

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

Lời cảm ơn	1
1. Giới thiệu	4
1.1 Mục tiêu của dự án	4
1.2 Phần cứng và phần mềm sử dụng.....	5
1.3 Ứng dụng thực tiễn	5
1.4 Kết luận	5
2. Cấu trúc hệ thống	6
2.1 Tổng quan hệ thống	6
2.2 Sơ đồ hệ thống.....	6
2.3 Quy trình hoạt động.....	7
3. Thiết kế phần cứng	7
3.1 ESP8266 - Vi điều khiển trung tâm.....	7
3.2 Cảm biến DHT11 - Đo nhiệt độ và độ ẩm.....	9
3.3 Module cảm biến ánh sáng (Quang trở)	11
3.4 LED - Đại diện cho các thiết bị.....	12
3.5 Tổng kết phần thiết kế phần cứng	13
4. Thiết kế phần mềm	13
4.1 Kiến trúc tổng quát	13
4.2 Frontend - Giao diện người dùng.....	14
4.3 Backend - Node.js và API.....	16
4.4 Cơ sở dữ liệu (MySQL)	17
4.5 Giao tiếp với phần cứng thông qua MQTT (HiveMQ)	17
4.6 Tổng kết phần thiết kế phần mềm.....	18
5. Kết quả và đánh giá	18
5.1 Kết quả thực nghiệm.....	18
5.2 Đánh giá hiệu suất.....	18
6. Kết luận	18
6.1 Tổng kết	18
6.2 Định hướng phát triển	19

1. Giới thiệu

Trong bối cảnh cuộc cách mạng công nghệ 4.0 hiện nay, Internet of Things (IoT) đang trở thành một xu hướng nổi bật, đóng vai trò quan trọng trong việc kết nối các thiết bị thông minh và nâng cao hiệu quả trong nhiều lĩnh vực. Dự án của chúng tôi tập trung vào việc phát triển một hệ thống IoT đơn giản nhưng mạnh mẽ, sử dụng ESP8266 cùng với các cảm biến và module để điều khiển thiết bị và thu thập dữ liệu từ môi trường. Hệ thống cũng cung cấp giao diện web để quản lý và hiển thị thông tin một cách trực quan và hiệu quả.

1.1 Mục tiêu của dự án

Mục tiêu chính của dự án là tạo ra một hệ thống IoT có khả năng:

- **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 môi trường.
- **Điều khiển thiết bị từ xa:** Kiểm soát các thiết bị gia dụng như quạt, đèn, và tivi thông qua giao tiếp MQTT, với LED tượng trưng cho trạng thái của thiết bị.
- **Hiển thị thông tin:** Giao diện web được xây dựng với Node.js, giúp người dùng giám sát dữ liệu cảm biến theo thời gian thực và xem các biểu đồ trực quan bằng Highcharts.
- **Giao tiếp thời gian thực:** Sử dụng HiveMQ và Socket.io để đảm bảo tính liên tục trong trao đổi dữ liệu giữa phần cứng và giao diện web.

1.2 Phần cứng và phần mềm sử dụng

Hệ thống được thiết kế để thực hiện các chức năng cơ bản nhưng thiết yếu trong môi trường IoT:

1. Phần cứng:

- **ESP8266:** Vi điều khiển với khả năng kết nối Wi-Fi tích hợp, làm trung tâm giao tiếp với các cảm biến và gửi dữ liệu lên máy chủ.
- **Cảm biến DHT11:** Đo nhiệt độ và độ ẩm, cung cấp thông tin quan trọng về điều kiện môi trường.
- **Module cảm biến ánh sáng (quang trở):** Đo cường độ ánh sáng môi trường, cho phép hệ thống điều chỉnh độ sáng của các thiết bị như đèn.
- **3 LED:** Phản ánh trạng thái của các thiết bị như tivi, đèn, quạt; có thể được điều khiển để báo hiệu trạng thái bật/tắt.

