TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN &

TRUYỀN THÔNG VIỆT HÀN

**Khoa Kĩ Thuật Máy Tính & Điện Tử**



ĐỒ ÁN CHUYÊN NGÀNH 3

**THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG GIÁM SÁT CHẤT LƯỢNG MÔI TRƯỜNG**

Sinh viên thực hiện: **Nguyễn Lê Ngọc Ánh**

Mã sinh viên: **21CE004**

Lớp: **21CE1**

Giảng viên hướng dẫn: **TS. NGUYỄN VŨ ANH QUANG**

Đà Nẵng, tháng 5 năm 2025

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN &

TRUYỀN THÔNG VIỆT HÀN

**Khoa Kĩ Thuật Máy Tính & Điện Tử**



ĐỒ ÁN CHUYÊN NGÀNH 3

**THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG GIÁM SÁT CHẤT LƯỢNG MÔI TRƯỜNG**

Sinh viên thực hiện: **Nguyễn Lê Ngọc Ánh**

Mã sinh viên: **21CE004**

Lớp: **21CE1**

Giảng viên hướng dẫn: **TS. NGUYỄN VŨ ANH QUANG**

Đà Nẵng, tháng 5 năm 2025

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

# 

# 

# 

# 

Đà Nẵng, tháng 5 năm 2025

Giảng viên hướng dẫn

TS. Nguyễn Vũ Anh Quang

LỜI CẢM ƠN

Đầu tiên, em xin gửi lời cảm ơn đến Trường Đại học Công Nghệ Thông Tin Và Truyền Thông Việt – Hàn đã luôn lắng nghe và luôn luôn tạo điều kiện học tập tốt nhất cho em và cũng như toàn thể sinh viên trong trường.

Tiếp đến em xin bày tỏ lòng biết ơn sâu sắc nhất tới giáo viên hướng dẫn TS.Nguyễn Vũ Anh Quang đã tận tình giúp đỡ, tìm ra nhiều lỗi sai của em trong quá trình tìm hiểu và nghiên cứu. Thầy còn hướng dẫn em rất nhiều để hoàn thành xong đồ án chuyên ngành .

Vì thời gian có hạn, trình độ hiểu biết của bản thân còn nhiều hạn chế. Cho nên trong đồ án không tránh khỏi những thiếu sót, em mong nhận được sự đóng góp ý kiến của các thầy cô giáo để đồ án của em được hoàn thiện hơn.

Em xin chân thành cảm ơn !

Đà Nẵng, tháng 5 năm 2025

*Sinh viên thực hiện*

*Nguyễn Lê Ngọc Ánh*

MỤC LỤC

[MỤC LỤC v](#_Toc197323993)

[MỞ ĐẦU xiv](#_Toc197323994)

[CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI 1](#_Toc197323995)

[1.1 Thực trạng hiện nay 1](#_Toc197323996)

[1.2 Mục tiêu chọn đề tài 1](#_Toc197323997)

[1.3 Lý do chọn đề tài 1](#_Toc197323998)

[1.4 Chức năng 1](#_Toc197323999)

[1.5 Các ứng dụng của đề tài 2](#_Toc197324000)

[CHƯƠNG 2. CƠ SỞ LÝ THUYẾT 3](#_Toc197324001)

[2.1 WIFI 3](#_Toc197324002)

[2.1.1 Giới thiệu 3](#_Toc197324003)

[2.1.2 Công nghệ truyền nhận dữ liệu 3](#_Toc197324004)

[2.1.3 Thành phần của mạng WIFI 6](#_Toc197324005)

[2.1.4 Cấu trúc liên kết 7](#_Toc197324006)

[2.1.5 Hostpot 8](#_Toc197324007)

[2.1.6 Cách thức hoạt động 8](#_Toc197324008)

[2.1.7 Giao tiếp trong WIFI 9](#_Toc197324009)

[2.1.8 Ưu nhược điểm 10](#_Toc197324010)

[2.1.9 Bảo mật 10](#_Toc197324011)

[2.2 Tổng quan về Internet of thing 12](#_Toc197324012)

[2.2.1 Giới thiệu 12](#_Toc197324013)

[2.2.2 Lịch sử hình thành 13](#_Toc197324014)

[2.2.3 Ứng dụng 14](#_Toc197324015)

[2.3 Giao thức MQTT 14](#_Toc197324016)

[2.3.1 Giới thiệu 14](#_Toc197324017)

[2.3.2 Lịch sử 15](#_Toc197324018)

[2.3.3 Các thành phần của MQTT 16](#_Toc197324019)

[2.3.4 MQTT QoS 17](#_Toc197324020)

[2.3.5 Broker trong IoT 17](#_Toc197324021)

[2.3.6 Ưu điểm 18](#_Toc197324022)

[2.3.7 Bảo mật 18](#_Toc197324023)

[2.4 HTTP 19](#_Toc197324024)

[2.4.1 Giới thiệu 19](#_Toc197324025)

[2.4.2 Các thành phần chính của HTTP 19](#_Toc197324026)

[2.5 Webserver 22](#_Toc197324027)

[2.5.1 Giới thiệu 22](#_Toc197324028)

[2.5.2 Giao tiếp với Webserver 22](#_Toc197324029)

[2.6 FreeRTOS 23](#_Toc197324030)

[2.6.1 Tổng quan về FreeRTOS 23](#_Toc197324031)

[2.6.2 Kiến trúc FreeRTOS 24](#_Toc197324032)

[2.6.3 Khái Niệm Cốt Lõi và Cơ Chế Hoạt Động của FreeRTOS 25](#_Toc197324033)

[2.6.4 Lợi ích của FreeRTOS trong phát triển ứng dụng ESP32 35](#_Toc197324034)

[2.7 Giới thiệu Mô hình Dự báo Chuỗi Thời gian Prophet 36](#_Toc197324035)

[2.7.1 Giới thiệu 36](#_Toc197324036)

[2.7.2 Cấu trúc Mô hình 37](#_Toc197324037)

[2.7.3 Nguyên lý hoạt động và Ưu điểm 38](#_Toc197324038)

[2.8 Node-RED 39](#_Toc197324039)

[2.8.1 Giới thiệu 39](#_Toc197324040)

[2.8.2 Node-RED trong IoT 40](#_Toc197324041)

[2.8.3 Tính năng 41](#_Toc197324042)

[2.9 MySQL 42](#_Toc197324043)

[2.9.1 Giới thiệu 42](#_Toc197324044)

[2.9.2 Các tính năng của MySQL 43](#_Toc197324045)

[2.10 Raspberry Pi 4 43](#_Toc197324046)

[2.10.1 Giới thiệu 44](#_Toc197324047)

[2.10.2 Các loại Raspberry Pi 4 45](#_Toc197324048)

[2.10.3 Tính năng 45](#_Toc197324049)

[2.10.4 Các chân input-output của Raspberry Pi 4 46](#_Toc197324050)

[2.11 ESP32 49](#_Toc197324051)

[2.11.1 Giới thiệu 49](#_Toc197324052)

[2.11.2 Các loại esp32 50](#_Toc197324053)

[2.11.3 Tính năng 51](#_Toc197324054)

[2.11.4 Các chân input-output của ESP32 WROOM 51](#_Toc197324055)

[2.12 DHT11 55](#_Toc197324056)

[2.12.1 Giới thiệu 55](#_Toc197324057)

[2.12.2 Cấu tạo 55](#_Toc197324058)

[2.12.3 Nguyên lý hoạt động 56](#_Toc197324059)

[2.13 GP2Y1010AU0F 57](#_Toc197324060)

[2.13.1 Giới thiệu 57](#_Toc197324061)

[2.13.2 Cấu tạo 57](#_Toc197324062)

[2.13.3 Nguyên lý hoạt động 59](#_Toc197324063)

[2.14 OLED 59](#_Toc197324064)

[2.14.1 Giới thiệu 60](#_Toc197324065)

[2.14.2 Cấu tạo 60](#_Toc197324066)

[2.14.3 Nguyên lý hoạt động 61](#_Toc197324067)

[2.15 Relay 2 chanel 61](#_Toc197324068)

[2.15.1 giới thiệu 61](#_Toc197324069)

[2.15.2 cấu tạo 62](#_Toc197324070)

[2.15.3 Nguyên Lý hoạt động 62](#_Toc197324071)

[CHƯƠNG 3. THIẾT KẾ HỆ THỐNG 65](#_Toc197324072)

[3.1 Tổng quan về hệ thống 65](#_Toc197324073)

[3.2 Sơ đồ khối chức năng 66](#_Toc197324074)

[3.3 Lưu đồ thuật toán 69](#_Toc197324075)

[3.4 Flow node red broker 80](#_Toc197324076)

[3.5 Phần mềm và ngôn ngữ lập trình 82](#_Toc197324077)

[CHƯƠNG 4 THIẾT KẾ MẠCH ĐIỆN 94](#_Toc197324078)

[4.1 sơ đồ nguyên lý 94](#_Toc197324079)

[4.2 Mạch PCB 95](#_Toc197324080)

[4.3 Sản phẩm hoàn thiện 96](#_Toc197324081)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 98](#_Toc197324082)

[TÀI LIỆU THAM KHẢO 101](#_Toc197324083)

[PHỤ LỤC 102](#_Toc197324084)

**DANH MỤC HÌNH ẢNH**

[Hình 2. 1 Biểu tượng WIFI 3](#_Toc197614301)

[Hình 2. 2 Cấu trúc liên kết ngang hàng 7](#_Toc197614302)

[Hình 2. 3 Cấu trúc liên kết dựa trên AP 8](#_Toc197614303)

[Hình 2. 4 Cách thức hoạt động của mạng WIFI 9](#_Toc197614304)

[Hình 2. 5 Cách thức giao tiếp trong mạng WIFI 10](#_Toc197614305)

[Hình 2. 6 Internet of Things 13](#_Toc197614306)

[Hình 2. 7 Lịch sử hình thành IoT 14](#_Toc197614307)

[Hình 2. 8 Giao thức MQTT 15](#_Toc197614308)

[Hình 2. 9 Hoạt động của client 16](#_Toc197614309)

[Hình 2. 10 Hoạt động của Topic 17](#_Toc197614310)

[Hình 2. 11 Thành phần của một MQTT trong hệ thống IoT 18](#_Toc197614311)

[Hình 2. 12 Giao thức HTTP 19](#_Toc197614312)

[Hình 2. 13 Cấu trúc HTTP Request 20](#_Toc197614313)

[Hình 2. 14 Phương thức Request 21](#_Toc197614314)

[Hình 2. 15 Giao tiếp với Webserver 22](#_Toc197614315)

[Hình 2. 16 FreeRTOS 23](#_Toc197614316)

[Hình 2. 17 Kiến trúc FreeRTOS 24](#_Toc197614317)

[Hình 2. 18 Task FreeRTOS 25](#_Toc197614318)

[Hình 2. 19 Scheduler FreeRTOS 27](#_Toc197614319)

[Hình 2. 20 Queue FreeRTOS 29](#_Toc197614320)

[Hình 2. 21 Semaphores FreeRTOS 30](#_Toc197614321)

[Hình 2. 22 Mutexes FreeRTOS 31](#_Toc197614322)

[Hình 2. 23 Prophet time-series forecasting 37](#_Toc197614323)

[Hình 2. 24 Node trong Node – RED cơ bản 40](#_Toc197614324)

[Hình 2. 25 Node - RED và IoT 41](#_Toc197614325)

[Hình 2. 26 Giao diện Node - RED 41](#_Toc197614326)

[Hình 2. 27 Cơ sở dữ liệu MySQL 42](#_Toc197614327)

[Hình 2. 28 Raspberry Pi 4 44](#_Toc197614328)

[Hình 2. 29 GPIO Raspberry Pi 4 46](#_Toc197614329)

[Hình 2. 30 ESP32 50](#_Toc197614330)

[Hình 2. 31 Chip ESP32-VROOM-32D 52](#_Toc197614331)

[Hình 2. 32 Module ESP32-WROOM-32D 52](#_Toc197614332)

[Hình 2. 33 Module DHT11 55](#_Toc197614333)

[Hình 2. 34 Cấu tạo DHT11 56](#_Toc197614334)

[Hình 2. 35 Nguyên lý hoạt động DHT11 56](#_Toc197614335)

[Hình 2. 36 module cảm biến bụi mịn GP2Y1010AU0F 57](#_Toc197614336)

[Hình 2. 37 Cấu tạo cảm biến bụi mịn GP2Y1010AU0F 59](#_Toc197614337)

[Hình 2. 38 Màn hình OLED 60](#_Toc197614338)

[Hình 2. 39 Relay 2 Chanel 62](#_Toc197614339)

[Hình 2. 40 Nguyên lý hoạt động của relay 63](#_Toc197614340)

[Hình 2. 41 Nguyên lý hoạt động của relay 63](#_Toc197614341)

[Hình 3. 1 Sơ đồ tổng quát 67](#_Toc197648033)

[Hình 3. 2 Sơ đồ khối node slave 68](#_Toc197648034)

[Hình 3. 3 Sơ đồ khối node master 69](#_Toc197648035)

[Hình 3. 4 Lưu đồ chương trình chính Node slave 70](#_Toc197648036)

[Hình 3. 5 Lưu đồ chương trình StartAPMode 71](#_Toc197648037)

[Hình 3. 6 Lưu đồ chương trình TaskControl 72](#_Toc197648038)

[Hình 3. 7 Lưu đồ chương trình TaskDisplay 73](#_Toc197648039)

[Hình 3. 8 Lưu đồ chương trình TaskMQTT 74](#_Toc197648040)

[Hình 3. 9 Lưu đồ chương trình TaskMQTTSubcribe 75](#_Toc197648041)

[Hình 3. 10 Lưu đồ chương trình TaskPrintData 76](#_Toc197648042)

[Hình 3. 11 Lưu đồ chương trình TaskReadSensor 77](#_Toc197648043)

[Hình 3. 12 Lưu đồ chương trình TaskResetTimeout 78](#_Toc197648044)

[Hình 3. 13 Lưu đồ chương trình dự báo 79](#_Toc197648045)

[Hình 3. 14 Flow View Dashboard Node - RED 80](#_Toc197648046)

[Hình 3. 15 Flow Control Dashboard Node - RED 80](#_Toc197648047)

[Hình 3. 16 Flow Alarm email 80](#_Toc197648048)

[Hình 3. 17 Flow Save database MySQL 81](#_Toc197648049)

[Hình 3. 18 Arduino IDE 82](#_Toc197648050)

[Hình 3. 19 Chức Năng của Arduino IDE 83](#_Toc197648051)

[Hình 3. 20 File menu Arduino IDE 84](#_Toc197648052)

[Hình 3. 21 Edit menu Arduino IDE 85](#_Toc197648053)

[Hình 3. 22 Sketch menu Arduino IDE 86](#_Toc197648054)

[Hình 3. 23 Tools menu Arduino IDE 87](#_Toc197648055)

[Hình 3. 24 Output Panel Arduino IDE 88](#_Toc197648056)

[Hình 3. 25 Visual Studio Code 89](#_Toc197648057)

[Hình 3. 26 C++ 90](#_Toc197648058)

[Hình 3. 27 Python 92](#_Toc197648059)

[Hình 4. 1 Sơ đồ nguyên lý Node slave 94](#_Toc197614409)

[Hình 4. 2 Mạch PCB Node slave 95](#_Toc197614410)

[Hình 4. 3 Mạch hoàn thiện 96](#_Toc197614411)

[Hình 4. 4 Webserver forecast and control Node master 97](#_Toc197614412)

[Hình 4. 5 Webserver chart Node master 97](#_Toc197614413)

**DANH MỤC TỪ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| **STT** | Từ viết tắt | Từ đầy đủ |
| 1 | IoT | Internet of Things |
| 2 | WiFi | Wireless Fidelity |
| 3 | IEEE | Institute of Electrical and Electronic Engineers |
| 4 | LAN | Local Area Network |
| 5 | MIMO | Multiple Input Multiple Output |
| 6 | QAM | Quadrature Amplitude Modulation |
| **7** | MU-MIMO | Multi-User MIMO |
| 8 | OFDMA | Orthogonal Frequency-Division Multiple Access |
| 9 | TWT | Target Wake Time |
| 10 | AP | Access Point |
| 11 | WEP | Wireless Equivalent Privacy |
| 12 | WPA | Wi-Fi Protected Access |
| 13 | TKIP | Temporal Key Integrity Protocol |
| 14 | AES | Advanced Encryption Standard |
| 15 | SAE | Simultaneous Authentication of Equals |
| 16 | OWE | Opportunistic Wireless Encryption |
| 17 | RFID | Radio-Frequency Identification |
| 18 | AI | Artificial Intelligence |
| 19 | MQTT | Message Queuing Telemetry Transport |
| 20 | M2M | Machine to Machine |
| 21 | WSN | Wireless Sensor Networks |
| 22 | QoS | Quality of Service |
| 23 | HTTP | HyperText Transfer Protocol |
| 24 | WWW | World Wide Web |
| 25 | TCP | Transmission Control Protocol |
| 26 | BLE | Bluetooth Low Energy |
| 27 | GPIO | General Purpose Input Output Pins |
| 28 | UART | Universal Asynchronous Receiver-Transmitter |
| 29 | I2C | Inter-Integrated Circuit |
| 30 | SPI | Serial Peripheral Interface |
| 31 | I2S | Inter-IC Sound |
| 32 | ADC | Analog-to-Digital Converter |
| 33 | DAC | Digital-to-Analog Converters |
| 34 | PWM | Pulse Width Modulation |
| 35 | OTA | Over-The-Air |
| 36 | LED | Light Emitting Diodes |
| 37 | OLED | Organic Light Emitting Diodes |
| 38 | SDA | Serial Data |
| 39 | SCL | Serial Clock |
| 40 | SWI | Software Interrupt |

MỞ ĐẦU

**Giới thiệu đề tài**

Hiện nay, chất lượng không khí ngày càng được chú trọng, đặc biệt trong các khu vực nhạy cảm như phòng bệnh viện, nơi yêu cầu một môi trường sạch sẽ và an toàn để đảm bảo sức khỏe cho bệnh nhân. Tình trạng ô nhiễm không khí không chỉ tồn tại ở không gian công cộng mà còn len lỏi vào trong các không gian kín như phòng bệnh, gây ảnh hưởng tiêu cực đến quá trình hồi phục của bệnh nhân và làm tăng nguy cơ lây nhiễm chéo.

Chúng em đã chọn đề tài “Thiết kế và xây dựng hệ thống giám sát chất lượng môi trường” để nghiên cứu và phát triển một mô hình giám sát thông minh, sử dụng các công nghệ IoT tiên tiến, với mục tiêu tối ưu hóa chất lượng không khí trong phòng bệnh. Đề tài này nhằm góp phần giải quyết vấn đề ô nhiễm không khí trong không gian kín, cải thiện điều kiện y tế và nâng cao chất lượng cuộc sống cho bệnh nhân.

**Lý do lựa chọn đề tài**

Tầm quan trọng của chất lượng không khí trong phòng bệnh: Không khí trong lành trong phòng bệnh có vai trò quan trọng trong việc đảm bảo môi trường phục hồi tốt nhất cho bệnh nhân. Hệ thống giám sát giúp phát hiện kịp thời các yếu tố gây ô nhiễm như bụi mịn (PM2.5, PM10) và đảm bảo độ ẩm, nhiệt độ luôn ở mức tối ưu.

Ứng dụng công nghệ IoT: Với sự phát triển của Internet of Things, việc tích hợp các cảm biến và hệ thống tự động vào giám sát không khí mang lại hiệu quả cao hơn. Hệ thống có thể theo dõi, phân tích và đưa ra các giải pháp kịp thời, giúp tiết kiệm thời gian và công sức so với các phương pháp giám sát truyền thống.

Bảo vệ sức khỏe và môi trường: Hệ thống không chỉ hỗ trợ bảo vệ sức khỏe cho bệnh nhân mà còn giúp giảm thiểu tác động tiêu cực lên môi trường bằng cách quản lý hiệu quả các thiết bị điều hòa không khí, máy lọc không khí.

Tăng cường kiến thức và kỹ năng: Đề tài này mang tính ứng dụng cao và yêu cầu sự kết hợp kiến thức về cảm biến, điện tử, lập trình, và phân tích dữ liệu. Đây là cơ hội để chúng em phát triển kỹ năng trong lĩnh vực IoT, từ đó mở rộng khả năng ứng dụng công nghệ vào các lĩnh vực khác.

Thực tiễn và tiềm năng phát triển: Các bệnh viện và cơ sở y tế ngày càng chú trọng cải thiện môi trường không khí, tạo ra tiềm năng phát triển cho các hệ thống giám sát thông minh. Đề tài này không chỉ giải quyết vấn đề hiện tại mà còn có thể mở rộng để áp dụng trong các khu vực khác như trường học, văn phòng, hoặc nhà ở.

**Mục tiêu nghiên cứu**

Phát triển hệ thống giám sát chất lượng không khí thông minh: Nghiên cứu và phát triển hệ thống IoT giám sát chất lượng không khí trong phòng bệnh, tích hợp các công nghệ cảm biến tiên tiến và kết nối mạng để cung cấp dữ liệu thời gian thực.

Tăng cường kiểm soát và cải thiện môi trường không khí: Đề xuất các giải pháp tự động hóa như điều chỉnh hoạt động của máy lọc không khí, điều hòa nhiệt độ, nhằm đảm bảo các chỉ số môi trường như nhiệt độ, độ ẩm, và mức độ ô nhiễm luôn ở ngưỡng an toàn.

Tăng cường sức khỏe và an toàn cho bệnh nhân: Đảm bảo môi trường trong lành, hỗ trợ quá trình hồi phục của bệnh nhân và giảm nguy cơ lây nhiễm chéo trong phòng bệnh.

Đánh giá tác động môi trường: Xem xét hiệu quả sử dụng năng lượng và tài nguyên của hệ thống, từ đó đánh giá khả năng áp dụng lâu dài và bền vững.

Chia sẻ kết quả nghiên cứu: Cung cấp một giải pháp tham khảo cho các bệnh viện và các không gian công cộng khác, thúc đẩy việc áp dụng công nghệ IoT trong quản lý môi trường tại Việt Nam.

**Phương pháp nghiên cứu**

Nghiên cứu tài liệu: Thu thập và phân tích các tài liệu, bài báo, và nghiên cứu liên quan đến các hệ thống giám sát môi trường, cảm biến, IoT, và tiêu chuẩn chất lượng không khí.

Nghiên cứu thực địa: Khảo sát môi trường thực tế tại các phòng bệnh viện, thu thập dữ liệu về điều kiện không khí và yêu cầu của hệ thống giám sát.

Thử nghiệm thực tế: Thiết kế và triển khai hệ thống mẫu tại một phòng bệnh cụ thể, tiến hành đo lường và đánh giá hiệu quả của hệ thống trong các điều kiện thực tế.

Phân tích dữ liệu: Sử dụng các công cụ phân tích dữ liệu để đánh giá hiệu quả hoạt động của hệ thống, phát hiện các yếu tố ảnh hưởng đến chất lượng không khí và tối ưu hóa giải pháp.

Phương pháp chuyên gia: Tham khảo ý kiến từ các chuyên gia trong lĩnh vực IoT, y tế, và môi trường để hoàn thiện thiết kế và chức năng của hệ thống.

Phương pháp thực tiễn: So sánh hệ thống đề xuất với các mô hình giám sát chất lượng không khí đã được áp dụng trước đó, đánh giá ưu điểm và khả năng mở rộng.

**Nội dung và kế hoạch thực hiện**

**a) Nội dung thực hiện**

**B1: Tìm hiểu đề tài**

Khám phá các công nghệ IoT hiện tại trong lĩnh vực giám sát môi trường, đặc biệt tập trung vào các mô hình sử dụng ESP32 và giao thức MQTT.

Phân tích nhu cầu giám sát môi trường trong không gian kín, như phòng bệnh viện, để xác định các thông số cần giám sát (AQI, PM2.5, nhiệt độ, độ ẩm).

Tìm hiểu về mô hình Prophet để dự báo chuỗi thời gian, bao gồm cách xử lý dữ liệu môi trường (xu hướng hàng ngày, mùa vụ, và các yếu tố bất thường) và tích hợp với Python trên Raspberry Pi 4.

**B2: Các công cụ thực hiện**

Ngôn ngữ lập trình: C++, python và JavaScript (Node-RED).

Thiết bị phần cứng:

* Node slave ESP32: Kết nối các cảm biến như DHT11, GP2Y1010AU0F, Relay, màn hình hiển thị.
* Raspberry Pi 4:
  + Chạy MQTT Broker để quản lý giao tiếp dữ liệu.
  + Lưu trữ và chạy web server để hiển thị trạng thái môi trường và cho phép điều khiển thủ công.
  + Thực thi tập lệnh Python với Prophet để dự báo chất lượng không khí.

Phần mềm và cơ sở dữ liệu:

* MQTT Broker: Sử dụng Node-RED trên Raspberry Pi 4 làm trung tâm giao tiếp, xử lý dữ liệu, và quản lý luồng MQTT.
* Web server: Cài trên Raspberry Pi 4 để hiển thị dữ liệu thời gian thực và kết quả dự báo từ Prophet và điều khiển.
* Python: Sử dụng Prophet để dự báo, với dữ liệu đầu vào từ cơ sở dữ liệu.
* Cơ sở dữ liệu: Lưu trữ dữ liệu trên Raspberry Pi 4, sử dụng MySQL để lưu dữ liệu cảm biến và kết quả dự báo, hỗ trợ phân tích và trực quan hóa.

