BỘ CÔNG THƯƠNG

**TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

---------------------------

A blue and white logo with a book and a square cap

Description automatically generated

**TIỂU LUẬN MÔN HỌC INTERNET OF THINGS**

**ĐỀ TÀI: ĐIỀU KHIỂN HỆ THỐNG GIÁM SÁT NHIỆT ĐỘ VÀ ĐỘ ẨM TRÊN ESP32 TỪ ĐIỆN THOẠI QUA MQTT**

**GVHD: Võ Hoàng Hải**

**Sinh viên thực hiện: 2033222432\_Lê Quang Long**

**Mã lớp:**

*TP. Hồ Chí Minh tháng 10 năm 2025*

BỘ CÔNG THƯƠNG

**TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

---------------------------

A blue and white logo with a book and a square cap

Description automatically generated

**TIỂU LUẬN MÔN HỌC INTERNET OF THINGS**

**ĐỀ TÀI: ĐIỀU KHIỂN HỆ THỐNG GIÁM SÁT NHIỆT ĐỘ VÀ ĐỘ ẨM TRÊN ESP32 TỪ ĐIỆN THOẠI QUA MQTT**

**GVHD: Võ Hoàng Hải**

**Sinh viên thực hiện: 2033222432\_Lê Quang Long**

**Mã lớp: 13DHBM03**

*TP. Hồ Chí Minh tháng 10 năm 2025*

**LỜI CAM ĐOAN**

Em xin cam đoan toàn bộ nội dung trong báo cáo “Điều khiển hệ thống giám sát nhiệt độ và độ ẩm trên ESP32 từ điện thoại qua MQTT” là kết quả học tập, nghiên cứu và thực nghiệm của bản thân, dưới sự hướng dẫn của giảng viên. Các nguồn tài liệu tham khảo đều được trích dẫn rõ ràng. Nếu có sai sót, em xin hoàn toàn chịu trách nhiệm.

Sinh viên thực hiện tiểu luận

Lê Quang Long

**LỜI CÁM ƠN**

Để hoàn thành đề tài "Điều khiển hệ thống giám sát nhiệt độ và dộ ẩm trên ESP32 từ điện thoại qua MQTT", không thể thiếu sự hỗ trợ và đồng hành từ nhiều phía. Trước tiên, xin gửi lời cảm ơn chân thành đến giảng viên Võ Hoàng Hải, người đã tận tình hướng dẫn, cung cấp những kiến thức chuyên sâu và góp ý quý báu trong suốt quá trình thực hiện. Sự định hướng và hỗ trợ kỹ thuật từ thầy đã giúp đề tài được xây dựng một cách logic, chuyên nghiệp và đáp ứng đầy đủ yêu cầu của môn học Internet of Things.

Bên cạnh đó, xin bày tỏ lòng biết ơn đến các nguồn tài liệu tham khảo uy tín như RandomNerdTutorials, HiveMQ, Instructables, và cộng đồng IoT trực tuyến. Những hướng dẫn chi tiết, code mẫu, và tài liệu kỹ thuật từ các nguồn này đã tạo điều kiện thuận lợi để nghiên cứu, triển khai và hoàn thiện hệ thống một cách hiệu quả.

Cuối cùng, cảm ơn bản thân vì đã nỗ lực không ngừng, từ việc lập kế hoạch, thiết kế mô phỏng trên Wokwi, lập trình, đến chuẩn bị demo và báo cáo. Những giờ làm việc miệt mài, dù đôi lúc gặp khó khăn, đã mang lại những bài học quý giá về quản lý dự án và ứng dụng công nghệ IoT. Hy vọng đề tài này không chỉ là một sản phẩm học tập mà còn là bước đệm cho những dự án sáng tạo hơn trong tương lai.

**NỘI DUNG TÌM HIỂU**

[CHƯƠNG 1. THÔNG TIN CHUNG 8](#_Toc211680895)

[1.1. Tên đề tài 8](#_Toc211680896)

[1.2. Thời gian thực hiện 9](#_Toc211680897)

[1.3. Số thành viên nhóm 10](#_Toc211680898)

[1.4. Giảng viên hướng dẫn 10](#_Toc211680899)

[1.5. Tài liệu tham khảo ban đầu 10](#_Toc211680900)

[CHƯƠNG 2. MỤC TIÊU ĐỀ TÀI 12](#_Toc211680901)

[2.1. Mục tiêu tổng quát 12](#_Toc211680902)

[2.2. Mục tiêu cụ thể 12](#_Toc211680903)

[2.2.1. Xác định và mô tả bài toán ứng dụng IoT thực tế 12](#_Toc211680904)

[2.2.2. Thiết kế kiến trúc tổng quan và chi tiết hệ thống IoT 13](#_Toc211680905)

[2.2.3. Lựa chọn phần mềm và công nghệ IoT phù hợp 14](#_Toc211680906)

[2.2.4. Lập trình và triển khai hệ thống để gửi/nhận lệnh qua MQTT 15](#_Toc211680907)

[2.2.5. Thuyết trình và báo cáo kết quả 15](#_Toc211680908)

[2.3. Tiêu chí đánh giá thành công 16](#_Toc211680909)

[CHƯƠNG 3. NỘI DUNG THỰC HIỆN 17](#_Toc211680910)

[3.1. Xác định bài toán ứng dụng IoT 17](#_Toc211680911)

[3.1.1. Mô tả bài toán 17](#_Toc211680912)

[3.1.2. Yêu cầu hệ thống 17](#_Toc211680913)

[3.1.3. Ý nghĩa thực tế và lợi ích 19](#_Toc211680914)

[3.1.4. Giới hạn bài toán 19](#_Toc211680915)

[3.2. Phân tích và thiết kế hệ thống IoT 19](#_Toc211680916)

[3.2.1. Kiến trúc tổng quan 19](#_Toc211680917)

[3.2.2. Thiết kế chi tiết thành phần 22](#_Toc211680918)

[3.2.3. Phân tích rủi ro và giải pháp 24](#_Toc211680919)

[3.3. Xây dựng hệ thống 24](#_Toc211680920)

[3.3.1. Chuẩn bị môi trường mô phỏng 25](#_Toc211680921)

[3.3.2. Lập trình firmware cho ESP32 25](#_Toc211680922)

[3.3.3. Cấu hình MQTT Broker và app điện thoại 26](#_Toc211680923)

[3.3.4. Tích hợp và test hệ thống 26](#_Toc211680924)

[3.4. Thuyết trình và báo cáo 27](#_Toc211680925)

[3.4.1. Demo hệ thống 27](#_Toc211680926)

[3.4.2. Viết báo cáo và slide 45](#_Toc211680927)

[CHƯƠNG 4. PHÂN CÔNG CÔNG VIỆC (VAI TRÒ) 46](#_Toc211680928)

[4.1. Bảng phân công chi tiết 46](#_Toc211680929)

[4.2. Trách nhiệm chung 51](#_Toc211680930)

[CHƯƠNG 5. TIẾN ĐỘ (2 TUẦN – 20 GIỜ) 52](#_Toc211680931)

[5.1. Bảng tiến độ chi tiết 52](#_Toc211680932)

[5.2. Theo dõi và điều chỉnh tiến độ 57](#_Toc211680933)

[CHƯƠNG 6: KẾT QUẢ DỰ KIẾN 58](#_Toc211680934)

[6.1. Kết quả hệ thống 58](#_Toc211680935)

[6.2. Kết quả báo cáo và thuyết trình 59](#_Toc211680936)

[6.3. Đánh giá và cải tiến 60](#_Toc211680937)

[KẾT LUẬN 62](#_Toc211680938)

**DANH MỤC HÌNH ẢNH**

[Hình 3. 1. Mô hình được thiết lập ở VS Code 43](#_Toc211680939)

[Hình 3. 2. Thiết lập điều khiển từ trình duyệt HiveMQ 44](#_Toc211680940)

[Hình 3. 3. Thiết lập trình biên soạn 44](#_Toc211680941)

[Hình 3. 4. Kết nối thành công 45](#_Toc211680942)

# 

# CHƯƠNG 1. THÔNG TIN CHUNG

## 1.1. Tên đề tài

Đề tài đồ án của em là một ứng dụng thực tế trong lĩnh vực **Internet of Things (IoT)**, tập trung vào việc xây dựng một **hệ thống giám sát môi trường thông minh**, có khả năng thu thập và truyền dữ liệu **nhiệt độ – độ ẩm** theo thời gian thực lên nền tảng đám mây.

Đề tài đầy đủ: Hệ thống giám sát nhiệt độ & độ ẩm sử dụng ESP32, cảm biến DHT22 và giao thức MQTT (HiveMQ).

Đề tài nhấn mạnh vào việc sử dụng giao thức MQTT – một giao thức truyền tin nhẹ và phổ biến trong IoT – để truyền dữ liệu cảm biến từ thiết bị nhúng ESP32 lên broker đám mây HiveMQ, giúp người dùng có thể theo dõi giá trị nhiệt độ và độ ẩm trực tiếp trên ứng dụng điện thoại hoặc web client.

Mô tả ngắn gọn: Hệ thống IoT cơ bản cho phép giám sát dữ liệu nhiệt độ và độ ẩm theo thời gian thực, sử dụng cảm biến DHT22 làm thiết bị đầu vào, ESP32 làm bộ điều khiển trung tâm, và HiveMQ Broker để truyền dữ liệu qua giao thức MQTT. Dữ liệu thu được có thể hiển thị trên ứng dụng di động (như MQTTool) hoặc trang web HiveMQ Client, giúp người dùng theo dõi điều kiện môi trường mọi lúc, mọi nơi.

Ý tưởng cốt lõi: Xây dựng một mô hình IoT giám sát môi trường, minh họa cách thiết bị cảm biến có thể gửi dữ liệu tự động lên cloud thông qua mạng Internet. Cụ thể, ESP32 sẽ đọc giá trị nhiệt độ và độ ẩm từ cảm biến DHT22, sau đó publish dữ liệu đến các topic MQTT như “/dht22/temperature” và “/dht22/humidity” trên HiveMQ Broker. Người dùng chỉ cần subscribe các topic này để nhận dữ liệu cập nhật liên tục. Mô hình này không chỉ giúp hiểu rõ nguyên lý truyền dữ liệu IoT mà còn có thể mở rộng thành các hệ thống cảnh báo môi trường (ví dụ cảnh báo nhiệt độ cao, độ ẩm thấp).

Công cụ chính:

* **Phần cứng mô phỏng:** ESP32 DevKit V1, cảm biến DHT22.
* **Phần mềm:** PlatformIO (VS Code), thư viện PubSubClient và DHT.h.
* **Giao thức & nền tảng:** MQTT (HiveMQ Cloud), ứng dụng MQTTool hoặc HiveMQ Web Client để hiển thị dữ liệu.

Phạm vi:

* **Đọc và giám sát hai thông số: nhiệt độ (°C) và độ ẩm (%).**
* **Cập nhật dữ liệu liên tục theo chu kỳ 5 giây.**
* **Gửi dữ liệu lên MQTT Broker HiveMQ và hiển thị trên client.**
* **Tự động reconnect khi mất kết nối, đảm bảo tính ổn định của hệ thống.**

## 1.2. Thời gian thực hiện

Đề tài được lập kế hoạch thực hiện trong khoảng thời gian ngắn gọn, tập trung vào hiệu quả và tính khả thi, nhằm đáp ứng yêu cầu của môn học Internet of Things.

Thời lượng tổng: 2 tuần (tổng 20 giờ, trung bình 10 giờ/tuần).  
Thời gian được phân bổ hợp lý giữa các giai đoạn nghiên cứu – thiết kế – lập trình – kiểm thử – báo cáo, đảm bảo đề tài hoàn thiện đúng tiến độ và đạt chất lượng tốt.

Lịch trình chi tiết:

* **Tuần 1 (01/10/2025 – 07/10/2025):**
* Tìm hiểu lý thuyết về MQTT và HiveMQ.
* Nghiên cứu cách kết nối cảm biến DHT22 với ESP32.
* Thiết kế sơ đồ mạch và tạo project trên PlatformIO (VS Code).
* Kiểm thử việc đọc dữ liệu cảm biến và hiển thị qua Serial Monitor.
* **Tuần 2 (08/10/2025 – 14/10/2025):**
* Lập trình hoàn chỉnh quá trình publish dữ liệu lên HiveMQ.
* Kiểm thử hiển thị dữ liệu trên ứng dụng MQTTool hoặc HiveMQ Web Client.
* Xử lý lỗi kết nối, tối ưu mã nguồn, và hoàn thiện báo cáo cùng slide thuyết trình.

Lý do thời gian:

* **Đề tài có phạm vi vừa phải, chỉ sử dụng các công cụ miễn phí và phần cứng sẵn có, phù hợp với thời lượng 2 tuần. Ngoài ra, việc lập trình và thử nghiệm trên PlatformIO giúp tiết kiệm thời gian cài đặt và quản lý thư viện, đảm bảo tiến độ nhanh chóng. Nếu có sự cố (như lỗi kết nối mạng hoặc dữ liệu sai lệch), em sẽ ưu tiên hoàn thiện các chức năng chính trước để đảm bảo hệ thống demo hoạt động ổn định, đáp ứng yêu cầu môn học.**

## 1.3. Số thành viên nhóm

Đề tài này được thực hiện cá nhân, giúp em tự chủ trong việc học hỏi và áp dụng kiến thức, đồng thời rèn luyện kỹ năng làm việc độc lập – một yếu tố quan trọng trong lĩnh vực công nghệ.

