

**ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA ĐIỆN – ĐIỆN TỬ  
BỘ MÔN ĐIỆN TỬ**



# **LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC**

## **NỀN TẢNG GIÁM SÁT**

### **CÁC HỆ THỐNG NĂNG LƯỢNG ĐIỆN MẶT TRỜI**

**GVHD:** ThS. Bùi Quốc Bảo  
**SVTH:** Phạm Thái Hoà 1711441  
Trần Ngọc Trâm 1713587

TP. HỒ CHÍ MINH, THÁNG 08 NĂM 2021

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

TRƯỜNG ĐẠI HỌC BÁCH KHOA

Độc lập – Tự do – Hạnh phúc.

----- ☆ -----

----- ☆ -----

Số: \_\_\_\_\_ /BKĐT  
Khoa: **Điện – Điện tử**  
Bộ Môn: **Điện Tử**

## NHIỆM VỤ LUẬN VĂN TỐT NGHIỆP

- HỌ VÀ TÊN: Phạm Thái Hoà MSSV: 1711441  
Trần Ngọc Trâm MSSV: 1713587
- NGÀNH: **ĐIỆN TỬ - VIỄN THÔNG** LỚP: DD17DV3&DD17DV7
- Đề tài: Nền tảng giám sát các hệ thống năng lượng điện mặt trời.
- Nhiệm vụ (Yêu cầu về nội dung và số liệu ban đầu):

Về Modbus IoT Gateway:

- Lập trình giao tiếp giữa IoT Gateway với các thiết bị đầu cuối
- Lập trình kết nối Gateway vào ứng dụng web trên server.
- Lập trình các phần mềm để kết nối, thay đổi các thông số trong Gateway.

Về ứng dụng web:

- Xây dựng ứng dụng web với các chức năng đăng nhập, đăng xuất, tạo người dùng và phân quyền người.
- Thiết kế giao diện web dùng để hiển thị, tính toán và phân tích các thông số của hệ thống pin năng lượng mặt trời.
- Kết nối với backend server tiến hành lấy dữ liệu thực được gửi lên từ gateway và hiển thị trên giao diện web đã xây dựng.
- Xây dựng thêm các tính năng của ứng dụng web: chức năng hiển thị dữ liệu dạng biểu đồ với các thông kê theo ngày, tháng, năm; chức năng xuất báo cáo dữ liệu.

5. Ngày giao nhiệm vụ luận văn: 03/2021

6. Ngày hoàn thành nhiệm vụ: 08/2021

7. Họ và tên người hướng dẫn:

ThS. Bùi Quốc Bảo

Phản hướng dẫn

.....

Nội dung và yêu cầu LVTN đã được thông qua Bộ Môn.

*Tp.HCM, ngày 04 tháng 08 năm 2021*

**CHỦ NHIỆM BỘ MÔN**

**NGƯỜI HƯỚNG DẪN CHÍNH**

**PHẦN DÀNH CHO KHOA, BỘ MÔN:**

Người duyệt (chấm sơ bộ):.....

Đơn vị:.....

Ngày bảo vệ : .....

Điểm tổng kết: .....

Nơi lưu trữ luận văn: .....

## LỜI CẢM ƠN

Quá trình làm luận văn tốt nghiệp là giai đoạn quan trọng nhất trong quãng đời mỗi sinh viên. Luận văn tốt nghiệp sẽ là tiền đề giúp cho chúng em xác định được định hướng nghề nghiệp trong tương lai và trong quá trình thực hiện chúng em tích lũy được những kiến thức quý báu, những kỹ năng nghiên cứu. Đó sẽ là những hành trang cực kỳ quan trọng để cho chúng có thể vững bước trong giai đoạn đầu của quá trình lập nghiệp.

Làm luận văn ở trường đại học Bách Khoa đã khó, làm luận văn trong tình hình dịch bệnh đang diễn biến khá phức tạp ảnh hưởng không nhỏ đến quá trình học tập của sinh viên là một điều càng khó hơn. Chính vì vậy, em xin chân thành cảm ơn thầy Bùi Quốc Bảo, người đã đồng hành với em trong suốt thời gian làm luận văn vừa qua. Cảm ơn thầy đã giúp em định hướng nghề nghiệp, truyền đạt những kiến thức và những kinh nghiệm quý báu của bản thân. Những điều này không những giúp em hoàn thành tốt luận văn mà còn sẽ là hành trang tiếp bước em trong quá trình học tập và lập nghiệp sau này.

Em cũng xin gửi lời cảm ơn đến quý thầy cô ở khoa Điện-Điện tử trường Đại học Bách Khoa ĐHQG Tp HCM. Cảm ơn quý thầy cô đã tận tình chỉ dạy và trang bị cho em những kiến thức cần thiết trong suốt thời gian ngoài trên ghế giảng đường, làm nền tảng cho em có thể hoàn thành bài luận văn này.

Xin cảm ơn đến trường Đại học Bách Khoa. Thời gian bốn năm ở trường đại học có thể chẳng đáng gì khi so với thời gian cả cuộc đời, nhưng có thể đó là tất cả thời gian của một thời thanh xuân. Cảm ơn trường đã tạo cho em một môi trường học tập và rèn luyện bản thân. Cảm ơn vì đã là một phần trong tuổi trẻ của của tất cả các bạn sinh viên như em.

Cuối cùng em xin cảm ơn đến gia đình, bạn bè đã luôn chia sẻ, ủng hộ, động viên và giúp đỡ trong suốt quá trình học tập của bản thân.

Em xin trân trọng cảm ơn!

Tp. Hồ Chí Minh, ngày 04 tháng 08 năm 2021.

Sinh viên

Phạm Thái Hòa

Trần Ngọc Trâm

## TÓM TẮT LUẬN VĂN

Luận văn này trình bày về quá trình thiết kế và thực hiện nền tảng giám sát các hệ thống năng lượng mặt trời. Trong đó chú trọng 2 phần chính sau đây:

### Xây dựng Gateway:

- Lập trình các chức năng và kết nối các giao thức trên nền tảng FreeRTOS cho các thiết bị chạy vi điều khiển.
- Tìm hiểu, lập trình ứng dụng các giao thức (TCP, MQTT ...) và các chuẩn đóng gói dữ liệu (JSON ...) để kết nối đến IoT Server.
- Lập trình giao diện Desktop application để cài đặt và kiểm tra trạng thái của Gateway.

### Xây dựng ứng dụng web với các chức năng sau đây:

- Có khả năng đăng nhập, đăng xuất, tạo người dùng và phân quyền người dùng. Người dùng dùng tài khoản đăng nhập vào ứng dụng web có thể thực hiện chức năng thay đổi mật khẩu. Đối với tài khoản admin có khả năng tạo tài khoản cho người dùng, xoá tài khoản, thay đổi mật khẩu của tài khoản người dùng. Ứng dụng cũng cho phép người dùng lấy lại mật khẩu qua email đã đăng ký trong trường hợp người dùng quên mật khẩu.
- Có khả năng tạo Site – một đối tượng ảo tượng trưng cho một trạm năng lượng mặt trời. Site sẽ lưu trữ các thông tin về tên, địa điểm, chủ sở hữu, và công suất định của một trạm năng lượng mặt trời. Ứng dụng web có khả năng tạo ra và quản lý nhiều Site tương ứng với các trạm Solar khác nhau. Mỗi Site sẽ được phân bổ cho một tài khoản người dùng. Người dùng có thể truy cập các Site thuộc phân quyền của mình.
- Có khả năng tạo Gateway – một gateway ảo được liên kết với gateway thực lớp vật lý về mặt dữ liệu. Gateway được tạo sẽ thuộc về một Site cụ thể. Một Site có thể có nhiều Gateway. Khi gateway được tạo sẽ sinh ra một API Key, và API Key sẽ được cung cấp cho gateway thực. Với API Key gateway thực có khả năng gửi dữ liệu lên server bằng giao thức MQTT hoặc HTTP request.
- Có khả năng tạo Device – một đối tượng ảo tượng trưng cho các thiết bị vật lý bên dưới. Device có thể là một Inverter, một Energy Meter, hay một Weather Station. Device cũng sẽ thuộc về một gateway cụ thể. Một gateway sẽ quản lý nhiều Device.
- Sau khi tiến hành tạo Site, Gateway và Device, ứng dụng web sẽ có khả năng hiển thị, phân tích và thống kê các thông số của một hệ thống năng lượng mặt trời.

- Ứng dụng web có khả năng hiển thị dữ liệu dưới dạng biểu đồ, có khả năng thực hiện các tính toán từ các thông số của hệ thống solar, có khả năng xuất báo cáo dữ liệu.

## MỤC LỤC

1. GIỚI THIỆU .....	1
1.1 Tổng quan .....	1
1.2 Tình hình nghiên cứu trong và ngoài nước .....	2
1.2.1 Một số nền tảng giám sát năng lượng điện mặt trời .....	2
1.2.2 Đánh giá chung các nền tảng .....	8
1.3 Nhiệm vụ luận văn .....	8
1.3.1 Phạm vi luận văn .....	8
1.3.2 Nhiệm vụ luận văn: .....	8
2. CƠ SỞ LÝ THUYẾT .....	10
2.1 Tìm hiểu về nền tảng giám sát điện mặt trời.....	10
2.1.1 Cấu trúc chung của hệ thống .....	10
2.1.2 Các thiết bị đầu cuối thu thập dữ liệu .....	11
2.1.2.1 Biến tần .....	11
2.1.2.2 Đồng hồ đo năng lượng .....	13
2.1.2.3 Bộ quan trắc thời tiết.....	13
2.1.3 IoT Gateway .....	14
2.1.3.1 IoT Gateway là gì?.....	14
2.1.3.2 Chức năng của IoT Gateway .....	14
2.1.3.3 Kết nối trong các IoT gateway.....	15
2.1.4 Ứng dụng web giám sát hệ thống .....	16
2.2 Giao thức Modbus .....	17
2.2.1 Giới thiệu.....	17
2.2.2 Đặc điểm .....	17

2.2.3 Hoạt động .....	19
2.2.3.1 Hoạt động của Modbus RTU.....	19
2.2.3.2 Hoạt động của Modbus TCP .....	21
2.3 Giao thức MQTT.....	23
2.3.1 Giới thiệu.....	23
2.3.2 Đặc điểm .....	23
2.3.3 Hoạt động .....	24
2.3.3.1 Mô hình Publisher – Subscriber .....	24
2.3.3.2 Cấu trúc gói tin trong MQTT .....	25
2.4 Hệ điều hành thời gian thực .....	27
2.4.1 Giới thiệu.....	27
2.4.2 Đặc điểm .....	28
2.4.3 Hoạt động .....	29
2.4.3.1 Bộ lập lịch.....	30
2.4.3.2 Các dịch vụ thời gian thực.....	32
2.4.3.3 Đồng bộ và xử lý thông điệp .....	32
2.5 Tìm hiểu về ứng dụng web (Web App) .....	33
2.5.1 Ứng dụng web là gì?.....	33
2.5.2 Cách thức hoạt động của một ứng dụng web .....	34
2.5.3 Quá trình thiết kế một ứng dụng web .....	36
2.6 Tìm hiểu về ReactJS .....	38
2.6.1 Giới thiệu về ReactJS.....	38
2.6.2 Virtual DOM .....	39
2.6.3 React Component .....	40

2.6.4 Props và State .....	41
2.6.5 Vòng đời component trong React.....	43
2.6.6 React Hooks .....	44
2.7 Tìm hiểu về NodeJS .....	46
2.7.1 NodeJS là gì?.....	46
2.7.2 Framework ExpressJS.....	47
2.8 Tìm hiểu về cơ sở dữ liệu MongoDB .....	48
2.8.1 Giới thiệu về MongoDB.....	48
2.8.2 Các thuật ngữ hay sử dụng trong MongoDB .....	49
2.8.3 Một số câu lệnh cơ bản trên MongoDB .....	50
2.8.4 Ưu điểm và nhược điểm của MongoDB .....	51
3. THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG.....	52
3.1 Yêu cầu .....	52
3.2 Phân tích .....	52
3.3 Sơ đồ khái tổng quát.....	54
3.4 Sơ đồ mạch chi tiết .....	55
4. THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM.....	56
4.1 Sơ đồ khái tổng quát của hệ thống .....	56
4.2 Xây dựng khái IoT Gateway.....	57
4.2.1 Phần mềm trên IoT Gateway.....	57
4.2.1.1 Yêu cầu.....	57
4.2.1.2 Lưu đồ giải thuật.....	57
4.2.2 Phần mềm trên máy tính .....	63
4.2.2.1 Yêu cầu.....	63

4.2.2.2 Lưu đồ giải thuật .....	63
4.3 Xây dựng khối Context Broker .....	64
4.3.1 API tạo Entity .....	65
4.3.2 API truy cập Entity .....	66
4.3.3 API cập nhật Entity .....	67
4.3.4 API xoá Entity .....	68
4.3.5 API cấu hình gateway .....	70
4.3.6 API gửi dữ liệu .....	73
4.3.7 API truy cập dữ liệu .....	74
4.4 Xây dựng Web App .....	75
4.4.1 Mô hình quản lý và cách thức hoạt động của Web App .....	75
4.4.1.1 Mô hình quản lý của Web App .....	75
4.4.1.2 Quy trình hoạt động của Web App .....	76
4.4.2 Xây dựng ứng dụng back-end hỗ trợ tính năng User Authentication (đăng ký, đăng nhập) và Authorization (phân quyền người dùng). ....	76
4.4.2.1 Sự khác nhau giữa Authentication và Authorization .....	77
4.4.2.2 Token Based Authentication .....	77
4.4.2.3 Tiến hành xây dựng ứng dụng Node.js với tính năng user authentication và authorization.....	80
4.4.3 Xây dựng giao diện Web App .....	96
4.4.3.1 Xây dựng giao diện trang Login .....	96
4.4.3.2 Giao diện trang Site Management .....	99
4.4.3.3 Giao diện trang Account Management .....	103
4.4.3.4 Giao diện trang Datatype Editor .....	105
4.4.3.5 Giao diện trang Fleetview .....	107

4.4.3.6 Giao diện trang Site List.....	108
4.4.3.7 Giao diện trang Leaderboard .....	109
4.4.3.8 Giao diện trang Site View .....	111
4.4.3.9 Giao diện trang Site KPI .....	113
4.4.3.10 Giao diện trang Device List .....	116
4.4.3.11 Giao diện trang Charting Tool.....	119
4.4.3.12 Giao diện trang Availability .....	123
4.4.3.13 Giao diện Topology Analysis .....	123
4.4.3.14 Giao diện trang Budget Production Input.....	125
4.4.3.15 Giao diện trang Budget Insolation Input .....	127
4.4.3.16 Giao diện trang Data Report.....	129
4.4.4 Deploy Web App lên Linux Server .....	131
5. KẾT QUẢ THỰC HIỆN .....	132
5.1 IoT Gateway.....	132
5.1.1 Phân cứng .....	132
5.1.2 Phân mềm.....	132
5.1.3 Ứng dụng Desktop .....	133
5.1.4 Các bài thử nghiệm, đánh giá IoT gateway.....	134
5.1.4.1 Thử nghiệm với mạng đơn thiết bị .....	134
5.1.4.2 Thử nghiệm với mạng nhiều thiết bị.....	136
5.1.4.3 Thử nghiệm tốc độ tối đa .....	139
5.2 Web App .....	141
5.2.1 Kết quả đạt được.....	141
5.2.2 Một số hình ảnh của Web App .....	141

6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....	144
6.1 Kết luận.....	144
6.1.1 Ưu điểm.....	144
6.1.2 Nhược điểm .....	145
6.2 Hướng phát triển.....	146
6.2.1 Đối với IoT Gateway .....	146
6.2.2 Đối với ứng dụng web .....	146
7. TÀI LIỆU THAM KHẢO .....	147
8. PHỤ LỤC .....	149
8.1 Mô hình demo hệ thống .....	149

## DANH SÁCH HÌNH MINH HỌA

Hình 1-1 Công suất lắp đặt tích luỹ điện mặt trời ở các khu vực từ năm 2010-2030.....	1
Hình 1-2 Nền tảng điện mặt trời của SolarEdge.....	3
Hình 1-3 Ứng dụng mobile, web giám sát hệ thống của SolarEdge .....	3
Hình 1-4 Các dịch vụ của Azure IoT hỗ trợ phát triển các nền tảng giám sát điện mặt trời.....	4
Hình 1-5 Tổng quan hệ thống PPC/PV SCADA .....	5
Hình 1-6 Giao diện HMI giám sát hệ thống của PPC/PV SCADA .....	6
Hình 1-7 Tổng quan nền tảng giám sát điện mặt trời của MES-Engineering .....	7
Hình 1-8 Giao diện web giám sát hệ thống được MES-Engineering phát triển.....	7
Hình 2-1 Các thiết bị chính dùng trong thu thập dữ liệu hệ thống pin mặt trời (Từ trái sang: Biến tần, đồng hồ đo năng lượng, bộ quan trắc thời tiết) .....	11
Hình 2-2 Một số biến tần dùng trong hệ thống điện mặt trời .....	11
Hình 2-3 Phân loại các biến tần (Từ trái sang: biến tần chuỗi, biến tần vi mô, biến tần tối ưu hóa năng lượng) .....	12
Hình 2-4 Các loại đồng hồ đo trong hệ thống điện mặt trời.....	13
Hình 2-5 Các thiết bị quan trắc thời tiết trong hệ thống năng lượng mặt trời .....	13
Hình 2-6 Gateway đóng vai trò là cầu nối giữa biến và đám mây .....	14
Hình 2-7 Các kết nối chính dùng trong IoT Gateway (Từ trái sang: RS485, Wifi, Ethernet, Cellular) .....	15
Hình 2-8 Mô hình các thiết bị trong mạng modbus .....	18
Hình 2-9 Tương quan giữa Modbus và mô hình OSI .....	19
Hình 2-10 Mô hình Master Slave trong Modbus RTU .....	20
Hình 2-11 Cấu trúc gói tin Modbus RTU.....	20
Hình 2-12 : Mô hình Client – Server trong Modbus TCP.....	21
Hình 2-13 Khối PDU trong Modbus RTU .....	22

Hình 2-14 Cấu trúc gói tin của Modbus TCP .....	22
Hình 2-15 Quá trình hoạt động giữa hai Client (Publisher, Subscriber) và Broker .....	25
Hình 2-16 Cấu trúc gói tin MQTT Control .....	26
Hình 2-17 Fix Header trong gói tin MQTT Control .....	27
Hình 2-18 Ví dụ về multi-tasking trong các hệ thống thời gian thực .....	28
Hình 2-19 Các thành phần cơ bản của một RTOS.....	29
Hình 2-20 Các trạng thái hoạt động của task.....	30
Hình 2-21 Preemptive scheduling.....	31
Hình 2-22 non-Preemptive scheduling.....	32
Hình 2-23 Khái niệm về ứng dụng web .....	33
Hình 2-24 Cách thức hoạt động của một web app.....	35
Hình 2-25 Virtual DOM trong ReactJS.....	40
Hình 2-26 Ví dụ cách sử dụng state trong ReactJS.....	43
Hình 2-27 Ví dụ về cách sử dụng state bằng hook .....	45
Hình 3-1 Sơ đồ khái quát của Gateway Modbus Serial .....	54
Hình 3-2 Olimex STM32-E407 Cortex-M4 Dev Board .....	55
Hình 4-1 Sơ đồ khái quát của hệ thống .....	56
Hình 4-2 Quá trình khởi động của gateway .....	58
Hình 4-3 Các khái nhiệm vụ của gateway .....	59
Hình 4-4 Luồng dữ liệu của Gateway Modbus Serial.....	60
Hình 4-5 Lưu đồ giải thuật của khái MQTT Handler.....	61
Hình 4-6 Lưu đồ giải thuật của khái Modbus Uplink.....	62
Hình 4-7 Lưu đồ giải thuật của khái Modbus Downlink .....	63
Hình 4-8 Luồng dữ liệu của phần mềm trên máy tính .....	64