**B3: Các nội dung chính**

Nghiên cứu cơ sở:

* Tìm hiểu về cảm biến, giao thức MQTT, và tích hợp ESP32 với Node-RED.
* Khám phá các mô hình tương tự trong lĩnh vực giám sát và điều khiển không khí trong không gian kín.
* Nghiên cứu mô hình Prophet, bao gồm cách chuẩn bị dữ liệu (định dạng cột ds cho thời gian, y cho giá trị như AQI/PM2.5), xử lý xu hướng, mùa vụ, và các tham số như seasonality\_mode, changepoint\_prior\_scale.
* Tìm hiểu cách tích hợp kết quả dự báo từ Prophet vào web server hoặc Node-RED Dashboard.
* Khám phá các mô hình dự báo môi trường tương tự sử dụng chuỗi thời gian.

Thu thập dữ liệu:

* Tích hợp các cảm biến với ESP32 để thu thập dữ liệu môi trường trong phòng bệnh.
* Thiết lập hệ thống truyền dữ liệu qua MQTT để đảm bảo việc giám sát theo thời gian thực.
* Lưu trữ dữ liệu cảm biến (AQI, PM2.5, nhiệt độ, độ ẩm) vào cơ sở dữ liệu trên Raspberry Pi 4 với định dạng phù hợp cho Prophet (cột ds là thời gian, y là giá trị).
* Đảm bảo dữ liệu được thu thập đủ lâu (ví dụ: vài tuần) để Prophet có thể nhận diện xu hướng và mùa vụ.

Thiết kế và xây dựng hệ thống :

* Node slave:
* Gửi dữ liệu cảm biến qua MQTT.
* Nhận lệnh điều khiển mode từ MQTT (Auto/Manual).
* Node master:
* MQTT Broker: Node-RED quản lý luồng dữ liệu.
* Web server: Hiển thị dữ liệu thời gian thực (AQI, PM2.5, nhiệt độ, độ ẩm) và kết quả dự báo từ Prophet (ví dụ: biểu đồ dự báo AQI/PM2.5 trong 24 giờ tới).
* Dự báo với Prophet:

- Chạy tập lệnh Python trên Raspberry Pi 4 sử dụng Prophet để dự báo chất lượng không khí.

- Dữ liệu đầu vào lấy từ cơ sở dữ liệu MySQL.

- Lưu kết quả dự báo vào cơ sở dữ liệu và hiển thị trực tiếp trên web server.

- Ví dụ: Dự báo AQI hoặc PM2.5 trong 24 giờ tới, hỗ trợ quyết định bật/tắt máy lọc không khí.

Tích hợp và thử nghiệm:

* Thiết lập MQTT Broker trên Node-RED để quản lý dữ liệu từ hai ESP32.
* Kiểm tra luồng dữ liệu và độ ổn định của các kết nối MQTT.
* Thực hiện thử nghiệm thực tế tại phòng bệnh hoặc không gian mô phỏng.
* Kiểm tra tích hợp web server trên Raspberry Pi 4, đảm bảo hiển thị dữ liệu thời gian thực và kết quả dự báo từ Prophet (bao gồm giá trị dự đoán và khoảng tin cậy).
* Kiểm tra hiệu suất mô hình Prophet:

- Đánh giá độ chính xác dự báo (so sánh giá trị dự đoán với dữ liệu thực tế).

- Điều chỉnh tham số Prophet (ví dụ: seasonality\_prior\_scale, holidays) để cải thiện độ chính xác.

- Đảm bảo Raspberry Pi 4 xử lý tốt đồng thời MQTT Broker, web server, và tập lệnh Prophet mà không bị quá tải.

Phân tích dữ liệu và đánh giá:

* Sử dụng dữ liệu thu thập được để đánh giá hiệu quả của hệ thống giám sát so với tiêu chuẩn môi trường an toàn.
* Xác định và cải thiện các yếu tố ảnh hưởng đến độ chính xác và hiệu suất của hệ thống.
* Sử dụng kết quả dự báo từ Prophet để so sánh với dữ liệu thực tế, tính toán các chỉ số như MAE (Mean Absolute Error) hoặc RMSE (Root Mean Square Error).
* Phân tích xu hướng và mùa vụ từ Prophet (ví dụ: AQI tăng vào buổi tối do lưu thông không khí kém) để đưa ra khuyến nghị vận hành thiết bị.
* Lưu trữ và phân tích dữ liệu lịch sử từ cơ sở dữ liệu để cải thiện mô hình Prophet (ví dụ: thêm dữ liệu dài hơn hoặc bổ sung yếu tố ngoại lai như thời tiết).
* Đánh giá hiệu quả của hệ thống giám sát và dự báo so với tiêu chuẩn môi trường an toàn.
* Cải thiện các yếu tố ảnh hưởng đến độ chính xác (ví dụ: chất lượng dữ liệu cảm biến, tần suất thu thập, hoặc cấu hình Prophet).

Xây dựng hướng dẫn sử dụng:

* Soạn thảo tài liệu hướng dẫn cài đặt và vận hành hệ thống.
* Đưa ra các khuyến nghị tối ưu hóa môi trường phòng bệnh dựa trên dữ liệu thu thập.

Đánh giá tác động: Phân tích tác động của hệ thống đến chất lượng không khí, sức khỏe bệnh nhân, và tiêu thụ năng lượng.

Báo cáo và công bố: Tổng hợp kết quả nghiên cứu vào báo cáo và công bố kết quả thông qua hội thảo, bài báo, và phương tiện truyền thông.

**B4: Làm slide thuyết trình về dự án**

Công cụ thực hiện: PowerPoint.

**B5: Làm báo cáo dự án**

Công cụ thực hiện: MS Word

**b) Kế hoạch thực hiện**

|  |  |
| --- | --- |
| **Thời gian** | **Nội dung thực hiện** |
| Tuần thứ 1 từ 20/03/2025 đến 26/03/2025 | * Tìm hiểu về ESP32, datasheets, cách sử dụng. * Tìm hiểu về Raspberry pi, cách sử dụng. |
| Tuần thứ 2 từ 27/3/2025 đến 02/04/2025 | * Tìm hiểu về các thư viện của ESP32 và phần mềm code. * Tìm hiểu về các thư viện cần thiết cho các mô hình machine learning. |
| Tuần thứ 3 từ 03/04/2025 đến 09/04/2025 | * Tìm hiểu MQTT và cách thức hoạt động. * Tìm hiểu mô hình machine learning cho việc dự báo. |
| Tuần thứ 4 từ 10/04/2025 đến 17/04/2025 | * Thiết kế mạch. * Hoàn thiện mạch. |
| Tuần thứ 5 từ 18/04/2025 đến 24/04/2025 | * Code cho ESP32. * Code cho Raspberry pi. |
| Tuần thứ 6 từ 25/04/2025 đến 03/05/2025 | * Kiểm tra và đánh giá tính ổn định của sản phẩm. |
| Tuần thứ 7 từ 04/05/2025 đến 10/05/2025 | * Hoàn thiện sản phẩm cuối cùng. |

**Bố cục báo cáo**

Sau phần *Mở đầu*, báo cáo được trình bày trong ba chương, cụ thể như sau:

Chương 1. *Tổng quan về đề tài*.

Chương 2. *Cơ sở lý thuyết.*

Chương 3. *Phân tích và thiết kế mạch*

Cuối cùng là *Tổng kết*, *Tài liệu tham khảo* và *Phụ* *lục*

1. TỔNG QUAN VỀ ĐỀ TÀI

### 1.1 Thực trạng hiện nay

Chất lượng không khí trong các cơ sở y tế, đặc biệt là phòng bệnh, đóng vai trò quan trọng trong việc bảo vệ sức khỏe bệnh nhân và hỗ trợ quá trình hồi phục. Tuy nhiên, không gian kín như phòng bệnh thường gặp phải tình trạng ô nhiễm không khí, bao gồm bụi mịn (PM2.5, PM10), nhiệt độ và độ ẩm, dẫn đến nguy cơ lây nhiễm và ảnh hưởng xấu đến sức khỏe người bệnh. Hiện nay, các hệ thống giám sát chất lượng không khí trong bệnh viện chủ yếu dựa trên thiết bị đo lường thủ công hoặc bán tự động, thiếu khả năng phân tích dự báo và tự động hóa toàn diện. Những hạn chế này gây khó khăn trong việc đưa ra các biện pháp can thiệp kịp thời, đặc biệt khi các thông số môi trường thay đổi bất thường.

Sự phát triển của công nghệ IoT, kết hợp với các mô hình dự báo chuỗi thời gian như Prophet, mở ra cơ hội xây dựng các hệ thống giám sát thông minh, không chỉ thu thập và phân tích dữ liệu thời gian thực mà còn dự báo xu hướng chất lượng không khí. Việc tích hợp các công nghệ này trên một nền tảng nhỏ gọn như Raspberry Pi 4 giúp tối ưu hóa chi phí và hiệu quả, đáp ứng nhu cầu giám sát và điều khiển môi trường trong các không gian nhạy cảm như phòng bệnh.

### 1.2 Mục tiêu chọn đề tài

Mục tiêu của đề tài là xây dựng một hệ thống giám sát chất lượng không khí trong phòng bệnh, sử dụng công nghệ IoT để thu thập và phân tích dữ liệu môi trường. Hệ thống này sẽ cung cấp thông tin về các yếu tố như bụi mịn, nhiệt độ và độ ẩm, từ đó đưa ra các giải pháp tự động điều chỉnh nhằm đảm bảo không khí trong phòng bệnh luôn sạch và an toàn cho bệnh nhân. Ngoài ra, hệ thống cũng hỗ trợ giám sát và điều khiển từ xa, giúp tiết kiệm thời gian và tăng cường hiệu quả quản lý.

### 1.3 Lý do chọn đề tài

Việc lựa chọn đề tài "Thiết kế và xây dựng hệ thống giám sát chất lượng môi trường" xuất phát từ nhu cầu cấp thiết trong việc cải thiện chất lượng không khí trong bệnh viện, nơi yêu cầu một môi trường trong lành và an toàn để phục vụ bệnh nhân. Chúng em chọn công nghệ IoT vì khả năng theo dõi và quản lý môi trường thời gian thực, mang lại giải pháp hiệu quả hơn so với các phương pháp truyền thống. Hệ thống này không chỉ giúp bảo vệ sức khỏe bệnh nhân mà còn giảm thiểu tác động tiêu cực đến môi trường, tối ưu hóa việc sử dụng các thiết bị như máy lọc không khí và điều hòa, đồng thời nâng cao chất lượng dịch vụ y tế.

### 1.4 Chức năng

Hệ thống giám sát chất lượng không khí trong phòng bệnh sẽ bao gồm các chức năng chính sau:

* Thu thập và giám sát dữ liệu môi trường: Cảm biến đo các thông số như nhiệt độ, độ ẩm, bụi mịn (PM2.5, PM10).
* Phân tích dữ liệu: Dữ liệu thu thập theo thời gian thực để đưa ra cảnh báo khi các thông số vượt ngưỡng an toàn, Sử dụng mô hình Prophet trên Raspberry Pi 4 để dự báo xu hướng chất lượng không khí, cung cấp thông tin dự đoán và khoảng tin cậy.
* Tự động hóa điều khiển: Hệ thống có thể tự động điều chỉnh các thiết bị như máy lọc không khí, điều hòa để duy trì môi trường tối ưu.
* Giám sát từ xa: Cho phép người dùng theo dõi chất lượng không khí trong phòng bệnh thông qua ứng dụng di động hoặc webserver.
* Cảnh báo và báo cáo: Cung cấp cảnh báo khi các chỉ số môi trường vượt ngưỡng và tạo báo cáo định kỳ về chất lượng không khí.

### 1.5 Các ứng dụng của đề tài

Hệ thống giám sát chất lượng không khí có thể ứng dụng rộng rãi trong các môi trường nhạy cảm như:

* Bệnh viện: Đảm bảo chất lượng không khí trong phòng bệnh, phòng chăm sóc đặc biệt, và khu vực điều trị.
* Trường học và văn phòng: Cải thiện môi trường học tập và làm việc, giảm nguy cơ mắc các bệnh về đường hô hấp do ô nhiễm không khí.
* Nhà ở: Tạo ra môi trường sống lành mạnh và an toàn cho gia đình, đặc biệt là với trẻ em và người già.
* Các khu vực công cộng: Các trung tâm thương mại, nhà ga, sân bay, nơi có nhiều người qua lại cần giám sát chất lượng không khí liên tục.

1. CƠ SỞ LÝ THUYẾT

2.1 WIFI

2.1.1 Giới thiệu

Wi-Fi (viết tắt của Wireless Fidelity) là một công nghệ không dây cho phép các thiết bị kết nối internet và giao tiếp với nhau thông qua sóng radio. Wi-Fi hoạt động dựa trên chuẩn IEEE 802.11 và thường được sử dụng trong các mạng LAN (Local Area Network) để cung cấp kết nối internet không dây cho các thiết bị như máy tính, điện thoại, máy tính bảng, và các thiết bị IoT.

Wi-Fi sử dụng tần số sóng vô tuyến (2.4 GHz và 5 GHz) để truyền tải dữ liệu giữa các thiết bị và điểm truy cập (access point). Các điểm truy cập này kết nối với một mạng có dây, chẳng hạn như mạng internet, và phát sóng tín hiệu Wi-Fi cho các thiết bị trong khu vực phủ sóng.

Wi-Fi được sử dụng rộng rãi trong các không gian công cộng như quán cà phê, sân bay, và các văn phòng, cũng như trong các hộ gia đình để cung cấp kết nối internet cho nhiều thiết bị cùng lúc mà không cần dây cáp.



Hình 2. Biểu tượng WIFI

2.1.2 Công nghệ truyền nhận dữ liệu

IEEE 802.11b (1999):

* Tần số: 2.4GHz.
* Tốc độ: 11Mbps (lý thuyết), thực tế khoảng 4-6Mbps.
* Ưu điểm: Tương đối rẻ, phù hợp cho các ứng dụng cơ bản.
* Nhược điểm: Phạm vi tín hiệu bị hạn chế, dễ bị nhiễu từ các thiết bị như Bluetooth, điện thoại di động, và các mạng không dây khác (ví dụ: lò vi sóng).
* Khả năng mở rộng: Không cao.

IEEE 802.11a (2001):

* Tần số: 5.0GHz.
* Tốc độ: 54Mbps (lý thuyết), thực tế khoảng 15-20Mbps.
* Ưu điểm: Có thể hỗ trợ băng thông lớn hơn và ít bị nhiễu hơn vì hoạt động ở dải tần 5.0GHz.
* Nhược điểm: Phạm vi tín hiệu ngắn hơn và khả năng xuyên qua các vật cản kém hơn so với 802.11b (do tần số cao hơn).
* Khả năng mở rộng: Tốt hơn 802.11b.
* Không tương thích với 802.11b: Vì khác nhau về tần số và phương pháp truyền tín hiệu.

IEEE 802.11g (2003):

* Tần số: 2.4GHz.
* Tốc độ: 54Mbps (lý thuyết), tương thích với chuẩn 802.11b và có hiệu suất truyền cao hơn.
* Ưu điểm: Tương thích ngược với 802.11b, cung cấp tốc độ truyền cao hơn (tương đương với 802.11a) và không dễ bị nhiễu như 802.11b.
* Nhược điểm: Bị nhiễu từ các thiết bị khác ở tần số 2.4GHz, chẳng hạn như Bluetooth.
* Khả năng mở rộng: Tốt hơn 802.11b, nhưng vẫn bị giới hạn bởi tần số 2.4GHz.

IEEE 802.11n (2009)

* Tần số: 2.4GHz và 5GHz (dual-band).
* Tốc độ: Lý thuyết đạt tối đa 600Mbps (thực tế khoảng 150-300Mbps tùy vào số lượng anten).
* Ưu điểm:
* MIMO (Multiple Input Multiple Output): Hỗ trợ nhiều anten, giúp cải thiện tốc độ và phạm vi tín hiệu.
* Tăng cường hiệu suất: Cải thiện tốc độ và độ ổn định so với các chuẩn trước.
* Dual-band: Có thể hoạt động trên cả tần số 2.4GHz và 5GHz, giúp giảm tắc nghẽn trên băng tần 2.4GHz.
* Nhược điểm: Dù tốc độ lý thuyết cao, nhưng phạm vi tín hiệu vẫn bị hạn chế bởi các yếu tố môi trường.
* Khả năng mở rộng: Tốt hơn các chuẩn trước (802.11b/g).

IEEE 802.11ac (2013)

* Tần số: 5.0GHz.
* Tốc độ: Lý thuyết đạt tối đa 1.3Gbps (thực tế khoảng 500-900Mbps, tùy vào thiết bị và môi trường).
* Ưu điểm:
* Tốc độ cao: Cải thiện tốc độ đáng kể so với 802.11n nhờ vào việc sử dụng Wide Channels (80-160MHz) và QAM (Quadrature Amplitude Modulation).
* MU-MIMO (Multi-User MIMO): Cho phép truyền dữ liệu đến nhiều thiết bị cùng lúc mà không làm giảm hiệu suất.
* Tương thích ngược: Hỗ trợ các thiết bị 802.11n và 802.11a.
* Nhược điểm: Dưới 5GHz, tín hiệu có thể dễ bị suy giảm trong môi trường có vật cản dày đặc.
* Khả năng mở rộng: Tốt, nhưng vẫn giới hạn trong phạm vi của tần số 5.0GHz.

IEEE 802.11ax (Wi-Fi 6) (2019)

* Tần số: 2.4GHz và 5GHz (với Wi-Fi 6E hỗ trợ thêm tần số 6GHz).
* Tốc độ: Lý thuyết đạt tối đa 9.6Gbps (thực tế khoảng 1.5-2Gbps).
* Ưu điểm:
* OFDMA (Orthogonal Frequency-Division Multiple Access): Cho phép chia sẻ băng thông hiệu quả hơn giữa nhiều thiết bị, giảm độ trễ và cải thiện hiệu suất mạng khi có nhiều thiết bị kết nối.
* MU-MIMO nâng cao: Hỗ trợ không chỉ trong tải lên mà còn trong tải xuống, giúp nhiều thiết bị có thể truyền nhận đồng thời mà không làm giảm hiệu suất.
* Tăng hiệu suất trong môi trường đông đúc: Wi-Fi 6 tối ưu hóa hiệu suất trong các môi trường có nhiều thiết bị như nhà ở, văn phòng, sân bay.
* Băng tần 6GHz (Wi-Fi 6E): Mở rộng thêm băng tần, giúp giảm tắc nghẽn và tăng khả năng truyền tải dữ liệu.
* Tối ưu hóa hiệu suất và tiết kiệm năng lượng: Sử dụng tính năng Target Wake Time (TWT) giúp các thiết bị tiết kiệm năng lượng khi không sử dụng, rất hữu ích cho các thiết bị IoT.
* Nhược điểm: Cần thiết bị tương thích với Wi-Fi 6 để tận dụng các tính năng mới. Mặc dù hiệu suất cao, nhưng cũng có thể bị giảm tốc trong môi trường nhiễu sóng hoặc khi sử dụng các thiết bị cũ không hỗ trợ Wi-Fi 6.
* Khả năng mở rộng: Rất tốt, phù hợp cho mạng đông đúc, nhiều thiết bị, và môi trường sử dụng các công nghệ mới.

2.1.3 Thành phần của mạng WIFI

Access Point (AP): AP là bộ thu phát không dây LAN (Local Area Network), hoặc là trạm cơ sở có thể kết nối đồng thời một hoặc nhiều thiết bị không dây với Internet.

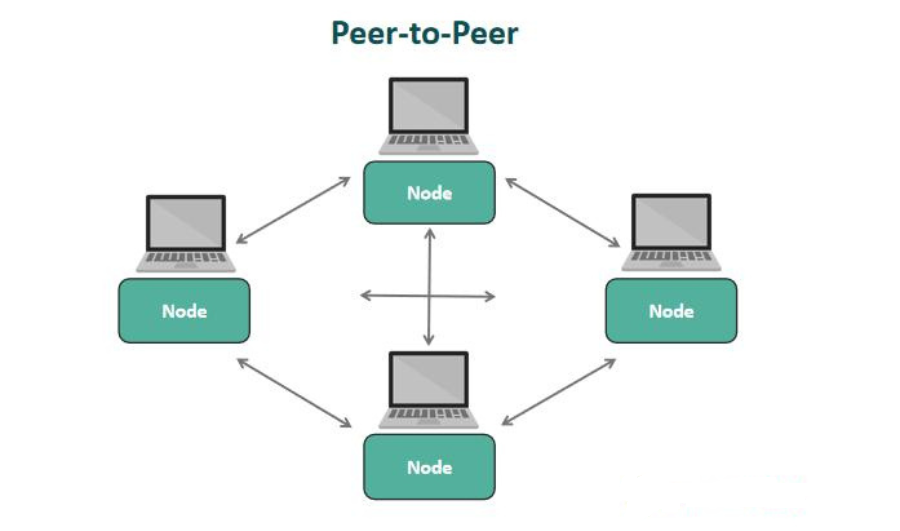
Wifi Card: Cho phép chấp nhận tín hiệu không dây và thông tin chuyển tiếp.

Safeguards: tường lửa và phần mềm chống virus giúp giữ an toàn thông tin cho người dùng.

2.1.4 Cấu trúc liên kết

Peer to peer

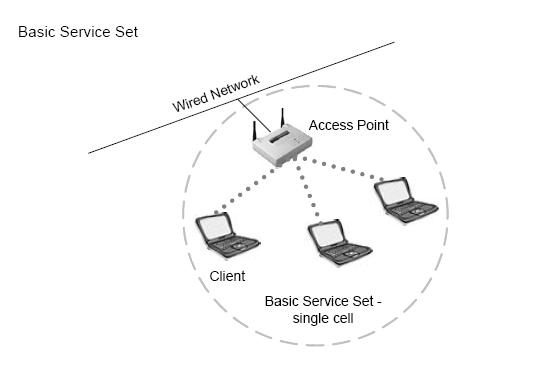
* Là cấu trúc liên kết ngang hàng, không bắt buộc AP, các thiết bị bên trong có thể giao tiếp trực tiếp với nhau.
* Phù hợp để thiết lập mạng không dây một cách nhanh chóng và dễ dàng.



Hình 2. Cấu trúc liên kết ngang hàng

Infrastructure Mode

* Là cấu trúc liên kết dựa trên AP, liên lạc với nhau qua địa điểm truy cập (Access Point). Bất kỳ thông tin truy cập nào đều phải thông qua AP.
* Nếu một trạm di động như máy tính hoặc điện thoại muốn giao tiếp với một trạm di động khác đầu tiên cần phải gửi thông tin đến AP, sau đó AP sẽ gửi ngược lại trạm di động đó.



Hình 2. Cấu trúc liên kết dựa trên AP

2.1.5 Hostpot

Hotspot là một khu vực dễ dàng truy cập mạng không dây.

Hotspot được trang bị kết nối Internet với băng thông rộng và có một hoặc nhiều

AP cho phép người dùng truy cập Internet không dây.

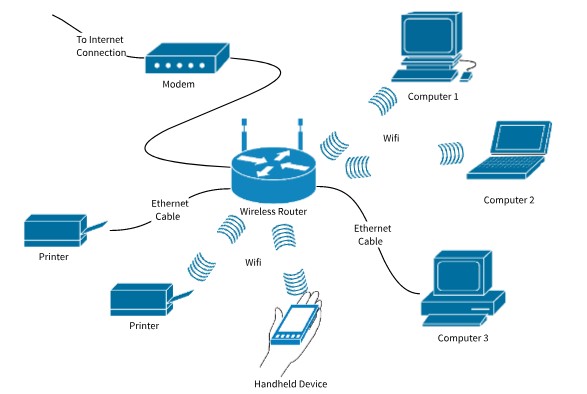
2.1.6 Cách thức hoạt động

Một Wifi Hotspot được tạo ra bằng cách cài đặt điểm truy cập vào kết nối Internet.

Một điểm truy cập hoạt động như một trạm cơ sở.

Khi thiết bị hỗ trợ Wifi bắt gặp điểm phát sóng, thiết bị có thể kết nối không dây với mạng đó.

Một điểm truy cập duy nhất có thể hổ trợ tối đa đến 30 người dùng, nhiều điểm truy cập có thể được kết nối với nhau thông qua cáp Ethernet để tạo ra một mạng lớn.



Hình 2. Cách thức hoạt động của mạng WIFI

2.1.7 Giao tiếp trong WIFI

Trao đổi dữ liệu trong Wifi được chia làm 3 giai đoạn:

* Giai đoạn 1

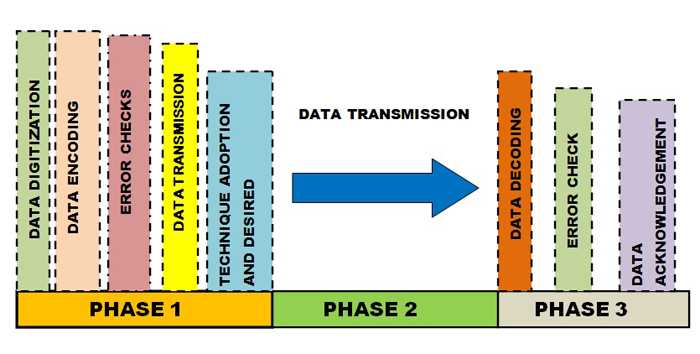
Dữ liệu được chuẩn bị để truyền, nó được mã hóa thay đổi thành tín hiệu số. Tại đây tần số truyền dữ liệu cũng được lựa chọn tùy thuộc vào kỹ thuật sử dụng để gửi tín hiệu không dây.

* Giai đoạn 2

Dữ liệu được truyền thông qua sóng vô tuyến.

* Giai đoạn 3

Dữ liệu được nhận sau đó tiến hành giải mã tín hiệu số, xác nhận và cuối cùng là sử dụng.



Hình 2. Cách thức giao tiếp trong mạng WIFI

2.1.8 Ưu nhược điểm

Ưu điểm

* Dễ dàng cài đặt.
* Tính linh hoạt cao.
* Chi phí phù hợp.
* Độ tin cậy.
* Khả năng bảo mật.
* Tốc độ.