* **Tổng số: 1 thành viên.**
* **Là đồ án cá nhân, em sẽ chịu trách nhiệm toàn bộ từ nghiên cứu đến triển khai, đảm bảo mọi phần đều được chăm chút kỹ lưỡng để đạt kết quả tốt nhất.**
* **Danh sách chi tiết:**
* SV1: 2033222432, Lê Quang Long, 13DHBM03, 0392736079/Lequanglong6504@gmail.com.

## 1.4. Giảng viên hướng dẫn

* Tên giảng viên: Võ Hoàng Hải. Thầy Hải là chuyên gia trong lĩnh vực IoT và hệ thống nhúng, với kinh nghiệm giảng dạy và nghiên cứu phong phú, giúp tôi nhận được lời khuyên quý báu.
* Vai trò: Hướng dẫn lý thuyết, hỗ trợ kỹ thuật, đánh giá tiến độ và kết quả cuối cùng. Cụ thể, thầy sẽ hướng dẫn về các khái niệm cốt lõi như giao thức MQTT, thiết kế hệ thống IoT, và cách debug lỗi phổ biến. Vai trò này không chỉ giúp khắc phục khó khăn mà còn khuyến khích em mở rộng ý tưởng, như thêm tính năng bảo mật cơ bản cho MQTT.
* Thông tin liên hệ: [Email/Số điện thoại giảng viên]

## 1.5. Tài liệu tham khảo ban đầu

Tài liệu chính:

* RandomNerdTutorials – “ESP32 MQTT Publish DHT22 Sensor Data to HiveMQ”  
  → Cung cấp hướng dẫn chi tiết về cách đọc dữ liệu cảm biến DHT22 bằng ESP32 và gửi qua giao thức MQTT.
* HiveMQ Documentation  
  → Giải thích cách tạo và cấu hình topic, publish/subscribe và kiểm thử trên web client.
* PlatformIO Documentation (Visual Studio Code)  
  → Hướng dẫn cấu hình môi trường phát triển, cài thư viện và debug trực tiếp cho ESP32.
* Instructables  
  → Một số hướng dẫn DIY về đo và hiển thị dữ liệu DHT22 bằng MQTT.

Lý do lựa chọn:

* Các tài liệu trên đều miễn phí, chi tiết và có minh họa code cụ thể, dễ dàng chỉnh sửa để phù hợp với hệ thống giám sát. Ngoài ra, việc tham khảo các nguồn này giúp đảm bảo tính chính xác, độ tin cậy và hiệu quả khi lập trình, đồng thời dễ dàng tích hợp với công cụ PlatformIO trong Visual Studio Code.

# CHƯƠNG 2. MỤC TIÊU ĐỀ TÀI

## 2.1. Mục tiêu tổng quát

Mục tiêu tổng quát của đề tài tập trung vào việc xây dựng một hệ thống IoT đơn giản nhưng hiệu quả, có khả năng giám sát nhiệt độ và độ ẩm môi trường theo thời gian thực, đồng thời làm rõ nguyên lý kết nối, truyền dữ liệu và hiển thị thông tin qua đám mây (cloud) trong công nghệ Internet of Things (IoT).

Xây dựng và triển khai hệ thống IoT thực tế để minh họa cách thiết bị cảm biến có thể thu thập dữ liệu và gửi lên cloud qua giao thức MQTT.  
→ Hệ thống sử dụng ESP32 kết hợp với cảm biến DHT22 để đo hai giá trị quan trọng là nhiệt độ (°C) và độ ẩm (%), sau đó publish dữ liệu lên MQTT Broker HiveMQ. Người dùng có thể subscribe các topic tương ứng để theo dõi dữ liệu trực tuyến trên ứng dụng điện thoại hoặc HiveMQ Web Client.  
→ Đề tài minh họa rõ ràng cách mà IoT giúp biến các cảm biến vật lý thành thiết bị thông minh kết nối Internet, góp phần hiểu sâu hơn về cách thu thập, truyền và hiển thị dữ liệu môi trường trong các hệ thống thực tế như giám sát nhà kính, phòng học, hoặc kho bảo quản hàng hóa.

Áp dụng kiến thức môn học IoT để thiết kế, lập trình và demo mô hình giám sát nhiệt độ & độ ẩm trên ESP32, chứng minh tính khả thi và khả năng mở rộng.  
→ Đề tài được xây dựng từ kiến thức nền tảng của môn học IoT, kết hợp giữa phần cứng, phần mềm và nền tảng đám mây. Hệ thống sẽ được demo để chứng minh tính ổn định và thời gian phản hồi nhanh (dữ liệu cập nhật mỗi 5 giây).  
→ Ngoài ra, đề tài hướng đến khả năng mở rộng, chẳng hạn thêm ngưỡng cảnh báo khi nhiệt độ vượt mức hoặc tích hợp dashboard hiển thị đồ thị dữ liệu, thể hiện sự sáng tạo và tiềm năng phát triển ứng dụng thực tế.

## 2.2. Mục tiêu cụ thể

### **2.2.1. Xác định và mô tả bài toán ứng dụng IoT thực tế**

Mô tả vấn đề:

* Trong cuộc sống hiện đại, việc giám sát các yếu tố môi trường như nhiệt độ và độ ẩm là vô cùng quan trọng – đặc biệt trong các không gian như phòng máy chủ, nhà kính, kho hàng hay phòng học. Tuy nhiên, nếu chỉ giám sát thủ công hoặc bằng thiết bị cục bộ, người dùng sẽ không thể nắm bắt dữ liệu khi vắng mặt.  
  → Đề tài này giải quyết vấn đề đó bằng cách xây dựng một hệ thống IoT kết nối Internet, giúp tự động đo và gửi dữ liệu cảm biến lên cloud, cho phép người dùng theo dõi mọi lúc, mọi nơi thông qua ứng dụng MQTT.

Phạm vi:

* Hệ thống tập trung vào việc đo hai chỉ số cơ bản là nhiệt độ và độ ẩm bằng cảm biến DHT22, và truyền dữ liệu lên MQTT Broker HiveMQ qua mạng WiFi.  
  Phạm vi được giới hạn trong việc thu thập – gửi – hiển thị dữ liệu, chưa bao gồm điều khiển phản hồi, nhằm đảm bảo tính khả thi trong thời gian ngắn.  
  Nếu mở rộng, hệ thống có thể thêm ngưỡng cảnh báo, tự động điều khiển quạt hoặc máy phun sương, tạo thành một mạng IoT giám sát môi trường thông minh hoàn chỉnh.

### **2.2.2. Thiết kế kiến trúc tổng quan và chi tiết hệ thống IoT**

1. **Sơ đồ khối tổng quan:**

| **Lớp IoT** | **Thành phần chính** | **Chức năng** |
| --- | --- | --- |
| **Perception Layer** | Cảm biến DHT22 + ESP32 | Thu thập dữ liệu nhiệt độ & độ ẩm |
| **Network Layer** | WiFi + giao thức MQTT | Kết nối Internet và truyền dữ liệu lên cloud |
| **Application Layer** | HiveMQ Web Client / App IoT MQTT Panel | Điều chỉnh dữ liệu thời gian thực |

→ Sơ đồ khối sẽ được thiết kế bằng Draw.io hoặc PowerPoint, mô tả luồng dữ liệu từ cảm biến → ESP32 → HiveMQ → Ứng dụng hiển thị.  
Điều này giúp hình dung rõ quá trình truyền dữ liệu và kiến trúc kết nối toàn hệ thống.

1. **Thiết kế chi tiết:**

* ESP32 đọc dữ liệu từ cảm biến DHT22 mỗi 5 giây.
* Dữ liệu được chuyển đổi sang chuỗi và publish lên các topic:
* Esp32/dht22/temperature – Nhiệt độ
* Esp32/dht22/humidity – Độ ẩm
* HiveMQ Broker tiếp nhận dữ liệu và gửi đến các client subscribe topic tương ứng.
* Người dùng quan sát dữ liệu thời gian thực qua HiveMQ Web Client hoặc ứng dụng di động.  
  → Thiết kế này giúp hệ thống hoạt động ổn định, dễ mở rộng, và minh họa trọn vẹn chu trình cảm biến – kết nối – cloud – hiển thị.

### **2.2.3. Lựa chọn phần mềm và công nghệ IoT phù hợp**

Tiêu chí lựa chọn:

* Chi phí thấp (phần mềm miễn phí).
* Dễ triển khai, giao diện thân thiện.
* Tương thích tốt với ESP32 và có tài liệu hỗ trợ rộng rãi.
* Có thể chạy trực tiếp trong PlatformIO (VS Code) để lập trình, nạp code và debug nhanh chóng.

Các thành phần được chọn:

* Phần cứng: ESP32 DevKit V1, cảm biến DHT22.
* Phần mềm: PlatformIO (VS Code), thư viện DHT.h và PubSubClient.
* Cloud: HiveMQ Broker (miễn phí, hỗ trợ kết nối không cần xác thực).
* Ứng dụng hiển thị: HiveMQ Web Client / IoT MQTT Panel (Android, iOS).

Lý do chọn MQTT:  
Giao thức MQTT được chọn vì nhẹ, ổn định và tiết kiệm băng thông, rất phù hợp cho thiết bị IoT.

* Mô hình publish/subscribe giúp ESP32 gửi dữ liệu và nhiều client khác nhau có thể nhận cùng lúc mà không cần kết nối trực tiếp.
* MQTT hỗ trợ QoS (Quality of Service) và Last Will Testament (LWT) để đảm bảo độ tin cậy trong quá trình truyền dữ liệu.
* So với HTTP, MQTT nhanh và linh hoạt hơn, giúp giảm độ trễ khi gửi dữ liệu cảm biến liên tục.

### **2.2.4. Lập trình và triển khai hệ thống để gửi/nhận lệnh qua MQTT**

Triển khai:

* ESP32 được lập trình bằng PlatformIO (VS Code).
* Chương trình thực hiện các bước:
* Kết nối WiFi.
* Kết nối đến MQTT Broker (broker.hivemq.com, port 1883).
* Đọc dữ liệu từ DHT22.
* Publish giá trị nhiệt độ và độ ẩm lên các topic định sẵn.
* Hiển thị log qua Serial Monitor để theo dõi.

Kiểm thử:

* Theo dõi dữ liệu gửi lên HiveMQ Web Client.
* Đảm bảo dữ liệu cập nhật định kỳ (5 giây/lần).
* Kiểm tra reconnect tự động khi mất WiFi → Hệ thống đạt yêu cầu nếu dữ liệu hiển thị đúng, ổn định và độ trễ dưới 3 giây.

### **2.2.5. Thuyết trình và báo cáo kết quả**

Demo:

* Video demo hiển thị nhiệt độ và độ ẩm thay đổi theo thời gian thực.
* Giải thích chi tiết quá trình: cảm biến đọc – ESP32 gửi – MQTT truyền – ứng dụng hiển thị.
* Minh họa bằng sơ đồ và dữ liệu thực tế, giúp khán giả dễ hình dung luồng hoạt động.

Báo cáo:

* Báo cáo hoàn chỉnh bằng Word hoặc PDF, có ảnh mạch kết nối, ảnh giao diện HiveMQ, mã nguồn và bảng kết quả đo mẫu.
* Phần trình bày logic, khoa học, dễ hiểu.
* Có thêm phần “mở rộng đề tài” để thể hiện sáng tạo và khả năng phát triển.

## 2.3. Tiêu chí đánh giá thành công

Để đánh giá hiệu quả của đề tài, các tiêu chí sau được áp dụng, tập trung vào hiệu năng hệ thống, tính ổn định và chất lượng báo cáo.

* **Hệ thống hoạt động đúng 100% trong demo**
* ESP32 đọc chính xác dữ liệu từ DHT22, gửi và hiển thị lên HiveMQ liên tục, kể cả khi WiFi chập chờn.
* Thời gian cập nhật dữ liệu nhanh (dưới 3 giây), không bị gián đoạn.
* Báo cáo đầy đủ, sáng tạo và có mở rộng.
* Báo cáo gồm các phần lý thuyết, thiết kế, kết quả, nhận xét và hướng phát triển.
* Thể hiện sáng tạo qua việc sử dụng công cụ PlatformIO thay cho Arduino IDE, và khả năng tích hợp mở rộng (như cảnh báo tự động).

# CHƯƠNG 3. NỘI DUNG THỰC HIỆN

Chương này trình bày chi tiết quá trình thực hiện đề tài “Hệ thống giám sát nhiệt độ & độ ẩm”, bao gồm từ giai đoạn xác định bài toán, phân tích và thiết kế hệ thống IoT, đến việc xây dựng, lập trình, cấu hình và kiểm thử.

Các phần được trình bày logic, kèm ví dụ minh họa và kết quả kiểm chứng, nhằm đảm bảo hệ thống hoạt động ổn định, chính xác và có tính ứng dụng cao trong thực tế.

## 3.1. Xác định bài toán ứng dụng IoT

### **3.1.1. Mô tả bài toán**

Bài toán chính: Xây dựng hệ thống IoT dùng ESP32 kết hợp cảm biến DHT22 để đo và giám sát nhiệt độ – độ ẩm theo thời gian thực, dữ liệu được gửi lên MQTT Broker (HiveMQ), từ đó người dùng có thể theo dõi trên ứng dụng MQTT client hoặc web dashboard.

Cụ thể, ESP32 sẽ đọc dữ liệu từ cảm biến DHT22 định kỳ (mỗi 5 giây), sau đó publish giá trị nhiệt độ và độ ẩm lên các topic MQTT như "esp32/dht22/temperature" và "esp32/dht22/humidity". Người dùng có thể subscribe các topic này thông qua ứng dụng MQTTool hoặc HiveMQ Web Client để theo dõi thông tin thời gian thực.