2. Giao diện Web:

- **Mô hình MVC (Model-View-Controller) với Node.js:** Tổ chức mã nguồn rõ ràng và dễ bảo trì.
- **Cơ sở dữ liệu MySQL:** Lưu trữ dữ liệu thu thập từ cảm biến và trạng thái thiết bị, cho phép quản lý hiệu quả.
- **Giao tiếp với phần cứng qua HiveMQ:** Broker MQTT cung cấp khả năng giao tiếp giữa các thiết bị IoT và máy chủ.
- **Giao tiếp thời gian thực qua Socket.io:** Kết nối thời gian thực giữa máy chủ và giao diện web, cho phép nhận dữ liệu ngay lập tức.
- **Vẽ biểu đồ bằng Highcharts:** Hiển thị dữ liệu từ cảm biến một cách trực quan và dễ hiểu.

1.3 Ứng dụng thực tiễn

Hệ thống này có thể được ứng dụng rộng rãi trong các lĩnh vực như:

- **Tự động hóa nhà thông minh:** Giám sát điều kiện môi trường trong nhà và điều khiển thiết bị gia dụng từ xa, giúp tiết kiệm năng lượng và tăng tính tiện lợi.
- **Quản lý trang trại thông minh:** Giám sát nhiệt độ, độ ẩm và ánh sáng trong nông nghiệp, điều chỉnh hệ thống tưới nước và ánh sáng để tối ưu hóa quá trình sản xuất.
- **Ứng dụng công nghiệp:** Giám sát và điều khiển các thiết bị và quy trình sản xuất trong nhà máy, nâng cao hiệu suất và an toàn lao động.

1.4 Kết luận

Dự án đã thành công trong việc xây dựng 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 dễ sử dụng giúp người dùng dễ dàng theo dõi và quản lý hệ thống. 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.

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.

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 được đặt ra. Dưới đây là chi tiết thiết kế phần cứng cho dự án, bao gồm các thành phần chính như ESP8266, cảm biến DHT11, module cảm biến ánh sáng (quang trở), và các đèn LED.

3.1 ESP8266 - Vi điều khiển trung tâm



ESP8266 là vi điều khiển được sử dụng trong hệ thống để quản lý việc thu thập dữ liệu cảm biến và điều khiển thiết bị. Nó có khả năng kết nối Wi-Fi và giao tiếp với các thiết bị khác qua các chân GPIO.

- **Kết nối Wi-Fi:** ESP8266 được lập trình để kết nối với mạng Wi-Fi nội bộ, cho phép nó truyền dữ liệu lên cloud và nhận lệnh điều khiển từ giao diện web thông qua MQTT (HiveMQ).
- **Giao tiếp cảm biến và thiết bị:** ESP8266 sử dụng các chân GPIO để thu thập dữ liệu từ cảm biến và điều khiển các thiết bị (LED).

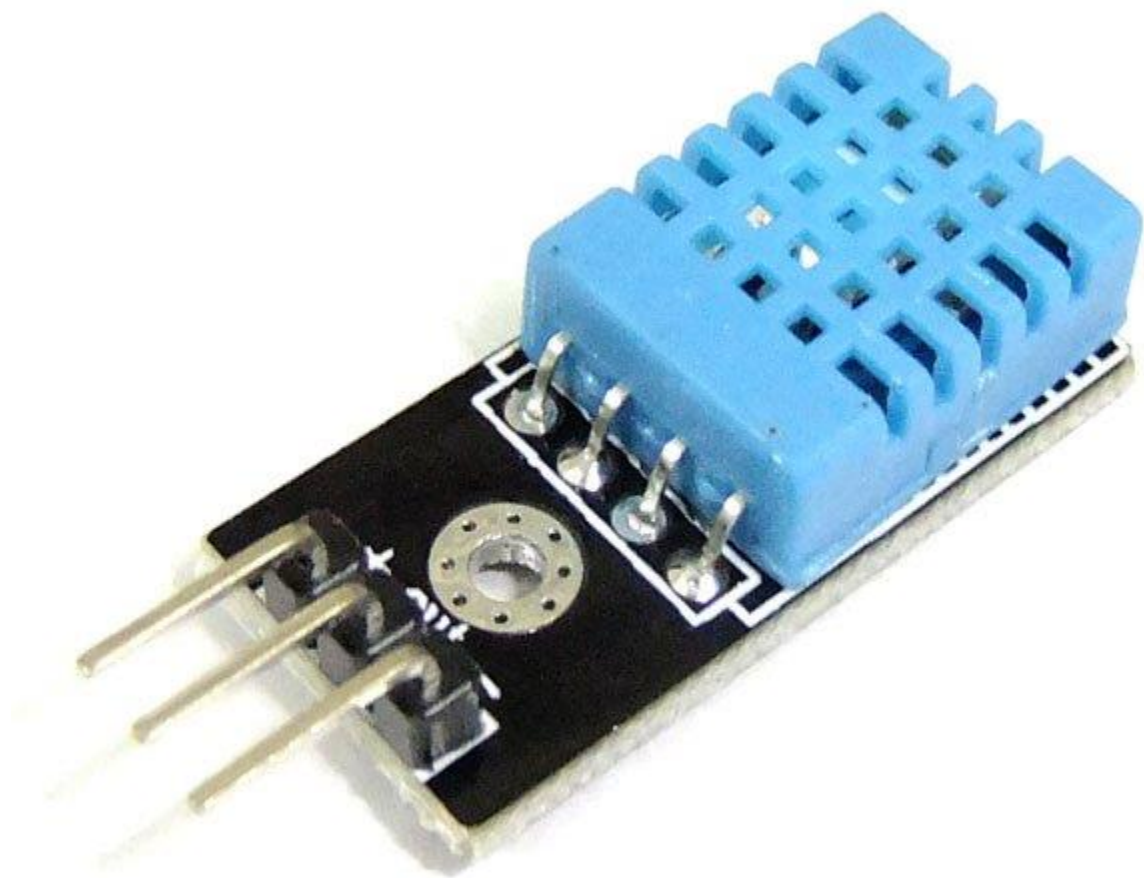
Cấu hình kết nối của các chân ESP8266 như sau:

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 các cảm biến DHT11 và module cảm biến ánh sáng để lấy thông tin về nhiệt độ, độ ẩm và cường độ ánh sáng.
- **Điều khiển thiết bị:** Các thiết bị như quạt, đèn, và tivi được điều khiển thông qua các chân D2, D3, và D4, tương ứng với 3 LED đại diện cho các thiết bị này.

3.2 Cảm biến DHT11 - Đo nhiệt độ và độ ẩm



Cảm biến DHT11 được sử dụng để đo nhiệt độ và độ ẩm của môi trường xung quanh. Đây là loại cảm biến phổ biến, dễ sử dụng, có độ chính xác tương đối cao và tiêu thụ ít điện năng.

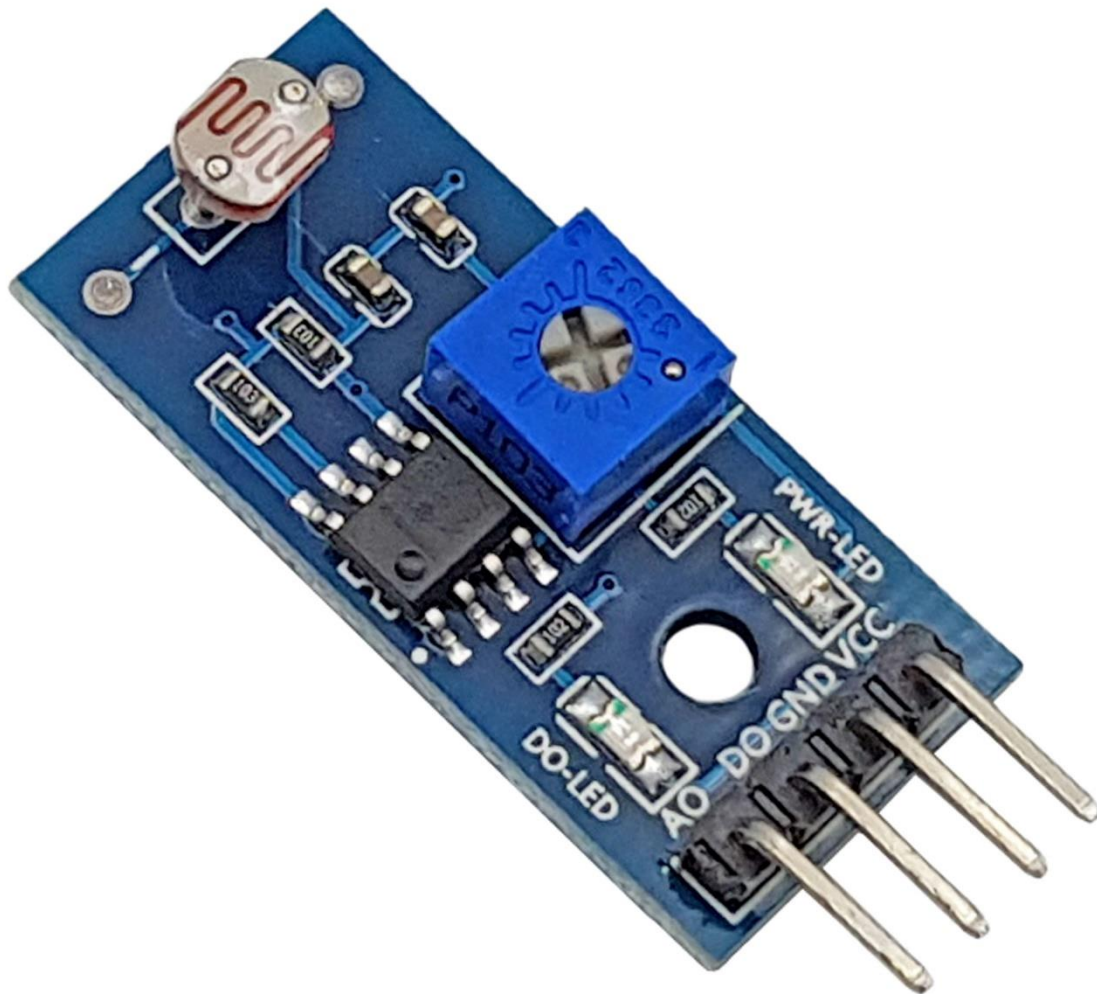
- **Chân kết nối:**
 - **Chân 1 (VCC):** Kết nối với nguồn 3.3V từ ESP8266.
 - **Chân 2 (Data):** Kết nối với chân D0 của ESP8266 để truyền dữ liệu.
 - **Chân 3 (GND):** Kết nối với chân GND của ESP8266.

Hoạt động:

- **Thu thập dữ liệu:** Cảm biến DHT11 gửi dữ liệu nhiệt độ và độ ẩm về vi điều khiển ESP8266 thông qua chân D0.

- **Chu kỳ đo:** Dữ liệu từ cảm biến được thu thập với chu kỳ cố định (ví dụ, mỗi 5 giây một lần) và được truyền qua giao thức MQTT lên HiveMQ để hiển thị trên giao diện web.

3.3 Module cảm biến ánh sáng (Quang trở)