Nhược điểm

* Tiêu thụ điện năng cao.
* Phạm vi hoạt động giới hạn.
* Vấn đề với nhiễu tín hiệu và bảo mật trong môi trường đông đúc hoặc khi có nhiều thiết bị kết nối cùng lúc.

2.1.9 Bảo mật

Từ khi ra đời, Wi-Fi đã gặp phải các vấn đề về bảo mật, khiến cho việc bảo vệ dữ liệu trở thành một yếu tố quan trọng. Các chuẩn bảo mật Wi-Fi đã liên tục được cải tiến để bảo vệ người dùng khỏi những mối đe dọa, bao gồm các cuộc tấn công man-in-the-middle, nghe lén, và truy cập trái phép.

* WEP (Wireless Equivalent Privacy) - 1997:
  + WEP là chuẩn bảo mật đầu tiên của Wi-Fi và được thiết kế để bảo vệ mạng không dây bằng cách mã hóa dữ liệu. Tuy nhiên, WEP đã bị phát hiện có nhiều lỗ hổng bảo mật nghiêm trọng, khiến nó trở thành không an toàn và dễ bị phá vỡ.
  + Hiện nay, WEP không còn được sử dụng trong các mạng Wi-Fi hiện đại.
* WPA (Wi-Fi Protected Access) - 2003:
* WPA được thiết kế để khắc phục các vấn đề của WEP, với mã hóa mạnh mẽ hơn sử dụng TKIP (Temporal Key Integrity Protocol). Tuy nhiên, WPA cũng có những điểm yếu, đặc biệt khi so với các chuẩn bảo mật sau này.
* WPA sử dụng các cơ chế bảo mật mới và đã được sử dụng rộng rãi trong các mạng Wi-Fi trước khi WPA2 ra đời.
* WPA2 (Wi-Fi Protected Access 2) - 2004:
* WPA2 thay thế WPA và là chuẩn bảo mật phổ biến nhất trong các mạng Wi-Fi hiện nay. WPA2 sử dụng AES (Advanced Encryption Standard), một phương thức mã hóa mạnh mẽ hơn nhiều so với TKIP của WPA.
* WPA2 là chuẩn bảo mật tiêu chuẩn cho tất cả các thiết bị Wi-Fi hiện đại và cung cấp một mức độ bảo mật cao, bảo vệ dữ liệu khỏi các mối đe dọa như tấn công từ bên ngoài.
* WPA3 (Wi-Fi Protected Access 3) - 2018:
* WPA3 là chuẩn bảo mật mới nhất và mạnh mẽ nhất hiện nay, cung cấp nhiều cải tiến so với WPA2.
* Cải tiến bảo mật: WPA3 sử dụng một hệ thống mã hóa mạnh mẽ hơn với Simultaneous Authentication of Equals (SAE), giúp chống lại các cuộc tấn công brute-force mà WPA2 không thể ngăn chặn.
* Bảo vệ mạnh mẽ hơn cho mạng công cộng: WPA3 cung cấp bảo mật cho các mạng Wi-Fi mở (không có mật khẩu) nhờ vào tính năng Opportunistic Wireless Encryption (OWE), giúp bảo vệ dữ liệu trong các mạng công cộng như quán cà phê hoặc sân bay.
* Bảo vệ mật khẩu mạnh mẽ hơn: WPA3 sử dụng các biện pháp bảo vệ mật khẩu tốt hơn, giúp các thiết bị khó bị hack hơn.
* WPA3 được coi là chuẩn bảo mật mạnh mẽ nhất cho Wi-Fi hiện nay và đang dần thay thế WPA2 trong các thiết bị mới.

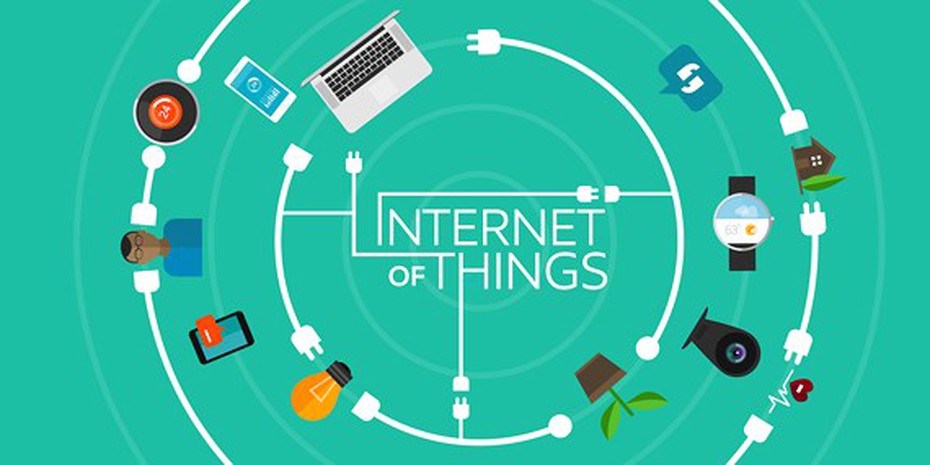
2.2 Tổng quan về Internet of thing

2.2.1 Giới thiệu

Internet of Things (IoT) là một khái niệm mô tả một mạng lưới các thiết bị vật lý được kết nối với Internet, cho phép chúng thu thập, chia sẻ và xử lý dữ liệu một cách tự động. Các thiết bị này có thể bao gồm cảm biến, thiết bị gia dụng, phương tiện giao thông, hệ thống công nghiệp, và nhiều đối tượng khác được tích hợp phần cứng, phần mềm, và kết nối mạng để thực hiện các nhiệm vụ thông minh.

IoT mang đến khả năng liên kết giữa thế giới thực và thế giới số, giúp con người không chỉ giám sát mà còn điều khiển các thiết bị từ xa thông qua Internet. Các thiết bị IoT hoạt động dựa trên sự kết hợp của cảm biến, bộ vi điều khiển, và các công nghệ kết nối như Wi-Fi, Bluetooth, Zigbee, LoRaWAN, hay mạng di động (3G, 4G, 5G).

Các thành phần chính trong một hệ thống IotT: Với một hệ thống IoT chúng sẽ bao gồm 4 thành phần chính đó là thiết bị (Things), trạm kết nối (Gateways), hạ tầng mạng (Network and Cloud) và bộ phân tích và xử lý dữ liệu (Services-creation and Solution Layers).



Hình 2. Internet of Things

2.2.2 Lịch sử hình thành

Khái niệm ban đầu (1960s-1990s):

* Ý tưởng về "kết nối thiết bị" xuất hiện từ những năm 1960 với các hệ thống điều khiển từ xa qua mạng.
* Năm 1982, máy bán nước tự động tại Đại học Carnegie Mellon được xem là thiết bị đầu tiên kết nối internet để cung cấp trạng thái kho hàng.

Thuật ngữ IoT (1999):

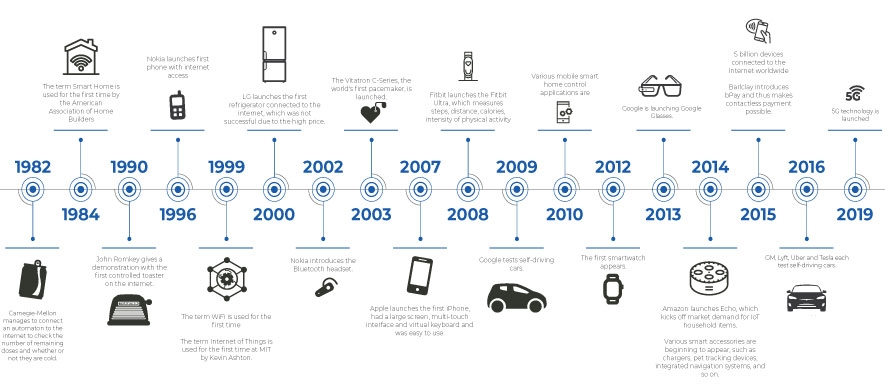
* Kevin Ashton, nhà nghiên cứu tại MIT, lần đầu tiên giới thiệu thuật ngữ Internet of Things trong một bài thuyết trình liên quan đến RFID (Radio-Frequency Identification).

Bùng nổ công nghệ (2000s):

* Sự phát triển mạnh mẽ của cảm biến, vi xử lý giá rẻ, và mạng không dây (Wi-Fi, Bluetooth, 3G/4G) tạo điều kiện cho IoT phát triển.
* Nhiều ứng dụng trong nhà thông minh, công nghiệp và sức khỏe xuất hiện.

Hiện tại (2010s đến nay):

* IoT trở thành một phần quan trọng của cách mạng công nghiệp 4.0.
* Các nền tảng đám mây và AI tăng cường khả năng xử lý dữ liệu IoT.
* Ứng dụng IoT phổ biến trong nhiều lĩnh vực như nông nghiệp, vận tải, năng lượng, y tế, và thành phố thông minh.



Hình 2. Lịch sử hình thành IoT

2.2.3 Ứng dụng

* Nhà thông minh (Smart Home): Điều khiển tự động thiết bị gia dụng và quản lý năng lượng hiệu quả.
* Y tế (Healthcare): Giám sát sức khỏe từ xa và thiết bị đeo thông minh.
* Công nghiệp (Industrial IoT): Tự động hóa sản xuất và bảo trì dự đoán.
* Giao thông (Transportation): Quản lý phương tiện, tối ưu hóa logistics và hỗ trợ xe tự hành.
* Nông nghiệp thông minh (Smart Agriculture): Giám sát cây trồng, chăn nuôi và tối ưu hóa tưới tiêu.
* Thành phố thông minh (Smart City): Quản lý giao thông, năng lượng và môi trường.

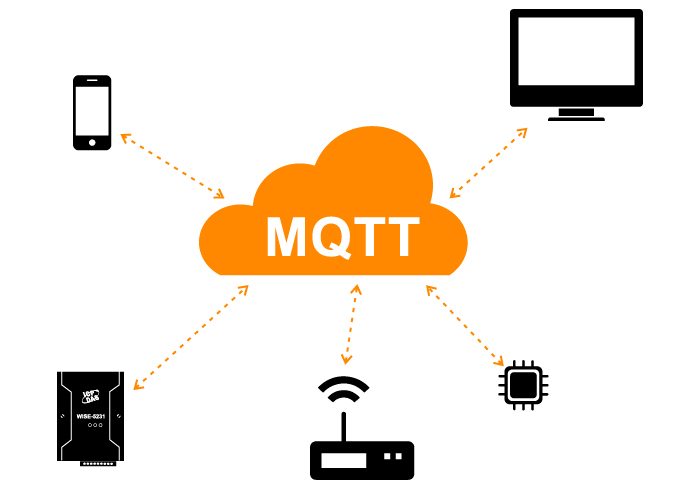
2.3 Giao thức MQTT

2.3.1 Giới thiệu

MQTT (Message Queuing Telemetry Transport) là giao thức truyền thông điệp (message) theo mô hình giao tiếp Publish/Subscribe, phù hợp cho việc vận chuyển dữ liệu từ xa.

Là một giao thức rất nhẹ do đó được sử dụng để giao tiếp các thiết bị (M2M Machine to Machine), WSN (Wireless Sensor Networks) và phổ biến nhất trong các dự án IoT.

Giao thức này được thiết kế để trao đổi dữ liệu giữa máy chủ (server) và khách hàng (client). Ngoài ra với kích thước nhỏ gọn, đơn giản, mức sử dụng năng lượng thấp, các gói dữ liệu được tối ưu hóa và dễ dàng thực hiện đã khiến nó trở nên lý tưởng hơn.



Hình 2. Giao thức MQTT

2.3.2 Lịch sử

MQTT (Message Queuing Telemetry Transport) là một giao thức nhắn tin nhẹ, được phát triển vào năm 1999 bởi Andy Stanford-Clark (IBM) và Arlen Nipper (Arcom), nhằm truyền tải thông tin giữa các thiết bị trong môi trường có băng thông thấp và kết nối không ổn định.

Giao thức này sử dụng mô hình "publish-subscribe", giúp các thiết bị gửi và nhận dữ liệu hiệu quả. MQTT đã được tiêu chuẩn hóa bởi OASIS vào năm 2010 với phiên bản 3.1 và tiếp tục được cải tiến với phiên bản 3.1.1 vào năm 2014. Phiên bản mới nhất, MQTT v5, ra mắt vào năm 2019, với nhiều tính năng nâng cao như cải thiện bảo mật và khả năng mở rộng.

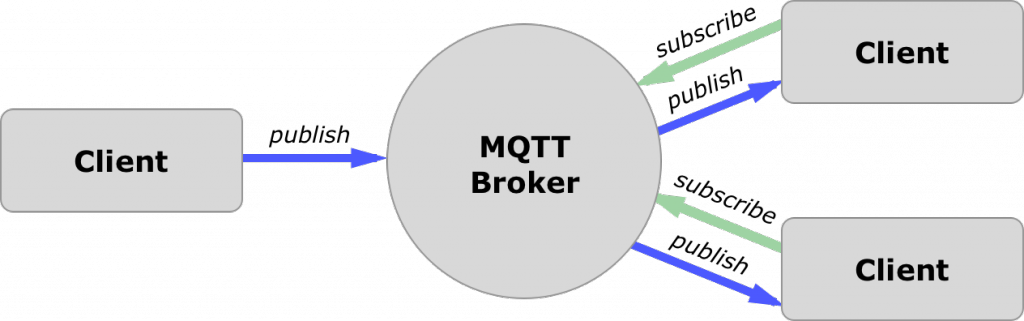
2.3.3 Các thành phần của MQTT

Client (Publisher, Subscriber)

Client (khách hàng) đăng ký một chủ đề để gửi và nhận message:

* Khi một client muốn gửi dữ liệu cho Broker: đây là hoạt động Publisher.
* Khi một client muốn nhận dữ liệu từ Broker: đây là hoạt động Subscriber.

Vì vậy Publisher và Subscriber đóng vai trò đặc biệt của client.



Hình 2. Hoạt động của client

Server (Broker)

Server trong MQTT gọi là một Broker được xem là trung tâm, là điểm giao của các kết nối đến từ client. Nhiệm vụ chính của Broker là nhận đăng ký từ các client về các chủ đề (topic), nhận tin nhắn (message), sắp xếp các message theo hàng đợi rồi chuyển chúng đến một địa chỉ dựa trên việc đăng ký của client. Nhiệm vụ phụ của Broker là có thể đảm nhận thêm một vài tính năng liên quan đến truyền thông như: bảo mật message, lưu trữ message.

Topic

Là nơi mà một client muốn đặt hoặc truy xuất message. Cụ thể khi một message được publish vào một Topic thì những subscriber của Topic đó sẽ nhận được message này.

A diagram of a company

Description automatically generated

Hình 2. Hoạt động của Topic

Message

Là đơn vị trao đổi dữ liệu mà thiết bị nhận được khi Subscribing (đăng ký) một Topic hoặc gửi đi khi Publishing một Topic.

2.3.4 MQTT QoS

QoS (Quality of Service): MQTT cung cấp các mức đảm bảo sự chắc chắn trong việc gửi và nhận dữ liệu giữa các client và Broker. MQTT hổ trợ 3 mức:

QoSO: Đảm bảo mức thấp nhất, dữ liệu được gửi đi đúng một lần và sẽ không được kiểm tra đã đến các Broker hay chưa.

QoS1: Đảm bảo đã đến nơi nhận, ít nhất một lần được xác nhận.

QoS2: Mức đảm bảo cao nhất, Broker đảm bảo các dữ liệu có QoS2 sẽ đến nơi nhận chỉ đúng một lần duy nhất, không bị trùng lặp, không thất lạc dữ liệu, đồng thời tốn băng thông hơn hai cách trên.

2.3.5 Broker trong IoT

Một trong những thành phần của hệ thống IoT là Platform (điện toán đám mây), nó chịu trách nhiệm kết nối các thiết bị với nhau, cho phép người dùng kiểm soát và giám sát các thiết bị của mình. Và Broker cũng chính là Platform, nó nằm trong hệ thống IoT.

Có hai cách tạo ra một Broker:

* Tự tạo Broker MQTT trên máy tính, raspberry, server, ...
* Sử dụng các dịch vụ MQTT Broker có sẵn như CloudMQTT.

A diagram of a server

Description automatically generated

Hình 2. Thành phần của một MQTT trong hệ thống IoT

2.3.6 Ưu điểm

Đây là một giao thức nhẹ. Do đó, dễ dàng thực hiện trong phần mềm và nhanh chóng trong việc truyền nhận dữ liệu, ít bị ảnh hưởng bởi tốc độ mạng.

Giao thức dựa trên kỹ thuật tin nhắn, vì vậy tốc độ khá nhanh.

Gói dữ liệu truyền được tối ưu hóa.

Sử dụng nguồn điện năng thấp, tiếp kiệm năng lượng cho thiết bị được kết nối.

Thời gian thực, đây là điều đặc biệt quan trọng trong các dự án IoT.

2.3.7 Bảo mật

MQTT được thiết kế một cách nhẹ và linh hoạt nhất có thể. Do đó nó chỉ có một lớp bảo mật ở tầng ứng dụng: bảo mật xác thực (xác thực các client được quyền truy cập đến Broker). Tuy nhiên, MQTT vẫn có thể được cài đặt kết hợp với các giải pháp bảo mật đa tầng khác như kết hợp với VNP ở tầng mạng học hoặc SSLL/TLS ở tầng transport.

MQTT được thiết kế nhằm phục vụ truyền thông machine-to-machine nhưng trên thực tế chứng minh nó lại hoạt động một cách linh hoạt hơn mong đợi. Nó hoàn toàn có thể áp dụng cho các kịch bản truyền thông khác nhau: machine to machine, app to app. Chỉ cần có một Broker phù hợp và MQTT Client được cài đặt đúng cách, các thiết bị xây dựng trên nhiều nền tảng khác nhau có thể giao tiếp được với nhau một cách dễ dàng.

2.4 HTTP

2.4.1 Giới thiệu

HTTP (HyperText Transfer Protocol - Giao thức truyền tải siêu văn bản) là một trong các giao thức chuẩn về mạng Internet, được dùng để liên hệ thông tin giữa Máy cung cấp dịch vụ (Web server) và Máy sử dụng dịch vụ (Web client), là giao thức Client/Server dùng cho World Wide Web - WWW. HTTP là một giao thức ứng dụng của bộ giao thức TCP/IP (các giao thức nên tảng cho Internet).

HTTP hoạt động dựa trên mô hình Client - Server. Trong mô hình này, các máy tính của người dùng sẽ đóng vai trò làm máy khách (Client). Sau một thao tác nào đó của người dùng, các máy khách sẽ gửi yêu cầu đến máy chủ (Server) và chờ đợi câu trả lời từ những máy chủ này.

A computer screen shot of a computer screen

Description automatically generated

Hình 2. Giao thức HTTP

2.4.2 Các thành phần chính của HTTP

a) HTTP – Requests

HTTP Request Method: Là phương thức để chỉ ra hành động mong muốn được thực hiện trên tài nguyên đã xác định.

Câu trúc của một HTTP Request:

* Một Request line = Giao thức HTTP (HTTP method) + URL Request (request target) + Phiên bản HTTP (HTTP version). Giao thức HTTP định nghĩa một tập các giao thức GET, POST, HEAD, PUT... Client có thể sử dụng một trong các phương thức đó để gửi request lên server.
* Có thể có hoặc không các trường header
* Một dòng trống để đánh dấu sự kết thúc của các trường Header.
* Request Header Fields: Các trường header cho phép client truyền thông tin bổ sung về yêu cầu, và về chính client, đến server. Một số trường: Accept-Charset, Accept-Encoding, Accept-Language, Authorization, Expect, From, Host, ..

Khi request đến server, server thực hiện một trong 3 hành động sau:

* Server phân tích request nhận được, maps yêu câu với tập tin trong tập tài liệu của server, và trả lại tập tin yêu cầu cho client.
* Server phân tích request nhận được, maps yêu câu vào một chương trình trên server, thực thi chương trình và trả lại kết quả của chương trình đó.
* Request từ client không thể đáp ứng, server trà lại thông báo lỗi

A diagram of a file

Description automatically generated

Hình 2. Cấu trúc HTTP Request

Giao thức HTTP định nghĩa một tập các phương thức request, client có thể sử dụng một trong các phương thức này để tạo request tới HTTP server, dưới đây liệt kê một số phương thức phổ biến.

A screenshot of a computer

Description automatically generated

Hình 2. Phương thức Request

b) HTTP – Responses

Cấu trúc của một HTTP response:

* Một Status line = Phiên bản HTTP + Mã trạng thái + Trạng thái
* Có thể có hoặc không có các trường header
* Một dòng trống để đánh dấu sự kết thúc của các trường header
* Tùy chọn một thông điệp

Mã trạng thái: Thông báo về kết quả khi nhận được yêu câu và xử lí bên server cho client. Các kiểu mã trạng thái:

* 1xx: Thông tin (100 -> 101). VD: 100 (Continue),
* 2xx: Thành công (200 -> 206). VD: 200 (OK), 201 (CREATED), ...
* 3xx: Sự điều hướng lại (300 -> 307). VD: 305 (USE PROXY), ...
* 4xx: Lỗi phía Client (400 -> 417). VD: 403 (FORBIDDEN), 404 (NOT FOUND),

…

* 5xx: Lỗi phía Server (500 -> 505). VD: 500 (INTERNAL SERVER ERROR)

2.5 Webserver

2.5.1 Giới thiệu

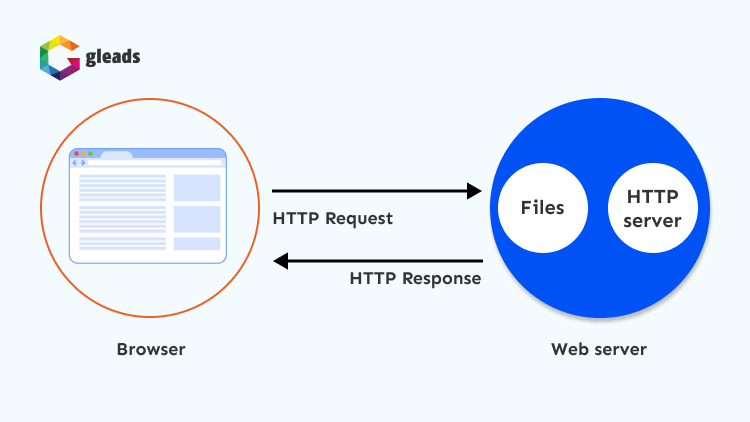
Web server có thể là phần cứng hoặc phần mềm, hoặc cả hai.

Ở khía cạnh phần cứng, web server là một máy tính lưu trữ các file thành phần của một website (các tài liệu, hình ảnh,...) và có thể phân phát chúng tới thiết bị của người dùng. Web server kết nối tới Internet và có thể truy cập tới thông qua một tên miền.

Ở khía cạnh phần mềm, web server điều khiển người sử dụng web truy cập tới các file được lưu trữ trên một HTTP server (máy chủ HTTP). HTTP server là một phần mềm hiểu được các địa chỉ web (URL) và giao thức trình duyệt web (HTTP).

2.5.2 Giao tiếp với Webserver

Khi một trình duyệt cần một file lưu trữ trên một web server, trình duyệt sẽ yêu cầu (request) file đó thông qua HTTP. Khi một yêu cầu gửi tới đúng web server (phần cứng), HTTP server (phần mềm) sẽ gửi file được yêu cầu cũng thông qua HTTP.



Hình 2. Giao tiếp với Webserver

Web server hỗ trợ giao thức HTTP (Giao thức truyền phát siêu văn bản).

HTTP là cách truyền các siêu văn bản giữa hai máy tính. HTTP cung cấp các quy tắc rõ ràng, về cách client và server giao tiếp với nhau:

Chỉ client có thể tạo ra các HTTP request tới các server. Các server chỉ có thể phản hồi HTTP request của client.

Khi yêu cầu một file thông qua HTTP, client phải cung cấp URL của file đó. Web server phải trả lời mọi HTTP request.

Trên web server, HTTP server chịu trách nhiệm xử lý và trả lời các request đã được client gửi đến:

* Khi nhận một request, HTTP server sẽ kiểm tra xem URL được yêu cầu có khớp với một file hiện có không.
* Nếu có, web server gửi nội dung file trả lại client. Nếu không, một application server sẽ tạo ra file cần thiết.
* Nếu không thể xử lý, web server trả lại một thông điệp lỗi cho client.

2.6 FreeRTOS

2.6.1 Tổng quan về FreeRTOS

FreeRTOS là một hệ điều hành thời gian thực (RTOS - Real-Time Operating System) nguồn mở, được thiết kế đặc biệt cho các hệ thống vi điều khiển và các ứng dụng nhúng. FreeRTOS được phát triển bởi Richard Barry và hiện nay thuộc sở hữu của Amazon Web Services (AWS).



