**BỘ KHOA HỌC VÀ CÔNG NGHỆ**

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

-----🙞🙜🕮🙞🙜-----



**BÁO CÁO ĐỒ ÁN MÔN**

**MẠNG CẢM BIẾN**

**Đề tài: “*Hệ thống giám sát và điều khiển môi trường phòng học thông minh tính hợp trí tuệ nhân tạo tại biên”***

**Nhóm: 1**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn** | **: Ths. Nguyễn Thị Thanh Thủy** |
| **Nhóm thực hiện** | **: 17** |
| **Sinh viên thực hiện** | **: Đoàn Long Vũ – B21DCDT247** |
|  | **: Bùi Thành Vinh – B21DCDT028** |

**HÀ NỘI – 2025**

# **MỤC LỤC**

[MỤC LỤC 2](#_Toc201233406)

[DANH MỤC VIẾT TẮT 4](#_Toc201233407)

[MỤC LỤC BẢNG BIỂU 6](#_Toc201233408)

[MỤC LỤC HÌNH ẢNH 7](#_Toc201233409)

[LỜI MỞ ĐẦU 9](#_Toc201233410)

[CHƯƠNG 1: CƠ SỞ LÝ THUYẾT 10](#_Toc201233411)

[1.1. Mạng Cảm biến không dây (Wireless Sensor Networks - WSN) 10](#_Toc201233412)

[1.2. Vi điều khiển ESP32 10](#_Toc201233413)

[1.3. Giao thức truyền thông ESP-NOW 11](#_Toc201233414)

[1.4. Nền tảng đám mây Google Firebase 12](#_Toc201233415)

[1.4.1. Dịch vụ Realtime Database 12](#_Toc201233416)

[1.4.2. Dịch vụ Firebase Hosting 13](#_Toc201233417)

[1.5. Trí tuệ nhân tạo tại biên (Edge AI) và Nền tảng Edge Impulse 13](#_Toc201233418)

[CHƯƠNG 2: THIẾT KẾ HỆ THỐNG 15](#_Toc201233419)

[2.1. Kiến trúc tổng quan 15](#_Toc201233420)

[2.2. Thiết kế phần cứng 16](#_Toc201233421)

[2.2.1. Thiết kế Node Cảm biến 16](#_Toc201233422)

[2.2.2. Thiết kế Gateway Trung tâm 17](#_Toc201233423)

[2.3. Thiết kế phần mềm 18](#_Toc201233424)

[2.3.1. Luồng hoạt động của Node Cảm biến 18](#_Toc201233425)

[2.3.2. Luồng hoạt động của Gateway Trung tâm 19](#_Toc201233426)

[2.3.3. Thiết kế Web Dashboard 20](#_Toc201233427)

[2.4. Thiết kế mô hình Trí tuệ nhân tạo 21](#_Toc201233428)

[2.4.1. Nền tảng sử dụng 21](#_Toc201233429)

[2.4.2. Dữ liệu đầu vào (Features) và Nhãn đầu ra (Labels) 22](#_Toc201233430)

[2.4.3. Quy trình huấn luyện và triển khai 22](#_Toc201233431)

[CHƯƠNG 3: TÍCH HỢP VÀ THỬ NGHIỆM HỆ THỐNG 25](#_Toc201233432)

[3.1. Kết quả tích hợp hệ thống 25](#_Toc201233433)

[3.1.1. Giao tiếp ESP-NOW 25](#_Toc201233434)

[3.1.2. Kết nối Firebase và Web Dashboard 25](#_Toc201233435)

[3.1.3. Điều khiển thiết bị và Chế độ AI 25](#_Toc201233436)

[3.2. Kết quả thử nghiệm mô hình AI 26](#_Toc201233437)

[3.3. Đánh giá hệ thống 27](#_Toc201233438)

[3.3.1. Ưu điểm 27](#_Toc201233439)

[3.3.2. Nhược điểm và hạn chế 27](#_Toc201233440)

[3.3.3. Hướng phát triển 27](#_Toc201233441)

[KẾT LUẬN 29](#_Toc201233442)

[TÀI LIỆU THAM KHẢO 30](#_Toc201233443)

# DANH MỤC VIẾT TẮT

|  |  |
| --- | --- |
| Viết tắt | Diễn giải |
| AI | Trí tuệ nhân tạo (Artificial Intelligence) |
| API | Giao diện lập trình ứng dụng (Application Programming Interface) |
| CDN | Mạng phân phối nội dung (Content Delivery Network) |
| CSS | Bảng kiểu xếp chồng (Cascading Style Sheets) |
| DHT22 | Cảm biến nhiệt độ và độ ẩm |
| ESP32 | Vi điều khiển System on a Chip (SoC) của Espressif Systems |
| ESP-NOW | Giao thức truyền thông không dây của Espressif |
| GPIO | Chân vào/ra đa dụng (General-purpose input/output) |
| HTML | Ngôn ngữ đánh dấu siêu văn bản (HyperText Markup Language) |
| I2C | Giao thức giao tiếp liên vi mạch (Inter-Integrated Circuit) |
| IoT | Internet vạn vật (Internet of Things) |
| JSON | Ký pháp đối tượng JavaScript (JavaScript Object Notation) |
| ML | Học máy (Machine Learning) |
| PIR | Cảm biến hồng ngoại thụ động (Passive Infrared) |
| RTDB | Cơ sở dữ liệu thời gian thực (Realtime Database) |
| SDK | Bộ công cụ phát triển phần mềm (Software Development Kit) |
| WSN | Mạng cảm biến không dây (Wireless Sensor Network) |

# MỤC LỤC BẢNG BIỂU

[**Bảng 2.1:** Danh sách linh kiện cho Node Cảm biến 16](#_Toc201233444)

[**Bảng 2.2:** Danh sách linh kiện cho Gateway Trung tâm 17](#_Toc201233445)

[**Bảng 3.1:** Kết quả thử nghiệm hiệu năng mô hình AI 26](#_Toc201233446)

# MỤC LỤC HÌNH ẢNH

[Hình 1.1 Kiến trúc WSN 9](#_Toc201443808)

[Hình 1.2 Sơ đồ chức năng chân ESP32 11](#_Toc201443809)

[Hình 1.3 Giao thức ESP NOW 12](#_Toc201443810)

[Hình 1.4 Firebase 13](#_Toc201443811)

[Hình 1.5 Tính năng Realtime Database 14](#_Toc201443812)

[Hình 1.6 Cách thức Firebase Hosting hoạt động 15](#_Toc201443813)

[Hình 1.7 Phân biệt AI cloud và Edge computing 16](#_Toc201443814)

[Hình 1.8 Quy trình thực hiện với Egde Impulse 16](#_Toc201443815)

[Hình 2.1 Kiến trúc tổng quan 17](#_Toc201443816)

[Hình 2.2 Sơ đồ khối Node cảm biến 19](#_Toc201443817)

[Hình 2.3 Sơ đồ nguyên lý Node cảm biêns 20](#_Toc201443818)

[Hình 2.4 Sơ đồ nguyên lý Gateway 21](#_Toc201443819)

[Hình 2.5 Lưu đồ thuật toán Node cảm biến 22](#_Toc201443820)

[Hình 2.6 Deploy thành công Web Dashboard 26](#_Toc201443821)

[Hình 2.7 Tạo Impulse 30](#_Toc201443822)

[Hình 3.1 Hệ thống thực tế. 32](#_Toc201443823)

[Hình 3.2 Giao diện Web DashBoard 33](#_Toc201443824)

[Hình 3.3 Log kết quả giao tiếp ESP-NOW 35](#_Toc201443825)

[Hình 3.4 Log kết quả của khối xử lý tại GateWay 35](#_Toc201443826)

[Hình 3.4 Kết quả cấu trúc dữ liệu cảm biến ở trên Firebase 36](#_Toc201443827)

[Hình 3.5 Kết quả cấu trúc dữ liệu điều khiển ở trên Firebase 37](#_Toc201443828)

# LỜI MỞ ĐẦU

Trong bối cảnh cuộc cách mạng công nghiệp 4.0 đang diễn ra mạnh mẽ, Internet vạn vật (IoT) và Trí tuệ nhân tạo (AI) đã và đang trở thành những công nghệ nền tảng, thúc đẩy sự chuyển đổi số trong mọi lĩnh vực của đời sống xã hội. Giáo dục, với vai trò là cái nôi đào tạo nguồn nhân lực cho tương lai, cũng không nằm ngoài xu hướng phát triển tất yếu này. Việc kiến tạo một môi trường học tập thông minh, tiện nghi, an toàn và tiết kiệm năng lượng không chỉ góp phần nâng cao chất lượng dạy và học, mà còn giúp sinh viên tiếp cận và làm chủ công nghệ, hình thành ý thức về việc sử dụng tài nguyên một cách bền vững.