Ứng dụng thực tế: Hệ thống có thể được áp dụng vào nhà thông minh, nông nghiệp thông minh hoặc phòng thí nghiệm, nơi cần giám sát môi trường liên tục. Ví dụ, trong nhà kính, dữ liệu có thể dùng để tự động điều khiển quạt hoặc máy phun sương khi nhiệt độ hoặc độ ẩm vượt ngưỡng cho phép. Mô hình phù hợp làm nền tảng cơ bản cho các hệ thống IoT giám sát môi trường thông minh.

### **3.1.2. Yêu cầu hệ thống**

***3.1.2.1. Yêu cầu chức năng***

* ESP32:
* Đọc dữ liệu từ cảm biến DHT22 (GPIO21).
* Publish dữ liệu nhiệt độ và độ ẩm lên MQTT Broker (HiveMQ) theo chu kỳ 5 giây.
* Gửi trạng thái hoạt động (“online/offline”) qua topic "esp32/status".
* MQTT Broker (HiveMQ):
* Trung chuyển dữ liệu giữa ESP32 và người dùng.
* Hỗ trợ retain message và QoS đảm bảo dữ liệu cập nhật chính xác.
* Ứng dụng giám sát:
* Subscribe các topic: "esp32/dht22/temperature", "esp32/dht22/humidity", "esp32/status".
* Hiển thị dữ liệu theo thời gian thực (trên MQTTool hoặc HiveMQ Web Client).

***3.1.2.2. Yêu cầu phi chức năng***

* Độ tin cậy:
* ESP32 phải có cơ chế reconnect tự động nếu mất kết nối Wi-Fi hoặc MQTT, kèm LWT (Last Will and Testament) để báo “offline” khi ngắt đột ngột.
* Bảo mật cơ bản:
* Dữ liệu được truyền qua topic riêng để tránh xung đột, có thể thêm username/password trong broker khi mở rộng.
* Chi phí:
* Sử dụng các công cụ miễn phí như HiveMQ, Wokwi và PlatformIO trong VS Code, giúp tiết kiệm chi phí hoàn toàn.
* Khả năng mở rộng:
* Dễ dàng bổ sung thêm cảm biến khác như ánh sáng, khí gas (MQ-2) hoặc gửi dữ liệu lên web server hoặc Google Sheet.

### **3.1.3. Ý nghĩa thực tế và lợi ích**

* Ý nghĩa:
* Đề tài giúp sinh viên hiểu rõ mô hình IoT cơ bản, từ cảm biến thu thập dữ liệu đến truyền tải và hiển thị trên cloud. Hệ thống thể hiện quy trình chuẩn của IoT: Device → Network → Cloud → Application.
* Lợi ích:
* Ứng dụng trong giám sát môi trường, nhà thông minh, nông nghiệp công nghệ cao.
* Giúp người học nắm vững kỹ năng lập trình ESP32, kết nối MQTT và xử lý dữ liệu cảm biến.
* Có thể mở rộng dễ dàng thành hệ thống tự động điều khiển thiết bị khi đạt ngưỡng nhiệt độ/độ ẩm nhất định.

### **3.1.4. Giới hạn bài toán**

* Hệ thống chỉ đo hai thông số nhiệt độ và độ ẩm, chưa hỗ trợ nhiều cảm biến khác.
* Dữ liệu chỉ được hiển thị dưới dạng text qua MQTT, chưa có biểu đồ trực quan hoặc lưu trữ lịch sử.
* Không có mã hóa SSL cho MQTT, nhưng có thể tích hợp trong các phiên bản sau.

## 3.2. Phân tích và thiết kế hệ thống IoT

Phần này phân tích sâu kiến trúc hệ thống IoT giám sát nhiệt độ & độ ẩm, từ tổng quan đến chi tiết, nhằm đảm bảo hệ thống hoạt động ổn định, chính xác và có khả năng mở rộng. Thiết kế được xây dựng theo mô hình IoT chuẩn gồm 4 lớp (Device – Network – Cloud – Application), giúp đề tài dễ triển khai, kiểm thử và phát triển trong tương lai.

### **3.2.1. Kiến trúc tổng quan**

**Sơ đồ khối**

Sơ đồ khối được vẽ bằng công cụ **Draw.io** hoặc **Lucidchart** (kèm ảnh trong báo cáo).

Hệ thống bao gồm bốn thành phần chính:

* Cảm biến DHT22 – Thu thập dữ liệu nhiệt độ và độ ẩm môi trường.
* ESP32 (Wokwi/PlatformIO) – Thiết bị xử lý trung tâm, đọc dữ liệu cảm biến và gửi lên cloud.
* MQTT Broker (HiveMQ) – Đóng vai trò trung gian đám mây, nhận và phân phối dữ liệu giữa ESP32 và ứng dụng.
* Ứng dụng client (MQTTool / HiveMQ Web Client) – Hiển thị dữ liệu thời gian thực cho người dùng.

Sơ đồ minh họa luồng dữ liệu hai chiều:

DHT22 → ESP32 → MQTT Broker → Ứng dụng hiển thị.

* Khi mất kết nối, ESP32 tự động reconnect và gửi lại dữ liệu mới nhất.

**Luồng dữ liệu**

* Dữ liệu được thu thập và truyền qua nhiều bước tuần tự, đảm bảo tính toàn vẹn và thời gian thực:
* **ESP32** đọc giá trị từ cảm biến DHT22 mỗi 5 giây (nhiệt độ °C, độ ẩm %).
* **ESP32** publish dữ liệu lên các topic:
* esp32/dht22/temperature
* esp32/dht22/humidity
* **MQTT Broker (HiveMQ) lưu tạm giá trị và phân phối đến các client đang subscribe.**
* **Ứng dụng MQTTool / HiveMQ Web Client hiển thị dữ liệu theo thời gian thực.**
* **Khi ESP32 mất kết nối, Broker tự động gửi thông báo LWT ("offline") đến topic esp32/status.**

**Thành phần chính**

* **ESP32 (Subscriber/Publisher): Đọc dữ liệu từ DHT22, xử lý và gửi dữ liệu lên broker, đồng thời nhận phản hồi hoặc trạng thái kết nối.**
* **DHT22 (Cảm biến): Thiết bị đo nhiệt độ và độ ẩm có độ chính xác cao (±0.5°C và ±2% RH). Giao tiếp 1-wire với ESP32 qua chân GPIO21.**
* **MQTT Broker (HiveMQ):  
  Trung gian cloud đảm nhiệm vai trò kết nối, quản lý các topic và chuyển tiếp dữ liệu giữa các thiết bị IoT.**
* **Ứng dụng client: Dùng để giám sát thông tin qua MQTT (như HiveMQ Web Client hoặc MQTTool), có thể hiển thị, lưu trữ hoặc xuất dữ liệu.**

**Luồng dữ liệu chi tiết**

* **DHT22 → Gửi dữ liệu đọc được đến ESP32 qua chân DATA.**
* **ESP32 → Publish dữ liệu lên MQTT Broker với định dạng JSON:**
* **MQTT Broker → Chuyển dữ liệu đến các client đang subscribe topic esp32/dht22/#.**
* **Client → Hiển thị dữ liệu trực quan (ví dụ trên ứng dụng HiveMQ Web Client).**
* **Nếu ESP32 bị ngắt kết nối, Broker gửi “offline” tới topic esp32/status.**

**Các lớp IoT**

Thiết kế được chia thành bốn lớp, đảm bảo hệ thống hoạt động đúng theo mô hình tham chiếu IoT:

| **Lớp** | **Thành phần** | **Chức năng chính** |
| --- | --- | --- |
| **Perception Layer (Lớp cảm nhận)** | Cảm biến DHT22 | Đo lường và thu thập dữ liệu môi trường (nhiệt độ, độ ẩm) |
| **Network Layer (Lớp mạng)** | ESP32 + Wi-Fi “Wokwi-GUEST” | Truyền dữ liệu từ cảm biến lên cloud qua giao thức MQTT |
| **Cloud Layer (Lớp đám mây)** | HiveMQ Broker | Lưu trữ, xử lý và phân phối dữ liệu đến các client |
| **Application Layer (Lớp ứng dụng)** | MQTTool / HiveMQ Web Client | Hiển thị thông tin cho người dùng cuối, có thể mở rộng thành dashboard giám sát |

Mối quan hệ giữa các lớp:

* **Perception ↔ Network: DHT22 gửi dữ liệu tới ESP32 qua giao diện GPIO.**
* **Network ↔ Cloud: ESP32 truyền dữ liệu qua Wi-Fi đến HiveMQ.**
* **Cloud ↔ Application: Người dùng truy cập Broker và nhận dữ liệu giám sát theo thời gian thực.**

### **3.2.2. Thiết kế chi tiết thành phần**

Thiết kế được chia thành mô phỏng, phần mềm và giao thức để đảm bảo tính tương thích cao.

***3.2.2.1. Mô phỏng phần cứng***

* ESP32 trên PlatformIO/Wokwi: Mô phỏng module ESP32-WROOM-32, hỗ trợ Wi-Fi tích hợp, dễ lập trình và quan sát dữ liệu trực tiếp trên Serial Monitor.
* Cảm biến DHT22:
* **Kết nối:**
* **VCC → 3.3V**
* **DATA → GPIO21**
* **GND → GND**
* **Giao tiếp 1-wire, tốc độ phản hồi nhanh (dưới 2 giây).**
* Sơ đồ mạch:
* **Vẽ bằng Fritzing hoặc Draw.io thể hiện rõ kết nối giữa ESP32 và DHT22.  
  Sơ đồ đơn giản, an toàn và tương thích với môi trường mô phỏng Wokwi.**

***3.2.2.2. Phần mềm và cloud***

* Môi trường lập trình:
* **PlatformIO tích hợp trong Visual Studio Code để biên dịch và nạp chương trình cho ESP32.**
* **Ưu điểm: dễ quản lý thư viện, log debug tốt, hỗ trợ mô phỏng hoặc nạp firmware thực tế.**
* Thư viện sử dụng:
* **WiFi.h: Kết nối mạng Wi-Fi.**
* **PubSubClient.h: Giao tiếp MQTT.**
* **DHT.h: Đọc dữ liệu từ cảm biến DHT22.**
* MQTT Broker:
* **Dùng HiveMQ Public Broker (broker.hivemq.com, port 1883).**
* **Không cần cài đặt server, hỗ trợ hàng nghìn kết nối miễn phí.**
* Ứng dụng hiển thị:
* **HiveMQ Web Client: Hiển thị dữ liệu trực tuyến qua trình duyệt.**
* **MQTTool (iOS/Android): Giúp subscribe và theo dõi giá trị cảm biến.**

***3.2.2.3. Giao thức và công nghệ***

* Giao thức chính: MQTT v3.1.1 (mô hình publish/subscribe).
* **Gọn nhẹ, tối ưu cho hệ thống IoT có băng thông thấp và độ trễ nhỏ.**
* Kết nối:
* **Wi-Fi: “Wokwi-GUEST” (không mật khẩu).**
* **MQTT port: 1883 (không SSL để đơn giản hoá mô phỏng).**
* QoS (Quality of Service):
* **Sử dụng QoS = 0 (chuyển nhanh, không xác nhận) do yêu cầu thời gian thực.**
* **Tần suất gửi dữ liệu: Cảm biến gửi dữ liệu mỗi 5 giây để giảm tải cho broker và đảm bảo phản hồi nhanh.**

### **3.2.3. Phân tích rủi ro và giải pháp**

* Rủi ro 1: Mất kết nối WiFi/MQTT → Giải pháp: Thêm hàm reconnect() trong loop() của code ESP32, retry mỗi 1 giây. Đảm bảo hệ thống tự phục hồi nhanh chóng.
* Rủi ro 2: Lỗi mô phỏng (LED không sáng) → Giải pháp: Test riêng trên Wokwi trước khi tích hợp, kiểm tra selfTestBlink(). Kiểm tra đấu nối và logic Active-HIGH.
* Rủi ro 3: Broker quá tải (miễn phí) → Giải pháp: Chọn HiveMQ có hạn mức cao, hoặc fallback sang broker khác. Giám sát và chuyển đổi nếu cần.
* Rủi ro 4: Bảo mật (lệnh bị chặn) → Giải pháp: Thêm authentication cho MQTT nếu cần nâng cao. Bắt đầu cơ bản, mở rộng để tăng an toàn.

## 3.3. Xây dựng hệ thống

Phần này mô tả chi tiết các bước thực tế triển khai hệ thống giám sát nhiệt độ và độ ẩm, từ giai đoạn chuẩn bị môi trường, lập trình firmware cho ESP32, cấu hình MQTT Broker đến bước tích hợp và kiểm thử toàn bộ hệ thống. Các bước được trình bày rõ ràng, giúp quá trình thực hiện trên môi trường PlatformIO (VS Code) diễn ra thuận lợi và có thể tái hiện dễ dàng.