Hình 2. FreeRTOS

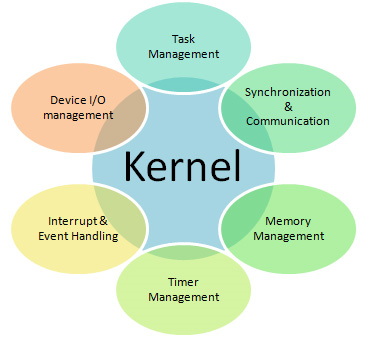
FreeRTOS được phân phối theo giấy phép MIT, cho phép sử dụng trong cả các dự án thương mại và phi thương mại mà không yêu cầu mở mã nguồn của ứng dụng. FreeRTOS đã trở thành một trong những RTOS phổ biến nhất cho các hệ thống nhúng nhờ vào những đặc điểm sau:

* Nhỏ gọn và hiệu quả, chiếm ít tài nguyên hệ thống
* Dễ dàng tích hợp với nhiều vi điều khiển khác nhau
* Hỗ trợ đa nền tảng
* Tính khả chuyển cao giữa các vi điều khiển khác nhau
* Cộng đồng hỗ trợ rộng lớn và tài liệu phong phú

2.6.2 Kiến trúc FreeRTOS

FreeRTOS có kiến trúc đơn giản với ba thành phần chính:

* Kernel (nhân): Quản lý và lập lịch các task, cung cấp các dịch vụ hệ thống cơ bản
* Tác vụ (Tasks): Đại diện cho các tác vụ chương trình độc lập
* Các cơ chế đồng bộ hóa và giao tiếp: Semaphores, mutexes, queues và event groups

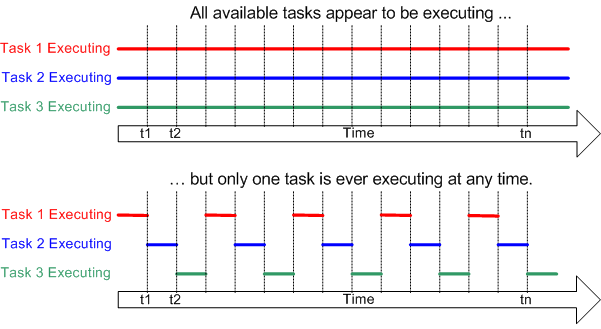


Hình 2. Kiến trúc FreeRTOS

FreeRTOS sử dụng cơ chế lập lịch ưu tiên (priority-based scheduling), trong đó mỗi task được gán một mức độ ưu tiên. Task có mức ưu tiên cao nhất sẽ được thực thi trước. FreeRTOS cũng hỗ trợ chế độ round-robin giữa các task có cùng mức ưu tiên.

2.6.3 Khái Niệm Cốt Lõi và Cơ Chế Hoạt Động của FreeRTOS

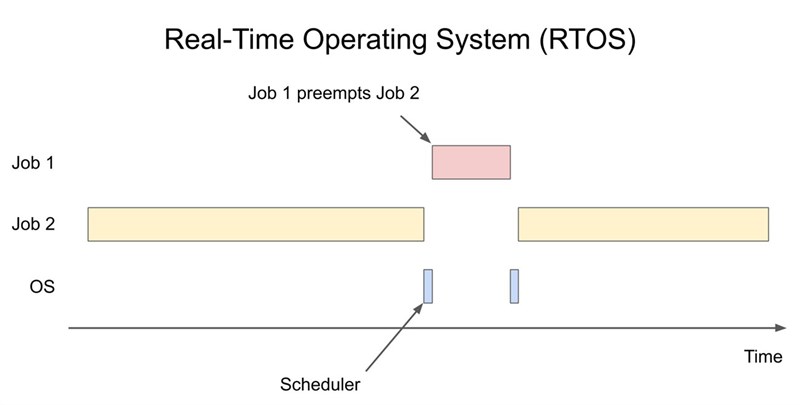
a) Task (Nhiệm vụ):



Hình 2. Task FreeRTOS

* Đơn vị thực thi: Task là một luồng thực thi độc lập trong ứng dụng. Mỗi task có một hàm riêng (task function), thường được viết dưới dạng một vòng lặp vô hạn. Bên trong vòng lặp này, task thực hiện công việc của mình và thường xuyên sử dụng các lời gọi API của FreeRTOS để trì hoãn (delay), chờ đợi sự kiện, hoặc giao tiếp với các task khác.
* Ngữ cảnh (Context): Mỗi task có một ngữ cảnh riêng, bao gồm trạng thái các thanh ghi CPU và con trỏ ngăn xếp (stack pointer). Khi FreeRTOS chuyển đổi giữa các task (context switch), nó sẽ lưu trữ ngữ cảnh của task đang chạy và phục hồi ngữ cảnh của task sắp chạy.
* Ngăn xếp (Stack): Mỗi task được cấp phát một vùng nhớ riêng gọi là ngăn xếp để lưu trữ các biến cục bộ, tham số hàm và địa chỉ trả về. Kích thước ngăn xếp phải đủ lớn để tránh tràn stack (stack overflow), một lỗi phổ biến trong lập trình nhúng đa nhiệm. FreeRTOS cung cấp cơ chế phát hiện tràn stack (tùy chọn cấu hình).
* Mức ưu tiên (Priority): Mỗi task được gán một mức ưu tiên (số nguyên không âm, 0 là thấp nhất, configMAX\_PRIORITIES - 1 là cao nhất). Bộ lập lịch sử dụng mức ưu tiên này để quyết định task nào được chạy. Task Idle (thực thi khi không có task nào khác ở trạng thái Ready) luôn có ưu tiên 0.
* Trạng thái Task: Một task có thể ở một trong các trạng thái sau:
  + Running: Task đang thực sự chiếm giữ CPU và thực thi mã lệnh. Trên hệ thống đơn nhân, chỉ có một task ở trạng thái Running tại một thời điểm.
  + Ready: Task đã sẵn sàng để chạy (không bị chặn bởi bất kỳ lý do gì) nhưng đang chờ CPU rảnh hoặc chờ một task có ưu tiên cao hơn kết thúc.
  + Blocked: Task đang chờ một sự kiện xảy ra (ví dụ: chờ dữ liệu trên queue, chờ semaphore được giải phóng, chờ hết thời gian delay). Task ở trạng thái này không tiêu tốn CPU.
  + Suspended: Task đã bị tạm dừng một cách tường minh (bằng lời gọi vTaskSuspend()) và sẽ không được lập lịch cho đến khi được đánh thức lại (bằng vTaskResume()).
* Tạo và Xóa Task: Task được tạo động trong quá trình chạy bằng các hàm như xTaskCreate() hoặc xTaskCreateStatic() (cấp phát bộ nhớ tĩnh). Có thể xóa task bằng vTaskDelete(). Việc quản lý vòng đời task là một phần quan trọng của thiết kế ứng dụng.

b. Scheduler (Bộ lập lịch):

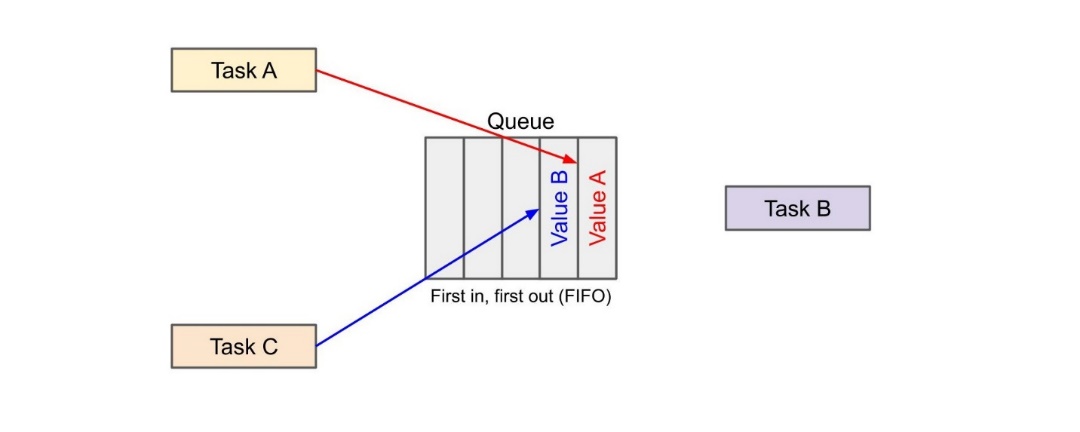


Hình 2. Scheduler FreeRTOS

* Trái tim của RTOS: Bộ lập lịch là thành phần quyết định xem task nào trong số các task ở trạng thái Ready sẽ được chuyển sang trạng thái Running.
* Lập lịch Ưu tiên Phủ đầu (Preemptive Priority-Based Scheduling): Đây là cơ chế mặc định và phổ biến nhất trong FreeRTOS. Nguyên tắc hoạt động:
  + Bộ lập lịch luôn đảm bảo rằng task có mức ưu tiên cao nhất trong số các task đang ở trạng thái Ready sẽ được chạy.
  + Nếu một task có ưu tiên cao hơn trở thành Ready (ví dụ: do nhận được dữ liệu từ queue, hoặc một task ưu tiên cao hơn kết thúc delay), nó sẽ ngay lập tức phủ đầu (preempt) task đang chạy có ưu tiên thấp hơn. Ngữ cảnh của task bị phủ đầu sẽ được lưu lại, và task mới có ưu tiên cao hơn sẽ bắt đầu chạy.
  + Khi task ưu tiên cao kết thúc hoặc bị block, task có ưu tiên cao nhất tiếp theo trong hàng đợi Ready sẽ được phục hồi và chạy tiếp.
* Chia sẻ thời gian: Nếu được cấu hình (configUSE\_TIME\_SLICING = 1), các task có cùng mức ưu tiên cao nhất và đang ở trạng thái Ready sẽ chia sẻ thời gian CPU theo kiểu round-robin. Mỗi task sẽ được chạy trong một khoảng thời gian gọi là "time slice" (thường bằng một hoặc nhiều tick hệ thống). Khi hết time slice, nếu vẫn còn task khác cùng ưu tiên đang Ready, bộ lập lịch sẽ chuyển sang task tiếp theo. Điều này đảm bảo sự công bằng giữa các task cùng ưu tiên.
* Lập lịch Hợp tác (Cooperative Scheduling): Nếu configUSE\_PREEMPTION = 0, bộ lập lịch sẽ hoạt động theo kiểu hợp tác. Một task sẽ tiếp tục chạy cho đến khi nó tự nguyện nhường CPU bằng cách gọi một hàm blocking (như vTaskDelay, xQueueReceive) hoặc gọi tường minh taskYIELD(). Cơ chế này đơn giản hơn nhưng đòi hỏi các task phải "hợp tác" tốt để tránh việc một task chiếm giữ CPU quá lâu.
* Tick Interrupt: Hoạt động của bộ lập lịch thường được điều khiển bởi một ngắt định kỳ từ bộ định thời phần cứng, gọi là System Tick (thường 1ms đến 10ms). Trong trình phục vụ ngắt Tick (Tick ISR), FreeRTOS cập nhật thời gian hệ thống, kiểm tra xem có task nào cần được chuyển từ trạng thái Blocked sang Ready do hết thời gian chờ (timeout hoặc delay), và có thể yêu cầu một context switch nếu cần (ví dụ: hết time slice hoặc có task ưu tiên cao hơn vừa trở thành Ready).
* Khóa Bộ lập lịch (Scheduler Locking): Đôi khi, ứng dụng cần thực hiện một chuỗi các thao tác ngắn mà không muốn bị gián đoạn bởi việc chuyển task (ví dụ: cập nhật các cấu trúc dữ liệu được chia sẻ bởi nhiều task mà không muốn dùng mutex). FreeRTOS cung cấp vTaskSuspendAll() để tạm dừng bộ lập lịch và xTaskResumeAll() để kích hoạt lại. Việc này cần được sử dụng cẩn thận vì nó ảnh hưởng đến tính thời gian thực của hệ thống.

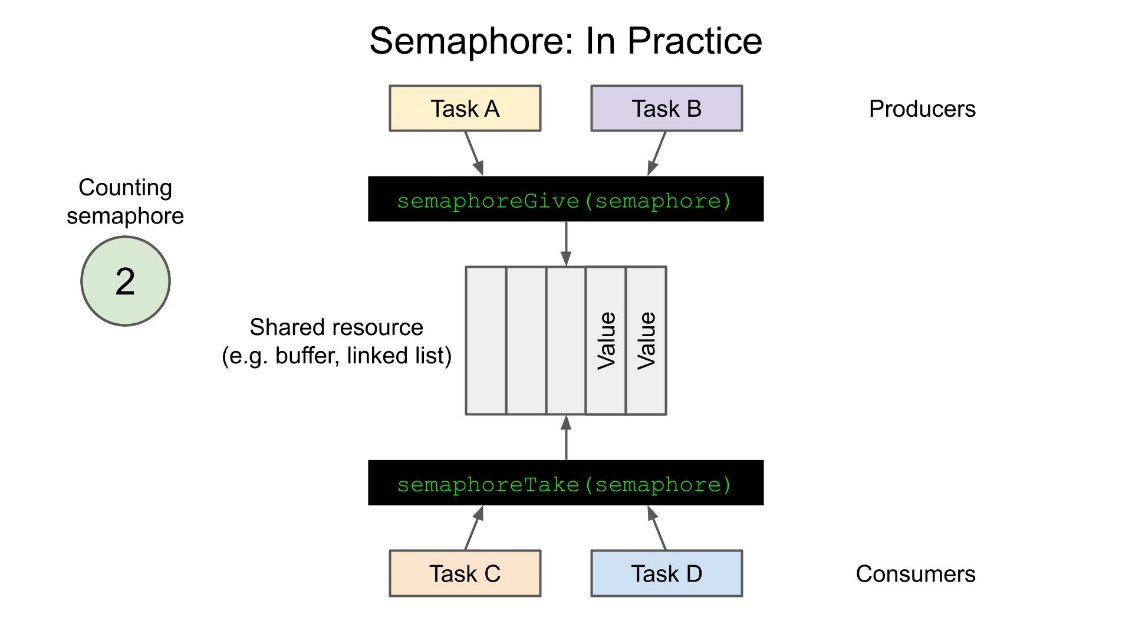
c. Giao tiếp và Đồng bộ hóa giữa các Task (Inter-Task Communication - ITC & Synchronization):

* Queues (Hàng đợi):



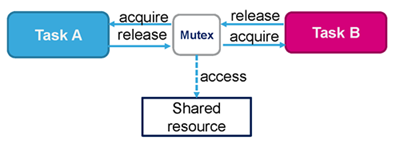
Hình 2. Queue FreeRTOS

* + Cơ chế: Là cơ chế chính để truyền dữ liệu giữa các task hoặc giữa ISR và task. Hoạt động theo nguyên tắc FIFO (First-In, First-Out). Một task (producer) gửi dữ liệu vào cuối hàng đợi (xQueueSend), và một task khác (consumer) nhận dữ liệu từ đầu hàng đợi (xQueueReceive).
  + Blocking và Timeout: Các hàm thao tác với queue có thể là blocking. Nếu queue đầy, task gửi sẽ bị block (chuyển sang trạng thái Blocked) cho đến khi có chỗ trống hoặc hết thời gian chờ (timeout). Tương tự, nếu queue rỗng, task nhận sẽ bị block cho đến khi có dữ liệu hoặc hết timeout. Việc sử dụng timeout giúp tránh tình trạng task bị block vô thời hạn.
  + Loại dữ liệu: Queue có thể dùng để truyền các bản sao (copy) của dữ liệu (cho các kiểu dữ liệu nhỏ) hoặc truyền con trỏ trỏ đến vùng dữ liệu lớn hơn.
  + Sử dụng trong ISR: Có các phiên bản hàm đặc biệt (xQueueSendFromISR, xQueueReceiveFromISR) an toàn để gọi từ bên trong các Trình phục vụ Ngắt (ISR). Các hàm này không được block và thường kết hợp với portYIELD\_FROM\_ISR để yêu cầu chuyển ngữ cảnh ngay sau khi ISR kết thúc nếu việc gửi/nhận dữ liệu đã đánh thức một task có ưu tiên cao hơn.
  + Ứng dụng: Truyền dữ liệu cảm biến từ task thu thập sang task xử lý, gửi lệnh từ task giao diện người dùng sang task điều khiển.
* Semaphores (Đèn tín hiệu):



Hình 2. Semaphores FreeRTOS

* + Mục đích: Dùng chủ yếu cho việc đồng bộ hóa và kiểm soát truy cập tài nguyên. Semaphore hoạt động như một bộ đếm có giới hạn.
  + Binary Semaphores: Chỉ có thể có giá trị 0 hoặc 1. Thường được dùng để báo hiệu (signaling) một sự kiện đã xảy ra. Một task hoặc ISR "give" (xSemaphoreGive) semaphore để báo hiệu, và một task khác "take" (xSemaphoreTake) semaphore để chờ đợi tín hiệu đó. Việc "take" sẽ block task nếu semaphore chưa được "give" (giá trị là 0).
  + Counting Semaphores: Có thể có giá trị từ 0 đến một giá trị tối đa định trước. Mỗi lần "take" thành công sẽ giảm bộ đếm đi 1, mỗi lần "give" sẽ tăng bộ đếm lên 1 (không vượt quá giá trị tối đa). Dùng để kiểm soát truy cập đến một số lượng giới hạn các tài nguyên giống hệt nhau (ví dụ: quản lý một pool các bộ đệm nhớ).
  + Ứng dụng: Đồng bộ hóa task với ISR (ISR "give", task "take"), giới hạn số lượng task có thể truy cập một dịch vụ cùng lúc.
* Mutexes (Mutual Exclusion):



Hình 2. Mutexes FreeRTOS

* + Mục đích: Được thiết kế đặc biệt để bảo vệ các tài nguyên dùng chung (shared resources) hoặc các đoạn mã tới hạn (critical sections) khỏi sự truy cập đồng thời của nhiều task, tránh tình trạng race condition.
  + Cơ chế hoạt động: Mutex tương tự như một binary semaphore nhưng có thêm khái niệm "chủ sở hữu" (ownership). Chỉ task đã "take" (xSemaphoreTake) thành công mutex mới có thể "give" (xSemaphoreGive) nó trở lại. Bất kỳ task nào khác cố gắng "take" mutex đang bị giữ sẽ bị block.
  + Priority Inheritance (Thừa kế Ưu tiên): Đây là một cơ chế quan trọng của mutex trong FreeRTOS để giải quyết vấn đề Priority Inversion (Đảo ngược Ưu tiên). Priority Inversion xảy ra khi một task ưu tiên thấp (L) đang giữ một mutex mà task ưu tiên cao (H) cần. Trong lúc đó, một task ưu tiên trung bình (M) chạy và chiếm CPU, khiến task H không thể lấy được mutex và bị trì hoãn bởi task M (dù M không liên quan đến mutex). Với Priority Inheritance, khi task H bị block chờ mutex do L giữ, task L sẽ tạm thời được nâng lên mức ưu tiên của H. Điều này giúp L chạy nhanh hơn, giải phóng mutex sớm hơn, cho phép H tiếp tục công việc của mình, giảm thiểu thời gian bị đảo ngược ưu tiên.
  + Recursive Mutexes: Cho phép cùng một task có thể "take" cùng một mutex nhiều lần. Task đó phải "give" lại đúng số lần tương ứng thì mutex mới được giải phóng hoàn toàn cho các task khác.
  + Ứng dụng: Bảo vệ truy cập vào các biến toàn cục, cấu trúc dữ liệu chung, hoặc các thiết bị ngoại vi không thể sử dụng đồng thời (ví dụ: một cổng UART, màn hình LCD).
* Event Groups (Nhóm Sự kiện):
  + Mục đích: Cung cấp một cơ chế đồng bộ hóa mạnh mẽ hơn, cho phép task chờ đợi một hoặc nhiều sự kiện (kết hợp logic AND/OR) xảy ra.
  + Cơ chế: Mỗi event group quản lý một tập các bit sự kiện (thường là 24 bit). Task hoặc ISR có thể "set" (xEventGroupSetBits) một hoặc nhiều bit. Các task khác có thể chờ (xEventGroupWaitBits) một tổ hợp các bit được set, với tùy chọn xóa bit sau khi chờ thành công hoặc giữ nguyên bit. Có thể chờ tất cả các bit được set (AND) hoặc chờ bất kỳ bit nào được set (OR).
  + Ứng dụng: Chờ nhiều điều kiện đầu vào được thỏa mãn trước khi thực hiện một hành động phức tạp, ví dụ: chờ cả kết nối WiFi thành công và dữ liệu cảm biến sẵn sàng.
* Task Notifications (Thông báo Task):
  + Mục đích: Là một cơ chế giao tiếp và đồng bộ hóa trực tiếp giữa các task (hoặc ISR và task) rất nhẹ và hiệu quả. Mỗi task có một "hộp thư" thông báo 32-bit.
  + Cơ chế: Một task/ISR có thể gửi thông báo trực tiếp đến một task cụ thể (xTaskNotifyGive, xTaskNotify) mà không cần tạo một đối tượng trung gian như queue hay semaphore. Task nhận có thể chờ thông báo (ulTaskNotifyTake, xTaskNotifyWait). Thông báo có thể hoạt động như binary semaphore, counting semaphore, hoặc truyền một giá trị 32-bit.
  + Ưu điểm: Nhanh hơn và tốn ít RAM hơn so với queues/semaphores cho các trường hợp sử dụng đơn giản.
  + Nhược điểm: Chỉ dùng cho giao tiếp 1-1 (hoặc N-1 từ ISR). Không thể gửi vào "hộp thư" của task nếu nó đầy (thông báo sẽ bị mất nếu dùng xTaskNotify).
  + Ứng dụng: Thay thế cho các trường hợp sử dụng đơn giản của binary semaphore hoặc queue một phần tử.

d. Quản lý Bộ nhớ (Memory Management):

* Nhu cầu cấp phát động: Nhiều ứng dụng cần cấp phát bộ nhớ động trong quá trình chạy (ví dụ: tạo task, queue, cấp phát bộ đệm). Các hàm malloc() và free() chuẩn của thư viện C có thể không phù hợp với hệ thống thời gian thực vì:
  + Không xác định thời gian (Non-deterministic): Thời gian thực thi có thể thay đổi tùy thuộc vào trạng thái heap.
  + Phân mảnh (Fragmentation): Việc cấp phát và giải phóng các khối bộ nhớ có kích thước khác nhau lặp đi lặp lại có thể tạo ra nhiều vùng trống nhỏ xen kẽ, khiến việc cấp phát một khối lớn sau này thất bại dù tổng bộ nhớ trống vẫn còn nhiều.
  + Không an toàn luồng (Not thread-safe): Cần có cơ chế bảo vệ (như mutex) nếu được gọi từ nhiều task, làm tăng độ phức tạp và overhead.
* Các Schema Cấp phát của FreeRTOS: FreeRTOS cung cấp nhiều thuật toán quản lý heap khác nhau (heap\_1 đến heap\_5), cho phép người dùng lựa chọn phù hợp với yêu cầu ứng dụng:
  + heap\_1.c (Simplest, No Free): Chỉ cấp phát, không giải phóng. Đơn giản nhất, không phân mảnh, xác định thời gian. Phù hợp khi chỉ cấp phát bộ nhớ lúc khởi tạo.
  + heap\_2.c (Allows Free, Fragmentation Risk): Cho phép giải phóng bộ nhớ, nhưng có thể gây phân mảnh. Ít dùng hơn hiện nay.
  + heap\_3.c (Wrapper for Standard Library): Đơn giản là một wrapper quanh malloc() và free() của trình biên dịch, đảm bảo thread-safety bằng cách tạm dừng scheduler. Yêu cầu trình biên dịch cung cấp malloc/free và cần cấu hình heap cho linker.
  + heap\_4.c (Coalescing Algorithm, Less Fragmentation): Cho phép cấp phát và giải phóng. Sử dụng thuật toán first-fit và tự động gộp các khối nhớ trống liền kề để giảm phân mảnh. Xác định thời gian hơn heap\_2. Lựa chọn phổ biến cho nhiều ứng dụng.
  + heap\_5.c (Multiple Memory Regions): Tương tự heap\_4 nhưng cho phép khởi tạo heap trên nhiều vùng nhớ không liền kề.
* API: FreeRTOS cung cấp pvPortMalloc() và vPortFree() tương ứng với malloc() và free().
* Lưu ý trên ESP-IDF: ESP-IDF thường sử dụng heap\_caps API, một hệ thống quản lý heap phức tạp hơn cho phép cấp phát bộ nhớ từ các vùng nhớ có thuộc tính khác nhau (ví dụ: RAM nội bộ nhanh, RAM ngoài chậm hơn, RAM có khả năng DMA). Bên dưới, nó có thể sử dụng các cơ chế tương tự như heap\_4 hoặc heap\_5.

e. Quản lý Thời gian (Time Management):

* System Tick: Là nhịp đập cơ bản của hệ thống thời gian trong FreeRTOS. Được tạo ra bởi một bộ định thời phần cứng với tần số cấu hình được (configTICK\_RATE\_HZ, ví dụ 100Hz hoặc 1000Hz). Mỗi tick tương ứng với một khoảng thời gian nhỏ (10ms hoặc 1ms). Độ phân giải thời gian của các hàm delay và timeout phụ thuộc vào tần số tick này. Tần số cao hơn cho độ phân giải tốt hơn nhưng cũng tăng overhead cho CPU do phải xử lý ngắt tick thường xuyên hơn.
* Delay Functions:
  + vTaskDelay(ticks): Đưa task gọi vào trạng thái Blocked trong một số lượng ticks xác định. Đây là delay tương đối.
  + vTaskDelayUntil(&previousWakeTime, increment): Đưa task vào trạng thái Blocked cho đến một thời điểm tuyệt đối trong tương lai. previousWakeTime lưu thời điểm task được đánh thức lần trước, increment là khoảng thời gian cần delay. Hàm này rất hữu ích để tạo các task định kỳ (periodic tasks) với chu kỳ chính xác, tránh bị tích lũy sai số (drift) như khi dùng vTaskDelay trong vòng lặp.
* Software Timers (Bộ định thời Phần mềm):
  + Khái niệm: Cho phép thực thi một hàm callback sau một khoảng thời gian định trước mà không cần tạo một task riêng chỉ để delay. FreeRTOS quản lý các software timer này thông qua một task hệ thống riêng gọi là Timer Daemon Task (hoặc Timer Service Task).
  + Loại Timer: Có hai loại: One-shot timers (chỉ chạy một lần sau khi được start) và Auto-reload timers (tự động chạy lại sau mỗi chu kỳ định trước).
  + Hoạt động: Khi một timer hết hạn, Timer Daemon Task sẽ gọi hàm callback tương ứng đã được đăng ký. Do đó, các hàm callback này phải thực thi nhanh chóng và không được gọi các hàm blocking của FreeRTOS.
  + API:xTimerDelete()**,** xTimerChangePeriod()**,** xTimerStop()**,** xTimerStart(), xTimerCreate()
  + xTimerDelete()Ứng dụng: Thực hiện các công việc định kỳ không quá gấp (ví dụ: kiểm tra trạng thái pin mỗi phút, gửi tín hiệu keep-alive mỗi 30 giây, debounce nút nhấn).