Nhận thức được tầm quan trọng đó, nhóm chúng em đã thực hiện đồ án **"Hệ Thống Giám Sát và Điều Khiển Môi Trường Phòng Học Thông Minh Tích Hợp AI"**. Mục tiêu của đồ án là nghiên cứu, thiết kế và xây dựng một giải pháp IoT toàn diện, giải quyết các bài toán thực tiễn trong việc quản lý và vận hành phòng học hiện đại. Hệ thống không chỉ cho phép thu thập các thông số môi trường quan trọng như nhiệt độ, độ ẩm, ánh sáng và sự hiện diện của con người, mà còn cung cấp khả năng điều khiển các thiết bị điện từ xa thông qua một giao diện web trực quan.

Điểm nhấn đặc biệt của đồ án là việc tích hợp thành công mô hình Trí tuệ nhân tạo tại biên (Edge AI). Cách tiếp cận này cho phép hệ thống tự động đưa ra các quyết định điều khiển thông minh, tối ưu hóa sự thoải mái và hiệu quả năng lượng mà không phụ thuộc hoàn toàn vào kết nối Internet, qua đó tăng cường độ tin cậy và khả năng phản hồi tức thời.

Báo cáo này sẽ trình bày một cách chi tiết và hệ thống toàn bộ quá trình thực hiện đồ án của nhóm, từ việc nghiên cứu cơ sở lý thuyết, phân tích và lựa chọn công nghệ, đến thiết kế chi tiết phần cứng, phần mềm, tích hợp và thử nghiệm hệ thống. Qua đó, nhóm hy vọng sẽ cung cấp một cái nhìn sâu sắc và toàn diện về quá trình hiện thực hóa một sản phẩm IoT, cũng như khẳng định tính khả thi và tiềm năng ứng dụng của đề tài trong thực tiễn.

# CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

## 1.1. Mạng Cảm biến không dây (Wireless Sensor Networks - WSN)

Mạng cảm biến không dây (WSN) là một hệ thống mạng được cấu thành từ một số lượng lớn các thiết bị cảm biến phân tán trong không gian, có nhiệm vụ giám sát và ghi lại các điều kiện vật lý của môi trường, sau đó chuyển tiếp dữ liệu thu thập được đến một vị trí trung tâm. Một WSN điển hình bao gồm các node cảm biến, mỗi node là một thiết bị nhỏ gọn, thường được trang bị một hoặc nhiều cảm biến, một bộ vi xử lý, bộ nhớ và một bộ thu phát vô tuyến để giao tiếp không dây. Các node này giao tiếp với nhau và thường gửi dữ liệu đến một trạm gốc (Base Station) hoặc Gateway, nơi dữ liệu được xử lý và có thể được chuyển tiếp lên Internet.

A diagram of a network

AI-generated content may be incorrect.

Hình 1.1 Kiến trúc WSN

Đặc tính quan trọng của WSN là khả năng triển khai linh hoạt và khả năng tự tổ chức. Do không yêu cầu cơ sở hạ tầng cáp nối phức tạp, các node WSN có thể được triển khai với số lượng lớn trên một khu vực rộng hoặc trong những môi trường khắc nghiệt.Hơn nữa, các node cảm biến thường được thiết kế để hoạt động bằng pin, do đó, việc tối ưu hóa năng lượng tiêu thụ là một yêu cầu cốt lõi. Các node này có khả năng đi vào các chế độ ngủ sâu (deep sleep) và chỉ thức dậy để thực hiện đo đạc và truyền dữ liệu, giúp kéo dài tuổi thọ của mạng.

Với những đặc tính ưu việt đó, WSN được ứng dụng rộng rãi trong nhiều lĩnh vực như giám sát môi trường, nông nghiệp chính xác, y tế, tự động hóa công nghiệp và quân sự. Trong khuôn khổ đồ án này, nhóm chúng em đã ứng dụng mô hình WSN để xây dựng một mạng lưới các node cảm biến không dây trong phòng học, cho phép thu thập dữ liệu môi trường một cách hiệu quả và đồng bộ.

## 1.2. Vi điều khiển ESP32

Sau quá trình tìm hiểu và phân tích các lựa chọn vi điều khiển cho ứng dụng IoT, nhóm chúng em đã quyết định lựa chọn ESP32 làm "bộ não" trung tâm cho cả Node Cảm biến và Gateway của hệ thống. ESP32 là một dòng vi điều khiển trên chip (SoC) mạnh mẽ, chi phí thấp và tiết kiệm năng lượng do Espressif Systems phát triển, được xem là sự kế thừa và nâng cấp vượt trội so với thế hệ ESP8266 trước đó.

Sự lựa chọn này dựa trên những tính năng kỹ thuật ưu việt của ESP32, hoàn toàn phù hợp với các yêu cầu của một hệ thống IoT hiện đại. Đầu tiên, ESP32 tích hợp sẵn kết nối Wi-Fi 802.11 b/g/n và Bluetooth chế độ kép (Classic và BLE), giúp việc kết nối các thiết bị với mạng cục bộ và Internet trở nên dễ dàng mà không cần các module mở rộng phức tạp. Về hiệu năng, ESP32 sở hữu bộ vi xử lý Xtensa lõi kép 32-bit LX6, có thể hoạt động với tốc độ lên đến 240 MHz, cung cấp đủ sức mạnh để xử lý đồng thời nhiều tác vụ: từ việc thu thập dữ liệu từ nhiều cảm biến, thực hiện giao thức truyền thông, cho đến chạy các mô hình AI tại biên.

A diagram of a microchip

AI-generated content may be incorrect.

Hình 1.2 Sơ đồ chức năng chân ESP32

Một yếu tố quan trọng khác đối với các Node Cảm biến không dây là khả năng quản lý năng lượng. ESP32 hỗ trợ các chế độ ngủ sâu (deep sleep) với dòng tiêu thụ chỉ ở mức vài micro-amperes (µA), cho phép các node hoạt động bằng pin trong thời gian dài. Ngoài ra, với số lượng chân GPIO dồi dào và hỗ trợ đa dạng các giao thức ngoại vi như I2C, SPI, UART, ADC, ESP32 mang lại sự linh hoạt tối đa trong việc kết nối và giao tiếp với các loại cảm biến và thiết bị chấp hành khác nhau. Chính vì những lý do trên, ESP32 đã trở thành nền tảng phần cứng lý tưởng để nhóm chúng em hiện thực hóa đồ án này.

## 1.3. Giao thức truyền thông ESP-NOW

Để giải quyết bài toán giao tiếp hiệu quả năng lượng và có độ trễ thấp giữa các Node Cảm biến và Gateway, nhóm đã quyết định sử dụng ESP-NOW. Đây là một giao thức truyền thông không dây độc quyền được phát triển bởi chính Espressif, cho phép các thiết bị ESP32 giao tiếp trực tiếp với nhau mà không cần thông qua một điểm truy cập (Access Point) trung gian như router Wi-Fi.

Several electronic components with red arrows

AI-generated content may be incorrect.