Module cảm biến ánh sáng sử dụng một quang trở để đo cường độ ánh sáng trong môi trường. Quang trở là một linh kiện có điện trở thay đổi theo lượng ánh sáng chiếu vào nó; cường độ ánh sáng càng cao, điện trở càng giảm và ngược lại.

- **Chân kết nối:**
 - **Chân 1 (VCC):** Kết nối với nguồn 3.3V từ ESP8266.

- **Chân 2 (A0 - Output):** Kết nối với chân A0 của ESP8266 để gửi tín hiệu cường độ ánh sáng dạng analog.
- **Chân 3 (GND):** Kết nối với chân GND của ESP8266.

Hoạt động:

- **Thu thập dữ liệu ánh sáng:** Quang trở trong module sẽ đo mức độ ánh sáng xung quanh và truyền dữ liệu tương ứng qua chân A0 của ESP8266. Dữ liệu này được chuyển đổi từ tín hiệu analog sang số để xử lý trong ESP8266.
- **Chu kỳ đo:** Tương tự như DHT11, dữ liệu ánh sáng cũng được thu thập theo chu kỳ định sẵn và được gửi lên server thông qua MQTT.

3.4 LED - Đại diện cho các thiết bị



Hệ thống sử dụng 3 LED để mô phỏng các thiết bị điện tử có thể điều khiển từ xa như quạt, đèn, và tivi. Mỗi LED được kết nối với một chân GPIO của ESP8266 và có thể bật/tắt dựa trên các lệnh điều khiển từ giao diện web.

- **Chân kết nối:**
 - **LED 1 (Quạt):** Kết nối với chân D2 của ESP8266.
 - **LED 2 (Đèn):** Kết nối với chân D3 của ESP8266.
 - **LED 3 (Tivi):** Kết nối với chân D4 của ESP8266.