2.6.4 Lợi ích của FreeRTOS trong phát triển ứng dụng ESP32

* Cấu trúc hóa ứng dụng: FreeRTOS cho phép chia nhỏ logic phức tạp của đồ án thành các task độc lập, mỗi task chịu trách nhiệm cho một chức năng cụ thể (ví dụ: Task đọc cảm biến A, Task đọc cảm biến B, Task xử lý dữ liệu, Task giao tiếp WiFi/Bluetooth, Task điều khiển động cơ, Task cập nhật hiển thị LCD). Điều này làm cho mã nguồn trở nên rõ ràng, dễ hiểu, dễ quản lý và gỡ lỗi hơn so với một cấu trúc super-loop lớn.
* Đảm bảo tính đáp ứng: Bằng cách gán ưu tiên phù hợp cho các task, có thể đảm bảo rằng các hoạt động quan trọng (ví dụ: xử lý tín hiệu cảnh báo khẩn cấp, phản hồi lệnh điều khiển tức thời) được ưu tiên thực thi, ngay cả khi các task khác đang bận rộn. Điều này rất quan trọng trong các ứng dụng cần phản ứng nhanh với môi trường hoặc người dùng.
* Xử lý đồng thời hiệu quả: Trong nhiều đồ án ESP32, cần thực hiện nhiều việc "cùng lúc" - ví dụ, vừa liên tục đọc dữ liệu từ cảm biến, vừa xử lý tính toán, vừa duy trì kết nối mạng và gửi/nhận dữ liệu. FreeRTOS cung cấp cơ chế lập lịch và chuyển đổi ngữ cảnh cho phép các hoạt động này diễn ra xen kẽ một cách hiệu quả, tạo ảo giác về sự đồng thời thực sự.
* Tận dụng kiến trúc Dual-Core của ESP32: FreeRTOS trong ESP-IDF cho phép gán các task chạy trên từng lõi CPU cụ thể thông qua hàm xTaskCreatePinnedToCore. Nhờ đó, công việc có thể được phân chia một cách hợp lý — ví dụ, một lõi có thể đảm nhận các tác vụ yêu cầu xử lý tính toán cao hoặc yêu cầu thời gian thực nghiêm ngặt (như xử lý tín hiệu), trong khi lõi còn lại có thể đảm nhiệm các tác vụ như giao tiếp mạng hoặc các tác vụ ít nhạy cảm về thời gian. Cách tiếp cận này giúp cải thiện hiệu suất tổng thể của hệ thống một cách đáng kể.
* Đơn giản hóa xử lý I/O Blocking: Nhiều hoạt động I/O (như đọc/ghi file, giao tiếp mạng) có thể mất thời gian chờ đợi. Trong môi trường không có RTOS, việc chờ đợi này sẽ chặn toàn bộ hệ thống. Với FreeRTOS, task thực hiện I/O có thể bị block trong khi chờ, cho phép các task khác tiếp tục chạy, cải thiện hiệu quả sử dụng CPU. Các driver của ESP-IDF thường tích hợp sẵn với FreeRTOS để hỗ trợ hoạt động non-blocking hoặc blocking hiệu quả.
* Cộng đồng hỗ trợ lớn và tài liệu phong phú: FreeRTOS là một dự án mã nguồn mở rất phổ biến, có cộng đồng người dùng đông đảo và rất nhiều tài liệu hướng dẫn, ví dụ mã nguồn, diễn đàn thảo luận. Khi gặp vấn đề trong quá trình phát triển đồ án, khả năng tìm kiếm được sự trợ giúp là rất cao.

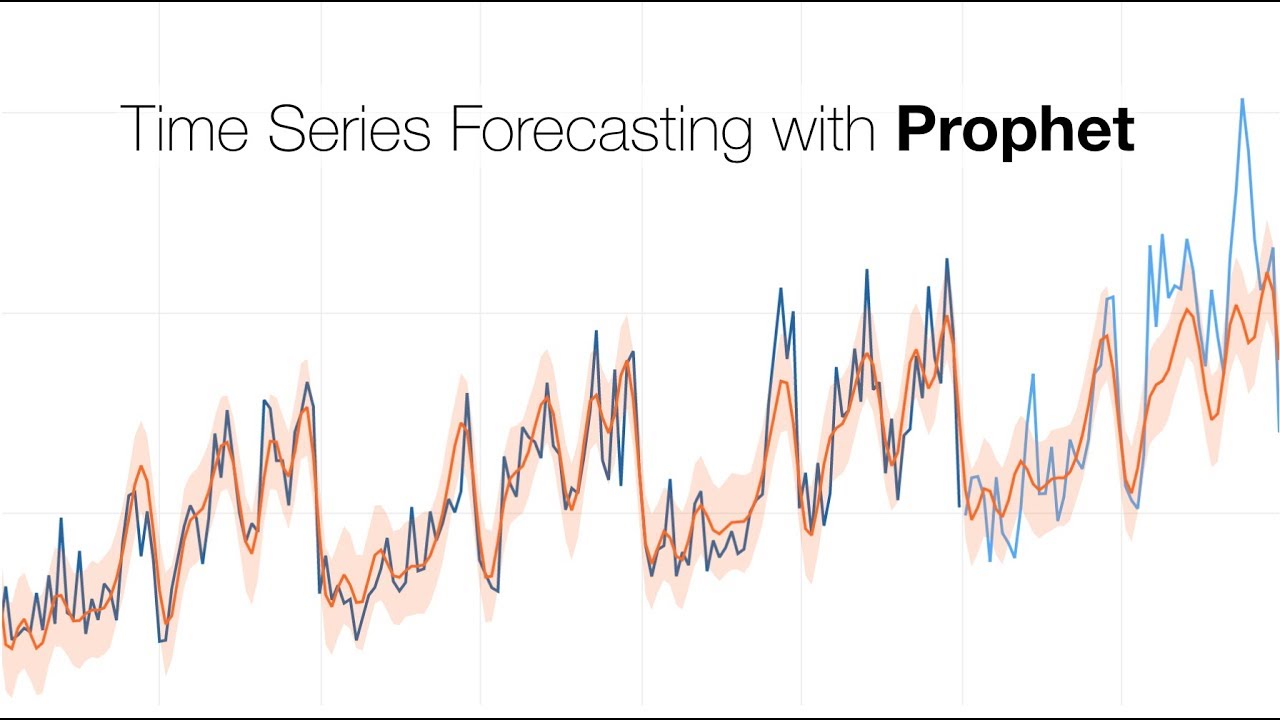
2.7 Giới thiệu Mô hình Dự báo Chuỗi Thời gian Prophet

2.7.1 Giới thiệu

Prophet là một thư viện mã nguồn mở được phát triển bởi đội ngũ Core Data Science tại Facebook (nay là Meta). Nó được thiết kế đặc biệt để dự báo dữ liệu chuỗi thời gian (time series data) có các đặc điểm thường thấy trong môi trường kinh doanh và thực tế:

* Có tính mùa vụ (seasonality) mạnh mẽ ở nhiều cấp độ (hàng ngày, hàng tuần, hàng năm).
* Chứa các ngày lễ hoặc sự kiện đặc biệt (holidays/special events) xảy ra không đều đặn.
* Có thể có dữ liệu bị thiếu (missing data) hoặc các giá trị ngoại lai (outliers).
* Xu hướng (trend) có thể thay đổi đột ngột (ví dụ: do ra mắt sản phẩm mới, thay đổi thị trường).

Prophet hướng đến việc tạo ra các dự báo chất lượng cao một cách tự động hoặc bán tự động, dễ dàng tinh chỉnh bởi cả những người không phải là chuyên gia sâu về mô hình chuỗi thời gian.



Hình 2. Prophet time-series forecasting

2.7.2 Cấu trúc Mô hình

Về cơ bản, Prophet là một mô hình cộng tính (additive model), nơi chuỗi thời gian y(t) được phân rã thành các thành phần chính:

y(t) = g(t) + s(t) + h(t) + εt

Trong đó:

* g(t) (Trend - Xu hướng): Đại diện cho sự tăng trưởng hoặc suy giảm phi chu kỳ (non-periodic) trong chuỗi thời gian. Prophet mô hình hóa xu hướng theo hai dạng chính:
  + Mô hình tăng trưởng bão hòa (Saturating Growth / Logistic Growth): Phù hợp khi xu hướng có giới hạn trên hoặc dưới tự nhiên (ví dụ: dung lượng thị trường).
  + Mô hình tuyến tính từng khúc (Piecewise Linear Model): Cho phép xu hướng thay đổi tốc độ tại các điểm cụ thể gọi là changepoints. Prophet có thể tự động phát hiện các changepoints này hoặc cho phép người dùng chỉ định.
* s(t) (Seasonality - Tính mùa vụ): Đại diện cho các thay đổi có tính chu kỳ (periodic changes) như tính mùa vụ hàng tuần, hàng năm. Prophet sử dụng chuỗi Fourier (Fourier series) để mô hình hóa các hiệu ứng mùa vụ, cho phép nó linh hoạt mô hình hóa các chu kỳ phức tạp và không nhất thiết phải là số nguyên (ví dụ: chu kỳ 30.5 ngày).
* h(t) (Holidays/Events - Ngày lễ/Sự kiện): Đại diện cho ảnh hưởng của các sự kiện đặc biệt, không đều đặn nhưng có thể dự đoán được (ví dụ: Tết Nguyên Đán, Black Friday, ngày ra mắt sản phẩm). Người dùng cần cung cấp danh sách các ngày lễ/sự kiện và khoảng thời gian ảnh hưởng (trước và sau sự kiện) để mô hình học được tác động của chúng.
* εt (Error Term - Sai số): Đại diện cho những biến động ngẫu nhiên không được giải thích bởi mô hình. Prophet thường giả định rằng sai số này tuân theo phân phối chuẩn (Normal distribution).

2.7.3 Nguyên lý hoạt động và Ưu điểm

* Phân rã thành phần: Cách tiếp cận phân rã giúp mô hình dễ diễn giải hơn. Người dùng có thể trực quan hóa và hiểu rõ từng thành phần (xu hướng, mùa vụ, ngày lễ) đóng góp vào dự báo cuối cùng như thế nào.
* Xử lý linh hoạt: Prophet được thiết kế để hoạt động tốt ngay cả khi có dữ liệu thiếu hoặc giá trị ngoại lai mà không cần tiền xử lý quá phức tạp. Nó cũng linh hoạt trong việc xử lý các chu kỳ mùa vụ khác nhau và các điểm thay đổi xu hướng.
* Tự động hóa và Tùy chỉnh: Mô hình có khả năng tự động phát hiện nhiều tham số (như changepoints, cường độ mùa vụ), nhưng cũng cung cấp các tham số dễ hiểu để người dùng có thể tinh chỉnh (ví dụ: độ linh hoạt của xu hướng, độ mạnh của mùa vụ, danh sách ngày lễ).
* Framework Bayes: Việc ước lượng tham số trong Prophet được thực hiện trong một khuôn khổ Bayes, cho phép nó cung cấp không chỉ dự báo điểm (point forecast) mà còn cả khoảng tin cậy (uncertainty intervals), thể hiện mức độ không chắc chắn của dự báo.

2.8 Node-RED

2.8.1 Giới thiệu

Node-RED là một công cụ phát triển đồ họa giúp kết nối các thiết bị phần cứng, API và dịch vụ trực tuyến với nhau một cách dễ dàng.

Cung cấp một trình soạn thảo trực quan cho phép nhà phát triển có thể cấu hình tùy chỉnh các chức năng bằng cách sử dụng các (node) từ bất kỳ trình duyệt nào trên máy tính.

Mỗi ứng dụng Node-RED bao gồm các node được liên kết với nhau dưới dạng input, operation và output.

A close-up of a band aid

Description automatically generated

Hình 2. Node trong Node – RED cơ bản

Node-RED ban đầu được IBM phát triển để sử dụng nội bộ nhưng đã được mở mã nguồn vào năm 2016.

Kể từ đó, ngành công nghiệp tự động hóa đã áp dụng công nghệ này như một cách dễ dàng để tạo ra các ứng dụng IoT công nghiệp nhằm thu thập, xử lý và chia sẻ dữ liệu qua Internet.

2.8.2 Node-RED trong IoT

Node-RED giúp phát triển các ứng dụng IoT công nghiệp dễ dàng theo hai cách:

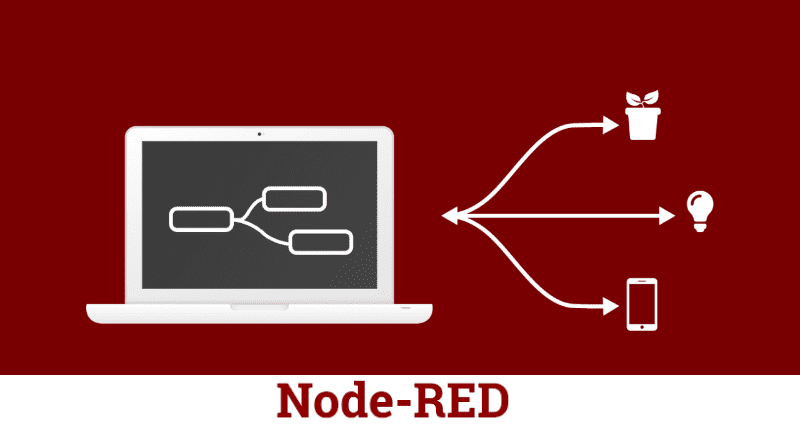
- Đầu tiên, Node-RED được tích hợp sẵn một số lượng lớn các nút được dựng sẵn. Là một nhà phát triển, bạn có thể sử dụng các nút này trong ứng dụng của mình để dễ dàng thực hiện các tác vụ phức tạp như gửi dữ liệu bằng giao thức MQTT , giao thức Modbus/TCP hoặc qua email.

Thậm chí còn có các nút cho phép bạn xuất bản dữ liệu lên các dịch vụ trực tuyến như Dropbox và Google Drive.

Hiện tại, có hơn hai nghìn nút có sẵn để bạn sử dụng trong các ứng dụng IoT công nghiệp của mình và nhờ vào nỗ lực không ngừng nghỉ của cộng đồng nguồn mở, con số đó đang không ngừng tăng lên.

- Thứ hai, Node-RED là một công cụ phát triển đồ họa. Mỗi nút trong ứng dụng được đặt trên một canvas và kết nối với các nút khác. Mỗi nút trong ứng dụng thực hiện một tác vụ cụ thể như thu thập dữ liệu, xử lý dữ liệu hoặc gửi dữ liệu.

Hai cách trên làm cho Node-RED trở thành một công cụ mạnh mẽ để phát triển các ứng dụng IoT, khiến nó có thể là một trong số các công cụ chính trong số các công cụ dành cho phát triển IoT



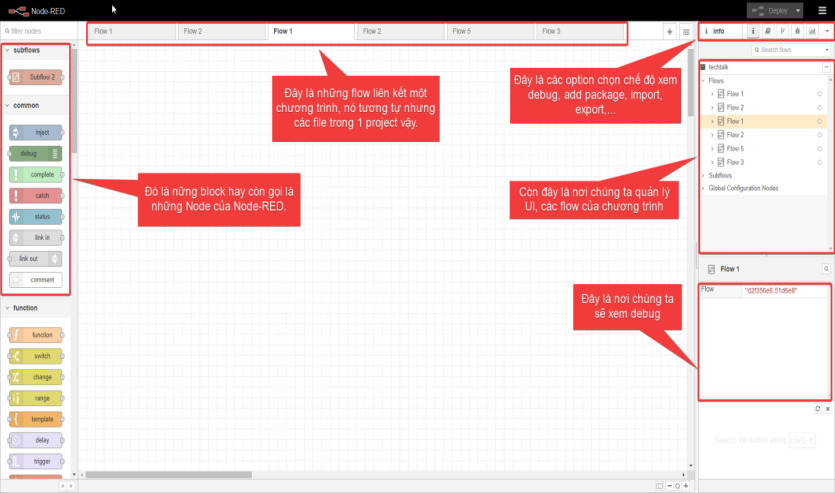
Hình 2. Node - RED và IoT

2.8.3 Tính năng

Được xây dựng trên Node.js, hỗ trợ môi trường thời gian chạy nhẹ.

Nhanh chóng trong việc phát triển các ứng dụng IoT.

Lấy dữ liệu từ các cảm biến và thiết bị. Phù hợp với hầu hết các thiết bị, được sử dụng rộng rãi trên Raspberry, Arduino, các thiết bị Android.



Hình 2. Giao diện Node - RED

Giao diện lập trình gồm 3 thành phần:

* Bên trái: tập hợp các nút có sẵn để xây dựng ứng dụng.
* Trung tâm: nơi thực hiện ứng dụng, kết nối các nút.
* Bên phải: thuộc tính và cài đặt cấu hình cho các nút.

2.9 MySQL

2.9.1 Giới thiệu

MySQL là hệ quản trị cơ sở dữ liệu tự do nguồn mở phổ biến nhất thế giới và được các nhà phát triển rất ưa chuộng trong quá trình phát triển ứng dụng. Vì MySQL là hệ quản trị cơ sở dữ liệu tốc độ cao, ổn định và dễ sử dụng, có tính khả chuyển, hoạt động trên nhiều hệ điều hành cung cấp một hệ thống lớn các hàm tiện ích rất mạnh. Với tốc độ và tính bảo mật cao, MySQL rất thích hợp cho các ứng dụng có truy cập CSDL trên internet. Người dùng có thể tải về MySQL miễn phí từ trang chủ. MySQL có nhiều phiên bản cho các hệ điều hành khác nhau: phiên bản Win32 cho các hệ điều hành dòng Windows, Linux, Mac OS X, Unix, FreeBSD, NetBSD, Novell NetWare, SGI Irix, Solaris, SunOS,.. MySQL là một trong những ví dụ rất cơ bản về Hệ Quản trị Cơ sở dữ liệu quan hệ sử dụng Ngôn ngữ truy vấn có cấu trúc (SQL). MySQL được sử dụng cho việc bổ trợ NodeJs, PHP, Perl, và nhiều ngôn ngữ khác, làm nơi lưu trữ những thông tin trên các trang web viết bằng NodeJs, PHP hay Perl,...



Hình 2. Cơ sở dữ liệu MySQL

2.9.2 Các tính năng của MySQL

Tạo và quản lý bảng: MySQL cho phép tạo ra các bảng để lưu trữ dữ liệu. Mỗi bảng chứa các cột và hàng, trong đó mỗi cột đại diện cho một thuộc tính và mỗi hàng đại diện cho một bản ghi trong cơ sở dữ liệu. Người dùng có thể định nghĩa các ràng buộc, quan hệ và chỉ mục trong bảng để xác định sự liên quan giữa các bảng.

Gửi yêu cầu SQL: Người dùng hoặc Client có thể gửi các yêu cầu SQL đặc biệt tới MySQL. Yêu cầu này có thể bao gồm các câu truy vấn SELECT, INSERT, UPDATE, DELETE và nhiều loại câu lệnh khác.

Xử lý yêu cầu: MySQL Server nhận và xử lý các yêu cầu SQL từ Client. Công cụ sẽ kiểm tra cú pháp và tính hợp lệ của yêu cầu, sau đó thực hiện các thao tác tương ứng trên cơ sở dữ liệu. MySQL sử dụng bộ trình diễn truy vấn (query optimizer) để tối ưu hóa hiệu suất truy vấn.

Trả kết quả: Sau khi xử lý yêu cầu, MySQL Server sẽ trả về kết quả cho Client. Kết quả có thể là dữ liệu truy vấn từ cơ sở dữ liệu hoặc thông báo về việc cập nhật, thêm, xóa dữ liệu.

2.9.3 Ưu điểm của MySQL

Được tối ưu hóa để đạt hiệu suất cao, cho phép thực thi truy vấn nhanh chóng và xử lý số lượng lớn dữ liệu một cách hiệu quả.

Có khả năng mở rộng linh hoạt, cho phép mở rộng cơ sở dữ liệu dựa trên nhu cầu. MySQL hỗ trợ cơ chế nhân rộng (scalability) để thêm các máy chủ hoặc node để xử lý tải lớn và đảm bảo sự ổn định của hệ thống.

Cung cấp nhiều tính năng phong phú để làm việc với cơ sở dữ liệu. Công cụ hỗ trợ các chuẩn SQL phổ biến và thêm những chức năng như khóa ngoại, xử lý giao dịch, truy vấn phức tạp, truy vấn văn bản đầy đủ,... Điều này giúp lập trình viên có thể thực hiện các hoạt động phức tạp trên cơ sở dữ liệu một cách thuận tiện.

Cung cấp các tính năng bảo mật mạnh mẽ để bảo vệ dữ liệu. Công cụ hỗ trợ xác thực người dùng, quản lý quyền truy cập, mã hóa dữ liệu và các biện pháp bảo mật khác.

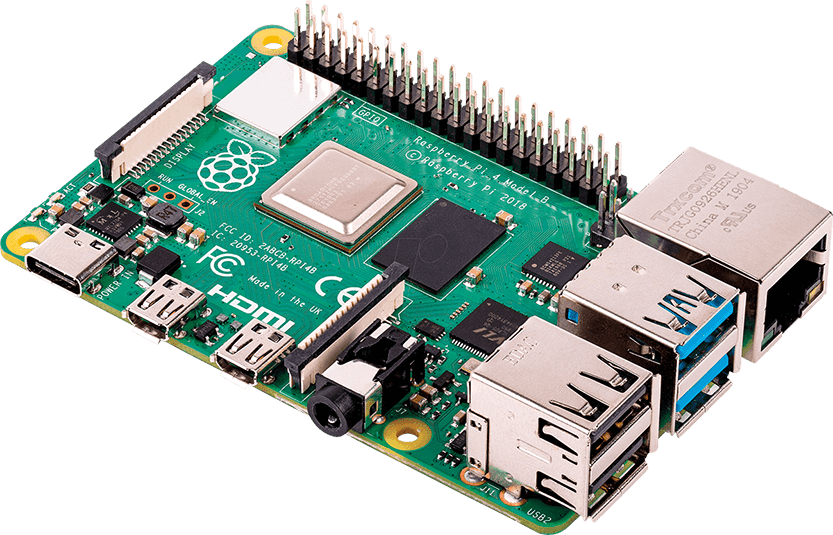
2.10 Raspberry Pi 4

2.10.1 Giới thiệu

Raspberry Pi 4 là một máy tính bảng đơn (Single Board Computer - SBC) được phát triển bởi Raspberry Pi Foundation, ra mắt vào tháng 6/2019. Nó tích hợp bộ vi xử lý Broadcom BCM2711 với bốn nhân Cortex-A72, bộ nhớ RAM tùy chọn từ 2GB đến 8GB, hỗ trợ Wi-Fi, Bluetooth, và nhiều giao diện ngoại vi như GPIO, USB, HDMI, và Ethernet. Với hiệu năng vượt trội so với các phiên bản trước, Raspberry Pi 4 là nền tảng lý tưởng cho các ứng dụng giáo dục, phát triển dự án IoT, máy chủ cá nhân, và giải trí.

Raspberry Pi 4 cung cấp khả năng kết nối mạng qua Wi-Fi 802.11ac băng tần kép và Bluetooth 5.0, cho phép tương tác với các thiết bị không dây và tích hợp vào các hệ thống IoT. Hỗ trợ nhiều giao thức như TCP/IP, HTTP, và MQTT, thiết bị này phù hợp cho các ứng dụng giám sát, điều khiển từ xa, và thu thập dữ liệu. Với cộng đồng người dùng lớn và giá thành hợp lý (từ 35 USD), Raspberry Pi 4 được sử dụng rộng rãi trong học tập, nghiên cứu, và các dự án công nghệ sáng tạo.

Các ứng dụng phổ biến bao gồm: học lập trình (Python, Scratch), xây dựng hệ thống nhà thông minh, phát triển máy chủ web, media center (phát video 4K), và máy chơi game cổ điển. Raspberry Pi 4 có các phiên bản khác nhau dựa trên dung lượng RAM, đáp ứng nhu cầu đa dạng từ các dự án đơn giản đến phức tạp.