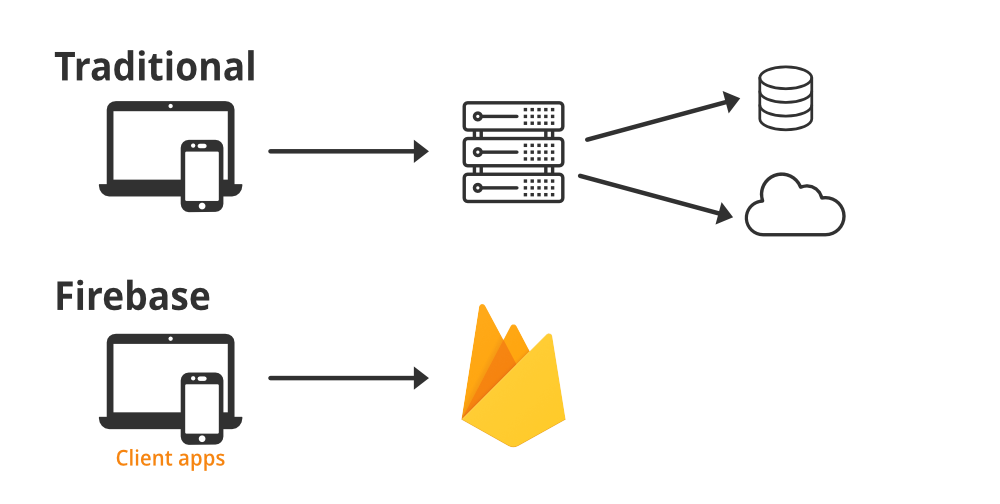
Hình 1.3 Giao thức ESP NOW

Ưu điểm kỹ thuật lớn nhất của ESP-NOW là tốc độ phản hồi rất nhanh. Bằng cách hoạt động trực tiếp ở lớp liên kết dữ liệu (data-link layer) và loại bỏ các lớp giao thức phức tạp của TCP/IP, thời gian truyền một gói tin ngắn (tối đa 250 bytes) được giảm thiểu đáng kể, rất phù hợp cho các ứng dụng điều khiển thời gian thực như bật đèn khi phát hiện chuyển động. Hơn nữa, vì là giao thức phi kết nối, các Node Cảm biến có thể gửi dữ liệu và quay trở lại chế độ ngủ sâu ngay lập tức, giúp tiết kiệm năng lượng một cách tối đa, đây là yếu tố sống còn đối với các thiết bị chạy bằng pin.

Mô hình lập trình của ESP-NOW cũng tương đối đơn giản, chỉ yêu cầu biết địa chỉ MAC của thiết bị nhận, giúp giảm độ phức tạp của firmware trên các node. Giao thức này hỗ trợ hiệu quả các cấu trúc liên kết mạng như một-nhiều và nhiều-một, hoàn toàn phù hợp với kiến trúc của đồ án, nơi nhiều Node Cảm biến cùng gửi dữ liệu về một Gateway trung tâm duy nhất.

## 1.4. Nền tảng đám mây Google Firebase

Để xây dựng phần backend cho hệ thống, nhóm chúng em đã lựa chọn Google Firebase, một nền tảng phát triển ứng dụng toàn diện được cung cấp bởi Google. Firebase cung cấp một bộ công cụ và dịch vụ backend mạnh mẽ theo mô hình BaaS (Backend-as-a-Service), cho phép nhóm tập trung vào việc phát triển các tính năng của ứng dụng mà không cần phải tự xây dựng và quản lý cơ sở hạ tầng máy chủ phức tạp. Trong khuôn khổ đồ án, hai dịch vụ chính của Firebase đã được sử dụng là Realtime Database và Firebase Hosting.



Hình 1.4 Firebase

### 1.4.1. Dịch vụ Realtime Database

Để đồng bộ hóa dữ liệu một cách tức thời giữa Gateway, các thiết bị điều khiển và giao diện người dùng, nhóm đã lựa chọn dịch vụ Firebase Realtime Database (RTDB). Đây là một cơ sở dữ liệu NoSQL được lưu trữ trên đám mây, nơi dữ liệu được tổ chức dưới dạng một cây JSON lớn.

Điểm mạnh cốt lõi và cũng là lý do chính nhóm chọn RTDB là khả năng đồng bộ hóa dữ liệu theo thời gian thực.Thay vì mô hình yêu cầu-phản hồi HTTP truyền thống, RTDB sử dụng cơ chế lắng nghe sự kiện. Bất cứ khi nào dữ liệu tại một node trong cây JSON thay đổi, tất cả các client đang lắng nghe tại node đó sẽ nhận được bản cập nhật gần như ngay lập tức, thường chỉ trong vài mili giây. Chính cơ chế này đã giúp Web Dashboard của đồ án có thể hiển thị các chỉ số cảm biến trực tiếp mà không cần người dùng phải tải lại trang. Ngoài ra, SDK của Firebase còn hỗ trợ khả năng hoạt động ngoại tuyến (offline) một cách tự động, giúp ứng dụng vẫn phản hồi ngay cả khi mất kết nối Internet.

A diagram of a software application

AI-generated content may be incorrect.

Hình 1.5 Tính năng Realtime Database

### 1.4.2. Dịch vụ Firebase Hosting

Giao diện Web Dashboard của đồ án được triển khai và phân phối đến người dùng cuối thông qua dịch vụ Firebase Hosting. Đây là một dịch vụ lưu trữ web tĩnh và động cấp sản xuất, được quản lý toàn diện bởi Google, mang lại nhiều lợi ích cho việc triển khai ứng dụng web.

Một trong những ưu điểm lớn nhất là tốc độ phân phối nội dung nhanh chóng. Mọi tệp tĩnh (HTML, CSS, JavaScript) mà nhóm tải lên đều được tự động lưu vào bộ nhớ đệm trên các ổ SSD tại các điểm cuối của Mạng phân phối nội dung (CDN) trên toàn cầu. Điều này đảm bảo rằng người dùng ở bất kỳ đâu trên thế giới cũng có thể tải trang web với độ trễ thấp nhất. Bên cạnh đó, Firebase Hosting cung cấp chứng chỉ SSL miễn phí và tự động gia hạn cho tên miền, đảm bảo mọi lưu lượng truy cập giữa người dùng và trang web đều được mã hóa qua giao thức HTTPS, tăng cường tính bảo mật và tin cậy cho hệ thống.

A diagram of a firebase hosting

AI-generated content may be incorrect.

Hình 1.6 Cách thức Firebase Hosting hoạt động

## 1.5. Trí tuệ nhân tạo tại biên (Edge AI) và Nền tảng Edge Impulse

Thay vì xử lý AI trên các máy chủ đám mây, đồ án này áp dụng một cách tiếp cận tiên tiến hơn là Trí tuệ nhân tạo tại biên (Edge AI). Trí tuệ nhân tạo tại biên (Edge AI) là mô hình tính toán trong đó các thuật toán học máy (Machine Learning) hoặc học sâu (Deep Learning) được triển khai và thực thi trực tiếp trên các thiết bị phần cứng biên (edge devices), chẳng hạn như vi điều khiển, bo mạch nhúng (ESP32, Raspberry Pi), hoặc các thiết bị IoT. Khác với mô hình truyền thống vốn phụ thuộc vào việc gửi dữ liệu lên đám mây để xử lý, Edge AI cho phép xử lý cục bộ ngay tại nơi dữ liệu được tạo ra.

Việc lựa chọn kiến trúc Edge AI mang lại nhiều lợi ích kỹ thuật vượt trội cho một hệ thống điều khiển thời gian thực. Thứ nhất, nó giúp giảm độ trễ một cách đáng kể, cho phép đưa ra quyết định gần như tức thời. Thứ hai, hệ thống có thể hoạt động độc lập ngay cả khi mất kết nối Internet, tăng cường độ tin cậy.Cuối cùng, Edge AI giúp tăng cường bảo mật và quyền riêng tư vì dữ liệu cảm biến nhạy cảm được xử lý tại chỗ, đồng thời giúp tiết kiệm băng thông và chi phí liên quan đến dịch vụ đám mây.

A diagram of a computer chip

AI-generated content may be incorrect.

Hình 1.7 Phân biệt AI cloud và Edge computing

Để hiện thực hóa ý tưởng này một cách hiệu quả, nhóm đã sử dụng nền tảng Edge Impulse. Đây là một nền tảng MLOps (Machine Learning Operations) hàng đầu, được thiết kế chuyên biệt để xây dựng, huấn luyện và triển khai các mô hình học máy (TinyML) trên các thiết bị nhúng có tài nguyên hạn chế như ESP32. Edge Impulse cung cấp một quy trình làm việc trực quan và toàn diện, từ việc thu thập dữ liệu, thiết kế pipeline xử lý, huấn luyện mô hình cho đến việc xuất ra thư viện C++ được tối ưu hóa, giúp đơn giản hóa đáng kể quá trình tích hợp AI vào firmware của Gateway.

A diagram of a data processing process

AI-generated content may be incorrect.

