

MỤC LỤC

DANH MỤC HÌNH VẼ VÀ BẢNG BIỂU	3
DANH MỤC NHỮNG TỪ VIẾT TẮT.....	5
MỞ ĐẦU.....	6
CHƯƠNG 1.TỔNG QUAN VỀ HỆ THỐNG NHÀ THÔNG MINH.....	8
1.1. Mô hình hệ thống nhà thông minh	8
1.2. Thành phần thiết bị trong hệ thống.....	9
1.2.1 Trung tâm điều khiển (Hub):	9
1.2.2. Cảm biến:.....	10
1.2.3. Thiết bị điều khiển:.....	10
1.2.4. Hệ thống an ninh:	10
1.2.5. Phần mềm quản lý:	11
1.3. Các công nghệ trong hệ thống nhà thông minh.....	11
Chương 2. PHÁT TRIỂN PHẦN MỀM CHO HỆ THỐNG IOT DỰA TRÊN NỀN TẢNG NODE-RED	12
2.1. Tổng quan về hệ thống IoT	12
2.1.1 Giới thiệu về IoT.....	12
2.1.2 Kiến trúc IoT	13
2.1.3 Các giao thức IoT	15
2.1.4 Các kỹ thuật phát triển ứng dụng IoT.....	16
2.2. Nền tảng phát triển cho IoT.....	16
2.2.1 Các nền tảng Iot (Iot platform).....	16
2.2.2 Các nền tảng phát triển phần mềm cho thiết bị IoT	18
2.3. Nền tảng phần mềm Node-Red	20
2.3.1 Tổng quan và cài đặt.....	20
2.3.2 Các thành phần cơ bản và quy tắc lập trình.....	25
CHƯƠNG 3. THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG NHÀ THÔNG MINH DỰA TRÊN NỀN TẢNG NODE-RED.....	33
3.1. Mô hình hệ thống nhà thông minh tích hợp IoT	33

3.2. Thiết kế các thành phần thiết bị trong hệ thống	34
3.2.1. Module ESP32	34
3.2.2. Thiết bị cảm biến	37
3.2.2.1. Cảm biến nhiệt độ, độ ẩm DHT11	37
3.2.2.2. Cảm biến khí Gas MQ135	39
3.2.2.3 Cảm biến cường độ ánh sáng Lux BH1750	40
3.2.2.4 Cảm biến thân nhiệt chuyển động PIR SR505 Mini	41
3.2.2.6 RFID RC522	43
3.2.3 Thiết bị chấp hành	44
3.2.3.1 Còi Buzzer 5VDC	44
3.2.3.2 Relay 5V 10A 5 Chân SRD 05VDC-SL-C	44
3.2.3.3 Động cơ servo SG90	45
3.2.3.4. Động cơ Step và Module A4988	46
3.2.4 Thiết bị quản lý trung tâm (máy chủ hệ thống)	48
3.2.4.1Giới thiệu về ThingsBoard	48
3.2.4.2 Gửi dữ liệu lên máy chủ Thingsboard	49
3.2.4.3 Truy vấn để đọc giá trị thuộc tính của thiết bị trên máy chủ Thingsboard ...	51
3.3. Giảm đồ chương trình cho hệ thống xây dựng trên nền tảng Node-Red	52
3.4. Thử nghiệm và đánh giá	56
KẾT LUẬN VÀ KIẾN NGHỊ	Error! Bookmark not defined.
TÀI LIỆU THAM KHẢO	Error! Bookmark not defined.

DANH MỤC HÌNH VẼ VÀ BẢNG BIỂU

Hình 1.1: Mô hình nhà thông minh	8
Hình 1.2: Các thành phần cơ bản của nhà thông minh.....	9
Hình 2.1. “Internet of Things”	12
Hình 2.2. Sự gia tăng nhanh chóng của giao tiếp máy – máy.	13
Hình 2.3 Kiến trúc IoT chuẩn.....	13
Hình 2.4 Các kiến trúc phân tầng của IoT.....	14
Hình 2.5 Ví dụ các thành phần của giao thức MQTT	16
Hình 2.6 Ví dụ về quy trình làm việc	20
Hình 2.7 Ví dụ về thiết kế FBP đơn giản	21
Hình 2.8 Node-RED Flow Editor là một công cụ FBP	21
Hình 2.9 Chọn phiên bản được đề xuất.....	23
Hình 2.10 Thỏa thuận cấp phép cho người dùng	24
Hình 2.11 Cài đặt thư mục đích.....	24
Hình 2.12 Không cần thiết lập tùy chỉnh.....	25
Hình 2.13 Cửa sổ công cụ cho modules gốc	25
Hình 2.14 Sử dụng Flow Editor	26
Hình 2.15 Trình biên tập luồng Node-RED	27
Hình 2.16 Danh sách luồng biên tập Node-RED	27
Hình 2.17 Node Inject	28
Hình 2.18 Thiết lập thuộc tính nút inject.....	29
Hình 2.19 Nhấn nút Done để đóng bảng cài đặt.	29
Hình 2.20 Đặt nút Debug và đầu dây	30
Hình 2.21 Bật bảng điều khiển Debug	30
Hình 2.22 Thực hiện luồng và kiểm tra kết quả	30
Hình 3.1 Mô hình nhà thông minh dựa trên nền tảng Node-red	33
Hình 3.2 Bố trí các chân GPIO RTC.....	34
Hình 3.3: Mô đun và chân của DHT11	37
Hình 3.4 Sơ đồ kết nối điện tử của mô đun DHT11 với ESP32	38

Hình 3.5	Quá trình truyền thông của DHT11	38
Hình 3.6	Mô đun MQ135 và các chân tín hiệu	39
Hình 3.7	Kết nối MQ135 với ESP32.....	40
Hình 3.8	Cảm Biến Cường Độ Ánh Sáng Lux BH1750	40
Hình 3.9	kết nối Lux BH1750 với ESP32.....	41
Hình 3.10	Cảm biến thân nhiệt chuyển động PIR SR505 Mini	42
Hình 3.11	kết nối SR505 với ESP32.	42
Hình 3.14	RFID NFC 13.56MHz RC522.....	43
Hình 3.15	kết nối RFID RC522 với ESP32.	43
Hình 3.16	Còi Buzzer 5VDC.....	44
Hình 3.17	Relay 5V	44
Hình 3.18	Động cơ servo SG90.....	45
Hình 3.19	Động cơ bước size 42 1.8 step.....	46
Hình 3.20	Mạch Điều Khiển Động Cơ Bước A4988	46
Hình 3.21	Kết nối giữa A4899 với ESP32	47
Hình 3.22	Cấu trúc của máy chủ Thingsboard	48
Hình 3.23	Giao diện quản lý, giám sát dữ liệu (dashboard) trên Thingsboard	48
Hình 3.24	Sơ đồ xử lý dữ liệu trên Node-red	52
Hình 3.25	Sơ đồ quá trình xử lý dữ liệu cảm biến độ ẩm trên Node-red	53
Hình 3.26	Sơ đồ xử lý dữ liệu của cảm biến nhiệt độ trên Node-red.....	53
Hình 3.27	Sơ đồ xử lý dữ liệu của nút bấm đèn trên Node-red.....	54
Hình 3.28	Sơ đồ xử lý dữ liệu của cảm biến ánh sáng trên Node-red.....	54
Hình 3.29	Sơ đồ xử lý dữ liệu của cửa cuốn trên Node-red.....	55
Hình 3.30.	Một số hình ảnh hệ thống thử nghiệm.....	56
Hình 3.31.	Kết quả chạy hệ thống với đám mây Thingsboard.....	57

DANH MỤC NHỮNG TỪ VIẾT TẮT

Từ viết tắt	Từ đầy đủ	Tiếng Việt
WSN	Wireless Sensor Network	Mạng cảm biến không dây
IoT	Internet of Things	Internet vạn vật
MQTT	Message Queuing Telemetry Transport	Giao thức truyền thông nhẹ cho IoT
HTTP	Hypertext Transfer Protocol	Giao thức truyền tải siêu văn bản
UI	User Interface	Giao diện người dùng
UX	User Experience	Trải nghiệm người dùng
AI	Artificial Intelligence	Trí tuệ nhân tạo
ML	Machine Learning	Học máy
RTOS	Real-Time Operating System	Hệ điều hành thời gian thực
FBP	Flow-Based Programming	Lập trình dựa trên luồng
LED	Light Emitting Diode	Điốt phát quang
DAC	Digital-to-Analog Converter	Bộ chuyển đổi từ số sang tương tự
ADC	Analog-to-Digital Converter	Bộ chuyển đổi từ tương tự sang số
PIR	Passive Infrared Sensor	Cảm biến hồng ngoại thụ động
JSON	JavaScript Object Notation	Định dạng dữ liệu JSON
XML	eXtensible Markup Language	Ngôn ngữ đánh dấu mở rộng
IDE	Integrated Development Environment	Môi trường phát triển tích hợp
SDK	Software Development Kit	Bộ công cụ phát triển phần mềm
GPIO	General Purpose Input/Output	Chân vào/ra mục đích chung
ROM	Read Only Memory	Bộ nhớ chỉ đọc
RAM	Random Access Memory	Bộ nhớ truy cập ngẫu nhiên
UART	Universal Asynchronous Receiver-Transmitter	Giao tiếp nối tiếp không đồng bộ
I2C	Inter-Integrated Circuit	Giao tiếp nối tiếp nội mạch
SPI	Serial Peripheral Interface	Giao diện ngoại vi nối tiếp
CAN	Controller Area Network	Mạng truyền thông điều khiển

MỞ ĐẦU

1. Tổng quan tình hình nghiên cứu thuộc lĩnh vực đề tài:

Trong những năm gần đây, sự phát triển mạnh mẽ của Internet of Things (IoT) đã mở ra nhiều cơ hội cho việc ứng dụng công nghệ vào đời sống, đặc biệt là trong lĩnh vực nhà thông minh. Các hệ thống nhà thông minh giúp tối ưu hóa việc quản lý và điều khiển các thiết bị điện trong nhà, nâng cao sự tiện nghi, tiết kiệm năng lượng và đảm bảo an toàn cho người sử dụng.

Một trong những thách thức lớn trong việc xây dựng hệ thống nhà thông minh là khả năng kết nối và điều khiển đồng bộ nhiều thiết bị từ các nhà sản xuất khác nhau. Đồng thời, việc phát triển một nền tảng quản lý trực quan, dễ sử dụng và linh hoạt cũng là một vấn đề quan trọng. Trong bối cảnh đó, Node-RED, một nền tảng lập trình trực quan mã nguồn mở của IBM, nổi lên như một giải pháp tiềm năng, giúp đơn giản hóa việc tích hợp các thiết bị IoT và phát triển hệ thống điều khiển.

2. Lý do lựa chọn đề tài

Việc nghiên cứu và xây dựng một hệ thống giám sát và điều khiển nhà thông minh dựa trên nền tảng Node-RED mang lại nhiều lợi ích, bao gồm:

- Tích hợp linh hoạt: Node-RED hỗ trợ kết nối với nhiều giao thức IoT khác nhau như MQTT, HTTP, WebSocket...
- Giao diện trực quan: Cho phép lập trình bằng cách kéo-thả các node, dễ dàng tùy chỉnh và mở rộng hệ thống.
- Khả năng mở rộng và bảo mật: Hệ thống có thể dễ dàng nâng cấp để tích hợp thêm nhiều thiết bị và đảm bảo an toàn trong quá trình truyền dữ liệu.

Từ những lợi ích trên, đề tài "Nghiên cứu và xây dựng hệ thống giám sát và điều khiển cho nhà thông minh dựa trên nền tảng Node-RED và công nghệ IoT" có ý nghĩa quan trọng trong việc phát triển các ứng dụng tự động hóa, tối ưu hóa việc quản lý thiết bị thông minh trong gia đình.

3. Mục tiêu, nội dung, phương pháp nghiên cứu của đề tài

a. Mục tiêu:

Thiết kế và thử nghiệm mô hình hệ thống nhà thông minh được xây dựng dựa trên nền tảng phát triển Node-Red, tích hợp công nghệ IoT.

Hệ thống bao gồm các thành phần thiết bị cơ bản được xây dựng dựa trên ESP32 có khả năng tích hợp vào bên trong các thành phần trong hệ thống quản lý nhà

b. Nội dung:

Đề tài bao gồm các nội dung chính sau:

- **Chương 1: Tổng quan về hệ thống nhà thông minh**
 - Giới thiệu về các mô hình, thiết bị và công nghệ trong hệ thống nhà thông minh.

- **Chương 2: Phát triển phần mềm cho hệ thống IoT dựa trên nền tảng Node-RED**
 - Nghiên cứu các nền tảng IoT và ứng dụng Node-RED trong việc lập trình hệ thống điều khiển.
- **Chương 3: Thiết kế và xây dựng hệ thống nhà thông minh dựa trên Node-RED**
 - Thiết kế mô hình hệ thống, lập trình các thiết bị ESP32, triển khai thử nghiệm và đánh giá kết quả.

c. Phương pháp nghiên cứu:

- Phương pháp nghiên cứu trong đề tài bao gồm: nghiên cứu tài liệu để thu thập thông tin về hệ thống nhà thông minh, IoT, Node-RED và ESP32; thiết kế và xây dựng hệ thống với các thành phần phần cứng (ESP32, cảm biến, thiết bị chấp hành) và phần mềm (Node-RED, MQTT, HTTP); thực nghiệm và kiểm thử bằng cách triển khai mô hình thực tế, đánh giá hiệu suất hoạt động, độ trễ và tính ổn định; cuối cùng, phân tích và đánh giá kết quả, so sánh với các mô hình khác để đề xuất cải tiến và hướng phát triển.

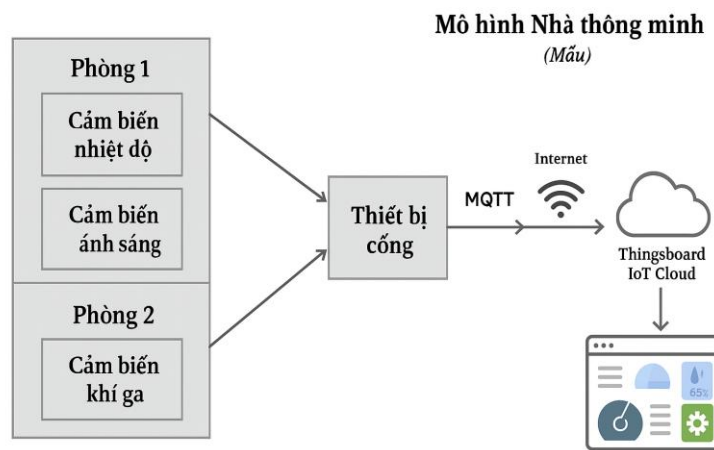
4. Đối tượng và phạm vi nghiên cứu

- Phát triển hệ thống phần mềm trên nền tảng Node-RED, tích hợp các giao thức IoT như MQTT, WebSocket để truyền và nhận dữ liệu.
- Thiết kế và lập trình các thiết bị phần cứng dựa trên ESP32, kết nối với hệ thống trung tâm để thực hiện giám sát và điều khiển.
- Thử nghiệm mô hình thực tế, đánh giá độ ổn định, tốc độ phản hồi và khả năng mở rộng của hệ thống.

CHƯƠNG 1. TỔNG QUAN VỀ HỆ THỐNG NHÀ THÔNG MINH

1.1. Mô hình hệ thống nhà thông minh

Nhà thông minh (tiếng anh là "Smart Home") hoặc hệ thống nhà thông minh là một ngôi nhà/ căn hộ được trang bị các hệ thống tự động thông minh cùng với các bố trí hợp lý, các hệ thống này có khả năng tự điều phối các hoạt động trong ngôi nhà theo thói quen sinh hoạt và nhu cầu cá nhân của gia chủ. Chúng ta cũng có thể hiểu ngôi nhà thông minh là một hệ thống chỉnh thể mà trong đó, tất cả các thiết bị điện tử gia dụng đều được liên kết với thiết bị điều khiển trung tâm và có thể phối hợp với nhau để cùng thực hiện một chức năng. Các thiết bị này có thể đưa ra cách xử lý tình huống được lập trình trước, hoặc là được điều khiển và giám sát từ xa nhằm mục đích là cho cuộc sống ngày càng tiện nghi, an toàn và góp phần sử dụng hợp lý các nguồn tài nguyên.



Hình 1.1: Mô hình nhà thông minh

Các thành phần của hệ thống nhà thông minh bao gồm các cảm biến (như cảm biến nhiệt độ, cảm biến ánh sáng hoặc do cửa mở), các bộ điều khiển hoặc máy chủ và các thiết bị chấp hành khác. Nhờ hệ thống cảm biến, các bộ điều khiển và máy chủ có thể theo dõi các trạng thái bên trong ngôi nhà để đưa ra các quyết định điều khiển các thiết bị chấp hành một cách phù hợp nhằm đảm bảo môi trường sống tốt nhất cho con người. Ngoài ra, cùng với sự phát triển của các thiết bị điện tử cá nhân như máy tính bảng và điện thoại thông minh cùng hạ tầng thông tin ngày càng tiên tiến như internet hoặc các mạng thông tin di động wifi, 4G, 5G, ngày nay các hệ thống nhà thông minh còn cung cấp khả năng tương tác với người sử dụng thông qua các giao diện cảm ứng trên smart phone cho phép con người có thể giám sát và điều khiển ngôi nhà từ bất cứ đâu. Tùy theo nhu cầu, người sử dụng có thể cấu hình hệ thống theo kịch bản bất kỳ như lập trình hẹn giờ tắt đèn khi ngủ, hoặc quên tắt tivi, kéo rèm cửa sổ,... khi tới nơi làm việc, họ có thể điều khiển qua điện thoại smartphone để điều khiển từ xa. Tùy theo mức

độ sử dụng mà mức giá của Nhà Thông Minh sẽ dao động từ vài triệu đến vài trăm triệu đồng cho một ngôi nhà.

1.2. Thành phần thiết bị trong hệ thống

Các thành phần cơ bản của hệ thống nhà thông minh bao gồm hệ thống cảm biến như cảm biến nhiệt độ, cảm biến ánh sáng hoặc do cửa chỉ, các bộ điều khiển hoặc máy chủ và các thiết bị chấp hành khác. Nhờ hệ thống cảm biến, các bộ điều khiển và máy chủ có thể theo dõi các trạng thái bên trong ngôi nhà để đưa ra các quyết định điều khiển các thiết bị chấp hành một cách phù hợp nhằm đảm bảo môi trường sống tốt nhất cho con người.



Hình 1.2: Các thành phần cơ bản của nhà thông minh

1.2.1 Trung tâm điều khiển (Hub):

Trung tâm điều khiển, hay còn gọi là Hub, là thành phần cốt lõi trong bất kỳ hệ thống nhà thông minh nào. Thiết bị này đóng vai trò như “bộ não” điều phối, kết nối và quản lý tất cả thiết bị trong hệ thống như đèn chiếu sáng, ổ cắm, cảm biến, camera an ninh hay rèm cửa thông minh.

Hub cho phép các thiết bị giao tiếp với nhau thông qua các giao thức phổ biến như Wi-Fi, Zigbee, Z-Wave hoặc Bluetooth. Nhờ đó, người dùng có thể điều khiển toàn bộ hệ thống từ xa qua ứng dụng trên smartphone hoặc máy tính bảng một cách nhanh chóng, tiện lợi và đồng bộ.

1.2.2. Cảm biến:

Cảm biến đóng vai trò như “giác quan” giúp ngôi nhà nhận biết và phản hồi với môi trường xung quanh. Chúng thu thập dữ liệu thực tế và gửi về trung tâm điều khiển để xử lý.

Một số loại cảm biến phổ biến gồm:

- **Cảm biến hiện diện:** Phát hiện sự có mặt của người trong phòng.
- **Cảm biến nhiệt độ:** Đo và theo dõi nhiệt độ không gian.
- **Cảm biến độ ẩm:** Giám sát độ ẩm không khí.
- **Cảm biến khói và khí gas:** Phát hiện khói hoặc khí độc hại, cảnh báo nguy cơ cháy nổ.

Nhờ sự hỗ trợ của các cảm biến này, hệ thống smarthome có thể tự động hóa các thiết bị điện, nâng cao tiện nghi và đảm bảo an toàn tối đa

1.2.3. Thiết bị điều khiển:

Thiết bị điều khiển là những phần tử trực tiếp tương tác với người dùng hoặc thực hiện các hành động trong kịch bản tự động hóa. Chúng cho phép bạn điều khiển ánh sáng, nhiệt độ, thiết bị điện... một cách thông minh và linh hoạt.

Bao gồm:

- Công tắc thông minh: Dùng để bật/tắt đèn và các thiết bị điện tử từ xa hoặc tự động theo lịch trình.
- Điều khiển rèm cửa và điều hòa: Tự động đóng/mở rèm cửa hoặc điều chỉnh nhiệt độ theo môi trường hoặc yêu cầu của người dùng.

Tất cả các thiết bị này đều được quản lý qua ứng dụng hoặc trung tâm điều khiển, giúp người dùng tối ưu hóa tiện nghi và tiết kiệm năng lượng trong ngôi nhà thông minh

1.2.4. Hệ thống an ninh:

An ninh là một trong những yếu tố quan trọng nhất trong hệ thống nhà thông minh. Các thiết bị an ninh được tích hợp nhằm bảo vệ tài sản và an toàn cho gia đình, đồng thời cho phép bạn giám sát ngôi nhà từ bất cứ đâu.

Thành phần bao gồm:

- Camera an ninh: Giám sát và ghi lại các hoạt động bên trong và bên ngoài ngôi nhà. Camera có thể truyền trực tiếp hình ảnh đến điện thoại của người dùng.
- Chuông cửa thông minh: Tích hợp camera và microphone, giúp chủ nhà giao tiếp với khách từ xa và theo dõi ai đang ở trước cửa.
- Khóa cửa thông minh: Điều khiển khóa từ xa qua ứng dụng hoặc mã PIN, giúp tăng cường an ninh và tiện lợi trong việc ra vào ngôi nhà.