Hình 4-9 API tạo entity là một phần tử gốc .....	65
Hình 4-10 API tạo entity là con của một Entity khác .....	65
Hình 4-11 API truy cập entity bằng id .....	66
Hình 4-12 API truy cập các entity là con trực tiếp của một entity khác .....	66
Hình 4-13 API truy cập các Entity cùng chung một gốc .....	67
Hình 4-14 API truy cập các entity thông thuộc tính của entity đó .....	67
Hình 4-15 API cập nhật entity thông qua id .....	68
Hình 4-16 API xoá một entity bằng id .....	68
Hình 4-17 API xoá các entity là con trực tiếp của một entity khác .....	69
Hình 4-18 API xoá các entity cùng chung một gốc .....	69
Hình 4-19 API xoá các entity thông qua thuộc tính.....	70
Hình 4-20 API kiểm tra trạng thái provision .....	70
Hình 4-21 API cho phép provision .....	71
Hình 4-22 API kết thúc provision.....	71
Hình 4-23 API gửi gói provision .....	72
Hình 4-24 API truy cập dữ liệu gói tin provision .....	73
Hình 4-25 API gửi gói tin telemetry .....	73
Hình 4-26 API truy cập dữ liệu .....	74
Hình 4-27 Mô hình quản lý Web App .....	75
Hình 4-28 Cấu trúc chuỗi JWT .....	78
Hình 4-29 Thuật toán tạo ra signature trong chuỗi JWT.....	79
Hình 4-30 Flow của authentication với JWT .....	79
Hình 4-31 Kiến trúc ứng dụng Node authentication với JWT .....	82
Hình 4-32 Cấu trúc thư mục ứng dụng.....	83

Hình 4-33 Nội dung package.js của dự án .....	84
Hình 4-34 Giao diện trang đăng nhập .....	97
Hình 4-35 Giao diện cho phép người dùng lấy lại mật khẩu thông qua email .....	98
Hình 4-36 Giao diện cho phép người dùng đặt lại mật khẩu mới .....	99
Hình 4-37 Giao diện tạo một Site mới .....	100
Hình 4-38 Giao diện quản lý các Site .....	101
Hình 4-39 Giao diện tạo một gateway .....	101
Hình 4-40 Giao diện bổ nhiệm quyền truy cập site cho tài khoản user .....	102
Hình 4-41 Giao diện quản lý các gateway thuộc một site.....	102
Hình 4-42 Giao diện quản lý thông tin cấu hình của gateway .....	102
Hình 4-43 Giao diện mapping channel và tạo device .....	103
Hình 4-44 Giao diện quản lý các device đã tạo .....	103
Hình 4-45 Giao diện tạo mới một tài khoản user.....	104
Hình 4-46 Giao diện quản lý các tài khoản user.....	104
Hình 4-47 Giao diện reset password cho tài khoản user .....	105
Hình 4-48 Giao diện trang Datatype Editor .....	107
Hình 4-49 Giao diện trang Fleetview.....	108
Hình 4-50 Giao diện trang Site List chế độ đôi tượng .....	108
Hình 4-51 Giao diện trang Site List chế độ danh sách.....	109
Hình 4-52 Giao diện hiển thị trang Leaderboard.....	110
Hình 4-53 Giao diện xem thông tin chi tiết Production Completion Rate .....	110
Hình 4-54 Giao diện hiển thị trang Site View ở chế độ ngày.....	112
Hình 4-55 Giao diện hiển thị trang Site View ở chế độ tháng .....	112
Hình 4-56 Giao diện hiển thị trang Site View ở chế độ năm .....	113

Hình 4-57 Giao diện hiển thị trang Site KPI ở chế độ ngày.....	114
Hình 4-58 Giao diện hiển thị trang Site KPI ở chế độ tháng.....	115
Hình 4-59 Giao diện hiển thị trang Site KPI ở chế độ năm .....	115
Hình 4-60 Giao diện danh sách các Inverter .....	116
Hình 4-61 Giao diện danh sách các Energy Meter .....	117
Hình 4-62 Giao diện danh sách các Weather Station.....	117
Hình 4-63 Giao diện hiển thị chi tiết các thông số của một Inverter .....	118
Hình 4-64 Giao diện hiển thị chi tiết các thông số của Energy Meter .....	119
Hình 4-65 Giao diện hiển thị chi tiết các thông số của Weather Station .....	119
Hình 4-66 Giao diện chính của trang Charting Tool .....	120
Hình 4-67 Giao diện cấu hình các phân tích của trang charting tool.....	121
Hình 4-68 Giao diện hiển thị dữ liệu phân tích dưới dạng biểu đồ của trang Charting tool.....	121
Hình 4-69 Giao diện hiển thị dữ liệu phân tích dạng biểu đồ của trang Charting Tool.....	122
Hình 4-70 Giao diện hiển thị dữ liệu phân tích dạng bảng của trang Charting Tool.....	122
Hình 4-71 Giao diện trang Availability.....	123
Hình 4-72 Giao diện trang Availability.....	123
Hình 4-73 Giao diện trang Topology Analysis.....	124
Hình 4-74 Giao diện trang Topology Analysis.....	125
Hình 4-75 Giao diện trang Budget Production Input.....	126
Hình 4-76 Giao diện nhập dữ liệu sản lượng điện lý thuyết .....	127
Hình 4-77 Giao diện trang Budget Insolation Input .....	128
Hình 4-78 Giao diện nhập giá trị năng lượng bức xạ mặt trời lý thuyết .....	129
Hình 4-79 Giao diện trang Data Report .....	130
Hình 4-80 Giao diện file PDF Report .....	130

Hình 4-81 Giao diện truy cập vào linux server.....	131
Hình 4-82 Giao diện quản lý các ứng dụng NodeJs sử dụng PM2 .....	132
Hình 5-1 Giao diện ứng dụng desktop .....	133
Hình 5-2 Các thiết bị giả lập Modbus Slave (Từ trái sang: Coil Register Slave, Input Register Slave, Holding Register Slave).....	134
Hình 5-3 View communication trên giao diện Modbus Slave (bên trái), giá trị đọc được từ các Modbus Slave (bên phải).....	135
Hình 5-4 Giá trị đọc được từ SHT20 quan sát qua giao diện terminal của gateway .....	136
Hình 5-5 Các thiết bị thực được giả lập dữ liệu (Inverter SUN2000 V200R002, Energy Meter PM5350, Weather Station VSN800-14).....	137
Hình 5-6 Inverter simulator .....	137
Hình 5-7 Energy Meter simulator .....	138
Hình 5-8 Weather station simulator .....	138
Hình 5-9 Mô hình thời gian gói tin đi trong mạng Modbus. ....	139
Hình 5-10 Giao diện trang Fleetview.....	142
Hình 5-11 Giao diện trang Site View.....	142
Hình 5-12 Giao diện trang Site KPI.....	143
Hình 5-13 Giao diện trang Device List .....	143
Hình 8-1 Các kết nối trên gateway sau khi hoàn thành.....	149

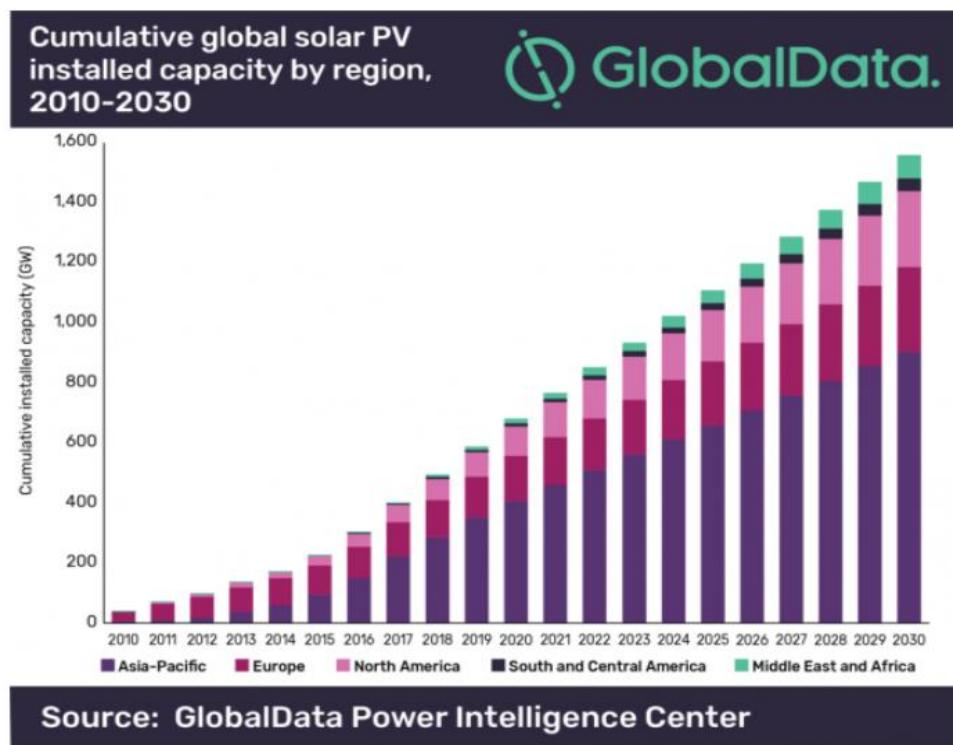
## **DANH SÁCH BẢNG SỐ LIỆU**

Bảng 1 Một số câu lệnh cơ bản trên MongoDB .....	50
Bảng 2 Phân tích các phương án cho Gateway Modbus Serial .....	52
Bảng 3 Danh sách những APIs cần thiết của web app.....	80
Bảng 4 Bảng thống kê kết quả phản hồi của Modbus Slave trong mạng đa thiết bị .....	138
Bảng 5 Bảng thống kê về thời gian trễ của giao thức MQTT .....	140
Bảng 6 Bảng thống kê tổng thời gian gói tin đi trong mạng Modbus và MQTT .....	140

## 1. GIỚI THIỆU

### 1.1 Tổng quan

Năng lượng mặt trời là nguồn năng lượng tái tạo phát triển nhanh nhất trên thế giới, tăng công suất trên toàn thế giới trung bình 40% mỗi năm. Nhiều công ty năng lượng đang mở rộng để cung cấp năng lượng mặt trời, đây là một trong những nguồn năng lượng tái tạo tiết kiệm năng lượng và sinh lợi nhất trên thị trường. Theo đánh giá của Hiệp hội năng lượng sạch Việt Nam, Việt Nam là một trong những quốc gia có ánh nắng mặt trời nhiều nhất trong bản đồ bức xạ mặt trời thế giới. Vì thế quy hoạch điện VII đã điều chỉnh đưa ra triển vọng và đặt kế hoạch khai thác điện mặt trời vào năm 2020 được khoảng 850MW; 4000MW vào năm 2025 và có thể khai thác khoảng 12000MW vào năm 2030. Và con số này đang có chiều hướng tăng lên. Trên thế giới, ngành công nghiệp cho lĩnh vực năng lượng mặt trời ở Trung Quốc, Hoa Kỳ, Áo Đô đã cho thấy một động lực phát triển mạnh mẽ, trong khi Nhật Bản và châu Âu tiếp tục có nhu cầu thị trường ổn định. Trung Đông và các thị trường mới nổi khác cũng đang tăng trưởng nhanh chóng.



Hình 1-1 Công suất lắp đặt tích luỹ điện mặt trời ở các khu vực từ năm 2010-2030

Với đặc điểm của các trạm năng lượng mặt trời đó là có độ phân tán cao, số lượng các tấm pin mặt trời là vô cùng nhiều, mặc dù có nhân viên ghi chép dữ liệu, đi kiểm tra nhưng sẽ không thể giám sát được trạng thái hoạt động của từng trạm, từng nhà máy trong thời gian thực. Vậy làm thế nào để có thể quản lý được các trạm năng lượng mặt trời, liệu rằng trạm có cung cấp đủ mức năng lượng? Làm thế nào để có thể phát hiện kịp thời các sự cố? Với các chủ đầu tư, điều họ quan tâm đó là làm sao để có doanh thu ổn định, kế hoạch đầu tư tối ưu là gì, đầu tư vào trạm nào, thiết bị nào để có thể sinh lời, doanh thu tốt nhất? Vì vậy, cần có giải pháp để có thể giải quyết, trả lời những câu hỏi trên.

Xuất phát từ những nhu cầu thực tế trên nên em quyết định chọn đề tài cho luận văn là xây dựng một nền tảng giám sát các hệ thống năng lượng điện mặt trời. Đề tài này không những là một thực tại khách quan mà nó còn đóng vai trò đặc biệt quan trọng ở hiện tại cũng như trong tương lai sau này. Các giải pháp giám sát năng lượng mặt trời chiếm một phần rất nhỏ trong tổng chi phí của một nhà máy điện mặt trời. Nhưng chúng mang lại lợi ích tài chính lớn đáng kể.

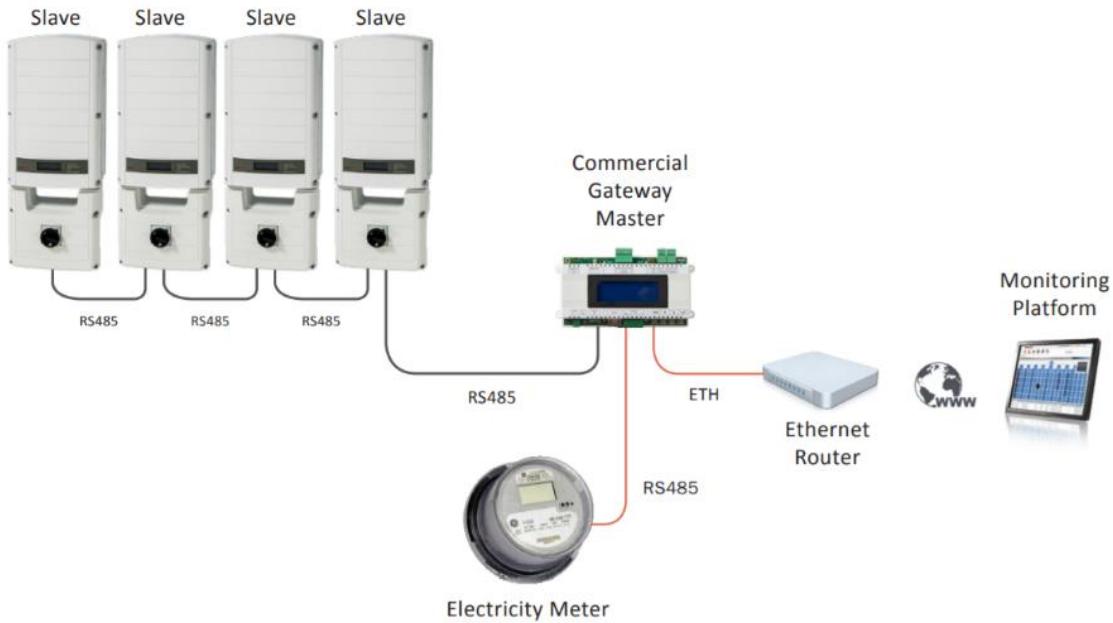
## 1.2 Tình hình nghiên cứu trong và ngoài nước

### 1.2.1 Một số nền tảng giám sát năng lượng điện mặt trời

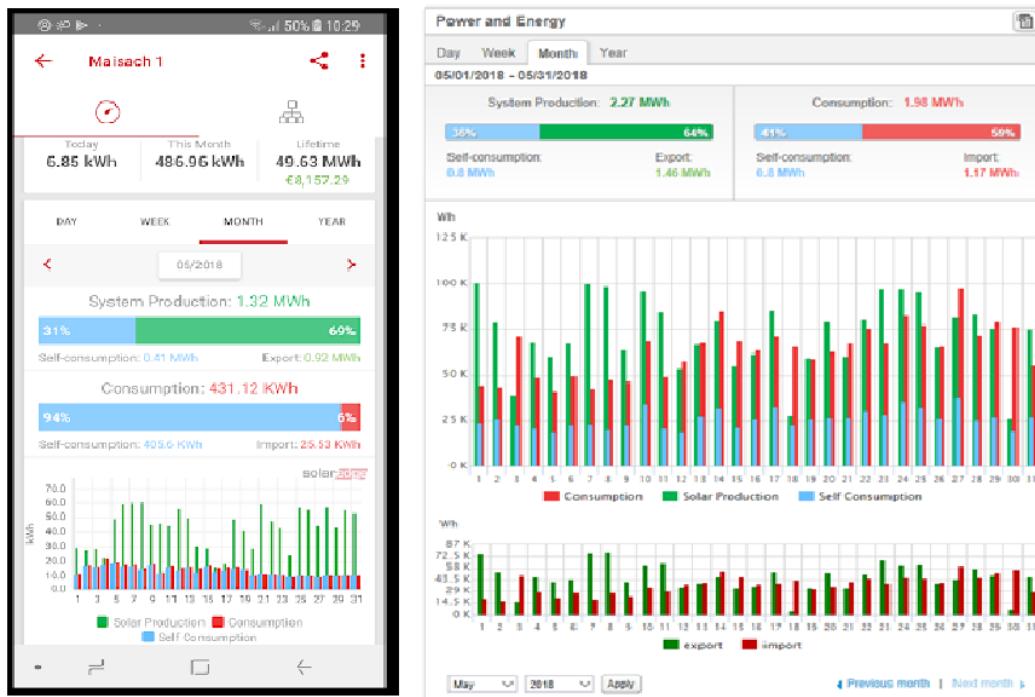
Trước khi xây dựng một nền tảng giám sát hệ thống điện mặt trời, ta cùng tham khảo, phân tích một số nền tảng trong và ngoài nước để đưa ra những đánh giá.

#### SolarEdge Monitoring Platform

SolarEdge [1] là công ty hàng đầu toàn cầu về năng lượng thông minh, cung cấp các giải pháp thương mại và dân dụng sáng tạo. SolarEdge cung cấp một nền tảng giám sát điện mặt trời bao gồm các thiết bị đầu cuối, thiết bị xử lý dữ liệu tại biên và ứng dụng web, mobile để giám sát hệ thống. Nền tảng này cung cấp khả năng giám sát hiệu suất năng lượng mặt trời nâng cao, cũng như đảm bảo năng suất. Thông qua việc phát hiện lỗi và cảnh báo nhanh chóng, cho dù đó là ở cấp độ mô-đun, chuỗi hay hệ thống, hệ thống đảm bảo bạn nhận được sản lượng tối ưu.



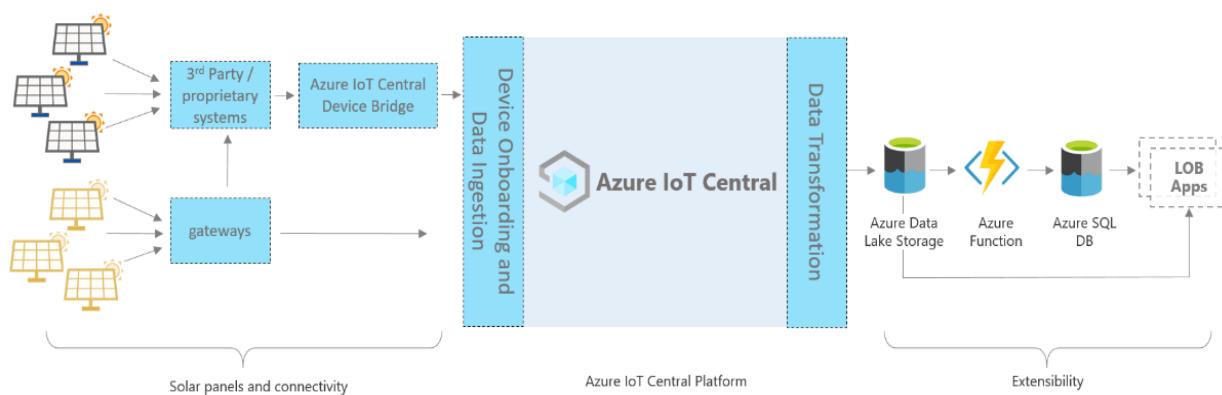
**Hình 1-2 Nền tảng điện mặt trời của SolarEdge**



**Hình 1-3 Ứng dụng mobile, web giám sát hệ thống của SolarEdge**

## Các nền tảng thuê dịch vụ đám mây

Các nền tảng thuê cơ sở hạ tầng, dịch vụ điện toán đám mây để đẩy nhanh phát triển hệ thống. Các dịch vụ đám mây được biến đến nhiều nhất trong loại hình này phải kể đến Google IoT Cloud, Amazon IoT Cloud, Microsoft Azure IoT. Cụ thể như các dịch vụ của Microsoft Azure IoT, họ cung cấp một nền tảng bao gồm: Azure IoT Edge đóng vai trò là một gateway thu thập dữ liệu từ các cảm biến theo các giao thức đã đặt ra, điểm đặc biệt trong việc cài đặt kết nối từ Gateway và các thiết bị đầu cuối là ta chỉ cần khai báo các thông tin của thiết bị cho Gateway thì Gateway sẽ tự động kết nối đến thiết bị và đọc các thông tin, dữ liệu về; Azure IoT Cloud [2] bao gồm các dịch vụ cầu nối chạy ngầm, giao diện hiển thị và các công cụ hỗ trợ phân tích, tính toán.