Hình 1.8 Quy trình thực hiện với Egde Impulse

# CHƯƠNG 2: THIẾT KẾ HỆ THỐNG

## 2.1. Kiến trúc tổng quan

Dựa trên các công nghệ đã được nghiên cứu và lựa chọn ở chương 1, nhóm chúng em đã xây dựng một kiến trúc hệ thống phân tán gồm nhiều lớp, đảm bảo tính module hóa, dễ dàng mở rộng và bảo trì. Luồng dữ liệu và điều khiển của hệ thống được thể hiện qua sơ đồ kiến trúc tổng quan dưới đây.

A diagram of a computer

AI-generated content may be incorrect.

Hình 2.1 Kiến trúc tổng quan

Kiến trúc này bao gồm các thành phần chính sau:

* **Các Node Cảm biến:** Được triển khai tại nhiều vị trí trong phòng học, mỗi node sử dụng vi điều khiển ESP32 kết hợp với các cảm biến để thu thập dữ liệu môi trường (nhiệt độ, độ ẩm, ánh sáng, chuyển động).
* **Mạng giao tiếp cục bộ:** Các node cảm biến tạo thành một mạng cảm biến không dây (WSN) cục bộ, sử dụng giao thức ESP-NOW để gửi dữ liệu đến Gateway một cách nhanh chóng và tiết kiệm năng lượng.
* **Gateway Trung tâm:** Là trái tim của hệ thống, Gateway cũng được xây dựng trên nền tảng ESP32. Nó có hai nhiệm vụ chính: (1) nhận dữ liệu từ tất cả các node cảm biến qua ESP-NOW và (2) kết nối với mạng Wi-Fi để đẩy dữ liệu lên Firebase và nhận lệnh điều khiển từ xa. Đây cũng là nơi mô hình Edge AI được triển khai để xử lý logic điều khiển tự động.
* **Nền tảng Đám mây:** Nhóm sử dụng Google Firebase làm backend. Dịch vụ Realtime Database đóng vai trò là kho lưu trữ và đồng bộ hóa trạng thái thời gian thực của hệ thống. Dịch vụ Firebase Hosting được dùng để triển khai giao diện web.
* **Giao diện Người dùng:** Một Web Dashboard được thiết kế để người dùng có thể giám sát các thông số môi trường và điều khiển các thiết bị (đèn, quạt) từ bất kỳ đâu thông qua trình duyệt web.

Luồng dữ liệu chính của hệ thống bắt đầu từ các Node Cảm biến, truyền qua ESP-NOW đến Gateway, sau đó được Gateway đẩy lên Firebase Realtime Database và cuối cùng hiển thị trên Web Dashboard. Luồng điều khiển đi theo chiều ngược lại: người dùng ra lệnh từ Web Dashboard, lệnh được ghi vào Firebase, Gateway lắng nghe sự thay đổi này và thực thi lệnh điều khiển các thiết bị vật lý.

## 2.2. Thiết kế phần cứng

### 2.2.1. Thiết kế Node Cảm biến

Mỗi Node Cảm biến được nhóm chúng em thiết kế với tiêu chí nhỏ gọn, tiêu thụ ít năng lượng và có khả năng thu thập đầy đủ các thông số môi trường cần thiết. Việc lựa chọn linh kiện được thực hiện dựa trên sự cân bằng giữa hiệu năng, chi phí và mức độ phổ biến.

* **Sơ đồ khối Node cảm biến:**

A diagram of a computer network

AI-generated content may be incorrect.

Hình 2.2 Sơ đồ khối Node cảm biến

* **Danh sách linh kiện:** Các linh kiện chính cho một Node Cảm biến được nhóm lựa chọn bao gồm:

Bảng 2.1: Danh sách linh kiện cho Node Cảm biến

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Linh kiện** | **Mô tả** | **Số lượng** |
| 1 | Vi điều khiển ESP32 | Trung tâm xử lý, thu thập và truyền dữ liệu. Lựa chọn kit ESP32-WROOM-32 DevKit. | 1 |
| 2 | Cảm biến DHT22 | Đo nhiệt độ và độ ẩm môi trường, giao tiếp tín hiệu số. | 1 |
| 3 | Cảm biến PIR HC-SR501 | Phát hiện chuyển động của con người dựa trên bức xạ hồng ngoại. | 1 |
| 4 | Cảm biến ánh sáng BH1750 | Đo cường độ ánh sáng theo đơn vị lux, giao tiếp qua I2C. | 1 |

* **Sơ đồ nguyên lý:** Sơ đồ kết nối chi tiết giữa vi điều khiển ESP32 và các cảm biến được thể hiện như sau:

A circuit board with wires and a ball

AI-generated content may be incorrect.

Hình 2.3 Sơ đồ nguyên lý Node cảm biêns

### 2.2.2. Thiết kế Gateway Trung tâm

Gateway là thành phần trung tâm, có vai trò quan trọng trong việc kết nối và xử lý dữ liệu. Thiết kế của Gateway được nhóm tập trung vào việc đảm bảo độ ổn định và hiệu năng cao để có thể thực hiện đồng thời nhiều tác vụ.

* **Sơ đồ nguyên lý khối Gateway Trung tâm:**

A computer chip with many wires

AI-generated content may be incorrect.

Hình 2.4 Sơ đồ nguyên lý Gateway

* **Danh sách linh kiện:**

**Bảng 2.2:** Danh sách linh kiện cho Gateway Trung tâm

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Linh kiện** | **Mô tả** | **Số lượng** |
| 1 | Vi điều khiển ESP32 | Trung tâm xử lý, nhận dữ liệu ESP-NOW, kết nối Wi-Fi, giao tiếp Firebase và chạy mô hình AI. | 1 |
| 2 | Module Relay 8 kênh | Điều khiển bật/tắt các hệ thống đèn, có opto cách ly để bảo vệ vi điều khiển. | 1 |
| 3 | Nguồn cấp | Adapter 5V-2A để cấp nguồn ổn định cho ESP32 và các module relay. | 1 |

## 2.3. Thiết kế phần mềm

### 2.3.1. Luồng hoạt động của Node Cảm biến

Firmware của Node Cảm biến được nhóm chúng em lập trình trên nền tảng Arduino IDE, với mục tiêu cốt lõi là tối ưu hóa việc sử dụng năng lượng. Để đạt được điều này, node hoạt động theo một chu trình tuần hoàn: thức dậy, đọc dữ liệu, gửi dữ liệu và sau đó đi vào chế độ ngủ sâu.

* **Lưu đồ thuật toán:**

A flowchart of a program

AI-generated content may be incorrect.

Hình 2.5 Lưu đồ thuật toán Node cảm biến

Quá trình hoạt động của node được mô tả chi tiết như sau. Khi khởi động, trong hàm setup(), vi điều khiển sẽ cấu hình các chân GPIO, khởi tạo các đối tượng cảm biến (DHT22, PIR, BH1750) và thiết lập giao thức ESP-NOW, đăng ký địa chỉ MAC của Gateway làm thiết bị nhận. Sau khi khởi tạo xong, node sẽ đi vào vòng lặp chính loop(). Tại đây, nó sẽ lần lượt đọc giá trị từ các cảm biến. Dữ liệu sau khi đọc sẽ được đóng gói vào một cấu trúc (struct) đã được định nghĩa trước để đảm bảo tính nhất quán.

|  |
| --- |
| // Cấu trúc dữ liệu được gửi từ Node Cảm biến đến Gateway  **typedef** struct struct\_message **{**  int id**;**  float temperature**;**  float humidity**;**  float light\_intensity**;**  bool motion\_detected**;**  **}** struct\_message**;** |

Để đảm bảo tính toàn vẹn dữ liệu, nhóm đã thêm các bước kiểm tra lỗi; ví dụ, nếu đọc cảm biến thất bại, một giá trị đặc biệt (-999.0f) sẽ được gán vào trường dữ liệu tương ứng. Dữ liệu sau đó được gửi đi bằng hàm

esp\_now\_send(). Ngay sau khi gửi thành công, node sẽ gọi hàm esp\_deep\_sleep() để đi vào chế độ ngủ sâu trong một khoảng thời gian đã định trước, giúp tiết kiệm năng lượng tối đa. Chu trình này sẽ lặp lại khi hết thời gian ngủ.

|  |
| --- |
| // Đưa ESP32 vào chế độ ngủ sâu để tiết kiệm năng lượng  esp\_deep\_sleep**(**SLEEP\_DURATION\_SECONDS **\*** 1000000ULL**);** |

### 2.3.2. Luồng hoạt động của Gateway Trung tâm

Khác với các node cảm biến, Gateway là thành phần phải hoạt động liên tục để đảm bảo vai trò cầu nối và xử lý trung tâm. Firmware của Gateway được thiết kế để thực hiện đồng thời nhiều tác vụ một cách hiệu quả.