Hệ thống này giúp phát hiện kịp thời các mối đe dọa và cung cấp biện pháp bảo vệ hiệu quả.

1.2.5. Phần mềm quản lý:

Phần mềm quản lý (ứng dụng điều khiển) là giao diện giúp người dùng điều hành toàn bộ hệ thống nhà thông minh từ xa. Thông qua ứng dụng trên điện thoại hoặc máy tính bảng, bạn có thể tương tác trực tiếp với từng thiết bị, thiết lập kịch bản và theo dõi trạng thái hệ thống.

Các chức năng chính:

- Giám sát thời gian thực: Người dùng có thể xem trực tiếp từ camera an ninh, theo dõi trạng thái thiết bị, và nhận cảnh báo khi có sự cố.
- Điều khiển thiết bị: Ứng dụng cho phép bật/tắt đèn, điều hòa, và nhiều thiết bị khác chỉ bằng một chạm, dù ở bất cứ đâu.
- Lập lịch và tự động hóa: Người dùng có thể lập lịch hoạt động cho các thiết bị, như tự động tắt đèn vào ban đêm hoặc bật điều hòa trước khi về nhà.

Ứng dụng thường được thiết kế để dễ sử dụng và tương thích với hầu hết các hệ điều hành như iOS và Android.

1.3. Các công nghệ trong hệ thống nhà thông minh

Internet of Things (IoT): Đây là nền tảng cốt lõi, cho phép các thiết bị trong nhà kết nối và giao tiếp với nhau qua internet. Nhờ IoT, bạn có thể điều khiển đèn, khóa cửa, điều hòa, camera an ninh và nhiều thiết bị khác từ xa thông qua điện thoại hoặc máy tính bảng.

Mạng không dây (Wi-Fi, Zigbee, Z-Wave): Các giao thức này tạo ra mạng lưới liên lạc không dây giữa các thiết bị thông minh và bộ điều khiển trung tâm. Mỗi giao thức có ưu và nhược điểm riêng về tốc độ, phạm vi phủ sóng và mức tiêu thụ năng lượng.

Điện toán đám mây (Cloud Computing): Nhiều hệ thống nhà thông minh sử dụng đám mây để lưu trữ dữ liệu, xử lý thông tin và cho phép bạn truy cập và điều khiển thiết bị từ bất kỳ đâu có kết nối internet.

Giao diện người dùng (User Interface - UI) và Trải nghiệm người dùng (User Experience - UX): Các ứng dụng di động, trợ lý ảo (như Google Assistant, Amazon Alexa), và màn hình cảm ứng đóng vai trò là giao diện để bạn tương tác với hệ thống nhà thông minh một cách dễ dàng và trực quan.

Cảm biến (Sensors): Các loại cảm biến khác nhau được tích hợp trong các thiết bị để thu thập thông tin về môi trường xung quanh, chẳng hạn như nhiệt độ, độ ẩm, ánh sáng, chuyển động, và trạng thái đóng/mở của cửa hoặc cửa sổ.

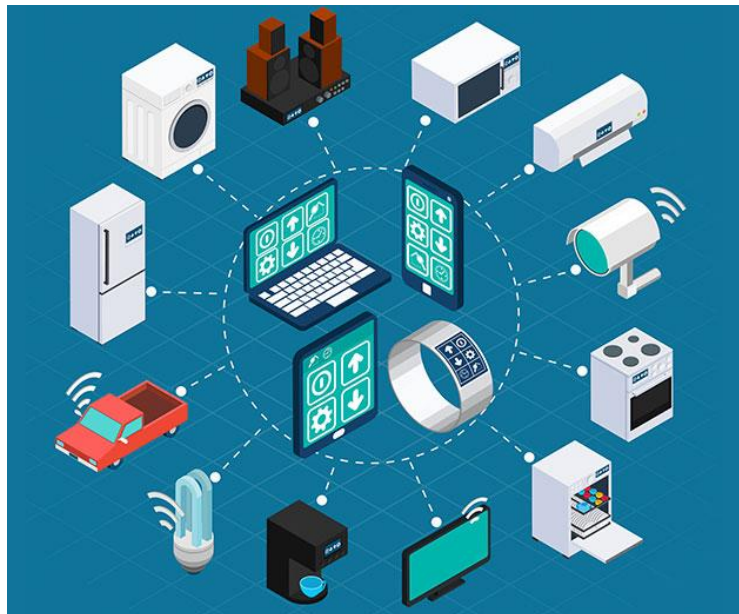
Bảo mật (Security): Các công nghệ và giao thức bảo mật được tích hợp để bảo vệ hệ thống khỏi các truy cập trái phép và đảm bảo an toàn cho dữ liệu cá nhân của bạn.

Chương 2. PHÁT TRIỂN PHẦN MỀM CHO HỆ THỐNG IOT DỰA TRÊN NỀN TẢNG NODE-RED

2.1. Tổng quan về hệ thống IoT

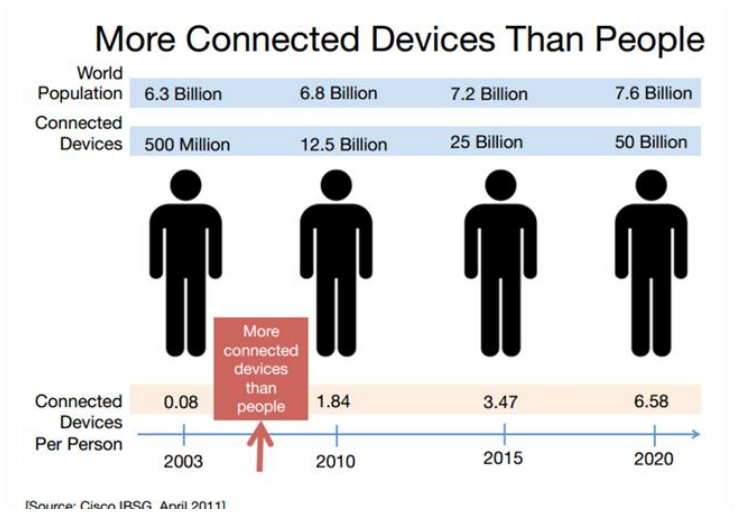
2.1.1 Giới thiệu về IoT

Internet of Things (IoT) là thuật ngữ dùng để chỉ các đối tượng có thể được nhận biết cũng như sự tồn tại của chúng trong một kiến trúc mạng tính kết nối. Đây là một viễn cảnh trong đó mọi vật, mọi con vật hoặc con người được cung cấp các định danh và khả năng tự động truyền tải dữ liệu qua một mạng lưới mà không cần sự tương tác giữa con người-với-con người hoặc con người-với-máy tính. IoT tiến hoá từ sự hội tụ của các công nghệ không dây, hệ thống vi cơ điện tử (MEMS) và Internet. Cụm từ này được đưa ra bởi Kevin Ashton vào năm 1999. Ông là một nhà khoa học đã sáng lập ra Trung tâm Auto-ID ở đại học MIT.



Hình 2.1. “Internet of Things”

"Thing" - sự vật - trong Internet of Things, có thể là một trang trại động vật với bộ tiếp sóng chip sinh học, một chiếc xe ô tô tích hợp các cảm biến để cảnh báo lái xe khi lốp quá non, hoặc bất kỳ đồ vật nào do tự nhiên sinh ra hoặc do con người sản xuất ra mà có thể được gán với một địa chỉ IP và được cung cấp khả năng truyền tải dữ liệu qua mạng lưới. IoT phải có 2 thuộc tính: một là đó phải là một ứng dụng internet. Hai là, nó phải lấy được thông tin của vật chủ.

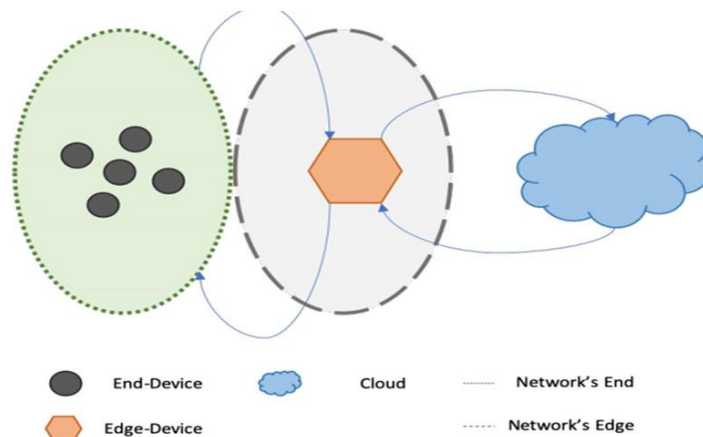


Hình 2.2. Sự gia tăng nhanh chóng của giao tiếp máy – máy.

Một ví dụ điển hình cho IoT là tủ lạnh thông minh, nó có thể là một chiếc tủ lạnh bình thường nhưng có gắn thêm các cảm biến bên trong giúp kiểm tra được số lượng các loại thực phẩm có trong tủ lạnh, cảm biến nhiệt độ, cảm biến phát hiện mở cửa,... và các thông tin này được đưa lên internet. Với một danh mục thực phẩm được thiết lập trước bởi người dùng, khi mà một trong các loại thực phẩm đó sắp hết thì nó sẽ thông báo ngay cho chủ nhân nó biết rằng cần phải bổ sung gấp, thậm chí nếu các loại sản phẩm được gắn mã ID thì nó sẽ tự động trực tiếp gửi thông báo cần nhập hàng đến siêu thị và nhân viên siêu thị sẽ gửi loại thực phẩm đó đến tận nhà.

2.1.2 Kiến trúc IoT

Công nghệ IoT gồm nhiều kiểu kiến trúc IoT khác nhau. Một kiểu kiến trúc IoT hiện nay đang được chấp nhận rộng rãi và coi là một chuẩn như hình 2.3. Nó gồm có các thiết bị đầu cuối, các thiết bị cạnh và điện toán đám mây được kết nối với nhau qua các mạng khác nhau và qua Internet.

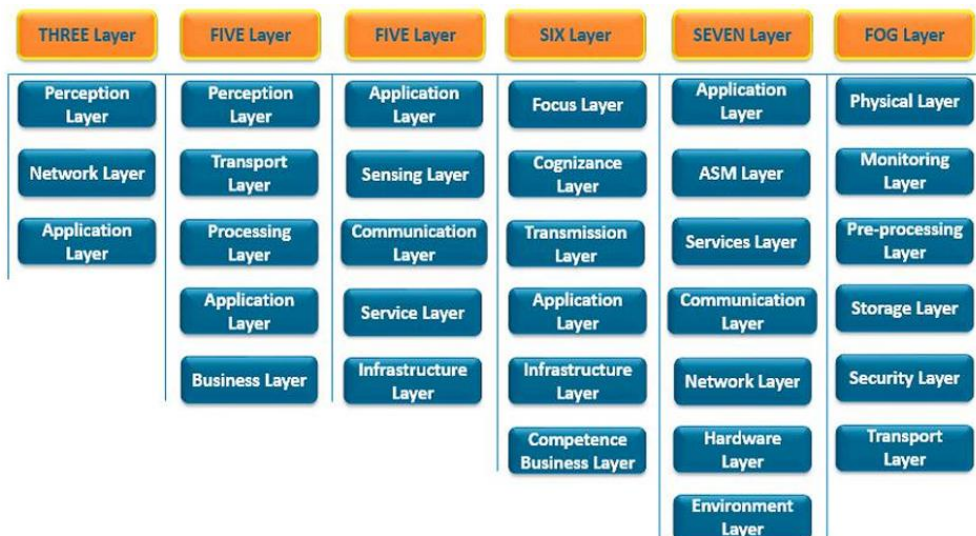


Hình 2.3 Kiến trúc IoT chuẩn

Theo kiến trúc phân tầng IoT có thể có 3 tầng, 4 tầng, 5 tầng ... Trong luận văn này mô hình kiến trúc phân tầng được sử dụng gồm 4 tầng: Các thiết bị (Things),

Gateways, hạ tầng mạng và điện toán đám mây (Network and Cloud), tầng tạo và cung cấp dịch vụ (Services-creation and Solutions Layers).

- **Thiết bị (Things):** Hiện nay, có hàng tỷ sản phẩm tồn tại trên thị trường gia dụng và công nghệ, có thể được sử dụng trong nhà hoặc mang theo bởi người dùng. Đây bao gồm các loại thiết bị như ô tô, cảm biến, thiết bị đeo và điện thoại di động, đều có khả năng kết nối trực tiếp qua mạng không dây và truy cập Internet. Giải pháp IoT giúp các thiết bị thông minh quản lý và chia sẻ dữ liệu một cách tự động, trong khi các thiết bị chưa thông minh có thể kết nối thông qua các trạm kết nối.
- **Cổng kết nối (Gateways):** Một thách thức lớn khi triển khai IoT là hầu hết các thiết bị hiện có không được thiết kế để kết nối trực tiếp với Internet và không thể chia sẻ dữ liệu với các hệ thống điện toán đám mây. Để giải quyết vấn đề này, các trạm kết nối hoạt động như một cầu nối trực tiếp, cho phép các thiết bị hiện có kết nối với các hệ thống điện toán đám mây một cách an toàn và dễ dàng quản lý.



Hình 2.4 Các kiến trúc phân tầng của IoT

- **Hạ tầng mạng và công nghệ điện toán đám mây (Network and cloud):**
 - **Cơ sở hạ tầng kết nối:** Internet được xem như một hệ thống toàn cầu gồm nhiều mạng IP kết nối với nhau và liên kết với các hệ thống máy tính. Cơ sở hạ tầng này bao gồm các thiết bị như định tuyến, trạm kết nối, thiết bị tổng hợp và lặp, cũng như nhiều thiết bị khác có khả năng quản lý lưu lượng dữ liệu trên mạng. Đồng thời, nó cũng được kết nối với các mạng viễn thông và cáp khác - đều được triển khai bởi các nhà cung cấp dịch vụ.
 - **Trung tâm dữ liệu/ hạ tầng điện toán đám mây:** Bao gồm các trung tâm dữ liệu và hạ tầng điện toán đám mây, nơi có một hệ thống lớn các máy chủ, hệ thống lưu trữ và mạng ảo hóa được kết nối.

- Các lớp tạo và cung cấp dịch vụ: Intel đã kết hợp những phần mềm quản lý API hàng đầu (Application programming interface) là Mashery* và Aepona* để giúp đưa các sản phẩm và giải pháp IoT ra thị trường một cách chóng và tận dụng được hết giá trị của việc phân tích các dữ liệu từ hệ thống và tài sản đang có sẵn.

2.1.3 Các giao thức IoT

Hệ thống IoT là hệ thống liên kết nhiều mạng với nhau và với Internet như Wifi, Bluetooth, mạng di động, ZigBee, LoRaWAN hoặc một số phương thức kết nối khác nên giao thức truyền thông của IoT rất phong phú. Các giao thức nổi bật đó là MQTT, AMQP, DDS, XMPP, CoAP.

- Giao thức MQTT (message queue telemetry transport) Giao thức MQTT, hay còn được gọi là giao thức Message Queuing Telemetry Transport, là một giao thức mạng nhẹ được sử dụng để vận chuyển các thông điệp đo từ xa giữa các thiết bị IoT. Thông thường, MQTT chạy trên giao thức TCP/IP, nhưng cũng có thể hoạt động trên các giao thức mạng khác miễn là chúng hỗ trợ các kết nối hai chiều và không mất dữ liệu.

MQTT được thiết kế nhẹ và là lựa chọn lý tưởng cho các tình huống kết nối trong đó các thiết bị IoT có thể có băng thông hạn chế hoặc các ràng buộc khác yêu cầu các thiết bị từ xa có kích thước mã nhỏ.

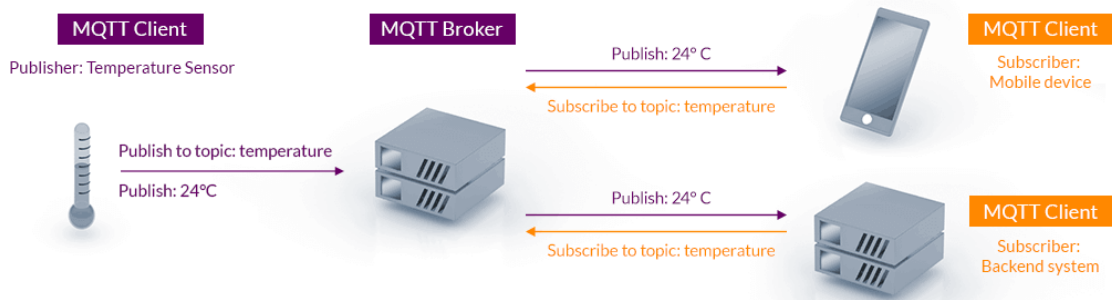
Các tính năng nổi bật của MQTT bao gồm:

- Giao thức nhẹ, phù hợp cho các mạng bị hạn chế.
- Hỗ trợ cơ chế publisher/subscriber nhắn tin.
- Tiêu thụ điện năng thấp.
- Kích thước gói dữ liệu được giảm thiểu.
- Chuẩn giao thức OASIS.

Mô hình hoạt động Pub/Sub của MQTT bao gồm các thành phần lõi sau:

- MQTT Broker: Là thành phần chính trong kiến trúc MQTT, chịu trách nhiệm lọc và phân phối các thông điệp dựa trên các chủ đề (topic).
- MQTT Client: Là các thiết bị hoặc ứng dụng kết nối đến Broker để truyền và nhận dữ liệu.

- Topic: Là cách MQTT Broker quản lý thông điệp, mỗi client có thể đăng ký (subscribe) để nhận dữ liệu từ một hoặc nhiều topic, và cũng có thể gửi dữ liệu đến một hoặc nhiều topic.



Hình 2.5 Ví dụ các thành phần của giao thức MQTT

Broker được coi như là thành phần trung tâm. Nó là điểm giao của tất cả các kết nối đến từ client (Pub/Sub). Nhiệm vụ chính của Broker là nhận thông điệp (message) từ Pub, xếp vào hàng đợi rồi chuyển đến một địa điểm cụ thể. Nhiệm vụ phụ của Broker là nó có thể đảm nhận thêm một vài tính năng liên quan tới quá trình truyền thông như: bảo mật thông điệp, lưu trữ thông điệp, logs, Client thì được chia thành hai nhóm là Publisher và Subscriber. Client chỉ làm ít nhất một trong 2 việc là xuất bản các thông điệp lên một/nhiều topic cụ thể hoặc đăng ký Sub một/nhiều topic nào đó để nhận thông điệp từ topic này. MQTT Clients tương thích với hầu hết các nền tảng hệ điều hành hiện có: MAC OS, Windows, Linux, Android, iOS,

2.1.4 Các kỹ thuật phát triển ứng dụng IoT

Theo điện toán đám mây các kỹ thuật phát triển hệ thống IoT gồm các kỹ thuật sau:

- Sử dụng IoT Google
- IoT AWS của Amazon
- IoT Azure của Microsoft
- IoT mã nguồn mở (IoT Thingsboard, ...)

2.2. Nền tảng phát triển cho IoT

2.2.1 Các nền tảng Iot (Iot platform)

Nền tảng IoT là một bộ các thành phần phần mềm và phần cứng, được thiết kế để hỗ trợ việc xây dựng, triển khai, quản lý và phân tích các ứng dụng và giải pháp IoT. Nó đóng vai trò là cầu nối giữa các thiết bị IoT, mạng lưới kết nối và các ứng dụng người dùng hoặc hệ thống phân tích dữ liệu.

- Kiến trúc cơ bản của một nền tảng IoT:

Mặc dù có nhiều biến thể, một kiến trúc nền tảng IoT thường bao gồm các lớp chính sau:

- Lớp thiết bị (Device Layer): Bao gồm các cảm biến, thiết bị chấp hành và các thiết bị IoT khác thu thập dữ liệu từ môi trường hoặc thực hiện các hành động.
- Lớp kết nối (Connectivity Layer): Chịu trách nhiệm truyền dữ liệu từ các thiết bị đến nền tảng và ngược lại. Lớp này bao gồm các giao thức truyền thông (ví dụ: MQTT, CoAP, HTTP), các công nghệ mạng (ví dụ: Wi-Fi, Bluetooth, Cellular, LoRaWAN) và các cổng (gateways) để quản lý kết nối.
- Lớp xử lý dữ liệu (Data Processing Layer): Nơi dữ liệu thu thập được xử lý, lọc, chuyển đổi và phân tích. Lớp này có thể bao gồm các chức năng như lưu trữ dữ liệu, xử lý luồng dữ liệu (stream processing), phân tích thời gian thực và tích hợp với các hệ thống khác.
- Lớp ứng dụng (Application Layer): Cung cấp giao diện người dùng và các công cụ để tương tác với dữ liệu IoT, xây dựng các ứng dụng và quản lý toàn bộ hệ thống IoT.