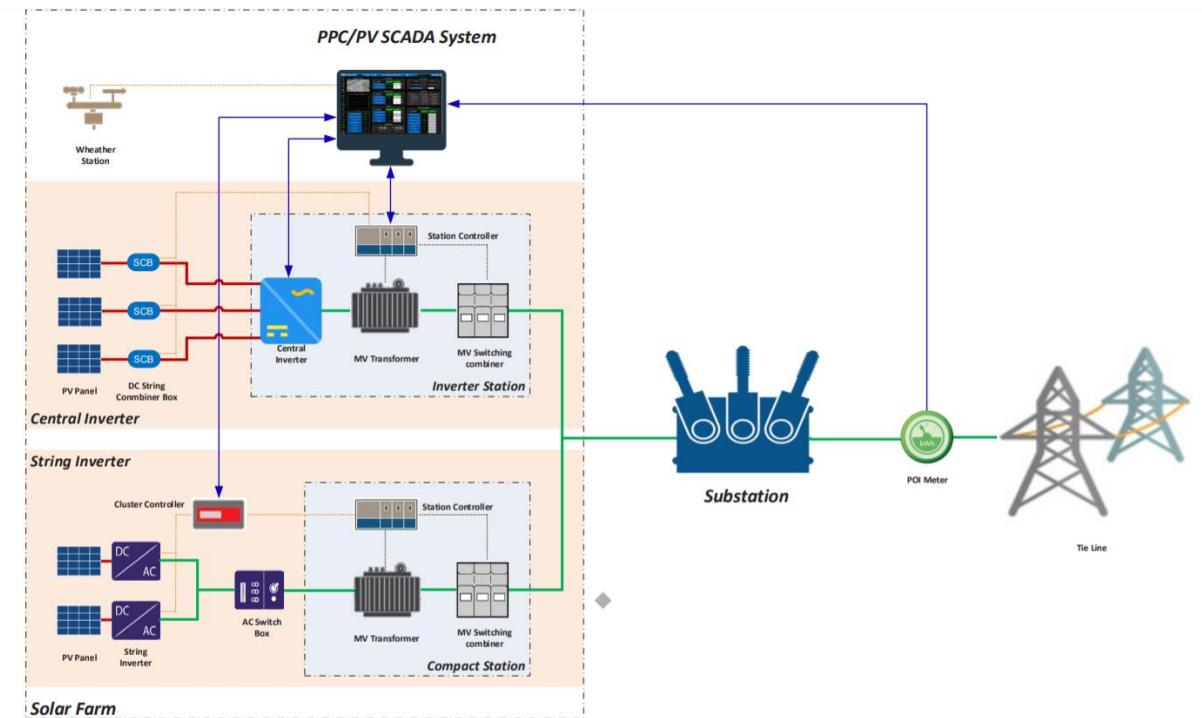
**Hình 1-4 Các dịch vụ của Azure IoT hỗ trợ phát triển các nền tảng giám sát điện mặt trời**

## Một số nền tảng điện mặt trời tại Việt Nam

Hiện nay, việc triển khai lắp đặt hệ thống giám sát năng lượng mặt trời ngày càng phổ biến. Khách hàng, chủ đầu tư khi quyết định trang bị cho gia đình, công trình của mình một hệ thống pin năng lượng mặt trời. Khách hàng luôn muốn được chủ động giám sát thông tin hệ thống của mình để thấy được hiệu quả, lợi ích kinh tế của công trình mà mình vừa đầu tư mang lại. Năm bắt được nhu cầu khách hàng và hiểu được mong muốn của các nhà đầu tư, ở Việt Nam đã cho ra đời nhiều giải pháp cho các hệ thống giám sát năng lượng mặt trời.

## Hệ thống PV SCADA & PPC

Hệ thống PV SCADA & PPC [3] của ATS thực hiện tất cả các chức năng thu thập, giám sát và kiểm soát dữ liệu của Nhà máy điện mặt trời. Tất cả các thông tin cần thiết liên quan tới cách thức hoạt động, tính toàn vẹn của thiết bị và bộ điều khiển, chức năng điều khiển tuần tự và chức năng báo động sẽ sẵn sàng ngay lập tức tại hệ thống vận hành. Hệ thống có tính năng mô-đun hóa và khả năng mở rộng, cho phép tùy chỉnh chức năng điều khiển nhà máy và cung cấp các chức năng cần thiết để đáp ứng nhu cầu về tính đa dạng cao của các yêu cầu kết nối lưới. Giao diện người-máy (HMI) trực quan hóa tất cả các giá trị đo được cục bộ và trong thời gian thực, cho phép quản lý vận hành kỹ thuật cho các nhà máy điện mặt trời tại chỗ.



**Hình 1-5 Tổng quan hệ thống PPC/PV SCADA**

Hệ thống PPC/PV SCADA chia làm 2 thành phần chính:

- Tại mỗi Inverter Station sẽ bao gồm station controller (gateway) để thu thập tất cả dữ liệu từ biến tần, biến áp MV, rơ le bảo vệ, đồng hồ đa năng, trạm thời tiết

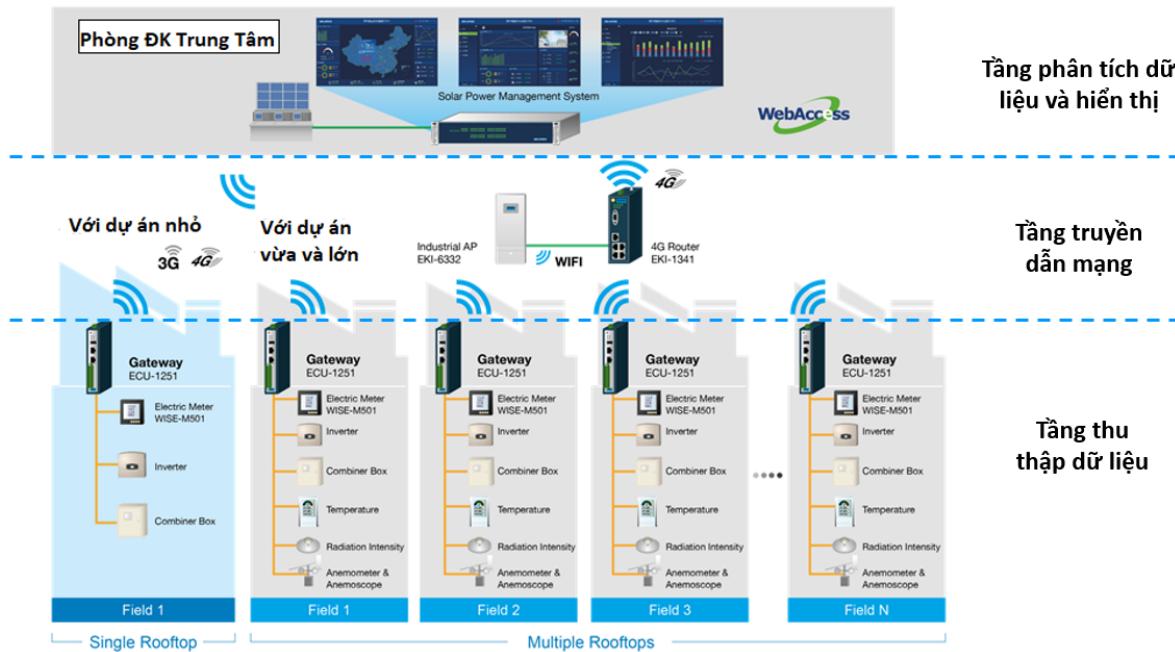
- Tại phòng điều hành: phòng điều khiển nhà máy điện và máy chủ SCADA, nơi tập trung dữ liệu được thu thập, xử lý dữ liệu, lưu trữ lịch sử dữ liệu, giám sát và kiểm soát toàn bộ nhà máy điện. Giao diện HMI hỗ trợ người vận hành thực hiện tất cả các chức năng giám sát và điều khiển nhà máy điện.



**Hình 1-6 Giao diện HMI giám sát hệ thống của PPC/PV SCADA**

### Giải pháp quản lý hệ thống năng lượng mặt trời của MES-Engineering

Giải pháp giám sát và quản lý hệ thống năng lượng mặt trời của MES-Engineering [4] Việt Nam là một giải pháp tổng thể để giám sát và quản lý các trạm năng lượng mặt trời với phần cứng và phần mềm tích hợp liền mạch hệ thống. Giải pháp giám sát và quản lý hệ thống năng lượng mặt trời của MES-Engineering Việt Nam có thể thực hiện việc thu thập dữ liệu và tối ưu chi phí. Đồng thời, giúp khách hàng có thể dễ dàng đưa ra quyết định và hệ thống luôn đảm bảo tính an toàn cho dữ liệu của khách hàng.



Hình 1-7 Tổng quan nền tảng giám sát điện mặt trời của MES-Engineering

Nền tảng điện mặt trời của MES-Engineering gồm 2 phần chính: cơ sở hạ tầng tại biên gồm các gateway thu thập dữ liệu từ các thiết bị đầu cuối; ứng dụng web, mobile để giám sát hệ thống.



Hình 1-8 Giao diện web giám sát hệ thống được MES-Engineering phát triển

### 1.2.2 Đánh giá chung các nền tảng

Các nền tảng tập trung xây dựng cơ sở hạ tầng biên để nâng cao hiệu suất xử lý dữ liệu tại biên cũng như xây dựng ứng dụng mobile, web trực quan hóa dữ liệu, đảm bảo trải nghiệm người dùng tốt, quá trình giám sát hệ thống đạt hiệu quả cao.

Từ các phân tích và đánh giá các nền tảng thực tế trong việc nâng cao xử lý dữ liệu tại biên và đảm bảo quá trình giám sát quản lý đạt hiệu quả cao, nhóm đề xuất thiết kế một nền tảng giám sát năng lượng điện mặt trời gồm 2 phần chính:

- Gateway đặt tại biên sẽ đảm bảo việc thu thập, xử lý lượng dữ liệu khổng lồ, giảm tải quá trình xử lý dữ liệu trên ứng dụng web.
- Ứng dụng web giúp trực quan hóa dữ liệu từ Gateway, thuận lợi cho người dùng tương tác với hệ thống, đảm bảo giám sát, quản lý hệ thống đạt hiệu quả cao.

## 1.3 Nhiệm vụ luận văn

### 1.3.1 Phạm vi luận văn

Xây dựng một nền tảng giám sát các hệ thống năng lượng mặt trời. Trong đó chú trọng 2 phần chính: xây dựng IoT Gateway thu thập dữ liệu theo các giao thức đặt ra và các phần mềm để kết nối Gateway với nền tảng ứng dụng trên server; thiết kế ứng dụng web để giám sát hệ thống, trực quan hóa dữ liệu thu được.

### 1.3.2 Nhiệm vụ luận văn:

Tù phạm vi đã xác định, nhóm phân tích các nội dung cần được thực hiện cho từng thành viên trong nhóm, cụ thể như sau:

#### **Trần Ngọc Trâm: Tìm hiểu và thiết kế phần mềm IoT Gateway**

Nội dung 1: Tìm hiểu các thành phần của IoT Gateway dùng trong hệ thống điện mặt trời.

Nội dung 2: Tìm hiểu và lựa chọn phần cứng phù hợp để phát triển hệ thống.

Nội dung 3: Tìm hiểu lý thuyết về các chuẩn giao tiếp, cách hoạt động, các thông số, ưu nhược điểm khi triển khai trên các hệ thống cụ thể.

Nội dung 4: Xây dựng firmware cho IoT Gateway, desktop application để cài đặt và kiểm tra trạng thái của Gateway

Nội dung 5: Thống nhất và chuẩn hoá các loại dữ liệu, cấu trúc gói tin để Gateway giao tiếp với ứng dụng web trên server.

Nội dung 6: Kiểm thử, đo đạc các thông số của hệ thống. Sửa lỗi phần mềm để phù hợp về hoạt động của hệ thống đã đề ra.

### **Phạm Thái Hoà: Tìm hiểu và thiết kế Solar Energy Monitoring Web App**

Nội dung 1: Tìm hiểu về mô hình hoạt động của một hệ thống IoT trong việc giám sát năng lượng.

- Tìm hiểu về sơ đồ khối của một hệ thống IoT
- Tìm hiểu về chức năng của từng khối, phương thức kết nối dữ liệu giữa các khối trong hệ thống, đặc biệt tìm hiểu kỹ về phần server trong hệ thống.

Nội dung 2: Tìm hiểu về cách xây dựng một ứng dụng web (Web App).

- Tìm hiểu ứng dụng web (web app) là gì? Cách thức hoạt động của một ứng dụng web.
- Các bước trong quá trình thiết kế một web app.
- Lựa chọn ngôn ngữ, thư viện, framework để xây dựng ứng web.

Nội dung 3: Tìm hiểu về ReactJS và NodeJS.

- Tìm hiểu về các khái niệm, cách sử dụng ReactJS và NodeJS trong việc xây dựng ứng dụng web.
- Ứng dụng kiến thức đã tìm hiểu tiến hành thực hành làm các ví dụ xây dựng các ứng dụng web nhỏ.

Nội dung 4: Sử dụng ReactJS để tiến hành thiết kế giao diện web.

- Sử dụng ReactJS tiến hành xây dựng giao diện của các trang trong ứng dụng web.

- Sử dụng kiến thức về HTML và CSS tiến hành điều chỉnh giao diện thêm phần chính chủ và đẹp mắt.

Nội dung 5: Sử dụng NodeJS để xây dựng backend server cho ứng dụng web

- Sử dụng Framework Express tiến hành dựng server cho ứng dụng web.
- Tìm hiểu về cách lưu trữ dữ liệu vào cơ sở dữ liệu sử dụng MongoDB.
- Liên kết back-end server với phần front-end (phần giao diện) đã được được tạo.

Nội dung 6: Tiến hành deploy ứng dụng web.

- Tìm hiểu về các bước để deploy ứng dụng web ra Internet.

Nội dung 7: Kết nối với dữ liệu thực được gửi lên từ gateway.

- Gateway tiến hành gửi dữ liệu lên server qua giao thức MQTT hoặc HTTP.
- Kiểm tra việc kết nối và hiển thị dữ liệu.
- Chỉnh sửa các lỗi xuất hiện.
- Đánh giá kết quả thực hiện và tiến hành viết báo cáo.

## 2. CƠ SỞ LÝ THUYẾT

### 2.1 Tìm hiểu về nền tảng giám sát điện mặt trời

#### 2.1.1 Cấu trúc chung của hệ thống

Một nền tảng giám sát điện mặt trời có thường có 3 phần chính:

- Các thiết bị đầu cuối: các thiết bị nhúng thu thập dữ liệu từ hệ thống pin mặt trời.
- IoT Gateway: nhận các gói tin từ các thiết bị đầu cuối, làm cầu nối để chuyển dữ liệu tại biên lên ứng dụng web và ngược lại.
- Ứng dụng web: Nhận dữ liệu từ Gateway, phân tích, tính toán các thông số để hiển thị trên giao diện người dùng.

### 2.1.2 Các thiết bị đầu cuối thu thập dữ liệu

Một hệ thống điện mặt trời thường có 3 thiết bị thu thập dữ liệu chính: Biến tần (Inverter), Đồng hồ đo năng lượng (Energy Meter), bộ quan trắc thời tiết (Weather Station).



**Hình 2-1 Các thiết bị chính dùng trong thu thập dữ liệu hệ thống pin mặt trời (Từ trái sang: Biến tần, đồng hồ đo năng lượng, bộ quan trắc thời tiết)**

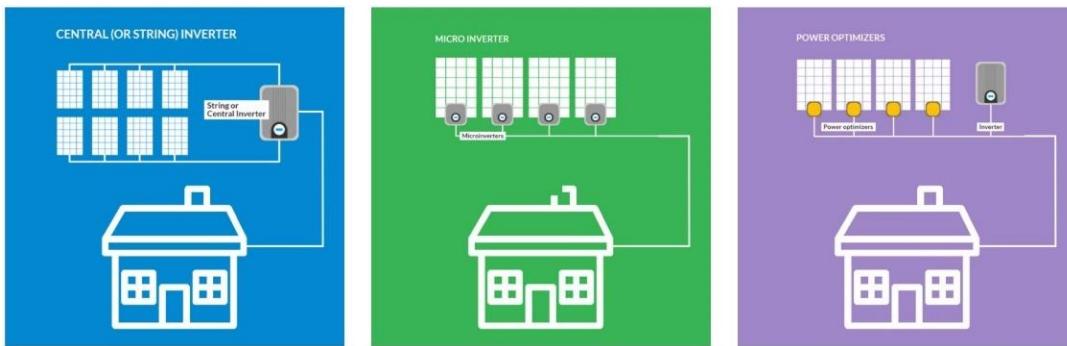
#### 2.1.2.1 Biến tần

Bộ biến tần (inverter) năng lượng mặt trời là bộ chuyển đổi dòng điện (DC) trực tiếp tạo ra bởi tấm pin mặt trời thành điện xoay chiều (AC) và hoà vào lưới điện xoay chiều của Quốc gia. Sau pin năng lượng mặt trời, biến tần là thiết bị quan trọng nhất trong hệ thống năng lượng mặt trời.



**Hình 2-2 Một số biến tần dùng trong hệ thống điện mặt trời**

## Phân loại biến tần



**Hình 2-3 Phân loại các biến tần (Từ trái sang: biến tần chuỗi, biến tần vi mô, biến tần tối ưu hóa năng lượng)**

- **Biến tần chuỗi (String Inverter):** là loại có hiệu quả về chi phí nhất và thường được các nhà lắp đặt điện mặt trời tại Việt Nam sử dụng. Các công ty sẽ xem xét các điều kiện mái nhà của bạn nếu không có bóng râm che khuất (*như cây cối, nhà cao tầng...*) thì họ sẽ đề xuất lựa chọn cho bạn. Việc các tấm pin bị bóng râm che ánh sáng mặt trời có thể làm ảnh hưởng đáng kể đến hiệu quả vận hành của biến tần chuỗi.
- **Biến tần vi mô (Micro-inverter):** Ngược lại với nguyên lý cơ bản của biến tần chuỗi, loại này sẽ hoạt động riêng rẽ với từng tấm pin, tức là bạn có bao nhiêu tấm thì sẽ đi kèm với bấy nhiêu bộ biến tần vi mô. Micro-inverter được xem là loại đắt nhất nhưng có hiệu quả vận hành tốt nhất.
- **Biến tần tối ưu hóa năng lượng (Power Optimizer):** cung cấp nhiều lợi ích khá giống với biến tần vi mô, nhưng chúng có xu hướng rẻ hơn một chút. Nó sẽ tối ưu hóa điện thường được coi là dạng ở “giữa” không quá đắt cũng không quá rẻ, hiệu quả hoạt động không quá cao cũng không quá thấp.

### 2.1.2.2 Đồng hồ đo năng lượng



**Hình 2-4 Các loại đồng hồ đo trong hệ thống điện mặt trời**

Đồng hồ đo năng lượng thường được lắp đặt tại đầu ra của các biến tần, được dùng để đo năng lượng, công suất được tạo ra bởi hệ thống pin mặt trời và sản lượng điện bán ra cho lưới điện quốc gia.

Các đồng hồ hoạt động bằng cách đo năng lượng theo 2 chiều: đo năng lượng tiêu thụ từ lưới điện và đo năng lượng dư thừa xuất ngược trở lại lưới điện. Đây là phần thiết bị cho phép nhận được các khoản phí từ việc bán sản lượng điện dư thừa cho lưới điện quốc gia.

### 2.1.2.3 Bộ quan trắc thời tiết



**Hình 2-5 Các thiết bị quan trắc thời tiết trong hệ thống năng lượng mặt trời**

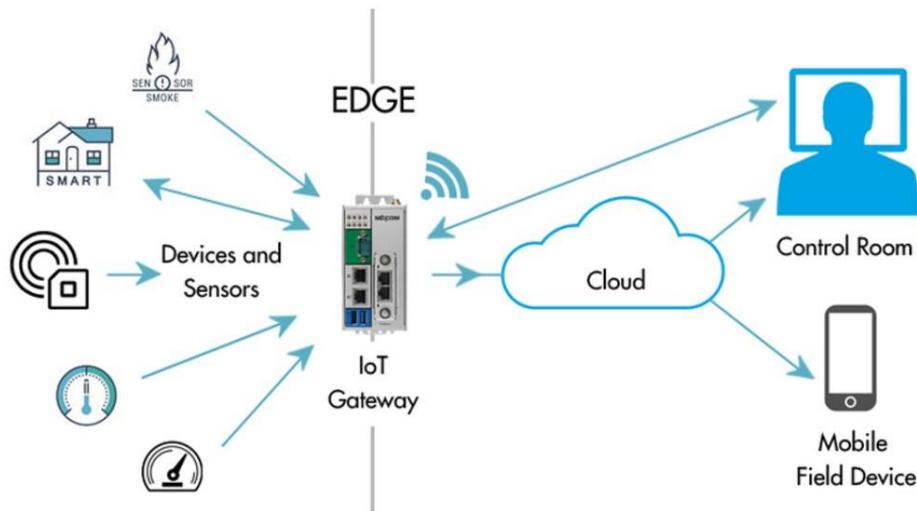
Trạm thời tiết quan trắc năng lượng mặt trời gồm các cảm biến: pyranometer quan trắc bức xạ mặt trời, temperature sensor quan trắc nhiệt độ mặt sau tấm pin, ambient temperature sensor quan trắc nhiệt độ không khí, ngoài ra có thể có thêm cảm biến đo tốc độ gió và hướng gió, cảm biến đo lượng mưa.