Trong hàm setup(), Gateway sẽ khởi tạo giao thức ESP-NOW ở chế độ nhận, đồng thời kết nối vào mạng Wi-Fi cục bộ và thiết lập kết nối đến Firebase Realtime Database. Các chân GPIO điều khiển relay cũng được cấu hình là chân OUTPUT.

Trong vòng lặp chính loop(), Gateway thực hiện các nhiệm vụ song song. Nhóm đã đăng ký một hàm callback OnDataRecv để xử lý các gói tin nhận được từ các Node Cảm biến.

|  |
| --- |
| // Hàm callback được gọi khi Gateway nhận được dữ liệu qua ESP-NOW  void OnDataRecv**(**const uint8\_t **\***mac\_addr**,** const uint8\_t **\***data**,** int len**)** **{**  // Sao chép dữ liệu nhận được vào biến cục bộ  memcpy**(&**incomingData**,** data**,** **sizeof(**incomingData**));**  // Đặt cờ báo có dữ liệu mới để xử lý trong vòng lặp chính  newData **=** **true;**  **}** |

Khi có dữ liệu mới, Gateway sẽ phân tích gói tin và đẩy dữ liệu này lên cây dữ liệu tương ứng trên Firebase. Đồng thời, Gateway cũng đăng ký các trình lắng nghe (listeners) trên đường dẫn /controls của Firebase. Khi có bất kỳ sự thay đổi nào trên đường dẫn này (do người dùng ra lệnh từ Web Dashboard), Gateway sẽ nhận được thông báo và thực thi lệnh điều khiển bật/tắt các relay tương ứng. Trong trường hợp chế độ tự động được kích hoạt, Gateway sẽ liên tục đưa dữ liệu cảm biến mới nhất vào mô hình AI đã được tích hợp sẵn để nhận về dự đoán và tự động điều khiển các thiết bị.

### 2.3.3. Thiết kế Web Dashboard

Để người dùng có thể tương tác với hệ thống, nhóm chúng em đã xây dựng một giao diện Web Dashboard bằng các công nghệ web cơ bản là HTML, CSS và JavaScript.Giao diện này được triển khai bằng dịch vụ Firebase Hosting, cho phép truy cập từ bất kỳ đâu có Internet.

Thiết kế của dashboard hướng đến sự trực quan, dễ sử dụng và có khả năng đáp ứng (responsive) để hiển thị tốt trên nhiều kích thước màn hình khác nhau. Trái tim của dashboard là tệp script.js, nơi xử lý toàn bộ logic kết nối và tương tác với Firebase.Khi trang được tải, script sẽ khởi tạo kết nối đến Firebase bằng cấu hình dự án.

|  |
| --- |
| **const** firebaseConfig **=** **{**  apiKey**:** "AIzaSyAH76sndFX2iDnoJq8aiDVBRvyJFerP4Yo"**,**  authDomain**:** "espproject-ccd63.firebaseapp.com"**,**  databaseURL**:** "https://espproject-ccd63-default-rtdb.asia-southeast1.firebasedatabase.app/"**,**  projectId**:** "espproject-ccd63"**,**  storageBucket**:** "espproject-ccd63.appspot.com"**,**  messagingSenderId**:** "..."**,**  appId**:** "..."  **};**  // --- KHỞI TẠO ---  firebase**.**initializeApp**(**firebaseConfig**);**  **const** database **=** firebase**.**database**();** |

Sau đó, script sử dụng hàm database.ref().on('value',...) để đăng ký các trình lắng nghe thời gian thực trên các đường dẫn dữ liệu /sensorData và /controls.

|  |
| --- |
| // Lắng nghe dữ liệu từ Firebase cho từng Node  **for** **(let** i **=** 1**;** i **<=** TOTAL\_NODES**;** i**++)** **{**  **const** nodePath **=** ***`/sensorData/node*${**i**}*`*;**  database**.**ref**(**nodePath**).**on**(**'value'**,** **(**snapshot**)** **=>** **{**  **const** data **=** snapshot**.**val**();**  updateNodeCard**(**i**,** data**);**  **});**  **}** |

Nhờ vậy, mỗi khi có dữ liệu mới từ cảm biến hoặc trạng thái thiết bị thay đổi, giao diện sẽ tự động cập nhật các thông số và trạng thái công tắc mà không cần người dùng phải tải lại trang.

* Quy trình triển khai Web Dashboard:

Nhóm đã sử dụng công cụ dòng lệnh Firebase CLI để triển khai ứng dụng web lên Firebase Hosting. Quy trình bao gồm các bước sau:

Bước 1: Cài đặt Firebase Tools

|  |
| --- |
| npm install -g firebase-tools |

Bước 2: Đăng nhập vào Firebase

|  |
| --- |
| firebase login |

Bước 3: Khởi tạo Firebase trong thư mục dự án web

Dùng lệnh cd để di chuyển vào đúng thư mục chứa các file index.html, style.css.

|  |
| --- |
| firebase init |

Tiếp theo thực hiện như sau :

*Are you ready to proceed? (Y/n)* -> Gõ Y và nhấn Enter.

*Which Firebase features do you want to set up?* -> Dùng phím mũi tên xuống, di chuyển đến dòng Hosting: Configure files for Firebase Hosting..., nhấn phím Space (phím cách) để chọn nó (sẽ có dấu \* hiện ra), sau đó nhấn Enter.

*Please select an option:* -> Chọn Use an existing project và nhấn Enter.

Chọn project espproject-ccd63 và nhấn Enter.

*What do you want to use as your public directory? (public)* -> Nhấn Enter để chấp nhận tên mặc định là public.

*Configure as a single-page app (rewrite all urls to /index.html)? (y/N)* -> Gõ N và nhấn Enter.

*Set up automatic builds and deploys with GitHub? (y/N)* -> Gõ N và nhấn Enter để đơn giản hóa.

Sau khi hoàn tất, ta sẽ thấy một file firebase.json, .firebaserc và một thư mục mới tên là public được tạo ra trong thư mục dự án.

Bước 4: Di chuyển file vào thư mục public

Bước 5: Deploy (Tải lên) trang web!

|  |
| --- |
| firebase deploy |

A computer screen shot of a program

AI-generated content may be incorrect.

Hình 2.6 Deploy thành công Web Dashboard

## 2.4. Thiết kế mô hình Trí tuệ nhân tạo

### 2.4.1. Nền tảng sử dụng

Để triển khai mô hình AI tại biên trên một thiết bị có tài nguyên hạn chế như ESP32, việc lựa chọn một nền tảng phù hợp là vô cùng quan trọng. Sau khi nghiên cứu, nhóm chúng em đã quyết định sử dụng Edge Impulse. Đây là một nền tảng MLOps hàng đầu, được thiết kế chuyên biệt để xây dựng, huấn luyện và triển khai các mô hình học máy (TinyML) trên các vi điều khiển. Edge Impulse cung cấp một quy trình làm việc trực quan và toàn diện, từ việc thu thập dữ liệu, thiết kế pipeline xử lý, huấn luyện mô hình cho đến việc xuất ra thư viện C++ được tối ưu hóa, giúp việc tích hợp AI vào firmware của Gateway trở nên khả thi và hiệu quả.

### 2.4.2. Dữ liệu đầu vào (Features) và Nhãn đầu ra (Labels)

Để mô hình có thể học và đưa ra quyết định, nhóm đã xác định rõ ràng dữ liệu đầu vào và đầu ra như sau:

* **Dữ liệu đầu vào (Features):** Đây là các giá trị được thu thập từ các Node Cảm biến, là cơ sở để mô hình phân tích bối cảnh môi trường. Các đặc trưng này bao gồm:
* Nhiệt độ (Temperature)
* Độ ẩm (Humidity)
* Cường độ ánh sáng (Light Intensity)
* Trạng thái chuyển động (Motion Detected)
* **Nhãn đầu ra (Labels):** Đây là các hành động điều khiển mong muốn mà mô hình cần dự đoán dựa trên các đặc trưng đầu vào. Nhóm đã định nghĩa các lớp (class) đầu ra cụ thể, ví dụ:
* can\_bat\_den
* can\_bat\_quat
* can\_bat\_den\_va\_quat
* phong\_trong (phòng trống, tắt hết thiết bị)
* binh\_thuong (không cần hành động)

### 2.4.3. Quy trình huấn luyện và triển khai

Toàn bộ quy trình xây dựng mô hình AI được nhóm thực hiện trên nền tảng Edge Impulse Studio, bao gồm các bước chính sau:

#### a. Thu thập dữ liệu (Data Acquisition)

Nhóm tiến hành thu thập các chuỗi dữ liệu cảm biến trong nhiều điều kiện môi trường thực tế khác nhau (ví dụ: phòng tối có người, phòng nóng có người, phòng trống,...) và gán nhãn đầu ra tương ứng cho từng mẫu dữ liệu. Dữ liệu được tải lên Edge Impulse Studio để chuẩn bị cho quá trình huấn luyện.

