BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC HUẾ

TRƯỜNG ĐẠI HỌC KHOA HỌC

ĐỀ TÀI TIỂU LUẬN
CÔNG NGHỆ THÔNG TIN

ĐIỀU KHIỂN TỪ XA QUA GIAO THỨC MQTT VỚI ESP32

Thừa Thiên Huế, 2025

MỤC LỤC

LỜI CẢM ƠN

Trong suốt thời gian học tập và rèn luyện tại học kỳ vừa qua cho đến nay, em đã nhận được rất nhiều sự quan tâm, giúp đỡ của thầy ThS.Võ Việt Dũng. Thầy đã cho em nhiều ý tưởng và thông tin để hoàn thành đề tài này.

Và đặc biệt, trong kỳ này, Khoa đã tổ chức cho chúng em được tiếp cận với môn học mà em thấy rất hữu ích đối với sinh viên ngành Khoa Học Máy Tính cũng như các chuyên ngành khác. Em một lần nữa xin cảm ơn thầy ThS.Võ Việt Dũng đã tận tâm hướng dẫn, giảng dạy chúng em qua từng buổi học trên lớp cũng như những buổi nói chuyện, thảo luận về lĩnh vực Phát triển ứng dụng IoT này.

Cuối cùng, trong điều kiện thời gian cũng như kinh nghiệm còn hạn chế, bài tiểu luận này không thể tránh được thiếu sót. Em rất mong được sự góp ý, chỉ bảo của các thầy, cô để em có điều kiện bổ sung, nâng cao kĩ năng của mình trong các đề tài sau.

Em xin chân thành cảm ơn!

Thừa Thiên Huế, tháng 04 năm 2025

DANH MỤC CHỮ VIẾT TẮT

Từ viết tắt	Từ hoặc cụm từ
MQTT	Message Queuing Telemetry Transport
IoT	Internet of Thing
OASIS	Organization for the Advancement of Structured Information Standards
HTTP	Hyper Text Transfer Protocol
LWT	Last Will and Testament
LED	Light Emitting Diode
SSL/TLS	Secure Sockets Layer/ Transport Layer Security

MỞ ĐẦU

Trong bối cảnh phát triển nhanh chóng của công nghệ, Internet vạn vật (Internet of Things - IoT) đã và đang trở thành một xu hướng không thể phủ nhận, mang lại những thay đổi đáng kể trong cách chúng ta tương tác với thế giới xung quanh. Sự kết nối giữa con người và thiết bị, cũng như giữa các thiết bị với nhau, đã tạo nên một mạng lưới thông minh cho phép tự động hóa và tối ưu hóa nhiều quy trình trong cuộc sống hàng ngày.

Một trong những thách thức lớn nhất của IoT là việc truyền thông hiệu quả giữa các thiết bị có nguồn tài nguyên hạn chế, trong môi trường mạng không ổn định hoặc có băng thông thấp. Điều này đòi hỏi một giao thức truyền thông nhẹ, tin cậy và tiết kiệm năng lượng. MQTT (Message Queuing Telemetry Transport) ra đời như một giải pháp lý tưởng cho bài toán này.

Bên cạnh đó, với sự xuất hiện của các vi điều khiển mạnh mẽ, giá thành phải chăng như ESP32, việc xây dựng các hệ thống IoT đã trở nên dễ dàng và tiếp cận hơn bao giờ hết. ESP32 với khả năng kết nối Wi-Fi và Bluetooth tích hợp, cùng với hiệu suất xử lý cao, đã trở thành lựa chọn phổ biến cho các dự án IoT từ quy mô nhỏ đến lớn.

Tiểu luận này hướng đến việc nghiên cứu và triển khai một hệ thống điều khiển từ xa thông qua giao thức MQTT với ESP32, nhằm cung cấp một cái nhìn toàn diện về cách thức hoạt động, ưu điểm và khả năng ứng dụng thực tiễn của mô hình này trong các hệ thống IoT hiện đại.

CHƯƠNG 1: TỔNG QUAN

1. Giới thiệu

1.1. Lý do chọn đề tài

Việc nghiên cứu đề tài "Điều khiển từ xa qua giao thức MQTT với ESP32" xuất phát từ nhiều lý do quan trọng:

Thứ nhất, IoT đang phát triển với tốc độ chóng mặt và đã trở thành một phần không thể thiếu trong cuộc cách mạng công nghiệp 4.0. Theo báo cáo của Statista, số lượng thiết bị IoT toàn cầu dự kiến sẽ tăng từ 13,8 tỷ thiết bị năm 2021 lên đến 30,9 tỷ vào năm 2025. Điều này tạo ra nhu cầu lớn về các giải pháp kết nối và điều khiển thiết bị từ xa hiệu quả.

Thứ hai, giao thức MQTT với đặc tính nhẹ, tiết kiệm băng thông và năng lượng, đã trở thành một tiêu chuẩn thực tế trong lĩnh vực IoT. Hiểu rõ về MQTT và cách ứng dụng nó là kiến thức thiết yếu đối với các nhà phát triển IoT hiện đại.

Thứ ba, ESP32 là một vi điều khiển mạnh mẽ nhưng giá thành phải chăng, mang lại khả năng xây dựng các hệ thống IoT hiệu quả mà không đòi hỏi đầu tư lớn. Sự kết hợp giữa MQTT và ESP32 tạo nên một nền tảng lý tưởng cho các dự án IoT từ quy mô nhỏ đến vừa.

Thứ tư, mô hình điều khiển từ xa qua MQTT có tính ứng dụng thực tiễn cao, đặc biệt trong các lĩnh vực như nhà thông minh, nông nghiệp thông minh, giám sát môi trường và nhiều ứng dụng công nghiệp khác.

Thứ năm, sự phát triển của các nền tảng mô phỏng như Wokwi cho phép nghiên cứu và thử nghiệm các hệ thống IoT mà không cần đầu tư nhiều vào phần cứng thực, giúp quá trình học tập và nghiên cứu trở nên dễ dàng và tiết kiệm hơn.

1.2. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu của tiểu luận này bao gồm:

- Giao thức MQTT: Cấu trúc, nguyên lý hoạt động, và ứng dụng trong IoT.

- Vi điều khiển ESP32: Kiến trúc, tính năng, và khả năng ứng dụng trong các hệ thống IoT.
- Mô hình điều khiển từ xa: Cách thức thiết kế, triển khai, và vận hành một hệ thống điều khiển từ xa sử dụng MQTT và ESP32.

Phạm vi nghiên cứu được giới hạn trong:

- Nghiên cứu về giao thức MQTT phiên bản 3.1.1, tập trung vào các đặc tính và ứng dụng cốt lõi.
- Sử dụng ESP32 làm vi điều khiển chính, với các thành phần phụ trợ bao gồm LED và cảm biến DHT22.
- Sử dụng Mosquitto làm MQTT broker, không đi sâu vào các broker thương mại khác.
- Mô phỏng hệ thống trên nền tảng Wokwi, không bao gồm triển khai thực tế trên quy mô lớn.
- Ứng dụng trong lĩnh vực nhà thông minh và các hệ thống IoT quy mô nhỏ đến vừa.

Tiểu luận không đi sâu vào các vấn đề về bảo mật cao cấp, tối ưu hóa hiệu suất cho các hệ thống lớn, hoặc các phiên bản MQTT mới nhất như MQTT 5.0.