Việc quan trắc các thông số trên tại khu vực lắp đặt các tấm pin, giúp người sử dụng có thể quản lý được hiệu suất chuyển đổi năng lượng, tối ưu hóa năng lượng hấp thụ của các tấm pin cũng như có các điều chỉnh, đánh giá cần thiết.

### 2.1.3 IoT Gateway

#### 2.1.3.1 IoT Gateway là gì?

IoT Gateway là cầu nối giữa các thiết bị IoT (cảm biến, thiết bị, hệ thống) và mạng internet. Dữ liệu được truyền lên mạng internet sẽ đi qua gateway này. Nó tạo ra một cầu nối giữa các cảm biến, thiết bị IoT với Internet. IoT Gateway tổng hợp tất cả dữ liệu, dịch các giao thức của cảm biến và xử lý dữ liệu trước khi gửi.



**Hình 2-6 Gateway đóng vai trò là cầu nối giữa biên và đám mây**

#### 2.1.3.2 Chức năng của IoT Gateway

Tùy vào ứng dụng sử dụng mà IoT Gateway có các chức năng chính sau:

- Cầu nối giao tiếp giữa các thiết bị biên và đám mây
- Tính năng kết nối mạng và lưu trữ lịch sử dữ liệu ngắn hạn.
- Tổng hợp dữ liệu thô, xử lý trước dữ liệu, làm sạch, lọc và tối ưu hóa.
- Quản lý cấu hình thiết bị.
- Bảo mật – quản lý truy cập người dùng và các tính năng bảo mật mạng.

- Trực quan hóa dữ liệu và phân tích dữ liệu cơ bản thông qua các ứng dụng của IoT gateway
- Chẩn đoán hệ thống để đưa ra các giải pháp kịp thời để giảm thiệt hại cho hệ thống khi có sự cố xảy ra.

#### 2.1.3.3 Kết nối trong các IoT gateway



**Hình 2-7 Các kết nối chính dùng trong IoT Gateway (Từ trái sang: RS485, Wifi, Ethernet, Cellular)**

Những giao thức phổ biến trong các IoT Gateway dùng trong hệ thống điện mặt trời hiện nay:

- **RS485:** Có tốc độ truyền tương đối cao và ổn định, tuy nhiên, RS485 là giao thức có dây nên chỉ phù hợp để kết nối các trạm trong mạng cục bộ.
- **Wifi:** Wifi là giao thức thường được sử dụng để truyền hình ảnh, âm thanh nhờ băng thông lớn. Tuy nhiên, Wifi có phạm vi truyền sóng nhỏ và công suất tiêu thụ lớn (dây), điện năng tiêu thụ, tốc độ truyền tải dữ liệu, độ tin cậy và bảo mật. Giao thức truyền thông tin có rất nhiều với nhiều chủng loại thiết bị khác nhau dành cho các nhu cầu khác nhau.
- **Ethernet:** là công nghệ truyền thông dùng để kết nối các mạng LAN cục bộ, có tốc độ cao hơn RS485, ít bị gián đoạn, độ bảo mật cao, tuy nhiên, giao thức này đòi hỏi các thiết bị quản lý trên nền tảng mạng và việc triển khai giao thức này lên các thiết bị nhúng khá phức tạp về phần cứng và phần mềm so với RS485.
- **Các công nghệ truyền thông không dây Cellular Network:** là công nghệ truyền thông dựa trên mạng di động (GPRS, 3G, 4G, 5G) được cung cấp bởi các nhà mạng, hoạt động trên các băng tần được cấp phép, có tốc độ truyền tải dữ liệu cao. Với 4G băng thông rộng hơn, tốc độ nhanh hơn, người dùng có thể tải và truyền lên hình ảnh động chất lượng cao. Hiện nay 4G đã được triển khai rộng rãi trong mạng Cellular IoT với 2 dạng chính là: LTE-M và NB-IoT. Tuy nhiên việc triển khai 4G trên các thiết bị nhúng cũng có hạn chế như:

tăng chi phí để đăng ký các gói data 4G từ nhà mạng, phụ thuộc vào sự ổn định của dịch vụ mà nhà mạng cung cấp ...

Từ đó có thể thấy, mỗi giao thức đều có điểm mạnh và yếu riêng trong từng điều kiện môi trường. Việc thu thập dữ liệu cảm biến, tổng hợp và truyền tải dữ liệu với một giao thức đơn thuần không thể đáp ứng được. Thiết bị Wifi tốc độ cao nhưng khoảng cách hạn chế và tiêu hao năng lượng lớn, các thiết bị có khoảng cách rộng như 4G thì phải phụ thuộc vào phạm vi phủ sóng và độ ổn định của dịch vụ nhà mạng cung cấp, thiết bị RS485 có tốc độ cao và ổn định nhưng chỉ có thể dùng trong các kết nối tại biên, thiết bị Ethernet có tốc độ cao, đường truyền ổn định hơn các thiết bị wifi khi dùng trong mạng LAN cục bộ.

Dựa trên đặc trưng của các giao thức trên, các kết nối trong hệ thống pin mặt trời, để đảm bảo một hệ thống cân bằng với các thông số khoảng cách, năng lượng, tốc độ và các đặc trưng về một IoT Gateway, trong đề tài sẽ triển khai 2 loại kết nối chính:

- **RS485:** thích hợp dùng trong các ứng dụng có độ trễ thấp, trong mạng nội bộ. Trong đề tài sẽ thực hiện triển khai giao thức Modbus Serial dựa trên nền tảng vật lý của RS485 để thuận tiện hơn trong việc quản lý, điều khiển và thu thập dữ liệu từ các thiết bị. Ngoài ra, có thể mở rộng mạng lưới các thiết bị nhờ vào khả năng tương thích với các thiết bị bên ngoài theo tiêu chuẩn Modbus.
- **Ethernet:** thích hợp trong các ứng dụng cần đảm bảo đường truyền ổn định, cung cấp mức độ bảo mật và kiểm soát mạng tốt hơn mạng LAN không dây. Trong đề tài sẽ thực hiện triển khai giao thức MQTT dựa trên nền tảng vật lý của Ethernet để thuận tiện hơn trong việc đảm bảo đường truyền kết nối mạng ổn định.

#### 2.1.4 Ứng dụng web giám sát hệ thống

Ngày nay với việc phát triển mạnh của kỹ nguyên kỹ thuật số, áp dụng vào ngành năng lượng tái tạo. Nay giờ, các kỹ sư và công nhân vận hành các hệ thống năng lượng mặt trời không còn phải áp dụng các phương pháp thủ công dùng chuẩn đoán, bảo trì hệ thống nữa. Thay vào đó, là sự thay thế của các hệ thống điều khiển – giám sát năng lượng điện mặt trời. Giúp chúng ta phân tích và đánh giá và đưa ra cảnh báo tình trạng của hệ thống. Việc xây dựng một ứng dụng web giám sát các hệ thống năng lượng mặt trời là một điều cần thiết giúp trực quan hoá dữ liệu thu thập. Từ đó, khách hàng, chủ đầu tư khi quyết định trang bị cho gia đình, công trình của mình một hệ thống pin

năng lượng mặt trời được chủ động giám sát thông tin hệ thống của mình để thấy được hiệu quả, lợi ích kinh tế của công trình mà mình vừa đầu tư mang lại.

Các yêu cầu được đặt ra cho một ứng dụng web giám sát hệ thống:

- Thu thập và giám sát các chỉ số từ các thiết bị phân tán trong hệ thống.
- Phân tích, xử lý dữ liệu.
- Lưu trữ dữ liệu phục vụ cho việc kiểm tra, rà soát, xuất báo cáo hệ thống.
- Tạo báo cáo theo yêu cầu hoặc định kỳ.
- Chuẩn đoán, xử lý sự cố và quản lý từ xa.
- Các báo cáo được tổng hợp để phân tích xử lý dữ liệu của các dãy pin mặt trời. Từ đó, tìm ra được dãy pin mặt trời có hiệu suất thấp để tiến hành công tác bảo trì, bảo dưỡng hợp lý.

## 2.2 Giao thức Modbus

### 2.2.1 Giới thiệu

Modbus do Modicon (hiện nay thuộc Schneider Electric) [5] phát triển năm 1979, là một phương tiện truyền thông với nhiều thiết bị thông qua một cặp dây xoắn đơn. Ban đầu, nó hoạt động trên RS232, nhưng sau đó nó sử dụng cho cả RS485 để đạt tốc độ cao hơn, khoảng cách dài hơn, và mạng đa điểm (multi-drop). Modbus đã nhanh chóng trở thành tiêu chuẩn thông dụng trong ngành tự động hóa, và Modicon đã cho ra mắt công chúng như một protocol miễn phí.

### 2.2.2 Đặc điểm

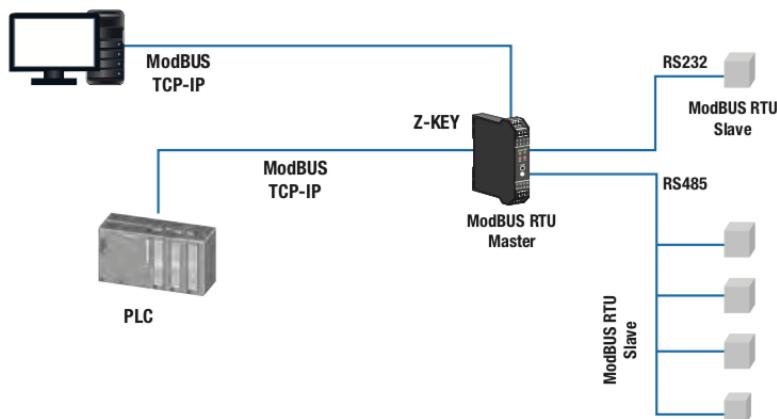
Modbus là một hệ thống “Master - Slave”, “Master” được kết nối với một hay nhiều “Slave”. “Master” thường là một PLC, PC, DCS, hay RTU. “Slave” Modbus thường là các thiết bị hiện trường, tất cả được kết nối với mạng trong cấu hình multi-drop.

Các thiết bị trên mạng Modbus không thể tạo ra kết nối, chúng chỉ có thể phản ứng. Nói cách khác, chúng “lên tiếng” chỉ khi được “nói tới”. Giao thức Modbus hỗ trợ tối đa 247 thiết bị Slave được kết nối tiếp vào Master (không qua các thiết bị chuyển tiếp). Với cổng kết nối RS232, Modbus có khoảng cách truyền tối đa là 200m; với cổng kết nối RS485 khoảng cách này là 1200m.

Nguyên tắc Master – Slave có những đặc điểm sau:

- Mỗi lần chỉ có một Master được kết nối mạng.
- Chỉ có Master mới có thể bắt đầu giao tiếp và gửi yêu cầu cho các slave.
- Modbus có thể giải quyết từng Slave, bằng cách sử dụng địa chỉ cụ thể của nó; hoặc tất cả các Slave đồng thời sử dụng địa chỉ 0.
- Slave chỉ có thể được gửi câu trả lời cho Master.
- Các Slave không thể bắt đầu giao tiếp với Master hoặc với các Slave khác.

**Gateway Modbus TCP-IP - Modbus RTU, 2 Master ports (2)**



**Hình 2-8 Mô hình các thiết bị trong mạng modbus**

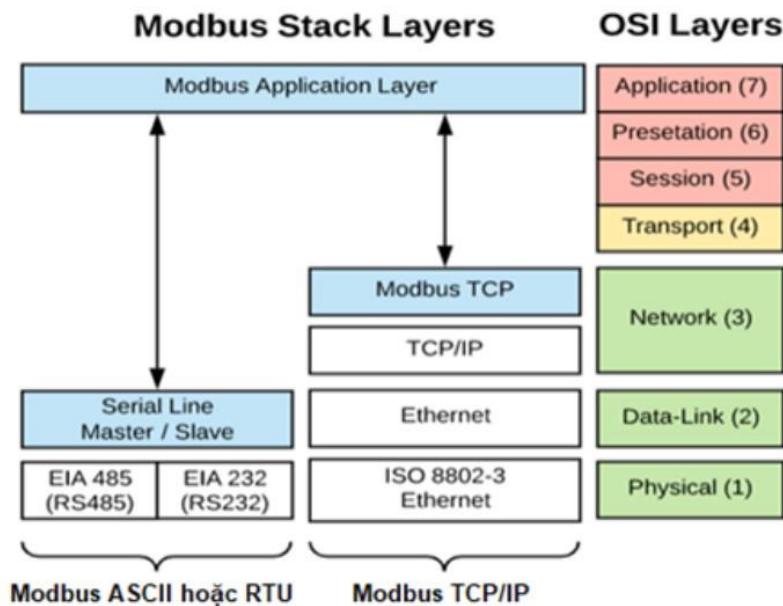
Hiện nay, có 03 chuẩn Modbus đang được sử dụng phổ biến trong công nghiệp - tự động hóa là: Modbus RTU, Modbus ASCII, Modbus TCP. Tất cả thông điệp được gửi dưới cùng một format. Sự khác nhau duy nhất giữa 3 loại Modbus là cách thức thông điệp được mã hóa. Cụ thể:

Modbus ASCII: Mọi thông điệp được mã hóa bằng hexadeci-mal, sử dụng đặc tính ASCII 4 bit. Đối với mỗi một byte thông tin, cần có 2-byte truyền thông, gấp đôi so với Modbus RTU hay Modbus TCP. Tuy nhiên, Modbus ASC II chậm nhất trong số 3 loại protocol, nhưng lại thích hợp khi modem điện thoại hay kết nối sử dụng song radio do ASCII sử dụng các tính năng phân định thông điệp. Do tính năng phân định này, mọi rắc rối trong phương tiện truyền dẫn sẽ không làm thiết bị nhận dịch sai thông tin. Điều này quan trọng khi đề cập đến các modem chậm, điện thoại di động, kết nối ồn hay các phương tiện truyền thông khó tính khác.

Modbus RTU: Modbus RTU là một giao thức nối tiếp đơn giản có thể truyền qua công nghệ UART truyền thông. Dữ liệu được mã hóa theo hệ nhị phân, truyền theo byte 8 bit, mỗi lần 1-bit với tốc độ baud dao động từ 1200bit/s – 115200 bit/s. Các thiết bị Modbus RTU chỉ hỗ trợ tốc độ lên tới 38400 bit/s.

Modbus TCP: Modbus TCP đơn giản là Modbus qua Ethernet. Thay vì sử dụng thiết bị này cho việc kết nối với các thiết bị tó, do đó các địa chỉ IP được sử dụng. Với Modbus TCP, dữ liệu Modbus được tóm lược đơn giản trong một gói TCP/IP. Do đó, bát cứ mạng Ethernet hỗ trợ TCP/IP sẽ ngay lập tức hỗ trợ Modbus TCP.

Trong mô hình mạng OSI, Modbus được triển khai ở tầng ứng dụng dựa trên nền vật lý RS232 hoặc RS485 đối với Modbus RTU và Ethernet đối với Modbus TCP.



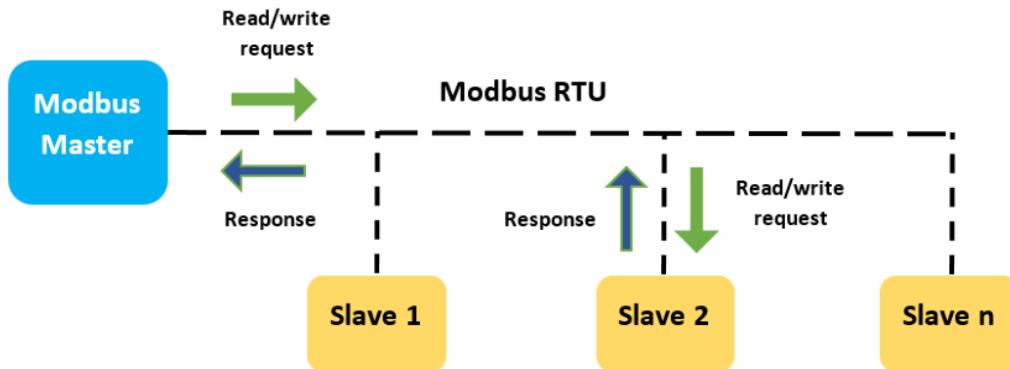
**Hình 2-9 Tương quan giữa Modbus và mô hình OSI**

## 2.2.3 Hoạt động

### 2.2.3.1 Hoạt động của Modbus RTU

Modbus RTU hoạt động dựa trên nguyên tắc Master – Slave tức là một bên nhận (Master) và một bên truyền tín hiệu (Slave) thông qua địa chỉ thanh ghi. Phương thức truyền của Modbus RTU bằng đường truyền vật lý RS232 hoặc RS485.

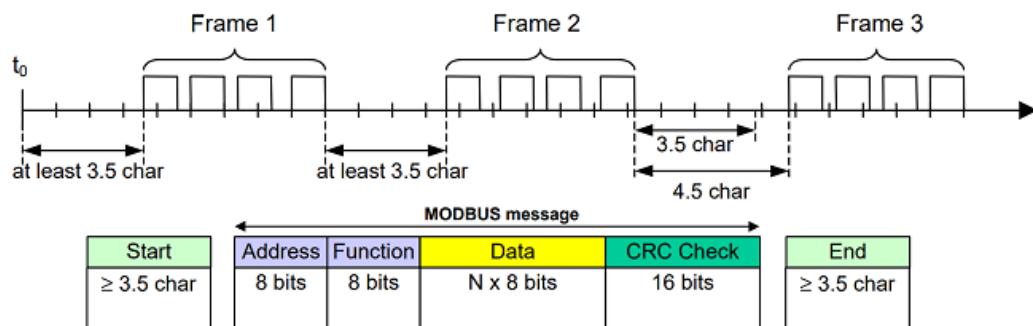
Khi một chủ Modbus muốn có thông tin từ thiết bị, chủ sẽ gửi một thông điệp về dữ liệu cần, tóm tắt dò lỗi tới địa chỉ thiết bị. Mọi thiết bị khác trên mạng sẽ nhận thông điệp này nhưng chỉ có thiết bị nào được chỉ định mới có phản ứng.



**Hình 2-10 Mô hình Master Slave trong Modbus RTU**

Khung truyền của giao thức Modbus RTU [6] xây dựng trên giao thức truyền thông nối tiếp UART và nội dung của các tin nhắn Modbus được xây dựng bằng nhiều gói tin ghép lại.

Một bản tin Modbus RTU bao gồm: 1-byte địa chỉ - 1-byte mã hàm – n-byte dữ liệu – 2-byte CRC như hình ở dưới:



**Hình 2-11 Cấu trúc gói tin Modbus RTU**

Chức năng và vai trò cụ thể như sau:

- Byte địa chỉ: xác định thiết bị mang địa chỉ được nhận dữ liệu (đối với Slave) hoặc dữ liệu nhận được từ địa chỉ nào (đối với Master). Địa chỉ này được quy định từ 0 – 254.

