HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

BÁO CÁO BÀI TẬP LỚN

HỆ THỐNG NHÀ THÔNG MINH

Giảng viên hướng dẫn : **ThS. Nguyễn Quốc Uy**
Họ và tên sinh viên : **Đinh Thế Anh**
Mã sinh viên : **B21DCCN004**
Lớp môn học : **06**

Hà Nội – 2024

Lời cảm ơn

Trước tiên, em xin bày tỏ lòng biết ơn chân thành tới thầy Uy, người đã tận tình hướng dẫn và hỗ trợ em trong suốt quá trình học tập và thực hiện đề tài cá nhân "Hệ thống nhà thông minh". Những bài giảng của thầy không chỉ cung cấp kiến thức chuyên môn về IoT mà còn giúp em phát triển tư duy, kỹ năng giải quyết vấn đề và hiểu rõ hơn về việc ứng dụng công nghệ trong thực tế. Sự tận tâm của thầy trong việc hướng dẫn và khích lệ học viên là nguồn động lực to lớn giúp em hoàn thành đề tài này.

Bên cạnh đó, em cũng xin gửi lời cảm ơn đến gia đình, bạn bè và những người đã đồng hành, hỗ trợ em trong suốt quá trình thực hiện dự án. Sự ủng hộ về mặt tinh thần cũng như những lời khuyên quý báu từ mọi người đã giúp em vượt qua những khó khăn trong quá trình nghiên cứu và triển khai hệ thống.

Em nhận thức được rằng, bài làm này còn nhiều thiếu sót do khả năng cá nhân cũng như những hạn chế về thiết bị và điều kiện thử nghiệm thực tế. Vì vậy, em rất mong nhận được những góp ý và phản hồi từ thầy để có thể hoàn thiện bài làm của mình hơn nữa trong tương lai. Những lời góp ý của thầy sẽ là cơ hội để em học hỏi, rút kinh nghiệm và tiếp tục phát triển bản thân.

Một lần nữa, em xin chân thành cảm ơn thầy và tất cả những người đã hỗ trợ em trong quá trình thực hiện đề tài này.

MỤC LỤC

Chương 1. Giới thiệu

Trong bối cảnh cuộc cách mạng công nghiệp 4.0, Internet of Things (IoT) đã và đang trở thành một trong những xu hướng phát triển công nghệ nổi bật nhất. Cách mạng 4.0 là sự hội tụ của các công nghệ hiện đại như trí tuệ nhân tạo (AI), dữ liệu lớn (Big Data), và đám mây (Cloud Computing), với IoT đóng vai trò là cầu nối giữa các thiết bị vật lý và thế giới số. IoT không chỉ thay đổi cách chúng ta sống, làm việc, mà còn cách mạng hóa hầu hết các ngành công nghiệp, từ sản xuất, nông nghiệp đến chăm sóc sức khỏe và vận tải.

Theo dự báo từ các tổ chức nghiên cứu, số lượng thiết bị IoT sẽ đạt đến hàng chục tỷ trên toàn thế giới trong vài năm tới. Tốc độ tăng trưởng của IoT phản ánh tầm quan trọng của việc kết nối các thiết bị và khai thác dữ liệu từ đó nhằm nâng cao hiệu suất và tối ưu hóa hoạt động trong nhiều lĩnh vực khác nhau. Hệ thống nhà thông minh, một ứng dụng tiêu biểu của IoT, mang lại nhiều lợi ích thiết thực như tiết kiệm năng lượng, nâng cao tiện nghi, và đảm bảo an ninh cho ngôi nhà.

Dự án của chúng tôi tập trung vào việc phát triển một hệ thống IoT đơn giản nhưng mạnh mẽ, sử dụng vi điều khiển ESP8266 kết hợp với các cảm biến để đo lường các thông số môi trường và điều khiển các thiết bị gia dụng từ xa. Hệ thống này không chỉ có khả năng thu thập dữ liệu mà còn cung cấp giao diện web, cho phép người dùng dễ dàng quản lý và theo dõi thông tin theo thời gian thực, từ đó ứng dụng trong các kịch bản nhà thông minh hay tự động hóa công nghiệp.

Mục tiêu của dự án không chỉ dừng lại ở việc tạo ra một giải pháp IoT cơ bản mà còn giúp mở rộng hiểu biết về cách IoT có thể được áp dụng để giải quyết các vấn đề thực tiễn trong cuộc sống và công nghiệp, góp phần thúc đẩy sự phát triển của công nghệ tại Việt Nam trong thời đại 4.0.

1.1 Mục tiêu của dự án

Mục tiêu chính của dự án là tạo ra một hệ thống IoT toàn diện với các chức năng cơ bản nhưng thiết yếu, đáp ứng được nhu cầu thu thập và điều khiển các thiết bị trong môi trường sống hoặc sản xuất, thông qua các tính năng sau:

- **Thu thập dữ liệu môi trường**: Sử dụng cảm biến DHT11 để đo nhiệt độ và độ ẩm, cùng với module cảm biến ánh sáng quang trở để giám sát điều kiện ánh sáng môi trường. Những dữ liệu này có thể được phân tích để cải thiện chất lượng không khí, điều chỉnh ánh sáng phù hợp, và tối ưu hóa nhiệt độ môi trường.
- **Điều khiển thiết bị từ xa**: Sử dụng các thiết bị như quạt, đèn, và tivi, thông qua giao thức MQTT, hệ thống cho phép điều khiển từ xa và cập nhật trạng thái theo thời gian thực thông qua LED. Việc này không chỉ tăng tính tiện lợi mà còn giúp tiết kiệm năng lượng bằng cách kiểm soát các thiết bị khi không cần thiết.

- **Hiển thị thông tin**: Dữ liệu từ các cảm biến được hiển thị thông qua giao diện web xây dựng bằng Node.js, giúp người dùng giám sát trạng thái của thiết bị và xem biểu đồ theo thời gian thực. Việc sử dụng Highcharts giúp trực quan hóa dữ liệu, hỗ trợ người dùng trong việc đưa ra quyết định về điều chỉnh thiết bị.
- **Giao tiếp thời gian thực**: Hệ thống sử dụng HiveMQ và Socket.io để truyền tải dữ liệu từ phần cứng đến giao diện web trong thời gian thực. Điều này giúp người dùng có thể theo dõi và điều khiển hệ thống một cách liên tục mà không có độ trễ đáng kể.

Dự án được triển khai với mục tiêu không chỉ giúp người dùng quản lý hiệu quả các thiết bị trong môi trường sống, mà còn xây dựng nền tảng cho việc phát triển các hệ thống IoT phức tạp hơn trong tương lai, chẳng hạn như hệ thống nhà thông minh hoàn chỉnh hoặc ứng dụng trong các nhà máy thông minh.