Hoạt động:

- **Điều khiển từ xa:** Người dùng có thể điều khiển bật/tắt các LED từ giao diện web. Khi nhận lệnh từ giao diện, ESP8266 sẽ thay đổi trạng thái của các LED để mô phỏng việc điều khiển các thiết bị trong thực tế.
- **Giao tiếp với server:** Các lệnh điều khiển LED được gửi từ giao diện web tới server Node.js, sau đó server sẽ gửi lệnh xuống ESP8266 thông qua MQTT để thực hiện.

3.5 Tổng kết phần thiết kế phần cứng

Hệ thống phần cứng của dự án IoT này được xây dựng dựa trên vi điều khiển ESP8266 với các cảm biến DHT11, module cảm biến ánh sáng, và 3 LED đại diện cho các thiết bị điều khiển. Tất cả các thành phần này đều được kết nối và hoạt động theo chu trình: thu thập dữ liệu cảm biến, gửi dữ liệu qua MQTT, nhận lệnh từ server Node.js, và thực hiện điều khiển thiết bị thông qua các LED. ESP8266 đóng vai trò trung tâm, vừa làm nhiệm vụ giao tiếp với các cảm biến và thiết bị, vừa kết nối mạng để giao tiếp với hệ thống server và giao diện web.

Phần cứng này cung cấp một nền tảng mạnh mẽ và linh hoạt cho các ứng dụng IoT, đặc biệt là trong việc điều khiển và giám sát môi trường theo thời gian thực.

4. Thiết kế phần mềm

Hệ thống phần mềm của dự án này được xây dựng dựa trên kiến trúc MVC (Model-View-Controller) với việc sử dụng Node.js làm backend, giao tiếp với phần cứng thông qua HiveMQ và xử lý dữ liệu cảm biến theo thời gian thực bằng WebSocket. Cơ sở dữ liệu MySQL được sử dụng để lưu trữ và quản lý thông tin hệ thống. Dưới đây là chi tiết thiết kế phần mềm.

4.1 Kiến trúc tổng quát

Hệ thống phần mềm gồm ba thành phần chính:

- **Frontend:** Được xây dựng dựa trên HTML, CSS và JavaScript để cung cấp giao diện cho người dùng tương tác với hệ thống. Người dùng có thể xem dữ liệu từ cảm biến và điều khiển các thiết bị thông qua giao diện này.
- **Backend (Node.js):** Phần backend được phát triển bằng Node.js, sử dụng các framework như Express.js để xây dựng API và Socket.io để xử lý giao tiếp thời gian thực giữa server

- **Database (MySQL):** Dữ liệu về các thông tin từ cảm biến, trạng thái của thiết bị và lịch sử điều khiển được lưu trữ trong cơ sở dữ liệu MySQL.

- **Model (M):** Đại diện cho các đối tượng và cơ chế quản lý dữ liệu, như các bảng trong cơ sở dữ liệu MySQL.
- **View (V):** Định nghĩa giao diện hiển thị cho người dùng, bao gồm các biểu đồ và giao diện điều khiển thiết bị.
- **Controller (C):** Quản lý các luồng xử lý giữa Model và View, nhận các yêu cầu từ người dùng và trả kết quả sau khi xử lý.

Giao diện web được thiết kế để người dùng có thể tương tác với hệ thống một cách trực quan. Các trang chính bao gồm:

- Menu


HOME

DATA SENSOR


ACTION

PROFILE


BẢNG ĐIỀU KHIỂN SMARTHOME




9/22/2024
11:49:21 PM




Nhiệt độ
29.30 °C



Ánh sáng
2091.50 lux




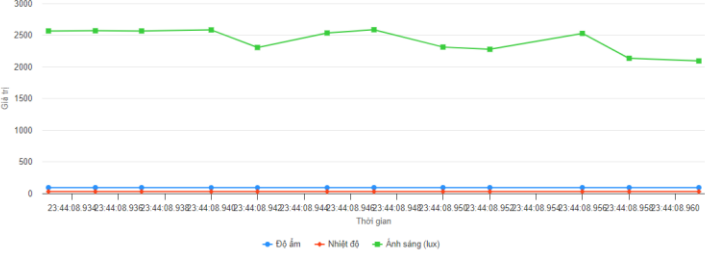
Độ ẩm
91.00 %



Tự động cảnh báo

Live Chart - Nhiệt độ, Độ ẩm và Ánh sáng





Giá trị

Thời gian


Độ ẩm


Nhiệt độ


Ánh sáng (lux)