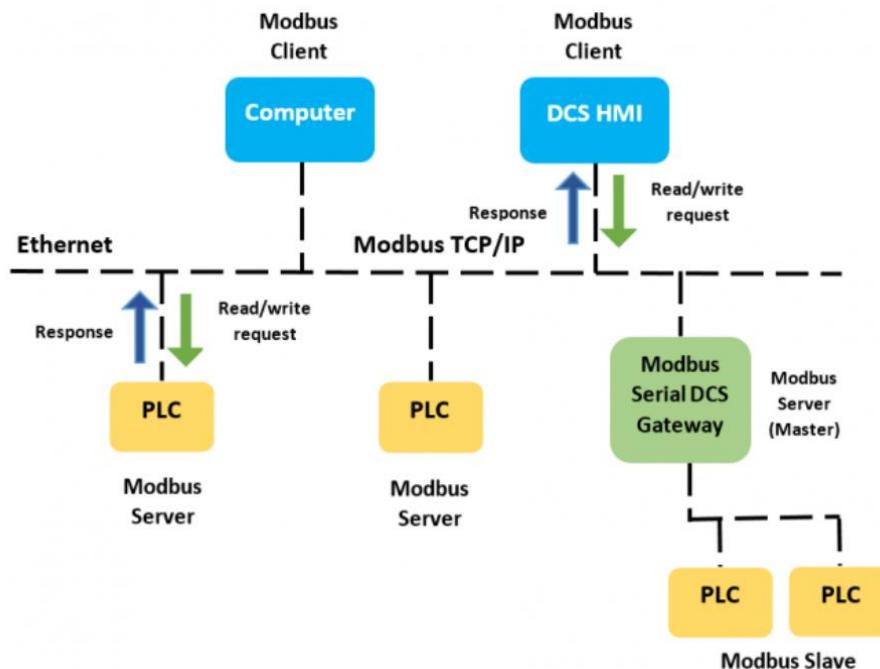
- Byte mã hàm: được quy định từ Master, xác định yêu cầu dữ liệu từ thiết bị Slave. Ví dụ mã 01: đọc dữ liệu lưu trữ dạng Bit, 03: đọc dữ liệu tức thời dạng Byte, 05: ghi dữ liệu 1-bit vào Slave, 15: ghi dữ liệu nhiều bit vào Slave ...
- Byte dữ liệu: xác định dữ liệu trao đổi giữa Master và Slave.
  - Đọc dữ liệu:
 

Master: 2-byte địa chỉ dữ liệu – 2-byte độ dài dữ liệu  
 Slave: 2-byte địa chỉ dữ liệu – 2-byte độ dài dữ liệu – n-byte dữ liệu đọc được
  - Ghi dữ liệu:
 

Master: 2-byte địa chỉ dữ liệu – 2-byte độ dài dữ liệu – n-byte dữ liệu cần ghi  
 Slave: 2-byte địa chỉ dữ liệu – 2-byte độ dài dữ liệu
- Byte CRC: 2-byte kiểm tra lỗi của hàm truyền. cách tính giá trị của Byte CRC 16-Bit

#### 2.2.3.2 Hoạt động của Modbus TCP

Modbus TCP hoạt động dựa trên mô hình client – server, trong mô hình này quan hệ giữa chủ nô bị đảo ngược. Gateway sẽ đóng vai trò là client, sẽ request đến các thiết bị Modbus (đóng vai trò là server) và chờ phản hồi từ server



**Hình 2-12 : Mô hình Client – Server trong Modbus TCP**

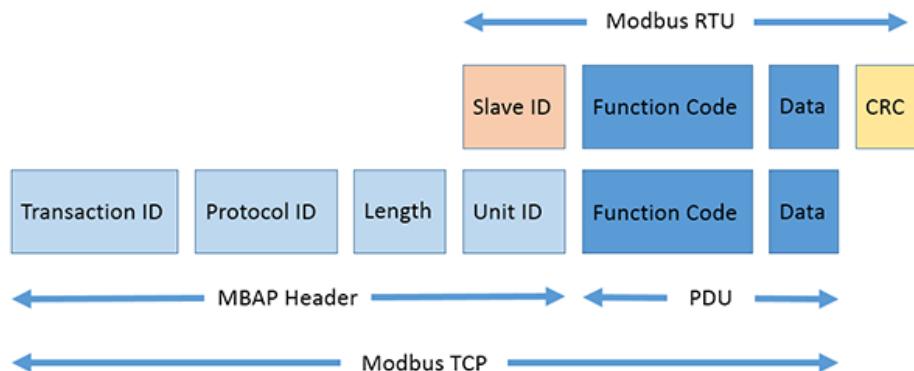
Modbus TCP/IP sử dụng truyền thông TCP/IP để đóng gói khói PDU của Modbus RTU vào gói tin TCP/IP, cho phép định tuyến việc truyền gói tin, hỗ trợ nhiều thiết bị hơn, đồng thời tăng khoảng cách truyền tin.

Modbus TCP/IP sử dụng khói PDU của khung tin nhǎn Modbus RTU



**Hình 2-13 Khối PDU trong Modbus RTU**

Sau đó chèn thêm phần MBAP (Modbus Application Protocol), được khung tin nhǎn Modbus TCP/IP MBAP Header, 7 bytes được thêm vào đầu của khung tin nhǎn.



**Hình 2-14 Cấu trúc gói tin của Modbus TCP**

Chức năng và vai trò cụ thể như sau:

- Gói MBAP Header: 2 bytes Transaction ID, 2 bytes Protocol ID, 2 bytes Length, 1 byte Unit ID
  - Transaction ID: mã định danh của một phiên làm việc.
  - Protocol ID: mã định dạng giao thức.
  - Length: độ dài tin nhǎn còn lại bắt đầu từ unit ID.
  - Unit ID: mã định danh của thiết bị modbus cần đọc dữ liệu.
- Gói PDU: bao gồm Function Code và Data tương tự như Modbus RTU

## 2.3 Giao thức MQTT

### 2.3.1 Giới thiệu

MQTT viết tắt của Message Telemetry Transport [7] (trước đó MQTT được định nghĩa là Message Queuing Telemetry Transport), là giao thức ở tầng ứng dụng chạy trên nền TCP thuộc tập giao thức TCP/IP.

MQTT được phát minh vào năm 1999 bởi Andy Stanford-Clark (IBM) và Arlen Nipper (Eurotech) [3]. Ban đầu, họ thiết kế MQTT để dùng trong hệ thống điều khiển giám sát và thu thập dữ liệu (SCADA) cho hệ thống dẫn dầu thông qua vệ tinh.

### 2.3.2 Đặc điểm

Giao thức MQTT có kiến trúc theo mô hình Publish/Subscribe giúp truyền tải dữ liệu giữa các thiết bị, ứng dụng với nhau. Dữ liệu trong giao thức này là một chuỗi nhị phân (binary) chứ không phải chuỗi văn bản (text string), được định dạng theo gói tin command hoặc gói tin command acknowledgement.

Các thành phần lõi trong kiến trúc của MQTT:

- MQTT Broker: Được cung cấp dưới dạng mã nguồn mở hoặc các phiên bản thương mại, có thể đi kèm với các dịch vụ điện toán đám mây. Công việc của Broker là lọc các tin nhắn dựa trên topic, sau đó phân phối các tin nhắn đến các thiết bị/ứng dụng đã đăng ký topic đó. Các bạn có thể tham khảo một số MQTT Broker như: HiveMQ, Mosquitto, MQTTRoute, Jmqtt, ...
- MQTT Client: Là các thiết bị/ứng dụng Client kết nối đến Broker để thực hiện truyền nhận dữ liệu. Hiện nay có rất nhiều mã nguồn mở MQTT Client được viết dưới nhiều ngôn ngữ khác nhau như HiveMQ MQTT Client được phát triển dựa trên ngôn ngữ Java, Eclipse Paho dựa trên C/C++, Python, ...
- Topic: Mỗi MQTT Client thực hiện truyền/nhận dữ liệu với nhau thông qua các Topic được quản lý bởi Broker. Một Client đăng ký nhận dữ liệu được gọi là một Subscriber còn một Client gửi dữ liệu đi được gọi là một Publisher. Để nhận dữ liệu từ Publisher, đầu tiên Subscriber phải subscribe (đăng ký theo dõi) đến một Topic, sau đó bất cứ Client nào publish

dữ liệu đến đúng Topic, thì Broker sẽ lọc và chuyển tiếp gói tin đến đúng Subscriber đó. Một Client có thể subscribe hoặc publish đến nhiều Topic khác nhau.

Một số ưu điểm của MQTT:

- MQTT là một “light weight protocol” giúp Broker và Client trao đổi gói tin có kích thước nhỏ, không chứa nhiều các dữ liệu phụ thêm vào nên có thể truyền nhận một cách mượt mà trong điều kiện bị giới hạn về băng thông đường truyền.
- Là một giao thức kết nối M2M/Internet of Things, MQTT được thiết kế để truyền tải tin nhắn trong các ứng dụng yêu cầu công suất thấp, kích thước bộ nhớ vi điều khiển nhỏ; băng thông thấp; đảm bảo việc truyền nhận các gói tin diễn ra thành công, ...
- Với giao thức MQTT, các gói tin sẽ được truyền đi theo hai chiều: từ thiết bị đến Broker và từ Broker đến thiết bị. Một Client có khả năng gửi gói tin đến nhiều Client khác trong một lần gửi và nó cũng có thể nhận nhiều gói tin khác nhau từ các Client khác nhau.
- MQTT có thể mở rộng quy mô để kết nối với hàng triệu thiết bị IoT.
- MQTT có 3 mức Quality of Service (QoS) được xác định đảm bảo độ tin cậy trong việc gửi gói tin. Các bạn có thể đọc thêm về QoS ở tài liệu tham khảo cuối bài.
- MQTT giúp dễ dàng mã hóa tin nhắn bằng TLS và xác thực phía client bằng các phương thức xác thực như OAuth.
- Hiện nay, các nền tảng điện toán đám mây lớn như Amazon WebService, Google IoT Core, Microsoft Azure ... đều hỗ trợ giao thức MQTT giúp các nhà phát triển dễ dàng thử nghiệm và sử dụng.

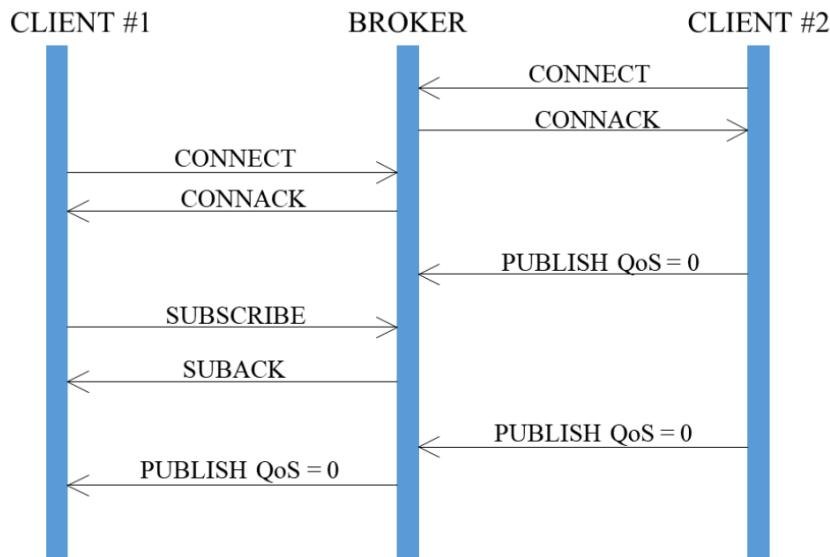
### 2.3.3 Hoạt động

#### 2.3.3.1 Mô hình Publisher – Subscriber

Để giao thức này hoạt động được, cần có một thành phần trung tâm gọi là Broker và các thiết bị/ứng dụng khách còn được gọi là các Client, các Client sẽ kết nối đến Broker. Broker và Client sẽ “nói chuyện” với nhau thông qua các gói tin MQTT được xây dựng dựa theo chuẩn OASIS. Tiêu chuẩn này còn định nghĩa các mức quality of service để đảm bảo độ tin cậy khi truyền nhận gói tin, các kịch bản always connected (luôn giữ kết nối) hoặc sometimes connected (thỉnh thoảng kết nối); khả năng mở rộng để hỗ trợ kết nối số lượng lớn thiết bị...

Một Client có thể đóng vai trò là một Publisher khi thực hiện publish một gói tin hoặc là một Subscriber khi thực hiện subscribe một Topic. Các Client muốn thực hiện publish hoặc subscribe thì phải thực hiện các bước sau:

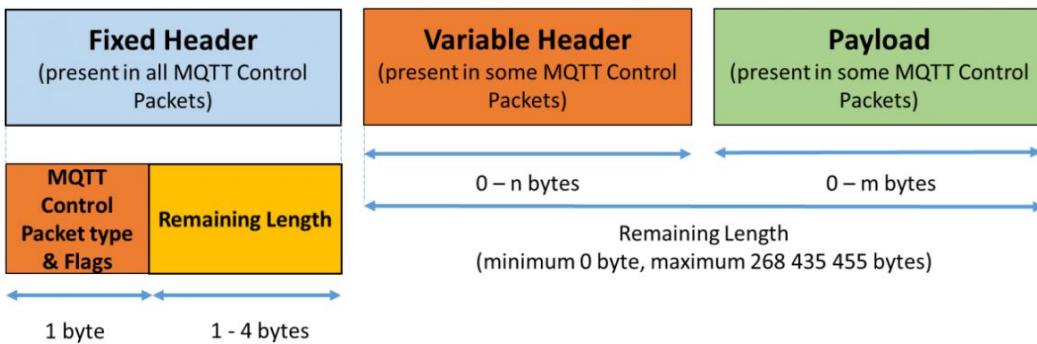
- Thiết lập kết nối TCP đến máy chủ có chứa Broker
- Thiết lập kết nối đến Broker bằng cách gửi gói tin CONNECT
- Đợi gói tin phản hồi CONNACK từ Broker để kiểm tra kết nối được thiết lập thành công hay thất bại
- Nếu kết nối thành công, Client có thể gửi gói tin PUBLISH hoặc SUBSCRIBE đến một Topic tại Broker.
- Khi quá trình truyền nhận kết thúc, Client có thể ngắt kết nối đến Broker bằng cách gửi gói tin DISCONNECT. Nếu Client muốn giữ kết nối đến Broker khi không có bất kỳ hoạt động nào diễn ra trong một khoảng thời gian nhất định thì Client thực hiện gửi gói tin PINGREQ để thông báo Broker rằng kết nối này vẫn được duy trì.



**Hình 2-15 Quá trình hoạt động giữa hai Client (Publisher, Subscriber) và Broker**

### 2.3.3.2 Cấu trúc gói tin trong MQTT

Gói tin MQTT Control bao gồm các gói tin CONNECT, CONNACK, PUBLISH, SUBSCRIBE, SUBACK [8].



**Hình 2-16 Cấu trúc gói tin MQTT Control**

Trong một gói tin MQTT Control sẽ có tối đa 3 thành phần gồm Fixed Header, Variable Header và Payload. Các gói tin MQTT Control ngoài chứa mã định danh còn chứa một số thông tin khác như tên Topic, Client ID, tên User, ...

Trường Fixed Header bắt buộc phải có trong các gói tin MQTT Control, đây là trường phân biệt các gói tin CONNECT, PUBLISH, SUBSCRIBE, PINGREQ, DISCONNECT...

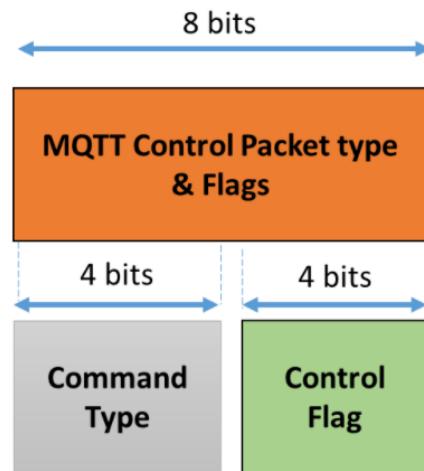
Hai trường Variable Header và Payload là hai trường không bắt buộc, tùy vào gói tin MQTT Control mà có thể có hoặc không 2 trường này. Ví dụ:

- Gói tin CONNACT: Chỉ có trường Fixed Header.
- Gói tin PUBACK: Gồm các trường Fixed Header và Variable Header.
- Gói tin CONNECT: Gồm các trường Fixed Header, Variable Header và Payload.

### Fixed Header

Trường này có tối thiểu 2 byte. Fixed Header chứa MQTT Control Packet type và Flags và Remaining Length. Trường Remaining Length sẽ cho biết tổng số byte của Variable Header và Payload (tối thiểu là 0 Byte và tối đa là 256MByte). Ở đây chúng ta sẽ phân tích trường MQTT Control Packet type và Flags, đây là trường được dùng để phân biệt các gói tin MQTT Control với nhau.

Trong 8 bits của trường MQTT Control Packet type và Flags này, 4 bits có trọng số cao (MSB) thể hiện Command Type, 4 bits còn lại là các bit Control Flag chứa cờ cụ thể cho từng loại Command Type.



**Hình 2-17 Fix Header trong gói tin MQTT Control**

### Variable Header

Đây là trường không bắt buộc có trong các gói tin MQTT Control. Một vài gói tin như CONNECT, PUBLISH, SUBSCRIBE ... sử dụng trường Variable Header này để cung cấp thêm thông tin và chúng khác nhau tùy thuộc vào các gói tin.

### Payload

Cuối một số gói tin MQTT Control có thể chứa một trường Payload vì trường Payload là tùy chọn và không bắt buộc phải xuất hiện trong tất cả các gói tin MQTT như trường Fixed Header. Trường này thường chứa nội dung dữ liệu được gửi. Ví dụ:

- Ở gói tin CONNECT, Payload sẽ là ClientID, username và password (nếu có).
- Ở gói tin PUBLISH, Payload sẽ là nội dung dữ liệu cần gửi đi.

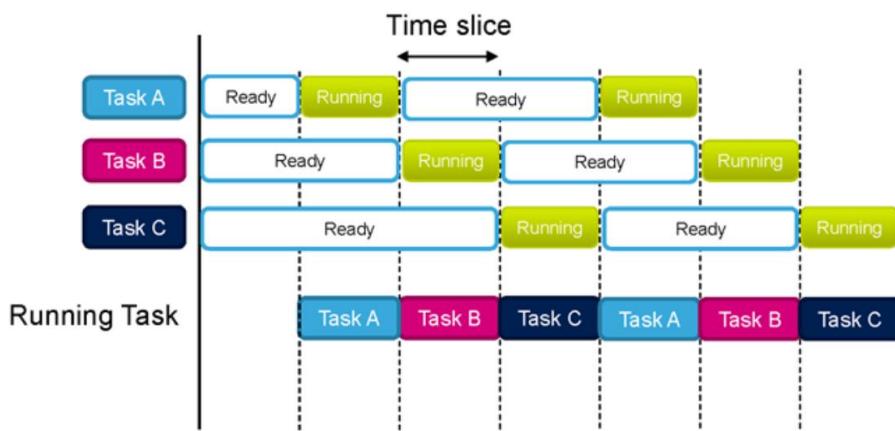
## 2.4 Hệ điều hành thời gian thực

### 2.4.1 Giới thiệu

FreeRTOS là một hệ điều hành nhúng thời gian thực (Real Time Operating System) mã nguồn mở được phát triển bởi Real Time Engineers Ltd, sáng lập và sở hữu bởi Richard Barry [9].

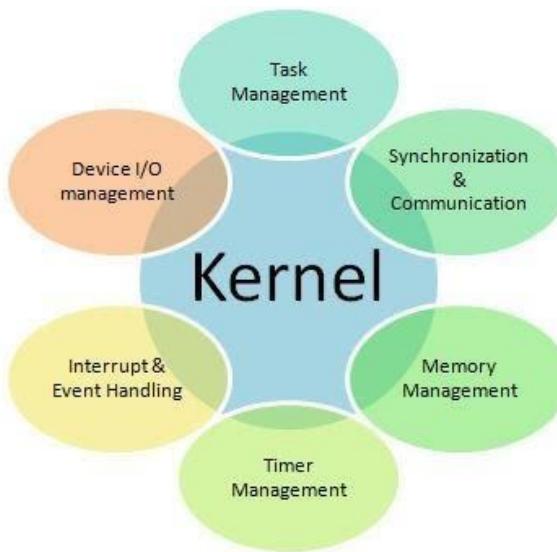
## 2.4.2 Đặc điểm

Hệ điều hành thời gian thực được thiết kế ra cho các nhiệm vụ đặc biệt. Các ứng dụng cần được thực thi với thời gian thật chính xác, các lỗi phát sinh cần được cô lập và xử lý nhanh chóng. Mọi sự chậm trễ, lỗi phát sinh không lường trước có thể khiến hệ thống bị đỗ vỡ. Vì vậy, RTOS sử dụng trong những ứng dụng yêu cầu thời gian đáp ứng nhanh, chính xác về thời gian. Vì điều khiển có tài nguyên rất giới hạn. Do đó, hệ điều hành này chỉ tập trung vào một số ít các tính năng. Chúng cố gắng tối ưu hóa số lượng, bộ lập lịch và các tác vụ (task) trên một hệ thống cỡ nhỏ.



