HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN I

BÁO CÁO BÀI TẬP IOT VÀ ỨNG DỤNG
ĐỀ TÀI: XÂY DỰNG HỆ THỐNG CẢM BIẾN IOT

Họ và tên: Tạ Kiều Yến

Mã sinh viên: B21DCCN810

Nhóm lớp học: 05

Giảng viên giảng dạy: Nguyễn Quốc Uy

Hà Nội – 2024

Mục lục

Phụ lục

| Chương 1: Tổng quan về IoT

1.1. Giới thiệu chung về IoT

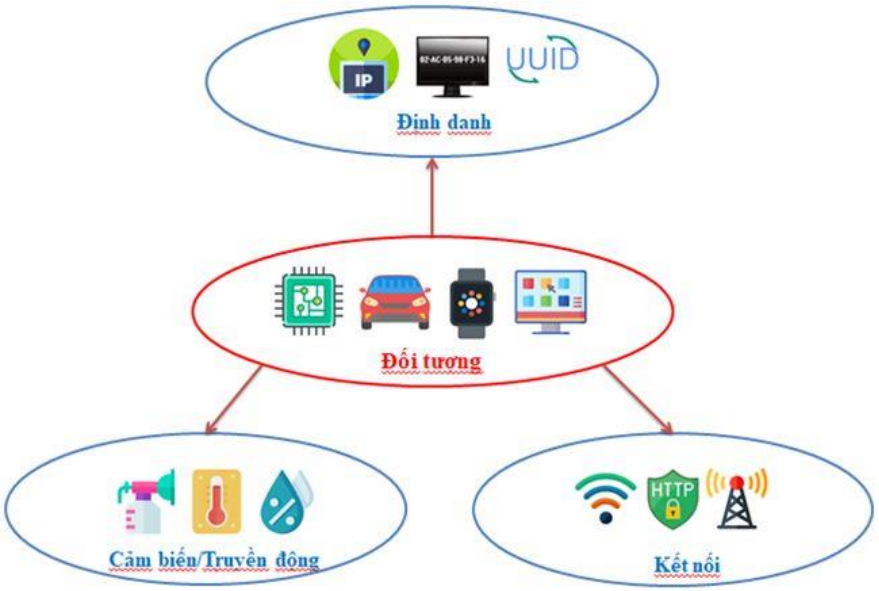
IoT là viết tắt của "Internet of Things". Đây là một khái niệm mô tả sự kết nối của các thiết bị vật lý thông qua internet, cho phép chúng giao tiếp và trao đổi dữ liệu một cách tự động và thông minh mà không cần sự can thiệp của con người.

Các thiết bị IoT có thể bao gồm cảm biến, máy móc, thiết bị điện tử, xe cộ, đèn đom, máy lạnh, và hầu hết mọi thiết bị khác có thể kết nối với internet. Nhờ vào IoT, các thiết bị này có khả năng thu thập dữ liệu, truyền tải thông tin và thực hiện các tác vụ một cách tự động và hiệu quả.

Ứng dụng của IoT rất đa dạng, từ việc tối ưu hóa quản lý năng lượng, giám sát môi trường, đến tự động hóa trong công nghiệp và nhà thông minh. IoT cũng mở ra khả năng sáng tạo mới và tạo ra những dịch vụ thông minh dựa trên dữ liệu, mang lại lợi ích rõ ràng cho cả cá nhân, doanh nghiệp và xã hội.

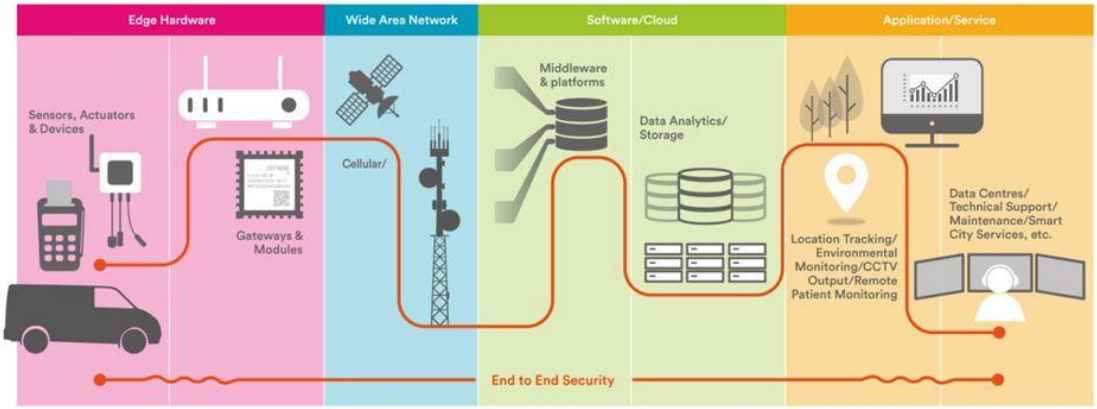
Với sự phát triển nhanh chóng của công nghệ và mạng lưới internet, IoT đang trở thành một nguồn lực quan trọng và mạnh mẽ, có tiềm năng thay đổi cách chúng ta tương tác với thế giới xung quanh.