### **3.3.1. Chuẩn bị môi trường mô phỏng**

* Tạo project:
* Mở Visual Studio Code, cài đặt extension PlatformIO IDE, sau đó tạo một project mới với board ESP32 Dev Module.
* Trong phần mô phỏng, bạn có thể sử dụng Wokwi tích hợp trong PlatformIO để kiểm tra kết nối giữa ESP32 và DHT22.
* Thêm cảm biến:
* Kết nối DHT22 với ESP32 như sau:
* VCC → 3.3V
* DATA → GPIO21
* GND → GND
* DHT22 được sử dụng để thu thập nhiệt độ và độ ẩm trong môi trường mô phỏng.
* WiFi ảo:
* Sử dụng mạng SSID “Wokwi-GUEST”, không mật khẩu.
* Đây là mạng mô phỏng sẵn trong Wokwi, đảm bảo kết nối ổn định và không cần cấu hình router thật.
* Test cơ bản: Trước khi lập trình MQTT, thực hiện test đơn giản để đọc giá trị cảm biến bằng đoạn code mẫu

### **3.3.2. Lập trình firmware cho ESP32**

* Cài đặt môi trường: Tải Arduino IDE, thêm board ESP32 qua Board Manager, cài thư viện PubSubClient qua Library Manager. Cài đặt nhanh chóng, miễn phí.
* Code cấu trúc: #include <WiFi.h>, #include <PubSubClient.h>; Setup: Kết nối WiFi, MQTT; Loop: Check connect, subscribe callback. Cấu trúc rõ ràng, dễ bảo trì.
* Callback function: void onMqttMessage(char\* topic, byte\* payload, unsigned int length) { parse payload thành lệnh, setLed() cho từng LED, publishStatus() }. Hàm xử lý lệnh từ hai topic.

### **3.3.3. Cấu hình MQTT Broker và app điện thoại**

* Broker:
* Đăng ký tài khoản miễn phí tại HiveMQ
* Sử dụng server "broker.hivemq.com" với port 1883 cho ESP32.
* Với ứng dụng trên điện thoại, có thể dùng port 8883 (TLS) để tăng bảo mật.
* Ứng dụng giám sát:
* Cài đặt IoT MQTT Panel (iOS/Android) hoặc dùng HiveMQ Web Client để hiển thị dữ liệu.
* Khi ESP32 hoạt động, giá trị nhiệt độ và độ ẩm sẽ hiển thị trực tiếp theo thời gian thực.
* Test riêng:
* Dùng HiveMQ Web Client để thử publish dữ liệu test (ví dụ: "temp":25.0, "hum":65.0") nhằm kiểm tra khả năng hiển thị trên ứng dụng.

### **3.3.4. Tích hợp và test hệ thống**

* Test end-to-end:
* Bước 1: Khởi động ESP32 → kết nối Wi-Fi và Broker HiveMQ.
* Bước 2: Mở HiveMQ Web Client → subscribe vào các topic.
* Bước 3: Quan sát dữ liệu được gửi định kỳ từ ESP32 (mỗi 5 giây) hiển thị đúng định dạng.
* Kết quả mong đợi:
* Khi cảm biến đọc nhiệt độ 29.2°C và độ ẩm 73.1%, trên client sẽ hiển thị tương ứng
* Mở rộng test:
* Ngắt Wi-Fi hoặc dừng ESP32 để kiểm tra LWT → Broker tự động publish "offline" lên esp32/status.
* Khởi động lại thiết bị, xác nhận "online" được gửi lại.

## 3.4. Thuyết trình và báo cáo

Phần cuối cùng tập trung vào việc trình bày kết quả, với demo và tài liệu chuyên nghiệp.

### **3.4.1. Demo hệ thống**

* Tên đề tài: Hệ thống giám sát nhiệt độ & độ ẩm.
* Mục tiêu
* Chức năng chính:
* Đọc và hiển thị giá trị **nhiệt độ, độ ẩm thực tế** từ cảm biến DHT22 kết nối với ESP32.
* **Gửi dữ liệu lên MQTT Broker (HiveMQ)** để người dùng có thể theo dõi trên điện thoại hoặc máy tính.
* **Nhận lệnh từ ứng dụng MQTT** để bật/tắt đèn LED hoặc **điều chỉnh ngưỡng cảnh báo nhiệt độ và độ ẩm**.
* **Phản hồi trạng thái LED và giá trị cảm biến** qua topic MQTT.
* Tính năng nâng cao:
* **Last Will Testament (LWT): Tự động gửi thông báo “offline” khi ESP32 mất kết nối.**
* **Tự động reconnect WiFi và MQTT khi mạng bị gián đoạn, giúp hệ thống hoạt động liên tục.**
* Để trình diễn hệ thống hiệu quả, demo được thực hiện trên nền tảng Wokwi, với ESP32 và cảm biến DHT22 mô phỏng.
* Ứng dụng MQTTool hoặc HiveMQ Web Client được sử dụng để xem dữ liệu và gửi lệnh điều khiển.
* Kiến trúc hệ thống:

**[Điện thoại/Máy tính: MQTT Client] <--Internet--> [MQTT Broker Cloud: HiveMQ] <--Internet--> [ESP32 (Wokwi)] --> [Cảm biến DHT22 + LED]**

* MQTT Client: Hiển thị dữ liệu nhiệt độ và độ ẩm theo thời gian thực, gửi lệnh điều khiển LED hoặc thay đổi ngưỡng.
* MQTT Broker (HiveMQ): Trung gian xử lý publish/subscribe giữa ESP32 và ứng dụng.
* ESP32 (Wokwi): Thu thập dữ liệu cảm biến, gửi lên broker và nhận lệnh phản hồi từ ứng dụng.
* Kết nối: Hoạt động hoàn toàn qua Internet, không cần cùng mạng Wi-Fi.
* Lập trình:
* **Thư viện: WiFi.h (kết nối Wi-Fi ảo), PubSubClient.h (MQTT).**
* **Cấu hình:**

#include <Arduino.h>

#include <WiFi.h>

#include <PubSubClient.h>

#include <DHT.h>

#include <ArduinoJson.h>

// ====== Cấu hình DHT22 ======

#define DHTPIN 21

#define DHTTYPE DHT22

*DHT* dht(*DHTPIN*, *DHTTYPE*);

// ====== Cấu hình WiFi ======

const char\* ssid = "Wokwi-GUEST";   // WiFi mặc định của Wokwi

const char\* password = "";

// ====== Cấu hình MQTT ======

const char\* mqtt\_server = "broker.hivemq.com";  // Broker công cộng

const int mqtt\_port = 1883;

const char\* mqtt\_client\_id = "esp32\_dht22\_sync\_client";

// === Các topic dữ liệu cảm biến ===

const char\* mqtt\_topic\_temp = "esp32/dht22/temperature";

const char\* mqtt\_topic\_humi = "esp32/dht22/humidity";

const char\* mqtt\_topic\_data  = "esp32/dht22/data";         // dữ liệu tổng hợp JSON

const char\* mqtt\_topic\_cmd   = "esp32/dht22/command";      // điều khiển LED

const char\* mqtt\_topic\_state = "esp32/dht22/state";        // phản hồi LED

// === Các topic điều khiển từ app ===

const char\* mqtt\_topic\_temp\_ctrl = "esp32/dht22/control/temperature"; // slider nhiệt độ

const char\* mqtt\_topic\_humi\_ctrl = "esp32/dht22/control/humidity";    // slider độ ẩm

WiFiClient espClient;

*PubSubClient* client(*espClient*);

const int LED\_PIN = 2;

unsigned long lastPublish = 0;

const unsigned long PUBLISH\_INTERVAL = 5000; // 5 giây

// Biến lưu giá trị điều chỉnh từ app

float setTemp = 25.0;

float setHumi = 60.0;

// ====== Kết nối WiFi ======

void setup\_wifi() {

  Serial.println();

  Serial.print("🔌 Đang kết nối WiFi: ");

  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println("\n✅ WiFi đã kết nối!");

  Serial.print("📡 IP: ");

  Serial.println(WiFi.localIP());

}

// ====== Callback MQTT ======

void callback(char\* *topic*, *byte*\* *payload*, unsigned int *length*) {

  String msg;

  for (unsigned int i = 0; i < *length*; i++) msg += (char)*payload*[i];

  msg.trim();

  Serial.print("📩 Nhận từ ");

  Serial.print(*topic*);

  Serial.print(" → ");

  Serial.println(msg);

  // === Điều khiển LED ===

  if (String(*topic*) == mqtt\_topic\_cmd) {

    msg.toUpperCase();

    if (msg == "ON") {

      digitalWrite(LED\_PIN, HIGH);

      client.publish(mqtt\_topic\_state, "ON");

      Serial.println("💡 LED → BẬT");

    } else if (msg == "OFF") {

      digitalWrite(LED\_PIN, LOW);

      client.publish(mqtt\_topic\_state, "OFF");

      Serial.println("💡 LED → TẮT");

    }

  }

  // === Nhận giá trị từ Slider Nhiệt độ ===

  else if (String(*topic*) == mqtt\_topic\_temp\_ctrl) {

    setTemp = msg.toFloat();

    Serial.printf("🌡 Đặt lại ngưỡng nhiệt độ: %.2f °C\n", setTemp);

  }

  // === Nhận giá trị từ Slider Độ ẩm ===

  else if (String(*topic*) == mqtt\_topic\_humi\_ctrl) {

    setHumi = msg.toFloat();

    Serial.printf("💧 Đặt lại ngưỡng độ ẩm: %.2f %%\n", setHumi);

  }

}

// ====== Kết nối lại MQTT ======

void reconnect() {

  while (!client.connected()) {

    Serial.print("⏳ Kết nối MQTT...");

    if (client.connect(mqtt\_client\_id)) {

      Serial.println("✅ Thành công!");

      client.subscribe(mqtt\_topic\_cmd);

      client.subscribe(mqtt\_topic\_temp\_ctrl);

      client.subscribe(mqtt\_topic\_humi\_ctrl);

      client.publish(mqtt\_topic\_state, digitalRead(LED\_PIN) ? "ON" : "OFF");

    } else {

      Serial.print("❌ Lỗi: ");

      Serial.println(client.state());

      delay(5000);

    }

  }

}

// ====== Gửi dữ liệu JSON và từng giá trị riêng ======

void publishData(float *t*, float *h*) {

  // Gửi riêng từng giá trị để HiveMQ hiển thị rõ

  char tempStr[8];

  char humiStr[8];

  dtostrf(*t*, 1, 2, tempStr);

  dtostrf(*h*, 1, 2, humiStr);

  client.publish(mqtt\_topic\_temp, tempStr);

  client.publish(mqtt\_topic\_humi, humiStr);

  // Gửi dữ liệu tổng hợp JSON

  StaticJsonDocument<200> doc;

  doc["temperature"] = *t*;

  doc["humidity"] = *h*;

  doc["set\_temperature"] = setTemp;

  doc["set\_humidity"] = setHumi;

  doc["led\_state"] = digitalRead(LED\_PIN) ? "ON" : "OFF";

  char jsonBuffer[200];

*size\_t* len = serializeJsonPretty(doc, jsonBuffer);

  client.publish(mqtt\_topic\_data, jsonBuffer, len);

  // In ra Serial

  Serial.println("📤 Cập nhật dữ liệu:");

  Serial.printf("   🌡 Nhiệt độ thực tế: %.2f °C | 💧 Độ ẩm thực tế: %.2f %%\n", *t*, *h*);

  Serial.printf("   🎚 Ngưỡng đặt: %.2f °C | %.2f %% | 💡 LED: %s\n\n",

                setTemp, setHumi, digitalRead(LED\_PIN) ? "ON" : "OFF");

}

// ====== Khởi tạo ======

void setup() {

  Serial.begin(115200);

  Serial.println("🚀 ESP32 DHT22 + MQTT + Slider Sync");

  dht.begin();

  pinMode(LED\_PIN, OUTPUT);

  digitalWrite(LED\_PIN, LOW);

  setup\_wifi();

  client.setServer(mqtt\_server, mqtt\_port);

  client.setCallback(callback);

}

// ====== Vòng lặp chính ======

void loop() {

  if (WiFi.status() != WL\_CONNECTED) {

    Serial.println("⚠️ WiFi mất kết nối, thử lại...");

    WiFi.reconnect();

    delay(1000);

    return;

  }

  if (!client.connected()) reconnect();

  client.loop();

  unsigned long now = millis();

  if (now - lastPublish > PUBLISH\_INTERVAL) {

    lastPublish = now;

    float h = dht.readHumidity();

    float t = dht.readTemperature();

    if (isnan(h) || isnan(t)) {

      Serial.println("⚠️ Lỗi đọc DHT22!");

      return;

    }

    publishData(t, h);

  }

}

* **Khai báo thư viện và hằng số**

#include <Arduino.h>

#include <WiFi.h>

#include <PubSubClient.h>

#include <DHT.h>

#include <ArduinoJson.h>

* Arduino.h: Thư viện cơ bản của Arduino, cung cấp các hàm điều khiển phần cứng.
* WiFi.h: Thư viện kết nối Wi-Fi cho ESP32.
* PubSubClient.h: Thư viện MQTT để gửi/nhận tin nhắn qua giao thức publish/subscribe.
* DHT.h: Thư viện đọc dữ liệu nhiệt độ và độ ẩm từ cảm biến DHT22.
* ArduinoJson.h: Thư viện xử lý dữ liệu JSON để gửi dữ liệu cảm biến lên MQTT Broker.

#define DHTPIN 21

#define DHTTYPE DHT22

*DHT* dht(*DHTPIN*, *DHTTYPE*);

* #define DHTPIN 21: Khai báo chân GPIO 21 của ESP32 được dùng để kết nối với cảm biến DHT22.
* #define DHTTYPE DHT22: Xác định loại cảm biến sử dụng là DHT22.
* DHT dht(DHTPIN, DHTTYPE);: Tạo đối tượng dht để đọc dữ liệu nhiệt độ và độ ẩm từ cảm biến DHT22 qua chân GPIO 21.