1.3. Cấu trúc của tiểu luận

Tiểu luận được tổ chức thành các chương như sau:

- **Mở đầu**: Giới thiệu tổng quan về tính cấp thiết và ý nghĩa của đề tài.
- **Chương 1: Tổng quan**: Trình bày lý do chọn đề tài, đối tượng và phạm vi nghiên cứu, cũng như cấu trúc của tiểu luận.
- **Chương 2: Cơ sở lý thuyết**: Cung cấp kiến thức nền tảng về giao thức MQTT, mô hình publish/subscribe, ESP32, các thành phần phần cứng sử dụng, và vai trò của MQTT broker (Mosquitto) trong hệ thống IoT.
- **Chương 3: Thiết kế hệ thống**: Mô tả chi tiết về kiến trúc hệ thống và mô hình hoạt động của hệ thống

- **Chương 4: Mô phỏng trên Wokwi**: Trình bày quy trình mô phỏng hệ thống trên nền tảng Wokwi, các kịch bản thử nghiệm, và kết quả đạt được. Chương này cũng cung cấp những đánh giá về hiệu suất của hệ thống.
- **Chương 5: Kết luận**: Tổng kết những ưu điểm, khả năng mở rộng của mô hình đã thiết kế, các ứng dụng thực tiễn trong nhà thông minh và IoT nói chung, cũng như đề xuất các hướng phát triển trong tương lai.

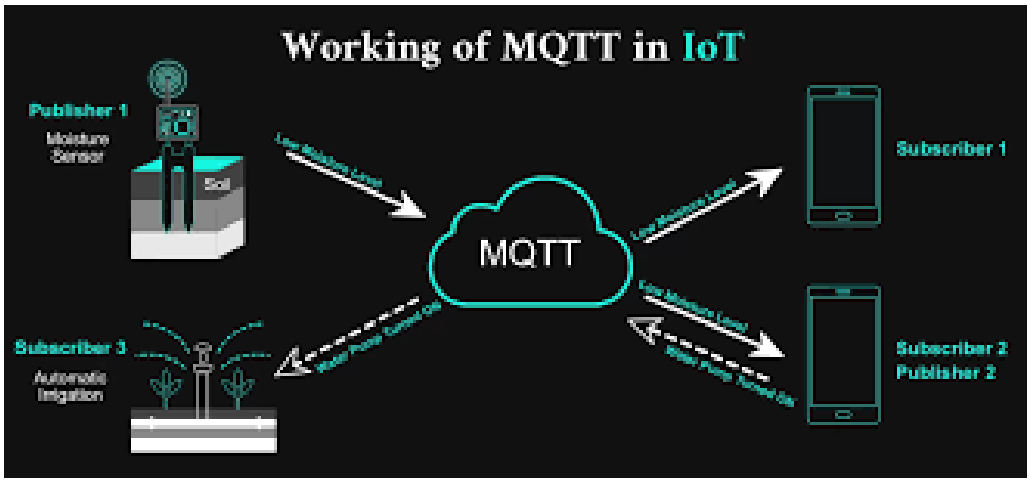
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Tổng quan về giao thức MQTT

2.1.1. Khái niệm về giao thức MQTT

MQTT (Message Queuing Telemetry Transport) là một giao thức truyền tin nhẹ, hoạt động theo mô hình Client-Server, được thiết kế đặc biệt cho các thiết bị có tài nguyên hạn chế và mạng có băng thông thấp, độ trễ cao hoặc không ổn định. Giao thức này được Andy Stanford-Clark (IBM) và Arlen Nipper (Eurotech) phát triển vào năm 1999, ban đầu để giám sát đường ống dầu qua sa mạc.

MQTT hoạt động trên nền tảng TCP/IP và hiện đã trở thành một tiêu chuẩn OASIS (Organization for the Advancement of Structured Information Standards) và ISO (ISO/IEC 20922). Các phiên bản phổ biến của MQTT hiện nay bao gồm MQTT 3.1.1 và MQTT 5.0 với nhiều cải tiến về tính năng.



Hình 2.1.1: Tổng quan giao thức MQTT

2.1.2. Cách thức hoạt động(Publish-Subscribe)

MQTT hoạt động theo mô hình Publish-Subscribe (Xuất bản-Đăng ký), khác với mô hình Request-Response truyền thống của HTTP. Trong mô hình này:

- **Publisher (Nhà xuất bản)**: Là thiết bị gửi thông điệp. Publisher không gửi trực tiếp đến Subscriber mà gửi đến một trung gian gọi là Broker.
- **Subscriber (Người đăng ký)**: Là thiết bị nhận thông điệp. Subscriber đăng ký với Broker để nhận các thông điệp từ một chủ đề (topic) cụ thể.
- **Broker (Trung gian)**: Là máy chủ trung tâm, nhận thông điệp từ Publishers và chuyển tiếp đến Subscribers đã đăng ký.
- **Topic (Chủ đề)**: Là chuỗi phân cấp được sử dụng để lọc và định tuyến thông điệp.

Quá trình hoạt động cơ bản như sau:

1. Subscriber đăng ký với Broker để nhận thông điệp từ một hoặc nhiều topic.
2. Publisher gửi thông điệp đến một topic cụ thể trên Broker.
3. Broker nhận thông điệp và chuyển tiếp đến tất cả Subscribers đã đăng ký với topic đó.

Mô hình này cho phép giao tiếp một-nhiều (one-to-many) hoặc nhiều-nhiều (many-to-many) một cách hiệu quả, đồng thời tách biệt Publisher và Subscriber về mặt không gian và thời gian.

2.1.3. Ưu điểm của MQTT so với HTTP trong IoT

So với HTTP, MQTT có nhiều ưu điểm khi ứng dụng trong môi trường IoT:

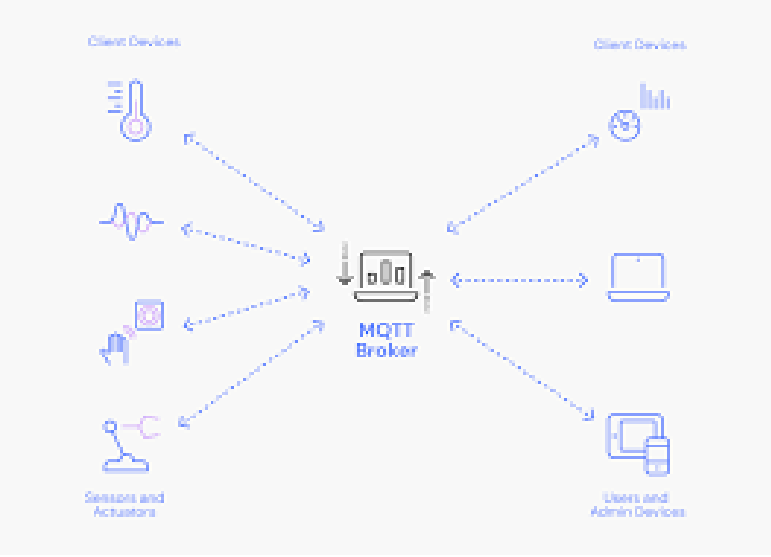
- **Hiệu quả về băng thông**: MQTT có overhead thấp hơn nhiều so với HTTP. Header của gói tin MQTT chỉ có vài byte, trong khi HTTP có thể lên đến hàng trăm byte.
- **Tiết kiệm năng lượng**: Do overhead thấp và không duy trì kết nối liên tục, MQTT giúp thiết bị tiết kiệm pin hơn.
- **Độ tin cậy cao**: MQTT cung cấp ba mức Quality of Service (QoS) để đảm bảo độ tin cậy trong việc gửi nhận thông điệp:
    - QoS 0: Gửi và quên (at most once)
    - QoS 1: Ít nhất một lần (at least once)
    - QoS 2: Đúng một lần (exactly once)

- **Mô hình Push**: MQTT sử dụng mô hình Push, cho phép server chủ động gửi thông điệp đến client mà không cần client liên tục truy vấn (polling).
- **Khả năng mở rộng**: Mô hình Publish-Subscribe cho phép hệ thống dễ dàng mở rộng với nhiều thiết bị.
- **Hỗ trợ kết nối không ổn định**: MQTT có khả năng phục hồi tốt khi mạng bị ngắt kết nối tạm thời nhờ tính năng Last Will and Testament và Retained Messages.