1.2. Mô hình hệ thống IoT



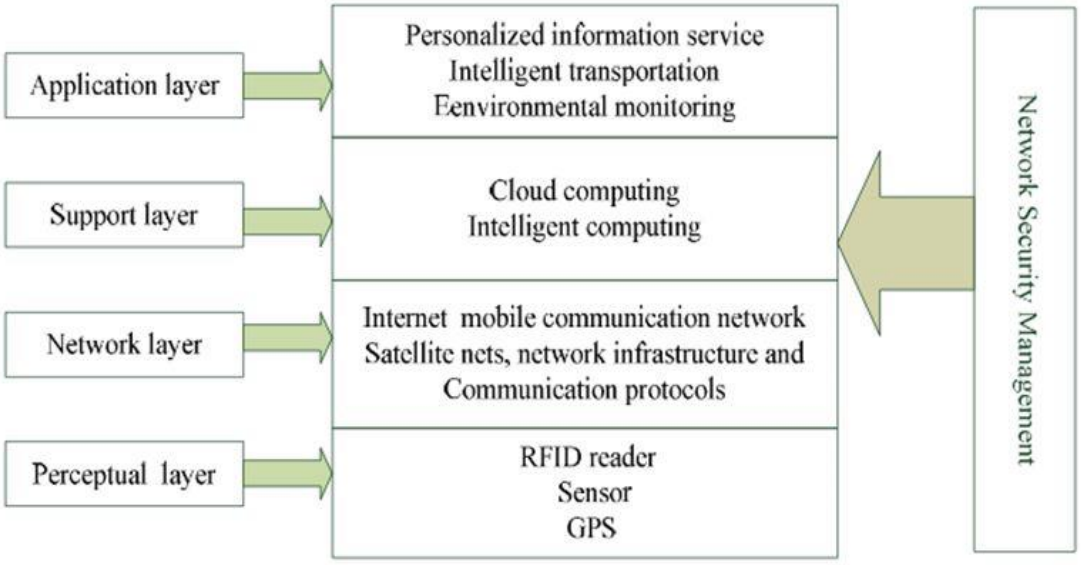
Hình 1: Các thành phần cơ bản của hệ thống Internet of Things

Sự phát triển của IoT không phải là sự phát triển của một công nghệ riêng lẻ nào mà là sự tổng hợp, thúc đẩy cải tiến không ngừng của hàng loạt các lĩnh vực công nghệ nền tảng khác nhau, trong đó gồm 4 lĩnh vực công nghệ chính luôn gắn liền với sự phát triển xu thế IoT bao gồm: phần cứng, truyền thông kết nối, phần mềm và ứng dụng.



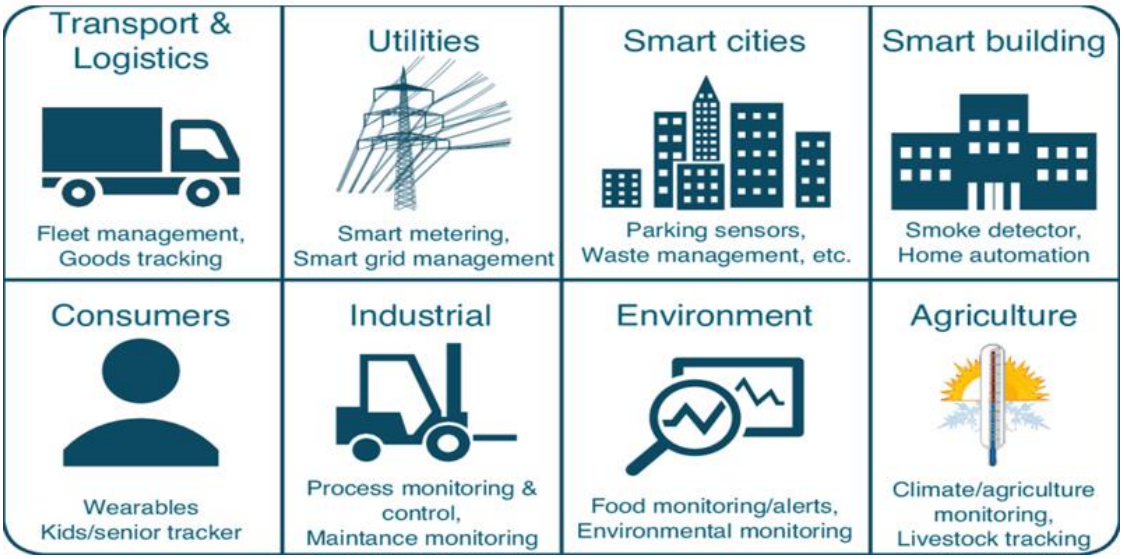
Hình 2: Bốn lĩnh vực công nghệ chính

Kiến trúc IoT có thể chia thành 3 lớp, 4 lớp hoặc thậm chí là 5 lớp tùy theo tính chi tiết của kiến trúc, nhưng nhìn chung mô hình 4 lớp phổ biến và được ứng dụng nhiều nhất gồm các lớp: tri giác (perceptual), mạng (network), hỗ trợ (support) và ứng dụng (application), bên cạnh đó yếu tố bảo mật đi kèm mỗi lớp cũng cực kỳ quan trọng.



Hình 3: Kiến trúc Internet of Things

1.3. Ứng dụng của IoT



Hình 4: ứng dụng của IoT trong các ngành công nghiệp

Một số ngành, lĩnh vực có ứng dụng hệ thống giải pháp IoT có thể kể đến bao gồm:

- Ứng dụng vào hoạt động sản xuất sản phẩm, dịch vụ thương mại.
- Ứng dụng trong quản lý và bảo trì hệ thống dây chuyền sản xuất.
- Sử dụng trong một số lĩnh vực dịch vụ như marketing, sale, kết nối với bộ phận sản xuất, ...
- Các ứng dụng nhà thông minh, đồ dùng nội thất thông minh.
- Ứng dụng quản lý các thiết bị điện tử cá nhân như thiết bị đeo tay để đo nhịp tim, huyết áp.
- Quản lý môi trường.
- Quản lý giao thông.
- Sử dụng, xử lý trong các tình huống khẩn cấp.
- Trong lĩnh vực mua sắm đồ thông minh.
- Trong các đồ dùng sinh hoạt hàng ngày như: bình nóng lạnh, máy pha cà phê, ...
- Tự động hóa trong các công xưởng sản xuất xe hơi áp dụng công nghệ IoT để giảm nhân lực vận hành.
- IoT giúp quản lý chất lượng không khí thông minh, kiểm soát truy cập an ninh, đo lường tác động môi trường thông minh, ...
- Bảo vệ tài sản, đo lường rủi ro, quản lý cơ sở.
- Trong các lĩnh vực an toàn, bảo mật, bảo vệ công nhất để đảm bảo năng suất lao động.

| Chương 2: Giới thiệu

2.1. Giới thiệu đề tài

Đề tài tập trung vào thiết kế và triển khai một hệ thống cảm biến dựa trên công nghệ IoT (Internet of Things) nhằm giám sát và điều khiển các thông số môi trường trong thời gian thực. Hệ thống này bao gồm các thiết bị cảm biến đo lường các thông số như nhiệt độ, độ ẩm, ánh sáng và các thiết bị điều khiển như quạt, đèn, điều hòa.