**Hình 2-18 Ví dụ về multi-tasking trong các hệ thống thời gian thực**

FreeRTOS được thiết kế phù hợp cho nhiều hệ nhúng nhỏ gọn vì nó chỉ triển khai rất ít các chức năng như: cơ chế quản lý bộ nhớ và tác vụ cơ bản, các hàm API quan trọng cho cơ chế đồng bộ. Nó không cung cấp sẵn các giao tiếp mạng, drivers, hay hệ thống quản lý tệp (file system) như những hệ điều hành nhúng cao cấp khác. Tuy vậy, FreeRTOS có nhiều ưu điểm, hỗ trợ nhiều kiến trúc vi điều khiển khác nhau, kích thước nhỏ gọn (4.3 Kbytes sau khi biên dịch trên Arduino), được viết bằng ngôn ngữ C và có thể sử dụng, phát triển với nhiều trình biên dịch C khác nhau (GCC, OpenWatcom, Keil, IAR, Eclipse, ...), cho phép không giới hạn các tác vụ chạy đồng thời, không hạn chế quyền ưu tiên thực thi, khả năng khai thác phần cứng. Ngoài ra, nó cũng cho phép triển khai các cơ chế điều độ giữa các tiến trình như: queues, counting semaphore, mutexes.



**Hình 2-19 Các thành phần cơ bản của một RTOS**

Các lợi ích của việc dùng RTOS:

- Chia sẻ tài nguyên một cách đơn giản: cung cấp cơ chế để phân chia các yêu cầu về bộ nhớ và ngoại vi của MCU.
- Dễ debug và phát triển: Mọi người trong nhóm có thể làm việc một cách độc lập, các lập trình viên thì có thể tránh được các tương tác với ngắn, timer, với phần cứng (cái này mình không khuyến khích lầm vì hiểu được phần cứng vẫn sẽ tốt hơn nhiều).
- Tăng tính linh động và dễ dàng bảo trì: thông qua API của RTOS, việc theo dõi hoạt động của hệ thống dễ dàng hơn.

Dựa trên mục tiêu đề tài, sử dụng FreeRTOS giúp đơn giản việc xây dựng hệ thống chạy đa tác vụ để đảm bảo các tính ra. Bên cạnh đó, khả năng phát triển, kiểm tra gỡ lỗi nền vi điều khiển cũng được tiện lợi hơn.

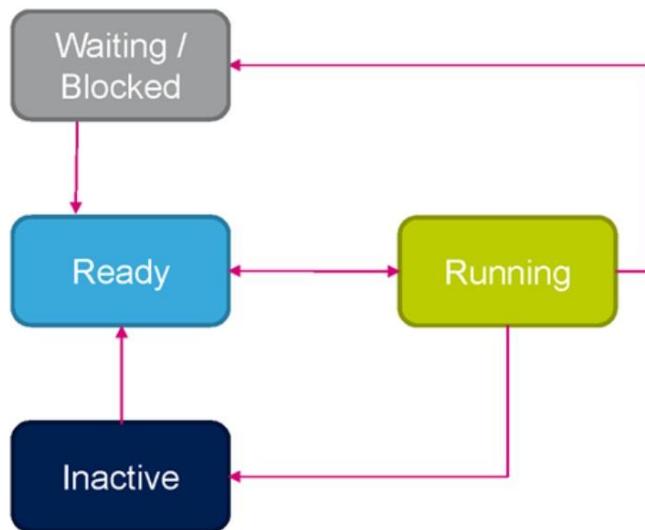
#### 2.4.3 Hoạt động

Một RTOS hoạt động có các chức năng cơ bản sau đây:

- Bộ lập lịch (Scheduler)
- Các dịch vụ thời gian thực (Realtime Servers)
- Đòng bộ và xử lý thông điệp

#### 2.4.3.1 BỘ lẬP lỊCH

Mỗi task có thể có các trạng thái hoạt động chính.



**Hình 2-20 Các trạng thái hoạt động của task**

**Ready:** Task đã sẵn sàng để có thể thực thi nhưng chưa được thực thi do có các task khác với độ ưu tiên ngang bằng hoặc hơn đang chạy.

**Running:** khi task thực sự đang chạy

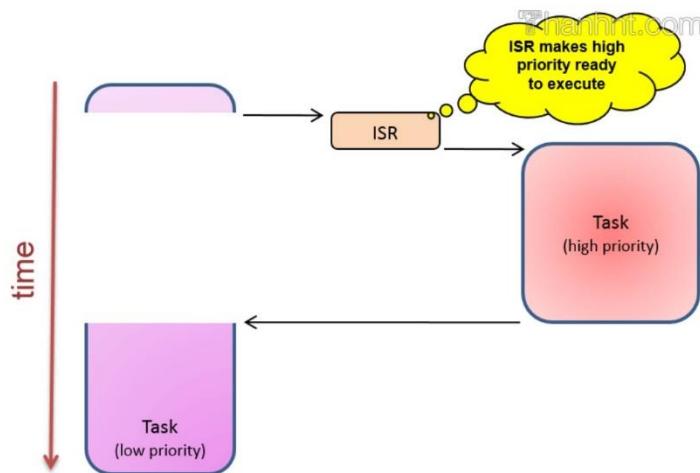
**Blocked (Waiting):** Task đang đợi một event tạm hoặc event từ bên ngoài

**Suspended:** Task không khả dụng để lên lịch (scheduling)

Để lập lịch cho Task, có 3 kỹ thuật được áp dụng:

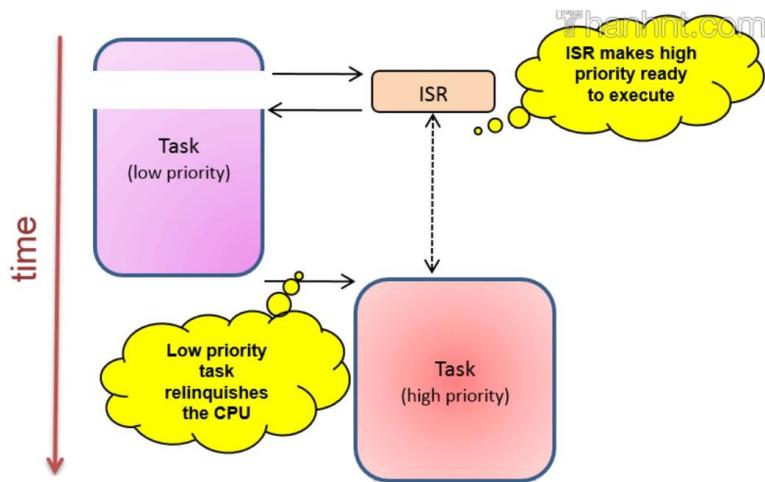
- **Round Robin Scheduling:** Mỗi task được chia cho một khe thời gian cố định, nếu trong khoảng thời gian được chia đó mà task chưa thực hiện xong thì sẽ bị tạm dừng, chờ đến lượt tiếp theo để thực hiện tiếp công việc sau khi hệ thống xử lý hết một lượt các task.
- **Co-operative scheduling:** Mỗi task được thực thi đến khi kết thúc quá trình thì task tiếp theo mới được thực thi.
- **Preemptive Scheduling:** Phương pháp này ưu tiên phân bổ thời gian cho các task có mức ưu tiên cao hơn. Mỗi task được gán 1 mức ưu tiên duy nhất. Có thể có 256 mức ưu tiên trong hệ thống, và có thể có nhiều task có cùng mức ưu tiên.

- **Preemptive:** Các task có mức ưu tiên cao nhất luôn được kiểm soát bởi CPU, khi phát sinh ISR thì hệ thống sẽ tạm dừng task đang thực thi, hoàn thành ISR sau đó hệ thống thực thi task có mức ưu tiên cao nhất tại thời điểm đó. Sau đó hệ thống mới tiến hành nối lại các task đang bị gián đoạn. Ở chế độ preemptive, hệ thống có thể đáp ứng các công việc khẩn cấp một cách nhanh chóng. Đa số các hệ thống thực tế đang chạy ở chế độ này.



**Hình 2-21 Preemptive scheduling**

- **Non-preemptive:** Ở chế độ non-preemptive thì các task được chạy cho đến khi nó hoàn tất. Khi phát sinh ISR thì hệ thống sẽ tạm dừng task đang thực thi và hoàn thành ISR, sau khi hoàn thành ISR thì hệ thống sẽ quay lại thực thi nốt phần việc còn lại của task bị gián đoạn. Task có mức ưu tiên cao hơn sẽ giành quyền kiểm soát CPU sau khi task bị gián đoạn thực thi xong.



**Hình 2-22 non-Preemptive scheduling**

Kernel tiến hành quản lý task ở nhiều giai đoạn. Chúng bao gồm: tạo task, huỷ task, thay đổi mức ưu tiên của task, thay đổi trạng thái của task.

#### 2.4.3.2 Các dịch vụ thời gian thực

Kernel thực hiện chức năng quản lý và lập lịch các quá trình sử dụng CPU và điều phối tài nguyên. Mỗi task chỉ được thực thi bởi CPU trong một khoảng thời gian, trong các khoảng thời gian còn lại thì task được quản lý bởi các dịch vụ quản lý của hệ điều hành.

Các dịch vụ của RTOS bao gồm:

- Xử lý ngắn (Interrupt handling services).
- Dịch vụ quản lý thời gian (Time services).
- Dịch vụ quản lý thiết bị (Device management services).
- Dịch vụ quản lý bộ nhớ (Memory management services).
- Dịch vụ quản lý các kết nối Vào - Ra (IO services).

#### 2.4.3.3 Đóng bộ và xử lý thông điệp

Các thông điệp sử dụng để trao đổi thông tin giữa các hệ thống khác nhau hoặc giữa các task. Các dịch vụ quản lý thông điệp bao gồm: Semaphores, Event flags, Mailboxes, Pipes, Message queues

Semaphores: Dùng để đồng bộ hóa quyền truy cập vào các tài nguyên dùng chung.

Event Flags: Dùng để đồng bộ hóa các hoạt động cần có sự phối hợp của nhiều task.

Mailboxes, Pipes, Message queues: Dùng để quản lý các thông điệp gửi đi - đến giữa các task.

## 2.5 Tìm hiểu về ứng dụng web (Web App)

### 2.5.1 Ứng dụng web là gì?

Ứng dụng web (web application hay web app) là những chương trình, ứng dụng máy tính được xây dựng bằng ngôn ngữ web (HTML, JS, CSS) chạy và sử dụng trình duyệt web với mục đích thực hiện một số những chức năng nhất định như chức năng lưu trữ (data, các loại file) và xử lý dữ liệu để phục vụ nhu cầu và tương tác với người dùng thông qua kết nối internet [10].



Hình 2-23 Khái niệm về ứng dụng web

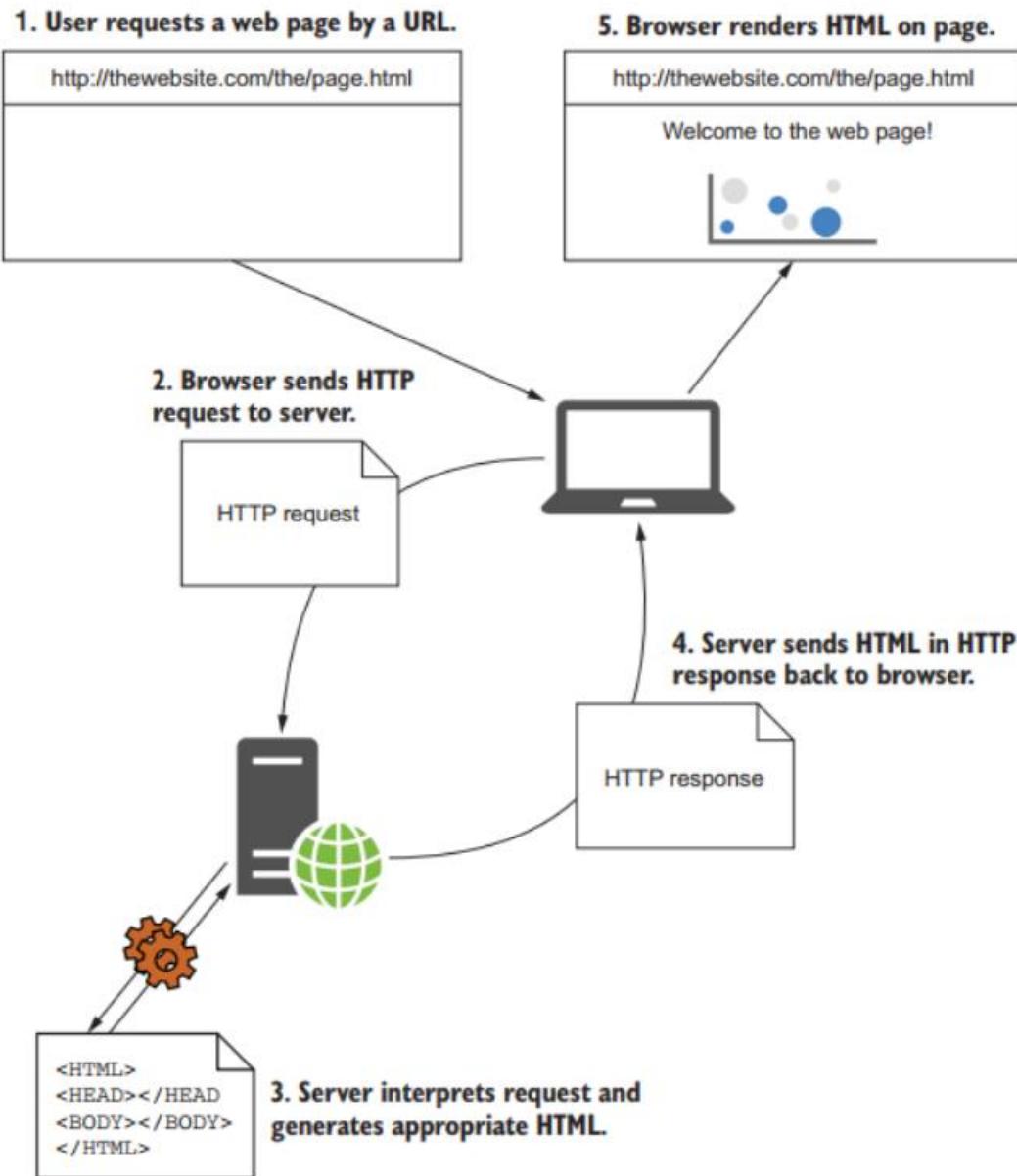
Điển hình cho các loại web application bạn có thể đã biết như các phần mềm, website chỉnh sửa trực tuyến như canva, các trang mạng xã hội, các trang thương mại điện tử, website bán hàng, website quản lý siêu thị, phần mềm quản lý nhà trọ Mona House (với cả bản web app và app điện

thoại) ... Tại đó, bạn có thể tìm kiếm tương tác, chia sẻ thông tin, đăng tin, thực hiện các thao tác đặc thù mà từng web app cung cấp.

### 2.5.2 Cách thức hoạt động của một ứng dụng web

Các ứng dụng web không cần tải xuống vì chúng được truy cập qua mạng. Người dùng có thể truy cập ứng dụng Web thông qua trình duyệt web như Google Chrome, Mozilla Firefox hoặc Safari.

Để một ứng dụng web hoạt động, nó cần có máy chủ Web, máy chủ ứng dụng và cơ sở dữ liệu. Máy chủ web quản lý các yêu cầu đến từ máy khách. Trong khi máy chủ ứng dụng hoàn thành tác vụ được yêu cầu. Một cơ sở dữ liệu có thể được sử dụng để lưu trữ bất kỳ thông tin cần thiết nào.

**Hình 2-24** Cách thức hoạt động của một web app

Sau đây là cách thức mà web application hoạt động [11]:

- Người dùng kích hoạt request tới web server qua Internet, thông qua trình duyệt web hoặc giao diện người dùng của ứng dụng.
- Web server chuyển tiếp request này đến web application server thích hợp.

- Máy chủ ứng dụng Web (web application server) thực hiện nhiệm vụ được yêu cầu - chẳng hạn như truy vấn cơ sở dữ liệu hoặc xử lý dữ liệu - sau đó tạo ra các kết quả của dữ liệu được yêu cầu.
- Máy chủ ứng dụng web gửi kết quả đến máy chủ web với thông tin được yêu cầu hoặc dữ liệu đã được xử lý.
- Máy chủ web phản hồi response lại cho khách hàng các thông tin được yêu cầu sau đó xuất hiện trên màn hình của người dùng.

### 2.5.3 Quá trình thiết kế một ứng dụng web

Dưới đây là các bước cơ bản nhất giúp xây dựng một web app đơn giản [10]:

#### Bước 1: Xây dựng ý tưởng

Trước khi tiến hành thiết kế web app, ta cần lên được ý tưởng xây dựng dựa trên một vấn đề nhất định. Ngoài ra, ta cần phải hiểu rõ được mình sẽ phải làm những gì, tại sao lại cần sử dụng ứng dụng này và sử dụng nó như thế nào?

#### Bước 2: Nghiên cứu thị trường

Khi đã có ý tưởng, ta nên bắt tay vào nghiên cứu thị trường để biết được mình cần phải làm gì với sản phẩm. Một số thông tin ta phải có trước tiên chính là web app của bạn có tương tự với sản phẩm nào trên thị trường hay không? Đối tượng nào đã sử dụng web app rồi? để tạo ra được sản phẩm, dịch vụ thực sự phù hợp với thị trường.

#### Bước 3: Xác định chức năng của web app

Xây dựng một web app có càng nhiều chức năng thì càng tốn nhiều thời gian. Điều này có thể dễ dàng gây ra sự chán nản cho nhà thiết kế khi phải mất quá nhiều thời gian để xây dựng nhưng mãi chưa có kết quả. Vì vậy, trước khi xây dựng web app ta cần phải xác định một số những chức năng cơ bản của web app cần có.

#### Bước 4: Phác thảo thiết kế

khi đã có ý tưởng và xác định được các chức năng, ta có thể phác thảo giao diện người dùng web app một cách sáng tạo nhưng cần phải xem xét dựa trên một số điều như cấu trúc trang, dẫn hướng,

tính thương hiệu, các button trên trang và các yếu tố tương tác. Phác thảo ra thật nhiều giao diện web app cùng những tính năng sẽ đưa vào để xác định hiệu quả của web app.

#### Bước 5: Xây dựng quy trình làm việc cho web app

Để có thể xây dựng được quy trình làm việc cho web app một cách tốt nhất thì ta cần đặt mình vào vị trí của người dùng. Ta có thể xem xét một số điều sau:

- Làm thế nào để đăng ký web app?
- Người dùng có nhận được email xác minh sau đăng ký không?
- Làm thế nào để đăng nhập web app?
- Cách để người dùng thay đổi mật khẩu?
- Người dùng điều hướng như thế nào trong web app?
- Cách người dùng thay đổi cài đặt?

#### Bước 6: Bắt đầu xây dựng với cơ sở dữ liệu

Khi đã tiến hành thử nghiệm, ta có thể bắt đầu tiến hành khởi công xây dựng cơ sở dữ liệu. Ta có thể tạo một thư mục trên ổ cứng hoặc lưu trữ một vài tài liệu trên web app của mình. Có rất nhiều loại cơ sở khác nhau được sử dụng với mục đích khác nhau.

Một số loại cơ sở dữ liệu được web app sử dụng là SQL, Document database, ...

#### Bước 7: Xây dựng frontend

Frontend hay giao diện người dùng giúp ta có thể xác định được những gì mà mình thấy và có thể tương tác được phát triển với JavaScript, CSS và HTML. Khi xây dựng frontend bằng SPA, bạn cần thiết lập được môi trường với các thành phần như trình chỉnh sửa mã, Webpack, Packaging framework, Frontend framework, ... Còn sử dụng các trang máy chủ thì việc bắt đầu sẽ dễ dàng hơn.

#### Bước 8: Xây dựng backend

Xây dựng backend là một bước vô cùng quan trọng giúp quản lý mọi dữ liệu. Trước khi xây dựng web app ta sẽ phải lựa chọn giữa trang máy chủ hoặc ứng dụng đơn. Một số những công việc chính

của backend bao gồm xác thực người dùng, phục vụ frontend và cung cấp HTTP endpoint cho frontend để nó hoạt động được trên dữ liệu.

#### Bước 9: Sử dụng hosting

Hosting có liên quan đến việc chạy web app trên một máy chủ cụ thể. Để sử dụng hosting ta nên làm theo những bước cơ bản sau:

- Mua miền và thiết lập chứng chỉ SSL.
- Chọn một trong những nhà cung cấp cloud như Amazon, MS Azure hoặc Nền tảng đám mây của Google.

#### Bước 10: Triển khai

Khi ta đã có đầy đủ ý tưởng, xác thực, thiết kế, phát triển ứng dụng web và chọn được nhà cung cấp hosting rồi thì bước cuối cùng chính là triển khai web app. Ta có thể sử dụng các công cụ cung cấp khả năng tích hợp liên tục giúp triển khai ứng dụng web lên dịch vụ lưu trữ đám mây như Bitbuckets, Jenkins hoặc GitLab.