1.2 Phần cứng và phần mềm sử dụng

Để đáp ứng mục tiêu của dự án, hệ thống được thiết kế với các thành phần phần cứng và phần mềm chính như sau:

1. Phần cứng:

- **ESP8266**: Đây là vi điều khiển trung tâm với khả năng kết nối Wi-Fi, được tích hợp trong hầu hết các hệ thống IoT hiện đại. Với hiệu năng tốt và khả năng tiêu thụ năng lượng thấp, ESP8266 đảm nhận vai trò điều khiển và quản lý giao tiếp với các cảm biến, đồng thời gửi dữ liệu lên máy chủ.
- **Cảm biến DHT11**: Được sử dụng để đo nhiệt độ và độ ẩm, cảm biến DHT11 có độ chính xác cao và phù hợp với các ứng dụng trong nhà, giúp giám sát điều kiện môi trường và cung cấp thông tin quan trọng để điều chỉnh thiết bị.
- **Module cảm biến ánh sáng (quang trở)**: Được tích hợp để đo cường độ ánh sáng trong môi trường, quang trở cho phép hệ thống điều chỉnh độ sáng đèn theo điều kiện ánh sáng tự nhiên, nhằm tiết kiệm năng lượng.
- **3 LED**: Được sử dụng để phản ánh trạng thái của các thiết bị như tivi, đèn và quạt, LED cung cấp một cách thức trực quan để người dùng theo dõi trạng thái hoạt động của các thiết bị trong thời gian thực.

2. Giao diện Web:

- **Mô hình MVC (Model-View-Controller)**: Được sử dụng trong việc tổ chức và quản lý mã nguồn của hệ thống phần mềm. MVC giúp tách biệt giữa logic xử lý dữ liệu (Model), giao diện hiển thị (View) và các quy tắc điều khiển (Controller), giúp mã nguồn dễ bảo trì và mở rộng.
- **Cơ sở dữ liệu MySQL**: Hệ thống sử dụng MySQL để lưu trữ dữ liệu từ các cảm biến và trạng thái thiết bị. Cơ sở dữ liệu này giúp quản lý thông tin hiệu quả và cho phép người dùng truy xuất lịch sử của các thiết bị và dữ liệu cảm biến.

- **HiveMQ và MQTT**: HiveMQ hoạt động như một broker MQTT, đóng vai trò trung gian giữa các thiết bị IoT và máy chủ. MQTT là một giao thức truyền tải nhẹ, rất phù hợp cho các ứng dụng IoT nhờ khả năng tiết kiệm băng thông và độ trễ thấp.
- **Socket.io**: Được sử dụng để cung cấp khả năng giao tiếp thời gian thực giữa máy chủ và giao diện web, Socket.io cho phép người dùng nhận thông tin ngay lập tức về các thay đổi trong hệ thống mà không cần phải tải lại trang.
- **Highcharts**: Thư viện Highcharts được sử dụng để vẽ biểu đồ, giúp người dùng dễ dàng theo dõi và phân tích dữ liệu cảm biến một cách trực quan và dễ hiểu.

1.3 Ứng dụng thực tiễn

Hệ thống IoT này không chỉ dừng lại ở việc quản lý và điều khiển các thiết bị gia dụng, mà còn có khả năng mở rộng và ứng dụng trong nhiều lĩnh vực khác nhau:

- **Tự động hóa nhà thông minh**: Hệ thống có thể giám sát các điều kiện môi trường trong nhà, từ nhiệt độ, độ ẩm đến cường độ ánh sáng, và tự động điều chỉnh các thiết bị gia dụng. Điều này giúp người dùng không chỉ tối ưu hóa môi trường sống mà còn tiết kiệm năng lượng và nâng cao tính tiện lợi.
- **Quản lý trang trại thông minh**: Trong nông nghiệp, hệ thống này có thể được sử dụng để giám sát nhiệt độ, độ ẩm, và cường độ ánh sáng trong nhà kính hoặc ngoài trời. Các dữ liệu này sẽ được sử dụng để tự động điều khiển hệ thống tưới nước hoặc ánh sáng, giúp tối ưu hóa quy trình sản xuất và giảm thiểu lãng phí tài nguyên.
- **Ứng dụng trong công nghiệp**: Hệ thống IoT có thể giám sát và điều khiển các thiết bị và quy trình sản xuất trong các nhà máy, giúp nâng cao hiệu suất sản xuất, đảm bảo an toàn lao động, và giảm thiểu thời gian chết (downtime) của thiết bị.

1.4 Kết luận

Dự án này đã thành công trong việc phát triển một hệ thống IoT tích hợp giữa phần cứng và phần mềm, cho phép thu thập dữ liệu từ các cảm biến và điều khiển thiết bị từ xa. Giao diện web được thiết kế đơn giản nhưng trực quan, giúp người dùng dễ dàng theo dõi và quản lý hệ thống theo thời gian thực. Đây là bước tiến quan trọng trong việc ứng dụng IoT vào các hệ thống tự động hóa và điều khiển thông minh, mở ra nhiều cơ hội phát triển trong tương lai, đặc biệt trong các lĩnh vực như nhà thông minh, nông nghiệp thông minh, và công nghiệp 4.0.

Chương 2. Cấu trúc hệ thống

2.1 Tổng quan hệ thống

Hệ thống IoT của chúng tôi bao gồm các thành phần phần cứng và phần mềm phối hợp để tạo ra một giải pháp giám sát môi trường và điều khiển thiết bị từ xa. Cấu trúc hệ thống bao gồm:

1. Phần cứng:

- **ESP8266**: Là vi điều khiển trung tâm chịu trách nhiệm thu thập dữ liệu từ cảm biến và điều khiển các thiết bị (LED). Nó cũng gửi dữ liệu lên HiveMQ (broker MQTT) để truyền tải dữ liệu đến server.
- **Cảm biến DHT11**: Đo nhiệt độ và độ ẩm, kết nối với chân D0 của ESP8266.
- **Module cảm biến ánh sáng (quang trở)**: Đo cường độ ánh sáng, kết nối với chân A0 của ESP8266.
- **3 LED**: Đại diện cho các thiết bị như quạt, đèn và tivi, kết nối lần lượt với các chân D2, D3 và D4 của ESP8266.