2.2. Công nghệ sử dụng

2.2.1. Cơ sở dữ liệu MongoDB

- Lý do chọn MongoDB:
  - Lưu trữ linh hoạt: MongoDB sử dụng cấu trúc lưu trữ dạng tài liệu JSON/BSON, linh hoạt trong việc lưu trữ các loại dữ liệu khác nhau mà không cần định dạng cứng nhắc như các cơ sở dữ liệu quan hệ.
  - Dữ liệu phi cấu trúc: Với đặc điểm phi cấu trúc, MongoDB phù hợp cho các hệ thống IoT, nơi các dữ liệu từ cảm biến có thể thay đổi hoặc mở rộng về loại thông số đo lường.
  - Khả năng mở rộng: MongoDB hỗ trợ phân tán và mở rộng theo chiều ngang, dễ dàng thích ứng khi hệ thống IoT cần mở rộng thêm thiết bị hoặc địa điểm giám sát.
  - Truy xuất nhanh: Tính năng tìm kiếm, lập chỉ mục và truy xuất nhanh của MongoDB giúp hệ thống cập nhật dữ liệu liên tục và xử lý kịp thời.
- Thiết kế cấu trúc cơ sở dữ liệu
  - Collection sensordatas: Lưu trữ dữ liệu môi trường từ các cảm biến. Mỗi tài liệu trong collection này đại diện cho một lần đo, bao gồm các trường:
    - _id: ID tự động của MongoDB.
    - temperature: Nhiệt độ đo được (°C).
    - humidity: Độ ẩm đo được (%).
    - lighting: Mức ánh sáng đo được (lux).
    - timestamp: Thời gian đo đạc (Date).
  - Collection histories: Lưu trữ lịch sử điều khiển thiết bị, để theo dõi các hành động điều khiển và phân tích hành vi sử dụng, gồm:
    - _id: ID tự động của MongoDB.
    - action: Hành động điều khiển (ON/OFF).
    - timestamp: Thời gian thực hiện hành động.

- Xử lý và truy vấn dữ liệu

  Dữ liệu được lưu trong MongoDB sẽ được truy xuất và xử lý thông qua các API của hệ thống:

  - Truy vấn dữ liệu cảm biến: API GET/sensor-data sẽ lấy dữ liệu từ collection sensordatas và hỗ trợ lọc theo các thông số như thời gian hoặc phạm vi giá trị.
  - Điều khiển thiết bị: Khi người dùng hoặc hệ thống yêu cầu thay đổi trạng thái thiết bị, hệ thống sẽ ghi lại hành động vào collection histories.

2.2.2. Backend (Node.js, Express.js)

- Lý do chọn Node.js và Express.js
  - Hiệu năng cao: Node.js hỗ trợ xử lý bất đồng bộ và sự kiện, phù hợp với các hệ thống IoT nơi dữ liệu từ các cảm biến được cập nhật liên tục.
  - Express.js: Framework nhẹ và linh hoạt, hỗ trợ nhanh chóng tạo API RESTful, giúp dễ dàng quản lý các yêu cầu và xử lý dữ liệu.
  - Hỗ trợ MQTT: Node.js có các thư viện hỗ trợ MQTT, giúp kết nối và trao đổi dữ liệu giữa backend và các thiết bị IoT dễ dàng.
- Kiến trúc backend:

  Kiến trúc backend bao gồm các thành phần chính như API xử lý dữ liệu cảm biến, API điều khiển thiết bị, và các chức năng lưu trữ dữ liệu vào MongoDB. Các thành phần chính trong kiến trúc này bao gồm:
  - Express.js server: Đóng vai trò là server chính, lắng nghe các yêu cầu từ frontend và xử lý yêu cầu từ thiết bị.
  - MQTT broker: Dùng để nhận và gửi dữ liệu tới các thiết bị IoT theo giao thức MQTT, quản lý và truyền thông giữa các cảm biến và backend.
  - MongoDB: Cơ sở dữ liệu NoSQL để lưu trữ thông tin thiết bị, lịch sử điều khiển và dữ liệu cảm biến.
- Thiết kế API:

  Backend cung cấp một số API chính để phục vụ các chức năng của hệ thống:
  - API cảm biến:
    - GET /datasensor: Lấy dữ liệu cảm biến từ MongoDB, bao gồm các thông số nhiệt độ, độ ẩm, và ánh sáng. Hỗ trợ lọc theo thời gian và các loại cảm biến.

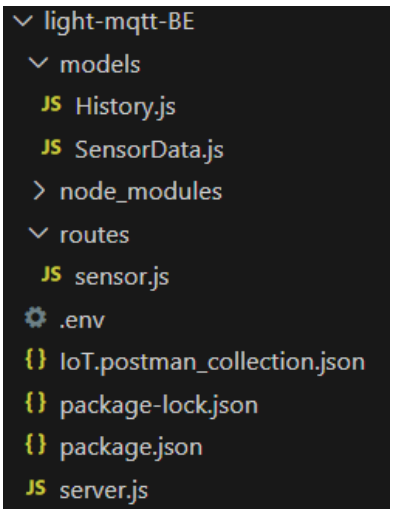
- API điều khiển thiết bị:
  - POST /control: Nhận yêu cầu từ người dùng để bật/tắt các thiết bị (đèn, quạt, máy lạnh) và gửi lệnh đến MQTT broker để điều khiển thiết bị. API này đồng thời ghi lại lịch sử điều khiển vào MongoDB.
  - GET /history: Lấy lịch sử điều khiển thiết bị, bao gồm ID thiết bị, hành động (ON/OFF), và thời gian thực hiện.