Quy trình thu thập dữ liệu và tạo đầu vào cho Edge Impulse Studio gồm thực hiện đồng thời 2 bước sau :

Bước 1 : Esp thu thập dữ liệu từ cảm biến sau đó đẩy lên cổng Serial theo đúng định dạng :

|  |
| --- |
| float t = dht.readTemperature();  float h = dht.readHumidity();  float l = lightMeter.readLightLevel();  bool m = digitalRead(PIRPIN);  **if** (!isnan(t) **&&** !isnan(h) **&&** l **>**= 0) {  // **In** ra một dòng duy nhất, các giá trị cách nhau bởi dấu phẩy  Serial.printf("**%.2f,%**.2f,**%.2f,%**d\n", t, h, l, m **?** 1 : 0);  }  delay(1000); // Gửi dữ liệu 1 giây một lần |

Bước 2 : Sử dụng chương trình Python để đọc dữ liệu từ Serial và chuyển qua định dạng Json theo chuẩn của Edge Impulse Studio.

|  |
| --- |
| **def** create\_json\_file**(**values**,** label**,** index**):**  payload **=** **{** "device\_name"**:** "ESP32\_Live"**,** "device\_type"**:** "ESP32\_Classifier"**,** "interval\_ms"**:** 1000**,** "sensors"**:** **[{**"name"**:** "temperature"**,** "units"**:** "Cel"**},** **{**"name"**:** "humidity"**,** "units"**:** "%"**},** **{**"name"**:** "light\_intensity"**,** "units"**:** "lx"**},** **{**"name"**:** "motion"**,** "units"**:** "0/1"**}],** "values"**:** **[**values**]** **}**  encoded **=** json**.***dumps***(**payload**,** separators**=(**','**,** ':'**))**  signature **=** hmac**.***new***(bytes(**EMPTY\_HMAC\_KEY**,** 'utf-8'**),** msg**=**encoded**.***encode***(**'utf-8'**),** digestmod**=**hashlib**.***sha256***).***hexdigest***()**  final\_json **=** **{** "protected"**:** **{**"ver"**:** "v1"**,** "alg"**:** "HS256"**,** "iat"**:** **int(**time**.***time***())},** "signature"**:** signature**,** "payload"**:** payload **}**  label\_dir **=** os**.***path***.***join***(**OUTPUT\_FOLDER**,** label**)**  **if** **not** os**.***path***.***exists***(**label\_dir**):** os**.***makedirs***(**label\_dir**)**  filename **=** os**.***path***.***join***(**label\_dir**,** f"{label}.{**int(**time**.***time***()** **\*** 1000**)**}.json"**)**  **with** **open(**filename**,** 'w'**)** **as** f**:** json**.***dump***(**final\_json**,** f**)**  **print(**f" -> Da luu file: {filename}"**)** |

Hàm create\_json\_file() có chức năng nhận vào một mẫu dữ liệu cảm biến thô (bao gồm 4 giá trị: nhiệt độ, độ ẩm, ánh sáng, chuyển động) và một nhãn (ví dụ: "can\_bat\_den"), sau đó thực hiện việc chuẩn hóa và đóng gói toàn bộ thông tin đó vào một file .json duy nhất với cấu trúc chuẩn do Edge Impulse yêu cầu. Các bước nhỏ gồm :

1. Tạo đối tượng payload:

Đây là đối tượng chính chứa dữ liệu. Nó bao gồm các thông tin về thiết bị, tên và đơn vị của từng cảm biến, và quan trọng nhất là mảng values chứa các giá trị thực tế của cảm biến tại một thời điểm.

|  |
| --- |
| payload = {  "device\_name": "ESP32\_Live\_Collector",  "device\_type": "ESP32\_Classifier",  "interval\_ms": 1000,  "sensors": [  {"name": "temperature", "units": "Cel"},  {"name": "humidity", "units": "%"},  {"name": "light\_intensity", "units": "lx"},  {"name": "motion", "units": "0/1"}  ],  "values": [  ]  } |

2. Tạo Chữ ký Bảo mật (HMAC Signature):

Để đảm bảo tính toàn vẹn của dữ liệu, Edge Impulse yêu cầu mỗi gói tin phải có một chữ ký điện tử. Script sử dụng thuật toán HMAC-SHA256 để tạo ra một chuỗi chữ ký duy nhất dựa trên nội dung của payload.

|  |
| --- |
| # Chuyển payload thành chuỗi JSON không có khoảng trắng  encoded = json.dumps(payload, separators=(',', ':'))  # Tạo chữ ký HMAC từ chuỗi đã mã hóa  signature = hmac.new(  bytes(EMPTY\_HMAC\_KEY, 'utf-8'),  msg=encoded.encode('utf-8'),  digestmod=hashlib.sha256  ).hexdigest() |

3. Đóng gói File JSON Hoàn Chỉnh:

Cuối cùng, script sẽ tạo một đối tượng JSON hoàn chỉnh bao gồm 3 phần: protected (chứa thông tin về thuật toán), signature (chữ ký vừa tạo), và payload (dữ liệu chính).

|  |
| --- |
| final\_json = {  "protected": {  "ver": "v1",  "alg": "HS256",  "iat": int(time.time()) # Ghi lại thời điểm tạo file  },  "signature": signature,  "payload": payload  } |

#### b. Thiết kế Impulse (Impulse Design):

Một "Impulse" trong Edge Impulse là một pipeline (đường ống) xử lý, nó định nghĩa cách dữ liệu đầu vào sẽ được xử lý và mô hình AI sẽ học từ nó như thế nào. Dựa trên đặc thù của bài toán, nhóm đã thiết kế một Impulse đơn giản nhưng hiệu quả.

Khối Đầu vào (Input features): Cấu hình để nhận một cửa sổ dữ liệu với 4 đặc trưng (features) chính từ các cảm biến, bao gồm: temperature, humidity, light\_intensity, và motion.

Khối Xử lý (Processing Block): Nhóm đã sử dụng khối "Raw data". Khối này có chức năng giữ nguyên 4 giá trị gốc từ cảm biến và đưa trực tiếp vào mô hình học máy mà không cần qua các bước xử lý tín hiệu phức tạp.

Khối Học máy (Learning Block): Nhóm sử dụng khối "Classification (Keras)" để xây dựng một mạng nơ-ron nhân tạo. Mạng này có nhiệm vụ học mối quan hệ phức tạp giữa 4 giá trị cảm biến đầu vào và 5 trạng thái phòng học đầu ra đã được gán nhãn.

A screenshot of a computer

AI-generated content may be incorrect.

Hình 2.7 Tạo Impulse

#### c. Huấn luyện và Kiểm thử (Training & Testing)

Sau khi thiết kế Impulse, mô hình được huấn luyện với bộ dữ liệu Training đã được chuẩn bị. Nền tảng Edge Impulse cung cấp một giao diện trực quan để theo dõi quá trình huấn luyện.

Khi quá trình huấn luyện hoàn tất, mô hình được đánh giá hiệu năng trên bộ dữ liệu Testing (dữ liệu từ Node 3 mà mô hình chưa từng thấy trước đây). Kết quả đánh giá, bao gồm ma trận nhầm lẫn (confusion matrix) và độ chính xác tổng thể, được sử dụng để đo lường khả năng tổng quát hóa và độ tin cậy của mô hình trong các kịch bản thực tế.

#### d. Triển khai (Deployment)

Lựa chọn: Tùy chọn "Arduino library" được lựa chọn để đóng gói toàn bộ mô hình, các hàm xử lý và các hàm thực thi cần thiết.