2.1.4. Các thành phần chính của MQTT

1. MQTT Broker

- Là trung tâm của hệ thống MQTT, quản lý việc nhận và phân phối thông điệp.
- Chịu trách nhiệm xác thực clients, quản lý đăng ký topics, lưu trữ và chuyển tiếp thông điệp.
- Các MQTT Broker phổ biến: Mosquitto, HiveMQ, EMQX, AWS IoT Core.



Hình 2.1.4: Tổng quan giao thức MQTT Broker

2. MQTT Client

- Bất kỳ thiết bị nào kết nối đến MQTT Broker đều được gọi là MQTT Client.
- Client có thể là Publisher, Subscriber hoặc cả hai.
- Ví dụ: Cảm biến IoT, điện thoại thông minh, máy tính, vi điều khiển như ESP32.

3. MQTT Connection

- Kết nối TCP/IP giữa Client và Broker.
- Có thể được bảo mật bằng TLS/SSL.
- Bắt đầu bằng gói tin CONNECT từ Client và CONNACK từ Broker.

4. MQTT Messages

- Là đơn vị dữ liệu cơ bản trong MQTT.
- Bao gồm header, topic và payload (nội dung thông điệp).
- Payload có thể là text, JSON, XML, binary, v.v.

5. MQTT Topics

- Chuỗi UTF-8 phân cấp, sử dụng dấu "/" để phân tách các cấp.
- Có thể sử dụng wildcard: "+" (một cấp) và "#" (nhiều cấp).
- Ví dụ: "home/+/temperature" sẽ khớp với "home/livingroom/temperature" và "home/bedroom/temperature".

6. Các tính năng bổ sung

- **Retained Messages**: Thông điệp được Broker lưu giữ và gửi cho Client mới đăng ký.
- **Last Will and Testament (LWT)**: Thông điệp được gửi khi Client ngắt kết nối bất thường.
- **Clean/Persistent Sessions**: Quyết định liệu Broker có lưu trạng thái Client khi ngắt kết nối hay không.
- **Keep Alive**: Cơ chế để duy trì kết nối giữa Client và Broker.

2.2. Giới thiệu về ESP32 và các thành phần cứng được sử dụng

2.2.1. Đặc điểm kỹ thuật của ESP32

ESP32 là vi điều khiển được phát triển bởi Espressif Systems, ra mắt vào năm 2016 như một bước tiến từ ESP8266. Nó được thiết kế để đáp ứng nhu cầu ngày càng tăng của các ứng dụng IoT, với sự kết hợp giữa hiệu năng cao, khả năng kết nối không dây và chi phí thấp.

Hình 2.2.1: Bản mạch ESP32

ESP32 được trang bị một CPU lõi kép Xtensa LX6, hoạt động ở tần số lên đến 240 MHz, cho phép nó xử lý đồng thời nhiều tác vụ mà không bị gián đoạn. Ngoài ra, module này tích hợp:

- **Wi-Fi:** Cho phép ESP32 kết nối trực tiếp với mạng internet hoặc mạng nội bộ, giao tiếp với MQTT broker mà không cần thiết bị trung gian.
- **Bluetooth:** Dù không được sử dụng trực tiếp trong hệ thống này, tính năng này mở ra tiềm năng kết nối với các thiết bị ngoại vi khác trong tương lai.
- **GPIO (General Purpose Input/Output):** Các chân giao tiếp đa dụng hỗ trợ kết nối với cảm biến, công tắc, đèn LED hoặc bất kỳ thiết bị nào cần điều khiển.
- **Bộ nhớ:** Với RAM 520 KB và khả năng mở rộng bộ nhớ flash, ESP32 đủ sức lưu trữ chương trình phức tạp và xử lý dữ liệu thời gian thực.

Những đặc điểm này khiến ESP32 không chỉ là một vi điều khiển đơn thuần mà còn là một nền tảng linh hoạt, phù hợp cho các ứng dụng IoT yêu cầu tính tương tác cao.

2.2.2. Vai trò của ESP32 trong hệ thống

Trong hệ thống điều khiển từ xa, ESP32 đóng vai trò là một **MQTT client**, vừa hoạt động như **publisher** (nhà xuất bản) vừa là **subscriber** (người đăng ký), tùy thuộc vào tình huống cụ thể. Hãy xem xét cách nó hoạt động trong từng vai trò:

- **ESP32 như một Publisher:** Khi được kết nối với các cảm biến hoặc thiết bị đầu vào (ví dụ: nút nhấn, cảm biến nhiệt độ), ESP32 thu thập dữ liệu từ những thiết bị này và gửi thông điệp đến MQTT broker thông qua một topic cụ thể. Chẳng hạn, nếu một nút nhấn được kích hoạt, ESP32 có thể gửi thông điệp "ON" đến topic "home/light/status" để thông báo trạng thái đèn đã thay đổi. Quá trình này diễn ra nhanh chóng nhờ khả năng kết nối Wi-Fi ổn định và hiệu suất xử lý cao của module.
- **ESP32 như một Subscriber:** Ngược lại, ESP32 cũng có thể nhận lệnh từ MQTT broker thông qua các topic mà nó đã đăng ký. Ví dụ, nếu người dùng gửi thông điệp "OFF" đến topic "home/light/control" thông qua một ứng dụng điều khiển, ESP32 sẽ nhận được thông điệp này từ broker và thực hiện hành động tương ứng, như tắt đèn LED hoặc ngắt relay. Khả năng phản hồi tức thời của ESP32 đảm bảo hệ thống vận hành mượt mà, ngay cả khi có nhiều lệnh được gửi cùng lúc.

2.2.3. Ưu điểm của ESP32 trong ứng dụng điều khiển từ xa

Việc lựa chọn ESP32 cho hệ thống mang lại nhiều lợi ích:

- **Tính linh hoạt:** ESP32 có thể được cấu hình để điều khiển nhiều loại thiết bị khác nhau, từ đèn đơn giản đến hệ thống cảm biến phức tạp.
- **Khả năng mở rộng:** Nhờ hỗ trợ Wi-Fi và số lượng lớn chân GPIO, ESP32 dễ dàng tích hợp thêm thiết bị mới mà không cần thay đổi cấu trúc hệ thống.
- **Chi phí thấp:** So với các giải pháp vi điều khiển khác, ESP32 có giá thành hợp lý, phù hợp cho cả ứng dụng cá nhân lẫn thương mại.
- **Hiệu suất cao:** CPU lõi kép và khả năng kết nối không dây giúp ESP32 xử lý tốt các tác vụ trong môi trường IoT đòi hỏi tốc độ và độ tin cậy.

2.2.4. Các thành phần cứng được sử dụng

- LED - Thiết bị đầu ra mẫu

LED (Light Emitting Diode) được sử dụng như một thiết bị đầu ra đơn giản, đại diện cho đèn trong nhà thông minh. Nó được kết nối với một chân GPIO của ESP32 và có thể bật hoặc tắt dựa trên tín hiệu điện (mức cao để bật, mức thấp để tắt). LED minh họa khả năng điều khiển từ xa của hệ thống thông qua MQTT.