Highcharts.com

PHÒNG KHÁCH

Đèn

Quạt

Tivi

- **Data Sensor:** Sử dụng thư viện Highcharts để vẽ biểu đồ thể hiện dữ liệu từ cảm biến nhiệt độ, độ ẩm và cường độ ánh sáng theo thời gian thực.

Menu				
HOME				
DATA SENSOR				
ACTION				
PROFILE				

DATA HISTORY				
Chọn trường cần lọc: Nhiệt độ Từ: <input type="text"/> Đến: <input type="text"/> Lọc				
ID	Thời gian	Nhiệt độ	Độ ẩm	Ánh sáng
3166	23:50:41 22/9/2024	29.3	91	2581.35
3165	23:50:39 22/9/2024	29.3	91	2539.45
3164	23:50:36 22/9/2024	29.3	91	2948.73
3163	23:50:34 22/9/2024	29.3	91	2948.73
3162	23:50:32 22/9/2024	29.3	91	2948.73
3161	23:50:29 22/9/2024	29.3	91	2964.84
3160	23:50:27 22/9/2024	29.3	91	3016.41
3159	23:50:24 22/9/2024	29.3	91	3016.41
3158	23:50:22 22/9/2024	29.3	91	3016.41
3157	23:50:20 22/9/2024	29.3	91	3006.74
3156	23:50:18 22/9/2024	29.3	91	3026.07
3155	23:50:15 22/9/2024	29.3	91	3026.07

Lưu
1/159
Tiếp
Page size: 20

- Action History:** Giao diện sẽ nhận dữ liệu cảm biến và trạng thái thiết bị từ server thông qua WebSocket để cập nhật liên tục mà không cần tải lại trang.

Menu				
HOME				
DATA SENSOR				
ACTION				
PROFILE				

ACTION HISTORY				
Từ: <input type="text"/> dd/mm/yyyy Đến: <input type="text"/> dd/mm/yyyy Lọc				
ID	Thời gian	Thiết bị	Hành động	
1	01:43:32 3/9/2024	light	off	
2	01:43:33 3/9/2024	fan	on	
3	01:43:33 3/9/2024	light	on	
4	01:43:33 3/9/2024	fan	off	
5	01:43:34 3/9/2024	fan	on	
6	01:43:34 3/9/2024	light	off	
7	01:43:46 3/9/2024	fan	on	
8	01:43:50 3/9/2024	light	on	
9	01:43:52 3/9/2024	light	on	
10	01:43:56 3/9/2024	light	on	
11	01:43:57 3/9/2024	fan	on	
12	01:44:05 3/9/2024	fan	on	

Lưu
1/7
Tiếp
Page size: 20

- Profile:** Hiện thị các thông tin như tên, mã sinh viên, link github project, báo cáo, api doc và ảnh.

Các công nghệ sử dụng:

- HTML, CSS, JavaScript:** Để xây dựng giao diện người dùng.
- Highcharts:** Thư viện vẽ biểu đồ để trực quan hóa dữ liệu cảm biến.
- Socket.io:** Cung cấp kết nối thời gian thực giữa client và server để nhận và gửi lệnh điều khiển và cập nhật dữ liệu cảm biến.

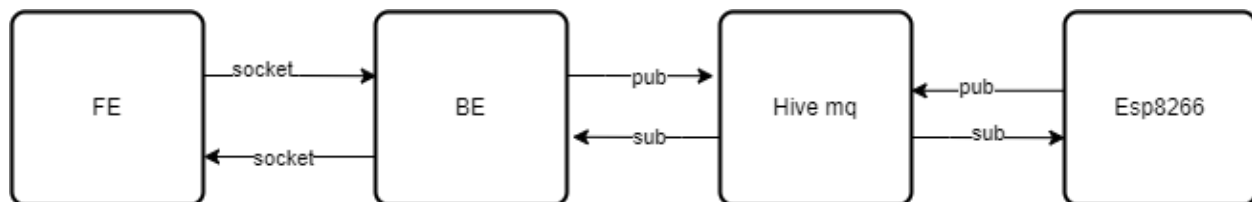
4.3 Backend - Node.js và API

Backend của hệ thống được phát triển trên nền tảng **Node.js**, sử dụng các framework và công cụ để giao tiếp với phần cứng và xử lý dữ liệu.