- Xử lý MQTT:

  Backend sử dụng thư viện mqtt của Node.js để kết nối với MQTT broker. Các chức năng xử lý MQTT bao gồm:
  - Nhận dữ liệu từ cảm biến: Khi cảm biến gửi dữ liệu tới MQTT broker, backend sẽ nhận dữ liệu này, xử lý và lưu trữ vào MongoDB.
  - Điều khiển thiết bị: Khi có yêu cầu bật/tắt từ API /control, backend gửi lệnh đến MQTT broker, sau đó MQTT gửi lệnh này đến thiết bị IoT tương ứng. Sau khi thiết bị phản hồi, backend sẽ cập nhật trạng thái thiết bị và ghi lại lịch sử.

- Cấu trúc mã nguồn backend:

  Backend được tổ chức thành các file và thư mục chính:
  - Server.js: Khởi tạo server Express, cài đặt các middleware, và định nghĩa các routes.
  - routes/: Thư mục chứa các file định nghĩa routes chính, bao gồm:
    - sensor.js: Định nghĩa các route xử lý dữ liệu cảm biến.
  - models/: Định nghĩa các mô hình MongoDB cho các collection như sensordatas, histories.
    - History.js: định nghĩa mô hình cho collection histories.
    - SensorData.js: định nghĩa mô hình cho collection sensordatas.

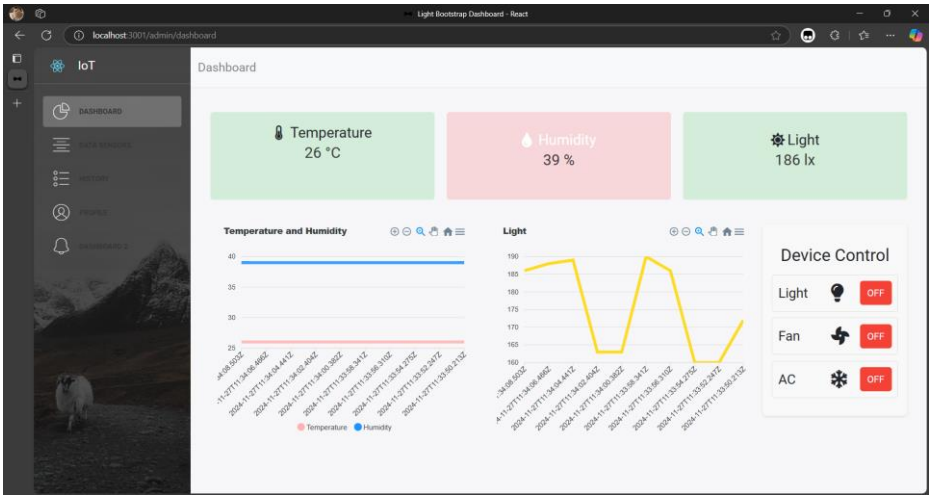


Hình 5: Cấu trúc backend

2.2.3. Frontend (React.js)

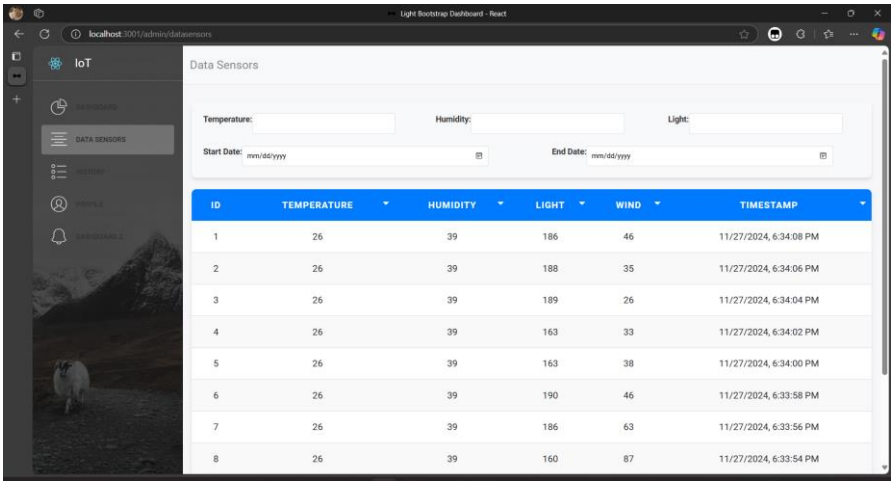
- Lý do chọn React.js
  - Hiệu suất cao: ReactJS sử dụng Virtual DOM, giúp cập nhật giao diện nhanh chóng khi có sự thay đổi dữ liệu từ các cảm biến hoặc các thao tác của người dùng.
  - Tái sử dụng component: ReactJS cho phép xây dựng các component tái sử dụng, dễ dàng quản lý và mở rộng khi thêm tính năng mới hoặc các loại cảm biến khác.
  - Hỗ trợ thư viện mạnh mẽ: Các thư viện và công cụ hỗ trợ như react-router-dom, axios, và react-apexcharts giúp xây dựng giao diện người dùng thân thiện, truy xuất dữ liệu API, và hiển thị biểu đồ cho hệ thống giám sát.
- Kiến trúc và thiết kế giao diện:

  Giao diện hệ thống được chia thành các thành phần chính, bao gồm bảng điều khiển (dashboard), trang giám sát dữ liệu cảm biến (Data Sensor), lịch sử điều khiển thiết bị (History) và trang cá nhân (Profile). Các thành phần này được tổ chức theo kiến trúc component-based của ReactJS.