Hình 2.2.4: Đèn LED

- Điện trở LED - Bảo vệ và ổn định dòng điện

Điện trở LED là một thành phần quan trọng trong mạch, được nối tiếp với LED để hạn chế dòng điện chạy qua, tránh tình trạng quá tải hoặc cháy hỏng.

- **Vai trò:** Điều chỉnh dòng điện từ chân GPIO của ESP32 (thường 3,3V) xuống mức an toàn cho LED (khoảng 10-20 mA, tùy loại LED).
- **Giá trị:** Điện trở thường được tính theo định luật Ohm. Ví dụ, với nguồn 3,3V, LED 2V, dòng 20 mA, giá trị điện trở là khoảng 65Ω (thường chọn 100Ω hoặc 220Ω cho an toàn).
- **Tích hợp:** Điện trở được nối giữa chân GPIO của ESP32 và cực dương (anode) của LED, trong khi cực âm (cathode) nối với GND.

Hình 2.2: Điện trở LED

- Cảm biến DHT22 - Nguồn dữ liệu giám sát

DHT22 là cảm biến đo nhiệt độ và độ ẩm phổ biến trong các dự án IoT. Đặc điểm của nó bao gồm:

- Phạm vi đo nhiệt độ: -40°C đến 80°C.
- Phạm vi đo độ ẩm: 0% đến 100%.
- Độ chính xác cao, dễ kết nối với ESP32 qua một chân GPIO.

DHT22 cung cấp dữ liệu thực tế để giám sát môi trường, như nhiệt độ phòng hoặc độ ẩm không khí. Trong trường hợp mô phỏng, dữ liệu ngẫu nhiên (random) có thể được sử dụng thay thế để kiểm tra hệ thống mà không cần phần cứng thực tế.

Hình 2.2.4: Cảm biến DHT22

2.3. Mosquitto MQTT Broker

2.3.1. Giới thiệu về Mosquitto

Mosquitto là một MQTT broker mã nguồn mở, nhẹ nhàng, phù hợp cho tất cả các thiết bị từ máy tính có hiệu suất thấp đến máy chủ đầy đủ. Dự án Mosquitto thuộc Eclipse Foundation và hỗ trợ các phiên bản MQTT 5.0, 3.1.1 và 3.1. Mosquitto được phát triển bởi Roger Light và được phân phối theo giấy phép EPL v2.0/EDL, cho phép nó được sử dụng miễn phí trong cả dự án cá nhân và thương mại (Eclipse Foundation, 2024).

Hình 2.3.1: Ứng dụng Mosquitto

Mosquitto là một trong những MQTT broker phổ biến nhất vì những lý do sau:

1. **Nhẹ nhàng:** Sử dụng rất ít tài nguyên hệ thống, phù hợp với các thiết bị như Raspberry Pi
2. **Đa nền tảng:** Hoạt động trên nhiều hệ điều hành, bao gồm Windows, Linux, macOS
3. **Đầy đủ tính năng:** Hỗ trợ các tính năng MQTT tiêu chuẩn như QoS, retained messages, last will và testament
4. **Bảo mật:** Hỗ trợ xác thực người dùng và mã hóa TLS/SSL
5. **Hiệu suất cao:** Có thể xử lý hàng nghìn kết nối đồng thời, phù hợp cho các ứng dụng IoT quy mô lớn

Cách Mosquitto hoạt động trong hệ thống bao gồm:

- **Nhận thông điệp:** Khi một ESP32 (publisher) gửi thông điệp đến một topic, Mosquitto sẽ tiếp nhận và lưu trữ tạm thời.
- **Chuyển tiếp thông điệp:** Mosquitto kiểm tra danh sách các subscriber đang theo dõi topic đó và gửi thông điệp đến từng thiết bị một cách chính xác.
- **Quản lý kết nối:** Mosquitto duy trì kết nối với tất cả các client, đảm bảo thông tin được truyền tải ngay cả khi mạng gặp sự cố (nhờ cơ chế QoS).

Mosquitto còn hỗ trợ các tính năng bảo mật như xác thực bằng tên người dùng/mật khẩu và mã hóa SSL/TLS, giúp bảo vệ dữ liệu trong quá trình truyền tải. Trong hệ thống điều khiển từ xa, Mosquitto đóng vai trò như một "trạm điều phối", đảm bảo các lệnh từ người dùng được gửi đến đúng thiết bị một cách nhanh chóng và an toàn.

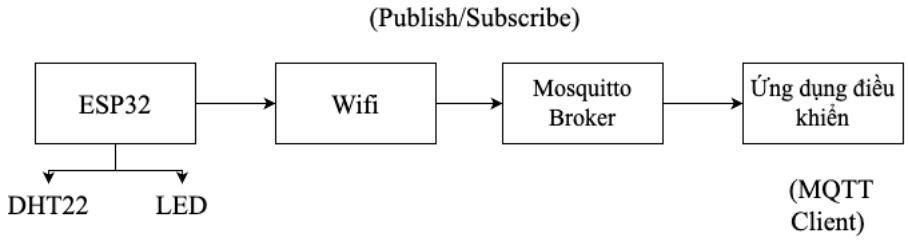
CHƯƠNG 3: THIẾT KẾ HỆ THỐNG

3. Thiết kế hệ thống

Phần này trình bày chi tiết thiết kế hệ thống điều khiển từ xa qua giao thức MQTT với ESP32, bao gồm kiến trúc tổng quan, sơ đồ kết nối phần cứng, mô hình hoạt động, và luồng dữ liệu qua các topic MQTT. Hệ thống được thiết kế để thực hiện hai chức năng chính: điều khiển thiết bị từ xa (bật/tắt đèn LED) và giám sát dữ liệu môi trường (nhiệt độ, độ ẩm từ cảm biến DHT22). Toàn bộ hệ thống được mô phỏng trên Wokwi, với trọng tâm là lý thuyết MQTT và khả năng triển khai thực tế. Thiết kế tận dụng mô hình publish/subscribe để đảm bảo giao tiếp hiệu quả, đồng thời tích hợp các thành phần phần cứng cơ bản để minh họa ứng dụng IoT trong nhà thông minh.

3.1. Kiến trúc hệ thống

Hệ thống điều khiển từ xa qua MQTT với ESP32 được xây dựng dựa trên mô hình publish/subscribe, với các thành phần chính phối hợp để đảm bảo giao tiếp hiệu quả giữa thiết bị và người dùng. Kiến trúc tổng quan bao gồm ba thành phần chính, được thiết kế để tối ưu hóa luồng dữ liệu và khả năng điều khiển từ xa.



Hình 3.1: Sơ đồ hoạt động

3.1.1. ESP32 Client

- **Vai trò:** ESP32 đóng vai trò là MQTT client, chịu trách nhiệm kết nối với cảm biến và thiết bị, giao tiếp với MQTT broker, và xử lý dữ liệu/lệnh. Nó là trung tâm điều phối giữa phần cứng vật lý và mạng MQTT.
- **Chức năng cụ thể:**
    - **Kết nối với cảm biến DHT22:** ESP32 đọc dữ liệu nhiệt độ và độ ẩm từ DHT22 qua một chân GPIO (ví dụ: GPIO14), cung cấp thông tin môi trường để giám sát.