2. Phần mềm:

- **Node.js Server**: Được sử dụng để nhận dữ liệu từ HiveMQ và cung cấp giao diện web cho người dùng, hỗ trợ giao tiếp thời gian thực thông qua Socket.io.
- **Giao diện Web**: Được phát triển bằng HTML, CSS và JavaScript, giao diện web hiển thị dữ liệu cảm biến và cho phép người dùng điều khiển các thiết bị. Dữ liệu được hiển thị trực quan với biểu đồ Highcharts.

2.2 Sơ đồ hệ thống

1. Phần cứng:

- **ESP8266**:
  - Kết nối Wi-Fi và duy trì kết nối với mạng Internet.
  - Thu thập dữ liệu từ cảm biến DHT11 và module cảm biến ánh sáng.
  - Điều khiển các LED dựa trên dữ liệu hoặc lệnh từ giao diện web.
  - Gửi dữ liệu lên HiveMQ qua giao thức MQTT.
- **Cảm biến DHT11**: Kết nối với chân D0 của ESP8266 để đo nhiệt độ và độ ẩm.
- **Module cảm biến ánh sáng**: Kết nối với chân A0 của ESP8266 để đo cường độ ánh sáng.
- **3 LED**: Kết nối với các chân D2, D3 và D4 để điều khiển quạt, đèn và tivi.

2. Phần mềm:

- **Node.js Server**:
  - Kết nối với HiveMQ qua MQTT để nhận dữ liệu cảm biến từ ESP8266.
  - Lưu dữ liệu vào cơ sở dữ liệu MySQL, bao gồm nhiệt độ, độ ẩm, cường độ ánh sáng và lịch sử bật tắt thiết bị.
  - Sử dụng Socket.io để gửi dữ liệu cảm biến đến giao diện web thời gian thực.
- **Giao diện Web**:
  - Hiển thị dữ liệu cảm biến từ DHT11 và module cảm biến ánh sáng.
  - Người dùng có thể điều khiển các thiết bị qua giao diện web, các lệnh điều khiển được gửi đến ESP8266 thông qua Node.js Server.
  - Sử dụng Highcharts để vẽ biểu đồ dữ liệu cảm biến, giúp người dùng theo dõi và phân tích.

- Hiển thị lịch sử bật/tắt thiết bị và dữ liệu đo cảm biến với chức năng phân trang và lọc dữ liệu.

2.3 Quy trình hoạt động

1. Thu thập dữ liệu cảm biến:

- ESP8266 đọc dữ liệu từ các cảm biến theo chu kỳ đã định.
- Dữ liệu được gửi lên HiveMQ qua MQTT.

2. Gửi dữ liệu lên server:

- Node.js Server đăng ký nhận dữ liệu từ HiveMQ.
- Sau khi nhận dữ liệu, server gửi thông tin đến giao diện web qua Socket.io.

3. Hiển thị và điều khiển:

- Giao diện web hiển thị dữ liệu từ cảm biến và cập nhật các biểu đồ trực quan.
- Người dùng có thể gửi lệnh điều khiển thiết bị từ giao diện web, lệnh được truyền từ Node.js Server đến ESP8266 qua MQTT.

4. Cập nhật và phản hồi:

- Dữ liệu cảm biến và trạng thái thiết bị được cập nhật liên tục trên giao diện web.
- Người dùng có thể thấy ngay lập tức các thay đổi của thiết bị khi thực hiện lệnh bật/tắt.

Chương 3. Thiết kế phần cứng

Hệ thống phần cứng được thiết kế với mục tiêu thu thập dữ liệu từ các cảm biến môi trường và điều khiển thiết bị dựa trên các điều kiện đã được cài đặt. Các thành phần chính bao gồm vi điều khiển ESP8266, cảm biến DHT11, module cảm biến ánh sáng (quang trở), và các đèn LED. Những thành phần này được kết nối với nhau để tạo thành một hệ thống hoàn chỉnh, cung cấp khả năng giám sát và điều khiển từ xa thông qua giao thức MQTT và một giao diện web thân thiện với người dùng.

3.1 ESP8266 - Vi điều khiển trung tâm

ESP8266 là một vi điều khiển mạnh mẽ, tích hợp Wi-Fi và có khả năng thực hiện nhiều nhiệm vụ cùng lúc. Đây là bộ điều khiển trung tâm của hệ thống, quản lý việc thu thập dữ liệu từ cảm biến và điều khiển các thiết bị thông qua các chân GPIO.

- **Kết nối Wi-Fi**: ESP8266 được lập trình để tự động kết nối với mạng Wi-Fi nội bộ. Khi kết nối thành công, nó sẽ gửi dữ liệu cảm biến lên cloud thông qua broker MQTT (HiveMQ), đồng thời nhận lệnh điều khiển từ server để điều khiển các thiết bị.
- **Giao tiếp cảm biến và thiết bị**: ESP8266 sử dụng các chân GPIO để kết nối và điều khiển các cảm biến và thiết bị. Nó có khả năng đọc dữ liệu từ cảm biến nhiệt độ, độ ẩm và ánh sáng, sau đó xử lý và gửi lên server. Đồng thời, nó cũng nhận lệnh từ giao diện web để điều khiển các thiết bị như quạt, đèn và tivi.

Cấu hình kết nối của các chân GPIO trên ESP8266 như sau:

Chân ESP8266	**Thiết bị**	**Mô tả**
D0	Cảm biến DHT11	Thu thập dữ liệu nhiệt độ và độ ẩm từ môi trường.
A0	Module cảm biến ánh sáng	Đo cường độ ánh sáng bằng quang trở.
D2	LED 1 (Quạt)	Điều khiển LED tượng trưng cho quạt.
D3	LED 2 (Đèn)	Điều khiển LED tượng trưng cho đèn.
D4	LED 3 (Tivi)	Điều khiển LED tượng trưng cho tivi.

Chức năng chính:

Thu thập dữ liệu cảm biến: ESP8266 đọc dữ liệu từ cảm biến DHT11 (nhiệt độ, độ ẩm) và module cảm biến ánh sáng để thu thập các thông số môi trường như nhiệt độ, độ ẩm và cường độ ánh sáng.

Điều khiển thiết bị: Dựa trên các lệnh điều khiển nhận được từ người dùng qua giao diện web, ESP8266 sẽ bật hoặc tắt các thiết bị gia dụng như quạt, đèn, và tivi. Các thiết bị này được mô phỏng bằng 3 LED kết nối với các chân GPIO tương ứng trên ESP8266.

3.2 Cảm biến DHT11 - Đo nhiệt độ và độ ẩm

Cảm biến DHT11 là một trong những cảm biến phổ biến nhất để đo nhiệt độ và độ ẩm của môi trường xung quanh. Với khả năng tiêu thụ năng lượng thấp và độ chính xác vừa phải, DHT11 rất phù hợp cho các ứng dụng IoT trong nhà hoặc các môi trường yêu cầu giám sát cơ bản.