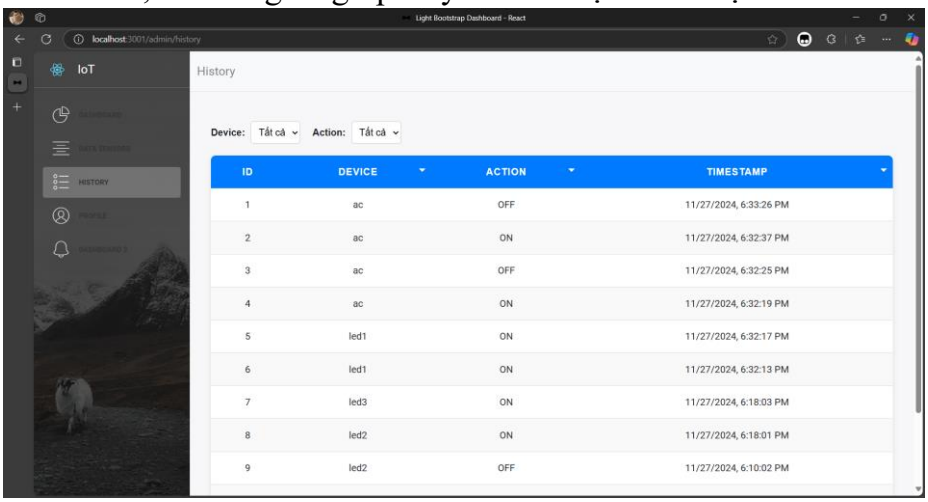
Hình 6: Giao diện Dashboard

  - Dashboard: Đây là trang chính, hiển thị tổng quan về các thông số môi trường từ các cảm biến như nhiệt độ, độ ẩm, và ánh sáng, đồng thời cho phép người dùng điều khiển các thiết bị (như quạt, đèn, điều hòa) thông qua các nút ON/OFF. Dữ liệu được cập nhật theo thời gian thực thông qua WebSocket hoặc các lần truy xuất API và hiển thị ra biểu đồ đường.



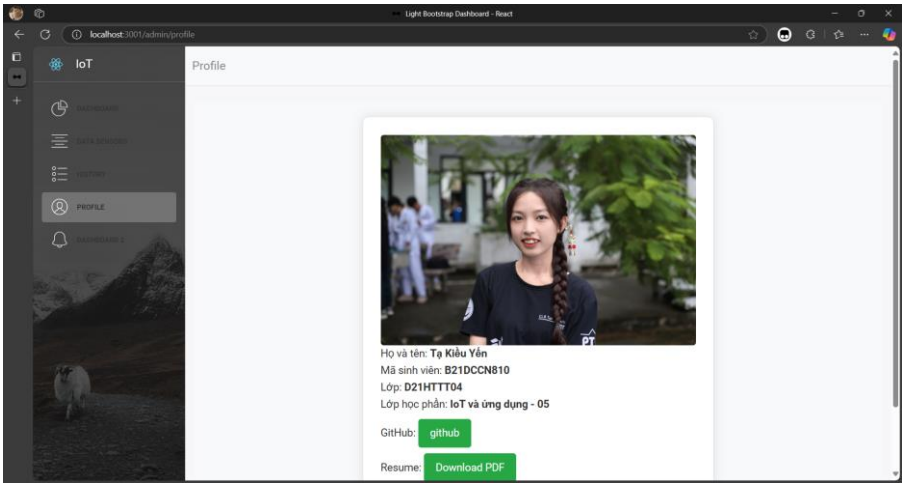
Hình 7: Giao diện Data Sensor

- Trang Data Sensors: Hiển thị chi tiết dữ liệu cảm biến theo dạng bảng với các cột như ID, nhiệt độ, độ ẩm, ánh sáng và thời gian đo. Bảng này hỗ trợ các tính năng như:
  - Phân trang (Pagination): Giúp hiển thị một lượng dữ liệu vừa phải và dễ dàng duyệt qua các trang dữ liệu cũ.
  - Lọc dữ liệu (Filtering): Cho phép người dùng lọc dữ liệu theo các tiêu chí như khoảng thời gian, mức nhiệt độ, độ ẩm, ánh sáng và giúp truy vấn dữ liệu linh hoạt.



Hình 8: Giao diện History

- Trang History: Hiển thị lịch sử điều khiển thiết bị với các cột như ID, tên thiết bị, hành động (ON/OFF) và thời gian điều khiển. Trang này giúp người dùng theo dõi các hoạt động điều khiển trong quá khứ.



Hình 9: Giao diện profile

  - Trang Profile: Cung cấp thông tin cá nhân và tài liệu về báo cáo.
- Tính năng chính
  - Cập nhật dữ liệu thời gian thực: Frontend sử dụng WebSocket để nhận các bản cập nhật dữ liệu từ backend, giúp giao diện luôn hiển thị thông tin mới nhất về tình trạng môi trường và trạng thái thiết bị.
  - Điều khiển thiết bị từ xa: Người dùng có thể điều khiển các thiết bị bằng cách nhấp vào các nút ON/OFF. Trạng thái nút sẽ chỉ thay đổi khi nhận được xác nhận từ backend rằng thiết bị đã được bật hoặc tắt thành công.
  - Biểu đồ trực quan: Sử dụng thư viện react-apexcharts để hiển thị dữ liệu cảm biến dạng biểu đồ thời gian thực, giúp người dùng dễ dàng theo dõi các biến động về nhiệt độ, độ ẩm và ánh sáng theo thời gian.
  - Giao diện dựa trên Light Bootstrap Dashboard của Creative Tim, giao diện được tối ưu hóa để đơn giản, thân thiện với người dùng và dễ dàng truy cập các thông tin quan trọng.
- Quy trình hoạt động
  - Lấy dữ liệu từ backend: Frontend thường xuyên lấy dữ liệu từ backend thông qua API hoặc WebSocket. Đối với các trang có bảng dữ liệu (như Data Sensors và History), dữ liệu sẽ được lấy từ MongoDB, với các bộ lọc và phân trang được hỗ trợ trực tiếp từ backend.
  - Xử lý phản hồi điều khiển: Khi người dùng điều khiển thiết bị, một yêu cầu sẽ được gửi đến backend qua API, đồng thời nút trạng thái của thiết bị sẽ chờ phản hồi từ backend để đảm bảo thiết bị đã thay đổi trạng thái chính xác trước khi cập nhật giao diện.