- Các chức năng chính của một nền tảng IoT:

- Quản lý thiết bị (Device Management): Cho phép đăng ký, cấu hình, giám sát, điều khiển và cập nhật các thiết bị IoT từ xa.
- Kết nối thiết bị (Device Connectivity): Hỗ trợ nhiều giao thức và công nghệ kết nối để giao tiếp với đa dạng các loại thiết bị.
- Thu thập và lưu trữ dữ liệu (Data Collection and Storage): Cung cấp khả năng thu thập dữ liệu từ thiết bị một cách an toàn và lưu trữ nó một cách hiệu quả để phân tích sau này.
- Xử lý và phân tích dữ liệu (Data Processing and Analytics): Cung cấp các công cụ để xử lý dữ liệu thời gian thực, thực hiện phân tích, phát hiện các mẫu và tạo ra thông tin chi tiết hữu ích.
- Trực quan hóa dữ liệu (Data Visualization): Cho phép hiển thị dữ liệu IoT một cách trực quan thông qua bảng điều khiển (dashboards), biểu đồ và các hình thức khác để người dùng dễ dàng hiểu và theo dõi.
- Bảo mật (Security): Cung cấp các cơ chế bảo mật để bảo vệ thiết bị, dữ liệu và toàn bộ hệ thống IoT khỏi các mối đe dọa.
- Khả năng mở rộng (Scalability): Có khả năng xử lý số lượng lớn thiết bị và lượng dữ liệu ngày càng tăng khi quy mô triển khai IoT mở rộng.
- Tích hợp ứng dụng (Application Integration): Cung cấp các API và công cụ để tích hợp dữ liệu và chức năng IoT với các ứng dụng và hệ thống doanh nghiệp khác.

- *Lợi ích của việc sử dụng nền tảng IoT:*

- Giảm độ phức tạp: Nền tảng IoT giúp đơn giản hóa quá trình phát triển và quản lý các giải pháp IoT.
- Tăng tốc thời gian đưa sản phẩm ra thị trường: Các công cụ và dịch vụ có sẵn giúp rút ngắn thời gian phát triển ứng dụng.
- Cải thiện khả năng mở rộng: Dễ dàng kết nối và quản lý số lượng lớn thiết bị.
- Nâng cao hiệu quả hoạt động: Dữ liệu thời gian thực giúp đưa ra quyết định nhanh chóng và tối ưu hóa quy trình.
- Cải thiện trải nghiệm khách hàng: Dữ liệu từ thiết bị có thể được sử dụng để cá nhân hóa dịch vụ và hỗ trợ khách hàng tốt hơn.
- Tạo ra các mô hình kinh doanh mới: Dữ liệu và khả năng kết nối mở ra những cơ hội kinh doanh sáng tạo.

2.2.2 Các nền tảng phát triển phần mềm cho thiết bị IoT

- *Nền tảng dựa trên hệ điều hành thời gian thực (RTOS):*

RTOS được thiết kế để cung cấp khả năng phản hồi nhanh chóng và đáng tin cậy cho các ứng dụng thời gian thực, rất phù hợp cho các thiết bị IoT có yêu cầu về độ trễ thấp.

- FreeRTOS: Một RTOS mã nguồn mở phổ biến, nhỏ gọn và linh hoạt, hỗ trợ nhiều kiến trúc vi điều khiển. Nó có một cộng đồng lớn và nhiều thư viện hỗ trợ.
- Zephyr Project: Một RTOS mã nguồn mở khác được quản lý bởi Linux Foundation, tập trung vào bảo mật và kết nối. Nó hỗ trợ nhiều kiến trúc và có các tính năng hiện đại.
- Mbed OS: Một nền tảng phát triển miễn phí được Arm phát triển, cung cấp các thư viện, công cụ và dịch vụ đám mây để phát triển các ứng dụng IoT dựa trên vi điều khiển Arm.
- Contiki-NG: Một hệ điều hành mã nguồn mở tập trung vào các thiết bị có nguồn tài nguyên hạn chế và mạng năng lượng thấp, hỗ trợ các giao thức như IPv6 và CoAP.
- RIOT OS: Một hệ điều hành mã nguồn mở thân thiện với người dùng, dựa trên kiến trúc microkernel, hỗ trợ nhiều kiến trúc và giao thức.

- *Nền tảng dựa trên Linux nhúng (Embedded Linux):*

Linux nhúng cung cấp một môi trường phát triển mạnh mẽ với nhiều thư viện và công cụ có sẵn, phù hợp cho các thiết bị IoT có tài nguyên phần cứng mạnh hơn.

- Yocto Project: Một dự án mã nguồn mở cho phép bạn xây dựng các bản phân phối Linux tùy chỉnh cho các thiết bị nhúng, cung cấp sự linh hoạt cao trong việc lựa chọn các thành phần phần mềm.
- Buildroot: Một công cụ đơn giản hơn Yocto để xây dựng hệ thống Linux nhúng, tập trung vào sự dễ sử dụng và tốc độ xây dựng.

- Ubuntu Core: Một phiên bản thu nhỏ của Ubuntu được thiết kế cho các thiết bị IoT và robot, tập trung vào bảo mật và cập nhật qua mạng (OTA) bằng cách sử dụng Snap.
- Raspberry Pi OS (trước đây là Raspbian): Hệ điều hành chính thức cho Raspberry Pi, dựa trên Debian, rất phổ biến cho các dự án IoT nguyên mẫu và cả sản phẩm thương mại.

- *Nền tảng phát triển đa nền tảng:*

Các nền tảng này cho phép bạn viết mã một lần và triển khai trên nhiều loại thiết bị khác nhau.

- Node.js: Một môi trường JavaScript runtime được xây dựng trên Chrome's V8 JavaScript engine. Nó nhẹ, hiệu quả và có một hệ sinh thái thư viện phong phú (npm), thường được sử dụng cho cả backend và phát triển ứng dụng IoT trên các thiết bị có khả năng chạy JavaScript.
- Python: Một ngôn ngữ lập trình thông dịch, dễ học và có nhiều thư viện cho IoT, khoa học dữ liệu và học máy. MicroPython là một phiên bản nhỏ gọn của Python được thiết kế cho các vi điều khiển.
- Java: Một ngôn ngữ lập trình mạnh mẽ và có tính di động cao, thường được sử dụng trong các hệ thống nhúng phức tạp hơn.
- .NET (C#): Với .NET IoT Libraries, bạn có thể phát triển ứng dụng IoT trên các thiết bị chạy .NET Micro Framework hoặc các nền tảng khác hỗ trợ .NET.

- *Các framework và thư viện cụ thể cho IoT:*

- IoTivity: Một framework mã nguồn mở được tài trợ bởi Open Connectivity Foundation (OCF), nhằm mục đích tạo ra một tiêu chuẩn kết nối cho các thiết bị IoT.
- Home Assistant: Một nền tảng tự động hóa nhà mã nguồn mở, cho phép tích hợp và điều khiển nhiều loại thiết bị IoT khác nhau. Mặc dù chủ yếu hướng đến tự động hóa nhà, nó cũng cung cấp một nền tảng để phát triển các ứng dụng IoT tùy chỉnh.
- WiredTiger: Một database engine hiệu suất cao, thường được sử dụng trong các ứng dụng IoT để lưu trữ và quản lý dữ liệu cục bộ trên thiết bị.

- *Các công cụ phát triển (IDEs và SDKs):*

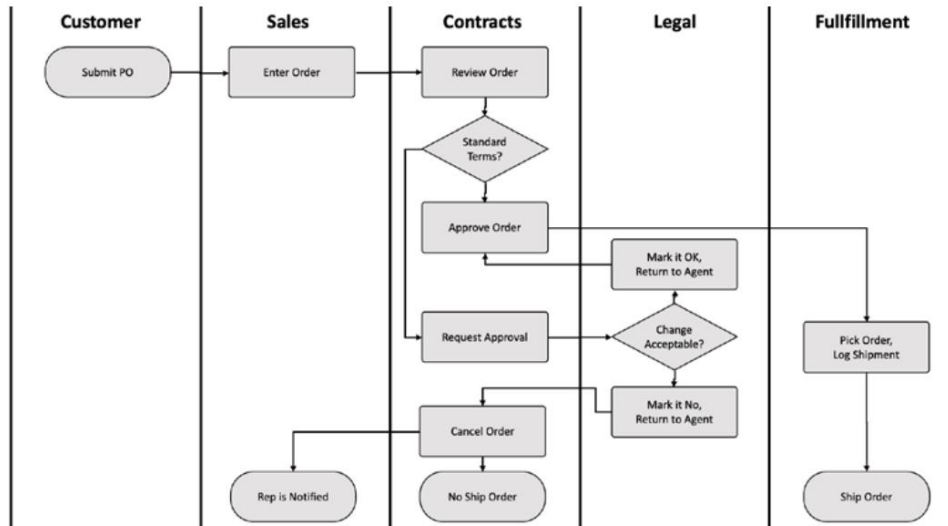
- Visual Studio Code (VS Code): Một trình soạn thảo mã nguồn mở phổ biến với nhiều extension hỗ trợ phát triển cho các nền tảng IoT khác nhau.
- Eclipse IDE: Một IDE mã nguồn mở mạnh mẽ, thường được sử dụng cho phát triển nhúng và Java, với nhiều plugin hỗ trợ IoT.
- Arduino IDE: Một IDE đơn giản và dễ sử dụng, đặc biệt phù hợp cho người mới bắt đầu với Arduino.
- PlatformIO: Một hệ sinh thái phát triển mã nguồn mở cho các hệ thống nhúng, hỗ trợ nhiều bo mạch và framework.

- Các SDK (Software Development Kits) cụ thể của nhà sản xuất chip và module: Các nhà sản xuất như Espressif (ESP32), STMicroelectronics, NXP thường cung cấp các SDK riêng để phát triển phần mềm cho chip của họ.

2.3. Nền tảng phần mềm Node-Red

2.3.1 Tổng quan và cài đặt

2.3.1.1 Tổng quan



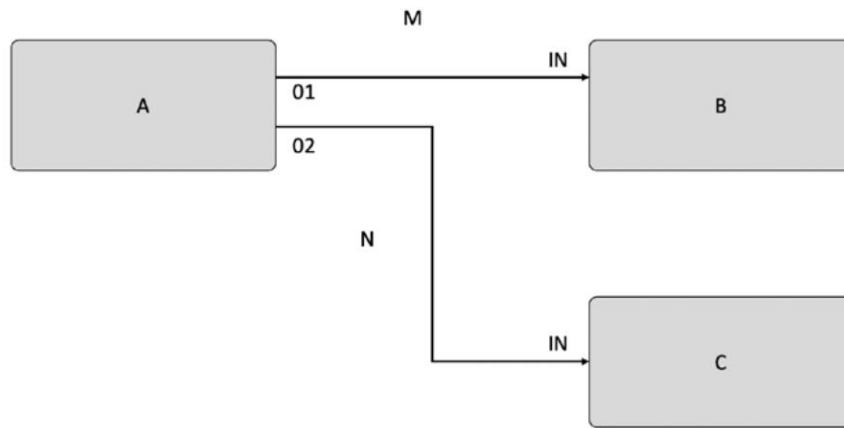
Hình 2.6 Ví dụ về quy trình làm việc

FBP (Flow-Based Programming) là một khái niệm trong lập trình phần mềm, định nghĩa một ứng dụng thông qua luồng dữ liệu. Mỗi phần của quy trình được coi như một “hộp đen”. Các hộp đen này trao đổi dữ liệu với nhau thông qua các kết nối đã được xác định trước.

FBP được xem là hướng thành phần (component-oriented) bởi vì các quy trình dạng hộp đen này có thể được kết nối lại nhiều lần để tạo ra nhiều ứng dụng khác nhau mà không cần sửa đổi bên trong chúng.

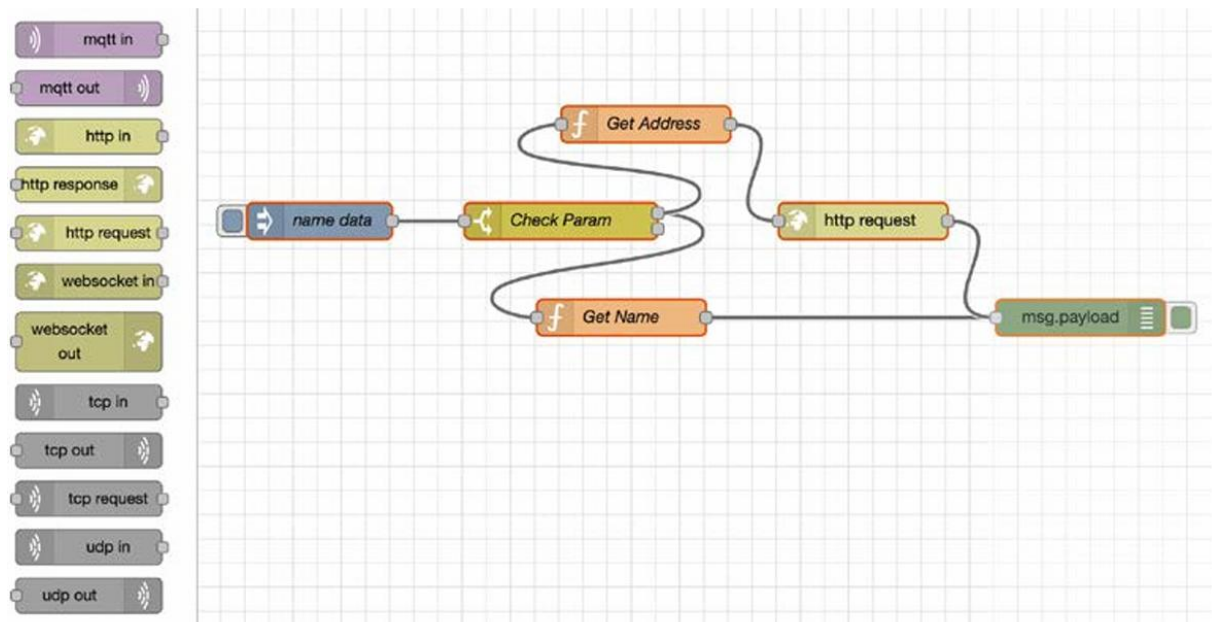
FBP mang phong cách hàm bậc cao (high-level, functional), giúp định nghĩa hành vi hệ thống một cách rõ ràng. Điều này đặc biệt hữu ích trong các mô hình phức tạp như giao thức phân tán đa bên (distributed multi-party protocols) hoặc mô hình luồng dữ liệu phân tán, nơi ta cần phân tích chính xác điều kiện để biết một biến hoặc lệnh có hoạt động đúng hay không.

Dưới đây là sơ đồ đơn giản về FBP.



Hình 2.7 Ví dụ về thiết kế FBP đơn giản

Node-RED là một trong những công cụ FBP mà người dùng đã đề cập. Được phát triển bởi nhóm Emerging Technology Services của IBM, Node-RED hiện đang trực thuộc OpenJS Foundation.



Hình 2.8 Node-RED Flow Editor là một công cụ FBP

Node-RED không chỉ là công cụ lập trình mà còn là một nền tảng thực thi, tích hợp sẵn môi trường chạy Node.js cho các ứng dụng được xây dựng bằng Node-RED. Người dùng cần sử dụng flow editor để tạo các ứng dụng Node-RED cho IoT, dịch vụ web và nhiều mục đích khác. Flow editor cũng là một ứng dụng web Node.js.

Trình soạn thảo luồng—chức năng cốt lõi của Node-RED—thực chất là một ứng dụng web được xây dựng bằng Node.js. Nó hoạt động trên môi trường chạy Node.js và vận hành ngay trong trình duyệt. Người dùng chỉ cần chọn node muốn dùng từ thanh palette, kéo thả vào workspace. Việc “wiring” (nối dây) chính là kết nối các node với nhau để tạo thành một ứng dụng. Người dùng (lập trình viên) có thể triển khai (deploy) ứng dụng lên môi trường chạy chỉ với một cú nhấp chuột.

Thanh palette chứa nhiều node khác nhau có thể dễ dàng mở rộng bằng cách cài thêm các node do cộng đồng phát triển, cho phép mọi người chia sẻ flow mình tạo ra dưới dạng file JSON một cách thuận tiện. Trước khi đi sâu vào lợi ích của Node-RED, hãy cùng xem qua lịch sử hình thành của nó.

****Lợi ích của Node-RED***

- Giản lược hóa (Simplicity):

No-code/Low-code: Gần như không cần viết mã, mọi thao tác đều thực hiện thông qua giao diện kéo-thả.

Trực quan: Mô hình lập trình dạng luồng (flow) giúp dễ dàng hình dung toàn bộ quá trình xử lý.

- Tăng hiệu suất (Efficiency):

Tập trung vào logic nghiệp vụ: Flow editor tự động thiết lập môi trường chạy, đồng bộ thư viện và cung cấp IDE tích hợp, cho phép người dùng tập trung hoàn toàn vào phát triển.

Chạy song song: Các tiến trình bất đồng bộ trong Node-RED vận hành liên tục khi còn dữ liệu đầu vào/ra, tận dụng tối đa tài nguyên đa lõi của máy chủ.

- Tái sử dụng thành phần (Reusability):

Component-oriented: Mỗi node là một “hộp đen” đã được đóng gói (module) và có thể dùng chung ở nhiều dự án khác nhau.

Mạng lưới linh hoạt: Nếu chưa có node phù hợp, người dùng dễ dàng tự tạo và chia sẻ dưới dạng gói NPM hoặc file JSON.

- Đảm bảo chất lượng cao (High Quality):

Đã kiểm thử đơn vị: Hầu hết node do cộng đồng và IBM phát hành đều được unit test kỹ càng.

Giảm lỗi lập trình: Tác giả ứng dụng chỉ cần kiểm tra luồng dữ liệu giữa các node mà không cần quan tâm đến chi tiết nội tại, hạn chế sai sót ở mức chi tiết.

- Mã nguồn mở (Open Source):

Giấy phép Apache 2: Cho phép sử dụng linh hoạt trong môi trường cá nhân và thương mại.

Cộng đồng phát triển rộng lớn: Node-RED là dự án của OpenJS Foundation, được duy trì và mở rộng bởi hơn 30 dự án JavaScript hàng đầu.

****Các nền tảng triển khai:***

Edge devices: Raspberry Pi, các board IoT tích hợp sẵn Node.js, giúp xử lý dữ liệu ngay tại thiết bị.

Dịch vụ đám mây: AWS, Azure, IBM Cloud, Google Cloud... cho phép vận hành quy mô lớn, tích hợp dễ dàng với các dịch vụ khác.

Hệ thống nhúng: các thiết bị hoặc gateway nhúng, hỗ trợ đóng gói gọn nhẹ, đáp ứng yêu cầu tài nguyên hạn chế.

Việc có thể khởi chạy trên nhiều nền tảng giúp Node-RED trở thành công cụ linh hoạt, từ phát triển thử nghiệm cho đến triển khai sản xuất.

2.3.1.2 Cài đặt

Yêu cầu kỹ thuật

Người dùng sẽ cần cài đặt những thứ sau cho chương này:

- Node.js (v12.18.1)*
- npm (v6.14.5)*

*Phiên bản LTS vào thời điểm viết bài cho cả hai.

Cài đặt npm và Node.js cho Windows

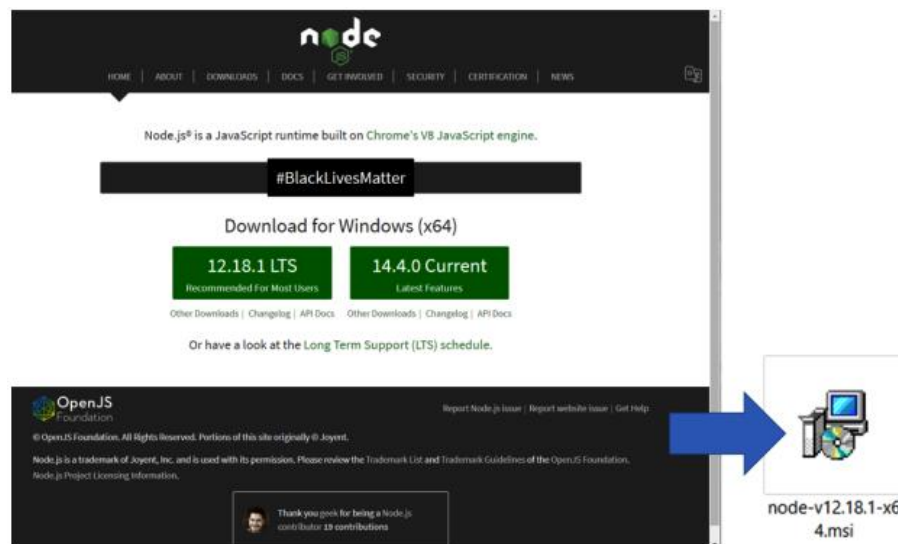
Nếu người dùng muốn sử dụng Node-RED trên Windows, người dùng cần cài đặt npm và Node.js qua trang web sau:

<https://nodejs.org/en/#home-downloadhead>.

Người dùng có thể tải trình cài đặt Node.js cho Windows trực tiếp từ đó. Sau đó, làm theo các bước sau:

Truy cập trang web gốc của Node.js và tải về trình cài đặt.

Người dùng có thể chọn cả hai phiên bản – Phiên bản Được Khuyến Nghị hoặc Phiên bản Các Tính Năng Mới Nhất – nhưng trong sách này, người dùng nên sử dụng phiên bản Được Khuyến Nghị:

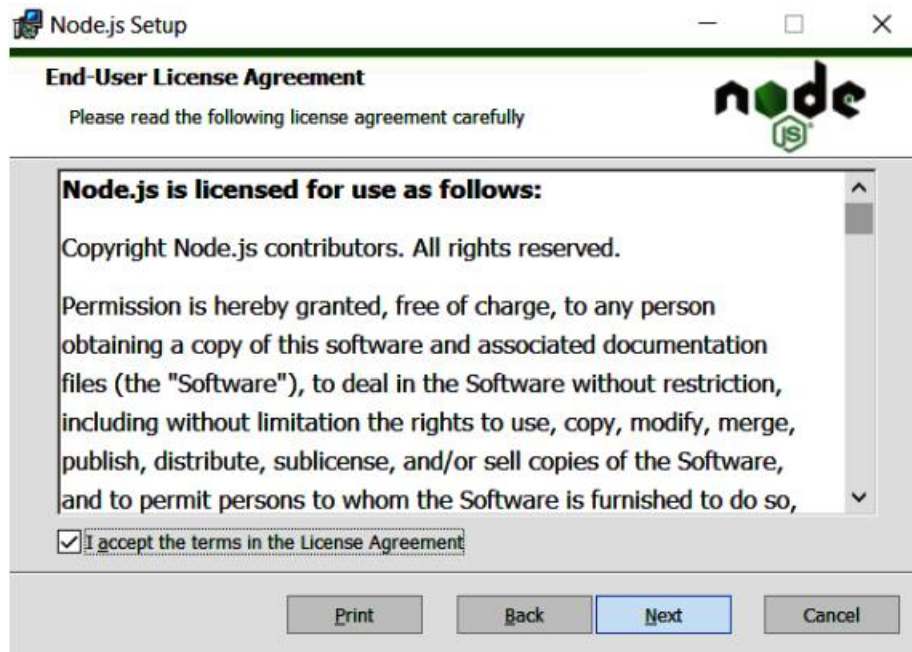


Hình 2.9 Chọn phiên bản được đề xuất

Nhấp vào file .msi mà người dùng đã tải về để bắt đầu cài đặt Node.js. Nó bao gồm phiên bản npm hiện tại. Node-RED chạy trên nền tảng Node.js runtime, vì vậy việc cài đặt Node.js là cần thiết.