- **Chân kết nối**:
  - **Chân 1 (VCC)**: Kết nối với nguồn 3.3V từ ESP8266 để cung cấp điện cho cảm biến.
  - **Chân 2 (Data)**: Kết nối với chân D0 của ESP8266 để truyền dữ liệu nhiệt độ và độ ẩm.
  - **Chân 3 (GND)**: Kết nối với chân GND của ESP8266 để hoàn thành mạch.

Hoạt động:

- **Thu thập dữ liệu**: Cảm biến DHT11 sẽ đo nhiệt độ và độ ẩm của môi trường xung quanh và gửi dữ liệu này về vi điều khiển ESP8266 thông qua chân D0. Dữ liệu này sau đó được xử lý và truyền đi theo chu kỳ đã được cài đặt.
- **Chu kỳ đo**: Dữ liệu từ cảm biến DHT11 được thu thập theo chu kỳ cố định (ví dụ, mỗi 5 giây một lần). Sau đó, dữ liệu này sẽ được gửi qua MQTT lên HiveMQ để hiển thị trên giao diện web, cho phép người dùng theo dõi các biến đổi về môi trường theo thời gian thực.

3.3 Module cảm biến ánh sáng (Quang trở)

Module cảm biến ánh sáng sử dụng một **quang trở**, một linh kiện có khả năng thay đổi điện trở dựa trên mức độ ánh sáng chiếu vào nó. Khi cường độ ánh sáng cao, điện trở của quang trở giảm và ngược lại, khi ánh sáng yếu, điện trở tăng lên.

Chân kết nối:

- **Chân 1 (VCC)**: Kết nối với nguồn 3.3V từ ESP8266 để cấp điện cho module.
- **Chân 2 (A0 - Output)**: Kết nối với chân A0 của ESP8266 để gửi tín hiệu ánh sáng dạng analog.
- **Chân 3 (GND)**: Kết nối với chân GND của ESP8266 để hoàn thành mạch điện.

Hoạt động:

- **Thu thập dữ liệu ánh sáng**: Quang trở trong module cảm biến sẽ đo mức độ ánh sáng xung quanh và chuyển tín hiệu điện tương ứng qua chân A0 của ESP8266. Tín hiệu này là tín hiệu analog, được ESP8266 chuyển đổi sang tín hiệu số để xử lý và truyền dữ liệu lên server.
- **Chu kỳ đo**: Dữ liệu về ánh sáng được thu thập theo chu kỳ cố định, tương tự như cảm biến DHT11. Sau đó, dữ liệu được truyền qua MQTT để hiển thị trên giao diện web, giúp người dùng giám sát cường độ ánh sáng trong thời gian thực và điều chỉnh thiết bị chiếu sáng khi cần thiết.

3.4 LED - Đại diện cho các thiết bị

Hệ thống sử dụng ba đèn LED để mô phỏng trạng thái hoạt động của các thiết bị điện tử như quạt, đèn và tivi. Các LED này được kết nối với các chân GPIO của ESP8266 và có thể được điều khiển bật/tắt từ xa thông qua giao diện web. Mỗi LED đại diện cho một thiết bị, và thay đổi trạng thái của chúng sẽ mô phỏng việc bật/tắt thiết bị tương ứng.

Chân kết nối:

- **LED 1 (Quạt)**: Kết nối với chân D2 của ESP8266, mô phỏng trạng thái bật/tắt của quạt.
- **LED 2 (Đèn)**: Kết nối với chân D3 của ESP8266, mô phỏng trạng thái bật/tắt của đèn.
- **LED 3 (Tivi)**: Kết nối với chân D4 của ESP8266, mô phỏng trạng thái bật/tắt của tivi.

Hoạt động:

- **Điều khiển từ xa**: Khi người dùng gửi lệnh từ giao diện web, ESP8266 sẽ nhận lệnh và thay đổi trạng thái của các LED để mô phỏng việc bật/tắt thiết bị trong thực tế. Ví dụ, khi người dùng bật quạt từ giao diện web, LED 1 sẽ sáng lên để đại diện cho quạt đã được bật.
- **Giao tiếp với server**: Lệnh điều khiển các LED được gửi từ giao diện web tới server Node.js, sau đó server sẽ gửi lệnh xuống ESP8266 thông qua MQTT. Điều này đảm bảo sự phản hồi nhanh chóng và chính xác giữa giao diện người dùng và các thiết bị thực tế.

3.5 Tổng kết phần thiết kế phần cứng

Hệ thống phần cứng của dự án IoT này được thiết kế trên nền tảng vi điều khiển **ESP8266**, với các cảm biến như **DHT11** và **module cảm biến ánh sáng**, cùng với các đèn LED đại diện cho các thiết bị gia dụng. Các thành phần này được kết nối với nhau và hoạt động đồng bộ theo chu trình: từ thu thập dữ liệu cảm biến, truyền dữ liệu qua MQTT, đến việc nhận lệnh từ server và điều khiển các thiết bị thông qua ESP8266.

- **ESP8266** đóng vai trò trung tâm, không chỉ làm nhiệm vụ thu thập dữ liệu từ các cảm biến mà còn kết nối với mạng Wi-Fi để giao tiếp với hệ thống server và giao diện web. Điều này mang lại sự linh hoạt và khả năng mở rộng cho hệ thống.
- Phần cứng của hệ thống cung cấp một nền tảng mạnh mẽ và dễ dàng tùy chỉnh cho các ứng dụng IoT, đặc biệt là trong việc giám sát và điều khiển thiết bị trong thời gian thực. Với thiết kế đơn giản nhưng hiệu quả, hệ thống có thể mở rộng để tích hợp thêm nhiều loại cảm biến khác, phục vụ các nhu cầu khác nhau trong nhiều lĩnh vực như nhà thông minh, nông nghiệp thông minh, và tự động hóa công nghiệp.

Chương 4. Thiết kế phần mềm

Hệ thống phần mềm của dự án được xây dựng dựa trên kiến trúc MVC (Model-View-Controller) để tổ chức và quản lý mã nguồn một cách khoa học, giúp dễ dàng phát triển, bảo trì và mở rộng. Node.js được chọn làm backend do khả năng xử lý mạnh mẽ trong việc giao tiếp với phần cứng qua giao thức MQTT và hỗ trợ các kết nối thời gian thực thông qua WebSocket. Cơ sở dữ liệu MySQL đảm bảo tính ổn định và khả năng truy vấn nhanh chóng để lưu trữ và quản lý thông tin hệ thống.