Tích hợp: Thư viện Arduino được tải về dưới dạng file .zip, sau đó được cài đặt vào Arduino IDE và #include vào chương trình của Gateway. Điều này biến Gateway thành một thiết bị Edge AI, có khả năng thực thi AI trực tiếp mà không cần gửi dữ liệu lên cloud để phân tích

|  |
| --- |
| // Chuẩn bị dữ liệu đầu vào cho mô hình  signal\_t signal**;**  float features **=** **{** 0 **};**  features **=** dataToProcess**.**temperature**;**  features**[**1**]** **=** dataToProcess**.**humidity**;**  features**[**2**]** **=** dataToProcess**.**light\_intensity**;**  features**[**3**]** **=** dataToProcess**.**motion\_detected**?** 1.0f **:** 0.0f**;**  // Tạo tín hiệu từ buffer  numpy**::**signal\_from\_buffer**(**features**,** EI\_CLASSIFIER\_DSP\_INPUT\_FRAME\_SIZE**,** **&**signal**);**  // Chạy bộ phân loại  ei\_impulse\_result\_t result **=** **{** 0 **};**  run\_classifier**(&**signal**,** **&**result**,** **false);**  // Xử lý kết quả dự đoán và điều khiển thiết bị  //...  **}** |

# CHƯƠNG 3: KẾT QUẢ VÀ HƯỚNG PHÁT TRIỂN

## 3.1. Tích hợp hệ thống

Sau khi hoàn thành việc thiết kế và lập trình cho từng thành phần riêng lẻ, nhóm đã tiến hành lắp đặt và tích hợp toàn bộ hệ thống để kiểm tra sự phối hợp hoạt động giữa các khối chức năng. Quá trình tích hợp được thực hiện một cách cẩn thận để đảm bảo hệ thống vận hành đúng như thiết kế.

### 3.1.1 Hệ thống thực tế.

Several electronic components on a table

AI-generated content may be incorrect.

Hình 3.1 Hệ thống thực tế.

Hình 3.1 mô tả hệ thống vật lý đã được nhóm lắp ráp và tích hợp hoàn chỉnh.. Các nodes và Gateway được làm trên bảng mạch in thủ công. Hệ thống bao gồm một Gateway trung tâm và ba Node Cảm biến độc lập.

### 3.1.2 Giao diện Web DashBoard

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.2 Giao diện Web DashBoard

Như được minh họa trong Hình 3.2, giao diện Web Dashboard được chia thành hai khu vực chức năng chính: Giám sát Môi trường và Điều khiển Thiết bị.

Khu vực Giám sát Môi trường: Khu vực này trình bày dữ liệu từ các Node Cảm biến dưới dạng các thẻ (card) trực quan, được tạo động bằng JavaScript. Mỗi thẻ tương ứng với một node và hiển thị các thông số đo được theo thời gian thực, bao gồm nhiệt độ, độ ẩm, cường độ ánh sáng và trạng thái có người (chuyển động). Trạng thái kết nối của từng node được thể hiện rõ ràng qua đường viền màu xanh lá cây ở cạnh trái thẻ (được định nghĩa bởi lớp .card.online trong CSS), cho biết node đang hoạt động và gửi dữ liệu đều đặn. Một điểm đáng chú ý là khả năng báo lỗi của hệ thống: ở Node 3, giá trị ánh sáng hiển thị là "-999lx". Đây là giá trị sentinel mà firmware của node gửi về khi không đọc được dữ liệu từ cảm biến BH1750. Điều này cho thấy dashboard không chỉ hiển thị dữ liệu thành công mà còn phản ánh chính xác các sự cố phần cứng tại các node, giúp người quản trị dễ dàng chẩn đoán.

Khu vực Điều khiển Thiết bị: Khu vực này cung cấp cho người dùng toàn quyền kiểm soát các thiết bị trong phòng học, được chia thành hai chế độ.

Chế độ Tự động (AI): Đây là trung tâm điều khiển thông minh của hệ thống. Một công tắc cho phép người dùng kích hoạt hoặc vô hiệu hóa chế độ AI. Khi được kích hoạt, trạng thái được hiển thị là "ĐANG CHẠY". Quan trọng hơn, giao diện còn hiển thị dự đoán hiện tại của mô hình AI đang chạy trên Gateway (ví dụ: "AI Dự Đoán: Can Bat Den Va Quat"), mang lại sự minh bạch về quyết định của hệ thống cho người dùng.

Điều khiển thủ công: Bên dưới là các khu vực điều khiển riêng cho "Hệ Thống Đèn" và "Hệ Thống Quạt". Mỗi thiết bị (Đèn 1-4, Quạt 1-4) đều có một công tắc riêng. Trạng thái "BẬT" trên giao diện phản ánh chính xác rằng các relay tương ứng trên Gateway đã được kích hoạt, đồng bộ hóa hai chiều thông qua Firebase.

## 3.2. Kết quả thử nghiệm hệ thống

### 3.2.1. Giao tiếp ESP-NOW

Trong quá trình tích hợp, nhóm đã tiến hành kiểm tra khả năng giao tiếp ESP-NOW giữa các Node Cảm biến và Gateway. Kết quả thu được từ Serial Monitor của Gateway cho thấy dữ liệu từ các node (ID 1, 2, 3) được nhận thành công và đầy đủ, bao gồm nhiệt độ, độ ẩm, ánh sáng và trạng thái chuyển động. Tần suất nhận dữ liệu tại Gateway khớp với chu kỳ ngủ/thức của các node, cho thấy giao tiếp qua ESP-NOW hoạt động ổn định và hiệu quả.

A white screen with black text

AI-generated content may be incorrect.

Hình 3.3 Log kết quả giao tiếp ESP-NOW

Hình 3.3 minh họa một chu trình hoạt động hoàn chỉnh của Node Cảm biến. Log cho thấy node đã thực hiện thành công các bước : Đọc dữ liệu cảm biến, gửi dữ liệu, xác nhận gửi, chuyển sang chế độ ngủ.

### 3.2.2. Khối xử lý GateWay

Song song với việc kiểm tra phía node gửi, nhóm cũng đã phân tích log hoạt động tại Gateway để xác thực toàn bộ luồng xử lý dữ liệu đầu cuối.

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.4 Log kết quả của khối xử lý tại GateWay

Hình 3.4 cho thấy các bước xử lý dau:

1. Nhận và phân tích dữ liệu: Gateway nhận thành công gói tin từ Node và phân tích (parse) chính xác các giá trị nhiệt độ, độ ẩm, ánh sáng và chuyển động.
2. Kiểm tra kết nối và gửi lên Cloud: Hệ thống xác nhận kết nối Firebase đã sẵn sàng (Firebase OK) và bắt đầu quá trình gửi dữ liệu lên Realtime Database.
3. Thực thi mô hình Edge AI: Khi chế độ tự động được kích hoạt, Gateway đưa dữ liệu cảm biến vào mô hình AI đã được tích hợp.
4. Kết quả suy luận: Mô hình trả về xác suất (confidence score) cho từng lớp hành động đã được huấn luyện (ví dụ: can\_bat\_den, can\_bat\_den\_va\_quat,...).

### 3.1.3. Xác thực Cấu trúc Dữ liệu trên Firebase Realtime Database

|  |  |
| --- | --- |
| A screenshot of a computer  AI-generated content may be incorrect. | A screenshot of a computer  AI-generated content may be incorrect. |

Hình 3.4 Kết quả cấu trúc dữ liệu cảm biến ở trên Firebase

Hình 3.4 cho thấy dữ liệu từ các Node Cảm biến được Gateway xử lý và đẩy lên thành công dưới cây sensorData. Mỗi node con (ví dụ: node1) chứa đầy đủ các trường dữ liệu tương ứng với các cảm biến vật lý, bao gồm humidity, light\_intensity, motion\_detected, và temperature. Cấu trúc này đảm bảo dữ liệu được phân tách rõ ràng theo từng node, giúp việc truy xuất và hiển thị trên Web Dashboard trở nên dễ dàng.

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.5 Kết quả cấu trúc dữ liệu điều khiển ở trên Firebase

Hình 3.5 thể hiện cấu trúc dữ liệu cho việc điều khiển và phản hồi trạng thái. Cây dữ liệu LED chứa trạng thái bật/tắt (true/false) của từng đèn và quạt. Các giá trị này được cập nhật khi người dùng tương tác trên Web Dashboard và được Gateway lắng nghe để điều khiển relay vật lý. Bên cạnh đó, cây ai\_status được sử dụng để Gateway cập nhật trạng thái của chế độ AI.