- Định dạng thời gian: Để hiển thị thời gian đo và thời gian điều khiển, hệ thống sử dụng thư viện date-fns để định dạng thời gian thành dạng dễ đọc như ngày, giờ và phút.

- Các thư viện và công cụ sử dụng:
  - React Router: Để điều hướng giữa các trang trong ứng dụng.
  - Axios: Để gửi yêu cầu HTTP đến backend, lấy và gửi dữ liệu từ/to MongoDB.
  - WebSocket: Để nhận dữ liệu cảm biến theo thời gian thực, cập nhật giao diện mà không cần phải tải lại trang.
  - React-ApexCharts: Để tạo biểu đồ trực quan hiển thị dữ liệu cảm biến theo thời gian.
  - Date-fns: Để định dạng ngày tháng và thời gian.

2.3. Thiết bị phần cứng

2.3.1. ESP8266

- Chức năng: ESP8266 là một vi điều khiển tích hợp Wi-Fi, giúp kết nối các cảm biến với hệ thống quản lý qua giao thức MQTT và truyền dữ liệu lên máy chủ. ESP8266 cũng có khả năng xử lý các lệnh điều khiển từ người dùng, kích hoạt các thiết bị đầu cuối như quạt, đèn và điều hòa.
- Thông số kỹ thuật chính:
  - CPU: Bộ xử lý 32-bit RISC với tốc độ 80 MHz.
  - Bộ nhớ: 64 KB bộ nhớ lệnh, 96 KB bộ nhớ dữ liệu.
  - Flash: Hỗ trợ lên đến 4MB flash.
  - Wi-Fi: Chuẩn IEEE 802.11 b/g/n, hỗ trợ chế độ AP (Access Point) và STA (Station).
  - Giao tiếp: Hỗ trợ nhiều giao thức giao tiếp như UART, SPI, I2C.
- Vai trò trong hệ thống: ESP8266 nhận dữ liệu từ các cảm biến như DHT11 và cảm biến ánh sáng, sau đó gửi dữ liệu này lên hệ thống quản lý qua Wi-Fi. Thiết bị này cũng nhận các lệnh điều khiển từ hệ thống qua giao thức MQTT để điều khiển các thiết bị như đèn hoặc quạt. Khả năng xử lý tín hiệu và kết nối Wi-Fi tích hợp giúp ESP8266 hoạt động như một trung tâm thu thập và điều khiển dữ liệu trong hệ thống IoT.

2.3.2. Cảm biến nhiệt độ và độ ẩm DHT11

- Chức năng: Cảm biến DHT11 đo nhiệt độ và độ ẩm trong môi trường.
- Thông số kỹ thuật chính:
  - Nguồn điện: 3.3V - 5V.
  - Dải đo nhiệt độ: 0°C đến 50°C với sai số ±2°C.
  - Dải đo độ ẩm: 20% đến 90% RH với sai số ±5%.
  - Tần suất cập nhật: 1 lần mỗi 5 giây.
  - Giao tiếp: Giao tiếp qua giao thức 1-Wire.

- Vai trò trong hệ thống: DHT11 được kết nối với ESP8266 để cung cấp dữ liệu nhiệt độ và độ ẩm cho hệ thống. Với khả năng đo chính xác và cập nhật nhanh chóng, DHT11 là lựa chọn phù hợp cho các ứng dụng giám sát môi trường trong thời gian thực.

2.3.3. Cảm biến ánh sáng LM393

- Chức năng: Cảm biến ánh sáng LM393 được sử dụng để phát hiện mức độ chiếu sáng xung quanh, giúp hệ thống tự động điều chỉnh thiết bị chiếu sáng hoặc theo dõi môi trường ánh sáng.
- Thông số kỹ thuật chính:
  - Nguồn điện: 3.3V - 5V.
  - Đầu ra:
  - Đầu ra kỹ thuật số (Digital) dựa trên tín hiệu ON/OFF khi ánh sáng vượt ngưỡng.
  - Đầu ra tương tự (Analog) phản ánh mức độ ánh sáng bằng tín hiệu điện áp.
  - Điều chỉnh độ nhạy: Có biến trở trên module để điều chỉnh ngưỡng kích hoạt đầu ra kỹ thuật số.
  - Dải đo: Phụ thuộc vào loại cảm biến ánh sáng tích hợp, thường phù hợp để phát hiện ánh sáng mạnh hoặc yếu.
- Vai trò trong hệ thống:
  - Theo dõi mức ánh sáng xung quanh và điều khiển các thiết bị chiếu sáng (bật/tắt đèn tự động) dựa trên ngưỡng đã thiết lập.
  - Cung cấp dữ liệu ánh sáng lên hệ thống để người dùng quản lý và theo dõi tình trạng chiếu sáng trong khu vực giám sát.
  - Tích hợp hiệu quả với các hệ thống IoT nhờ khả năng cung cấp tín hiệu kỹ thuật số đơn giản.