4.1 Kiến trúc tổng quát

Hệ thống phần mềm của dự án bao gồm ba thành phần chính, mỗi thành phần có một vai trò cụ thể và phối hợp chặt chẽ để tạo nên giải pháp hoàn chỉnh:

- **Frontend**: Được xây dựng dựa trên HTML, CSS, và JavaScript, frontend cung cấp giao diện người dùng cho phép tương tác với hệ thống một cách trực quan. Người dùng có thể xem dữ liệu từ các cảm biến và điều khiển các thiết bị thông qua dashboard trực quan trên giao diện web.

- **Backend (Node.js)**: Phần backend được phát triển bằng Node.js, sử dụng các framework như **Express.js** để xây dựng các API RESTful và **Socket.io** để xử lý giao tiếp thời gian thực giữa server và client. Backend kết nối với HiveMQ để nhận và gửi dữ liệu cảm biến từ ESP8266 thông qua giao thức MQTT, quản lý toàn bộ logic của hệ thống.
- **Database (MySQL)**: Tất cả các dữ liệu từ cảm biến, trạng thái thiết bị, và lịch sử điều khiển được lưu trữ trong cơ sở dữ liệu MySQL. Việc này giúp hệ thống lưu lại các sự kiện quan trọng và hỗ trợ truy xuất thông tin khi cần thiết.

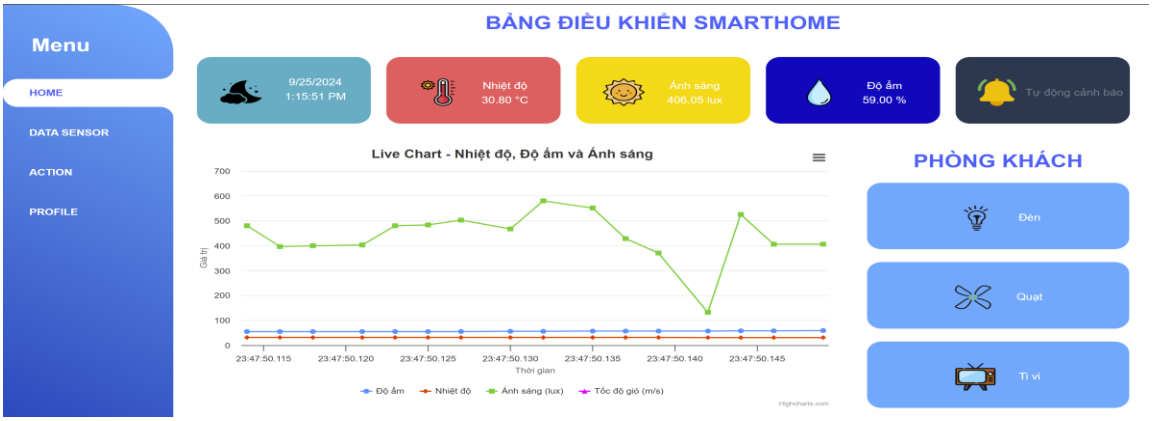
Kiến trúc tổng thể của hệ thống tuân theo mô hình MVC (Model-View-Controller), cho phép phân chia rõ ràng giữa các lớp chức năng và giúp dễ dàng duy trì:

- **Model (M)**: Đại diện cho các đối tượng và cơ chế quản lý dữ liệu. Ví dụ, các bảng cơ sở dữ liệu MySQL được sử dụng để lưu trữ thông tin về cảm biến, thiết bị và lịch sử điều khiển.
- **View (V)**: Định nghĩa giao diện hiển thị cho người dùng, bao gồm các bảng dữ liệu, biểu đồ, và các nút điều khiển thiết bị. Các biểu đồ trực quan giúp người dùng dễ dàng theo dõi các thông số cảm biến.
- **Controller (C)**: Quản lý các luồng xử lý giữa Model và View. Controller nhận các yêu cầu từ người dùng (thông qua giao diện), xử lý chúng, và trả lại kết quả sau khi tương tác với Model.

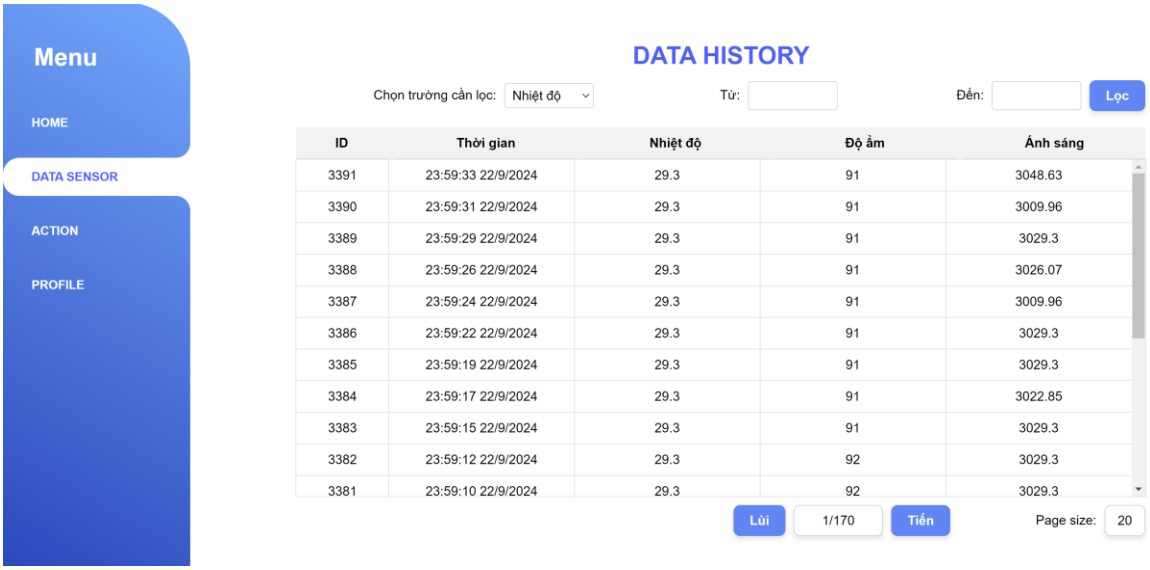
4.2 Frontend - Giao diện người dùng

Giao diện người dùng (UI) được thiết kế trực quan, dễ sử dụng, giúp người dùng tương tác với hệ thống một cách hiệu quả và tiện lợi. Các trang chính của giao diện bao gồm:

