng Wi-Fi).
- **Hạn chế:** Mặc dù ESP8266 là một mô-đun mạnh mẽ và linh hoạt, nhưng nó có những hạn chế, chẳng hạn như chân GPIO hạn chế và dung lượng bộ nhớ flash tương đối nhỏ. Trong một số trường hợp, những hạn chế này có thể yêu cầu phần cứng bổ sung hoặc quản lý tài nguyên cẩn thận trong phát triển phần mềm.



Hình 4.3: Mô-đun ESP8266

Trong kịch bản trên, ta sẽ sử dụng tính năng nhảy đèn có sẵn trên mô-đun để kiểm chứng xem ta đã kết nối thành công chưa. Bằng cách cắm trực tiếp mô-đun vào máy tính và cho chạy chương trình Arduino và thực hiện kết nối bằng MQTT Explorer, ta sẽ có được sự liên kết giữa mô-đun và MQTT broker.

V. Kết quả

Sau khi thiết lập đầy đủ các bên, ta sẽ tiến hành thực nghiệm kịch bản trên. Đầu tiên ta cho chạy chương trình đã lập trình trên Arduino lên mô-đun, sau đó khởi chạy MQTT Explorer để tiến hành kết nối. Thiết lập "Connection" như trên để tạo kết nối MQTT Broker. Sau khi kết nối thành công, giao diện sẽ hiện ra dạng gửi tin nhắn. Nhiệm vụ của ta đơn giản chỉ cần gõ lệnh on/off ở phần topic như bên dưới để điều khiển đèn.



Hình 5.1: Gửi thông điệp sang ESP8266

Và sau khi gửi lệnh, đèn sẽ bật/tắt theo yêu cầu. Kiểm tra trên Serial Monitor ta sẽ nhận được thông tin tương ứng. Chi tiết cách thực nghiệm trên video đính kèm.

Trong ví dụ điều khiển đèn LED của ESP8266 thông qua MQTT, chúng ta chứng kiến một ứng dụng thực tế của MQTT trong môi trường IoT. Giao thức này không chỉ giúp thiết lập một cầu nối linh hoạt giữa thiết bị và broker mà còn tạo ra một mô hình truyền thông mạnh mẽ giữa chúng. Việc sử dụng mô hình xuất bản - đăng ký của MQTT cho phép ESP8266 gửi và nhận thông điệp một cách hiệu quả, tạo ra một cơ chế đơn giản nhưng hiệu quả để điều khiển đèn LED.

Cơ chế "Keep-Alive" trong MQTT đảm bảo tính ổn định của kết nối giữa thiết bị và broker, giúp ngăn chặn việc mất kết nối không mong muốn. Thông điệp phản hồi về trạng thái của đèn LED tạo ra một cơ sở cho việc giám sát và duy trì tính nhất quán trong hệ thống IoT.

Với mã nguồn ngắn và dễ hiểu, ví dụ này là một bước nhảy vọt chắc chắn cho những người mới bắt đầu với MQTT và ESP8266. Qua ví dụ này, chúng ta thấy cách MQTT có thể được tích hợp một cách linh hoạt để tạo ra các ứng dụng IoT đơn giản nhưng hiệu quả. Điều này thúc đẩy sự phát triển và triển khai của các giải pháp IoT thực tế, giúp kết nối và quản lý thiết bị một cách hiệu quả trong thế giới ngày càng kết nối.

VI. Kết luận

MQTT (Message Queuing Telemetry Transport) là một giao thức truyền thông nhẹ và linh hoạt, chủ yếu được sử dụng trong môi trường IoT và nhà thông minh. Với mô hình xuất bản - đăng ký và hỗ trợ ba cấp độ chất lượng dịch vụ (QoS), MQTT cung cấp một giải pháp hiệu quả cho việc truyền thông giữa các thiết bị khác nhau. Sự độc lập và tích hợp dễ dàng với các nền tảng nhà thông minh như OpenHAB, Home Assistant làm cho MQTT trở thành một tiêu chuẩn quan trọng, đồng thời đảm bảo đảm bảo an toàn và bảo mật thông tin trong quá trình giao tiếp. Với sự linh hoạt và độ ổn định, MQTT tiếp tục đóng vai trò quan trọng trong việc xây dựng các hệ thống thông minh và kết nối, góp phần vào sự phát triển của IoT và các ứng dụng nhà thông minh.

MQTT, với thiết kế nhẹ và khả năng mở rộng, đặc biệt phù hợp cho các thiết bị có tài nguyên hạn chế. Khả năng đảm bảo tin cậy thông qua các cấp độ QoS cùng với mô hình xuất bản - đăng ký tạo ra một cơ sở cho truyền thông linh hoạt và hiệu quả. Sự đa nền tảng và tích hợp dễ dàng với nhiều loại thiết bị và hệ điều hành khác nhau làm cho MQTT trở thành một tiêu chuẩn phổ cập trong cộng đồng IoT. Với khả năng mở và sự linh hoạt, MQTT đã trở thành một phần quan trọng của các hệ thống nhà thông minh và ứng dụng IoT đa dạng. Sự an toàn và bảo mật được tích hợp giúp đối mặt với thách thức bảo mật ngày càng cao trong các môi trường kết nối mạng không an toàn. Khả năng đồng bộ và duy trì trạng thái của các thiết bị thông minh giúp tạo ra trải nghiệm người dùng mượt mà và nhất quán. Tóm lại, MQTT không chỉ là một giao thức truyền thông mạnh mẽ mà còn là một tiêu chuẩn đóng vai trò quan trọng trong sự phát triển của IoT và nhà thông minh.

Tài liệu tham khảo

- [1] <https://mqtt.org/>
- [2] <https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt/>
- [3] https://www.researchgate.net/figure/MQTT-frame-format-of-CONNECT-packet_fig1_331529805
- [4] <https://data-flair.training/blogs/mqtt-protocol/>
- [5] Ebleme, M. A., Bayilmis, C., & Cavusoglu, U. (2018, September). Examination and Performance Evaluation of MQTT. In *3rd International Conference on Computer Science and Engineering* (pp. 246-250).
- [6] <https://data-flair.training/blogs/mqtt-protocol/>

Phân công công việc

Thành viên	Công việc	Tiến độ
Trương Đình Trọng Thanh	Thiết kế slide + viết báo cáo + demo	100%
Trần Thúy Anh	Lên ý tưởng, nội dung + viết báo cáo + demo	100%