Hình 2. Raspberry Pi 4

2.10.2 Các loại Raspberry Pi 4

Raspberry Pi 4 có các phiên bản khác nhau, chủ yếu dựa trên dung lượng RAM và một số cải tiến nhỏ:

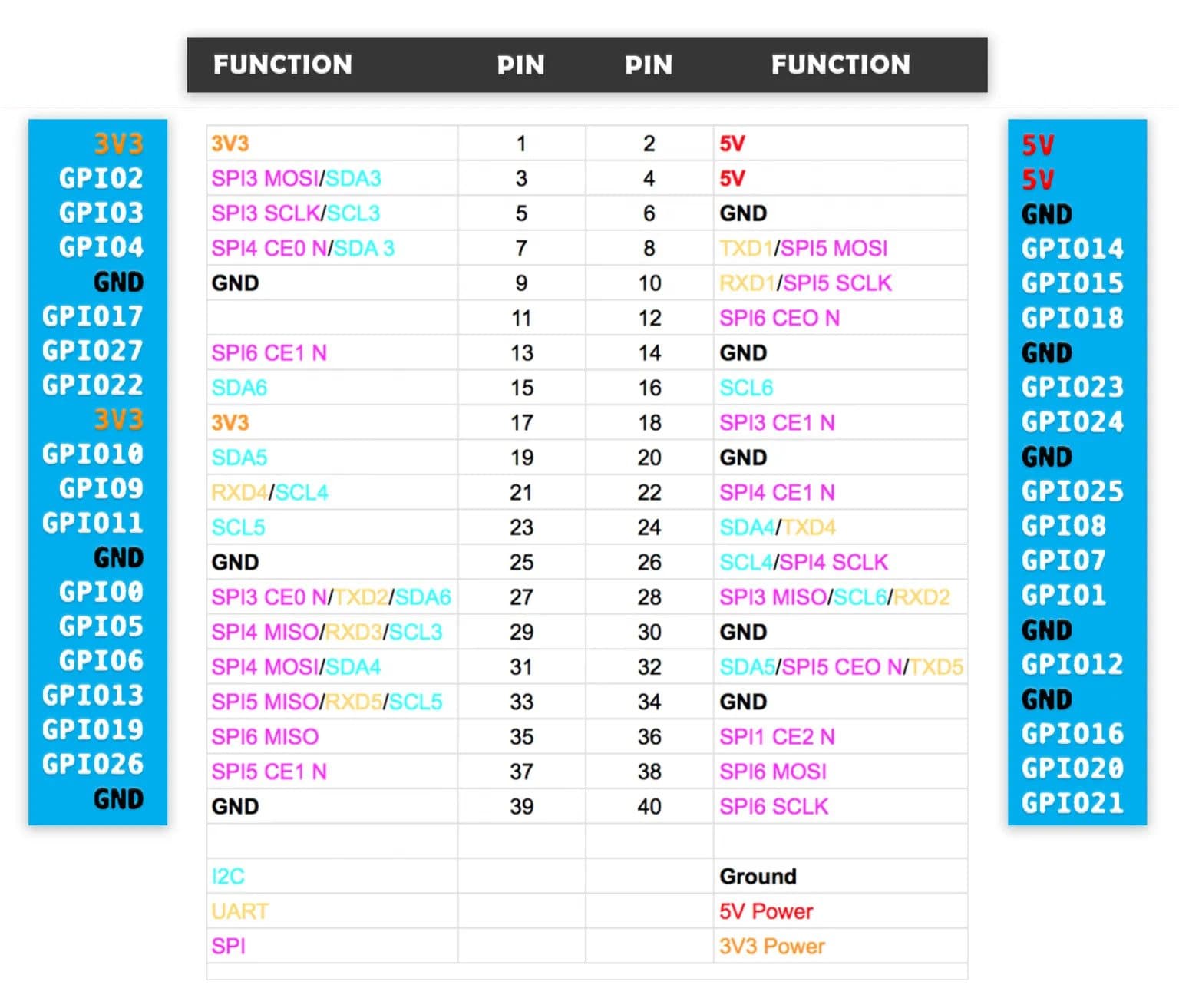
* Raspberry Pi 4 Model B (2GB): Phiên bản cơ bản với 2GB RAM LPDDR4, phù hợp cho các ứng dụng nhẹ như học lập trình, dự án IoT đơn giản, hoặc máy chủ nhỏ. Raspberry Pi 4 Model B (4GB): Phiên bản phổ biến nhất, với 4GB RAM, lý tưởng cho đa nhiệm, trình duyệt web, và các dự án yêu cầu hiệu năng trung bình như media center hoặc retro gaming.
* Raspberry Pi 4 Model B (8GB): Phiên bản cao cấp với 8GB RAM, hỗ trợ các ứng dụng đòi hỏi nhiều tài nguyên như máy tính để bàn, máy chủ web lớn, hoặc xử lý dữ liệu phức tạp.
* Raspberry Pi 4 (phiên bản cải tiến): Một số bo mạch được cập nhật từ năm 2020 với CPU xung nhịp 1.8GHz (thay vì 1.5GHz) và cải thiện khả năng quản lý nhiệt, nhưng vẫn giữ nguyên các thông số khác.

2.10.3 Tính năng

Raspberry Pi 4 được trang bị các tính năng kỹ thuật nổi bật, đáp ứng nhu cầu phát triển đa dạng:

* Wi-Fi: Hỗ trợ Wi-Fi 802.11ac băng tần kép (2.4GHz và 5GHz), đảm bảo kết nối không dây tốc độ cao và ổn định.
* Bluetooth: Tích hợp Bluetooth 5.0, hỗ trợ kết nối với các thiết bị như bàn phím, chuột, loa, hoặc cảm biến không dây.
* Giao diện ngoại vi đa dạng: Cung cấp GPIO (40 chân), USB (2 x USB 3.0, 2 x USB 2.0), micro-HDMI, Ethernet, và cổng âm thanh/video analog, cho phép kết nối với nhiều thiết bị và cảm biến.
* Bộ nhớ flash: Sử dụng thẻ microSD làm bộ nhớ chính (khuyến nghị Class 10, từ 16GB trở lên) để lưu trữ hệ điều hành và dữ liệu.
* Bộ nhớ RAM: Tùy chọn 2GB, 4GB, hoặc 8GB LPDDR4-3200, hỗ trợ đa nhiệm và các ứng dụng yêu cầu tài nguyên lớn.
* Hỗ trợ giao thức mạng: Tích hợp TCP/IP, HTTP, MQTT, và các giao thức IoT khác, giúp dễ dàng tích hợp vào các hệ thống mạng.
* Hỗ trợ video 4K: Hai cổng micro-HDMI cho phép xuất video 4K 60Hz, phù hợp cho media center hoặc hiển thị dữ liệu chất lượng cao.
* Quản lý năng lượng: Sử dụng nguồn USB-C 5V/3A, với khả năng tối ưu tiêu thụ năng lượng trong các ứng dụng nhẹ.
* Hỗ trợ OTA (Over-The-Air): Cho phép cập nhật hệ điều hành và phần mềm từ xa thông qua Wi-Fi, đảm bảo thiết bị luôn được nâng cấp dễ dàng.

2.10.4 Các chân input-output của Raspberry Pi 4



Hình 2. GPIO Raspberry Pi 4

Raspberry Pi 4 có 40 chân, bao gồm:

* 26 chân GPIO đa năng: Có thể cấu hình làm đầu vào (input) hoặc đầu ra (output), hỗ trợ các giao thức như I2C, SPI, UART, và PWM.
* Nguồn điện:
  + 2 chân 5V (pin 2, 4): Cung cấp nguồn 5V (tùy thuộc vào nguồn USB-C).
  + 2 chân 3.3V (pin 1, 17): Cung cấp nguồn 3.3V.
  + 8 chân GND (pin 6, 9, 14, 20, 25, 30, 34, 39): Đất (Ground).
* I2C:
  + 2 kênh I2C: GPIO 2 (SDA) và GPIO 3 (SCL) cho I2C0, hỗ trợ giao tiếp với cảm biến, màn hình LCD, hoặc các thiết bị I2C khác.
  + Có thể cấu hình các chân GPIO khác làm I2C nếu cần.
* SPI:
  + 2 kênh SPI: SPI0 (GPIO 9-MISO, GPIO 10-MOSI, GPIO 11-SCLK) và SPI1 (GPIO 19-MISO, GPIO 20-MOSI, GPIO 21-SCLK).
  + Hỗ trợ kết nối với các thiết bị như màn hình SPI hoặc cảm biến.
* UART:
  + 1 giao diện UART chính: GPIO 14 (TXD) và GPIO 15 (RXD).
  + Hỗ trợ truyền thông nối tiếp với các thiết bị như GPS, mô-đun GSM, hoặc Arduino.
* PWM:
  + 2 kênh PWM phần cứng: GPIO 12, GPIO 13, GPIO 18, GPIO 19.
  + Hỗ trợ điều khiển động cơ, đèn LED, hoặc các thiết bị yêu cầu tín hiệu PWM.
* I2S:
  + Hỗ trợ giao tiếp âm thanh qua I2S, thường sử dụng các chân GPIO 18, 19, 20, 21.
  + Phù hợp cho các ứng dụng âm thanh chất lượng cao.
* Chân đặc biệt:
  + GPIO 0 và GPIO 1: Hỗ trợ ID\_SD và ID\_SC, dùng để nhận diện các mô-đun HAT.
  + GPIO 5-11: Có thể được sử dụng cho các chức năng đặc biệt như SPI hoặc UART phụ.

Cảnh báo khi sử dụng các chân GPIO:

* Mức điện áp: Các chân GPIO hoạt động ở mức 3.3V. Kết nối với tín hiệu 5V có thể làm hỏng bo mạch.
* Dòng điện tối đa: Mỗi chân GPIO chỉ chịu được dòng tối đa 16mA, tổng dòng qua tất cả các chân không vượt quá 50mA.
* Chân khởi động (Boot Pins): Một số chân như GPIO 0, GPIO 1 có vai trò đặc biệt trong quá trình khởi động. Không nên kết nối thiết bị gây nhiễu tín hiệu trên các chân này khi khởi động.
* Chân PWM và ADC: Raspberry Pi 4 không có ADC (Analog-to-Digital Converter) tích hợp. Nếu cần ADC, phải sử dụng mô-đun bên ngoài qua I2C hoặc SPI.
* Chân HIGH tại khởi động: Một số chân như GPIO 3 (SCL) và GPIO 2 (SDA) có thể ở trạng thái HIGH khi khởi động do tích hợp bộ kháng kéo (pull-up). Cần lưu ý khi thiết kế mạch.

Cảnh báo về dòng điện:

* Nguồn USB-C: Yêu cầu bộ nguồn 5V/3A chất lượng cao để đảm bảo hoạt động ổn định, đặc biệt khi sử dụng nhiều thiết bị ngoại vi.
* Tiêu thụ dòng qua GPIO: Không vượt quá 50mA tổng dòng qua các chân GPIO để tránh hỏng bo mạch.
* Tản nhiệt: Khi sử dụng ở cường độ cao (ví dụ: chạy video 4K hoặc xử lý đa nhiệm), Raspberry Pi 4 có thể nóng. Nên sử dụng quạt hoặc tản nhiệt để duy trì hiệu năng.

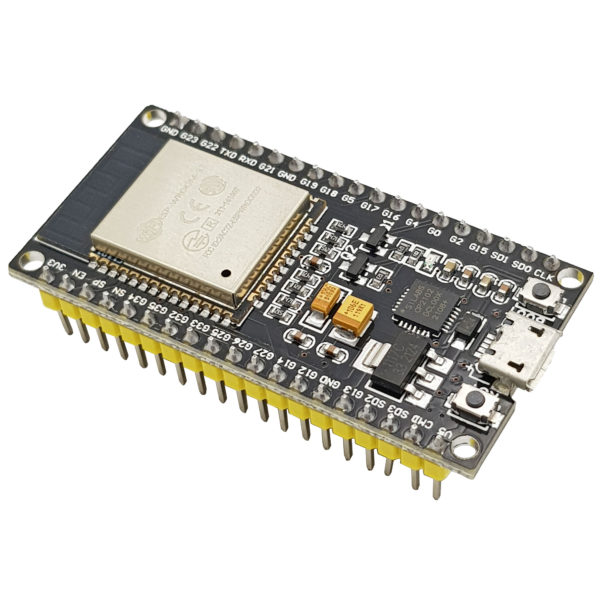
2.11 ESP32

2.11.1 Giới thiệu

ESP32 kết hợp một bộ vi xử lý hai nhân Tensilica LX6, bộ nhớ flash tích hợp, Wi-Fi, Bluetooth và nhiều giao diện ngoại vi khác nhau như GPIO, UART, I2C và SPI. Điều này khiến nó trở thành một nền tảng phát triển mạnh mẽ cho việc xây dựng các ứng dụng IoT đa dạng, từ các dự án nhỏ đến các hệ thống phức tạp.

ESP32 cung cấp khả năng kết nối mạng với Wi-Fi và Bluetooth, cho phép thiết bị được kết nối với các mạng không dây và tương tác với các thiết bị khác. Nó cũng hỗ trợ nhiều giao thức và tiện ích như TCP/IP, HTTP, MQTT và nhiều giao thức khác, giúp việc giao tiếp và truyền thông dễ dàng trong các ứng dụng IoT.

ESP32 được sử dụng phổ biến trong việc phát triển các ứng dụng như hệ thống giám sát, điều khiển thiết bị từ xa, đo lường và cảm biến, thiết bị định vị và nhiều ứng dụng IoT khác. Nó có sẵn trong các phiên bản khác nhau với các tính năng và giao diện ngoại vi khác nhau, giúp phù hợp với nhu cầu cụ thể của từng dự án.



Hình 2. ESP32

2.11.2 Các loại esp32

ESP32-WROOM-32: Đây là phiên bản gốc của ESP32, có kích thước nhỏ gọn và đầy đủ các giao diện ngoại vi như GPIO, UART, I2C, SPI, Wi-Fi và Bluetooth. Nó là phiên bản phổ biến và thích hợp cho các dự án IoT đa dạng.

ESP32-WROVER: Phiên bản này tương tự như ESP32-WROOM-32, nhưng có bộ nhớ RAM mở rộng hơn để hỗ trợ các ứng dụng đòi hỏi nhiều bộ nhớ hơn.

ESP32-WROVER-B: Đây là phiên bản tương tự như ESP32-WROVER, nhưng có bộ nhớ flash mở rộng hơn để lưu trữ các ứng dụng phức tạp hơn.

ESP32-PICO-D4: Đây là phiên bản nhỏ gọn của ESP32 với kích thước nhỏ hơn và tích hợp bộ nhớ flash và các giao diện ngoại vi như Wi-Fi, Bluetooth, GPIO, UART, I2C và SPI. Nó thích hợp cho các ứng dụng có yêu cầu về không gian vàng.

ESP32-S2: Phiên bản này chỉ hỗ trợ Wi-Fi và được tối ưu cho các ứng dụng tiết kiệm năng lượng, không có Bluetooth.

ESP32-C3: Phiên bản tiết kiệm chi phí với bộ xử lý RISC-V, hỗ trợ Wi-Fi và Bluetooth LE

ESP32-CAM: Đây là phiên bản ESP32 được tích hợp camera. Nó cho phép bạn thực hiện các ứng dụng liên quan đến hình ảnh và video như nhận dạng khuôn mặt, giám sát, và các dự án IoT có tính năng hình ảnh.

2.11.3 Tính năng

Wi-Fi: ESP32 hỗ trợ Wi-Fi 802.11 b/g/n, cho phép kết nối với các mạng Wi-Fi và truyền dữ liệu qua mạng không dây.

Bluetooth: ESP32 tích hợp Bluetooth 4.2 và Bluetooth Low Energy (BLE), cho phép kết nối và truyền dữ liệu với các thiết bị Bluetooth khác.

Giao diện ngoại vi đa dạng: ESP32 cung cấp nhiều giao diện ngoại vi như GPIO, UART, I2C, SPI, ADC, DAC, PWM và nhiều giao diện khác, cho phép kết nối và tương tác với các thiết bị và cảm biến khác.

Bộ nhớ flash tích hợp: ESP32 tích hợp bộ nhớ flash để lưu trữ mã chương trình và dữ liệu, giúp việc phát triển ứng dụng trở nên dễ dàng.

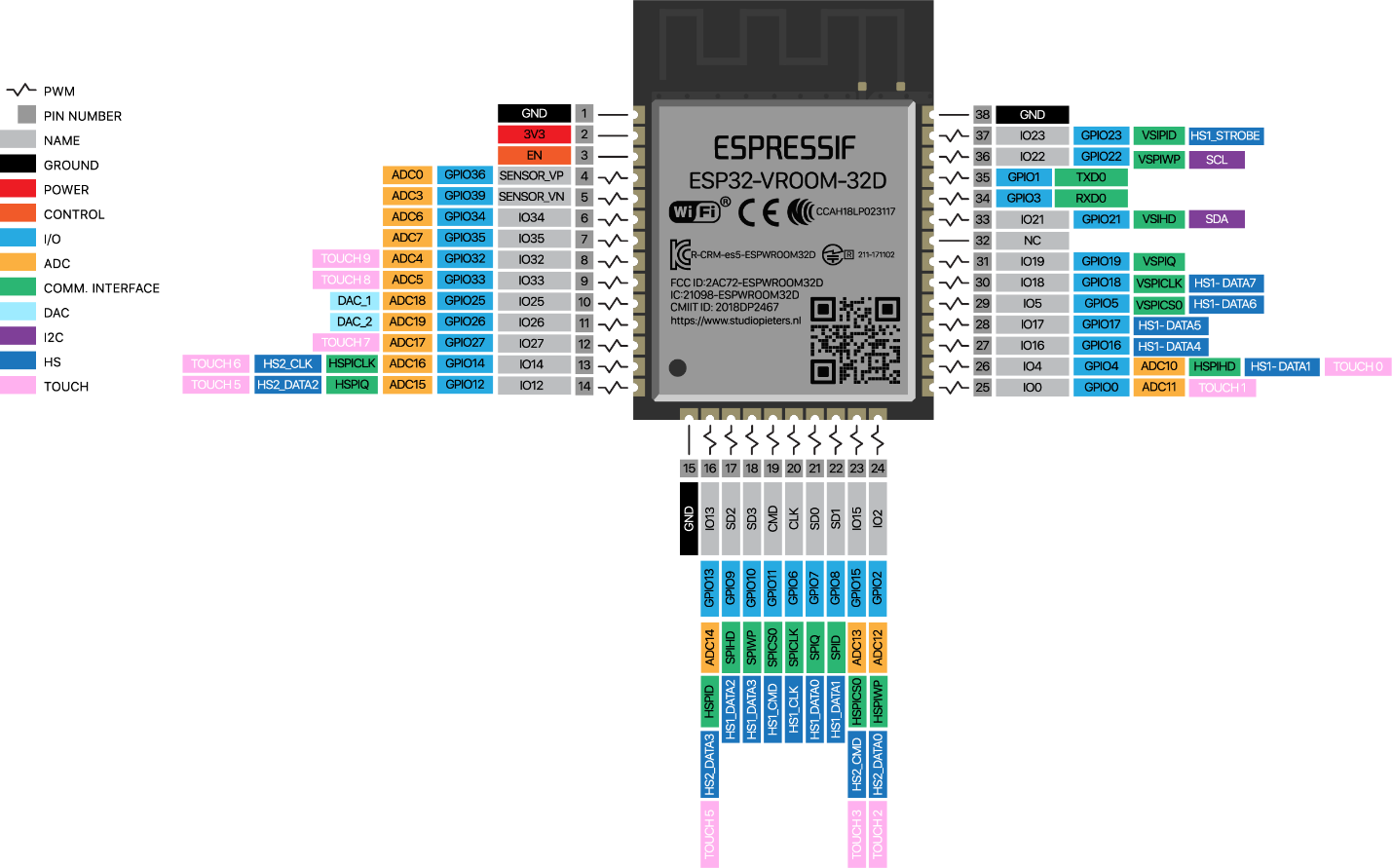
Bộ nhớ RAM: ESP32 có bộ nhớ RAM tích hợp, cho phép xử lý dữ liệu lớn và hỗ trợ các ứng dụng đòi hỏi nhiều bộ nhớ hơn.

Hỗ trợ giao thức mạng: ESP32 hỗ trợ nhiều giao thức mạng như TCP/IP, HTTP, MQTT, CoAP và nhiều giao thức khác, giúp việc kết nối và giao tiếp trong các mạng IoT dễ dàng.

Tiết kiệm năng lượng: ESP32 có các chế độ hạ thấp tiêu thụ năng lượng, cho phép tiết kiệm pin và phù hợp với các ứng dụng yêu cầu tuổi thọ pin dài.

Hỗ trợ OTA (Over-The-Air) cập nhật: ESP32 cho phép cập nhật phần mềm từ xa thông qua kết nối Wi-Fi hoặc Bluetooth, giúp nâng cấp và sửa lỗi ứng dụng một cách thuận tiện.

2.11.4 Các chân input-output của ESP32 WROOM



Hình 2. Chip ESP32-VROOM-32D

A computer chip with many different colored buttons

Description automatically generated

Hình 2. Module ESP32-WROOM-32D

Các tính năng của ESP32:

* 18 ADC Channels: ESP32 có 18 kênh ADC (Analog-to-Digital Converter) với độ phân giải 12 bit, cho phép đọc giá trị từ 0 đến 4095, ứng với điện áp từ 0V đến 3.3V.
* 3 SPI Interfaces: Cho phép kết nối với các thiết bị ngoại vi qua giao thức SPI (Serial Peripheral Interface).
* 3 UART Interfaces: Hỗ trợ 3 giao diện UART cho việc truyền thông nối tiếp (Serial).
* 2 I2C Interfaces: Hỗ trợ 2 giao diện I2C cho việc giao tiếp với các thiết bị như cảm biến hoặc màn hình LCD.
* 16 PWM Channels: ESP32 hỗ trợ 16 kênh PWM (Pulse Width Modulation) để điều khiển động cơ, đèn LED, v.v.
* 2 DAC (Digital-to-Analog Converters): Cung cấp khả năng chuyển đổi tín hiệu số thành tín hiệu analog.
* 2 I2S Interfaces: Hỗ trợ giao tiếp âm thanh qua giao thức I2S (Inter-IC Sound).
* 10 Capacitive Sensing GPIOs: Các chân cảm ứng điện dung có thể dùng làm cảm biến chạm, thay thế các nút bấm vật lý.

Các chân GPIO của ESP32:

* Chân đầu vào (Input Only): GPIO 34 đến 39 chỉ có thể được sử dụng làm đầu vào (không thể xuất tín hiệu). Các chân này không có bộ kháng kéo (pull-up hoặc pull-down) và không hỗ trợ tính năng đầu ra.
* Chân dành riêng cho Flash: GPIO 6 đến 11 gắn liền với bộ nhớ flash SPI tích hợp trên ESP-WROOM-32, do đó không thể sử dụng làm chân đầu vào hoặc đầu ra.
* Chân ADC (Analog-to-Digital Converter): Các chân ADC được phân bổ cho các GPIO khác nhau, nhưng có sự khác biệt giữa ADC1 và ADC2. ADC2 không thể sử dụng khi ESP32 đang sử dụng Wi-Fi, vì vậy trong trường hợp đó, nên sử dụng ADC1 để tránh gặp sự cố.
* Chân DAC (Digital-to-Analog Converter): GPIO 25 và GPIO 26 có thể sử dụng để xuất tín hiệu analog.
* Chân Capacitive Touch: Có 10 chân GPIO hỗ trợ cảm biến chạm, có thể dùng thay thế nút bấm vật lý hoặc để đánh thức ESP32 khỏi chế độ deep sleep.

Chân PWM (Pulse Width Modulation):

* PWM Pin: ESP32 hỗ trợ PWM trên hầu hết các chân có khả năng xuất tín hiệu (trừ GPIO 34 đến 39). PWM có thể được sử dụng để điều khiển động cơ, đèn LED, v.v.
* Cài đặt PWM: Để sử dụng PWM, bạn cần xác định tần số, độ rộng xung (duty cycle), và kênh PWM cũng như chân GPIO xuất tín hiệu.

I2C và SPI:

* I2C: ESP32 có 2 kênh I2C với các chân mặc định là GPIO 21 (SDA) và GPIO 22 (SCL), nhưng bạn có thể cấu hình lại bất kỳ chân GPIO nào làm SDA hoặc SCL.
* SPI: ESP32 hỗ trợ 2 bus SPI (VSPI và HSPI) với các chân mặc định là GPIO 23 (MOSI), GPIO 19 (MISO), GPIO 18 (CLK), và GPIO 5 (CS). Có thể chọn bus SPI phù hợp và cấu hình các chân lại theo nhu cầu của dự án.

Các chân Strapping (Khởi động):

* Strapping Pins: Một số chân có chức năng đặc biệt trong quá trình khởi động, ví dụ GPIO 0, GPIO 2, GPIO 12 và GPIO 15 ảnh hưởng đến cách ESP32 vào chế độ boot hoặc chế độ flash. Cần lưu ý không kết nối các thiết bị làm thay đổi trạng thái của những chân này trong quá trình khởi động.

Cảnh báo khi sử dụng các chân GPIO:

* Pins HIGH at Boot: Một số chân (ví dụ GPIO 1, 3, 5, 14, 15) có thể tự động trở thành HIGH khi khởi động. Điều này có thể gây ra sự cố nếu bạn đang sử dụng chúng để xuất tín hiệu và không chú ý đến trạng thái của chúng trong quá trình khởi động.
* Chân có chức năng đặc biệt: Ví dụ, GPIO 12 phải là LOW trong quá trình khởi động và GPIO 5 phải là HIGH.

Cảnh báo về dòng điện:

* Dòng điện tối đa: Các chân GPIO có thể chịu đựng dòng điện tối đa là 40mA, vì vậy bạn không nên vượt quá giới hạn này khi kết nối các thiết bị ngoại vi.

2.12 DHT11

2.12.1 Giới thiệu

Cảm biến nhiệt độ độ ẩm DHT11 là một loại cảm biến kỹ thuật số được sử dụng để đo độ ẩm và nhiệt độ trong không khí. Nó được phát triển bởi công ty Aosong Electronics và sử dụng một cảm biến đo nhiệt độ và một cảm biến đo độ ẩm để cung cấp các giá trị đo chính xác. DHT11 có thiết kế nhỏ gọn, dễ sử dụng và chi phí thấp, nên nó được sử dụng phổ biến trong các ứng dụng điện tử như đo nhiệt độ và độ ẩm trong các thiết bị gia dụng, các hệ thống tự động hóa, các thiết bị đo lường và nhiều ứng dụng khác.