Các chức năng chính:

- **Giao tiếp với MQTT (HiveMQ):**
Node.js sử dụng thư viện **MQTT.js** để lắng nghe dữ liệu cảm biến được pub lên từ ESP8266, đồng thời gửi các lệnh điều khiển thiết bị từ frontend xuống ESP8266 thông qua giao thức MQTT. Lệnh điều khiển và dữ liệu cảm biến đều được quản lý thông qua các topic trên 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 tới frontend theo thời gian thực. Khi Node.js nhận được dữ liệu cảm biến từ MQTT, nó sẽ đẩy ngay dữ liệu này đến frontend thông qua các sự kiện Socket.io. Người dùng cũng có thể gửi lệnh điều khiển thiết bị (bật/tắt các LED tượng trưng cho quạt, đèn, tivi) từ giao diện web, và lệnh này được chuyển tiếp tới ESP8266 thông qua MQTT.
- **Lưu trữ dữ liệu:**
Dữ liệu cảm biến, lịch sử điều khiển thiết bị, và các sự kiện khác được lưu trữ trong cơ sở dữ liệu **MySQL** để người dùng có thể truy xuất và xem lại lịch sử.

Luồng xử lý:



1. **ESP8266 gửi dữ liệu cảm biến** (nhiệt độ, độ ẩm, cường độ ánh sáng) lên HiveMQ thông qua giao thức MQTT.
2. **Node.js lắng nghe dữ liệu từ HiveMQ:** Dữ liệu cảm biến từ ESP8266 được backend xử lý và lưu trữ vào cơ sở dữ liệu MySQL.
3. **Socket.io đẩy dữ liệu tới frontend:** Backend sử dụng Socket.io để gửi dữ liệu cảm biến cập nhật đến giao diện web theo thời gian thực.
4. **Người dùng điều khiển thiết bị:** Khi người dùng gửi lệnh điều khiển từ giao diện web (bật/tắt LED), lệnh này được gửi tới Node.js thông qua Socket.io. Node.js sau đó pub lệnh này lên HiveMQ để ESP8266 thực hiện điều khiển thiết bị.

Các công nghệ sử dụng:

- **Node.js:** Nền tảng server-side để quản lý toàn bộ luồng giao tiếp và xử lý của hệ thống.
- **Express.js:** Framework để xây dựng backend và quản lý routing cho ứng dụng, mặc dù các API RESTful không được sử dụng cho giao tiếp trực tiếp với frontend trong trường hợp này.

- **MQTT.js:** Thư viện MQTT cho Node.js để giao tiếp với HiveMQ, dùng cho việc gửi và nhận dữ liệu giữa backend và ESP8266.
- **Socket.io:** Thư viện để xử lý giao tiếp thời gian thực giữa frontend và backend, đảm bảo việc truyền tải dữ liệu cảm biến và lệnh điều khiển thiết bị.

4.4 Cơ sở dữ liệu (MySQL)

Cơ sở dữ liệu MySQL được sử dụng để lưu trữ các thông tin liên quan đến cảm biến và điều khiển thiết bị. Cấu trúc bảng được thiết kế để đảm bảo lưu trữ dữ liệu một cách hợp lý và dễ dàng truy vấn khi cần.

Các bảng chính:

- **data_log:** Lưu trữ thông tin dữ liệu cảm biến từ DHT11 (nhiệt độ, độ ẩm) và 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 để phục vụ việc theo dõi và kiểm tra.
 - **Các trường chính:** id, device_name, action, action_time.

MySQL cung cấp một cách thức hiệu quả để lưu trữ và truy xuất dữ liệu cảm biến cũng như các sự kiện điều khiển, cho phép hệ thống dễ dàng hiển thị và phân tích thông tin.

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

MQTT là một giao thức truyền tải nhẹ và rất phù hợp với các ứng dụng IoT. 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).