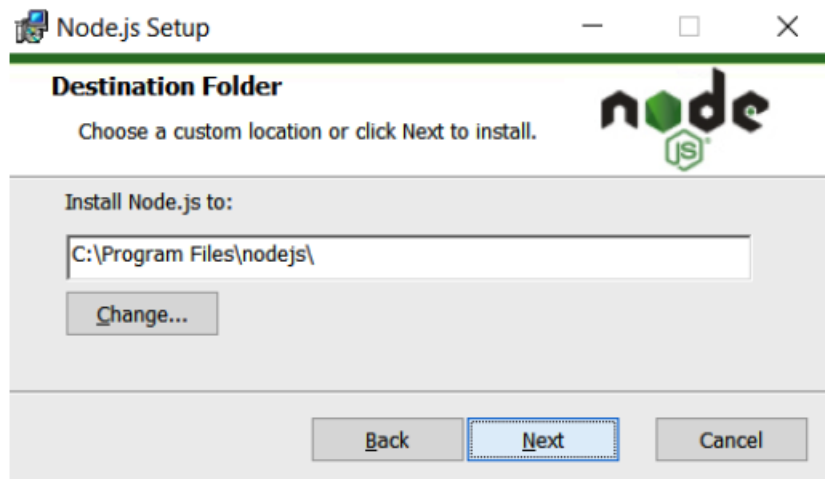
Chỉ cần nhấn vào các nút trong cửa sổ hướng dẫn cài đặt theo trình tự, mặc dù có một số điểm cần lưu ý trong quá trình cài đặt.

Tiếp theo, người dùng cần chấp nhận Thỏa thuận Cấp phép Người dùng Cuối (End-User License Agreement).



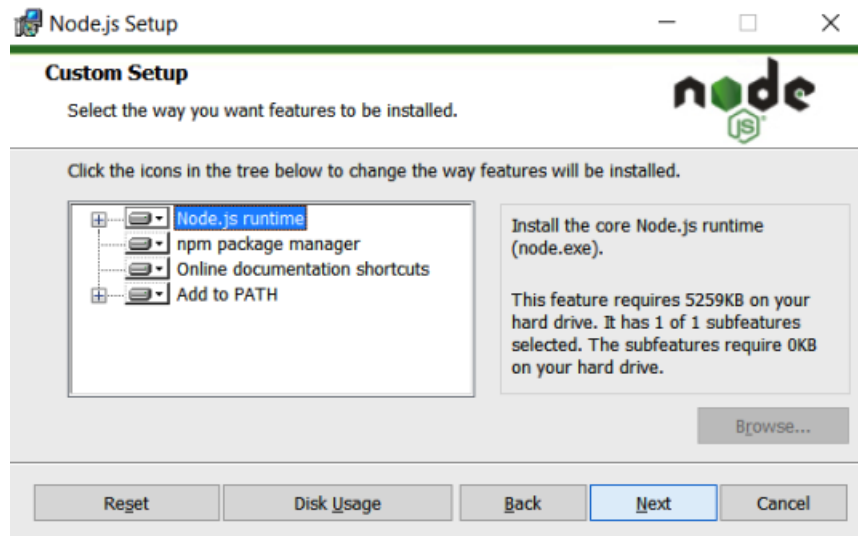
Hình 2.10 Thỏa thuận cấp phép cho người dùng

Người dùng cũng có thể thay đổi thư mục cài đặt. Trong sách này, thư mục mặc định (C:/Program Files/nodejs/) sẽ được sử dụng.



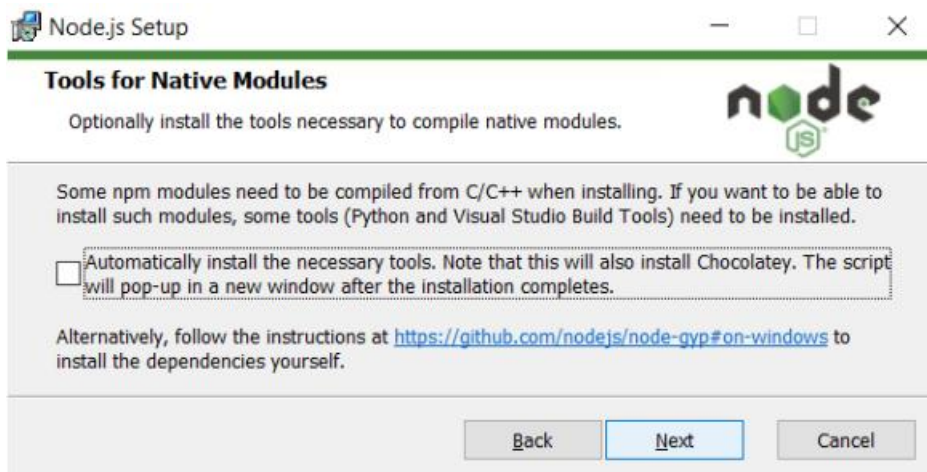
Hình 2.11 Cài đặt thư mục đích

Không cần thiết lập tùy chỉnh ở màn hình tiếp theo. Người dùng chỉ cần nhấn Next với các tính năng mặc định đã được chọn sẵn.



Hình 2.12 Không cần thiết lập tùy chỉnh

Ở màn hình tiếp theo, người dùng có thể nhấn Next mà không cần chọn thêm bất cứ tùy chọn nào. Tuy nhiên, người dùng cũng có thể cài đặt các công cụ được liệt kê ở đây nếu muốn. Các công cụ này bao gồm việc cài đặt và thiết lập đường dẫn cho các môi trường như Visual C++, windows-build-tools và Python



Hình 2.13 Cửa sổ công cụ cho modules gốc

Sau khi quá trình cài đặt Node.js hoàn tất, hãy kiểm tra phiên bản của các công cụ bằng cách sử dụng các lệnh sau:

```
$ node --version
v12.18.1

$ npm --version
6.14.5
```

2.3.2 Các thành phần cơ bản và quy tắc lập trình

Trong chương này, người dùng sẽ thực sự tạo một luồng sử dụng Node-RED Flow Editor. Bằng cách tạo một luồng đơn giản, người dùng sẽ hiểu cách sử dụng công cụ và các đặc điểm của nó. Để hiểu rõ hơn, người dùng sẽ tạo một số luồng mẫu.

Từ bây giờ, người dùng sẽ tạo các ứng dụng gọi là "flows" sử dụng Node-RED. Trong chương này, người dùng sẽ học cách sử dụng Node-RED và cách tạo một ứng dụng dưới dạng luồng. Để làm điều này, người dùng sẽ đề cập đến các chủ đề sau:

- Cơ chế của Node-RED Flow Editor
- Sử dụng Flow Editor
- Tạo một luồng cho ứng dụng xử lý dữ liệu
- Tạo một luồng cho ứng dụng web
- Nhập khẩu và xuất khẩu định nghĩa luồng

Vào cuối chương này, người dùng sẽ thành thạo cách sử dụng Node-RED Flow Editor và biết cách xây dựng một ứng dụng đơn giản với nó.

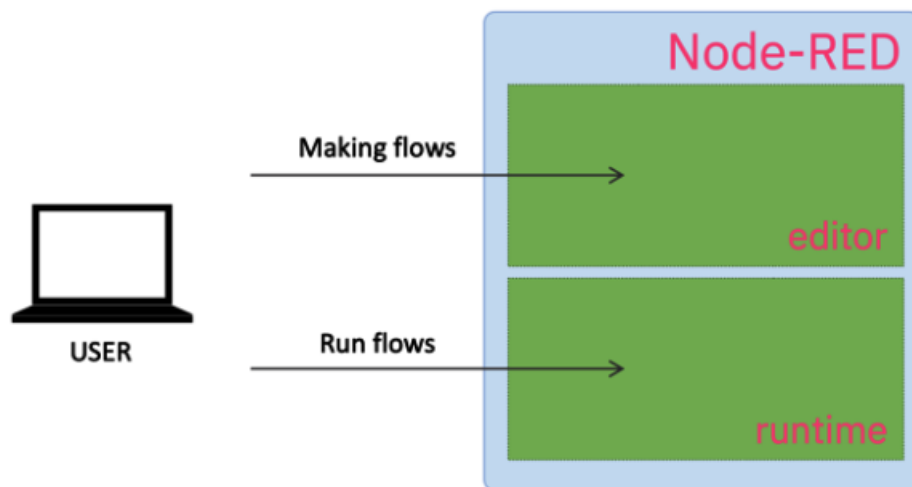
Cơ chế của Node-RED Flow Editor:

Node-RED có hai phần logic chính: môi trường phát triển gọi là Flow Editor và môi trường thực thi để thực thi ứng dụng đã được tạo ra ở đó. Những phần này lần lượt được gọi là *runtime* và *editor*. Hãy cùng xem chi tiết hơn về chúng:

Runtime: Bao gồm một môi trường thực thi ứng dụng Node.js. Nó chịu trách nhiệm chạy các luồng đã được triển khai.

Editor: Đây là một ứng dụng web nơi người dùng có thể chỉnh sửa các luồng của mình.

Gói cài đặt chính bao gồm cả hai thành phần này, với một máy chủ web cung cấp Flow Editor cũng như một REST Admin API để quản lý runtime. Trong nội bộ, các thành phần này có thể được cài đặt riêng biệt và tích hợp vào các ứng dụng Node.js hiện có, như được thể hiện trong sơ đồ dưới đây:



Hình 2.14 Sử dụng Flow Editor

Các tính năng chính của Flow Editor như sau:

Node: Là thành phần chính của các ứng dụng Node-RED, các node đại diện cho các phần chức năng rõ ràng.

Flow: Là một chuỗi các node được kết nối với nhau, đại diện cho chuỗi các bước mà các tin nhắn đi qua trong một ứng dụng.

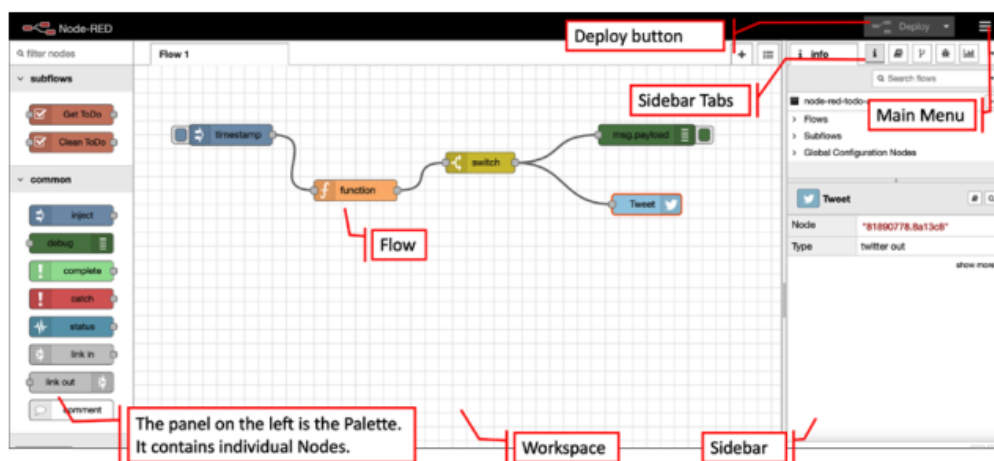
Palette (Bảng điều khiển bên trái): Là bộ sưu tập các node có sẵn trong editor mà người dùng có thể sử dụng để xây dựng ứng dụng của mình.

Deploy button (Nút triển khai): Nhấn nút này để triển khai ứng dụng của người dùng sau khi người dùng đã chỉnh sửa xong.

Sidebar (Thanh công cụ bên): Là một bảng điều khiển hiển thị các chức năng khác nhau, chẳng hạn như cài đặt tham số xử lý, thông số kỹ thuật và hiển thị trình gỡ lỗi.

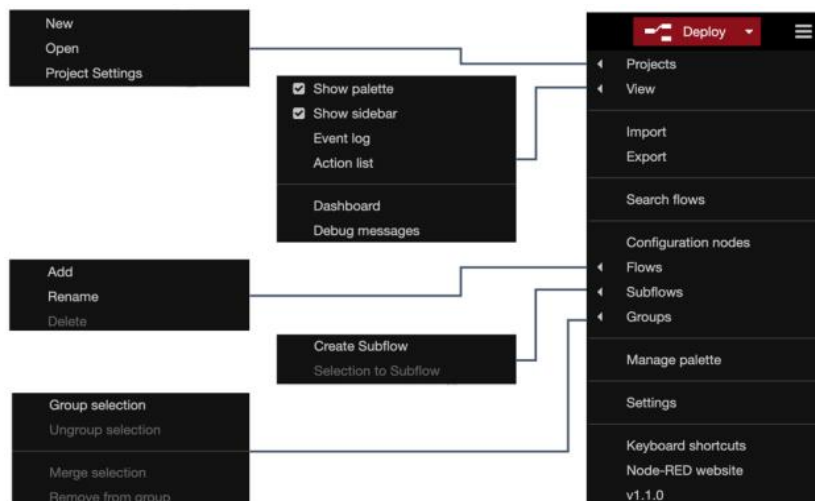
Sidebar tabs (Thẻ của thanh công cụ bên): Cài đặt cho mỗi node, đầu ra tiêu chuẩn, quản lý thay đổi, v.v.

Main menu (Menu chính): Xóa luồng, nhập/xuất định nghĩa, quản lý dự án, v.v



Hình 2.15 Trình biên tập luồng Node-RED

Người dùng cần hiểu những gì có trong menu Flow trước khi bắt đầu sử dụng Node-RED. Nội dung của menu này có thể khác nhau tùy thuộc vào phiên bản Node-RED mà người dùng đang sử dụng, nhưng nó có một số mục cài đặt chung như quản lý dự án của luồng, sắp xếp chế độ xem, nhập/xuất luồng, cài đặt các node đã được công bố trong thư viện, và các mục khác.



Hình 2.16 Danh sách luồng biên tập Node-RED

Cài đặt Môi trường Phát triển, để học cách thiết lập Node-RED với môi trường của người dùng, chẳng hạn như Windows, Mac, hoặc Raspberry Pi, nếu người dùng chưa thực hiện điều này.

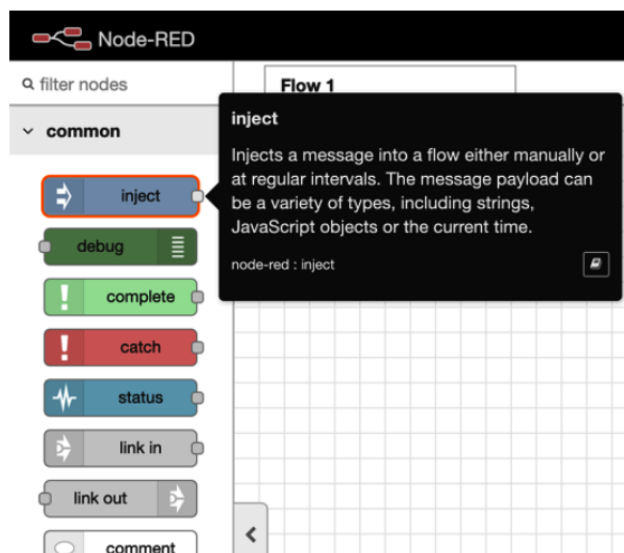
Khi Node-RED đã được chạy, hãy chuyển sang phần tiếp theo, nơi người dùng sẽ tạo luồng đầu tiên của mình.

Tạo một luồng cho ứng dụng xử lý dữ liệu

Trong phần này, người dùng sẽ tạo một ứng dụng hoạt động (gọi là luồng trong Node-RED). Dù là Internet of Things (IoT) hay xử lý máy chủ dưới dạng ứng dụng web, hoạt động cơ bản mà Node-RED thực hiện là chuyển dữ liệu theo chuỗi.

Ở đây, người dùng sẽ tạo một luồng, trong đó dữ liệu JSON được tạo ra một cách giả lập, và dữ liệu này sẽ được xuất ra đầu ra tiêu chuẩn thông qua một số node trong Node-RED.

Có rất nhiều node ở phía bên trái của palette. Hãy chú ý đến các thể loại chung ở đây. Người dùng sẽ dễ dàng tìm thấy node *inject*, như được hiển thị trong ảnh chụp màn hình dưới đây:

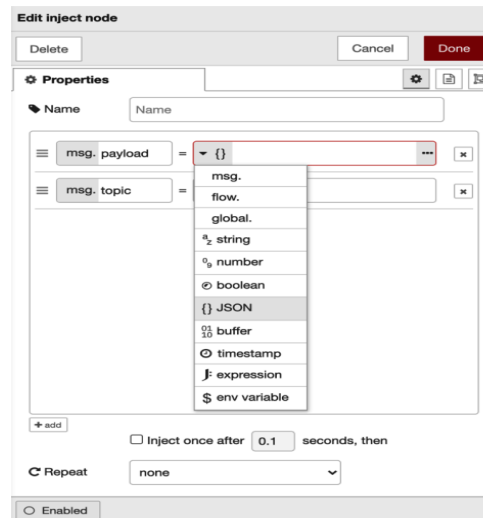


Hình 2.17 Node Inject

Node này có thể tiêm một tin nhắn vào node tiếp theo. Hãy bắt đầu:

Kéo và thả node này vào palette của Flow 1 (tab luồng mặc định). Người dùng sẽ thấy node này được gắn nhãn với từ timestamp. Điều này là do payload tin nhắn mặc định của nó là giá trị timestamp. Người dùng có thể thay đổi kiểu dữ liệu, vì vậy hãy thay đổi nó thành kiểu JSON.

Nhấp đúp vào node và thay đổi cài đặt của nó khi bảng điều khiển Properties của node mở ra.



Hình 2.18 Thiết lập thuộc tính nút inject

Nhấp vào menu thả xuống của tham số đầu tiên và chọn {} JSON. Người dùng có thể chỉnh sửa dữ liệu JSON bằng cách nhấp vào nút [...] ở phía bên phải.

Nhấp vào nút [...], trình chỉnh sửa JSON sẽ mở ra. Người dùng có thể tạo dữ liệu JSON bằng trình chỉnh sửa dạng văn bản hoặc trình chỉnh sửa trực quan.

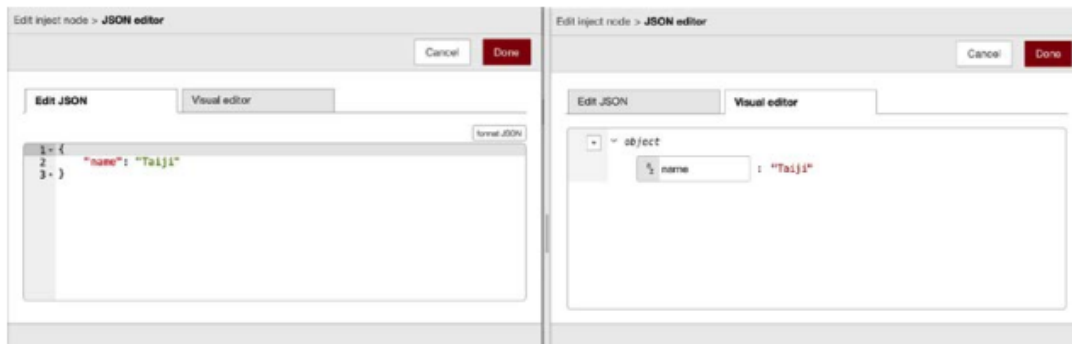
Lần này, hãy tạo dữ liệu JSON với một mục như sau:

json

Sao chép mã

```
{"name": "Taiji"}
```

Người dùng nên thay thế tên Taiji bằng tên của người dùng.

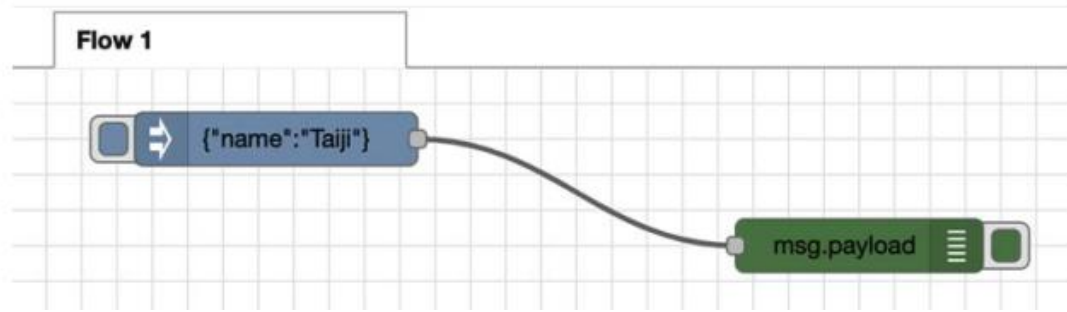


Hình 2.19 Nhấn nút Done để đóng bảng cài đặt.