    - **Điều khiển đèn LED:** ESP32 bật/tắt LED (kết nối qua GPIO4) dựa trên lệnh từ người dùng, minh họa khả năng điều khiển từ xa.
    - **Publish và Subscribe qua MQTT:** ESP32 gửi (publish) dữ liệu cảm biến đến các topic giám sát và nhận (subscribe) lệnh điều khiển từ topic liên quan, tận dụng giao thức MQTT để giao tiếp hai chiều.
- **Đặc điểm kỹ thuật:** Sử dụng module ESP32-WROOM-32, với lõi kép Tensilica Xtensa LX6 (tốc độ tối đa 240 MHz), Wi-Fi (chuẩn 802.11 b/g/n), 34 chân GPIO, 520 KB SRAM, và bộ nhớ Flash 16 MB. Module này được chọn nhờ khả năng kết nối mạng mạnh mẽ, hiệu suất xử lý cao, và hỗ trợ mô phỏng trên Wokwi. CPU lõi kép cho phép ESP32 xử lý đồng thời việc đọc cảm biến và nhận lệnh, đảm bảo không bị gián đoạn.
- **Tích hợp phần cứng:** LED được nối với GPIO4 qua một điện trở (100-220Ω) để hạn chế dòng điện, bảo vệ LED khỏi hư hỏng khi bật/tắt.

3.1.2. MQTT Broker

- **Vai trò:** Máy chủ trung gian quản lý luồng dữ liệu giữa ESP32 và ứng dụng điều khiển, đảm bảo thông điệp được định tuyến chính xác qua các topic. Mosquitto đóng vai trò "bộ não trung gian" trong hệ thống.
- **Đặc điểm kỹ thuật:** Sử dụng Mosquitto, một broker mã nguồn mở do Eclipse Foundation duy trì, hỗ trợ MQTT phiên bản 3.1.1 và 5.0. Mosquitto có thể triển khai trên cloud (như test.mosquitto.org) hoặc server cục bộ (như Raspberry Pi trong ứng dụng thực tế). Các thành phần chính của Mosquitto bao gồm:
    1. **MQTT Protocol Engine:** Xử lý các gói tin MQTT như CONNECT, PUBLISH, SUBSCRIBE, đảm bảo giao tiếp chính xác.
    2. **Topic Management System:** Định tuyến thông điệp qua cấu trúc topic phân cấp, như "home/sensors/dht22/temperature".
    3. **Connection Manager:** Quản lý kết nối TCP với các client (ESP32, ứng dụng), duy trì trạng thái ổn định.
    4. **Message Queue and Storage:** Lưu trữ thông điệp retained (nếu cấu hình) và quản lý hàng đợi QoS để đảm bảo độ tin cậy.

- **Cấu hình trong Wokwi:** Sử dụng broker công cộng test.mosquitto.org (port 1883) để giả lập, không yêu cầu bảo mật TLS nhằm đơn giản hóa mô phỏng. Trong thực tế, TLS có thể được thêm để tăng cường an toàn.
- **Lý do chọn:** Mosquitto nhẹ, hiệu quả, và phù hợp với hệ thống IoT quy mô nhỏ. Nó dễ tích hợp với ESP32 qua Wi-Fi và hỗ trợ định tuyến thông điệp chính xác trong mô hình publish/subscribe.

3.1.3. Ứng dụng điều khiển

- **Vai trò:** Giao diện người dùng để điều khiển thiết bị (bật/tắt LED) và xem dữ liệu cảm biến (nhiệt độ, độ ẩm). Đây là điểm tương tác trực tiếp với hệ thống.
- **Đặc điểm:** Trong mô phỏng Wokwi, ứng dụng điều khiển được giả lập bằng công cụ MQTT client như MQTT Explorer hoặc giao diện Wokwi (Serial Monitor). Trong thực tế, ứng dụng này có thể là phần mềm trên máy tính (như Node-RED) hoặc ứng dụng di động hỗ trợ MQTT (như MQTT Dashboard).
- **Chức năng cụ thể:**
    - **Publish lệnh:** Gửi "ON" hoặc "OFF" đến topic điều khiển để bật/tắt LED.
    - **Subscribe dữ liệu:** Theo dõi các topic cảm biến để nhận nhiệt độ và độ ẩm, cùng với trạng thái LED.
- **Lý do chọn:** Ứng dụng điều khiển cung cấp giao diện trực quan, minh họa luồng dữ liệu hai chiều trong hệ thống MQTT. Nó giúp người dùng tương tác dễ dàng với ESP32 qua Mosquitto, đồng thời phản ánh tính thực tiễn của thiết kế.

3.1.4. Sơ đồ kết nối phần cứng

- **Mô tả:**
    - ESP32 kết nối với LED qua GPIO4, với điện trở 220Ω nối tiếp để bảo vệ (LED dương nối qua điện trở đến GPIO4, âm nối GND).
    - DHT22 kết nối với ESP32 qua GPIO14 (chân dữ liệu), với nguồn 3,3V từ ESP32 và GND chung.
    - ESP32 sử dụng Wi-Fi để giao tiếp với Mosquitto qua mạng giả lập trên Wokwi.
- **Ý nghĩa:** Sơ đồ này đảm bảo phần cứng hoạt động an toàn và ổn định, đồng thời hỗ trợ mô phỏng chính xác trên Wokwi.

3.2. Mô hình hoạt động của hệ thống

Hệ thống hoạt động dựa trên giao thức MQTT theo mô hình publish/subscribe, với ESP32 là trung tâm xử lý, Mosquitto broker làm cầu nối, và ứng dụng điều khiển đóng vai trò giao diện người dùng. Quy trình hoạt động được thiết kế chi tiết qua các bước sau, phản ánh luồng dữ liệu qua các topic MQTT và cách hệ thống đáp ứng yêu cầu từ xa.

3.2.1. ESP32 kết nối với mạng Wi-Fi và MQTT broker

- **Quy trình:**
  - ESP32 khởi động, kết nối đến mạng Wi-Fi giả lập trên Wokwi (SSID và password được cấu hình trước trong mô phỏng).
  - Sau khi kết nối Wi-Fi thành công, ESP32 thiết lập liên kết MQTT với Mosquitto broker (test.mosquitto.org, port 1883) bằng gói tin CONNECT, bao gồm client ID (ví dụ: "ESP32_Client") và thời gian "keep-alive" (60 giây). Mosquitto trả về gói CONNACK để xác nhận kết nối.
  - ESP32 gửi lệnh "subscribe" đến topic "home/controls/led" để sẵn sàng nhận lệnh từ người dùng.
- **Tối ưu hóa:** ESP32 tự động tái kết nối nếu mạng gián đoạn, đảm bảo hệ thống luôn sẵn sàng trong môi trường mô phỏng.

3.2.2. ESP32 đọc dữ liệu từ cảm biến DHT22

- **Quy trình:**
  - ESP32 đọc dữ liệu nhiệt độ và độ ẩm từ DHT22 qua GPIO14 theo chu kỳ (mỗi 10 giây để tránh quá tải Mosquitto).
  - Dữ liệu được xử lý thành giá trị cụ thể (ví dụ: nhiệt độ 25,5°C, độ ẩm 60%) và định dạng thành chuỗi đơn giản (như "25.5" hoặc "60"). Trong triển khai nâng cao, dữ liệu có thể được định dạng JSON (ví dụ: {"temp": 25.5, "humidity": 60}).
- **Tính năng:** Chu kỳ 10 giây được chọn để cân bằng giữa việc cập nhật dữ liệu thường xuyên và giảm tải hệ thống.