A blue and black electronic device

Description automatically generated

Hình 2. Module DHT11

2.12.2 Cấu tạo

VCC (cấp nguồn): Chân này được kết nối với nguồn 5V của ESP. GND (chân đất): Chân này được kết nối với đất của ESP. Data: Chân này được sử dụng để truyền dữ liệu từ cảm biến đến ESP. Nó được kết nối với một chân trên ESP để truyền dữ liệu kỹ thuật số.

A diagram of a device

Description automatically generated

Hình 2. Cấu tạo DHT11

2.12.3 Nguyên lý hoạt động

Diagram of a diagram of a electrode

Description automatically generated

Hình 2. Nguyên lý hoạt động DHT11

Cảm biến nhiệt độ và độ ẩm DHT11 là một cảm biến kỹ thuật số sử dụng để đo độ ẩm và nhiệt độ trong môi trường. Nguyên lý hoạt động của DHT11 dựa trên việc sử dụng một cặp điện cực nhạy cảm để đo độ ẩm trong không khí. Điện cực này thay đổi điện trở theo độ ẩm, và sau đó được đọc bởi một vi mạch điện tử để tính toán độ ẩm. Cảm biến cũng sử dụng một cặp điện cực khác để đo nhiệt độ. Khi nhiệt độ thay đổi, điện trở của điện cực cũng thay đổi, cho phép vi mạch tính toán nhiệt độ của môi trường.

Dữ liệu được truyền từ cảm biến DHT11 sang vi điều khiển thông qua giao tiếp 1-write, bao gồm một tín hiệu số và một chuỗi bit dữ liệu. Vi điều khiển sử dụng giao thức này để giải mã dữ liệu độ ẩm và nhiệt độ và sử dụng nó cho các ứng dụng đo nhiệt độ và độ ẩm trong môi trường.

2.13 GP2Y1010AU0F

2.13.1 Giới thiệu

Cảm biến bụi GP2Y1010AU0F của Sharp là một thiết bị quang học được thiết kế để đo nồng độ bụi mịn trong không khí, bao gồm bụi PM2.5 và PM10. Nhờ tính năng đo chính xác và giá thành hợp lý, cảm biến này được sử dụng rộng rãi trong các ứng dụng như giám sát chất lượng không khí, máy lọc không khí, thiết bị IoT và các hệ thống phân tích môi trường. Cảm biến hoạt động dựa trên nguyên lý tán xạ ánh sáng, cho phép phát hiện các hạt bụi mịn kích thước nhỏ tới 0.8 µm. GP2Y1010AU0F có ưu điểm nổi bật là kích thước nhỏ gọn, tiêu thụ điện năng thấp và dễ dàng tích hợp với các vi điều khiển như ESP32 hoặc Arduino, phù hợp với các hệ thống nhúng.



Hình 2. module cảm biến bụi mịn GP2Y1010AU0F

2.13.2 Cấu tạo

* LED phát ánh sáng hồng ngoại:
* Là nguồn phát ánh sáng hồng ngoại, được điều khiển bởi vi điều khiển thông qua tín hiệu xung.
* Đèn LED tạo ra tia sáng định hướng thông qua LED Optical Axis.
* Lens (Thấu kính):
* Có nhiệm vụ hội tụ hoặc định hướng ánh sáng hồng ngoại từ LED để tăng độ chính xác trong việc chiếu sáng bụi.
* Một cặp thấu kính được đặt phía trước LED và photodiode để tối ưu hóa hiệu suất.
* Photodiode (Detector):
* Là cảm biến ánh sáng được đặt bên phía đối diện của luồng bụi.
* Photodiode nhận ánh sáng tán xạ từ các hạt bụi khi chúng đi qua vùng đo.
* Dust Path (Luồng bụi):
* Là đường dẫn không khí nơi bụi đi qua để được đo.
* Không khí mang bụi được đưa vào vùng Detector Area, nơi ánh sáng hồng ngoại tương tác với bụi.
* Detector Area (Vùng phát hiện):
* Đây là vùng giao cắt giữa LED Optical Axis và PD Optical Axis, nơi ánh sáng từ LED bị tán xạ bởi các hạt bụi và được photodiode phát hiện.
* PD Optical Axis:
* Là trục quang học của photodiode, nơi ánh sáng phản xạ hoặc tán xạ được thu nhận.

A diagram of a light

Description automatically generated

Hình 2. Cấu tạo cảm biến bụi mịn GP2Y1010AU0F

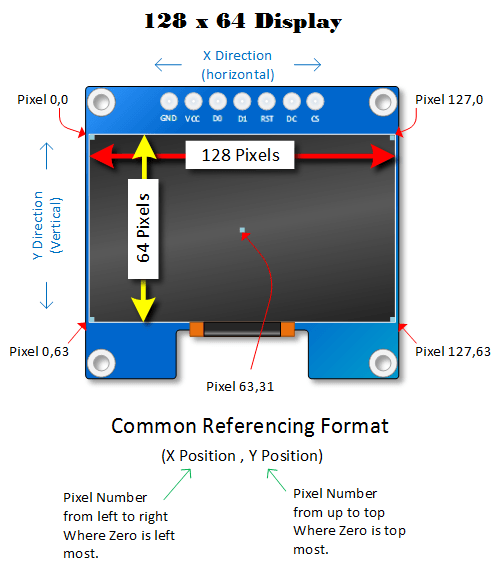
2.13.3 Nguyên lý hoạt động

* Không có bụi và/hoặc khói
* Trong trường hợp không khí sạch (không có bụi hoặc khói), ánh sáng từ LED phát ra sẽ không bị tán xạ.
* Chỉ một lượng rất nhỏ ánh sáng phản xạ ngược (ánh sáng phản xạ catoptric) đến photodetector.
* Kết quả: Điện áp đầu ra (Vo) rất thấp, tương ứng với nồng độ bụi cực thấp hoặc bằng 0.
* Có bụi và/hoặc khói
* Khi có bụi hoặc khói, các hạt bụi sẽ cản trở đường truyền ánh sáng từ LED, làm cho ánh sáng bị tán xạ.
* Ánh sáng tán xạ này được photodetector thu nhận.
* Kết quả: Tín hiệu điện áp đầu ra (Vo) tăng lên, giá trị điện áp này tỷ lệ với nồng độ bụi trong không khí.

2.14 OLED

2.14.1 Giới thiệu

Màn hình OLED 0.96 inch I2C là một loại màn hình hiển thị sử dụng công nghệ OLED (Organic Light Emitting Diodes), có kích thước 0.96 inch (khoảng 2.44 cm). Nó sử dụng giao thức truyền thông I2C (Inter-Integrated Circuit) để giao tiếp với các vi điều khiển như Arduino, Raspberry Pi và các mạch điện tử khác..



Hình 2. Màn hình OLED

2.14.2 Cấu tạo

* Mặt hiển thị:
* Kích thước: 0.96 inch, với độ phân giải thông thường là 128x64 pixel hoặc 128x32 pixel.
* Màu sắc: Phổ biến nhất là đơn sắc (trắng, xanh dương, hoặc vàng), nhưng cũng có loại đa sắc.
* Mạch điều khiển:
* Driver IC: Chip điều khiển phổ biến là SSD1306 hoặc SH1106, chịu trách nhiệm giao tiếp với vi điều khiển và điều khiển các điểm ảnh trên màn hình.
* Điện áp hoạt động: Thường là 3.3V hoặc 5V.
* Giao tiếp I2C:
* SDA (Serial Data): Dây truyền dữ liệu.
* SCL (Serial Clock): Dây đồng hồ để đồng bộ dữ liệu.
* Địa chỉ I2C: Mặc định là 0x3C hoặc 0x3D, có thể thay đổi tùy màn hình.
* Kết nối:
* VCC: Nguồn điện (3.3V hoặc 5V).
* GND: Nối đất.
* SCL: Clock I2C.
* SDA: Data I2C.

2.14.3 Nguyên lý hoạt động

Màn hình OLED 0.96 inch hoạt động dựa trên công nghệ phát sáng của các điểm ảnh được làm từ vật liệu hữu cơ. Khi có dòng điện chạy qua, các electron và lỗ trống tái hợp tại lớp phát sáng, tạo ra ánh sáng mà không cần đèn nền. Điều này không chỉ giúp tiết kiệm năng lượng mà còn mang lại độ tương phản cao, cho màu đen sâu và hình ảnh rõ nét.

Khi sử dụng với vi điều khiển như ESP32, màn hình giao tiếp qua giao thức I2C với hai dây tín hiệu chính: SCL (đồng hồ) và SDA (dữ liệu). Vi điều khiển gửi lệnh và dữ liệu đến driver IC, phổ biến là SSD1306, để điều khiển màn hình.

SSD1306 có một bộ nhớ hiển thị gọi là GDDRAM, nơi lưu trữ dữ liệu về trạng thái của từng pixel trên màn hình. SSD1306 quét dữ liệu từ GDDRAM và điều khiển từng điểm ảnh bật/tắt hoặc điều chỉnh độ sáng, tạo ra các ký tự, đồ họa hoặc hình ảnh theo yêu cầu. Màn hình OLED thường có độ phân giải 128x64 hoặc 128x32, với các pixel được tổ chức thành một ma trận, cho phép hiển thị nội dung chi tiết trên diện tích nhỏ.

2.15 Relay 2 chanel

2.15.1 giới thiệu

Module Relay 2 kênh là một module điện tử được sử dụng để điều khiển các thiết bị hoặc mạch điện thông qua tín hiệu điều khiển từ một nguồn bên ngoài như Arduino, Raspberry Pi… Nó cung cấp hai kênh relay độc lập, cho phép người dùng điều khiển hai thiết bị riêng biệt. Nó bao gồm 2 Relay, mỗi Relay có dòng điện là 10A và hoạt động ở điện áp 250VAC hoặc 30VDC.

A red and blue circuit board with black text

Description automatically generated

Hình 2. Relay 2 Chanel

2.15.2 cấu tạo

Relay: Có 2 relay để đóng/ngắt thiết bị điện, với các chân COM, NO, NC.

Cổng điều khiển (INPUT): Chân IN1, IN2 nhận tín hiệu từ vi điều khiển để bật/tắt relay.

Cổng nguồn (POWER): Chân VCC, GND cấp nguồn cho module.

Optocoupler: Cách ly tín hiệu điều khiển và mạch điện áp cao.

Transistor & Diode: Điều khiển cuộn dây relay và bảo vệ mạch.

LED trạng thái: Báo hiệu trạng thái relay.

2.15.3 Nguyên Lý hoạt động

Khi dòng điện chạy qua mạch thứ nhất (1), nó sẽ kích hoạt nam châm điện. Từ đó tạo ra từ trường để thu hút một tiếp điểm (màu đỏ). Sau đó sẽ kích hoạt mạch thứ hai (2). Khi tắt nguồn, một lò xo được lắp trước vào tiếp điểm sẽ có nhiệm vụ là kéo tiếp điểm trở lại vị trí ban đầu, tắt mạch thứ hai lại một lần nữa.

A diagram of a mechanical system

Description automatically generated with medium confidence

Hình 2. Nguyên lý hoạt động của relay

Bên dưới là một hình ảnh cho thấy cách một relay liên kết hai mạch với nhau. Ở bên trái, có mạch đầu vào được cung cấp bởi một công tắc hoặc loại cảm biến nào đó.

A diagram of a wire and a coil

Description automatically generated with medium confidence

Hình 2. Nguyên lý hoạt động của relay

Khi mạch này được kích hoạt thì nó cung cấp dòng điện cho một nam châm điện, sau đó sẽ kéo công tắc kim loại đóng lại. Từ đó kích hoạt mạch đầu ra thứ hai (ở phía bên phải).

Thứ nhất: Mạch đầu vào là vòng màu xanh sẽ bị tắt và không có dòng điện chạy qua. Nhưng khi một cái gì đó (có thể là cảm biến hoặc đóng công tắc) bật nó thù nó sẽ bật trở lại. Mạch đầu ra là vòng lặp màu đỏ cũng bị tắt.

Thứ hai: Khi một dòng điện nhỏ được chạy trong mạch đầu vào. Nó sẽ bắt đầu kích hoạt nam châm điện. Và từ đó tạo ra một từ trường xung quanh nó.

Thứ ba: Nam châm điện kéo thanh kim loại trong mạch đầu ra về phía nó. Đóng công tắc và cho phép các dòng điện lớn hơn nhiều chạy qua mạch đầu ra.

Thứ tư: Mạch đầu ra vận hành như một thiết bị có dòng điện cao.

1. THIẾT KẾ HỆ THỐNG

## 3.1 Tổng quan về hệ thống

**Giới thiệu chung**

Hệ thống IoT được thiết kế để thu thập, giám sát, phân tích, và dự báo chất lượng không khí trong phòng bệnh, đồng thời cung cấp khả năng điều khiển thiết bị từ xa thông qua giao thức MQTT. Hệ thống tích hợp các node slave (dựa trên ESP32) để thu thập dữ liệu cảm biến và Raspberry Pi 4 đóng vai trò trung tâm, thực hiện các chức năng MQTT Broker, web server, và chạy mô hình Prophet để dự báo chất lượng không khí.

**Mô tả hệ thống**

* Node slave: Node slave được trang bị các cảm biến môi trường như DHT11, GP2Y1010AU0F. Nhiệm vụ:
* Đọc dữ liệu cảm biến: Thu thập các thông số môi trường như nhiệt độ, độ ẩm, nồng độ bụi.
* Gửi dữ liệu lên MQTT broker: Dữ liệu được định dạng và gửi qua topic MQTT tới Raspberry Pi 4.
* Nhận lệnh điều khiển relay: Lắng nghe lệnh từ MQTT broker qua topic điều khiển và thực hiện đóng/mở relay tương ứng.
* Raspberry Pi 4 (Node trung tâm): đảm nhiệm vai trò điều khiển, hiển thị, và dự báo:
* MQTT Broker: Chạy trên Node-RED, quản lý giao tiếp giữa các node slave và các client (bao gồm web server)
* Web server: Cung cấp giao diện web để:
  + Hiển thị dữ liệu cảm biến thời gian thực (nhiệt độ, độ ẩm, PM2.5, AQI).
  + Hiển thị kết quả dự báo chất lượng không khí từ mô hình Prophet (ví dụ: dự báo nhiệt độ, độ ẩm, AQI, PM2.5 trong 24 giờ tới).
  + Cho phép điều khiển relay thủ công (bật/tắt máy lọc không khí, điều hòa).
* Dự báo chất lượng không khí: Chạy tập lệnh Python với mô hình Prophet để dự báo các thông số môi trường dựa trên dữ liệu lịch sử.
* Cơ sở dữ liệu: Lưu trữ dữ liệu cảm biến và kết quả dự báo vào MySQL để phân tích và trực quan hóa.
* MQTT broker: MQTT broker đóng vai trò trung gian trong hệ thống, đảm bảo các node có thể giao tiếp với nhau một cách hiệu quả:
* Xử lý dữ liệu: Nhận dữ liệu từ node slave và phân phối cho các client subscribe topic tương ứng (bao gồm node master).
* Điều khiển relay: Nhận lệnh từ node master và gửi lệnh đó đến node slave qua topic MQTT.

**Kết nối và giao tiếp**

Hệ thống hoạt động trên giao thức MQTT với các topic được định nghĩa như sau:

* Topic dữ liệu cảm biến: “node1/state/sensordata” Node slave gửi dữ liệu cảm biến.
* Topic điều khiển relay: “node1/control/mode” Node master gửi lệnh điều khiển relay, node slave lắng nghe và thực hiện.

**Sơ lược quy trình hoạt động**

* Node slave đọc dữ liệu từ cảm biến, gửi dữ liệu lên MQTT broker qua topic node1/state/sensordata.
* Raspberry Pi 4 (MQTT Broker):
* Lưu dữ liệu cảm biến vào cơ sở dữ liệu (MySQL).
* Phân phối dữ liệu đến web server để hiển thị thời gian thực.
* Raspberry Pi 4 chạy mô hình Prophet định kỳ (ví dụ: mỗi giờ), dự báo chất lượng không khí (AQI, PM2.5), lưu kết quả vào cơ sở dữ liệu và hiển thị trên web.
* Người dùng điều khiển relay thông qua: Giao diện web (gửi lệnh qua topic control/mode).

Khi người dùng nhấn nút vật lý hoặc điều khiển relay từ web server gửi lệnh qua topic node1/control/mode.

Node slave nhận lệnh từ topic node1/control/mode và thực hiện đóng/mở relay tương ứng.

## 3.2 Sơ đồ khối chức năng

**Sơ đồ khối hệ thống tổng quát**

* Node slave có chức năng gửi dữ liệu lên MQTT broker và nhận lệnh điều khiển chế độ từ MQTT broker.
* Raspberry Pi 4 (Node trung tâm):
* Chạy MQTT Broker để quản lý giao tiếp.
* Chạy web server để hiển thị dữ liệu và kết quả dự báo.
* Chạy mô hình Prophet để dự báo chất lượng không khí.
* Lưu trữ dữ liệu vào cơ sở dữ liệu.
* MQTT broker đóng vai trò trung gian trao đổi dữ liệu giữa node slave và node master

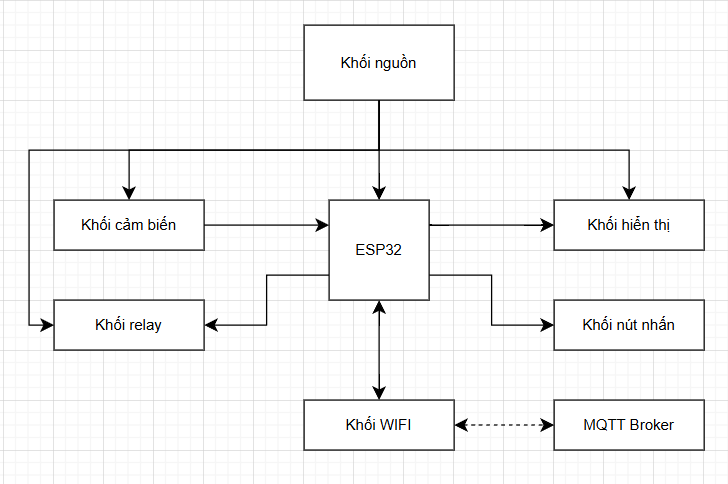
A diagram of a cloud

AI-generated content may be incorrect.

Hình 3. Sơ đồ tổng quát

**Sơ đồ khối node slave**

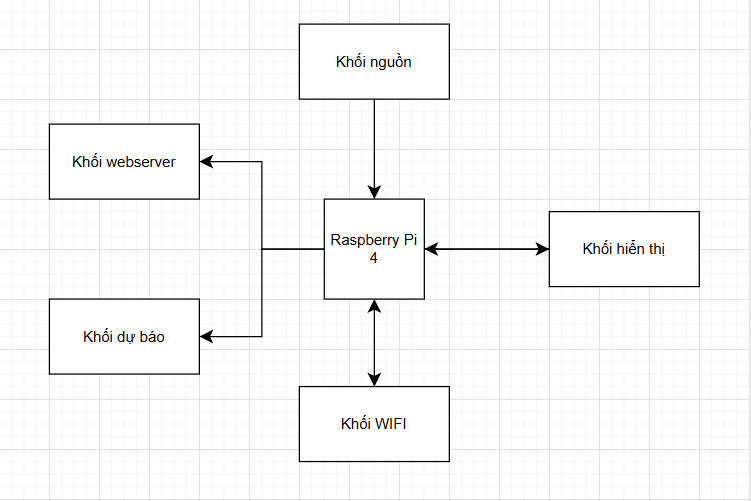
* Khối nguồn là khối cơ bản nhất nó cung cấp dòng nuôi ổn định cho toàn bộ linh kiện trong mạch.
* Khối ESP32: đây là khối xử lý trung tâm, có nhiệm vụ
* Đọc dữ liệu từ cảm biến.
* Giao tiếp với MQTT Broker qua WiFi.
* Điều khiển khối hiển thị và relay.
* Khối WIFI: Gửi/nhận dữ liệu qua MQTT Broker.
* Khối nút nhấn: gửi tín hiệu chế độ relay về khối xử lý trung tâm.
* Khối cảm biến:
* Cảm biến DHT11 để thu thập dữ liệu nhiệt độ độ ẩm.
* Cảm biến bụi GP2Y1010AU0F để thu thập dữ liệu nồng độ bụi mịn trong không khí.
* Khối hiển thị: hiển thị thông tin nhiệt độ, độ ẩm, bụi mịn, chế độ relay
* Khối relay: nhận lệnh từ khối xử lý trung tâm để bật tắt theo chế độ tương ứng



Hình 3. Sơ đồ khối node slave

**Sơ đồ khối node master**

* Khối nguồn: Cung cấp nguồn ổn định cho Raspberry Pi 4.
* Khối Raspberry Pi 4:
  + Chạy MQTT Broker (Node-RED) để quản lý giao tiếp.
  + Chạy web server để hiển thị dữ liệu và dự báo.
  + Chạy Prophet (Python) để dự báo chất lượng không khí.
  + Quản lý cơ sở dữ liệu (MySQL) để lưu trữ dữ liệu cảm biến và dự báo.
* Khối WiFi: Kết nối với MQTT Broker và giao tiếp với node slave.
* Khối hiển thị (tùy chọn): Hiển thị dữ liệu trên màn hình gắn trực tiếp vào Raspberry Pi 4 hoặc qua giao diện web.
* Khối Webserver: có các nhiệm vụ sau
  + Nhận dữ liệu từ MQTT Broker và cơ sở dữ liệu.
  + Hiển thị dữ liệu thời gian thực và kết quả dự báo từ Prophet.
  + Gửi lệnh điều khiển relay qua MQTT Broker.
* Khối dự báo Prophet:
  + Nhận dữ liệu lịch sử từ cơ sở dữ liệu.
  + Chạy mô hình Prophet để dự báo (AQI, PM2.5, v.v.).
  + Lưu kết quả dự báo vào cơ sở dữ liệu.



Hình 3. Sơ đồ khối node master

## 3.3 Lưu đồ thuật toán

a) Lưu đồ chương trình node slave

Lưu đồ chương trình chính Node slave

A diagram of a flowchart

AI-generated content may be incorrect.

Hình 3. Lưu đồ chương trình chính Node slave

Lưu đồ chương trình StartAPMode

A diagram of a computer program

AI-generated content may be incorrect.

Hình 3. Lưu đồ chương trình StartAPMode

Lưu đồ chương trình TaskControl

A diagram of a flowchart

AI-generated content may be incorrect.

Hình 3. Lưu đồ chương trình TaskControl

Lưu đồ chương trình TaskDisplay

A diagram of a software project

AI-generated content may be incorrect.

Hình 3. Lưu đồ chương trình TaskDisplay

Lưu đồ chương trình TaskMQTT

A diagram of a company

AI-generated content may be incorrect.

Hình 3. Lưu đồ chương trình TaskMQTT

Lưu đồ chương trình TaskMQTTSubcribe

A diagram of a flowchart

AI-generated content may be incorrect.

Hình 3. Lưu đồ chương trình TaskMQTTSubcribe

Lưu đồ chương trình TaskPrintData

A diagram of a flowchart

AI-generated content may be incorrect.

Hình 3. Lưu đồ chương trình TaskPrintData

Lưu đồ chương trình TaskReadSensor

A diagram of a process

AI-generated content may be incorrect.

Hình 3. Lưu đồ chương trình TaskReadSensor

Lưu đồ chương trình TaskResetTimeout

A diagram of a software flowchart

AI-generated content may be incorrect.

Hình 3. Lưu đồ chương trình TaskResetTimeout

b) Lưu đồ chương trình node master

A diagram of a flowchart

AI-generated content may be incorrect.

Hình 3. Lưu đồ chương trình dự báo

## 3.4 Flow node red broker

A diagram of a program

AI-generated content may be incorrect.

Hình 3. Flow View Dashboard Node - RED

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3. Flow Control Dashboard Node - RED

A diagram of a system

AI-generated content may be incorrect.

Hình 3. Flow Alarm email

A diagram of a diagram

AI-generated content may be incorrect.

Hình 3.13 Flow Control mode device and thresholds Web

A diagram of a red state

AI-generated content may be incorrect.

Hình 3.14 Flow Notification state web and node red dashboard

A diagram of a function

AI-generated content may be incorrect.

Hình 3. Flow Save database MySQL

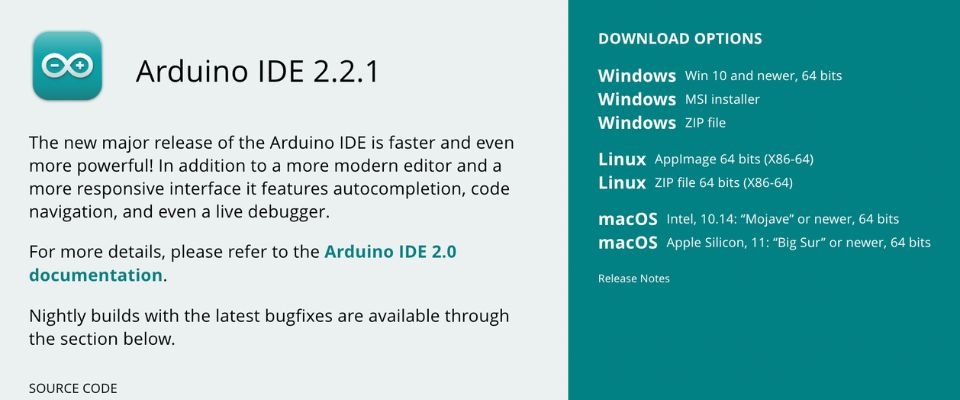
## 3.5 Phần mềm và ngôn ngữ lập trình

### 3.5.1 Arduino IDE

a) Khái niệm và lịch sử phát triển Arduino IDE

Arduino là một nền tảng mã nguồn mở được sử dụng để xây dựng các dự án điện tử. Arduino bao gồm cả bảng mạch lập trình vật lý (thường được gọi là vi điều khiển) và một phần mềm hoặc IDE (Môi trường phát triển tích hợp) chạy trên máy tính của bạn. IDE được sử dụng để viết và tải mã máy tính lên bảng vật lý.