Tương tự, hãy kéo một Debug node vào palette.

Sau khi đặt nó vào, hãy kết nối (wire) node Inject và node Debug lại với nhau.

Khi người dùng thực thi luồng này, dữ liệu JSON được gửi từ node Inject sẽ được xuất ra bảng điều khiển debug (standard output) thông qua node Debug. Người dùng không cần cấu hình gì thêm cho node Debug.



Hình 2.20 Đặt nút Debug và đầu dây

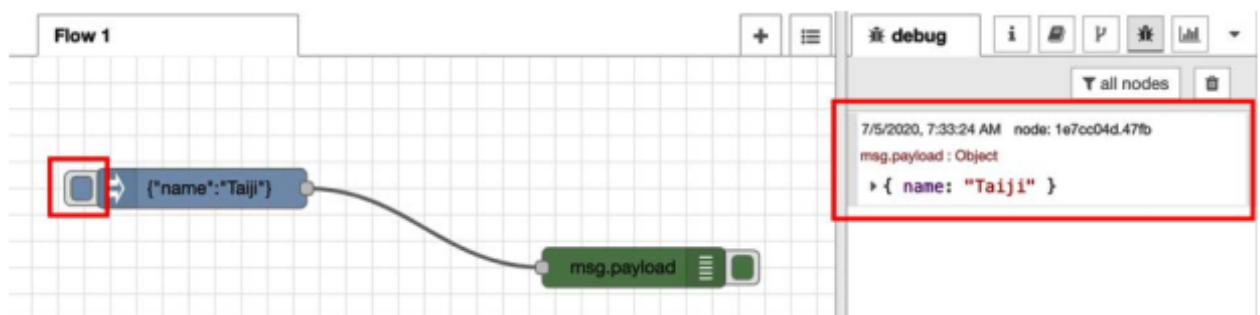
Cuối cùng, người dùng cần triển khai (deploy) luồng mà người dùng đã tạo. Trong Node-RED Flow Editor, người dùng có thể triển khai tất cả các luồng trong không gian làm việc của mình lên runtime của Node-RED bằng cách nhấp vào nút Deploy ở góc trên bên phải.

Trước khi chạy luồng, người dùng nên chọn tab Debug từ bảng điều khiển bên phải của menu node để bật bảng điều khiển debug, như được minh họa trong ảnh chụp màn hình sau:



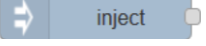
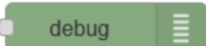

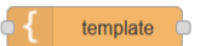
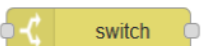
Hình 2.21 Bật bảng điều khiển Debug







Hãy chạy luồng này. Nhấp vào công tắc (nút hình ô tròn) của node Inject để xem kết quả thực thi luồng trên bảng điều khiển debug



Hình 2.22 Thực hiện luồng và kiểm tra kết quả

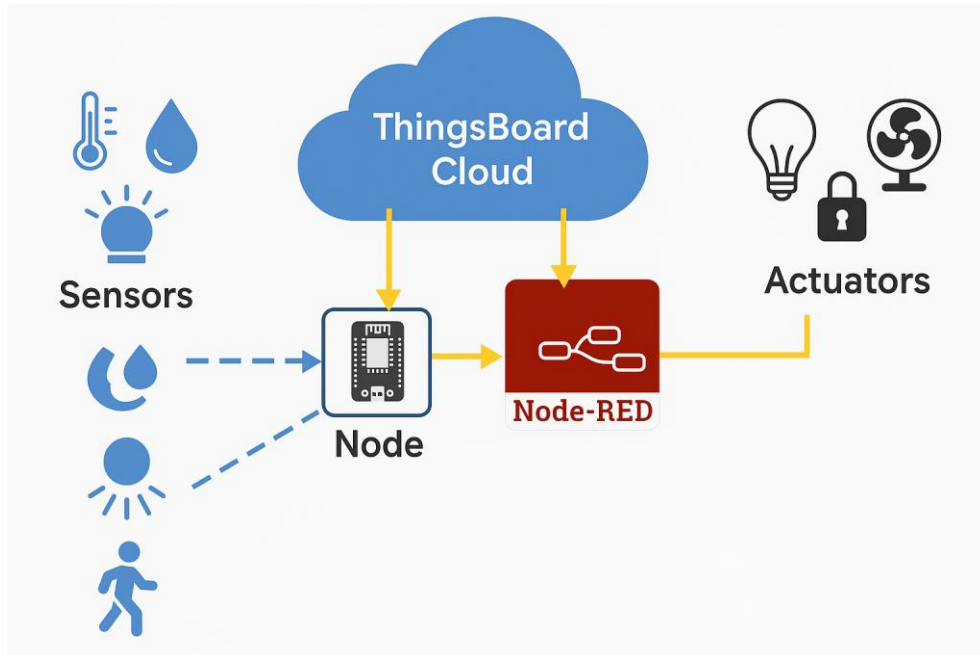
Các node chức năng:

	<p>Inject (Tiêm dữ liệu) một thông điệp vào luồng, có thể thực hiện thủ công hoặc theo các khoảng thời gian định kỳ. Payload (dữ liệu tải) của thông điệp có thể ở nhiều dạng khác nhau, bao gồm chuỗi (string), đối tượng JavaScript hoặc thời gian hiện tại. Node Inject có thể khởi tạo một luồng với một giá trị payload cụ thể. Node này cũng hỗ trợ việc gửi vào các giá trị như dấu thời gian (timestamp), chuỗi, số, kiểu boolean, đối tượng JavaScript, hoặc các giá trị từ ngữ cảnh của flow/global (ngữ cảnh cục bộ hoặc toàn cục).</p> <p>Theo mặc định, node được kích hoạt thủ công bằng cách nhấn nút trong trình chỉnh sửa. Nó cũng có thể được thiết lập để tự động gửi dữ liệu theo khoảng thời gian định kỳ hoặc theo lịch trình định sẵn.</p>
	<p>Hiển thị các thuộc tính của thông điệp đã chọn trong tab debug (gỡ lỗi) ở thanh bên và tùy chọn ghi vào log của môi trường runtime. Mặc định, nó hiển thị msg.payload, nhưng có thể được cấu hình để hiển thị bất kỳ thuộc tính nào, toàn bộ thông điệp, hoặc kết quả của một biểu thức JSONata.</p> <p>Thanh bên debug cung cấp cái nhìn có cấu trúc về các thông điệp mà nó nhận được, giúp dễ dàng hiểu được cấu trúc của chúng. Các đối tượng và mảng JavaScript có thể được thu gọn hoặc mở rộng khi cần thiết.</p> <p>Bên cạnh mỗi thông điệp, thanh bên debug cũng hiển thị thông tin về thời gian thông điệp được nhận, node đã gửi thông điệp đó, và loại của thông điệp.</p> <p>Nút trên node có thể được sử dụng để bật hoặc tắt việc hiển thị kết quả đầu ra.</p>
	<p>Một khối hàm JavaScript để chạy với các message (thông điệp) được nhận bởi node. Các message được truyền vào dưới dạng một đối tượng JavaScript có tên là msg. Theo quy ước, đối tượng này sẽ có một thuộc tính msg.payload chứa nội dung của message.</p> <p>Hàm này được kỳ vọng sẽ trả về một đối tượng message (hoặc nhiều đối tượng message), nhưng cũng có thể chọn không trả về gì để dừng dòng xử lý (halt a flow).</p>
	<p>Node Template có thể được sử dụng để tạo văn bản bằng cách sử dụng các thuộc tính của message để điền vào một mẫu. Nó sử dụng ngôn ngữ mẫu Mustache để tạo ra kết quả.</p> <p>Ví dụ, một mẫu sau:</p> <p>This is the payload: {{payload}} !</p> <p>Sẽ thay thế {{payload}} bằng giá trị của thuộc tính payload trong message</p>
	<p>Node Switch cho phép chuyển hướng các message đến các nhánh khác nhau của dòng xử lý bằng cách đánh giá một tập hợp các quy tắc đối với từng message. Node này được cấu hình với thuộc tính cần kiểm tra – có thể là thuộc tính của message hoặc thuộc tính trong context.</p>

 change	<p>Node Change có thể được sử dụng để sửa đổi các thuộc tính của message và thiết lập các thuộc tính context mà không cần phải sử dụng đến Function node. Mỗi node có thể được cấu hình với nhiều thao tác được áp dụng theo thứ tự. Các thao tác có sẵn bao gồm: Set (thiết lập), Change (thay đổi), Move (di chuyển), Delete (xóa).</p>
 http request	<p>Gửi các yêu cầu HTTP và trả về phản hồi.</p> <p>Các đầu vào chính:</p> <ul style="list-style-type: none"> • url (chuỗi): địa chỉ URL của yêu cầu. • method (chuỗi): phương thức HTTP của yêu cầu, ví dụ: GET, PUT, POST, PATCH hoặc DELETE. • headers (đối tượng): các tiêu đề (headers) HTTP của yêu cầu. • payload: nội dung (body) của yêu cầu. <p>Ở đầu ra, payload chứa nội dung (body) của phản hồi. Node có thể được cấu hình để trả về nội dung dưới dạng chuỗi, cố gắng phân tích cú pháp dưới dạng chuỗi JSON hoặc giữ nguyên dưới dạng buffer nhị phân</p>
 json	<p>Chuyển đổi giữa chuỗi JSON và biểu diễn đối tượng JavaScript của nó, theo cả hai chiều.</p>
 xml	<p>Chuyển đổi giữa chuỗi XML và biểu diễn đối tượng JavaScript của nó, theo cả hai chiều.</p>
 file	<p>Đọc nội dung của một tệp dưới dạng chuỗi hoặc buffer nhị phân.</p>
 file	<p>Ghi msg.payload vào một tệp, có thể là thêm vào cuối tệp hoặc ghi đè nội dung hiện có. Ngoài ra, node cũng có thể xóa tệp.</p>

CHƯƠNG 3. THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG NHÀ THÔNG MINH DỰA TRÊN NỀN TẢNG NODE-RED

3.1. Mô hình hệ thống nhà thông minh tích hợp IoT



Hình 3.1 Mô hình nhà thông minh dựa trên nền tảng Node-red

Hình trên mô tả tổng quan kiến trúc hệ thống nhà thông minh ứng dụng nền tảng Internet of Things (IoT) do nhóm nghiên cứu thiết kế và triển khai. Hệ thống được xây dựng với mục tiêu giám sát các thông số môi trường bên trong khu vực sinh hoạt và điều khiển các thiết bị điện trong nhà một cách tự động và từ xa, nhằm nâng cao tính an toàn, tiện nghi cũng như tối ưu hóa việc sử dụng năng lượng.

Trong hệ thống này, các cảm biến đóng vai trò thu thập dữ liệu từ môi trường bao gồm: nhiệt độ, độ ẩm không khí, nồng độ khí gas và chuyển động của con người. Các tín hiệu cảm biến được truyền trực tiếp về bộ xử lý trung tâm là vi điều khiển ESP32, một vi điều khiển tích hợp kết nối Wi-Fi có hiệu suất xử lý cao và chi phí thấp, rất phù hợp cho các ứng dụng IoT quy mô nhỏ đến trung bình.

Sau khi thu thập và xử lý dữ liệu tại nút ESP32, các giá trị đo được sẽ được gửi lên nền tảng ThingsBoard Cloud thông qua kết nối không dây Wi-Fi. ThingsBoard đóng vai trò là nền tảng giám sát và quản lý thiết bị IoT trên nền điện toán đám mây, cho phép lưu trữ, hiển thị dữ liệu theo thời gian thực và thiết lập các ngưỡng cảnh báo khi thông số vượt quá giá trị an toàn cho phép.

Song song với quá trình gửi dữ liệu lên đám mây, ESP32 còn truyền dữ liệu đến hệ thống Node-RED. Đây là một công cụ lập trình trực quan dạng luồng (flow-based programming) chuyên dụng cho các ứng dụng IoT và tự động hóa công nghiệp, cho

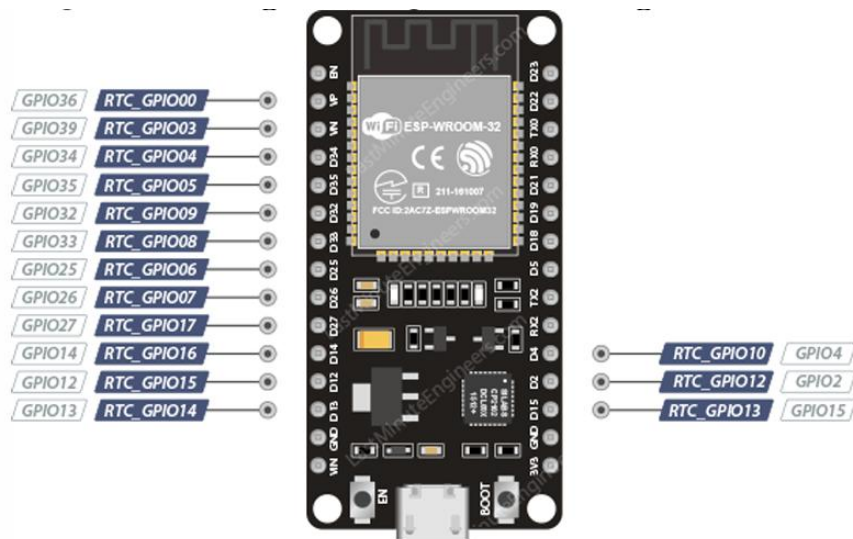
phép người dùng dễ dàng xây dựng các kịch bản điều khiển thiết bị hoặc các quy trình cảnh báo một cách linh hoạt và trực quan thông qua giao diện đồ họa.

Node-RED tiếp nhận dữ liệu từ ESP32 và thực hiện xử lý logic theo các kịch bản được lập trình sẵn. Khi phát hiện các điều kiện bất thường như nhiệt độ hoặc nồng độ khí vượt ngưỡng, hoặc có chuyển động bất thường trong khu vực giám sát, hệ thống sẽ ngay lập tức kích hoạt các thiết bị chấp hành như: bật/tắt đèn chiếu sáng, quạt thông gió, hoặc khóa điện tử, đồng thời gửi thông báo cảnh báo đến người dùng qua nền tảng ThingsBoard.

Việc kết hợp giữa ThingsBoard Cloud và Node-RED giúp hệ thống vừa đảm bảo khả năng giám sát và lưu trữ dữ liệu dài hạn trên nền tảng đám mây, vừa cho phép xử lý nhanh các tình huống và điều khiển thiết bị ngay tại mạng cục bộ, đảm bảo tính liên tục, độ trễ thấp và an toàn cho hệ thống nhà thông minh. Mô hình này có thể dễ dàng mở rộng bằng cách tích hợp thêm các cảm biến và thiết bị chấp hành khác tùy theo nhu cầu thực tế của người sử dụng.

3.2. Thiết kế các thành phần thiết bị trong hệ thống

3.2.1. Module ESP32



Hình 3.2 Bố trí các chân GPIO RTC

Các thông số kỹ thuật của ESP 32:

Wi-Fi

- 802.11b/g/n
- 802.11n(2.4GHz), up to 150 Mbps
- WMM
- TX/RX A-MPDU, RX A-MSDU
- Immediate block ACK - Defragmentation 30

- Giám sát đèn hiệu tự động (hardware TSF)
 - 4 × virtual Wi-Fi interfaces
 - Hỗ trợ đồng thời cho các chế độ Trạm cơ sở hạ tầng, softAP và Promiscuous
- Lưu ý khi ESP32 ở chế độ Station, thực hiện quét thì kênh softAP sẽ bị thay đổi.
- Ăng-ten đa dạng

Bluetooth

- Tuân thủ thông số kỹ thuật Bluetooth v4.2 BR/EDR và bluetooth LE
- Class-1, class-2 and class-3 không có bộ khuếch đại công suất bên ngoài
- Kiểm soát năng lượng nâng cao
- Công suất +9 dBm
- Bộ thu NZIF có độ nhạy Bluetooth LE –94 dBm
- Adaptive frequency hopping (AFH)
- HCI tiêu chuẩn dựa trên SDIO/SPI/UART
- UART HCI tốc độ cao, lên tới 4 Mbps
- Bộ điều khiển chế độ kép bluetooth 4.2 BR/EDR và bluetooth LE
- Định hướng kết nối đồng bộ/Mở rộng (SCO/eSCO)
- CVSD và âm thanh SBC - Bluetooth piconet and scatternet
- Đa kết nối trong bluetooth classic và bluetooth LE
- Simultaneous advertising and scanning

CPU và Bộ nhớ

- Xtensa® Bộ xử lý lõi đơn / lõi kép 32 bit LX6 với tối đa 600 DMIPS
- 448 KB ROM
- 520 KB SRAM
- SRAM 16 KB trong RTC
- QSPI có thể kết nối tối đa 4 Flash / SRAM, tối đa 16 MB mỗi flash
- Điện áp cung cấp: 2.2V đến 3.6V
- Dòng điện làm việc: trung bình: 80 mA
- Kích thước gói hàng: 18 mm x 25,5 mm x 2,8 mm
- Phạm vi nhiệt độ: -40 ° C ~ + 85 ° C

*** Tín hiệu xung nhịp và bộ định thời**

- Bộ tạo dao động 8 MHz tích hợp với tự hiệu chuẩn
- Bộ tạo dao động RC tích hợp với tự hiệu chuẩn
- Hỗ trợ cho bộ dao động tinh thể 2 MHz đến 40 MHz bên ngoài 31
- Hỗ trợ tinh thể 32 kHz bên ngoài cho RTC, tự hiệu chuẩn
- 2 nhóm hẹn giờ, mỗi nhóm gồm 2 bộ định thời đa năng 64 bit và 1 bộ giám sát hệ thống chính
- Bộ đếm thời gian RTC với độ chính xác phụ thứ hai

- Giám sát RTC

***Giao diện ngoại vi**

- 34 × programmable GPIOs
- 5 strapping GPIOs
- 6 input-only GPIOs
- 6 GPIOs needed for in-package flash/PSRAM (ESP32-D0WDR2 V3, ESP32-U4WDH)
- 12-bit SAR ADC up to 18 channels
- 2 × 8-bit DAC
- 10 × touch sensors
- 4 × SPI
- 2 × I2S
- 2 × I2C - 3 × UART
- 1 host (SD/eMMC/SDIO)
- 1 slave (SDIO/SPI)
- Ethernet MAC interface with dedicated DMA and IEEE 1588 support
- TWAI®, compatible with ISO 11898-1 (CAN Specification 2.0)
- RMT (TX/RX)
- Motor PWM
- LED PWM tối đa 16 kênh Bảo mật Hỗ trợ tất cả các tính năng bảo mật chuẩn IEEE 802.11, bao gồm WFA, WPA/WPA2 và WAPI
- Khởi động an toàn (Secure boot)
- Mã hóa flash (Flash encryption)
- 1024-bit OTP, lên đến 768-bit cho khách hàng

Tăng tốc phần cứng mật mã: AES, SHA-2, RSA, mật mã đường cong elliptic (ECC – elliptic curve cryptography), bộ tạo số ngẫu nhiên (RNG – random number generator)

***Ứng Dụng:** Với mức tiêu thụ điện năng thấp, ESP32 là sự lựa chọn lý tưởng cho các thiết bị IoT trong các lĩnh vực sau:

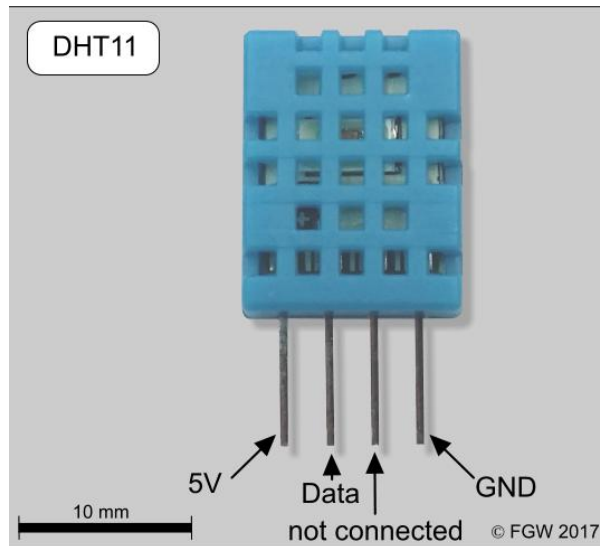
- Nhà thông minh
- Tự động trong công nghiệp
- Chăm sóc sức khỏe
- Điện tử dân dụng
- Nông nghiệp thông minh
- Máy POS - Robot dịch vụ

.....

3.2.2. Thiết bị cảm biến

3.2.2.1. Cảm biến nhiệt độ, độ ẩm DHT11

Module cảm biến DHT11 là một loại cảm biến trả về tín hiệu số rất hay được ứng dụng trong các ứng dụng đo nhiệt độ và độ ẩm thời gian thực. Hình 2.6 là hình ảnh mô đun và các chân của DHT11.



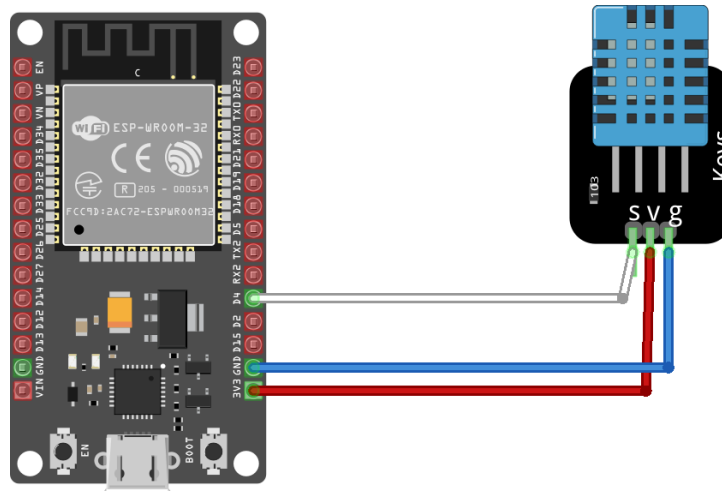
Hình 3.3: Mô đun và chân của DHT11

Chức năng của các chân:

- Vcc : chân nguồn dương 5V
- GND: Chân cấp nguồn 0V
- DATA : Chân đọc tín hiệu 1-Dây.