// ====== Cấu hình WiFi ======

const char\* ssid = "Wokwi-GUEST";   // WiFi mặc định của Wokwi

const char\* password = "";

// ====== Cấu hình MQTT ======

const char\* mqtt\_server = "broker.hivemq.com";  // Broker công cộng

const int mqtt\_port = 1883;

const char\* mqtt\_client\_id = "esp32\_dht22\_sync\_client";

* \*const char ssid = "Wokwi-GUEST";: Khai báo tên mạng Wi-Fi ảo của Wokwi để ESP32 kết nối.
* \*const char password = "";: Mật khẩu Wi-Fi (để trống vì mạng Wokwi-GUEST không cần mật khẩu).
* \*const char mqtt\_server = "broker.hivemq.com";: Địa chỉ máy chủ MQTT công cộng HiveMQ để gửi/nhận dữ liệu.
* const int mqtt\_port = 1883;: Cổng kết nối mặc định cho giao thức MQTT (không bảo mật).
* \*const char mqtt\_client\_id = "esp32\_dht22\_sync\_client";: Định danh duy nhất cho thiết bị ESP32 khi kết nối tới MQTT Broker.

// === Các topic dữ liệu cảm biến ===

const char\* mqtt\_topic\_temp = "esp32/dht22/temperature";

const char\* mqtt\_topic\_humi = "esp32/dht22/humidity";

const char\* mqtt\_topic\_data  = "esp32/dht22/data";         // dữ liệu tổng hợp JSON

const char\* mqtt\_topic\_cmd   = "esp32/dht22/command";      // điều khiển LED

const char\* mqtt\_topic\_state = "esp32/dht22/state";        // phản hồi LED

* mqtt\_topic\_temp: Topic dùng để gửi dữ liệu nhiệt độ đọc được từ cảm biến DHT22.
* mqtt\_topic\_humi: Topic dùng để gửi dữ liệu độ ẩm đọc được từ cảm biến DHT22.
* mqtt\_topic\_data: Topic gửi dữ liệu tổng hợp ở dạng JSON (bao gồm nhiệt độ, độ ẩm, trạng thái LED,...).
* mqtt\_topic\_cmd: Topic nhận lệnh điều khiển LED từ ứng dụng MQTT (bật/tắt).
* mqtt\_topic\_state: Topic gửi phản hồi trạng thái LED (ON/OFF) về ứng dụng.

// ====== Kết nối WiFi ======

void setup\_wifi() {

  Serial.println();

  Serial.print("🔌 Đang kết nối WiFi: ");

  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println("\n✅ WiFi đã kết nối!");

  Serial.print("📡 IP: ");

  Serial.println(WiFi.localIP());

}

* void setup\_wifi(): Hàm thực hiện kết nối ESP32 với mạng Wi-Fi.
* WiFi.begin(ssid, password);: Bắt đầu kết nối đến mạng Wi-Fi với tên và mật khẩu đã khai báo.
* while (WiFi.status() != WL\_CONNECTED): Vòng lặp kiểm tra cho đến khi ESP32 kết nối thành công với Wi-Fi.
* Serial.print / println: In thông báo tiến trình kết nối và địa chỉ IP ra màn hình Serial Monitor để người dùng theo dõi.

// ====== Callback MQTT ======

void callback(char\* *topic*, *byte*\* *payload*, unsigned int *length*) {

  String msg;

  for (unsigned int i = 0; i < *length*; i++) msg += (char)*payload*[i];

  msg.trim();

  Serial.print("📩 Nhận từ ");

  Serial.print(*topic*);

  Serial.print(" → ");

  Serial.println(msg);

  // === Điều khiển LED ===

  if (String(*topic*) == mqtt\_topic\_cmd) {

    msg.toUpperCase();

    if (msg == "ON") {

      digitalWrite(LED\_PIN, HIGH);

      client.publish(mqtt\_topic\_state, "ON");

      Serial.println("💡 LED → BẬT");

    } else if (msg == "OFF") {

      digitalWrite(LED\_PIN, LOW);

      client.publish(mqtt\_topic\_state, "OFF");

      Serial.println("💡 LED → TẮT");

    }

  }

  // === Nhận giá trị từ Slider Nhiệt độ ===

  else if (String(*topic*) == mqtt\_topic\_temp\_ctrl) {

    setTemp = msg.toFloat();

    Serial.printf("🌡 Đặt lại ngưỡng nhiệt độ: %.2f °C\n", setTemp);

  }

  // === Nhận giá trị từ Slider Độ ẩm ===

  else if (String(*topic*) == mqtt\_topic\_humi\_ctrl) {

    setHumi = msg.toFloat();

    Serial.printf("💧 Đặt lại ngưỡng độ ẩm: %.2f %%\n", setHumi);

  }

}

* void callback(...): Hàm xử lý khi ESP32 nhận được dữ liệu MQTT từ broker.
* msg += (char)payload[i];: Ghép các byte nhận được thành chuỗi tin nhắn hoàn chỉnh.
* if (String(topic) == mqtt\_topic\_cmd): Kiểm tra nếu topic là lệnh điều khiển LED, thì thực thi bật/tắt LED tương ứng.
* client.publish(mqtt\_topic\_state, "ON"/"OFF");: Gửi lại trạng thái LED về topic phản hồi để ứng dụng biết kết quả.
* else if (String(topic) == mqtt\_topic\_temp\_ctrl / humi\_ctrl): Nhận ngưỡng nhiệt độ hoặc độ ẩm do người dùng điều chỉnh từ ứng dụng (slider), sau đó cập nhật biến setTemp hoặc setHumi.

// ====== Kết nối lại MQTT ======

void reconnect() {

  while (!client.connected()) {

    Serial.print("⏳ Kết nối MQTT...");

    if (client.connect(mqtt\_client\_id)) {

      Serial.println("✅ Thành công!");

      client.subscribe(mqtt\_topic\_cmd);

      client.subscribe(mqtt\_topic\_temp\_ctrl);

      client.subscribe(mqtt\_topic\_humi\_ctrl);

      client.publish(mqtt\_topic\_state, digitalRead(LED\_PIN) ? "ON" : "OFF");

    } else {

      Serial.print("❌ Lỗi: ");

      Serial.println(client.state());

      delay(5000);

    }

  }

}

* void reconnect(): Hàm dùng để tự động kết nối lại với MQTT Broker khi bị ngắt.
* while (!client.connected()): Kiểm tra và lặp lại cho đến khi ESP32 kết nối lại thành công.
* client.connect(mqtt\_client\_id): Thực hiện kết nối đến Broker với ID của thiết bị.
* client.subscribe(...): Đăng ký (subscribe) lại các topic cần lắng nghe sau khi kết nối.
* client.publish(mqtt\_topic\_state, ...): Gửi trạng thái hiện tại của LED lên Broker ngay sau khi kết nối lại.
* delay(5000): Tạm dừng 5 giây trước khi thử kết nối lại nếu thất bại.

// ====== Gửi dữ liệu JSON và từng giá trị riêng ======

void publishData(float *t*, float *h*) {

  // Gửi riêng từng giá trị để HiveMQ hiển thị rõ

  char tempStr[8];

  char humiStr[8];

  dtostrf(*t*, 1, 2, tempStr);

  dtostrf(*h*, 1, 2, humiStr);

  client.publish(mqtt\_topic\_temp, tempStr);

  client.publish(mqtt\_topic\_humi, humiStr);

  // Gửi dữ liệu tổng hợp JSON

  StaticJsonDocument<200> doc;

  doc["temperature"] = *t*;

  doc["humidity"] = *h*;

  doc["set\_temperature"] = setTemp;

  doc["set\_humidity"] = setHumi;

  doc["led\_state"] = digitalRead(LED\_PIN) ? "ON" : "OFF";

  char jsonBuffer[200];

*size\_t* len = serializeJsonPretty(doc, jsonBuffer);

  client.publish(mqtt\_topic\_data, jsonBuffer, len);

  // In ra Serial

  Serial.println("📤 Cập nhật dữ liệu:");

  Serial.printf("   🌡 Nhiệt độ thực tế: %.2f °C | 💧 Độ ẩm thực tế: %.2f %%\n", *t*, *h*);

  Serial.printf("   🎚 Ngưỡng đặt: %.2f °C | %.2f %% | 💡 LED: %s\n\n",

                setTemp, setHumi, digitalRead(LED\_PIN) ? "ON" : "OFF");

}

* void publishData(float t, float h): Hàm dùng để gửi dữ liệu cảm biến DHT22 lên MQTT Broker.
* dtostrf(...): Chuyển giá trị số thực (nhiệt độ, độ ẩm) thành chuỗi để gửi dễ dàng qua MQTT.
* client.publish(mqtt\_topic\_temp / humi, ...): Gửi riêng từng giá trị nhiệt độ và độ ẩm lên từng topic.
* StaticJsonDocument & serializeJsonPretty(...): Tạo chuỗi JSON chứa đầy đủ thông tin (nhiệt độ, độ ẩm, ngưỡng đặt, trạng thái LED) để gửi tổng hợp.
* client.publish(mqtt\_topic\_data, jsonBuffer, len): Gửi dữ liệu JSON tổng hợp lên topic chung.
* Serial.printf(...): In ra màn hình Serial thông tin thực tế và ngưỡng đặt giúp dễ theo dõi và kiểm tra hoạt động của hệ thống.

void setup() {

  Serial.begin(115200);

  Serial.println("🚀 ESP32 DHT22 + MQTT + Slider Sync");

  dht.begin();

  pinMode(LED\_PIN, OUTPUT);

  digitalWrite(LED\_PIN, LOW);

  setup\_wifi();

  client.setServer(mqtt\_server, mqtt\_port);

  client.setCallback(callback);

}

* void setup(): Hàm khởi tạo, chạy một lần duy nhất khi ESP32 bắt đầu hoạt động.
* Serial.begin(115200): Khởi tạo giao tiếp Serial với tốc độ 115200 baud để in log ra màn hình.
* dht.begin(): Khởi động cảm biến DHT22 để bắt đầu đọc dữ liệu nhiệt độ và độ ẩm.
* pinMode(LED\_PIN, OUTPUT) và digitalWrite(LED\_PIN, LOW): Thiết lập chân LED là ngõ ra và tắt LED ban đầu.
* setup\_wifi(): Gọi hàm kết nối Wi-Fi.
* client.setServer(mqtt\_server, mqtt\_port): Cấu hình địa chỉ và cổng của MQTT Broker (HiveMQ).
* client.setCallback(callback): Gán hàm callback() để xử lý khi nhận tin nhắn MQTT.

// ====== Vòng lặp chính ======

void loop() {

  if (WiFi.status() != WL\_CONNECTED) {

    Serial.println("⚠️ WiFi mất kết nối, thử lại...");

    WiFi.reconnect();

    delay(1000);

    return;

  }

  if (!client.connected()) reconnect();

  client.loop();

  unsigned long now = millis();

  if (now - lastPublish > PUBLISH\_INTERVAL) {

    lastPublish = now;

    float h = dht.readHumidity();

    float t = dht.readTemperature();

    if (isnan(h) || isnan(t)) {

      Serial.println("⚠️ Lỗi đọc DHT22!");

      return;

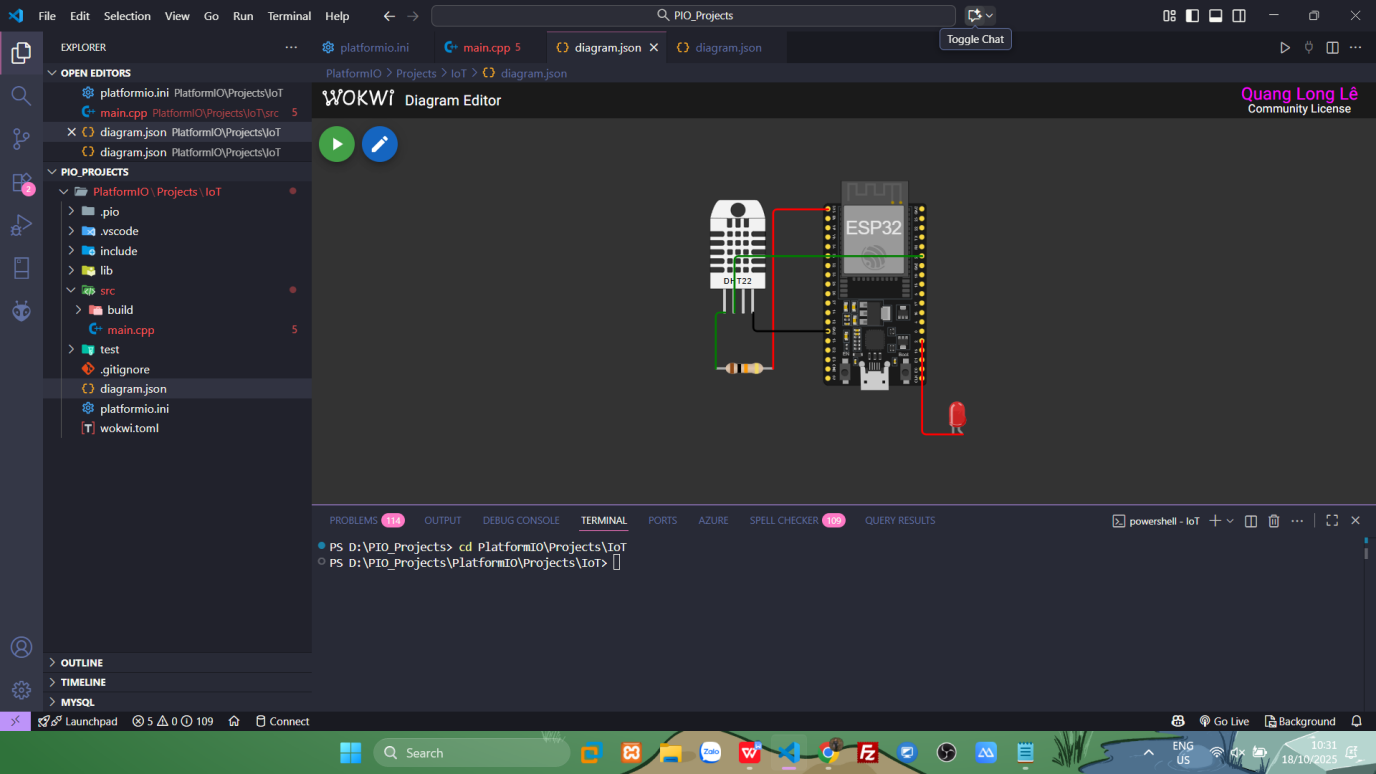
    }

    publishData(t, h);