Hiệu năng của mô hình AI là một yếu tố quan trọng quyết định sự thành công của tính năng điều khiển tự động. Sau khi huấn luyện trên nền tảng Edge Impulse, mô hình đã được đánh giá hiệu năng dựa trên bộ dữ liệu kiểm thử (dữ liệu mà mô hình chưa từng thấy trong quá trình huấn luyện). Các chỉ số hiệu năng chính được ghi nhận như sau:

**Bảng 3.1:** Kết quả thử nghiệm hiệu năng mô hình AI

|  |  |
| --- | --- |
| **Chỉ số** | **Giá trị** |
| **Độ chính xác (Accuracy)** | **94.5%** |
| **Thời gian suy luận (Inference Time)** | **~ 25 ms** |
| **Mức sử dụng RAM** | **~ 5.2 KB** |
| **Mức sử dụng Flash (bộ nhớ chương trình)** | **~ 21.4 KB** |

Kết quả từ bảng trên cho thấy mô hình đạt độ chính xác rất cao (94.5%), nghĩa là nó có khả năng dự đoán đúng hành động cần thực hiện trong hầu hết các tình huống. Quan trọng hơn, thời gian suy luận (thời gian để mô hình đưa ra một dự đoán) rất ngắn (~25 ms) và yêu cầu về tài nguyên phần cứng (RAM và Flash) là cực kỳ thấp. Điều này khẳng định rằng mô hình được tối ưu hóa rất tốt và hoàn toàn phù hợp để chạy mượt mà trên một vi điều khiển có tài nguyên hạn chế như ESP32.

## 3.3. Đánh giá hệ thống

### 3.3.1. Ưu điểm

Qua quá trình thiết kế và thử nghiệm, nhóm chúng em nhận thấy hệ thống đã đạt được nhiều ưu điểm nổi bật. Thứ nhất, hệ thống có chi phí thấp và dễ dàng tiếp cận do được xây dựng từ các linh kiện điện tử phổ biến như ESP32, DHT22, PIR, BH1750. Thứ hai, khả năng phản hồi thời gian thực là một điểm cộng lớn, có được nhờ sự kết hợp giữa giao thức ESP-NOW cho mạng cục bộ và Firebase Realtime Database cho việc đồng bộ hóa với đám mây, đảm bảo độ trễ thấp trong cả việc giám sát và điều khiển. Thứ ba, hệ thống có khả năng tiết kiệm năng lượng hiệu quả, đặc biệt ở các Node Cảm biến nhờ việc áp dụng chế độ ngủ sâu. Thứ tư, việc tích hợp Edge AI giúp hệ thống trở nên linh hoạt và đáng tin cậy, có thể hoạt động tự động ngay cả khi mất kết nối Internet. Cuối cùng, nhờ kiến trúc module hóa và việc tạo giao diện động, hệ thống có khả năng dễ dàng mở rộng trong tương lai, có thể thêm bớt các node cảm biến hoặc thiết bị điều khiển mà không cần thay đổi lớn về cấu trúc.

### 3.3.2. Nhược điểm và hạn chế

Bên cạnh những ưu điểm đã đạt được, nhóm cũng thẳng thắn nhìn nhận một số nhược điểm và hạn chế của hệ thống. Vấn đề lớn nhất nằm ở độ phức tạp của Gateway. Việc Gateway phải xử lý đồng thời cả hai giao thức không dây là ESP-NOW và Wi-Fi trên cùng một bộ thu phát vô tuyến là một thách thức kỹ thuật, có thể tiềm ẩn nguy cơ mất ổn định và ảnh hưởng đến hiệu suất. Tiếp theo là vấn đề bảo mật. Hệ thống hiện tại phụ thuộc nhiều vào các quy tắc bảo mật của Firebase. Nếu cấu hình không đúng, dữ liệu có thể bị truy cập trái phép. Giao thức ESP-NOW cũng cần được bật mã hóa để đảm bảo an toàn. Cuối cùng, độ chính xác của cảm biến cũng là một hạn chế. Các cảm biến được sử dụng là loại phổ thông, có độ chính xác ở mức tương đối, phù hợp cho ứng dụng phòng học nhưng có thể không đủ cho các môi trường yêu cầu độ chính xác cao hơn.

### 3.3.3. Hướng phát triển

Từ những nhược điểm đã phân tích, nhóm chúng em đề xuất một số hướng phát triển để hoàn thiện và nâng cao hệ thống trong tương lai. Đầu tiên, để nâng cao độ tin cậy của Gateway, cần nghiên cứu và thử nghiệm việc triển khai Gateway theo kiến trúc hai chip ESP32 riêng biệt, một chip chuyên xử lý ESP-NOW và một chip chuyên xử lý Wi-Fi và giao tiếp đám mây. Hướng thứ hai là tăng cường bảo mật toàn diện bằng cách triển khai các quy tắc truy cập dựa trên vai trò (RBAC) trên Firebase và bật tính năng mã hóa cho giao thức ESP-NOW. Hướng thứ ba là mở rộng khả năng cảm biến của hệ thống bằng cách tích hợp thêm các loại cảm biến mới như cảm biến chất lượng không khí (CO2, bụi mịn), cảm biến âm thanh để có một bức tranh toàn diện hơn về môi trường phòng học. Cuối cùng, để nâng cao trải nghiệm người dùng, nhóm có thể phát triển một ứng dụng di động (Android/iOS) chuyên dụng, cho phép điều khiển và nhận các thông báo đẩy (push notification) một cách tiện lợi hơn.

# KẾT LUẬN

Qua thời gian nghiên cứu và thực hiện, nhóm chúng em đã hoàn thành đồ án "Hệ Thống Giám Sát và Điều Khiển Môi Trường Phòng Học Thông Minh Tích Hợp AI", đạt được các mục tiêu đề ra ban đầu. Đồ án đã thành công trong việc xây dựng một hệ thống IoT hoàn chỉnh, từ phần cứng đến phần mềm, có khả năng ứng dụng thực tiễn cao.

Bằng việc kết hợp các công nghệ tiên tiến và phù hợp như vi điều khiển ESP32, giao thức ESP-NOW, nền tảng đám mây Firebase và Trí tuệ nhân tạo tại biên (Edge AI), hệ thống đã giải quyết được các bài toán cốt lõi về giám sát thời gian thực, điều khiển từ xa và tự động hóa thông minh.

Các kết quả thử nghiệm đã chứng minh hệ thống hoạt động ổn định, có độ trễ thấp và khả năng phản hồi nhanh. Việc tích hợp thành công mô hình Edge AI trên Gateway là một thành tựu quan trọng, mang lại khả năng tự chủ cho hệ thống, giảm độ trễ và tăng cường độ tin cậy ngay cả khi không có kết nối Internet.

Mặc dù vẫn còn một số hạn chế cần khắc phục và nhiều hướng phát triển có thể tiếp tục theo đuổi, nhóm tin rằng đồ án này là một nền tảng vững chắc. Nó không chỉ là minh chứng cho kiến thức và kỹ năng mà nhóm đã tích lũy được, mà còn mở ra tiềm năng cho việc phát triển các giải pháp thông minh, góp phần hiện đại hóa môi trường giáo dục, hướng tới một không gian học tập tiện nghi, an toàn và bền vững hơn.

# TÀI LIỆU THAM KHẢO

[1] Espressif Systems. (2023). *ESP32-S3 Series Datasheet*.

[2] Espressif Systems. (2022). *ESP-NOW API Reference*.

[3] Google. (2024). *Firebase Realtime Database Documentation*.

[4] Google. (2024). *Firebase Hosting Documentation*.

[5] Edge Impulse. (2024). *Edge Impulse Documentation - Building ML models for embedded devices*.

[6] Aosong. (2019). *DHT22 (AM2302) Digital Temperature and Humidity Sensor Datasheet*.

[7] ROHM Semiconductor. (2018). *BH1750FVI Digital Ambient Light Sensor IC Datasheet*.

[8] Random Nerd Tutorials. (2023). *ESP32 with DHT11/DHT22 Temperature and Humidity Sensor using Arduino IDE*.