## 2.6 Tìm hiểu về ReactJS

### 2.6.1 Giới thiệu về ReactJS

ReactJS (đôi lúc cũng có thể gọi là React.JS, React) là một thư viện Javascript giúp bạn nhanh chóng xây dựng giao diện ứng dụng. React có thể xây dựng website hoàn toàn sử dụng Javascript (để thao tác với HTML), được tối ưu hiệu năng bởi kỹ thuật VirtualDOM [12].

React được Facebook giới thiệu vào năm 2011, họ quảng bá đây là thư viện cho phép developer tạo các dự án ứng dụng web lớn, có giao diện UI phức tạp từ một đoạn mã nguồn nhỏ và độc lập.

Ưu điểm của React:

- Component độc lập – có thể tái sử dụng:

Trong React, giao diện được xây dựng từ việc kết hợp các components. Bạn có thể hiểu đơn giản component là một hàm, nó nhận giá trị bạn truyền vào và trả về một số đầu ra. Và cũng giống như function, component có thể tái sử dụng ở bất kỳ đâu. Vì vậy, chúng ta có

thể tái sử dụng lại, hợp nhất các component để tạo thành giao diện người dùng phức tạp hơn.

- React làm cho front-end javascript dễ sử dụng hơn, nhanh hơn bằng cách sử dụng Virtual DOM:

Với React, bạn chỉ cần thay đổi state của UI, và React sẽ tự động cập nhật DOM để phù hợp với state đó. Kỹ thuật được React sử dụng đó là Virtual DOM. Điều này cho phép React chỉ cập nhật những cái cần thiết (kiểu như một phần trang web) thay vì phải tải lại toàn bộ trang.

Việc sử dụng Javascript để tạo ra mã HTML cho phép React có một cây đối tượng HTML ảo — VirtualDOM. Khi bạn tải trang web sử dụng React, một VirtualDOM được tạo ra và lưu trong bộ nhớ.

Mỗi khi state thay đổi, chúng ta sẽ cần phải có một cây đối tượng HTML mới để hiển thị lên trình duyệt. Thay vì tạo lại toàn bộ cây, React sử dụng một thuật toán thông minh để chỉ tạo lại các thành phần khác biệt giữa cây mới và cây cũ. Bởi vì cả hai cây HTML cũ và mới đều được lưu trong bộ nhớ, quá trình xử lý này diễn ra siêu nhanh.

## 2.6.2 Virtual DOM

DOM (viết tắt của Document Object Model) là một giao diện lập trình ứng dụng, cho phép Javascript hoặc các loại script khác đọc và thao tác nội dung của document (trong trường hợp này là HTML) [12].

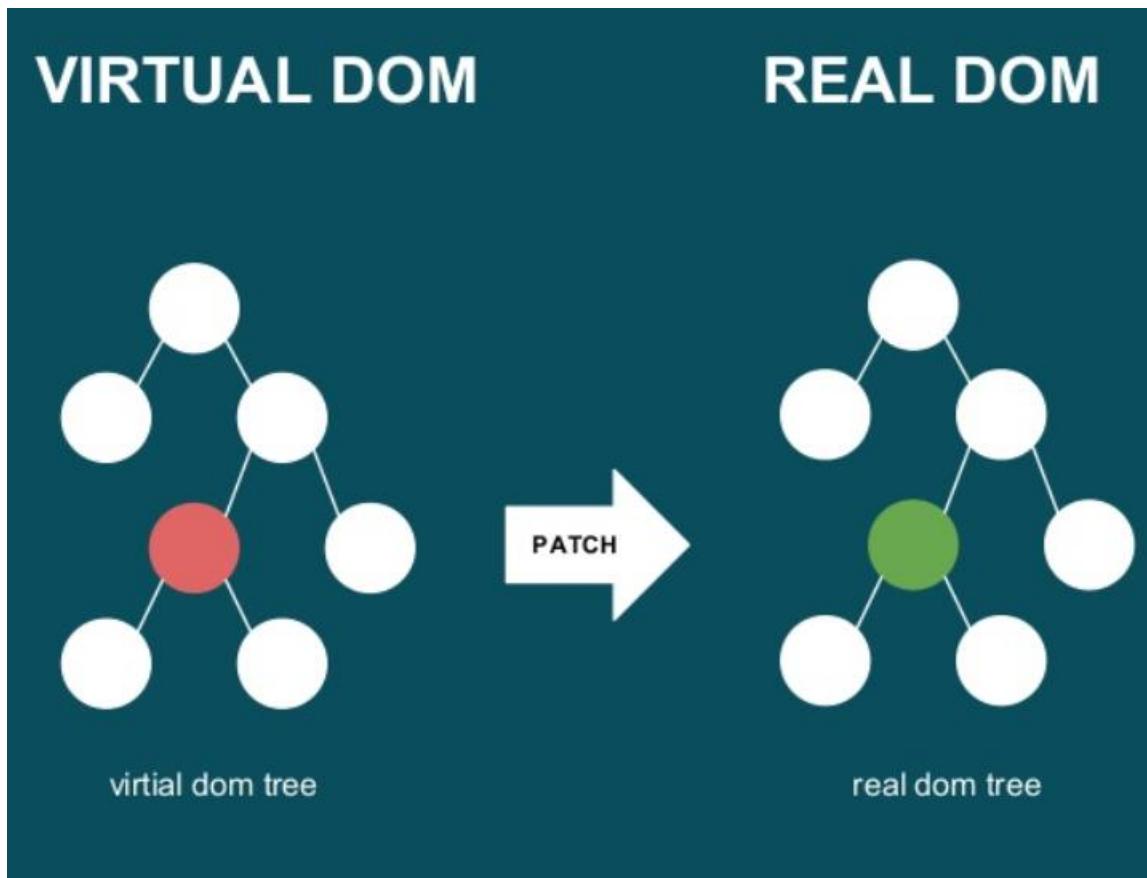
Bất cứ khi nào một HTML document được tải xuống trình duyệt dưới dạng một trang web, một DOM sẽ được tạo cho trang đó. Theo cách này, Javascript có thể thao tác với DOM như thêm, sửa, xóa nội dung... hoặc thực hiện một hành động nào đó như ẩn/hiện một view.

Việc thao túng DOM là không hiệu quả, có thể gây chậm ứng dụng. Đặc biệt là khi bạn thường xuyên phải cập nhật DOM. Vấn đề ở đây là không phải do DOM chậm (bởi vì DOM chỉ là một object). Ứng dụng bị chậm là do trình duyệt phải xử lý khi DOM thay đổi. Mỗi khi DOM thay đổi, trình duyệt sẽ phải tính toán lại CSS, chạy bô cục và render lại trang web.

Để tăng tốc ứng dụng, chúng ta cần tìm cách giảm thiểu thời gian khi trình duyệt render lại trang. Đó là lý do mà nhà phát triển React nghĩ tới Virtual DOM.

### Cơ chế hoạt động của Virtual DOM

- Bất cứ khi nào có yếu tố mới được thêm vào UI, một Virtual DOM sẽ được tạo.
- Khi state của bất cứ yếu tố nào thay đổi, React sẽ cập nhật Virtual DOM trong khi vẫn giữ phiên bản Virtual DOM trước để so sánh và tìm ra đối tượng Virtual DOM thay đổi. Khi tìm được sự thay đổi, React chỉ cập nhật đối tượng đó trên DOM thật.



**Hình 2-25 Virtual DOM trong ReactJS**

### 2.6.3 React Component

React được xây dựng xung quanh các component, chứ không dùng template như các framework khác. Trong React, chúng ta xây dựng trang web sử dụng những thành phần (component) nhỏ. Chúng ta có thể tái sử dụng một component ở nhiều nơi, với các trạng thái hoặc các thuộc tính khác nhau, trong một component lại có thể chứa thành phần khác. Mỗi component trong React có

một trạng thái riêng, có thể thay đổi, và React sẽ thực hiện cập nhật component dựa trên những thay đổi của trạng thái. Mọi thứ React đều là component.

React có 2 kiểu viết component: Functional component và Class component.

- Functional Component

Functional component là một hàm Javascript (hoặc ES6) trả về 1 React element. Theo tài liệu chính thức của React, hàm dưới đây là một component hợp lệ.

```
function Welcome(props) {  
  return <h1>Hello, {props.name}</h1>;  
}
```

Function này là một component React hợp lệ vì nó nhận một "props" làm tham số và trả về 1 React element.

Tóm lại, 1 React function component:

- Là một function Javascript / ES6 function
- Phải trả về 1 React element
- Nhập props làm tham số nếu cần

- Class Component

Các Class components là những class ES6. Chúng phức tạp hơn functional components ở chỗ nó còn có: phương thức khởi tạo, life cycle, hàm render () và quản lý state (data).

Ví dụ một class component:

```
class Welcome extends React.Component {  
  render() {  
    return <h1>Hello, {this.props.name}</h1>;  
  }  
}
```

Tóm lại, một Class Component là:

- Là một class kế thừa từ React Component
- Có thể nhập props (trong hàm khởi tạo) nếu cần
- Phải có hàm render () và trong đó trả về 1 React element hoặc NULL

## 2.6.4 Props và State

*Props:*

Props (nó là từ viết tắt của properties) có thể được coi là các thuộc tính của phần HTML.

Props là cách để bạn có thể truyền dữ liệu từ component cha xuống component con. Khi thực hiện truyền dữ liệu qua props, dữ liệu trong component con chỉ được đọc, không thể thay đổi. Nhờ đó mà component được sử dụng ở bất kỳ đâu cũng luôn hiển thị cùng 1 đầu ra khi có cùng 1 giá trị đầu vào.

Ví dụ cách sử dụng props

```
class Welcome extends React.Component {  
    render() {  
        return <h1>Hello {this.props.name}</h1>;  
    }  
}  
  
const element = <Welcome name="Sơn Dương" />;
```

Trong ví dụ trên, trong Welcome component có một props có tên là name. Và khi nó được gọi thì chúng ta có thể truyền giá trị vào props đó.

*State:*

State giống như một kho lưu trữ dữ liệu cho các component trong ReactJS. Nó chủ yếu được sử dụng để cập nhật component khi người dùng thực hiện một số hành động như nhập vào nút, nhập một số văn bản, nhấn một số phím...

Ví dụ cách sử dụng state:

```

class Form extends React.Component {
  constructor (props) {
    super(props)
    this.state = {
      input: ''
    }
  }
  handleChange = (text) => {
    this.setState({ input: text })
  }

  render () {
    const { input } = this.state
    return (
      <div>
        <label>
          Name:
          <input type="text" value={this.state.value} onChange={this.handleChange} />
        </label>
        <input type="submit" value="Submit" />
      </div>
    )
  }
}

```

### Hình 2-26 Ví dụ cách sử dụng state trong ReactJS

Tóm lại:

- State của một component là dữ liệu mà chỉ có component đó sử dụng và quản lý.
- Props là cách để truyền dữ liệu từ component cha xuống component con và chỉ component cha mới có thể thay đổi giá trị được.

#### 2.6.5 Vòng đời component trong React

Tất cả các component được tạo ra sẽ phải trải qua một loạt các events hoặc các giai đoạn từ khi được gọi tới khi bị hủy.

Trong React, một component sẽ phải trải qua 3 giai đoạn chính:

- Mounting: Đúng như tên gọi của nó, đây là giai đoạn mà Component được tạo và thêm vào cây DOM. Đây có thể coi là giai đoạn bắt đầu của một Component. Các methods được gọi trong giai đoạn này:

- componentWillMount ()
  - render ()
  - componentDidMount ()
- Updating: Sau khi Component qua giai đoạn đầu tiên, tức là đã được thêm vào DOM, giai đoạn tiếp theo là cập nhật nếu có yêu cầu. Component sẽ được update khi giá trị state hoặc props thay đổi, lúc này Component sẽ được render lại. Các methods trong giai đoạn này gồm:
- componentWillMountReceiveProps ()
  - shouldComponentUpdate ()
  - componentWillUpdate ()
  - render ()
  - componentDidUpdate ()
- Unmounting: Đây là giai đoạn kết thúc vòng đời của Component, khi nó được xóa khỏi DOM. Chỉ có duy nhất 1 method trong giai đoạn này:
- componentWillUnmount ()

## 2.6.6 React Hooks

Như trong phần tìm hiểu về Component, chúng ta biết rằng, có 2 kiểu viết component: Class Component và functional Component. Trong đó, functional Component có những hạn chế nhất định như: không có state, không có life-cycle method...

React Hooks ra đời để khắc phục những nhược điểm của functional component và còn hơn thế nữa. React Hooks cho phép chúng ta sử dụng state và các tính năng khác của React mà không phải dùng đến Class.

Các phương thức Hooks cơ bản:

- useState ()

Ta sử dụng hook có tên là useState () để khai báo và cập nhật giá trị State. useState () trả về 1 cặp gồm:

- Giá trị của state hiện tại
- Phương thức để cập nhật state đó

Ví dụ về cách sử dụng state bằng hook:

```
import React, { useState } from 'react';
function VidaHook() {
    //Khai báo một biến trạng thái mới, đặt tên là là "count"
    const [count, setCount] = useState(0);

    const handleChangeCount = () => {
        setCount(count + 1);
    }
    return (
        <div>
            <p>Bạn đã click {count} lần</p>
            <button onClick={handleChangeCount}>
                Click vào đây
            </button>
        </div>
    );
}
```

### Hình 2-27 Ví dụ về cách sử dụng state bằng hook

Hàm setCount có tác dụng như khi dùng this.setState () trong class component. Chúng giúp chúng ta update giá trị của state khi cần. Còn giá trị "0" truyền vào hook useState (0) chỉ là giá trị khởi tạo mặc định ban đầu của state count.

- useEffect ()

Về cơ bản, useEffect Hook được dùng mục đích để quản lý vòng đời của một component. Chúng ta sử dụng hook này trong các function component thay thế các lifecycle trong class component (cơ bản là giống nhau).

Cú pháp của useEffect () cơ bản như sau:

```
useEffect(effectFunction, arrayDependencies)
```

Trong đó:

- effectFunction: gọi là side-effect function, thường được để bạn thực hiện logic chương trình khi useEffect được gọi.
- arrayDependencies: mảng phụ thuộc, cơ bản là để bạn xác định khi nào thì hàm side-effect được gọi.

Các cách dùng của useEffect ()

- useEffect Hook: luôn luôn gọi

Đây là cách sử dụng đơn giản nhất. Trong đó, chúng ta chỉ truyền một đối số – là một hàm số. Hàm này sẽ được gọi mọi lúc – bất cứ khi nào component được render để hiển thị (bao gồm cả lúc update hoặc tạo mới component).

```
useEffect(() => {
    // Bạn viết code xử lý logic tại đây
});
```

- useEffect Hook: chỉ gọi lúc component được mount xong

Tham số thứ hai – ở đây chúng ta truyền vào là mảng rỗng – gọi là dependency array. Nếu dependency array là rỗng, thì hàm side-effect trong đối số thứ nhất không có dependencies. Điều này có nghĩa là nó chỉ chạy duy nhất lần đầu tiên khi component hiển thị.

```
useEffect(() => {
    // Bạn viết code xử lý logic tại đây
}, []);
```

- useEffect Hook: UnMount

useEffect cho phép chúng ta return một function, function này sẽ thực thi trước khi mà component đó được unmounted.

```
useEffect(() => {
    // hàm được trả về sẽ được gọi khi component unmount
    return () => {
        // Bạn viết code xử lý logic tại đây khi component unmount.
    }
}, [])
```

## 2.7 Tìm hiểu về NodeJS

### 2.7.1 NodeJS là gì?

NodeJS là một mã nguồn được xây dựng dựa trên nền tảng Javascript V8 Engine, nó được sử dụng để xây dựng các ứng dụng web như các trang video clip, các forum và đặc biệt là trang mạng xã hội phạm vi hẹp. NodeJS là một mã nguồn mở được sử dụng rộng bởi hàng ngàn lập trình viên trên toàn thế giới [13].

NodeJS có thể chạy trên nhiều nền tảng hệ điều hành khác nhau từ Window cho tới Linux, OS nên đó cũng là một lợi thế. NodeJS cung cấp các thư viện phong phú ở dạng Javascript Module khác nhau giúp đơn giản hóa việc lập trình và giảm thời gian ở mức thấp nhất.

Có một số ngôn ngữ lập trình được thiết kế để viết các ứng dụng cho server như PHP, Ruby, Python, ASP, Java... Nếu trước kia, Javascript được thiết kế để chạy trên các trình duyệt, cung cấp thêm khả năng tương tác của trang web với người dùng. Ví dụ như các menu có hiệu ứng dropdown, hay hiệu ứng tuyệt vời...

Nhưng mọi chuyện đã thay đổi vào năm 2009, khi Node.js ra đời, sử dụng động V8 engine làm bộ chạy các mã Javascript. Giờ đây, Javascript đã vượt ra khỏi khuôn khổ của trình duyệt, và cho phép nó chạy trên các server.

Những lợi ích mà NodeJS mang lại:

- Đầu tiên, V8 Javascript engine là một Javascript engine mạnh mẽ, được sử dụng trong trình duyệt Chrome của Google. Điều này sẽ làm ứng dụng của bạn có tốc độ rất nhanh.
- NodeJS khuyến khích viết mã kiểu asynchronous (bất đồng bộ) để cải thiện tốc độ ứng dụng, tránh được những vấn đề phát sinh của việc sử dụng đa luồng.
- Javascript là một ngôn ngữ phổ biến, do vậy bạn sẽ thừa hưởng rất nhiều thư viện.

### 2.7.2 Framework ExpressJS

ExpressJS là một web framework được xây dựng trên nền tảng NodeJs. Expressjs cung cấp các hàm HTTP và middleware để tạo ra API đơn giản và dễ sử dụng. Có rất nhiều lý do để chọn ExpressJS:

- Có nhiều tính năng hỗ trợ tất cả những gì bạn cần trong việc xây dựng Web và API
- Quản lý các route dễ dàng
- Cung cấp một nền tảng phát triển cho các API
- Hỗ trợ nhiều thư viện và plugin
- Bảo mật và an toàn hơn so với việc code thuần
- Hỗ trợ cộng đồng tuyệt vời

Các thành phần chính:

- Route Handlers

Route Handlers quyết định ứng dụng sẽ đáp ứng các request từ phía client như thế nào.

Route là 1 URI (hoặc đường dẫn) và method (POST, GET, PUT, ...), mỗi route có thể có 1 hoặc nhiều functions, các function này sẽ được thực thi khi route khớp. Về mặt cơ bản, tất cả route đều có thể được định nghĩa trong file js chính của application, định nghĩa 1 route như sau:

```
app.METHOD(PATH, HANDLER)
```

Trong đó app là 1 thực thể của express, PATH là đường dẫn, METHOD là 1 HTTP request method viết thường, HANDLER là các function sẽ được thực thi khi route khớp.

- Middleware

Function Middleware là các function có thể truy xuất object request (req), object response (res), và function middleware kế tiếp trong chu kỳ request-response của ứng dụng. Các function Middleware có thể:

- Thực thi code bất kỳ.
- Thay đổi các object request response
- Kết thúc chu kỳ request-response
- Gọi function middleware kế tiếp

- Template Engine

Template Engine cho phép ta sử dụng các file view template trong ứng dụng, tại thời điểm thực thi, template engine sẽ thay thế các biến trong file template bằng giá trị được truyền, và chuyển đổi template đó thành file HTML để trả về phía client. Các template engine được sử dụng phổ biến trong Express là Pug, Mustache, EJS, Jade, ...

## 2.8 Tìm hiểu về cơ sở dữ liệu MongoDB

### 2.8.1 Giới thiệu về MongoDB

Khái niệm về cơ sở dữ liệu phi quan hệ (NoSql)

- NoSQL là 1 dạng CSDL mã nguồn mở và được viết tắt bởi: None-Relational SQL hay có nơi thường gọi là Not-Only SQL.
- NoSQL được phát triển trên Javascript Framework với kiểu dữ liệu là JSON và dạng dữ liệu theo kiểu key và value.

- NoSQL ra đời như là 1 mảnh vá cho những khuyết điểm và thiếu sót cũng như hạn chế của mô hình dữ liệu quan hệ RDBMS (Relational Database Management System - Hệ quản trị cơ sở dữ liệu quan hệ) về tốc độ, tính năng, khả năng mở rộng, ...
- Với NoSQL bạn có thể mở rộng dữ liệu mà không lo tới những việc như tạo khóa ngoại, khóa chính, kiểm tra ràng buộc.v.v ...
- NoSQL bỏ qua tính toàn vẹn của dữ liệu và transaction để đổi lấy hiệu suất nhanh và khả năng mở rộng.
- NoSQL được sử dụng ở rất nhiều công ty, tập đoàn lớn, ví dụ như FaceBook sử dụng Cassandra do FaceBook phát triển, Google phát triển và sử dụng BigTable, ...