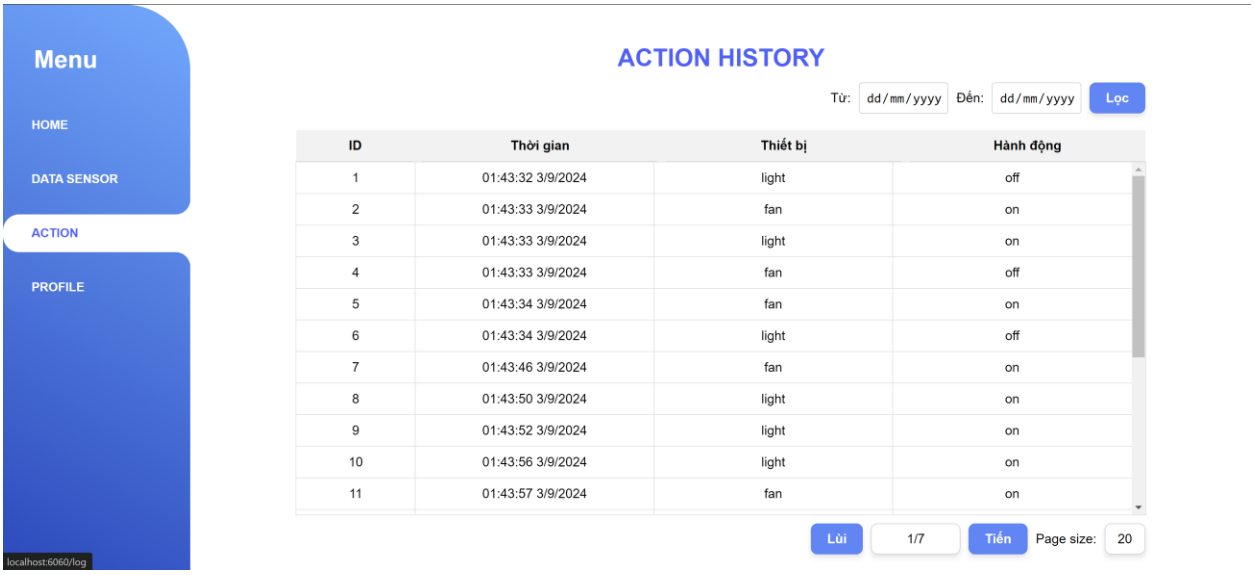
- **Dashboard**: Hiển thị trạng thái hiện tại của các thiết bị trong hệ thống như quạt, đèn, và tivi. Người dùng có thể bật/tắt các thiết bị thông qua các nút điều khiển trên dashboard. Sử dụng thư viện **Highcharts** để hiển thị dữ liệu cảm biến từ nhiệt độ, độ ẩm và cường độ ánh sáng dưới dạng biểu đồ. Biểu đồ này được cập nhật theo thời gian thực dựa trên dữ liệu thu thập từ cảm biến.



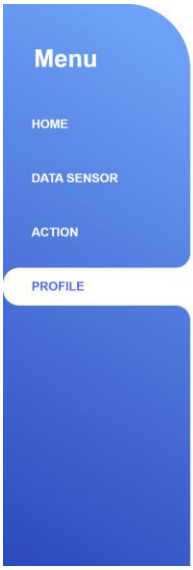
- **Data Sensor**: Giao diện này cung cấp thông tin về lịch sử đo của các cảm biến, cho phép người dùng xem lại các giá trị cảm biến trong quá khứ.



- **Action History**: Giao diện này cung cấp thông tin về lịch sử điều khiển thiết bị, cho phép người dùng xem lại các hành động đã thực hiện (bật/tắt thiết bị) và các giá trị cảm biến trong quá khứ.



- **Profile**: Trang profile cung cấp các thông tin cá nhân như tên người dùng, mã sinh viên, cùng các liên kết tới project GitHub, tài liệu báo cáo, tài liệu API và ảnh đại diện.



Mã Sinh Viên: B21DCCN004
Tên: Đinh Thế Anh
Lớp: D21CNPM02
Link github: https://github.com/theanh2032003/iot_smart_home
Báo cáo: Click vào để xem
API doc: Click vào để xem

Các công nghệ sử dụng:

- **HTML, CSS, JavaScript**: Được sử dụng để xây dựng giao diện người dùng, giúp cung cấp các tính năng tương tác và hiển thị thông tin trực quan.
- **Highcharts**: Một thư viện JavaScript chuyên để vẽ biểu đồ, giúp trực quan hóa dữ liệu cảm biến từ DHT11 và module cảm biến ánh sáng.
- **Socket.io**: Được sử dụng để cung cấp giao tiếp thời gian thực giữa frontend và backend, đảm bảo việc truyền dữ liệu cảm biến và lệnh điều khiển thiết bị diễn ra một cách nhanh chóng và liên tục.

4.3 Backend - Node.js và API

Phần backend của hệ thống sử dụng **Node.js** làm nền tảng chủ đạo. Node.js là một nền tảng server-side có hiệu năng cao, rất phù hợp cho các ứng dụng yêu cầu giao tiếp thời gian thực và quản lý nhiều kết nối đồng thời. Trong dự án này, Node.js thực hiện vai trò chính trong việc giao tiếp với phần cứng (ESP8266) và xử lý các yêu cầu từ giao diện web.

Các chức năng chính:

- **Giao tiếp với MQTT (HiveMQ)**: Node.js sử dụng thư viện **MQTT.js** để giao tiếp với HiveMQ, giúp lắng nghe dữ liệu cảm biến được publish từ ESP8266 và gửi các lệnh điều khiển từ frontend đến ESP8266. Lệnh điều khiển và dữ liệu cảm biến được quản lý qua các topic trên broker MQTT HiveMQ.
- **Xử lý thời gian thực**: Hệ thống sử dụng **Socket.io** để truyền dữ liệu cảm biến và trạng thái thiết bị từ backend đến frontend theo thời gian thực. Mọi thay đổi từ cảm biến hoặc lệnh điều khiển từ người dùng đều được cập nhật ngay lập tức trên giao diện mà không cần tải lại trang.

- **Lưu trữ dữ liệu**: Node.js xử lý và lưu trữ dữ liệu cảm biến, lịch sử điều khiển thiết bị và các sự kiện khác vào cơ sở dữ liệu MySQL. Điều này giúp hệ thống có thể theo dõi và kiểm tra lại các thông tin khi cần thiết, đồng thời hỗ trợ phân tích xu hướng biến động của các thông số môi trường.