3.2.3. ESP32 publish dữ liệu lên các topic cụ thể

- **Quy trình:**
  - Dữ liệu nhiệt độ được gửi (publish) đến topic "home/sensors/dht22/temperature".
  - Dữ liệu độ ẩm được gửi đến topic "home/sensors/dht22/humidity".
  - Mosquitto broker nhận dữ liệu, lưu trữ tạm thời dưới dạng retained message (nếu cấu hình), và phân phối đến các subscriber như ứng dụng điều khiển.
- **Cấu trúc topic:** Sử dụng phân cấp "home/sensors/dht22" để dễ mở rộng (ví dụ: thêm cảm biến khác như "home/sensors/light").

3.2.4. ESP32 subscribe vào topic điều khiển

- **Quy trình:**
  - ESP32 đã subscribe topic "home/controls/led" từ bước kết nối ban đầu.
  - Ứng dụng điều khiển gửi (publish) lệnh "ON" hoặc "OFF" đến topic này với QoS 1 để đảm bảo truyền ít nhất một lần.
  - Mosquitto định tuyến lệnh đến ESP32 dựa trên danh sách subscriber.
- **Ý nghĩa:** QoS 1 được chọn để cân bằng giữa độ tin cậy và hiệu suất trong mô phỏng.

3.2.5. ESP32 thực hiện hành động tương ứng

- **Quy trình:**
  - Nếu nhận "ON" từ "home/controls/led", ESP32 đặt GPIO4 ở mức cao (HIGH) để bật LED qua điện trở 220Ω.
  - Nếu nhận "OFF", ESP32 đặt GPIO4 ở mức thấp (LOW) để tắt LED.
  - ESP32 gửi (publish) trạng thái mới (ví dụ: "LED is ON") đến topic "home/controls/led/status" để phản hồi.
- **An toàn:** Điện trở LED đảm bảo dòng điện ổn định, tránh hư hỏng khi bật/tắt liên tục.

3.2.6. Người dùng theo dõi dữ liệu và điều khiển thiết bị

- **Quy trình:**
  - Ứng dụng điều khiển (giả lập qua MQTT Explorer hoặc Wokwi Serial Monitor) subscribe các topic "home/sensors/dht22/temperature", "home/sensors/dht22/humidity", và "home/controls/led/status" để nhận dữ liệu và trạng thái.
  - Người dùng xem nhiệt độ, độ ẩm, trạng thái LED trên giao diện và gửi lệnh điều khiển (như "ON") khi cần.
- **Tính trực quan:** Giao diện đơn giản hóa việc tương tác, minh họa luồng dữ liệu hai chiều trong MQTT.

3.2.7. Luồng dữ liệu qua topic MQTT

- **Giám sát:** DHT22 → ESP32 → Mosquitto ("home/sensors/dht22/temperature", "home/sensors/dht22/humidity") → ứng dụng.
- **Điều khiển:** Ứng dụng → Mosquitto ("home/controls/led") → ESP32 → LED.
- **Phản hồi:** ESP32 → Mosquitto ("home/controls/led/status") → ứng dụng.
- **Ý nghĩa:** Luồng này thể hiện sự hiệu quả của mô hình publish/subscribe, nơi các thành phần giao tiếp gián tiếp qua Mosquitto.

CHƯƠNG 4: MÔ PHỎNG HỆ THỐNG TRÊN WOKWI

4. Mô phỏng hệ thống trên wokwi

Phần này trình bày chi tiết quá trình mô phỏng hệ thống điều khiển từ xa qua giao thức MQTT với ESP32 trên nền tảng Wokwi. Mô phỏng bao gồm việc thiết lập môi trường, kết nối đến MQTT broker công cộng, và kiểm tra hoạt động của hệ thống (đọc dữ liệu cảm biến, điều khiển LED, gửi tin nhắn). Wokwi cho phép kiểm tra lý thuyết mà không cần phần cứng thực tế, giúp minh họa rõ ràng cách hệ thống hoạt động trong môi trường giả lập.

4.1. Thiết lập môi trường wokwi

Wokwi là một nền tảng mô phỏng phần cứng trực tuyến, ra mắt vào khoảng năm 2020, được thiết kế để hỗ trợ lập trình viên và nhà phát triển IoT thử nghiệm các dự án vi điều khiển như Arduino, ESP32, và Raspberry Pi Pico mà không cần phần cứng vật lý. Wokwi hoạt động trực tiếp trên trình duyệt web, tích hợp trình chỉnh sửa mã (dựa trên VS Code) và trình biên dịch, cho phép mô phỏng các linh kiện như cảm biến, LED, và kết nối mạng. Trong hệ thống này, Wokwi được sử dụng để mô phỏng ESP32, cảm biến DHT22, và đèn LED, cùng với giao tiếp MQTT.

Hình 4.1: Nền tảng Wokwi

4.1.1. Thiết lập môi trường mô phỏng

- **Truy cập Wokwi:** Mô phỏng bắt đầu bằng cách truy cập wokwi.com, chọn "New Project" và chọn "ESP32" làm nền tảng chính.
- **Thêm linh kiện:** Từ menu linh kiện, các thành phần sau được thêm vào:
    - **ESP32:** Module trung tâm, đại diện cho MQTT client.

    - **LED:** Một đèn LED (màu đỏ) được chọn để mô phỏng thiết bị đầu ra, kết nối qua điện trở.
    - **Điện trở 220Ω:** Nối giữa GPIO4 của ESP32 và cực dương của LED để bảo vệ dòng điện.
    - **DHT22:** Cảm biến nhiệt độ và độ ẩm, đại diện cho thiết bị đầu vào.
- **Kết nối phần cứng:**
    - LED: Cực dương nối qua điện trở 220Ω đến GPIO4, cực âm nối với GND của ESP32.
    - DHT22: Chân VCC nối với 3.3V, chân GND nối với GND, và chân DATA nối với GPIO14 của ESP32.
- **Cấu hình Wi-Fi:** Trong Wokwi, ESP32 được cấu hình để kết nối với mạng Wi-Fi giả lập bằng cách sử dụng SSID và mật khẩu mặc định (do Wokwi cung cấp), cho phép giao tiếp với Mosquitto broker.

4.1.2. Thiết lập kết nối MQTT

- **Broker công cộng:** Mosquitto broker công cộng tại "test.mosquitto.org" (port 1883) được chọn để đơn giản hóa mô phỏng, không yêu cầu cấu hình bảo mật TLS.
- **Kết nối ESP32:** ESP32 được cấu hình để gửi gói tin CONNECT đến broker với client ID (ví dụ: "ESP32_Client") và thời gian "keep-alive" 60 giây. Sau khi nhận gói CONNACK từ Mosquitto, ESP32 subscribe topic "home/controls/led" để nhận lệnh.
- **Kiểm tra kết nối:** Wokwi Serial Monitor hiển thị thông báo như "Connected to MQTT Broker" để xác nhận kết nối thành công.

4.1.3. Cấu hình topic và thông điệp

- **Topic giám sát:**
    - "home/sensors/dht22/temperature" cho dữ liệu nhiệt độ.
    - "home/sensors/dht22/humidity" cho dữ liệu độ ẩm.
- **Topic điều khiển:** "home/controls/led" cho lệnh bật/tắt LED ("ON" hoặc "OFF").

- **Topic trạng thái:** "home/controls/led/status" để phản hồi trạng thái LED (ví dụ: "LED is ON").
- **Thông điệp:** Dữ liệu cảm biến được định dạng thành chuỗi đơn giản (như "25.5"), lệnh và trạng thái dùng "ON"/"OFF" hoặc chuỗi mô tả (như "LED is ON").

4.1.4. Công cụ hỗ trợ

- **MQTT Explorer:** Một ứng dụng bên ngoài (hoặc công cụ tương tự) được sử dụng để giả lập ứng dụng điều khiển, cho phép publish lệnh và subscribe dữ liệu từ các topic.
- **Wokwi Serial Monitor:** Hiển thị trạng thái kết nối, dữ liệu cảm biến, và phản hồi từ ESP32 trong quá trình mô phỏng.