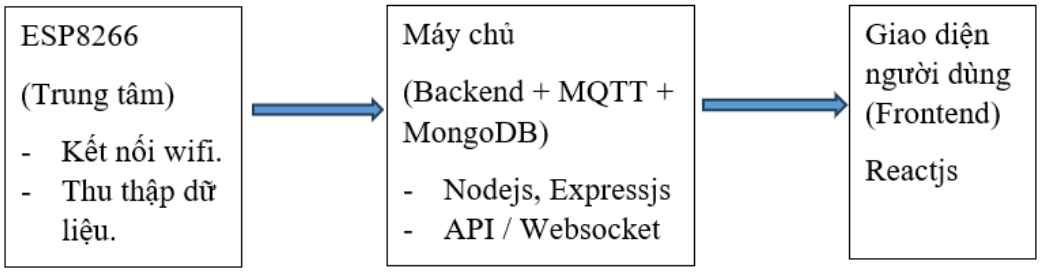
| Chương 3: Thiết kế hệ thống và kết nối mạng, giao thức

3.1. Thiết kế hệ thống

3.1.1. Sơ đồ hệ thống

Sơ đồ tổng quan của hệ thống IoT giám sát và điều khiển bao gồm các thành phần chính:

- Thiết bị điều khiển trung tâm (ESP8266): Có vai trò thu thập dữ liệu từ các cảm biến (DHT11 và cảm biến ánh sáng) và truyền dữ liệu lên máy chủ qua kết nối Wi-Fi.
- Các cảm biến môi trường: Cảm biến DHT11 thu thập thông tin nhiệt độ và độ ẩm, còn cảm biến ánh sáng LM393 đo mức độ chiếu sáng xung quanh.
- Máy chủ (backend): Nhận dữ liệu từ ESP8266, lưu trữ vào cơ sở dữ liệu MongoDB và gửi phản hồi điều khiển thiết bị khi nhận được yêu cầu từ người dùng.
- Giao diện người dùng (frontend): Xây dựng bằng ReactJS, cho phép người dùng giám sát thông tin môi trường và điều khiển thiết bị từ xa.
- Giao thức MQTT: Được dùng để truyền dữ liệu cảm biến từ ESP8266 đến máy chủ và gửi các lệnh điều khiển từ máy chủ đến ESP8266.



Hình 10: Sơ đồ hệ thống cảm biến IoT

3.1.2. Thiết kế mạch

Thiết kế mạch cho hệ thống IoT giám sát và điều khiển sử dụng các thành phần chính là ESP8266, cảm biến DHT11, cảm biến ánh sáng LM393, và các thiết bị điều khiển bao gồm đèn, quạt và điều hòa. Dưới đây là chi tiết kết nối của từng phần:

- Cảm biến nhiệt độ, độ ẩm DHT11:
  - Chân VCC (+): Kết nối với chân 3.3V của ESP8266.
  - Chân GND (-): Kết nối với chân GND của ESP8266.
  - Chân Data (OUT): Kết nối với chân D2 của ESP8266 để thu thập dữ liệu nhiệt độ và độ ẩm.

- Cảm biến ánh sáng LM393:
  - Chân VCC: Kết nối với chân 3.3V của ESP8266.
  - Chân GND: Kết nối với chân GND của ESP8266.
  - Chân D0: Không kết nối.
  - Chân A0: Kết nối với chân D0 của ESP8266 để thu thập dữ liệu cường độ ánh sáng.
- Đèn LED (đèn led - led1):
  - Chân Dương (Anode, dài hơn): Kết nối với chân D3 của ESP8266.
  - Chân Âm (Cathode, ngắn hơn): Kết nối với chân GND của ESP8266.
- Đèn LED (Quạt - fan):
  - Chân Dương (Anode, dài hơn): Kết nối với chân D4 của ESP8266.
  - Chân Âm (Cathode, ngắn hơn): Kết nối với chân GND của ESP8266.
- Đèn LED (Điều hòa - ac):
  - Chân Dương (Anode, dài hơn): Kết nối với chân D5 của ESP8266.
  - Chân Âm (Cathode, ngắn hơn): Kết nối với chân GND của ESP8266.

3.2. Kết nối mạng và giao thức

Hệ thống sử dụng kết nối Wi-Fi để truyền dữ liệu từ ESP8266 đến máy chủ và nhận các lệnh điều khiển từ giao diện người dùng qua giao thức MQTT.

- Kết nối mạng (Wi-Fi): ESP8266 kết nối vào mạng Wi-Fi để truyền dữ liệu và lệnh điều khiển qua Internet. Địa chỉ IP của ESP8266 được cấu hình cố định hoặc cấp phát động qua DHCP của bộ định tuyến để đảm bảo thiết bị có thể truy cập liên tục.
- Giao thức truyền thông (MQTT):
  - MQTT là giao thức nhắn tin nhẹ, tối ưu cho các thiết bị IoT có tài nguyên hạn chế, như ESP8266. Giao thức này sử dụng mô hình publish-subscribe, giúp dữ liệu được gửi và nhận linh hoạt giữa các thiết bị và hệ thống.
  - Cấu trúc hoạt động:
    - Publish: ESP8266 đóng vai trò “publish” dữ liệu cảm biến đến các topic.
    - Subscribe: Máy chủ sẽ “subscribe” vào các chủ đề này để nhận dữ liệu thời gian thực. Ngoài ra, máy chủ có thể gửi lệnh điều khiển như home/led1, home/ac, home/fan, home/all để ESP8266 nhận và điều khiển các thiết bị tương ứng.

| Chương 4: Kết luận

4.1. Đánh giá hiệu quả

Hệ thống giám sát và điều khiển sử dụng công nghệ IoT đã đạt được các mục tiêu đề ra ban đầu, cụ thể là:

- Tính ổn định: Hệ thống hoạt động ổn định trong việc thu thập dữ liệu từ các cảm biến nhiệt độ, độ ẩm, ánh sáng và truyền dữ liệu về máy chủ qua giao thức MQTT. Việc lưu trữ và xử lý dữ liệu trong MongoDB cho phép hệ thống quản lý thông tin một cách hiệu quả, giúp người dùng có thể dễ dàng truy xuất lịch sử dữ liệu khi cần.
- Khả năng điều khiển từ xa: Hệ thống cho phép người dùng điều khiển các thiết bị như đèn, quạt, và điều hòa thông qua giao diện ReactJS trên trình duyệt. Các lệnh điều khiển được thực hiện nhanh chóng và có phản hồi trực tiếp từ ESP8266 để xác nhận trạng thái của thiết bị.
- Giao diện người dùng thân thiện: Giao diện người dùng được xây dựng bằng ReactJS trực quan, hiển thị dữ liệu môi trường theo thời gian thực và cung cấp các công cụ điều khiển tiện lợi. Khả năng sử dụng các biểu đồ trực quan hóa giúp người dùng theo dõi và phân tích thông tin dễ dàng.