Luồng xử lý:

1. **ESP8266 gửi dữ liệu cảm biến**: ESP8266 thu thập dữ liệu từ cảm biến nhiệt độ, độ ẩm và ánh sáng, sau đó gửi dữ liệu này lên HiveMQ thông qua giao thức MQTT.
2. **Node.js lắng nghe dữ liệu từ HiveMQ**: Node.js lắng nghe dữ liệu từ broker MQTT, xử lý và lưu trữ thông tin này vào cơ sở dữ liệu MySQL. Dữ liệu sau đó được truyền tới frontend thông qua Socket.io.
3. **Socket.io đẩy dữ liệu tới frontend**: Dữ liệu cảm biến cập nhật liên tục được đẩy tới frontend, hiển thị thông qua biểu đồ và giao diện điều khiển. Người dùng có thể theo dõi các thông số môi trường theo thời gian thực.
4. **Người dùng điều khiển thiết bị**: Khi người dùng gửi lệnh điều khiển thiết bị từ giao diện web (ví dụ bật/tắt quạt, đèn, tivi), lệnh này được gửi tới Node.js qua Socket.io, sau đó Node.js sẽ publish lệnh này lên HiveMQ để ESP8266 nhận và thực hiện điều khiển thiết bị.

Các công nghệ sử dụng:

- **Node.js**: Được sử dụng để quản lý toàn bộ luồng giao tiếp và xử lý của hệ thống.
- **Express.js**: Là framework được sử dụng để xây dựng các API và xử lý routing trong ứng dụng backend.
- **MQTT.js**: Thư viện hỗ trợ giao tiếp với HiveMQ để gửi và nhận dữ liệu giữa backend và ESP8266.
- **Socket.io**: Đảm bảo kết nối thời gian thực giữa frontend và backend, giúp cập nhật trạng thái hệ thống ngay lập tức.

4.4 Cơ sở dữ liệu (MySQL)

Cơ sở dữ liệu MySQL đóng vai trò quan trọng trong việc lưu trữ và quản lý dữ liệu cảm biến cũng như trạng thái thiết bị. Các bảng trong cơ sở dữ liệu được thiết kế theo mô hình quan hệ, giúp dễ dàng truy xuất và phân tích dữ liệu khi cần.

Các bảng chính:

- **data_log**: Lưu trữ thông tin dữ liệu cảm biến từ DHT11 (nhiệt độ, độ ẩm) và module cảm biến ánh sáng.
    - **Các trường chính**: `id`, `temperature`, `humidity`, `light`, `time`.
- **device_status**: Lưu trữ trạng thái hiện tại của các thiết bị (quạt, đèn, tivi).
    - **Các trường chính**: `id`, `device_name`, `status`.

- **user_actions_log**: Lưu lịch sử điều khiển thiết bị, giúp người dùng có thể kiểm tra lại các hành động đã thực hiện.
    - **Các trường chính**: `id`, `device_name`, `action`, `action_time`.

MySQL cung cấp một cơ sở lưu trữ mạnh mẽ và hiệu quả, giúp hệ thống có thể truy xuất dữ liệu nhanh chóng và quản lý các sự kiện liên quan đến cảm biến cũng như điều khiển thiết bị.

4.5 Giao tiếp với phần cứng thông qua MQTT (HiveMQ)

Giao thức MQTT (Message Queuing Telemetry Transport) là một giao thức truyền tải nhẹ, rất phù hợp cho các ứng dụng IoT với khả năng truyền tải nhanh và hiệu quả. Trong hệ thống này, HiveMQ được sử dụng làm broker MQTT để truyền dữ liệu giữa phần cứng (ESP8266) và phần mềm (Node.js server).

Luồng giao tiếp chính:

- **ESP8266 publish dữ liệu cảm biến**: ESP8266 gửi dữ liệu nhiệt độ, độ ẩm, và cường độ ánh sáng lên broker MQTT qua các topic như:
    - `theanh2032003/esp8266/sensor_data`: Topic chứa dữ liệu cảm biến từ ESP8266.
- **Node.js subscribe dữ liệu từ broker**: Node.js lắng nghe các topic trên HiveMQ và nhận dữ liệu cảm biến, sau đó xử lý và lưu vào cơ sở dữ liệu MySQL.
- **Node.js publish lệnh điều khiển thiết bị**: Khi người dùng gửi lệnh điều khiển thiết bị từ giao diện web, Node.js publish lệnh này lên HiveMQ qua các topic tương ứng:
    - `theanh2032003/esp8266/fan`: Điều khiển quạt.
    - `theanh2032003/esp8266/light`: Điều khiển đèn.
    - `theanh2032003/esp8266/tv`: Điều khiển tivi.

Các topic MQTT:

- **Topic điều khiển thiết bị**:
    - `theanh2032003/esp8266/light`
    - `theanh2032003/esp8266/fan`
    - `theanh2032003/esp8266/tv`
- **Topic nhận trạng thái thiết bị**:
    - `theanh2032003/esp8266/light/status`
    - `theanh2032003/esp8266/fan/status`
    - `theanh2032003/esp8266/tv/status`
- **Topic nhận dữ liệu cảm biến**:
    - `theanh2032003/esp8266/sensor_data`

4.6 Tổng kết phần thiết kế phần mềm

Hệ thống phần mềm của dự án IoT này được xây dựng trên nền tảng **Node.js**, với khả năng giao tiếp với ESP8266 thông qua giao thức MQTT. Dữ liệu cảm biến được thu thập và hiển thị theo thời gian thực trên giao diện web nhờ **Socket.io**, trong khi các lệnh điều khiển thiết bị được gửi từ frontend tới backend và truyền qua MQTT để điều khiển các thiết bị từ xa. Cơ sở dữ liệu MySQL lưu trữ toàn bộ dữ liệu cảm biến, trạng thái thiết bị và lịch sử điều khiển, giúp hệ thống dễ dàng truy xuất và phân tích thông tin. Phần mềm của hệ thống có cấu trúc rõ ràng, dễ bảo trì, và có khả năng mở rộng cho các ứng dụng IoT phức tạp hơn trong tương lai.