4.2. Các kịch bản thử nghiệm

Để đánh giá hiệu suất và tính khả thi của hệ thống, ba kịch bản thử nghiệm chính được thực hiện trên Wokwi, tập trung vào hai chức năng: điều khiển LED và giám sát dữ liệu cảm biến.

4.2.1. Kịch bản 1: Điều khiển LED từ xa

- **Mục tiêu:** Kiểm tra khả năng bật/tắt LED từ xa qua MQTT.
- **Quy trình:**
    1. ESP32 subscribe topic "home/controls/led".
    2. Từ MQTT Explorer, publish lệnh "ON" đến topic "home/controls/led" với QoS 1.
    3. ESP32 nhận lệnh, đặt GPIO4 ở mức cao để bật LED, và publish "LED is ON" đến "home/controls/led/status".
    4. Lặp lại với lệnh "OFF", ESP32 đặt GPIO4 ở mức thấp và publish "LED is OFF".
- **Kỳ vọng:** LED trên Wokwi thay đổi trạng thái (sáng/tắt) theo lệnh, và trạng thái được phản hồi chính xác qua Serial Monitor hoặc MQTT Explorer.

4.2.2. Kịch bản 2: Giám sát dữ liệu cảm biến DHT22

- **Mục tiêu:** Kiểm tra khả năng thu thập và gửi dữ liệu nhiệt độ/độ ẩm từ DHT22 qua MQTT.
- **Quy trình:**
    1. ESP32 đọc dữ liệu từ DHT22 qua GPIO14 mỗi 10 giây (Wokwi mô phỏng giá trị ngẫu nhiên, ví dụ: 25,5°C và 60%).
    2. ESP32 publish dữ liệu đến "home/sensors/dht22/temperature" (như "25.5") và "home/sensors/dht22/humidity" (như "60").
    3. MQTT Explorer subscribe hai topic này để nhận dữ liệu.
- **Kỳ vọng:** Dữ liệu cảm biến được hiển thị đều đặn trên MQTT Explorer hoặc Serial Monitor, phản ánh chu kỳ 10 giây.

4.2.3. Kịch bản 3: Kết hợp điều khiển và giám sát

- **Mục tiêu:** Đánh giá khả năng xử lý đồng thời của ESP32 khi vừa điều khiển LED vừa giám sát cảm biến.
- **Quy trình:**
    1. ESP32 subscribe "home/controls/led" và publish dữ liệu cảm biến mỗi 10 giây.
    2. Trong khi dữ liệu cảm biến đang được gửi, publish lệnh "ON" từ MQTT Explorer.
    3. ESP32 bật LED và tiếp tục gửi dữ liệu cảm biến mà không gián đoạn.
    4. Lặp lại với lệnh "OFF".
- **Kỳ vọng:** LED bật/tắt đúng lệnh, dữ liệu cảm biến vẫn được gửi liên tục, chứng minh khả năng đa nhiệm của ESP32.

4.3. Kết quả đạt được

4.3.1. Kết quả từ Kịch bản 1

- LED trên Wokwi sáng lên khi nhận lệnh "ON" và tắt khi nhận "OFF".
- Trạng thái "LED is ON" hoặc "LED is OFF" được gửi đến "home/controls/led/status" và hiển thị trên MQTT Explorer trong vòng dưới 1 giây sau lệnh.

- Không có hiện tượng trễ hoặc mất lệnh, nhờ QoS 1 đảm bảo truyền ít nhất một lần.

4.3.2. Kết quả từ Kịch bản 2

- Dữ liệu nhiệt độ (ví dụ: "25.5") và độ ẩm (ví dụ: "60") được gửi đều đặn mỗi 10 giây đến các topic tương ứng.
- MQTT Explorer hiển thị dữ liệu chính xác, với giá trị thay đổi ngẫu nhiên trong phạm vi hợp lý (do Wokwi mô phỏng DHT22).
- Serial Monitor của Wokwi ghi lại mỗi lần publish, xác nhận chu kỳ 10 giây được duy trì ổn định.

4.3.3. Kết quả từ Kịch bản 3

- ESP32 xử lý đồng thời lệnh bật/tắt LED và dữ liệu cảm biến mà không bị gián đoạn.
- Khi gửi "ON", LED sáng ngay lập tức, và dữ liệu cảm biến tiếp tục được publish mỗi 10 giây.
- Tương tự với "OFF", hệ thống duy trì hoạt động mượt mà, chứng minh CPU lõi kép của ESP32 đáp ứng tốt yêu cầu đa nhiệm.

4.4. Đánh giá hiệu suất của hệ thống

Dựa trên kết quả mô phỏng, hiệu suất của hệ thống được đánh giá qua các khía cạnh sau:

4.4.1. Độ tin cậy

- Hệ thống hoạt động ổn định với Mosquitto broker công cộng, không ghi nhận mất kết nối trong suốt quá trình thử nghiệm (khoảng 30 phút).
- Lệnh và dữ liệu được truyền chính xác nhờ QoS 1, phù hợp với ứng dụng cơ bản.

4.4.2. Độ trễ

- Thời gian từ khi publish lệnh "ON"/"OFF" đến khi LED thay đổi trạng thái dưới 1 giây, phụ thuộc vào tốc độ mạng giả lập của Wokwi.

- Dữ liệu cảm biến xuất hiện trên MQTT Explorer gần như tức thời sau mỗi chu kỳ 10 giây, cho thấy độ trễ thấp.

4.4.3. Khả năng đa nhiệm

- ESP32 xử lý tốt cả điều khiển và giám sát cùng lúc, không có hiện tượng xung đột hay trễ đáng kể, nhờ CPU lõi kép và thiết kế topic phân cấp.

4.4.4. Hạn chế

- Mô phỏng trên Wokwi không phản ánh đầy đủ các vấn đề thực tế như gián đoạn mạng hoặc lỗi phần cứng (như cảm biến hỏng).
- Dữ liệu DHT22 là giá trị ngẫu nhiên do Wokwi tạo ra, không phản ánh chính xác điều kiện môi trường thực.

4.4.5. Tổng quan

Hệ thống đạt hiệu suất cao trong môi trường mô phỏng, xác nhận tính khả thi của thiết kế. Tuy nhiên, để áp dụng thực tế, cần thử nghiệm với phần cứng thật và mạng thực để đánh giá thêm độ bền và khả năng xử lý lỗi.

CHƯƠNG 5: KẾT LUẬN

5. Kết luận

Phần này tổng kết các kết quả đạt được từ việc thiết kế và mô phỏng hệ thống điều khiển từ xa qua giao thức MQTT với ESP32 trên Wokwi, đồng thời phân tích các ứng dụng thực tế, khả năng mở rộng, hạn chế của hệ thống, và đề xuất hướng cải thiện trong tương lai. Hệ thống đã minh họa rõ ràng khả năng của MQTT trong việc điều khiển thiết bị và giám sát môi trường, mở ra tiềm năng ứng dụng trong nhiều lĩnh vực.

5.1. Ưu điểm và khả năng mở rộng của mô hình đã thiết kế

Hệ thống điều khiển từ xa được thiết kế và mô phỏng trong tiểu luận này thể hiện nhiều ưu điểm nổi bật, đồng thời sở hữu tiềm năng mở rộng đáng kể nhờ cấu trúc linh hoạt và sử dụng các công nghệ IoT hiện đại.