- **ESP8266** sẽ gửi (publish) các dữ liệu cảm biến như nhiệt độ, độ ẩm và cường độ ánh sáng lên broker MQTT tại topic sau:
 - theanh2032003/esp8266/sensor_data: Đo cường độ ánh sáng, nhiệt độ, độ ẩm và thời gian đo.
- **Node.js server** sẽ đăng ký (subscribe) các topic này để nhận và xử lý dữ liệu từ cảm biến.
- Khi người dùng trên giao diện web gửi lệnh điều khiển các thiết bị, lệnh sẽ được gửi từ server Node.js qua MQTT, đăng (publish) vào các topic sau:
 - theanh2032003/esp8266/fan: Điều khiển quạt.
 - theanh2032003/esp8266/light: Điều khiển đèn.
 - theanh2032003/esp8266/tv: Điều khiển TV.
- Hệ thống giao tiếp này đảm bảo rằng các dữ liệu từ cảm biến được cập nhật nhanh chóng và các lệnh điều khiển được truyền đi một cách chính xác.

Các topic MQTT cụ thể được sử dụng như sau:

- **Topic để điều khiển thiết bị:**
 - theanh2032003/esp8266/light
 - theanh2032003/esp8266/fan
 - theanh2032003/esp8266/television
- **Topic để nhận trạng thái thiết bị:**
 - theanh2032003/esp8266/light/status
 - theanh2032003/esp8266/fan/status
 - theanh2032003/esp8266/television/status
- **Topic để 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 sử dụng Node.js làm nền tảng backend, giao tiếp với ESP8266 thông qua MQTT để thu thập dữ liệu cảm biến và điều khiển thiết bị. Dữ liệu từ các cảm biến như DHT11 và module quang trở được hiển thị thời gian thực trên giao diện web bằng Socket.io, và các biểu đồ được vẽ bằng Highcharts để người dùng dễ dàng quan sát. Hệ thống phần mềm với cấu trúc rõ ràng, dễ mở rộng và duy trì, đáp ứng tốt yêu cầu của ứng dụng IoT trong thực tế.

5. Kết quả và đánh giá

5.1 Kết quả thực nghiệm

Sau khi triển khai, hệ thống đã hoạt động tốt, với các kết quả cụ thể như sau:

- **Thu thập dữ liệu cảm biến:** Dữ liệu về nhiệt độ, độ ẩm và cường độ ánh sáng được thu thập từ DHT11 và module quang trở, được cập nhật liên tục và chính xác.
- **Điều khiển thiết bị:** Các thiết bị (quạt, đèn, tivi) được điều khiển thành công qua giao diện web. Trạng thái bật/tắt của các thiết bị được cập nhật đúng và kịp thời.
- **Biểu đồ trực quan:** Dữ liệu cảm biến được hiển thị thông qua biểu đồ thời gian thực trên giao diện web, giúp người dùng dễ dàng theo dõi.

5.2 Đánh giá hiệu suất

Hệ thống có thể xử lý và điều khiển thiết bị trong thời gian thực với độ trễ thấp, nhờ việc sử dụng giao thức MQTT và WebSocket. ESP8266 có khả năng thu thập dữ liệu ổn định, và Node.js server xử lý dữ liệu và giao tiếp giữa các thành phần nhanh chóng.

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 qua MQTT và WebSocket. Node.js server đã được cấu hình để

xử lý dữ liệu thời gian thực, và MySQL được sử dụng để lưu trữ dữ liệu lịch sử. Frontend cung cấp giao diện thân thiện, dễ sử dụng với các biểu đồ trực quan.

6.2 Định hướng phát triển

- **Mở rộng hệ thống:** Thêm các loại cảm biến khác (chẳng hạn như cảm biến chuyển động, độ ẩm đất) và các thiết bị điều khiển khác (như hệ thống tưới nước tự động).
- **Cải thiện giao diện:** Nâng cấp giao diện người dùng với các tính năng mới, như lập lịch bật/tắt thiết bị tự động, hoặc gửi thông báo khi có sự cố từ cảm biến.
- **Bảo mật hệ thống:** Triển khai các cơ chế bảo mật như mã hóa dữ liệu MQTT, xác thực người dùng và bảo vệ API để tăng cường tính an toàn.