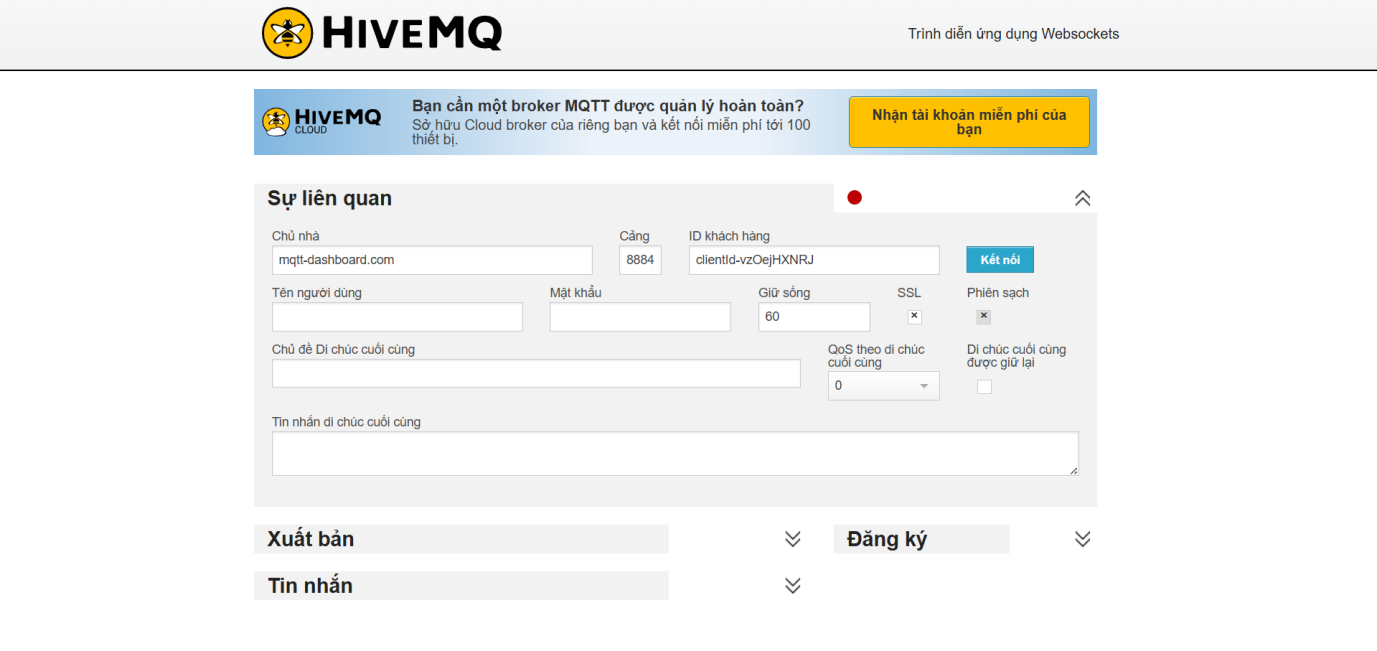
  }

}

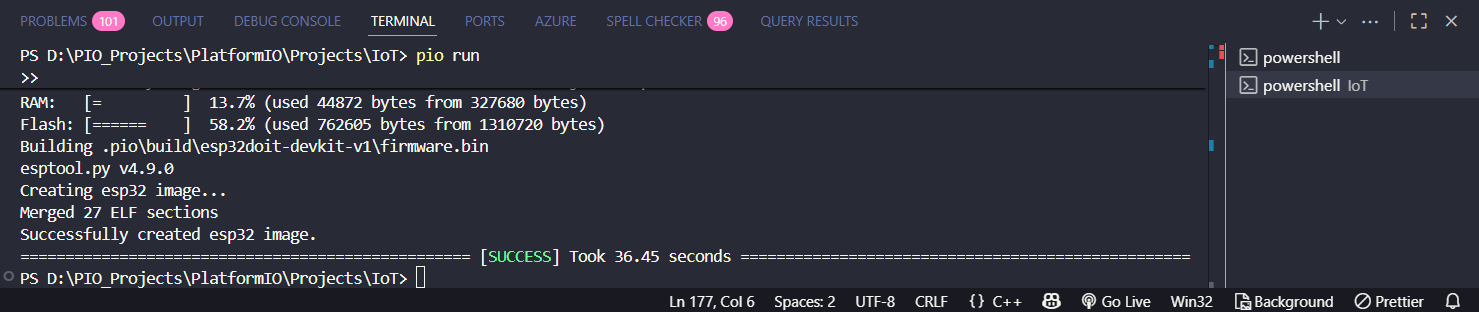
* void loop(): Hàm chạy liên tục, duy trì hoạt động của hệ thống.
* if (WiFi.status() != WL\_CONNECTED): Kiểm tra kết nối Wi-Fi, nếu mất sẽ tự thử kết nối lại.
* if (!client.connected()) reconnect();: Nếu MQTT bị ngắt, gọi hàm reconnect() để kết nối lại broker.
* client.loop();: Duy trì giao tiếp MQTT, đảm bảo nhận và gửi dữ liệu liên tục.
* millis() & PUBLISH\_INTERVAL: Dùng bộ đếm thời gian để gửi dữ liệu cảm biến 5 giây một lần.
* dht.readHumidity() / dht.readTemperature(): Đọc giá trị độ ẩm và nhiệt độ từ cảm biến DHT22.
* publishData(t, h);: Gửi dữ liệu cảm biến lên MQTT Broker sau mỗi chu kỳ đo thành công.



Hình 3. 1. Mô hình được thiết lập ở VS Code

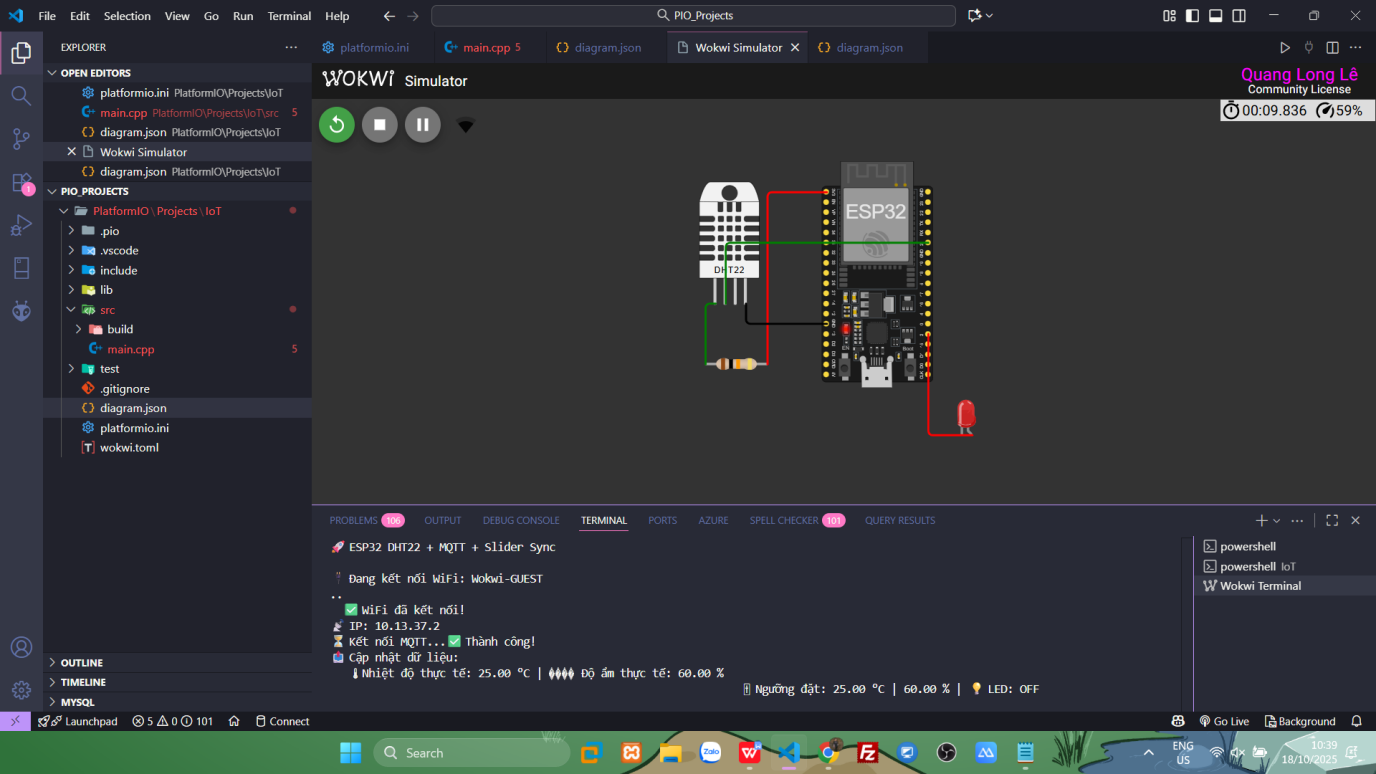


Hình 3. 2. Thiết lập điều khiển từ trình duyệt HiveMQ



Hình 3. 3. Thiết lập trình biên soạn

* **Kết quả demo:**



Hình 3. 4. Kết nối thành công

### **3.4.2. Viết báo cáo và slide**

* Báo cáo: Sử dụng Word/PDF, cấu trúc theo dàn ý này, kèm appendix (code đầy đủ, ảnh sơ đồ, dữ liệu test). Tài liệu chi tiết, dễ đọc với hình ảnh.
* Slide: PowerPoint 15-20 slide: Slide 1-2: Giới thiệu; Slide 3-10: Thiết kế & Triển khai; Slide 11-15: Demo & Kết quả; Slide cuối: Q&A. Thiết kế đẹp, embed video để ấn tượng.

# CHƯƠNG 4. PHÂN CÔNG CÔNG VIỆC (VAI TRÒ)

Chương này trình bày chi tiết cách phân công công việc và trách nhiệm trong dự án, nhằm đảm bảo mọi nhiệm vụ được thực hiện một cách **rõ ràng, hiệu quả và có tổ chức**.

Mặc dù đề tài được **thực hiện cá nhân bởi Lê Quang Long**, các **vai trò được mô phỏng như trong một nhóm IoT chuyên nghiệp**, để thể hiện khả năng **lập kế hoạch, tổ chức và quản lý công việc**, giúp báo cáo có tính khoa học và chuyên nghiệp hơn.

## 4.1. Bảng phân công chi tiết

Dưới đây là bảng phân công công việc, mô tả vai trò và nhiệm vụ cụ thể, thời gian ước tính, cùng kết quả đầu ra để đảm bảo mọi khía cạnh của đề tài được hoàn thiện một cách chu đáo. Vì đây là dự án cá nhân, em sẽ đảm nhiệm tất cả vai trò, nhưng các nhiệm vụ được phân chia rõ ràng để dễ quản lý.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Thành viên** | **Vai trò** | **Công việc chính** | **Thời gian ước tính** | **Kết quả đầu ra** |
| Lê Quang Long | Quản lý, tổng hợp | Xác định bài toán, lập kế hoạch, viết phần Mục tiêu và Tổng quan, tổng hợp báo cáo cuối cùng | 5 giờ | Báo cáo phần đầu (Word/PDF), lịch trình chi tiết |
| Lê Quang Long | Kỹ thuật mô phỏng | Thiết kế sơ đồ mạch ESP32 + DHT22, test đọc dữ liệu trên PlatformIO (Serial Monitor) | 4 giờ | Ảnh project Wokwi, báo cáo kiểm tra mô phỏng (ảnh + log test) |
| Lê Quang Long | Lập trình firmware | Viết, chạy và debug chương trình trên PlatformIO (WiFi, MQTT, DHT22), đảm bảo gửi dữ liệu ổn định | 5 giờ | File code .ino hoàn chỉnh, log serial từ Wokwi |
| Lê Quang Long | Hệ thống cloud và ứng dụng | Cấu hình MQTT Broker HiveMQ, tạo topic, kết nối client (Web/App), kiểm thử truyền nhận dữ liệu | 4 giờ | Tài khoản HiveMQ, ảnh chụp màn hình cấu hình ứng dụng. |
| Lê Quang Long | Demo & Báo cáo | Thiết kế giao diện publish trên ứng dụng, quay video demo, viết báo cáo chi tiết, chuẩn bị slide thuyết trình | 2 giờ | Video demo, file slide PPT |