Nền tảng Arduino đã trở nên khá phổ biến với những người mới bắt đầu làm quen với thiết bị điện tử. Không giống như hầu hết các bảng mạch lập trình trước đây, Arduino không cần một phần cứng riêng biệt (được gọi là bộ lập trình) để tải mã mới lên bảng, bạn chỉ cần sử dụng cáp USB. Ngoài ra, Arduino IDE sử dụng phiên bản C++ đơn giản hóa, giúp việc học lập trình dễ dàng hơn. Cuối cùng, Arduino cung cấp một hệ số dạng tiêu chuẩn để chia nhỏ các chức năng của bộ điều khiển vi mô thành một gói dễ tiếp cận hơn.



Hình 3. Arduino IDE

Arduino được tạo ra vào năm 2003 bởi một nhóm học giả tại Viện Thiết kế Ivrea ở Ý. Nền tảng này được đặt theo tên một quán bar ở cùng thị trấn nơi các học giả gặp nhau để thảo luận về dự án. Một trong những mục tiêu của dự án là xóa bỏ rào cản gia nhập đối với những sinh viên muốn hiện thực hóa ý tưởng tự động hóa nhưng không có đủ nguồn lực để mua bộ điều khiển đắt tiền hoặc kiến thức để chế tạo bảng mạch in của mình.

Phần cứng và phần mềm Arduino được thiết kế dành cho bất kỳ ai quan tâm đến việc tạo ra các đối tượng hoặc môi trường tương tác. Arduino có thể tương tác với các nút, đèn LED, động cơ, loa, thiết bị GPS, máy ảnh, internet và thậm chí cả điện thoại thông minh hoặc TV của bạn! Nhờ tính linh hoạt này kết hợp với phụ kiện phần cứng không quá đắt đỏ, phần mềm thiết kế dễ học dẫn đến một cộng đồng lớn người dùng đã đóng góp và đưa ra rất nhiều tài liệu, dự án thú vị khác nhau.

b) Các chức năng của Arduino IDE

Arduino IDE bao gồm các phần khác nhau:

* Window bar
* Menu bar
* Phím tắt
* Text Editor
* Output Panel

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3. Chức Năng của Arduino IDE

Thanh menu bao gồm: File, Edit, Sketch, Tools, Help

* File

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3. File menu Arduino IDE

* New: tạo một file mới. (Ctrl + N)
* Open: sử dụng để mở file đã được lưu trước đó. (Ctrl + O)
* Open Recent: hiển thị danh sách rút gọn các chương trình đã mở gần đây.
* Sketchbook: hiển thị các sketch hiện tại mà bạn đã sử dụng cho project của mình
* Examples: Ví dụ về một vài vấn đề cơ bản để tham khảo.
* Close: đóng cửa sổ màn hình chính. (Ctrl + W)
* Save: được sử dụng để lưu sketch hiện tại. (Ctrl + S)
* Save as… : cho phép lưu sketch hiện tại với một tên khác. (Ctrl + Shift + S)
* Page setup: cài đặt trang để sửa đổi trang (Văn bản). (Ctrl + Shift + P)
* Print: được sử dụng để in chương trình hiện tại. (Ctrl + P)
* Preferences: cài đặt của phần mềm IDE có thể được thay đổi tại đây. (Ctrl +,)
* Quit: đóng tất cả các cửa sổ IDE. (Ctrl + Q)
* Edit

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3. Edit menu Arduino IDE

* Undo / Redo: quay lại một hoặc nhiều bước bạn đã làm trong khi chỉnh sửa.
* Cut: cắt văn bản đã chọn khỏi trình chỉnh sửa.
* Copy: sao chép văn bản đã chọn từ trình chỉnh sửa
* Copy for Forum: sao chép và thay đổi kiểu mã phù hợp với diễn đàn.
* Copy as HTML: sao chép và thay đổi kiểu mã phù hợp với HTML.
* Paste: dán văn bản từ văn bản đã sao chép.
* Select All: chọn tất cả nội dung từ trình chỉnh sửa.
* Comment / Uncomment: sử dụng để ghi chú và bỏ ghi chú các dòng mã đã chọn.
* Increase / Decrease Indent: thêm hoặc xóa một khoảng trắng ở đầu mỗi dòng đã chọn
* Find: tìm văn bản đã nhập trong trình chỉnh sửa
* Find next: tìm vị trí tiếp theo của từ đang tìm kiếm.
* Find previous: tìm vị trí trước đó của từ đang tìm kiếm.
* Sketch

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3. Sketch menu Arduino IDE

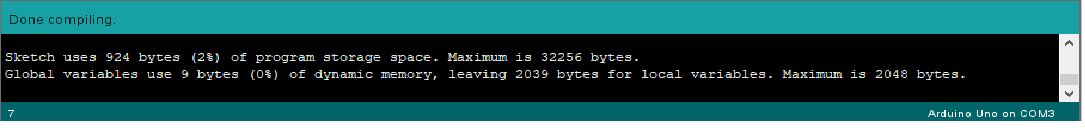
* Verify / Compile: kiểm tra hoặc xác minh chương trình của bạn nếu có bất kỳ lỗi nào và hiển thị trong bảng đầu ra.
* Upload: biên dịch và tải mã lên bo mạch.
* Upload using programmer: tải mã lên bằng Programmer có sẵn trong tab Tools.
* Export Compiled Binary: lưu file .hex trong hệ thống
* Show Sketch Folder: mở thư mục sketch hiện tại.
* Include Library: thêm thư viện vào sketch của bạn bằng cách chèn các câu lệnh #include vào đầu mã
* Add File… : thêm một file vào sketch và file mới xuất hiện trong tab mới trong cửa sổ.
* Tools

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3. Tools menu Arduino IDE

* Auto Format: định dạng mã của bạn thành một định dạng đẹp để mọi người có thể hiểu.
* Archive Sketch: sao chép mã sang định dạng winrar (.zip)
* Fix Encoding & Reload: khắc phục sự khác biệt có thể có giữa mã hóa bản đồ char của trình soạn thảo và các bản đồ char của hệ điều hành khác.
* Serial Monitor: màn hình nối tiếp hiển thị giao tiếp trực quan bằng cách gửi và nhận dữ liệu
* Board: để chọn loại bo mạch
* Port: để chọn cổng mà bạn đã kết nối mạch
* Programmer: để chọn một programmer phần cứng khi lập trình bo mạch hoặc chip và không sử dụng kiểu giao tiếp USB.
* Burn Bootloader: được sử dụng để ghi bộ nạp khởi động vào bo mạch
* Output panel



Hình 3. Output Panel Arduino IDE

* Bảng đầu ra này được sử dụng để đưa ra nhận xét về mã
* Nếu mã được biên dịch thành công hoặc bất kỳ lỗi nào xảy ra.
* Nếu mã đã được tải lên bo thành công.

### 3.5.2 Visual Studio Code (VS Code)

a) Giới thiệu

Visual Studio Code chính là ứng dụng cho phép biên tập, soạn thảo các đoạn code để hỗ trợ trong quá trình thực hiện xây dựng, thiết kế website một cách nhanh chóng. Visual Studio Code hay còn được viết tắt là VS Code. Trình soạn thảo này vận hành mượt mà trên các nền tảng như Windows, macOS, Linux. Hơn thế nữa, VS Code còn cho khả năng tương thích với những thiết bị máy tính có cấu hình tầm trung vẫn có thể sử dụng dễ dàng.

Visual Studio Code hỗ trợ đa dạng các chức năng Debug, đi kèm với Git, có Syntax Highlighting. Đặc biệt là tự hoàn thành mã thông minh, Snippets, và khả năng cải tiến mã nguồn. Nhờ tính năng tùy chỉnh, Visual Studio Code cũng cho phép các lập trình viên thay đổi Theme, phím tắt, và đa dạng các tùy chọn khác. Mặc dù trình soạn thảo Code này tương đối nhẹ, nhưng lại bao gồm các tính năng mạnh mẽ.

Dù mới được phát hành nhưng VSCode là một trong những Code Editor mạnh mẽ và phổ biến nhất dành cho lập trình viên. Nhờ hỗ trợ nhiều ngôn ngữ lập trình phổ biến, tích hợp đầy đủ các tính năng và khả năng mở rộng, nên VSCode trở nên cực kì thân thuộc với bất kì lập trình viên nào.



Hình 3. Visual Studio Code

b) Những ưu điểm nổi bật của Visual Studio Code

Visual Studio Code là gì được rất nhiều người tìm hiểu. Đây cũng là một trong các ứng dụng được dân IT “săn đón” và tải về và sử dụng rất nhiều. Visual Studio Code cũng luôn có những cải tiến và tạo ra đa dạng các tiện ích đi kèm từ đó giúp cho các lập trình viên sử dụng dễ dàng hơn. Trong đó có thể kể đến những ưu điểm sau:

* Đa dạng ngôn ngữ lập trình giúp người dùng thỏa sức sáng tạo và sử dụng như HTML, CSS, JavaScript, C++,…
* Ngôn ngữ, giao diện tối giản, thân thiện, giúp các lập trình viên dễ dàng định hình nội dung.
* Các tiện ích mở rộng rất đa dạng và phong phú.
* Tích hợp các tính năng quan trọng như tính năng bảo mật (Git), khả năng tăng tốc xử lý vòng lặp (Debug),…
* Đơn giản hóa việc tìm quản lý hết tất cả các Code có trên hệ thống.

Visual Studio Code là một trong những trình biên tập Code rất phổ biến nhất hiện nay. Ứng dụng này cũng ngày càng chứng tỏ ưu thế vượt trội của mình khi so sánh với những phần mềm khác. Tuy bản miễn phí không có nhiều các tính năng nâng cao nhưng Visual Studio Code thực sự có thể đáp ứng được hầu hết nhu cầu cơ bản của lập trình viên.

### 3.5.3 C++

a) Giới thiệu

C++ là một ngôn ngữ lập trình được phát triển vào năm 1979 tại phòng thí nghiệm Bell bởi Bjarne Stroustrup. Đây được gọi là ngôn ngữ bậc trung (middle-level) và là một phần mở rộng của ngôn ngữ lập trình C hoặc “C với các lớp Class” vì C++ là sự kết hợp các tính năng của ngôn ngữ cấp cao và cấp thấp.

Ngôn ngữ lập trình này có thể chạy trên nhiều nền tảng khác nhau như: Windows, macOS và Unix. Từ những thập niên 1990, C++ đã trở thành một trong những ngôn ngữ thương mại được ưa thích của các lập trình viên. Theo thời gian nó đã mở rộng đáng kể và hiện nay có nhiều tính năng như: lập trình tổng quát, lập trình thủ tục, lập trình hướng đối tượng,...



Hình 3. C++

b) Đặc điểm nổi bật:

Hiệu suất cao: C++ cung cấp khả năng thao tác bộ nhớ cấp thấp và kiểm soát tối ưu tài nguyên hệ thống, phù hợp cho các ứng dụng yêu cầu hiệu suất cao như game, hệ thống nhúng và phần mềm hệ thống.

Tính linh hoạt: Hỗ trợ lập trình hướng đối tượng, tổng quát và hàm, giúp giải quyết nhiều loại vấn đề khác nhau.

Nền tảng vững chắc: Là nền tảng cho nhiều thư viện và framework, hỗ trợ phát triển phần mềm phức tạp.

Ứng dụng rộng rãi: Sử dụng trong nhiều lĩnh vực như hệ điều hành, trình biên dịch, phần mềm mô phỏng khoa học, tài chính, và nhiều lĩnh vực khác.

Tiêu chuẩn hóa và cộng đồng: C++ được tiêu chuẩn hóa bởi ISO và có một cộng đồng lập trình viên lớn hỗ trợ.

c) Ứng dụng

C++ được sử dụng theo nhiều cách khác nhau để tận dụng tính linh hoạt và hiệu suất của nó. Một số ứng dụng chính của C++ bao gồm:

* Hệ điều hành (OS): Khả năng cấp thấp của C++ cho phép quản lý bộ nhớ hiệu quả và kiểm soát tài nguyên hệ thống, phù hợp cho phát triển các hệ điều hành.
* Trò chơi và đồ họa: Hiệu suất cao của C++ và khả năng tương tác với phần cứng làm cho nó là lựa chọn tốt cho các engine game như Unreal Engine của Epic Games và Unity của Unity Technologies, cũng như các tác vụ lập trình đồ họa như render thời gian thực, xử lý hình ảnh và mô phỏng vật lý.
* Hệ thống nhúng: C++ thường được sử dụng trong phát triển các hệ thống nhúng, như thiết bị y tế, hệ thống ô tô và các thiết bị Internet of Things.
* Thư viện phần mềm: C++ là nền tảng cho nhiều thư viện cấp cao, như Boost và Thư viện Mẫu Chuẩn (STL), cung cấp các cấu trúc dữ liệu và thuật toán tổng quát.
* Tính toán hiệu suất cao (HPC): C++ phù hợp cho các ứng dụng HPC đòi hỏi hiệu suất tính toán tối đa và xử lý song song, thường được sử dụng trong mô phỏng khoa học, phân tích số và mô hình toán học.
* Phát triển web: Mặc dù C++ không thường được sử dụng cho phát triển web, nó đóng vai trò quan trọng trong backend của các ứng dụng web, như máy chủ web và các giao thức mạng.
* Lập trình hệ thống: C++ thường được sử dụng cho các tác vụ lập trình hệ thống, như driver thiết bị, giao thức mạng và tiện ích hệ thống.

### 3.5.4 Python

a) Giới thiệu

Python là một ngôn ngữ lập trình cấp cao (high-level programing language) mã nguồn mở và đa năng, được sử dụng rộng rãi trong nhiều lĩnh vực như phát triển web, khoa học dữ liệu, trí tuệ nhân tạo, và tự động hóa hệ thống, ...

A person sitting at a desk working on a computer

AI-generated content may be incorrect.

Hình 3. Python

Python là ngôn ngữ kịch bản (scripting language) giống với Javascript hay Ruby, thay vì biên dịch toàn bộ chương trình thành mã máy như các ngôn ngữ như C, C++, Golang hoặc Java, các ngôn ngữ scripting thường được diễn giải (interpreted) trực tiếp khi chạy. Điều này giúp quá trình phát triển nhanh hơn, vì lập trình viên có thể viết, chạy và sửa lỗi một cách dễ dàng mà không cần qua nhiều bước biên dịch.

b) Đặc điểm nổi bật

* Python có cú pháp đơn giản và dễ hiểu, làm cho nó trở thành ngôn ngữ lập trình phù hợp cho các developer chưa nhiều kinh nghiệm và thu hút được một cộng đồng người dùng lớn. Vì có nhiều học giả và giáo sư trong cộng đồng, khi gặp sự cố, các developer có thể nhận được sự hỗ trợ nhanh chóng và không phải lo lắng về tính phức tạp của ngôn ngữ.
* Python được phát triển theo giấy phép nguồn mở được phê chuẩn bởi OSI, vì vậy nó là một ngôn ngữ tự do sử dụng và phân phối, bao gồm cả cho mục đích thương mại. Điều này giúp giảm chi phí cho việc bảo trì và nhà phát triển có thể chia sẻ, sao chép và tùy chỉnh nó. Đối với cộng đồng Python, đây là cơ hội để chia sẻ kiến thức và kinh nghiệm với những người mới vào nghề.
* Người lập trình cho biết Python rất dễ sử dụng. Mặc dù khi phát triển ứng dụng hoặc trò chơi cho thiết bị di động, có thể có các ngôn ngữ khác như C++ hoặc các ngôn ngữ lập trình kịch bản khác dễ sử dụng hơn, nhưng Python đều cho kết quả tốt hơn trong việc xây dựng ứng dụng trên máy chủ, tự động hóa quá trình xây dựng hệ thống và thu thập dữ liệu thử nghiệm.
* Python có nhiều thư viện và framework để lựa chọn, đây là một trong những ưu điểm lớn của Python. Từ NumPy đến TensorFlow, các thư viện Python được sử dụng cho mọi thứ từ trực quan hóa dữ liệu, học máy, khoa học dữ liệu, xử lý ngôn ngữ tự nhiên và phân tích dữ liệu phức tạp.
* Python có một thư viện lớn hỗ trợ quản lý bộ nhớ và có thiết kế sáng sủa để tăng năng suất cho các lập trình viên Python. Nhờ đó, họ có thể quản lý cơ sở dữ liệu, tài liệu, trình duyệt web, thực hiện kiểm thử đơn vị và nhiều chức năng khác. Bên cạnh đó, Python có thể được sử dụng để phát triển ứng dụng web, máy tính để bàn, hệ thống tính toán phức tạp, hệ thống quản lý hỗ trợ cuộc sống, Internet of Things (IoT), trò chơi, và nhiều hơn thế nữa.

CHƯƠNG 4 THIẾT KẾ MẠCH ĐIỆN

## 4.1 sơ đồ nguyên lý

A diagram of a circuit board

AI-generated content may be incorrect.

Hình 4. Sơ đồ nguyên lý Node slave

## 4.2 Mạch PCB

A computer screen shot of a circuit board

AI-generated content may be incorrect.

Hình 4. Mạch PCB Node slave

## 4.3 Sản phẩm hoàn thiện

A computer screen shot of a circuit board

AI-generated content may be incorrect.

Hình 4. Mạch hoàn thiện

A screenshot of a computer

AI-generated content may be incorrect.

Hình 4. Webserver forecast and control Node master

A screenshot of a graph

AI-generated content may be incorrect.

Hình 4. Webserver chart Node master

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

**Kết luận**

Hệ thống giám sát và dự báo chất lượng không khí trong phòng bệnh được xây dựng dựa trên công nghệ IoT đã đạt được các mục tiêu đề ra. Hệ thống sử dụng ESP32 (node slave) để thu thập dữ liệu từ các cảm biến như DHT11, MQ-7, MQ-135, GP2Y1010AU0F, và BMP280, truyền dữ liệu qua giao thức MQTT đến Raspberry Pi 4 (node trung tâm). Raspberry Pi 4 đóng vai trò MQTT Broker, chạy web server, và triển khai mô hình Prophet để dự báo các thông số môi trường như AQI và PM2.5.

Các chức năng chính của hệ thống bao gồm:

* Thu thập và giám sát thời gian thực: Hệ thống cung cấp dữ liệu chính xác về nhiệt độ, độ ẩm, bụi mịn, CO, CO₂, và áp suất, hỗ trợ theo dõi chất lượng không khí liên tục.
* Dự báo chất lượng không khí: Mô hình Prophet dự báo xu hướng AQI và PM2.5 trong 24 giờ tới, giúp đưa ra quyết định chủ động trong việc điều chỉnh môi trường.
* Tự động hóa và điều khiển từ xa: Hệ thống tự động điều khiển các thiết bị như máy lọc không khí và điều hòa thông qua relay, đồng thời cho phép điều khiển thủ công qua giao diện web.
* Lưu trữ và phân tích dữ liệu: Dữ liệu cảm biến và kết quả dự báo được lưu trữ trong cơ sở dữ liệu (MySQL/SQLite), hỗ trợ phân tích và báo cáo.

Hệ thống không chỉ đáp ứng nhu cầu giám sát môi trường trong phòng bệnh mà còn có tiềm năng ứng dụng trong các môi trường nhạy cảm khác như trường học, văn phòng, và nhà ở. Việc tích hợp FreeRTOS trên ESP32 đảm bảo hiệu suất đa nhiệm, trong khi Raspberry Pi 4 cung cấp nền tảng mạnh mẽ cho dự báo và giao diện người dùng. Dự án đã chứng minh tính khả thi của việc sử dụng các công nghệ IoT giá rẻ để cải thiện chất lượng không khí và hỗ trợ sức khỏe cộng đồng.

**Hạn chế và khó khăn**

Mặc dù hệ thống đã hoạt động hiệu quả, vẫn tồn tại một số hạn chế và khó khăn:

* Hiệu suất của Raspberry Pi 4: Raspberry Pi 4 phải xử lý đồng thời nhiều tác vụ (MQTT Broker, web server, mô hình Prophet), dẫn đến nguy cơ quá tải khi dữ liệu lớn hoặc tần suất dự báo cao. Hiệu năng có thể bị ảnh hưởng trên các phiên bản Pi 4 có RAM thấp (2GB).
* Độ chính xác của cảm biến: Các cảm biến giá rẻ như DHT11 và GP2Y1010AU0F có độ chính xác hạn chế, đặc biệt trong điều kiện môi trường thay đổi nhanh (nhiệt độ cao, độ ẩm cao), ảnh hưởng đến chất lượng dữ liệu đầu vào cho mô hình Prophet.
* Dữ liệu lịch sử hạn chế: Mô hình Prophet yêu cầu dữ liệu dài hạn để nhận diện xu hướng và mùa vụ chính xác. Trong giai đoạn thử nghiệm, lượng dữ liệu thu thập được còn ít, dẫn đến kết quả dự báo chưa tối ưu.
* Kết nối mạng: Hệ thống phụ thuộc vào WiFi ổn định để truyền dữ liệu qua MQTT. Trong môi trường có tín hiệu yếu hoặc mất kết nối, dữ liệu có thể bị gián đoạn.
* Tích hợp giao diện người dùng: Giao diện web server hiện tại còn đơn giản, thiếu các tính năng tương tác nâng cao như biểu đồ động hoặc thông báo đẩy (push notification) khi chất lượng không khí vượt ngưỡng.

Các khó khăn trong quá trình thực hiện bao gồm:

* Cấu hình Raspberry Pi 4: Việc cài đặt và tối ưu hóa MQTT Broker, web server, và Prophet trên Pi 4 đòi hỏi thời gian và kiến thức về Linux, đặc biệt khi xử lý xung đột phần mềm.
* Tích hợp FreeRTOS: Việc triển khai FreeRTOS trên ESP32 để quản lý đa nhiệm gặp khó khăn trong việc gỡ lỗi và điều chỉnh ưu tiên tác vụ, đặc biệt khi tích hợp với thư viện MQTT.
* Dữ liệu cảm biến không đồng nhất: Các cảm biến có tần suất đọc khác nhau, gây khó khăn trong việc đồng bộ hóa dữ liệu trước khi gửi qua MQTT hoặc lưu trữ.

**Hướng phát triển**

Để nâng cao hiệu quả và mở rộng ứng dụng của hệ thống, các hướng phát triển sau được đề xuất:

* + - 1. Nâng cấp phần cứng:
* Sử dụng Raspberry Pi 4 với RAM cao hơn (4GB/8GB) hoặc chuyển một số tác vụ (như chạy Prophet) sang máy chủ đám mây để giảm tải.
* Thay thế các cảm biến giá rẻ bằng cảm biến công nghiệp có độ chính xác cao (như cảm biến bụi PMS5003 hoặc cảm biến khí chất lượng cao) để cải thiện dữ liệu đầu vào.
  + - 1. Cải thiện mô hình dự báo:
* Thu thập dữ liệu dài hạn hơn (vài tháng) để tăng độ chính xác của Prophet trong việc nhận diện xu hướng và mùa vụ.
* Tích hợp các yếu tố ngoại lai (như dữ liệu thời tiết từ API bên ngoài) vào mô hình Prophet để nâng cao chất lượng dự báo.
* Thử nghiệm các mô hình dự báo khác (như LSTM hoặc ARIMA) để so sánh và chọn mô hình phù hợp nhất.
  + - 1. Tối ưu hóa kết nối:
* Triển khai cơ chế lưu trữ cục bộ trên ESP32 khi mất kết nối WiFi, sau đó đồng bộ dữ liệu khi kết nối được khôi phục.
* Sử dụng giao thức thay thế như LoRa hoặc Zigbee trong môi trường có WiFi không ổn định.
  + - 1. Nâng cấp giao diện người dùng:
* Phát triển ứng dụng di động hoặc cải thiện web server với các tính năng như biểu đồ động (sử dụng Plotly), thông báo đẩy qua email/SMS, và tùy chỉnh ngưỡng cảnh báo.
* Tích hợp trí tuệ nhân tạo để đưa ra khuyến nghị tự động (ví dụ: đề xuất thời điểm bật máy lọc không khí dựa trên dự báo).
  + - 1. Mở rộng ứng dụng:
* Áp dụng hệ thống vào các môi trường khác như trường học, nhà máy, hoặc khu công nghiệp, với khả năng giám sát nhiều phòng hoặc khu vực cùng lúc.
* Tích hợp với hệ thống quản lý tòa nhà thông minh (BMS) để điều khiển đồng bộ các thiết bị môi trường.
  + - 1. Tăng cường bảo mật:
* Triển khai mã hóa dữ liệu MQTT (sử dụng TLS/SSL) để bảo vệ thông tin truyền qua mạng.
* Thêm xác thực người dùng trên web server để ngăn chặn truy cập trái phép.

# TÀI LIỆU THAM KHẢO

[] Achim Pieters - ESP32 PinOut

[] randomnerdtutorials – ESP32

[] randomnerdtutorials - Raspberry Pi

[] Upton, E., & Halfacree, G. (2021). Raspberry Pi User Guide (5th ed.). Wiley.

[] Dr V M Senthil kumar, Professor Mr.M.Anantha Guptha, Assistant Professor - INTERNET OF THINGS & ITS APPLICATIONS

[] MDPI (Ed.). (2018). Air Quality Monitoring and Forecasting. MDPI.

[] Fine-Grained Time Series Forecasting At Scale With Facebook Prophet And Apache Spark

[] Time-Series Forecasting With Facebook Prophet

[] Rob J Hyndman and George Athanasopoulos Monash University, Australia Forecasting: Principles and Practice (3rd ed)

# PHỤ LỤC

Source code: https://drive.google.com/drive/folders/1d5IvDrUDwNZsXmMsXplHw2risLW8JuJY?usp=sharing