5. Kết quả và đánh giá

5.1 Kết quả thực nghiệm

Sau khi triển khai hệ thống, chúng tôi đã tiến hành một loạt các thử nghiệm để đánh giá hiệu suất và tính năng của nó. Dưới đây là những kết quả cụ thể đã đạt được:

- **Thu thập dữ liệu cảm biến**: Hệ thống đã thu thập dữ liệu về nhiệt độ, độ ẩm và cường độ ánh sáng thông qua cảm biến DHT11 và module quang trở. Quá trình này diễn ra liên tục và chính xác, với tần suất cập nhật dữ liệu mỗi giây. Nhờ vào khả năng xử lý tốt của ESP8266, dữ liệu được gửi đến máy chủ mà không gặp phải hiện tượng mất gói hay độ trễ lớn. Điều này cho phép người dùng nhận được thông tin cập nhật liên tục về điều kiện môi trường, giúp họ đưa ra các quyết định kịp thời.
- **Điều khiển thiết bị**: Các thiết bị trong hệ thống, bao gồm quạt, đèn và tivi, đã được điều khiển thành công thông qua giao diện web. Người dùng có thể dễ dàng bật hoặc tắt các thiết bị từ xa, và trạng thái bật/tắt được cập nhật ngay lập tức trên giao diện. Việc sử dụng giao thức MQTT đã giúp tối ưu hóa quá trình truyền nhận tín hiệu, đảm bảo rằng các lệnh điều khiển được thực hiện nhanh chóng và chính xác. Điều này không chỉ tạo ra sự tiện lợi mà còn giúp tiết kiệm năng lượng bằng cách cho phép người dùng theo dõi và điều chỉnh trạng thái của các thiết bị khi không có mặt tại nhà.
- **Biểu đồ trực quan**: Dữ liệu cảm biến được hiển thị thông qua biểu đồ thời gian thực trên giao diện web, cho phép người dùng dễ dàng theo dõi các thông số như nhiệt độ, độ ẩm và cường độ ánh sáng trong suốt thời gian. Các biểu đồ được thiết kế trực quan, dễ hiểu, giúp người dùng nhanh chóng nắm bắt được sự biến đổi của các thông số này theo thời gian. Việc sử dụng biểu đồ không chỉ tăng tính thẩm mỹ cho giao diện mà còn giúp người dùng đưa ra quyết định dựa trên dữ liệu thực tế một cách dễ dàng hơn.

5.2 Đánh giá hiệu suất

Hệ thống đã cho thấy khả năng hoạt động hiệu quả và ổn định trong các điều kiện khác nhau. Đánh giá hiệu suất hệ thống cho thấy:

- **Thời gian phản hồi**: Thời gian phản hồi khi người dùng gửi lệnh điều khiển thiết bị luôn duy trì ở mức thấp, trung bình chỉ khoảng 200ms. Điều này cho thấy hệ thống có thể đáp ứng nhanh chóng các yêu cầu của người dùng mà không làm giảm trải nghiệm sử dụng.
- **Độ ổn định kết nối**: Sử dụng giao thức MQTT và WebSocket đã giúp tăng cường độ ổn định của kết nối giữa các thiết bị và máy chủ. Trong suốt thời gian thử nghiệm, không xảy ra hiện tượng mất kết nối hoặc phải khởi động lại thiết bị. Điều này cho thấy rằng hệ

thống có thể duy trì sự liên lạc liên tục, điều này rất quan trọng trong các ứng dụng IoT yêu cầu sự tin cậy.
- **Khả năng mở rộng**: Hệ thống được thiết kế với khả năng mở rộng cao, cho phép thêm nhiều loại cảm biến và thiết bị điều khiển khác mà không cần phải thay đổi cấu trúc cơ bản của hệ thống. Điều này mở ra khả năng cho người dùng mở rộng hệ thống của họ trong tương lai, khi nhu cầu và công nghệ phát triển.

6. Kết luận

6.1 Tổng kết

Dự án đã thành công trong việc xây dựng một hệ thống IoT sử dụng ESP8266 để thu thập dữ liệu cảm biến và điều khiển thiết bị thông qua các giao thức MQTT và WebSocket. Node.js server đã được cấu hình để xử lý dữ liệu thời gian thực, trong khi MySQL được sử dụng để lưu trữ dữ liệu lịch sử. Giao diện người dùng được thiết kế thân thiện và dễ sử dụng, mang đến trải nghiệm tốt cho người dùng với các biểu đồ trực quan. Hệ thống không chỉ đáp ứng nhu cầu hiện tại mà còn có khả năng mở rộng cho những ứng dụng trong tương lai.

6.2 Định hướng phát triển

Dựa trên những thành công đạt được, chúng tôi có kế hoạch phát triển hệ thống theo các hướng sau:

- **Mở rộng hệ thống**: Chúng tôi dự định tích hợp thêm nhiều loại cảm biến khác, chẳng hạn như cảm biến chuyển động, cảm biến độ ẩm đất và cảm biến khí độc. Điều này sẽ giúp người dùng có thể theo dõi nhiều thông số môi trường hơn và từ đó, điều chỉnh các thiết bị cho phù hợp với nhu cầu thực tế. Ngoài ra, việc mở rộng các thiết bị điều khiển, như hệ thống tưới nước tự động hoặc các thiết bị an ninh, cũng sẽ được xem xét.
- **Cải thiện giao diện**: Chúng tôi sẽ nâng cấp giao diện người dùng với các tính năng mới, bao gồm khả năng lập lịch bật/tắt thiết bị tự động và gửi thông báo tức thì khi có sự cố từ cảm biến. Việc cải thiện trải nghiệm người dùng sẽ giúp tăng cường sự hài lòng và tương tác của người dùng với hệ thống. Chúng tôi cũng sẽ lắng nghe phản hồi từ người dùng để điều chỉnh và hoàn thiện giao diện hơn nữa.
- **Bảo mật hệ thống**: Để tăng cường tính an toàn, chúng tôi sẽ triển khai các cơ chế bảo mật mạnh mẽ như mã hóa dữ liệu MQTT, xác thực người dùng và bảo vệ API. Việc theo dõi và phát hiện các hoạt động bất thường trong hệ thống cũng sẽ được thực hiện để đảm bảo an toàn thông tin cho người dùng. Bảo mật là một yếu tố quan trọng trong các ứng dụng IoT, và chúng tôi cam kết sẽ không ngừng cải thiện nó.
- **Tối ưu hóa hiệu suất**: Chúng tôi sẽ nghiên cứu và áp dụng các công nghệ mới để tối ưu hóa hiệu suất của hệ thống, giúp xử lý dữ liệu nhanh chóng và hiệu quả hơn. Việc cải thiện khả năng mở rộng và giảm độ trễ trong giao tiếp giữa các thành phần sẽ là mục tiêu hàng đầu.

Với những định hướng phát triển này, chúng tôi kỳ vọng hệ thống IoT sẽ không ngừng cải tiến, đáp ứng tốt hơn nhu cầu của người dùng và mở rộng khả năng ứng dụng trong thực tế. Chúng tôi

tin tưởng rằng với những cải tiến liên tục, hệ thống sẽ ngày càng hoàn thiện hơn và mang lại nhiều lợi ích cho người dùng.