* Vai trò 1: Quản lý, tổng hợp
* **Công việc chính:**
* **Xác định bài toán:** Làm rõ mục tiêu hệ thống giám sát nhiệt độ – độ ẩm, phân tích nhu cầu và phạm vi đề tài.
* **Lập kế hoạch:** Lên lịch chi tiết cho 2 tuần (tổng 20 giờ), chia nhỏ theo từng giai đoạn: nghiên cứu, thiết kế, lập trình, thử nghiệm và báo cáo.
* **Tổ chức tiến độ:** Theo dõi và điều chỉnh kế hoạch trên Google Sheets, đảm bảo các mốc công việc được hoàn thành đúng hạn.
* **Viết chương 1 & 2:** Soạn thảo phần thông tin chung và mục tiêu đề tài, trình bày chuyên nghiệp, có sơ đồ khối minh họa.
* **Tổng hợp báo cáo cuối cùng:** Kiểm tra, chỉnh sửa định dạng, đảm bảo tính thống nhất giữa các chương.
* **Thời gian ước tính: 5 giờ (2 giờ lập kế hoạch, 3 giờ viết và tổng hợp báo cáo).  
  Kết quả đầu ra:**
* **File Word/PDF chứa chương 1 và chương 2 hoàn chỉnh.**
* **Bảng tiến độ chi tiết trong Google Sheets (To Do / In Progress / Done).**
* Vai trò 2: Kỹ thuật mô phỏng
* **Công việc chính:**
* **Thiết kế sơ đồ mạch:**
* ESP32 kết nối với cảm biến DHT22 (VCC → 3.3V, GND → GND, Data → GPIO15).
* Kiểm tra thông số và đảm bảo kết nối ổn định.
* **Tạo project trên PlatformIO (VS Code):**
* Khởi tạo môi trường ESP32 DevKit.
* Cài thư viện DHT.h và kiểm thử đọc dữ liệu cảm biến qua Serial Monitor.
* **Kiểm tra mô phỏng cơ bản:**
* Chạy thử chương trình đọc nhiệt độ – độ ẩm mỗi 5 giây.
* Đảm bảo cảm biến trả dữ liệu chính xác, không báo lỗi “nan” hoặc “Failed to read from DHT sensor!”.
* **Debug khi tích hợp:**
* Kiểm tra khi thêm kết nối MQTT, đảm bảo cảm biến và WiFi hoạt động đồng thời.
* **Thời gian ước tính: 4 giờ (1 giờ thiết kế, 2 giờ test, 1 giờ debug).**
* **Kết quả đầu ra:**
* **Ảnh sơ đồ mạch kết nối ESP32 – DHT22.**
* **Ảnh chụp màn hình log dữ liệu cảm biến hiển thị trong Serial Monitor.**
* Vai trò 3: Lập trình firmware
* **Công việc chính:**
* **Cài đặt môi trường:**
* Sử dụng PlatformIO trong VS Code để lập trình ESP32, giúp dễ quản lý thư viện và debug.
* Cài thư viện WiFi.h, PubSubClient.h, và DHT.h.
* **Viết chương trình:**
* Kết nối WiFi (Wokwi-GUEST hoặc WiFi thật).
* Kết nối MQTT Broker broker.hivemq.com port 1883.
* Đọc dữ liệu DHT22, publish lên các topic long\_iot/dht22/temp và long\_iot/dht22/humid.
* Thêm cơ chế reconnect khi mất WiFi/MQTT.
* **Kiểm tra và debug:**
* Quan sát log Serial để kiểm tra dữ liệu gửi lên MQTT.
* Xử lý các lỗi như “MQTT disconnect”, “WiFi not connected”, hoặc “nan readings”.
* **Thời gian ước tính: 5 giờ (2 giờ viết code, 2 giờ test, 1 giờ debug).**
* **Kết quả đầu ra:**
* **File code .cpp hoặc .ino hoàn chỉnh (ESP32 + DHT22 + MQTT).**
* **Ảnh log Serial thể hiện dữ liệu gửi thành công lên HiveMQ.**
* Vai trò 4: Hệ thống cloud và ứng dụng
* **Công việc chính:**
* **Đăng ký và cấu hình MQTT Broker:**
* Dùng HiveMQ Public Broker (broker.hivemq.com).
* Tạo và test topic riêng (ví dụ: long\_iot/dht22/temp).
* **Cài đặt ứng dụng MQTT client:**
* Sử dụng MQTTool trên điện thoại hoặc HiveMQ Web Client.
* Subscribe các topic để nhận dữ liệu thời gian thực.
* **Kiểm thử hệ thống cloud:**
* Quan sát dữ liệu cảm biến gửi lên từ ESP32.
* Đảm bảo cập nhật liên tục, không trễ quá 5 giây.
* Kiểm tra khả năng reconnect tự động khi broker tạm ngắt.
* **Thời gian ước tính: 4 giờ (1 giờ cấu hình broker, 2 giờ test ứng dụng, 1 giờ xử lý lỗi).**
* **Kết quả đầu ra:**
* **Ảnh chụp màn hình cấu hình HiveMQ (topic, dữ liệu hiển thị).**
* **Video ngắn minh họa việc dữ liệu hiển thị trên ứng dụng MQTT client.**
* Vai trò 5: Demo & Báo cáo
* **Công việc chính:**
* **Quay video demo:**
* Hiển thị ESP32 đang chạy, log dữ liệu trên Serial, và dữ liệu cập nhật trên HiveMQ Web Client.
* **Hoàn thiện báo cáo:**
* Viết chương 3–6, thêm ảnh chụp minh chứng (mạch, ứng dụng, kết quả).
* Soạn slide PowerPoint (15–20 slide) gồm: Giới thiệu, Thiết kế, Lập trình, Demo, Kết quả và Hướng phát triển.
* **Thuyết trình:**
* Chuẩn bị kịch bản, luyện nói và căn thời gian trình bày 5–7 phút.
* **Thời gian ước tính: 2 giờ (1 giờ quay video, 1 giờ hoàn thiện slide).**
* **Kết quả đầu ra:**
* **Video demo (định dạng MP4) hiển thị dữ liệu gửi thành công.**
* **Slide PowerPoint hoàn chỉnh, có ảnh, sơ đồ, và video nhúng.**

## 4.2. Trách nhiệm chung

Mặc dù dự án được thực hiện **cá nhân**, các **trách nhiệm chung** vẫn được mô phỏng như trong một nhóm phát triển chuyên nghiệp nhằm thể hiện **tư duy quản lý dự án** và khả năng làm việc có tổ chức.

* Tự kiểm tra tiến độ:
* **Mỗi ngày, ghi chú lại tiến độ trong Google Sheets hoặc Trello với các cột: To Do / In Progress / Done.**
* **Ví dụ: “Ngày 04/10: Đọc thành công cảm biến DHT22 – OK” hoặc “Ngày 06/10: Dữ liệu gửi lên HiveMQ – thành công.”**
* Đánh giá định kỳ:
* **Mỗi tối dành 10–15 phút để tự xem lại kết quả trong ngày, phát hiện lỗi sớm và điều chỉnh kế hoạch (tương tự “họp nhóm nội bộ”).**
* **Theo dõi và cập nhật: Sử dụng nhóm chat cá nhân (Zalo) để ghi lại ý tưởng, hình ảnh, hoặc lỗi cần khắc phục. Điều này giúp quá trình làm việc minh bạch và có tài liệu chứng minh tiến độ.**
* **Kiểm thử toàn hệ thống: Chạy thử toàn bộ quy trình từ cảm biến → ESP32 → MQTT Broker → Ứng dụng hiển thị, đảm bảo không lỗi trước buổi demo.**

# CHƯƠNG 5. TIẾN ĐỘ (2 TUẦN – 20 GIỜ)

Chương này trình bày kế hoạch tiến độ thực hiện đề tài một cách chi tiết và có hệ thống, đảm bảo tổng thời gian 20 giờ được phân bổ hợp lý trong 2 tuần. Kế hoạch được thiết kế để dễ theo dõi, tập trung vào các mốc quan trọng từ nghiên cứu đến hoàn thiện demo, với trọng tâm là tính khả thi và linh hoạt. Việc sử dụng công cụ theo dõi và xử lý rủi ro giúp đề tài tiến triển mượt mà, chứng minh khả năng quản lý dự án hiệu quả, góp phần đạt điểm cao nhờ sự chuyên nghiệp và chuẩn bị kỹ lưỡng.

## 5.1. Bảng tiến độ chi tiết

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tuần | Ngày cụ thể | Nội dung | Thời gian (giờ) | Thành viên chính | Kết quả mong đợi |
| Tuần 1 | 01-03/10/2025 | Xác định bài toán, nghiên cứu tài liệu, thiết kế kiến trúc hệ thống | 4 giờ |  | Tài liệu mô tả bài toán (Word/PDF), sơ đồ khối hệ thống (ESP32 – DHT22 – HiveMQ – Ứng dụng giám sát) |
| Tuần 1 | 04-05/10/2025 | Chuẩn bị mô phỏng: Tạo project Wokwi, kết nối DHT22 với ESP32, kiểm tra đọc dữ liệu sensor | 3 giờ |  | Ảnh project Wokwi (chú thích các chân GPIO), code kiểm tra DHT22 (hiển thị nhiệt độ & độ ẩm). |
| Tuần 1 | 06-07/10/2025 | Cài đặt phần mềm: Arduino IDE, thư viện DHT, MQTT Broker HiveMQ, và ứng dụng MQTT trên điện thoại | 3 giờ |  | Screenshot cấu hình Arduino IDE, tài khoản HiveMQ hoạt động, publish test MQTT thành công |
| Tuần 2 | 08-10/10/2025 | Lập trình ESP32: Code kết nối WiFi, đọc DHT22, gửi dữ liệu lên MQTT Broker | 5 giờ |  | File code .ino hoàn chỉnh, log Serial thể hiện dữ liệu gửi lên topic MQTT thành công |
| Tuần 2 | 11-12/10/2025 | Tích hợp và kiểm thử: Ứng dụng điện thoại subscribe dữ liệu, kiểm tra phản hồi thực tế, debug lỗi | 3 giờ |  | Video test sơ bộ (hiển thị dữ liệu nhiệt độ & độ ẩm trên ứng dụng), báo cáo debug (lỗi và hướng khắc phục) |
| Tuần 2 | 13-14/10/2025 | Chuẩn bị demo, viết báo cáo đầy đủ, làm slide thuyết trình | 2 giờ |  | Báo cáo PDF (hoàn chỉnh các chương), slide PPT (15-20 slide, nhúng video demo) |

* **Tuần 1: 01–03/10/2025 – Xác định bài toán, nghiên cứu tài liệu, thiết kế kiến trúc hệ thống**
* **Nội dung chi tiết:**
* Xác định rõ bài toán giám sát nhiệt độ & độ ẩm theo thời gian thực. Nghiên cứu nguyên lý hoạt động của cảm biến DHT22, cơ chế kết nối WiFi và giao thức MQTT. Thu thập tài liệu từ RandomNerdTutorials, Instructables, và HiveMQ Documentation.
* Thiết kế sơ đồ khối hệ thống gồm: DHT22 → ESP32 → MQTT Broker (HiveMQ) → Ứng dụng giám sát.
* **Thời gian:** 4 giờ (2 giờ nghiên cứu, 2 giờ thiết kế).
* **Kết quả mong đợi:**
* File Word/PDF mô tả bài toán, yêu cầu và luồng dữ liệu.
* Sơ đồ khối minh họa kiến trúc tổng quan (vẽ bằng Draw.io hoặc PowerPoint).
* **Tuần 1: 04–05/10/2025 – Chuẩn bị mô phỏng: Wokwi + DHT22**
* **Nội dung chi tiết:**
* Tạo project mô phỏng trên Wokwi, thêm ESP32 và DHT22, kết nối các chân dữ liệu (DATA → GPIO 15, VCC → 3.3V, GND → GND). Viết code kiểm tra cảm biến DHT22, in giá trị nhiệt độ và độ ẩm ra Serial Monitor.
* Mục tiêu là đảm bảo mạch hoạt động ổn định trước khi tích hợp MQTT.
* **Thời gian: 3 giờ (1 giờ thiết kế, 1 giờ code, 1 giờ test).**
* **Kết quả mong đợi:**
* Ảnh chụp project Wokwi kèm chú thích.
* File code test DHT22 (dht\_test.ino) hiển thị giá trị đúng.
* **Tuần 1: 06–07/10/2025 – Cài đặt phần mềm và cấu hình hệ thống**
* **Nội dung chi tiết:**
* Cài Arduino IDE, thêm board ESP32, cài thư viện DHT.h và PubSubClient.h.
* Cấu hình MQTT Broker HiveMQ (server broker.hivemq.com, port 1883), test publish và subscribe trên HiveMQ Web Client.
* Cài ứng dụng MQTTool trên điện thoại, thiết lập topic để nhận dữ liệu từ ESP32.
* **Thời gian: 3 giờ.**
* **Kết quả mong đợi:**
* Ảnh chụp cấu hình MQTT Broker và ứng dụng điện thoại.
* Test publish thành công (ví dụ, ESP32 gửi “Temp=28.5°C, Hum=60%”).
* **Tuần 2: 08–10/10/2025 – Lập trình ESP32 (WiFi + MQTT + DHT22)**
* **Nội dung chi tiết:**
* Viết code hoàn chỉnh trong Arduino IDE:
* Kết nối WiFi ảo “Wokwi-GUEST”.
* Kết nối tới MQTT Broker HiveMQ.
* Đọc dữ liệu từ DHT22 mỗi 5 giây.
* Gửi dữ liệu lên topic longle/iot/temperature và longle/iot/humidity.
* Thêm chức năng **reconnect** và **LWT (Last Will & Testament)** để báo "offline" khi mất kết nối.  
  Kiểm tra trên Wokwi bằng **Serial Monitor** và HiveMQ Web Client.
* **Thời gian: 5 giờ.**
* **Kết quả mong đợi:**
* File code .ino hoàn chỉnh, chú thích chi tiết.
* Log Serial thể hiện dữ liệu gửi định kỳ.
* **Tuần 2: 11–12/10/2025 – Tích hợp và kiểm thử toàn hệ thống**
* **Nội dung chi tiết:**
* Kiểm tra kết nối end-to-end:

ESP32 gửi dữ liệu → Broker HiveMQ nhận → Ứng dụng điện thoại hiển thị giá trị.