Tuy nhiên, hệ thống cũng tồn tại một số hạn chế:

- Phụ thuộc vào Wi-Fi: Hệ thống chỉ hoạt động trong môi trường có kết nối Wi-Fi, điều này có thể gây hạn chế khi kết nối gặp sự cố hoặc không ổn định.
- Giới hạn thiết bị điều khiển: Hiện tại, hệ thống chỉ hỗ trợ điều khiển một số thiết bị cơ bản (đèn, quạt, điều hòa), và chưa hỗ trợ quản lý nhiều loại thiết bị khác nhau.

4.2. Hướng phát triển trong tương lai: Mở rộng thiết bị và chức năng điều khiển.

- Thêm cảm biến và thiết bị mới: Mở rộng các loại cảm biến để hệ thống có thể áp dụng vào nhiều lĩnh vực khác nhau. Việc tích hợp các loại cảm biến như cảm biến khí gas, cảm biến chuyển động và cảm biến khói có thể mở rộng khả năng của hệ thống sang các mục đích an ninh, phòng chống cháy nổ và giám sát an toàn trong không gian.
    - Cảm biến khí gas có thể phát hiện các chất khí dễ cháy hoặc độc hại trong không khí như metan, CO (carbon monoxide), giúp cảnh báo sớm cho người dùng trong trường hợp rò rỉ khí gas, giảm nguy cơ cháy nổ.
    - Cảm biến chuyển động sẽ theo dõi sự xuất hiện của các đối tượng trong phạm vi cảm biến, giúp hệ thống có khả năng phát hiện xâm nhập bất hợp pháp hoặc tự động bật/tắt đèn khi có người ra vào, tiết kiệm năng lượng và tăng tính bảo mật.

- Cảm biến khói có thể phát hiện khói hoặc lửa trong không gian, đưa ra cảnh báo kịp thời để phòng ngừa các sự cố cháy nổ.

Ngoài ra, hệ thống cũng có thể mở rộng với các thiết bị điều khiển khác như rèm cửa, hệ thống tưới cây tự động, hệ thống điều hòa không khí, hay hệ thống điều khiển âm thanh. Ví dụ, hệ thống tưới cây tự động có thể giúp điều chỉnh mức độ ẩm trong đất dựa trên dữ liệu từ cảm biến độ ẩm đất, giúp quản lý việc chăm sóc cây trồng thông minh và hiệu quả.

- Phân quyền điều khiển: Phát triển tính năng phân quyền cho phép nhiều người dùng cùng quản lý hệ thống với các mức độ điều khiển khác nhau, phục vụ cho các tình huống sử dụng trong gia đình hoặc doanh nghiệp. Hệ thống có thể phân cấp quyền điều khiển theo các mức độ như sau:
  - Quyền quản trị viên: Được toàn quyền điều khiển, cài đặt cấu hình và thêm hoặc loại bỏ các thiết bị, người dùng trong hệ thống. Quản trị viên cũng có quyền xem toàn bộ lịch sử dữ liệu, kiểm tra, và phân quyền cho các thành viên khác.
  - Quyền người dùng: Người dùng với quyền hạn này có thể kiểm soát một số thiết bị nhất định trong hệ thống, nhận thông báo và xem các thông số môi trường như nhiệt độ, độ ẩm, ánh sáng nhưng không có quyền thay đổi các thiết lập cấu hình hệ thống.
  - Quyền khách: Quyền hạn này cho phép truy cập giới hạn, ví dụ như chỉ được phép xem dữ liệu cảm biến, nhưng không có khả năng điều khiển hoặc thay đổi bất kỳ thiết lập nào. Quyền này có thể áp dụng cho những người ngoài gia đình hoặc nhân viên tạm thời trong doanh nghiệp.

Phân quyền điều khiển không chỉ giúp tăng cường bảo mật mà còn hỗ trợ việc quản lý hệ thống linh hoạt hơn, phù hợp với nhu cầu và quyền riêng tư của từng thành viên trong hệ thống. Việc áp dụng phân quyền cho các cấp độ người dùng sẽ giúp hệ thống IoT trở nên đa dạng hơn trong ứng dụng, đặc biệt là trong môi trường có nhiều người sử dụng hoặc khi có sự phân công nhiệm vụ rõ ràng.

Tài liệu tham khảo

1. **Creative Tim, “Light Bootstrap Dashboard React,” GitHub Repository**
2. ***HiveMQ, "HiveMQ Blog, Connecting the world with MQTT" – Tài liệu về MQTT trên trang HiveMQ Blog***
3. **IEEE, “Internet of Things (IoT): An Overview,” - Tài liệu từ IEEE giới thiệu tổng quan về IoT, từ các khái niệm cơ bản đến ứng dụng.**
4. **MongoDB, Inc., “MongoDB Documentation,” - Tài liệu chính thức của MongoDB, bao gồm hướng dẫn về cách thiết lập cơ sở dữ liệu, thực hiện các truy vấn và lưu trữ dữ liệu cảm biến.**
5. **EMQX, “How to Use MQTT in Node.js,”**