5.1.1. Ưu điểm

- **Hiệu quả và nhẹ:** Giao thức MQTT với mô hình publish/subscribe cho phép truyền dữ liệu nhanh chóng và tiết kiệm băng thông, phù hợp với các thiết bị hạn chế tài nguyên như ESP32. Kết quả mô phỏng trong Chương 4 cho thấy độ trễ dưới 1 giây và khả năng xử lý đa nhiệm tốt.
- **Chi phí thấp:** Hệ thống sử dụng các thành phần phần cứng giá rẻ như ESP32-WROOM-32, LED, điện trở 220Ω, và DHT22, kết hợp với Mosquitto broker mã nguồn mở, giúp giảm chi phí triển khai so với các giải pháp thương mại.
- **Dễ triển khai:** Thiết kế đơn giản với sơ đồ kết nối phần cứng rõ ràng (Chương 3, Sơ đồ 3.2) và cấu trúc topic phân cấp dễ hiểu, giúp cả người mới bắt đầu và chuyên gia đều có thể áp dụng. Mô phỏng trên Wokwi cũng chứng minh tính dễ tiếp cận của hệ thống.
- **Tính linh hoạt:** Mô hình hoạt động (Chương 3, mục 3.2) cho phép hệ thống hoạt động ổn định trong môi trường mô phỏng, với khả năng xử lý đồng thời điều khiển LED và giám sát dữ liệu cảm biến, phản ánh tiềm năng ứng dụng đa dạng.

5.1.2. Khả năng mở rộng

- **Thêm thiết bị:** Cấu trúc topic phân cấp (như "home/sensors/" và "home/controls/") hỗ trợ tích hợp thêm các thiết bị mới mà không cần thay đổi hệ thống hiện tại. Ví dụ, có thể thêm quạt ("home/controls/fan") hoặc cảm biến ánh sáng ("home/sensors/light") bằng cách mở rộng ESP32 hoặc dùng nhiều module.
- **Tăng quy mô không gian:** Topic "home" có thể được mở rộng thành "kitchen", "bedroom", hoặc "office", cho phép quản lý nhiều khu vực trong một hệ thống lớn hơn như tòa nhà thông minh.
- **Tăng cường chức năng:** Hệ thống có thể tích hợp các tính năng nâng cao như tự động hóa (bật LED khi nhiệt độ vượt ngưỡng) hoặc lưu trữ dữ liệu dài hạn trên Mosquitto (retained message), nhờ thiết kế linh hoạt của MQTT.

5.2. Các ứng dụng thực tiễn trong nhà thông minh và IoT nói chung

Mô hình điều khiển từ xa này không chỉ là một bài tập lý thuyết mà còn có giá trị thực tiễn cao trong các lĩnh vực nhà thông minh và IoT rộng lớn hơn, dựa trên khả năng giám sát và điều khiển từ xa đã được kiểm chứng.

5.2.1. Ứng dụng trong nhà thông minh

- **Điều khiển thiết bị gia dụng:** Hệ thống có thể được áp dụng để bật/tắt đèn, quạt, hoặc máy lạnh từ xa thông qua ứng dụng di động hỗ trợ MQTT (như MQTT Dashboard). Kịch bản thử nghiệm bật/tắt LED (Chương 4, 4.2.1) minh họa rõ khả năng này.
- **Giám sát môi trường:** Dữ liệu nhiệt độ và độ ẩm từ DHT22 có thể được sử dụng để theo dõi điều kiện trong nhà, chẳng hạn như phòng khách hoặc nhà kính. Kết quả từ kịch bản giám sát cho thấy hệ thống cung cấp dữ liệu đáng tin cậy mỗi 10 giây.
- **Tích hợp hệ sinh thái:** Hệ thống có thể kết nối với các nền tảng nhà thông minh như Home Assistant hoặc Google Home thông qua Mosquitto, mở rộng khả năng điều khiển bằng giọng nói hoặc tự động hóa.

5.2.2. Ứng dụng trong IoT nói chung

- **Nông nghiệp thông minh:** Dữ liệu từ DHT22 có thể được dùng để giám sát nhiệt độ và độ ẩm trong nhà kính hoặc trang trại, kết hợp với điều khiển hệ thống tưới nước qua ESP32.
- **Công nghiệp:** Hệ thống có thể được triển khai để theo dõi môi trường trong kho hàng (nhiệt độ, độ ẩm) và điều khiển các thiết bị như đèn báo hoặc quạt thông gió, tận dụng khả năng mở rộng topic.
- **Y tế:** Ứng dụng giám sát nhiệt độ phòng bệnh hoặc điều khiển thiết bị y tế từ xa, với yêu cầu độ trễ thấp đã được xác nhận trong mô phỏng.

5.3. Đề xuất các hướng phát triển trong tương lai

Dựa trên kết quả thiết kế và mô phỏng, hệ thống này có thể được phát triển thêm để tăng cường tính thực tiễn và ứng dụng trong các kịch bản phức tạp hơn. Dưới đây là các hướng phát triển được đề xuất:

5.3.1. Triển khai trên phần cứng thực tế

- **Thử nghiệm thực địa:** Chuyển từ mô phỏng Wokwi sang phần cứng thật (ESP32, LED, DHT22) để kiểm tra hiệu suất trong điều kiện thực tế, bao gồm gián đoạn mạng và lỗi phần cứng.
- **Nguồn năng lượng:** Thêm pin hoặc nguồn điện ổn định cho ESP32, đảm bảo hệ thống hoạt động liên tục trong các ứng dụng không dây.

5.3.2. Tích hợp bảo mật và cảm biến đa dạng

- **Bảo mật:** Triển khai TLS/SSL trên Mosquitto (port 8883) và thêm xác thực username/password để bảo vệ dữ liệu và lệnh, đặc biệt khi dùng broker đám mây.
- **Cảm biến đa dạng:** Thay thế hoặc bổ sung DHT22 bằng các cảm biến khác như cảm biến ánh sáng (LDR), cảm biến chuyển động (PIR), hoặc cảm biến khí (MQ-2), mở rộng khả năng giám sát môi trường.

5.3.3. Phát triển giao diện người dùng

- **Ứng dụng di động:** Xây dựng một ứng dụng di động chuyên dụng (dùng Flutter hoặc React Native) thay vì MQTT Explorer, với giao diện trực quan để điều khiển và hiển thị dữ liệu theo thời gian thực.
- **Tích hợp AI:** Thêm thuật toán học máy để phân tích dữ liệu cảm biến (như dự đoán nhiệt độ) hoặc tự động hóa (bật quạt khi độ ẩm cao), tăng tính thông minh của hệ thống.

5.3.4. Tối ưu hóa hiệu suất

- **Giảm độ trễ:** Tối ưu chu kỳ gửi dữ liệu (từ 10 giây xuống 5 giây) hoặc dùng QoS 0 cho các ứng dụng ít quan trọng để tăng tốc độ phản hồi.
- **Lưu trữ dữ liệu:** Kết nối Mosquitto với cơ sở dữ liệu (như MySQL) để lưu lịch sử dữ liệu cảm biến, phục vụ phân tích dài hạn.

TÀI LIỆU THAM KHẢO

[1] Bai 6 ESP với MQTT - Wokwi ESP32, STM32, Arduino Simulator. (n.d.). https://wokwi.com/projects/394043592527848449

[2] Nga Q. (2023, November 28). Hướng dẫn MQTT ESP32 cho người mới bắt đầu học điện tử - IoT Zone. IoT Zone. https://www.iotzone.vn/esp32/esp32-co-ban/mqtt-esp32-cho-nguoi-moi/