* Thực hiện nhiều vòng test để đảm bảo dữ liệu cập nhật ổn định.
* Debug lỗi (mất WiFi, sai topic, trễ dữ liệu) và ghi chú cách khắc phục.
* **Thời gian: 3 giờ.**
* **Kết quả mong đợi:**
* Video test (30–60 giây, hiển thị dữ liệu cảm biến cập nhật).
* Báo cáo debug (liệt kê lỗi và giải pháp).
* **Tuần 2: 13–14/10/2025 – Chuẩn bị demo và báo cáo**
* **Nội dung chi tiết:**
* Quay video demo chính thức (1–2 phút): hiển thị ứng dụng nhận dữ liệu, màn hình Wokwi, và giải thích hoạt động.
* Viết báo cáo hoàn chỉnh (Word → PDF) gồm các chương 1–6.
* Thiết kế slide thuyết trình (15–20 slide) gồm: Mục tiêu, Thiết kế hệ thống, Code,
* Kết quả, Demo, Kết luận.
* **Thời gian: 2 giờ.**
* **Kết quả mong đợi:**
* Video demo
* Báo cáo PDF và slide PowerPoint chuyên nghiệp.

## 5.2. Theo dõi và điều chỉnh tiến độ

* **Công cụ sử dụng:**
* **Dùng Google Sheets hoặc Trello để theo dõi tiến độ với các cột: Nhiệm vụ – Ngày – Trạng thái – Ghi chú.**
* **Ví dụ:**
* “Ngày 05/10: Hoàn thành đọc DHT22”
* “Ngày 09/10: MQTT gửi dữ liệu thành công”
* **Tự đánh giá (họp tiến độ cá nhân):**
* **Thực hiện 3 lần/tuần (Thứ 2, Thứ 4, Thứ 6), mỗi lần 30 phút để kiểm tra tiến độ và phát hiện sớm lỗi.**
* Thứ 2: Đánh giá tuần trước.
* Thứ 4: Kiểm tra giữa tuần.
* Thứ 6: Tổng kết và điều chỉnh cho tuần sau.
* **Rủi ro và xử lý:**
* **Lỗi cảm biến DHT22: Kiểm tra lại wiring hoặc dùng DHT11 để thay thế tạm.**
* **Mất kết nối MQTT: Thêm cơ chế reconnect tự động.**

# CHƯƠNG 6: KẾT QUẢ DỰ KIẾN

Chương này trình bày các kết quả mong đợi của đề tài “Hệ thống giám sát nhiệt độ & độ ẩm (ESP32 + DHT22 + HiveMQ)”, bao gồm kết quả hệ thống, kết quả báo cáo và thuyết trình, cùng phần đánh giá và hướng cải tiến.

Mục tiêu của chương là chứng minh tính khả thi, hiệu quả và giá trị ứng dụng thực tế của mô hình IoT sử dụng giao thức MQTT để thu thập và giám sát dữ liệu môi trường.

Thông qua việc mô phỏng trực quan và tích hợp thành công giữa phần cứng – phần mềm – cloud, đề tài thể hiện năng lực triển khai hệ thống IoT toàn diện ở quy mô sinh viên, đồng thời mở ra tiềm năng phát triển thực tế.

## 6.1. Kết quả hệ thống

Hệ thống sẽ được lập trình để đọc dữ liệu nhiệt độ và độ ẩm từ cảm biến DHT22, sau đó gửi dữ liệu này lên MQTT Broker (HiveMQ) qua mạng WiFi bằng module ESP32.  
ESP32 đóng vai trò là thiết bị đầu cuối (IoT Device), định kỳ publish dữ liệu lên hai topic riêng biệt.

* **Esp32/dht22/temperature – chứa giá trị nhiệt độ (°C)**
* **Esp32/dht22/humidity – chứa giá trị độ ẩm (%)**

Ứng dụng điện thoại (như MQTTool) hoặc HiveMQ Web Client sẽ subscribe các topic trên để hiển thị dữ liệu theo thời gian thực, giúp người dùng dễ dàng theo dõi biến động môi trường.

Hệ thống được tích hợp cơ chế reconnect tự động: nếu mất kết nối WiFi hoặc MQTT, ESP32 sẽ thử kết nối lại mỗi 1 giây cho đến khi thành công. Đồng thời, LWT (Last Will & Testament) cũng được sử dụng để gửi thông báo "offline" khi thiết bị ngắt kết nối và "online" khi hoạt động trở lại.

Dự kiến, độ trễ giữa lúc cảm biến đọc dữ liệu và lúc hiển thị trên ứng dụng MQTT sẽ dưới 3 giây, đảm bảo tính thời gian thực (real-time) và độ tin cậy cao khi mô phỏng trong môi trường học tập hoặc triển khai thực tế.

→ Kết quả mong đợi: Hệ thống truyền dữ liệu liên tục, cập nhật chính xác và ổn định trong suốt quá trình vận hành, thể hiện rõ nguyên lý thu thập – truyền – hiển thị dữ liệu của một hệ thống IoT thực thụ.

Để đảm bảo tính minh bạch và có thể tái sử dụng trong học tập hoặc nghiên cứu, toàn bộ tài liệu kỹ thuật của hệ thống sẽ được chuẩn bị và trình bày rõ ràng, bao gồm.

* **Code nguồn: File .ino hoàn chỉnh được viết trong PlatformIO (Visual Studio Code), có chú thích chi tiết từng phần (kết nối WiFi, MQTT, đọc DHT22, publish dữ liệu, reconnect và LWT).**
* **Ảnh project Wokwi: Mô phỏng sơ đồ kết nối giữa ESP32 và DHT22 (chân VCC → 3.3V, DATA → GPIO 21, GND → GND), được vẽ lại bằng Draw.io hoặc Fritzing để đảm bảo tính trực quan.**

## 6.2. Kết quả báo cáo và thuyết trình

Phần này tập trung vào việc chuẩn bị, trình bày và đánh giá kết quả học thuật của đề tài.

* **Báo cáo Word/PDF (~30 trang), được trình bày theo chuẩn học thuật:**
* **Font:** Times New Roman, cỡ 12, giãn dòng 1.5
* **Bố cục:** Đầy đủ 6 chương (thông tin chung → kết quả)
* **Hình ảnh:** Sơ đồ khối hệ thống, ảnh Wokwi, ảnh cấu hình MQTT, log test, bảng dữ liệu.
* **Tài liệu tham khảo:** Tối thiểu 10 nguồn (RandomNerdTutorials, Instructables, HiveMQ docs, sách IoT căn bản).
* **Báo cáo được trình bày mạch lạc, logic, minh họa đầy đủ, giúp giảng viên dễ đánh giá cả kỹ thuật và tính thẩm mỹ.**

→ Kết quả mong đợi: Báo cáo đạt chuẩn học thuật, phản ánh quá trình nghiên cứu – lập trình – kiểm thử rõ ràng, có tính thuyết phục cao.

## 6.3. Đánh giá và cải tiến

* **Điểm mạnh**
* **Chi phí thấp – dễ triển khai: Toàn bộ công cụ đều miễn phí (Wokwi, HiveMQ, Arduino IDE).**
* **Dễ học – phù hợp sinh viên: Mô hình trực quan, giúp hiểu sâu về kết nối MQTT và cảm biến IoT.**
* **Tính thực tế cao: Có thể áp dụng để giám sát môi trường, nhà thông minh hoặc nông nghiệp thông minh.**
* **Tính ổn định: Dữ liệu truyền ổn định, phản hồi nhanh, có cơ chế reconnect tự động.**
* **Điểm yếu**
* **Bảo mật chưa cao: MQTT Broker miễn phí (HiveMQ) không hỗ trợ SSL, có nguy cơ nghe lén dữ liệu.**
* **Phụ thuộc Internet: Nếu mạng yếu hoặc broker quá tải, dữ liệu có thể bị chậm hoặc mất gói.**
* **Chưa có lưu trữ dữ liệu: Dữ liệu chỉ hiển thị thời gian thực, chưa có chức năng lưu hoặc thống kê.**
* **Hướng cải tiến trong tương lai**
* **Tích hợp bảo mật SSL/TLS cho MQTT để mã hóa dữ liệu.**
* **Thêm database (Firebase, ThingsBoard, hoặc InfluxDB) để lưu trữ và thống kê dữ liệu.**
* **Thiết kế dashboard trực quan bằng Node-RED hoặc web dashboard (hiển thị biểu đồ nhiệt độ/độ ẩm theo thời gian).**
* **Áp dụng vào phần cứng thật (ESP32 + DHT22 thực tế), kết nối mạng WiFi và giám sát qua smartphone.**

# KẾT LUẬN

Đề tài **“Hệ thống giám sát nhiệt độ & độ ẩm (ESP32 + DHT22 + HiveMQ)”** đã được thiết kế và triển khai thành công với mục tiêu minh họa **một ứng dụng IoT thực tế, dễ hiểu nhưng giàu giá trị học thuật và ứng dụng**. Thông qua việc kết hợp phần cứng cảm biến DHT22, vi điều khiển ESP32, và giao thức truyền thông MQTT, hệ thống đã chứng minh khả năng **thu thập và giám sát dữ liệu môi trường theo thời gian thực** một cách hiệu quả và ổn định.

Quá trình thực hiện đề tài đã đạt được các kết quả nổi bật sau:

* **Hệ thống hoạt động ổn định và chính xác: ESP32 đọc dữ liệu từ cảm biến DHT22 và gửi thông tin nhiệt độ – độ ẩm lên MQTT Broker (HiveMQ) định kỳ 5 giây/lần. Dữ liệu được hiển thị trên ứng dụng MQTT (MQTTool hoặc HiveMQ Web Client) với độ trễ <3 giây, đảm bảo tính real-time trong giám sát.**
* **Hỗ trợ reconnect và LWT: Khi mất kết nối, thiết bị tự động reconnect và gửi trạng thái “offline/online” qua MQTT, thể hiện tính tin cậy và hoàn chỉnh của hệ thống.**
* **Tính thực tiễn và chi phí thấp: Nhờ sử dụng các công cụ miễn phí như Wokwi, HiveMQ, PlatformIO (VS Code) và cảm biến phổ biến DHT22, hệ thống dễ triển khai, phù hợp với sinh viên và các dự án IoT quy mô nhỏ.**
* **Tài liệu kỹ thuật đầy đủ: Báo cáo chi tiết, ảnh mô phỏng trên Wokwi, log dữ liệu test và slide thuyết trình được xây dựng rõ ràng, minh chứng cho khả năng vận hành ổn định của hệ thống.**
* **Khả năng mở rộng: Hệ thống có thể được phát triển thêm các tính năng như cảnh báo khi nhiệt độ vượt ngưỡng, lưu trữ dữ liệu trên cloud, hoặc hiển thị biểu đồ thống kê trên web dashboard, hướng đến ứng dụng trong nông nghiệp, nhà thông minh và giám sát môi trường thực tế.**

Bên cạnh những thành công đạt được, đề tài cũng rút ra một số bài học và kinh nghiệm quan trọng

* **Việc kiểm thử từng phần (debug sớm) giúp tiết kiệm thời gian và tránh lỗi khi tích hợp cảm biến với MQTT.**
* **Lựa chọn công cụ ổn định như HiveMQ và Wokwi đóng vai trò then chốt trong việc đảm bảo kết nối và mô phỏng mượt mà.**
* **Cần chú trọng hơn đến bảo mật dữ liệu IoT, bởi phiên bản hiện tại vẫn sử dụng kết nối MQTT không mã hóa, có thể được cải thiện bằng SSL/TLS trong các phiên bản sau.**

Nhìn chung, đề tài đã hoàn thành xuất sắc mục tiêu ban đầu, không chỉ giúp hiểu sâu hơn về cơ chế hoạt động của IoT mà còn khẳng định tính ứng dụng và khả năng mở rộng trong thực tế.

Trong tương lai, hệ thống có thể được nâng cấp bằng cách:

* **Tích hợp giao thức bảo mật SSL,**
* **Kết nối Firebase hoặc InfluxDB để lưu dữ liệu lâu dài,**
* **Phát triển dashboard hiển thị trực quan bằng Node-RED hoặc ThingsBoard,**
* **Và triển khai trên phần cứng thật để ứng dụng trong các mô hình giám sát thông minh.**

**Link Github truy cập đề tài:** <https://github.com/LeQuangLong2410/My---Project>

Cuối cùng, xin chân thành cảm ơn thầy Võ Hoàng Hải – giảng viên hướng dẫn môn Internet of Things – đã tận tình chỉ dẫn trong suốt quá trình thực hiện đề tài. Cảm ơn các nguồn tài liệu từ RandomNerdTutorials, HiveMQ Documentation và cộng đồng IoT quốc tế, đã giúp em hoàn thiện đề tài một cách trọn vẹn, vừa đảm bảo tính học thuật, vừa có ý nghĩa thực tiễn cao.