Thông số kỹ thuật của DHT11:

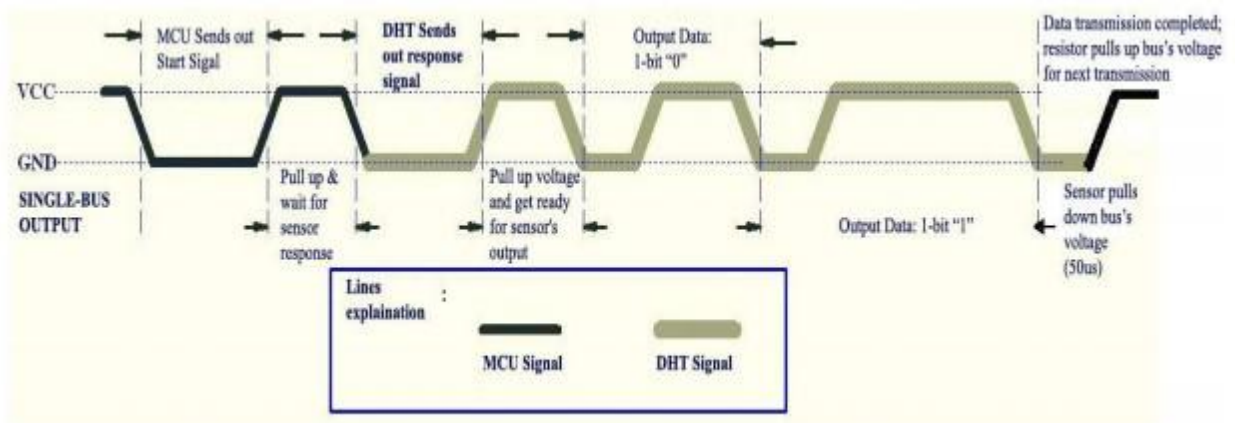
- Điện áp hoạt động : 4.2V – 5V
- Dòng điện : 60uA
- Công suất : 0.3mW
- Chuẩn truyền : 1 dây (Onewire)
- Nhiệt độ hoạt động : 0°C – 50°C
- Độ ẩm hoạt động : 0% – 80%
- Sai số nhiệt độ : 2°C
- Sai số độ ẩm : 5%
- Số chân : 3
- Loại : Module
- Kiểu chân : TO-92
- Kích thước : 4.3mm * 4.3mm



Hình 3.4 Sơ đồ kết nối điện tử của mô-đun DHT11 với ESP32

Hoạt động của DHT11:

Khi ESP32 gửi tín hiệu khởi động, mô-đun chuyển từ chế độ tiêu thụ điện năng thấp sang chế độ hoạt động, chờ ESP32 hoàn thành quá trình khởi động. Khi đã hoàn thành, DHT11 sẽ phản hồi bằng việc gửi dữ liệu 40 bit chứa thông tin về độ ẩm và nhiệt độ tương đối cho ESP32. Người dùng có thể lựa chọn thu thập (đọc) một số dữ liệu từ đó. Trong trường hợp không có tín hiệu khởi động từ ESP32, DHT11 sẽ không gửi bất kỳ tín hiệu phản hồi nào. Sau khi dữ liệu đã được thu thập, DHT11 sẽ chuyển về chế độ tiêu thụ điện năng thấp cho đến khi nhận được tín hiệu khởi động tiếp theo từ ESP32. Toàn bộ quá trình truyền thông của DHT11 với ESP thể hiện như hình ? .

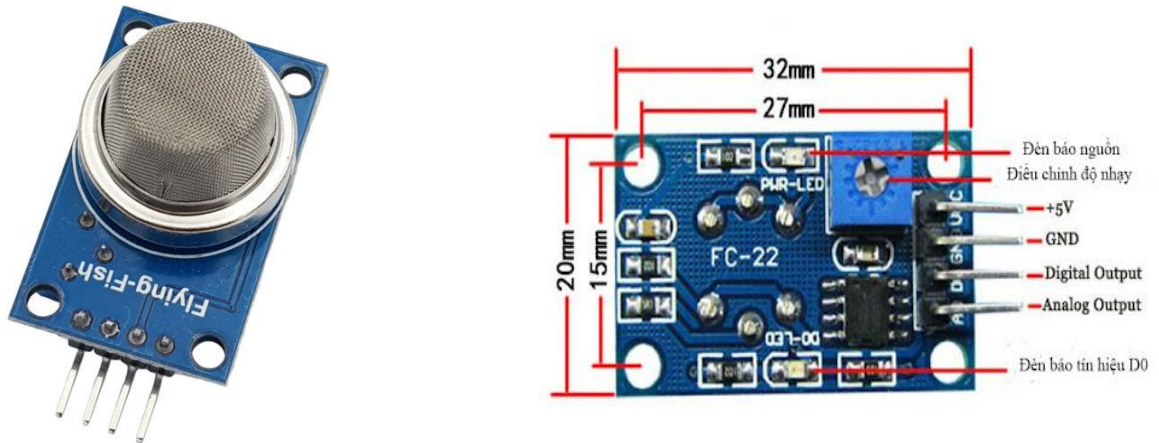


Hình 3.5 Quá trình truyền thông của DHT11

Để thực hiện khởi tạo và đọc nhiệt độ của DHT11 với ESP32 với ngôn ngữ C/C++ trong Arduino Studio, cần sử dụng thư viện DHT.h. Để khởi tạo DHT11 dùng hàm `begin()`, để đọc nhiệt độ, độ ẩm từ DHT11 sử dụng các hàm thư viện: `readHumidity()`, `readTemperature()`.

3.2.2.2. Cảm biến khí Gas MQ135

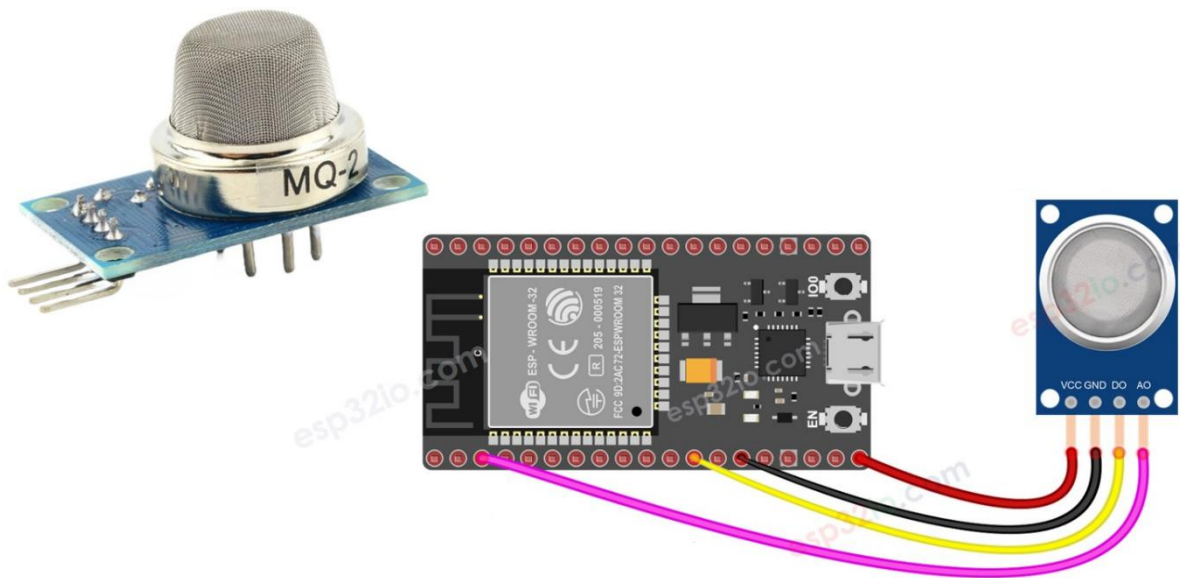
Cảm biến MQ135 có thể nhận biết được các chất khí như NH₃, Nox, Ancol, Benzen, Khói, gas, CO₂..... Đa số khí nó nhận biết đều là khí tạp chất và không có lợi cho sức khỏe nên chính vì vậy người ta gọi nó là cảm biến chất lượng không khí.



Hình 3.6 Mô-đun MQ135 và các chân tín hiệu

Thông số kỹ thuật:

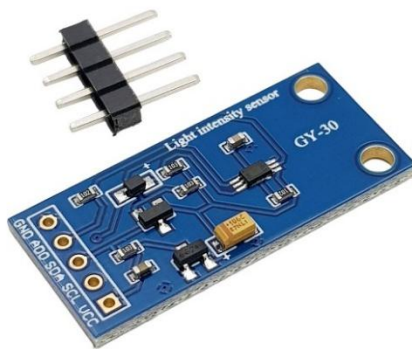
- Điện áp nguồn: $\leq 24\text{VDC}$
- Điện áp của heater: $5\text{V} \pm 0.1 \text{ AC/DC}$
- Điện trở tải: thay đổi được ($2\text{k}\Omega - 47\text{k}\Omega$)
- Điện trở của heater: $33\Omega \pm 5\%$
- Công suất tiêu thụ của heater: ít hơn 800mW
- Khoảng phát hiện: 10 – 300 ppm NH₃, 10 – 1000 ppm Benzene, 10 – 300 Alcol
- Kích thước: $32\text{mm} \times 20\text{mm}$
- Khoảng đo rộng
- Bền, tuổi thọ cao
- Phát hiện nhanh, độ nhạy cao
- Mạch đơn giản



Hình 3.7 Kết nối MQ135 với ESP32.

3.2.2.3 Cảm biến cường độ ánh sáng Lux BH1750

Cảm biến cường độ ánh sáng Lux BH1750 được sử dụng để đo cường độ ánh sáng theo đơn vị lux, cảm biến có ADC nội và bộ tiền xử lý nên giá trị được trả ra là giá trị trực tiếp cường độ ánh sáng lux mà không phải qua bất kỳ xử lý hay tính toán nào thông qua giao tiếp I2C .



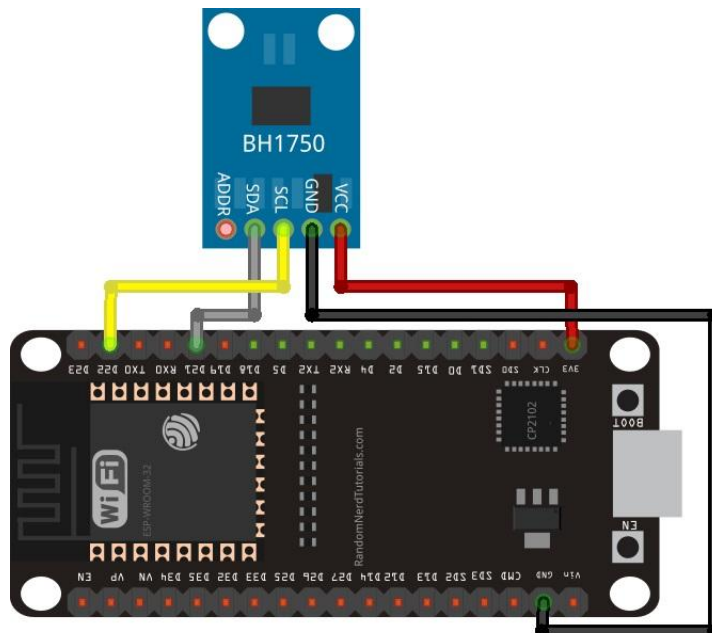
Hình 3.8 Cảm Biến Cường Độ Ánh Sáng Lux BH1750

THÔNG SỐ KỸ THUẬT

- Nguồn: 3~5VDC
- Giao tiếp: I2C
- Khoảng đo: 1 -> 65535 lux
- Kích cỡ: 21*16*3.3mm

Một số ví dụ về độ rọi của ánh sáng:

- Vào buổi tối : 0.001 – 0.02 Lux
- Ánh trắng : 0.02 – 0.3 lux
- Trời nhiều mây trong nhà : 5 – 50 lux
- Trời nhiều mây ngoài trời : 50 – 500 lux
- Trời nắng trong nhà : 100 – 1000 lux
- Ánh sáng cần thiết để đọc sách: 50 – 60 lux



Hình 3.9 kết nối Lux BH1750 với ESP32.

3.2.2.4 Cảm biến thân nhiệt chuyển động PIR SR505 Mini

Cảm biến thân nhiệt chuyển động PIR (Passive infrared sensor) HC-SR505 Mini có kích thước nhỏ gọn chỉ 10mm, được sử dụng để phát hiện chuyển động của các vật thể phát ra bức xạ hồng ngoại: con người, con vật, các vật phát nhiệt,...

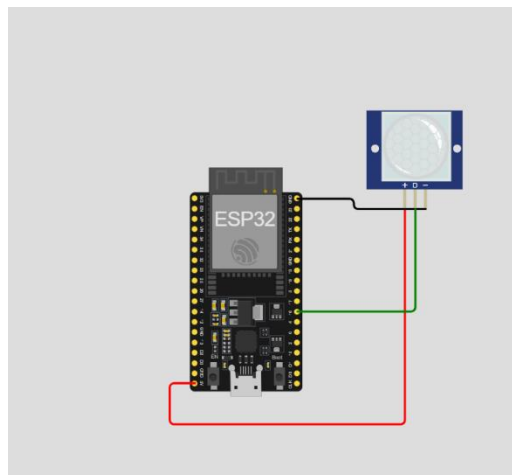
Cảm biến thân nhiệt chuyển động PIR HC-SR505 Mini sẽ xuất ra tín hiệu mức cao (High) khi phát hiện vật thể nhiệt chuyển động trong vùng quét, tín hiệu này sau đó sẽ được giữ ở mức cao trong khoảng thời gian trễ T sau khi kích hoạt, lúc này nếu cảm biến vẫn bắt được tín hiệu sẽ vẫn duy trì chân tín hiệu mức cao trong thời gian trễ T, chỉ khi trong khoảng thời gian trễ T mà cảm biến không bắt được tín hiệu thì chân tín hiệu cảm biến mới trở về mức thấp (Low).



Hình 3.10 Cảm biến thân nhiệt chuyển động PIR SR505 Mini

Thông số kỹ thuật:

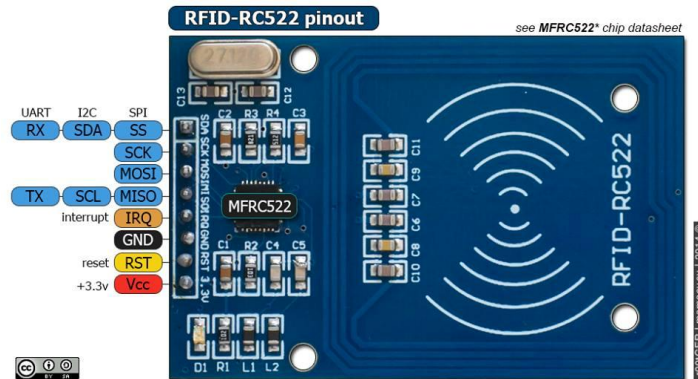
- Điện áp hoạt động: 4.5~20VDC
- Dòng tiêu thụ: <60uA
- Mức tín hiệu: High 3.3V / Low 0V
- Trigger: repeatable trigger
- Thời gian trễ T sau khi kích hoạt: 8s + -30%
- Góc quét: Max 100 độ (hình nón có tâm là cảm biến)
- Khoảng cách bắt: 3 meters
- Đường kính thấu kính: 10mm
- Kích thước: 10 x 23mm



Hình 3.11 kết nối SR505 với ESP32.

3.2.2.6 RFID RC522

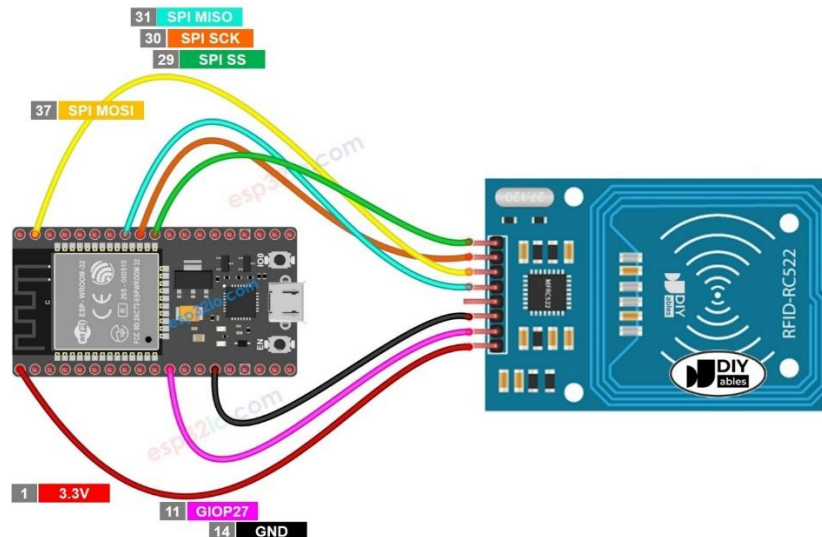
Mạch RFID NFC 13.56MHz RC522 sử dụng IC MFRC522 được sử dụng để đọc và ghi dữ liệu cho thẻ RFID / NFC tần số 13.56MHz, mạch có thiết kế nhỏ gọn được sử dụng rất phổ biến hiện nay với Arduino hoặc các loại Vi điều khiển khác trong các ứng dụng cần ghi, đọc thẻ RFID / NFC.



Hình 3.14 RFID NFC 13.56MHz RC522

Thông số kỹ thuật:

- Nguồn sử dụng: 3.3VDC
- Dòng điện: 13~26mA
- Tần số hoạt động: 13.56Mhz
- Khoảng cách hoạt động: 0~60mm (mifare1 card)
- Chuẩn giao tiếp: SPI
- Tốc độ truyền dữ liệu: tối đa 10Mbit/s
- Các loại card RFID hỗ trợ: mifare1 S50, mifare1 S70, mifare UltraLight, mifare Pro, mifare Desfire
- Kích thước: 40mm × 60mm



Hình 3.15 kết nối RFID RC522 với ESP32.

3.2.3 Thiết bị chấp hành

3.2.3.1 Còi Buzzer 5VDC

Còi Buzzer 5VDC có tuổi thọ cao, hiệu suất ổn định, chất lượng tốt, được sản xuất nhỏ gọn phù hợp thiết kế với các mạch còi buzzer nhỏ gọn, mạch báo động.



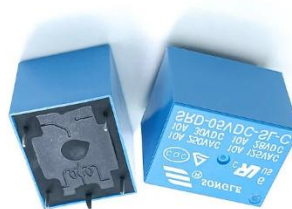
Hình 3.16 Còi Buzzer 5VDC

Thông số kỹ thuật:

- Nguồn : 3.5V - 5.5V
- Dòng điện tiêu thụ: <25mA
- Tần số cộng hưởng: 2300Hz \pm 500Hz
- Biên độ âm thanh: >80 dB
- Nhiệt độ hoạt động: -20 °C đến +70 °C
- Kích thước : Đường kính 12mm, cao 9,7mm

3.2.3.2 Relay 5V 10A 5 Chân SRD 05VDC-SL-C

Relay 5 chân SRD 5VDC là loại linh kiện đóng ngắt điện cơ đơn giản. Nó gồm 2 phần chính là cuộn hút và các tiếp điểm.



Hình 3.17 Relay 5V

Thông số kỹ thuật

- Dòng AC max: 10 A

- Dòng AC min: 6 A
- Diameter, PCB hole: 1.3 mm
- Length / Height, external: 22 mm
- Material, contact: Silver alloy
- Nhiệt độ hoạt động: - 45 °C to 75 °C
- Công suất cuộn dây (coil) DC: 360 mW
- Thời gian tác động: 10 ms
- Thời gian nhả hãm: 5 ms
- Điện áp điều khiển cuộn dây (coil): 5 V

Ứng dụng của relay

- Nhìn chung, công dụng của relay là “dùng một năng lượng nhỏ để đóng cắt nguồn năng lượng lớn hơn”.
- Relay được dùng khá thông dụng trong các ứng dụng điều khiển động cơ và chiếu sáng.
- Khi cần đóng cắt nguồn năng lượng lớn, relay thường được ghép nối tiếp. Nghĩa là một relay nhỏ điều khiển một relay lớn hơn, và relay lớn sẽ điều khiển nguồn công suất.

3.2.3.3 Động cơ servo SG90

Động cơ servo SG90 180 độ có tốc độ phản ứng nhanh, các bánh răng được làm bằng nhựa nên cần lưu ý khi nâng tải nặng vì có thể làm hư bánh răng, động cơ RC Servo 9G có tích hợp sẵn Driver điều khiển động cơ bên trong nên có thể dễ dàng điều khiển góc quay bằng phương pháp điều độ rộng xung PWM.



Hình 3.18 Động cơ servo SG90

THÔNG SỐ KỸ THUẬT

- Điện áp hoạt động: 4.8-5VDC
 - Tốc độ: 0.12 sec/ 60 deg (4.8VDC)
 - Lực kéo: 1.6 Kg.cm
 - Kích thước: 21x12x22mm
 - Trọng lượng: 9g.
- Điều khiển PWM:

- Độ rộng xung 0.5ms ~ 2.5ms tương ứng 0-180 độ
- Tần số 50Hz, chu kỳ 20ms

Sơ đồ dây:

- Đỏ: Dương nguồn
- Nâu: Âm nguồn
- Cam: Tín hiệu

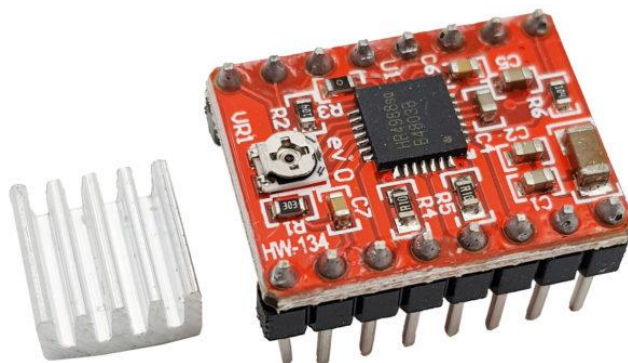
3.2.3.4. Động cơ Step và Module A4988



Hình 3.19 Động cơ bước size 42 1.8 step

Thông số kỹ thuật của Động cơ bước size 42 1.8 step

- Kích thước bao: 42mm x 42mm x 32mm
- Kích thước trục: 5mm
- Lưu ý sản phẩm không có chân nguồn trực tiếp mà sẽ lấy nguồn từ driver động cơ bước để chạy
- Trọng lượng: 220g



Hình 3.20 Mạch Điều Khiển Động Cơ Bước A4988

Mạch điều khiển động cơ bước A4988 là driver điều khiển động cơ bước cực kỳ nhỏ gọn, hỗ trợ nhiều chế độ làm việc, điều chỉnh được dòng ra cho động cơ, tự động

ngắt điện khi quá nóng. Mạch điều khiển A4988 hỗ trợ nhiều chế độ hoạt động của động cơ bước lưỡng cực như: Full, 1/2, 1/4, 1/8 và 1/16.

THÔNG SỐ KỸ THUẬT

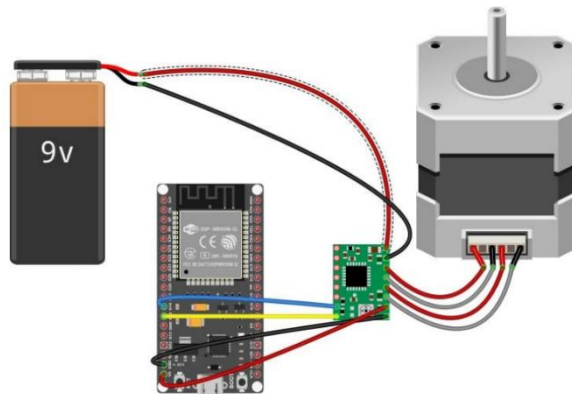
- Công suất ngõ ra lên tới 35V, dòng đỉnh 2A.
- Có 5 chế độ: Full bước, 1/2 bước, 1/4 bước, 1/8 bước, 1/16 bước
- Điều chỉnh dòng ra bằng triết áp, nằm bên trên $Current\ Limit = VREF \times 2.5$
- Tự động ngắt điện khi quá nhiệt.

CÁCH SỬ DỤNG

- Lựa chọn chế độ full hay 1/2 hay 1/4... sẽ được thông qua 3 pin MS1 MS2 MS3. Mình thường nối thẳng 3 pin này với công tắc bit 3p để dễ thiết lập từ trên phần cứng. Lưu ý là nếu thả nổi 3 pin này tức là mode full step.
- Bật tắt động cơ thì thông qua pin ENABLE, mức LOW là bật module, mức HIGH là tắt module
- Điều khiển chiều quay của động cơ thông qua pin DIR
- Điều khiển bước của động cơ thông qua pin STEP, mỗi xung là tương ứng với 1 bước (hoặc vi bước)
- Hai chân Sleep với Reset luôn nối với nhau.

MS1	MS2	MS3	Microstep Resolution
Low	Low	Low	Full step
High	Low	Low	Half step
Low	High	Low	Quarter step
High	High	Low	Eighth step
High	High	High	Sixteenth step

SƠ ĐỒ KẾT NỐI

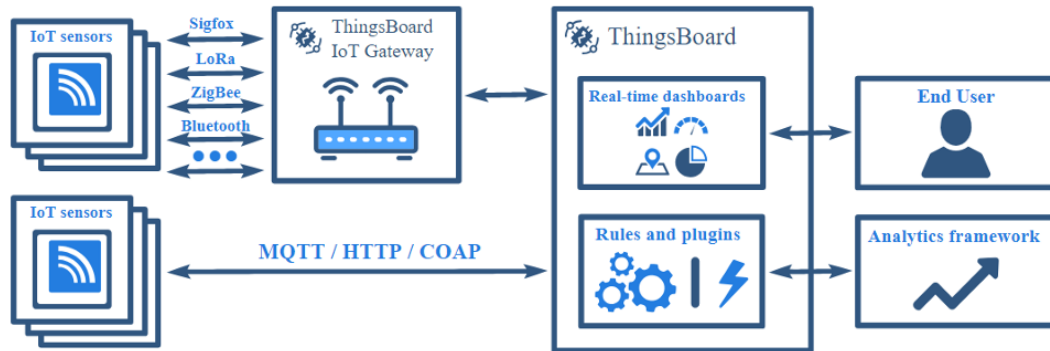


Hình 3.21 Kết nối giữa A4899 với ESP32

3.2.4 Thiết bị quản lý trung tâm (máy chủ hệ thống)

3.2.4.1 Giới thiệu về ThingsBoard

ThingsBoard là một nền tảng IoT mã nguồn mở. Nó cho phép phát triển nhanh chóng, quản lý và mở rộng các dự án IoT. Với nền tảng Thingsboard bạn có thể thu thập, xử lý, hiển thị trực quan và quản lý thiết bị.



Hình 3.22 Cấu trúc của máy chủ Thingsboard

Các tính năng của Thingsboard

- Thu thập dữ liệu từ xa

Thingsboard hỗ trợ thu thập và lưu trữ dữ liệu từ xa theo cách đáng tin cậy. Người dùng có thể truy cập dữ liệu đã thu thập thông qua web với khả năng tùy chỉnh linh hoạt. Thingsboard cho phép các thiết bị kết nối thông qua các giao thức IoT tiêu chuẩn như MQTT, CoAP và HTTP thông qua một API chỉ định riêng cho từng giao thức kết nối.

- Hiển thị trực quan dữ liệu đã thu thập

Thingsboard cung cấp nhiều tiện ích có sẵn để người dùng xây dựng giao diện giám sát, được gọi là dashboard, để hiển thị dữ liệu trực quan. Một số tiện ích có sẵn được hỗ trợ bởi Thingsboard sử dụng trong hiển thị trực quan hóa dữ liệu như tiện ích Google map, đồ thị thời gian thực, các thẻ HTML hiển thị. Ngoài ra Thingsboard cũng cho phép người dùng tạo các tiện ích riêng để tích hợp trên dashboard.



Hình 3.23 Giao diện quản lý, giám sát dữ liệu (dashboard) trên Thingsboard

- Quản lý thiết bị

Thingsboard cung cấp khả năng đăng ký và quản lý thiết bị (device) một cách dễ dàng, linh hoạt. Nó cho phép theo dõi các loại thuộc tính gắn với thiết bị cả ở phía máy khách và phía máy chủ Thingsboard, thông qua API theo giao thức kết nối giữa máy khách, thiết bị với máy chủ TB. TB cũng hỗ trợ API cho phép đầu cuối và máy chủ giao tiếp lệnh từ xa – RPC với nhau.

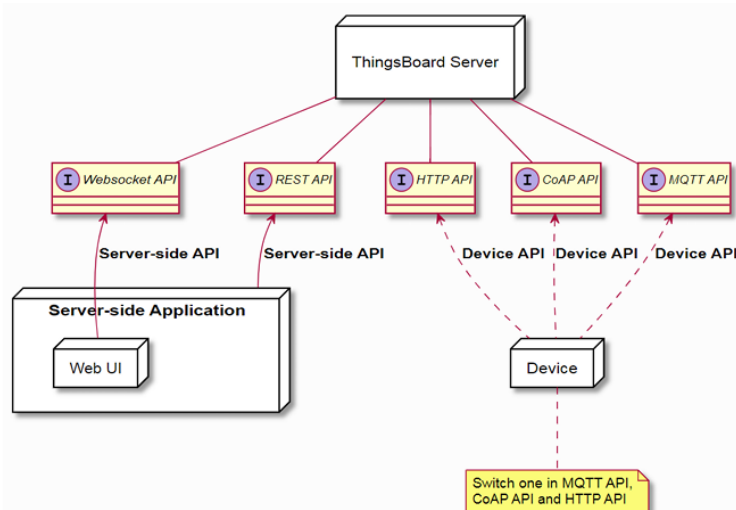
- Quản lý các báo động

Thingsboard cung cấp khả năng tạo và quản lý các cảnh báo liên quan đến các thực thể được thiết lập, quản lý trên máy chủ như device, asset,... Cho phép giám sát thông báo theo thời gian thực và tạo cơ chế cảnh báo cho các đối tượng có liên quan.

- 100% mã nguồn mở

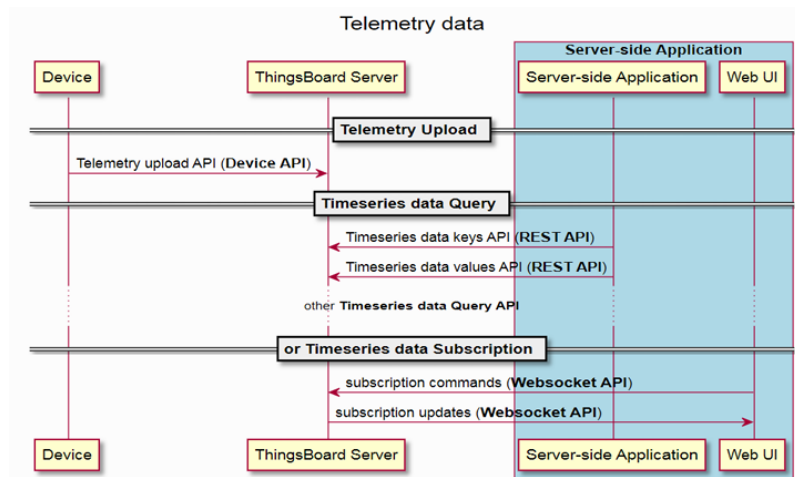
3.2.4.2 Gửi dữ liệu lên máy chủ Thingsboard

Để truyền thông, trao đổi dữ liệu với Thingsboard, phía đầu cuối (bao gồm các thiết bị, phần mềm,...) cần kết nối với mạng Internet. Ưu điểm của ThingsBoard là hỗ trợ nhiều giao thức kết nối, là các giao thức truyền thông của IoT và Internet, cho phép phía đầu cuối có thể kết nối theo các phương thức khác nhau



Để gửi dữ liệu tới máy chủ TB, thiết bị đầu cuối có thể thực hiện theo một số giao thức truyền thông mà TB hỗ trợ, bao gồm HTTP, MQTT, REST, Websocket. Với từng giao thức, thiết bị đầu cuối sẽ gửi một bản tin theo giao thức truyền thông, kèm theo dữ liệu được định dạng theo chuẩn JSON.

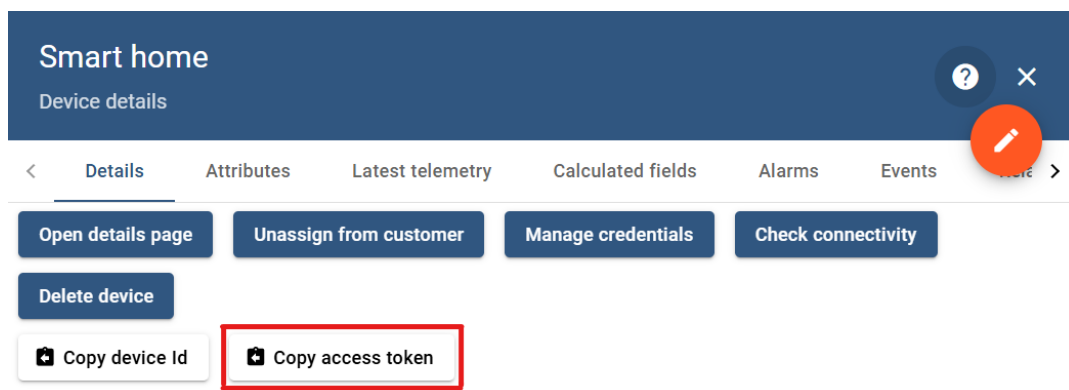
Thingsboard đưa ra một nhóm API, chỉ định cách thức kết nối và gửi dữ liệu tới máy chủ từ thiết bị đầu cuối (device). Dữ liệu cảm biến được gửi từ device tới máy chủ TB được gọi là Telemetry, dữ liệu từ xa. Tuy thuộc vào giao thức kết nối được sử dụng, MQTT hay HTTP, dạng thức API để gửi telemetry tới máy chủ sẽ có một số điểm khác biệt.



❖ Kết nối ThingsBoard với Node-RED bằng MQTT.

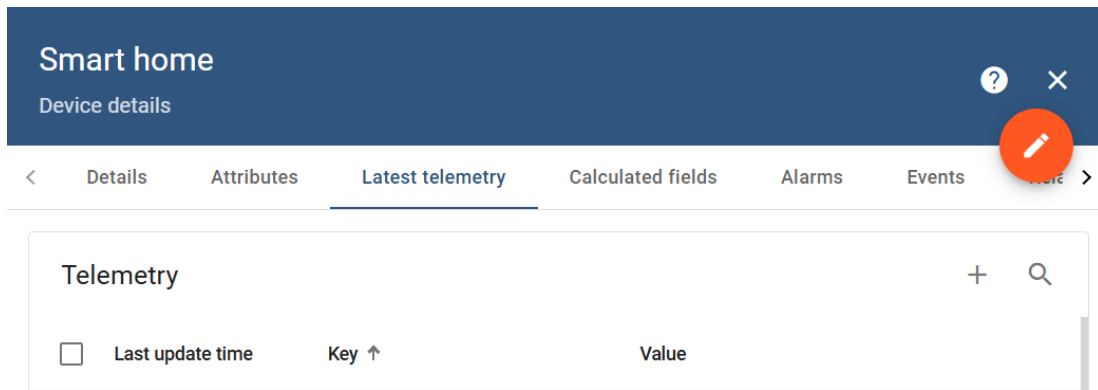
Trong quá trình xây dựng hệ thống giám sát và điều khiển thiết bị IoT, việc kết nối nền tảng ThingsBoard với môi trường lập trình trực quan Node-RED đóng vai trò quan trọng nhằm đảm bảo luồng dữ liệu được truyền nhận một cách linh hoạt và hiệu quả. Việc kết nối này được thực hiện thông qua giao thức MQTT – một giao thức truyền thông nhẹ, phổ biến trong lĩnh vực IoT.

Đầu tiên, trong nền tảng ThingsBoard, cần tạo một thiết bị mới (Device) và lấy thông tin Access Token – đây chính là khóa định danh cho thiết bị MQTT. Việc tạo thiết bị được thực hiện trong phần quản trị của ThingsBoard, sau khi đăng nhập bằng tài khoản tenant. Sau khi tạo thiết bị, vào mục "Credentials" để lấy Access Token dùng cho việc xác thực khi Node-RED kết nối.



Ở phía Node-RED, cần cài đặt node MQTT (node-red-node-mqtt) thông qua mục “Manage Palette”. Sau đó, tạo một node mqtt out để gửi dữ liệu từ Node-RED lên ThingsBoard. Trong cấu hình node MQTT Broker, khai báo địa chỉ server (ví dụ: demo.thingsboard.io hoặc localhost nếu sử dụng cài đặt nội bộ), cổng là 1883, và chèn Access Token vừa lấy vào phần Username (password để trống).

Node inject được sử dụng để tạo dữ liệu thử nghiệm và kết nối với node mqtt out. Sau khi nhấn “Deploy” và kích hoạt node inject, dữ liệu sẽ được gửi và hiển thị trên giao diện ThingsBoard, cụ thể tại phần “Latest Telemetry” của thiết bị.



Ngoài việc gửi dữ liệu từ Node-RED lên ThingsBoard, hệ thống còn có thể nhận dữ liệu ngược lại từ ThingsBoard thông qua cơ chế RPC hoặc khi có cập nhật về thuộc tính (attribute). Để thực hiện điều này, sử dụng node mqtt in trong Node-RED với topic: `v1/devices/me/rpc/request/+`. Khi dashboard của ThingsBoard gửi yêu cầu điều khiển (ví dụ bật/tắt thiết bị), Node-RED sẽ nhận được thông điệp dưới dạng JSON và có thể xử lý theo ý muốn. Ngoài ra, các thuộc tính (shared attributes) cập nhật từ server có thể nhận qua topic: `v1/devices/me/attributes`.

Việc kết nối thành công giữa ThingsBoard và Node-RED giúp xây dựng một hệ thống IoT hoàn chỉnh, cho phép lập trình xử lý logic, tự động hóa, hiển thị và điều khiển thiết bị từ xa một cách linh hoạt, trực quan và hiệu quả.

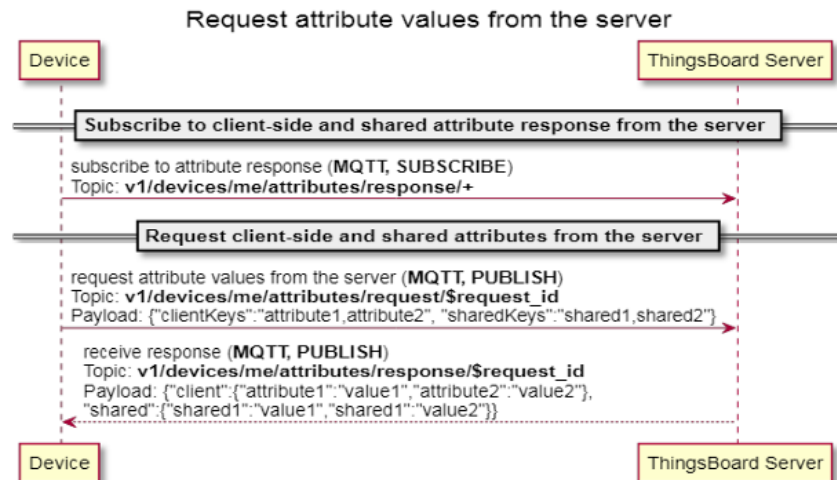
3.2.4.3 Truy vấn để đọc giá trị thuộc tính của thiết bị trên máy chủ Thingsboard

Truy vấn để đọc giá trị thuộc tính trên TB thông qua giao thức MQTT

❖ Thiết lập và cập nhật thuộc tính trực tiếp từ thiết bị đầu cuối (Request attribute values from the server)

Để thành phần đầu cuối (thiết bị hoặc phần mềm đầu cuối) xác định giá trị thuộc tính thiết bị được thiết lập trên máy chủ TB, thì trước tiên thành phần đầu cuối cần đăng ký, thông qua thủ tục subscribe (theo giao thức MQTT) tại topic `1/devices/me/attributes/response/+`

Tiếp theo, khi đầu cuối muốn xác định thuộc tính của thiết bị, sẽ gửi một bản tin yêu cầu, theo thủ tục publish (theo giao thức MQTT), theo dạng thức như mô tả trong hình 3.1. Khi đó máy chủ TB sẽ phản hồi lại một bản tin, chứa thông tin các thuộc tính yêu cầu. Thông tin thuộc tính trong bản tin phản hồi được định dạng dữ liệu JSON, bao gồm các thuộc tính client (thuộc tính thiết lập đầu cuối) và các thuộc tính shared (thuộc tính chia sẻ).

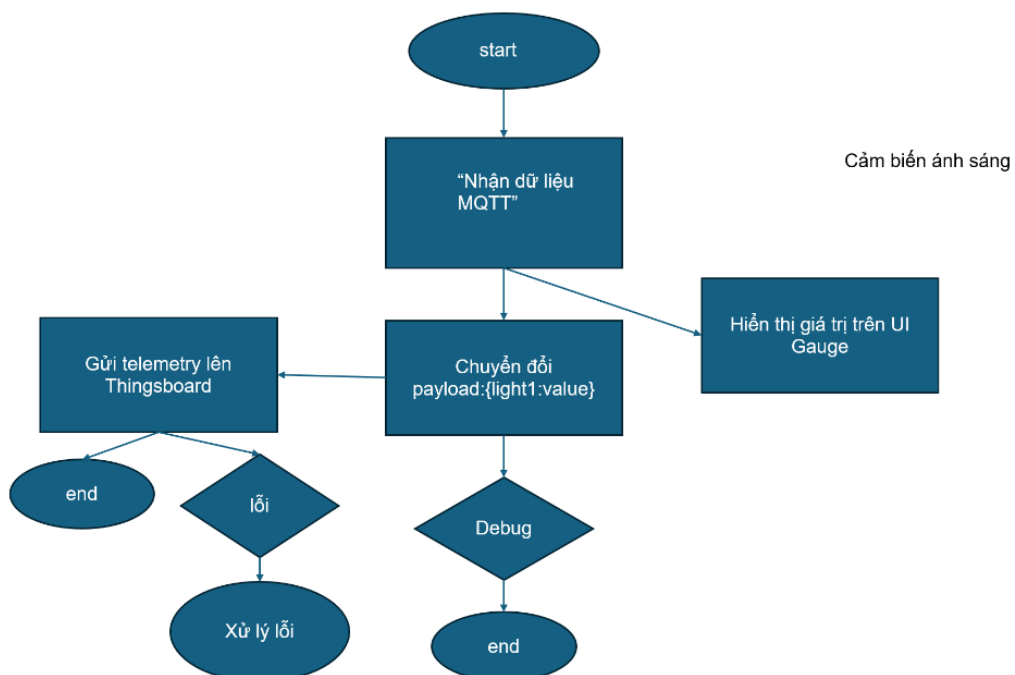


Kết nối, truyền thông theo giao thức MQTT giữa đầu cuối và máy chủ TB, để đọc giá trị của thuộc tính thiết bị trên máy chủ, thực hiện theo các bước mô tả trong hình 3.1. Để xác định các thuộc tính thiết bị, phía đầu cuối sẽ gửi bản tin yêu cầu cho TB, theo thủ tục publish với topic: `v1/devices/me/attributes/request/$request_id`

Trong đó `$request_id` là một giá trị chỉ số (chỉ số yêu cầu) do đầu cuối thiết lập, đồng thời gửi kèm theo bản tin này là thông tin về thuộc tính cần xác định giá trị. Thông tin này được định dạng theo chuẩn JSON.

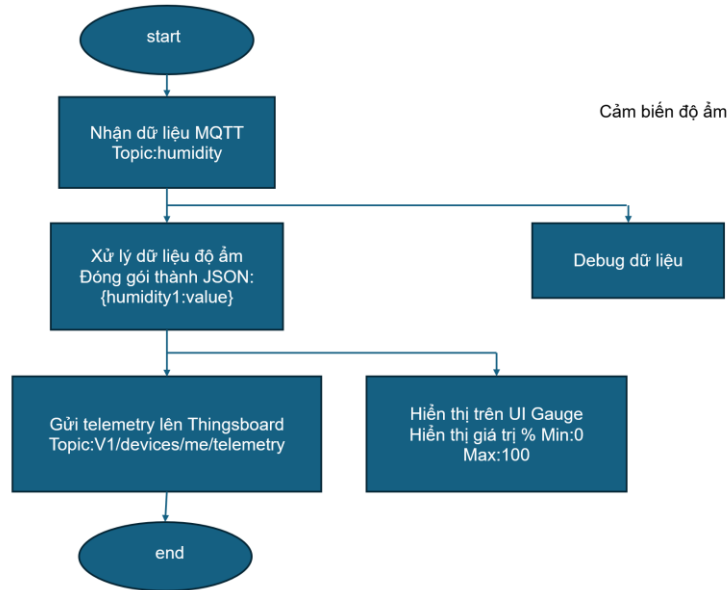
Tuy vậy trước khi gửi bản tin yêu cầu này, thì phía đầu cuối cần đăng ký máy chủ TB gửi thông tin phản hồi lại tương ứng với các yêu cầu từ phía đầu cuối, thông qua thủ tục subscribe tại topic: `v1/devices/me/attributes/response/+`

3.3. Giải đồ chương trình cho hệ thống xây dựng trên nền tảng Node-Red



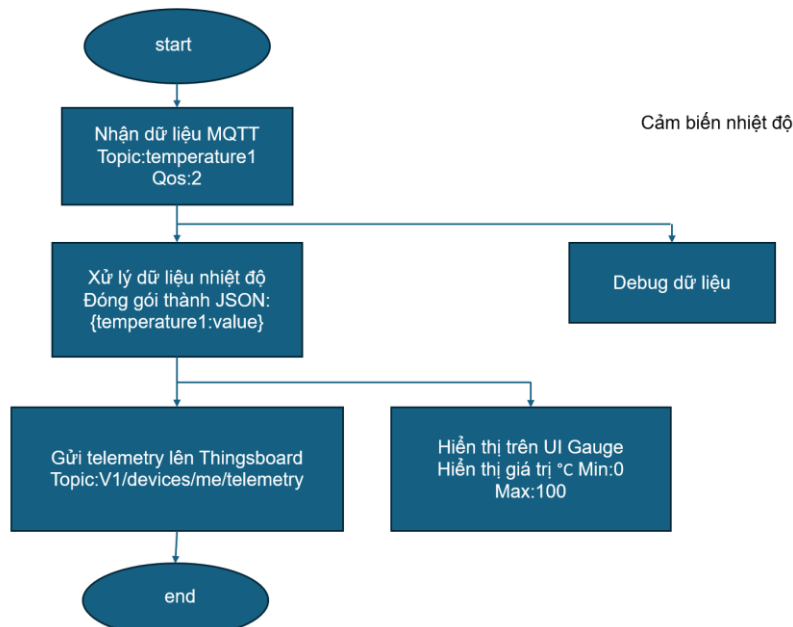
Hình 3.24 Sơ đồ xử lý dữ liệu trên Node-red

Sơ đồ mô tả quy trình xử lý dữ liệu từ cảm biến ánh sáng thông qua giao thức MQTT. Dữ liệu sau khi được nhận sẽ được chuyển đổi sang định dạng payload {light1:value} và đồng thời hiển thị giá trị trên giao diện người dùng dạng đồng hồ đo (UI Gauge). Sau đó, dữ liệu được gửi lên nền tảng Thingsboard để giám sát; nếu xảy ra lỗi, hệ thống sẽ chuyển sang bước xử lý lỗi hoặc debug trước khi kết thúc quy trình.



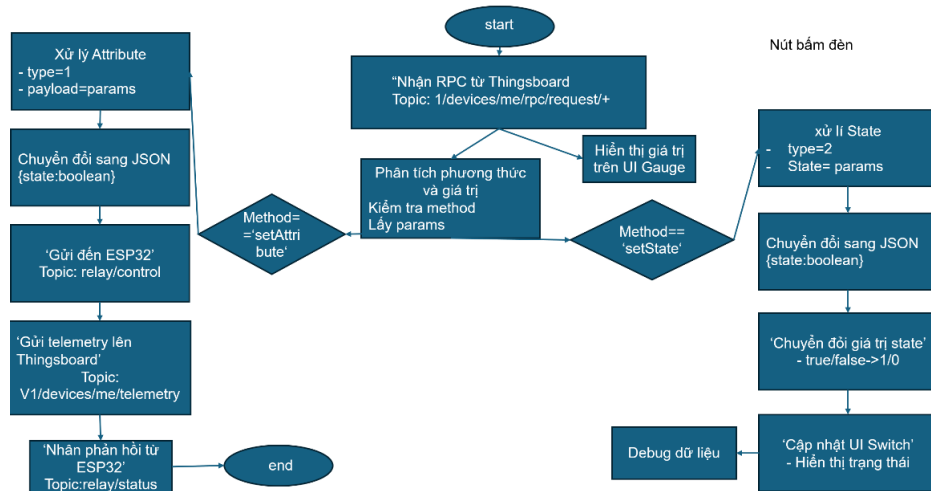
Hình 3.25 Sơ đồ quá trình xử lý dữ liệu cảm biến độ ẩm trên Node-red

Sơ đồ trên mô tả quy trình xử lý dữ liệu từ cảm biến độ ẩm sử dụng giao thức MQTT. Dữ liệu độ ẩm được nhận từ topic humidity, sau đó xử lý và đóng gói thành định dạng JSON {humidity1: value}. Giá trị này sẽ được hiển thị trên giao diện người dùng dưới dạng phần trăm với khoảng giá trị từ 0 đến 100, đồng thời được gửi lên nền tảng Thingsboard qua topic V1/devices/me/telemetry. Ngoài ra, hệ thống cũng hỗ trợ debug dữ liệu trong quá trình xử lý nếu cần.



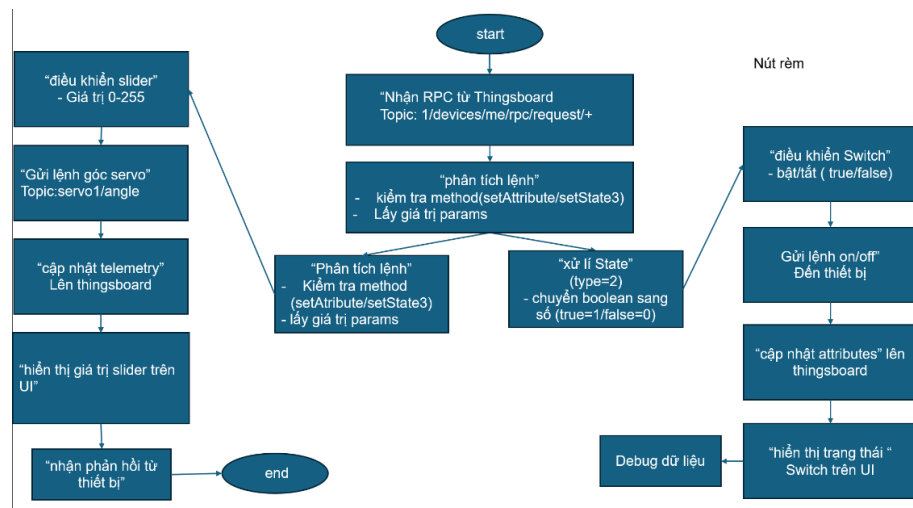
Hình 3.26 Sơ đồ xử lý dữ liệu của cảm biến nhiệt độ trên Node-red

Sơ đồ trên mô tả quy trình xử lý dữ liệu từ cảm biến nhiệt độ với giao thức MQTT và mức độ tin cậy QoS:2. Dữ liệu nhiệt độ được nhận từ topic temperature1, sau đó xử lý và đóng gói thành định dạng JSON {temperature1: value}. Giá trị này được hiển thị trên giao diện người dùng với đơn vị °C (giới hạn từ 0 đến 100), đồng thời gửi lên Thingsboard qua topic V1/devices/me/telemetry. Ngoài ra, hệ thống có thể thực hiện bước debug để kiểm tra dữ liệu nếu cần thiết.



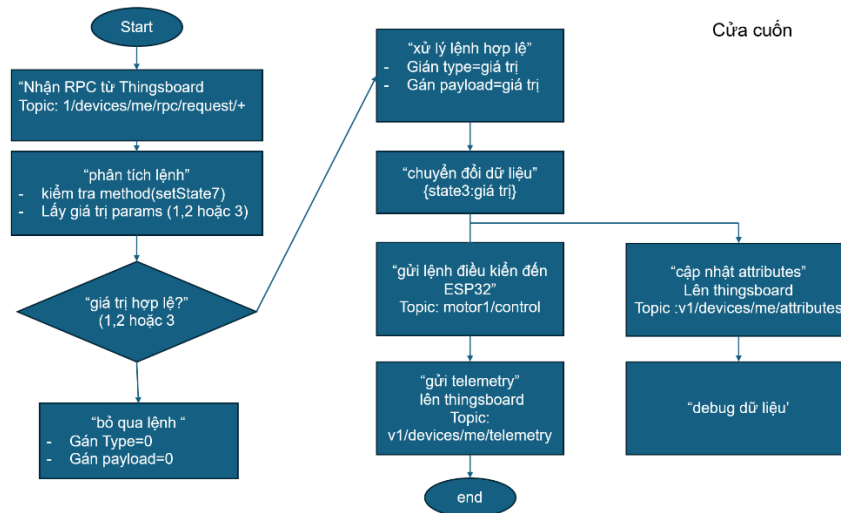
Hình 3.27 Sơ đồ xử lý dữ liệu của nút bấm đèn trên Node-red

Sơ đồ trên mô tả quá trình điều khiển thiết bị (bật/tắt đèn) thông qua giao tiếp RPC từ Thingsboard đến ESP32. Khi nhận yêu cầu RPC từ topic v1/devices/me/rpc/request/+, hệ thống sẽ phân tích phương thức (method) và tham số (params). Nếu method là setAttribute, dữ liệu được xử lý và gửi đến ESP32 qua topic relay/control, sau đó phản hồi và telemetry được gửi lại Thingsboard. Nếu method là setState, dữ liệu được chuyển đổi và cập nhật lên giao diện người dùng (UI Switch) để hiển thị trạng thái. Debug dữ liệu cũng được tích hợp để kiểm tra quá trình.



Hình 3.28 Sơ đồ xử lý dữ liệu của cảm biến ánh sáng trên Node-red

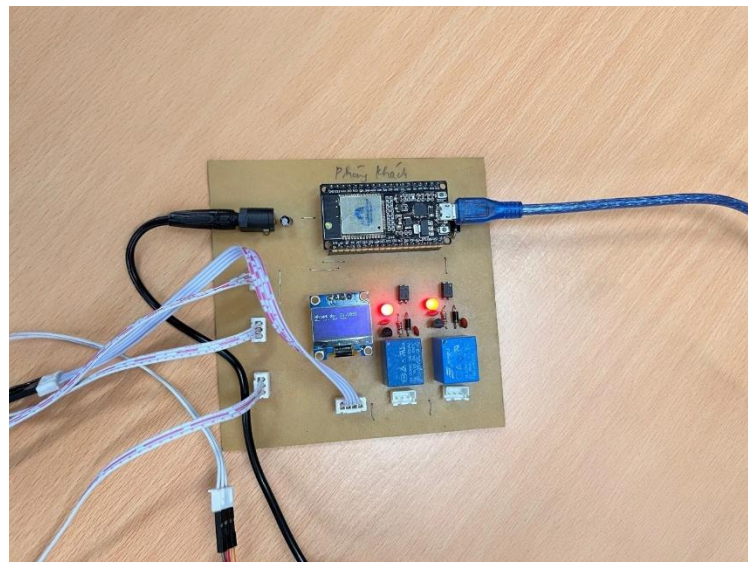
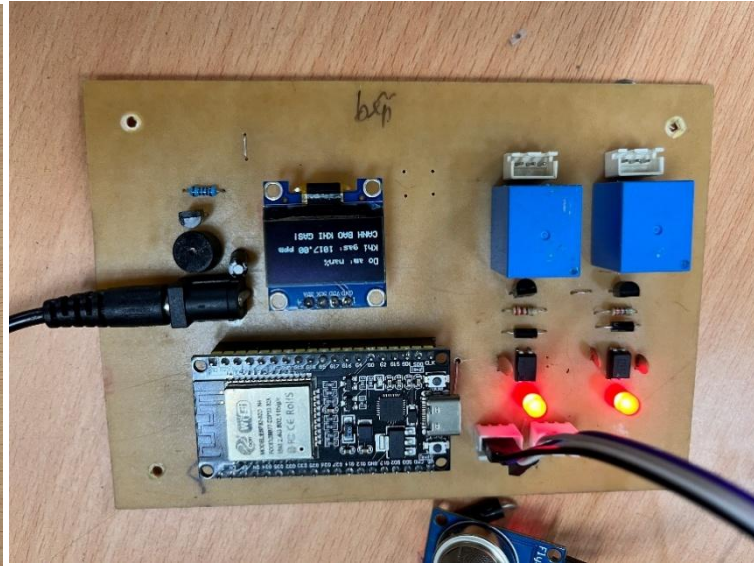
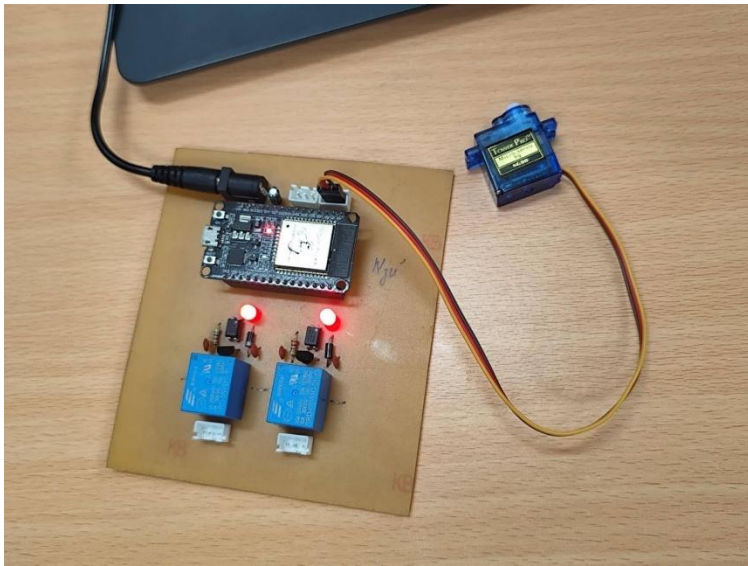
Sơ đồ trên mô tả quá trình điều khiển và giám sát thiết bị rèm cửa sử dụng ESP32 và nền tảng Thingsboard. Người dùng có thể điều chỉnh góc quay của servo thông qua slider (giá trị từ 0–255) hoặc bật/tắt rèm bằng nút Switch (giá trị true/false). Các lệnh được gửi qua giao thức MQTT đến ESP32, sau đó được cập nhật lại lên Thingsboard để hiển thị trạng thái trên giao diện người dùng. Hệ thống cũng xử lý dữ liệu và phản hồi từ thiết bị để đảm bảo tính đồng bộ và chính xác.



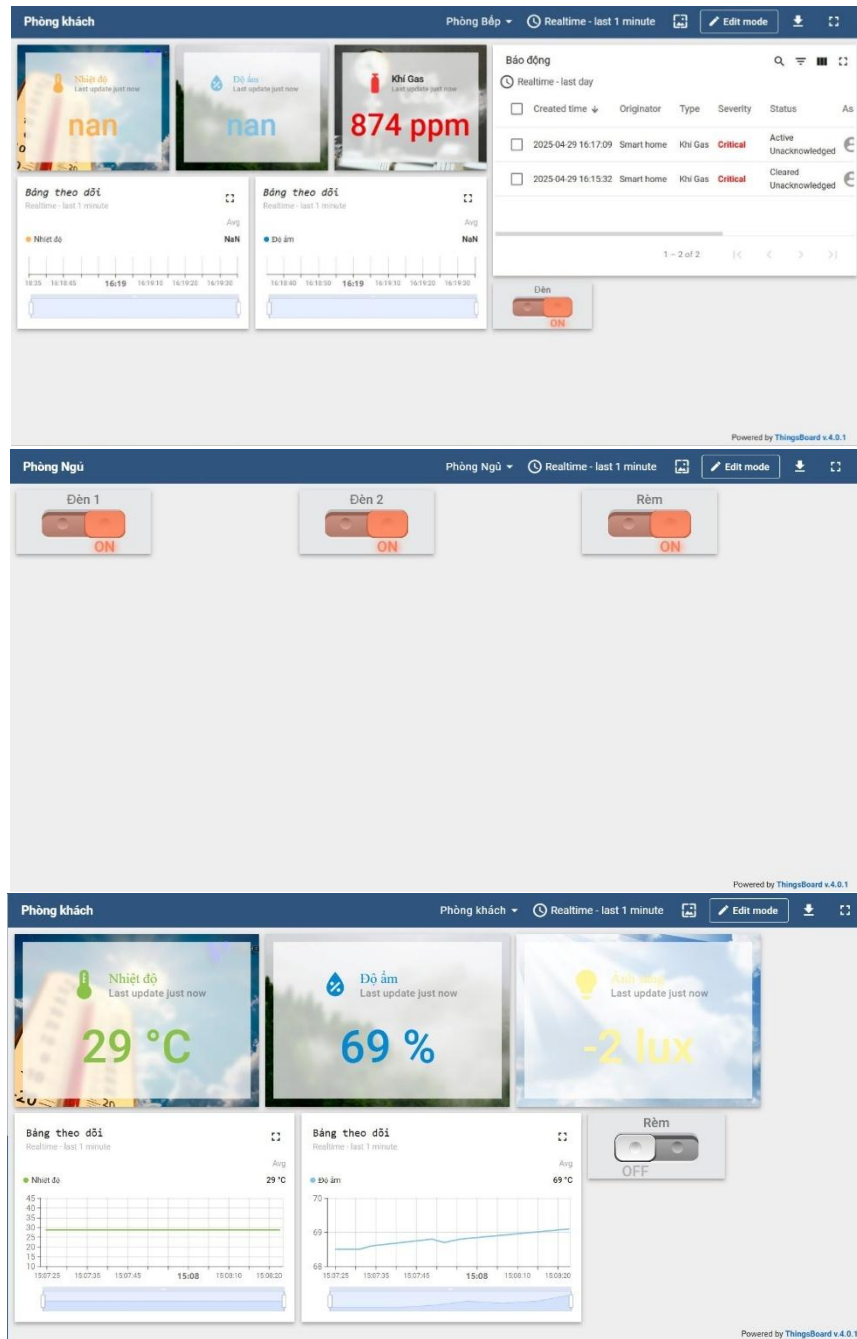
Hình 3.29 Sơ đồ xử lý dữ liệu của cửa cuốn trên Node-red

Sơ đồ trên mô tả quy trình điều khiển cửa cuốn thông qua nền tảng Thingsboard và thiết bị ESP32. Khi Thingsboard gửi RPC tới thiết bị, hệ thống sẽ phân tích lệnh và kiểm tra giá trị params có hợp lệ (1, 2 hoặc 3) hay không. Nếu hợp lệ, lệnh sẽ được chuyển đổi dữ liệu và gửi điều khiển đến ESP32 qua topic motor1/control. Đồng thời, dữ liệu trạng thái cũng được cập nhật về Thingsboard qua các topic telemetry và attributes, giúp hiển thị và debug thông tin trên giao diện. Nếu giá trị không hợp lệ, hệ thống sẽ bỏ qua lệnh bằng cách gán giá trị mặc định.

3.4. Thử nghiệm và đánh giá



Hình 3.30. Một số hình ảnh hệ thống thử nghiệm



Hình 3.31. Kết quả chạy hệ thống với đám mây Thingsboard

Hệ thống đã được lắp ráp và thử nghiệm thành công với các cảm biến và thiết bị điều khiển như relay, cảm biến nhiệt độ, độ ẩm, ánh sáng và khí gas. Các tín hiệu từ thiết bị được thu thập và truyền lên nền tảng đám mây Thingsboard để hiển thị theo thời gian thực. Giao diện Thingsboard cho phép theo dõi thông số môi trường nhà thông minh và điều khiển thiết bị từ xa thông qua internet. Kết quả thử nghiệm cho thấy hệ thống hoạt động ổn định, đáp ứng đúng yêu cầu giám sát và điều khiển từ xa.