#### Khái niệm về MongoDB [14]

- MongoDB là một hệ quản trị cơ sở dữ liệu mã nguồn mở, là CSDL thuộc NoSql và được hàng triệu người sử dụng.
- MongoDB là một database hướng tài liệu (document), các dữ liệu được lưu trữ trong document kiểu JSON thay vì dạng bảng như CSDL quan hệ nên truy vấn sẽ rất nhanh.
- Với CSDL quan hệ chúng ta có khái niệm bảng, các cơ sở dữ liệu quan hệ (như MySQL hay SQL Server...) sử dụng các bảng để lưu dữ liệu thì với MongoDB chúng ta sẽ dùng khái niệm là collection thay vì bảng
- So với RDBMS thì trong MongoDB collection ứng với table, còn document sẽ ứng với row, MongoDB sẽ dùng các document thay cho row trong RDBMS.
- Các collection trong MongoDB được cấu trúc rất linh hoạt, cho phép các dữ liệu lưu trữ không cần tuân theo một cấu trúc nhất định.
- Thông tin liên quan được lưu trữ cùng nhau để truy cập truy vấn nhanh thông qua ngôn ngữ truy vấn MongoDB

#### 2.8.2 Các thuật ngữ hay sử dụng trong MongoDB

Một số khái niệm được sử dụng trong MongoDB:

- \_id: Là trường bắt buộc có trong mỗi document. Trường \_id đại diện cho một giá trị duy nhất trong document MongoDB. Trường \_id cũng có thể được hiểu là khóa chính trong document. Nếu bạn thêm mới một document thì MongoDB sẽ tự động sinh ra một \_id đại diện cho document đó và là duy nhất trong cơ sở dữ liệu MongoDB.

- Collection: Là nhóm của nhiều document trong MongoDB. Collection có thể được hiểu là một bảng tương ứng trong cơ sở dữ liệu RDBMS (Relational Database Management System). Collection nằm trong một cơ sở dữ liệu duy nhất. Các collection không phải định nghĩa các cột, các hàng hay kiểu dữ liệu trước.
- Cursor: Đây là một con trỏ đến tập kết quả của một truy vấn. Máy khách có thể lặp qua một con trỏ để lấy kết quả.
- Database: Nơi chứa các Collection, giống với cơ sở dữ liệu RDMS chúng chứa các bảng. Mỗi Database có một tập tin riêng lưu trữ trên bộ nhớ vật lý. Một máy chủ MongoDB có thể chứa nhiều Database.
- Document: Một bản ghi thuộc một Collection thì được gọi là một Document. Các Document lần lượt bao gồm các trường tên và giá trị.
- Field: Là một cặp name – value trong một document. Một document có thể có không hoặc nhiều trường. Các trường giống các cột ở cơ sở dữ liệu quan hệ.
- Index: Là những cấu trúc dữ liệu đặc biệt, dùng để chứa một phần nhỏ của các tập dữ liệu một cách dễ dàng để quét. Chỉ số lưu trữ giá trị của một fields cụ thể hoặc thiết lập các fields, sắp xếp theo giá trị của các fields này. Index hỗ trợ độ phân tích một cách hiệu quả các truy vấn. Nếu không có chỉ mục, MongoDB sẽ phải quét tất cả các documents của collection để chọn ra những document phù hợp với câu truy vấn. Quá trình quét này là không hiệu quả và yêu cầu MongoDB để xử lý một khối lượng lớn dữ liệu.

### 2.8.3 Một số câu lệnh cơ bản trên MongoDB

**Bảng 1 Một số câu lệnh cơ bản trên MongoDB**

Tạo CSDL	use test;
Tạo bảng	db.createCollection('students');
Tạo bản ghi	db.students.insert({ name:'thanh', gender: 'male'});
Cập nhật	db.students.update({ _id: 1 },{\$set:{ name: 'thanh update' }})

Xoá bảng ghi	db.students.remove({ _id: 1 });
Tìm kiếm tất cả	db.students.find({});
Tìm kiếm có điều kiện	db.students.find({ name: 'thanh' });

#### 2.8.4 Ưu điểm và nhược điểm của MongoDB

Ưu điểm:

- Bởi vì Mongodbs sử dụng các dữ liệu dưới dạng Document JSON nên mỗi một collection đều có kích cỡ và document khác nhau. Nhưng chúng lại rất linh hoạt khi thực hiện lưu trữ bởi vậy nếu bạn muốn thứ gì thì chỉ cần insert thoải mái.
- Các dữ liệu có trong Mongodbs thường không ràng buộc lẫn nhau, chúng không có join như trong RDBMS, nên khi bạn insert, xóa hoặc update thì sẽ không phải bỏ ra quá nhiều thời gian để kiểm tra chúng có thỏa mãn các ràng buộc như trong RDBMS hay không.
- Mongodbs dễ mở rộng được, và trong Mongodbs luôn có khái niệm cluster chính là cụm các node sẽ có chứa các dữ liệu giao tiếp với nhau. Nên chỉ cần bạn muốn mở rộng hệ thống thì chỉ việc thêm một node mới vào cluster.
- Các trường hợp dữ liệu “\_id” sẽ luôn được đánh tự động index, nên tốc độ truy vấn thông tin sẽ luôn đạt hiệu suất cao nhất.
- Nếu như có một truy vấn dữ liệu, thì bản ghi sẽ được cached lên bộ nhớ Ram. Khi đó, việc phục vụ lượt truy vấn sau sẽ diễn ra nhanh hơn mà bạn không cần phải đọc từ ổ cứng.
- Tốc độ truy vấn của Mongodbs luôn nhanh hơn so với các hệ quản trị cơ sở dữ liệu quan hệ. Nhờ có một lượng đủ dữ liệu nên việc thử nghiệm cho thấy tốc độ insert của Mongodbs sẽ nhanh gấp 100 lần so với MySQL.

Nhược điểm:

- Mongodbs không sở hữu các tính chất ràng buộc như trong RDBMS nên khi bạn thao tác với Mongodbs cần phải cẩn thận hết sức.
- Có thể sẽ tồn bộ nhớ do dữ liệu được lưu trữ dưới dạng key-value, nên các collection sẽ chỉ khác về value do vậy mà key có thể sẽ bị lặp lại. Mongodbs còn không hỗ trợ join nên rất dễ bị dữ thừa dữ liệu.

- Khi thực hiện insert/update/remove bản ghi thì MongoDB sẽ chưa thể cập nhật ngay vào ổ cứng. Chỉ sau 60 giây thì Mongodbs mới có thể ghi được toàn bộ dữ liệu được thay đổi từ RAM xuống phần ổ cứng. Điều này chính là nhược điểm bởi nó có thể mang lại nguy cơ mất dữ liệu khi các tình huống xấu như mất điện xảy ra.

### 3. THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG

#### 3.1 Yêu cầu

- Có các cổng RS485 để giao tiếp với các thiết bị chuẩn Modbus có sẵn trên thị trường, cổng RS232, UART để dễ dàng cài đặt, hiệu chỉnh.
- Có cổng Ethernet để triển khai mô hình Modbus TCP hoặc các dịch vụ trên lớp mạng để giao tiếp với IoT Platform trên server.
- Có khe cắm SD Card để lưu các thông tin cấu hình cho Gateway và các sự kiện.
- Phần cứng vi điều khiển đủ mạnh để đảm bảo chạy các ứng dụng một cách ổn định.
- Đèn báo trạng thái để thông báo cho người dùng.
- Hỗ trợ tối đa 2 cổng RS485 để quản lý được nhiều thiết bị

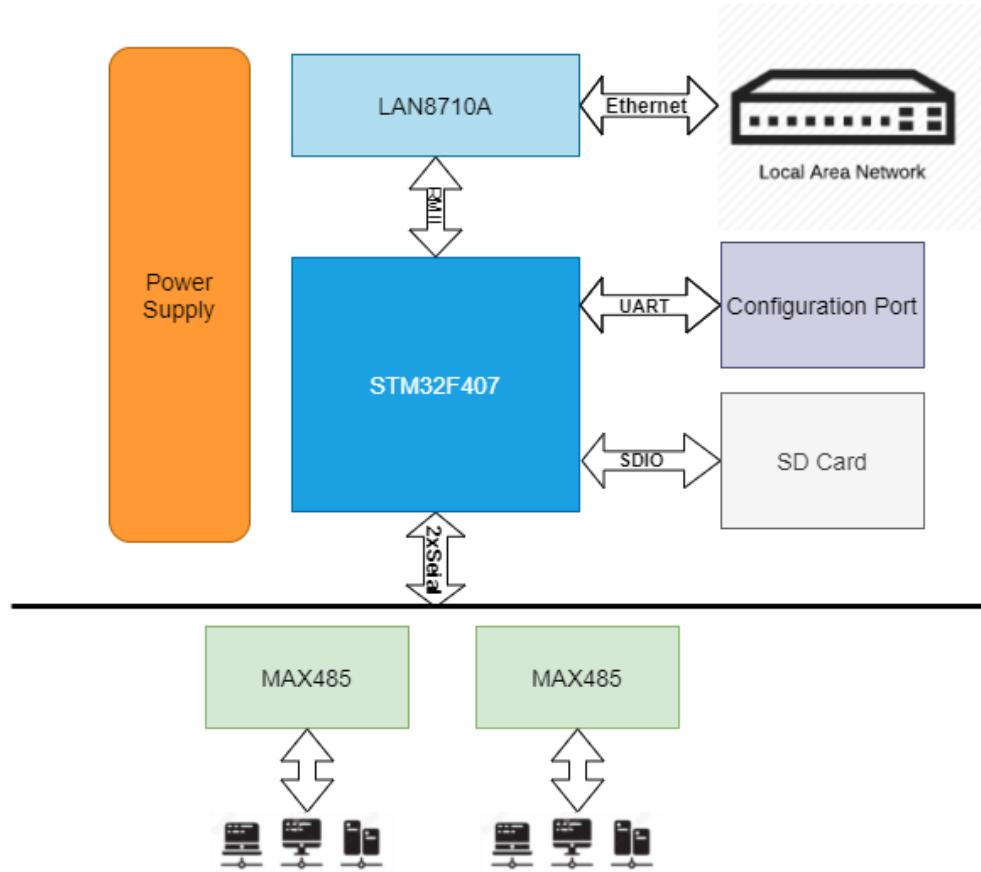
#### 3.2 Phân tích

**Bảng 2 Phân tích các phương án cho Gateway Modbus Serial**

Phương án	Mô tả hoạt động	Ưu điểm	Nhược điểm
Sử dụng STM32F407 làm vi điều khiển chính cho Gateway	Triển khai Modbus Protocol ở chế độ master. Kết nối mạng nội bộ LAN để có thể gửi nhận dữ liệu tổng hợp được đến IoT Platform trên server Các thông báo chức năng qua đèn Led	Cấu hình và ngoại vi cơ bản đáp ứng được yêu cầu. Được dùng phổ biến nên có sẵn các thư viện cơ bản, cộng đồng hỗ trợ động đảo. Dễ tìm thấy và mua ở thị trường trong nước, giá thành vừa phải khi	

		so cùng cấu hình so với một số nhà sản xuất khác	
Sử dụng IC LAN8710A	IC giao tiếp Ethernet, hỗ trợ tốc độ cao 10/100 Mbps	Rẻ, nhỏ gọn, giao tiếp RMII dễ sử dụng. Đơn giản hóa trong việc xử lý các gói ở lớp PHY	
Sử dụng module đổi mức tín hiệu MAX485	Module chuyển tiếp UART-TTL sang RS485	Chuẩn giao tiếp thích hợp cho tốc độ cao và truyền xa. Chuẩn giao tiếp hỗ trợ kết nối đa điểm	Một số IC không chính hãng hoạt động không ổn định
Sử dụng module thẻ SD	Lưu các thông tin cấu hình Gateway và các sự kiện	Hỗ trợ chuẩn SDIO/SPI giúp tăng tốc độ truy cập dữ liệu	Một số module hỗ trợ SDIO ngoài thị trường hoạt động không ổn định

### 3.3 Sơ đồ khái tổng quát



Hình 3-1 Sơ đồ khái tổng quát của Gateway Modbus Serial

Khối vi điều khiển STM32F407ZG đóng vai trò làm trung tâm xử lý dữ liệu, bao gồm các nhiệm vụ chính:

- Giao tiếp với MAX485 để tương tác với các thiết bị trọng mạng thông qua Modbus protocol.
- Giao tiếp UART để cung cấp môi trường cài đặt với máy tính.
- Giao tiếp với LAN8720 để kết nối đến mạng nội bộ thông qua cổng giao tiếp Ethernet.
- Giao tiếp với SD card để lấy thông tin cài đặt cho gateway và lưu các sự kiện.

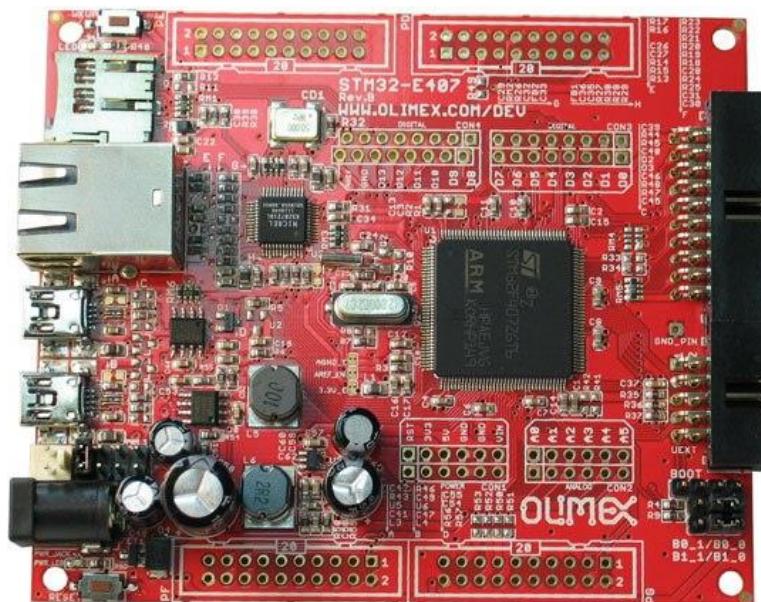
Các khối khác bao gồm: Khối nguồn để đảm bảo các mức điện áp ổn định cho các linh kiện hoạt động chính xác. MAX485 để chuyển tiếp qua lại tín hiệu UART và RS485. Khối LAN8720 để

giao tiếp và xử lý các gói tin ở lớp PHY qua cổng Ethernet. Khối SD Card để giao tiếp và xử lý các thông tin cấu hình Gateway và các sự kiện.

### 3.4 Sơ đồ mạch chi tiết

Do thời gian gấp rút và khả năng thiết kế mạch còn hạn chế nên nhóm sử dụng development board Olimex STM32-E407 để đảm bảo thời gian thực hiện hệ thống đúng hạn.

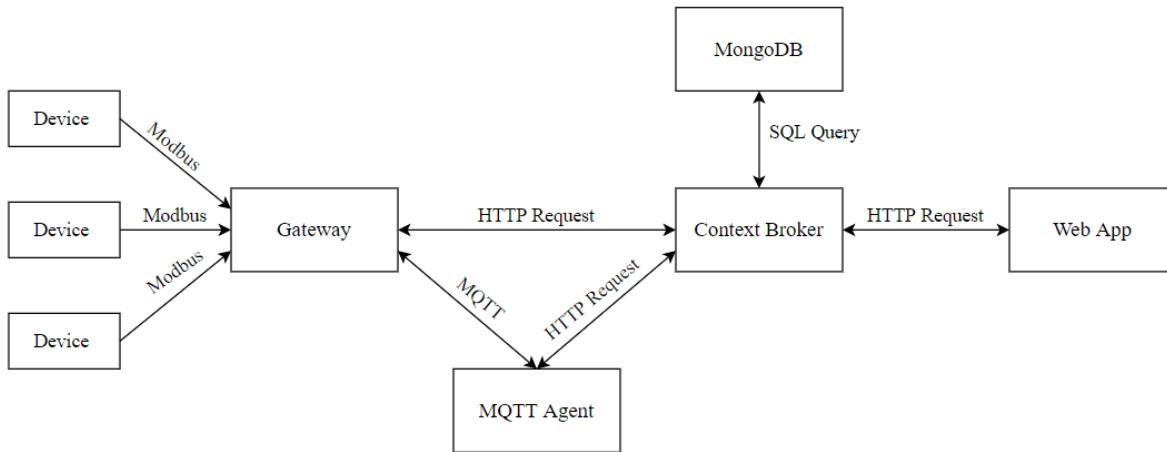
OLIMEX-STM32-E407 là một phần cứng mã nguồn mở, dựa trên ST Microelectronics STM32F407ZG ARM Cortex-M4 CPU



Hình 3-2 Olimex STM32-E407 Cortex-M4 Dev Board

## 4. THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM

### 4.1 Sơ đồ khái tổng quát của hệ thống



Hình 4-1 Sơ đồ khái tổng quát của hệ thống

Giải thích:

- Device: là các thiết bị trong hệ thống giám sát năng lượng mặt trời. Thiết bị ở đây có thể là Inverter, Energy Meter hoặc là trạm Weather Station.
- Gateway: đóng vai trò trung gian trong việc truyền dữ liệu giữa device và server. Gateway sẽ thu thập dữ liệu của các device thông qua giao thức modbus, sau đó chuyển tiếp dữ liệu qua server (Context Broker) bằng giao thức MQTT hoặc HTTP.
- MQTT Agent: hỗ trợ gateway trong việc truyền dữ liệu đến Context Broker bằng giao thức MQTT. MQTT Agent đóng vai trò là subscriber, sẽ nhận dữ liệu từ gateway qua giao thức MQTT, sau đó chuyển tiếp dữ liệu qua Context Broker bằng giao thức HTTP.
- Context Broker: nhận dữ liệu được gửi lên từ gateway vào lưu dữ liệu vào cơ sở dữ liệu. Ngoài ra, Context Broker còn cung cấp các API cho web app sử dụng trong việc tạo các entity (là các đối tượng ảo đại diện cho các thiết bị thực trong hệ thống trên server) và lấy dữ liệu.

- Web app: là ứng dụng web được xây dựng với các chức năng đăng nhập, đăng xuất, đăng ký, quản lý và phân quyền người dùng. Cung cấp giao diện trong việc giám sát và quản lý một hệ thống năng lượng mặt trời.

## 4.2 Xây dựng khôi IoT Gateway

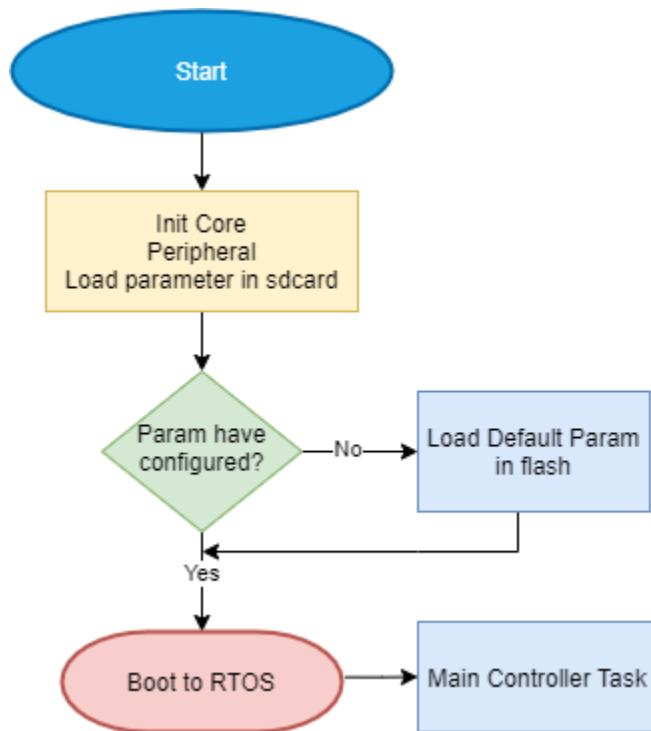
### 4.2.1 Phần mềm trên IoT Gateway

#### 4.2.1.1 Yêu cầu

- Gateway phải chạy được Modbus Stack ở chế độ Master để thu thập, thay đổi, quản lý dữ liệu cho các thiết bị đầu cuối. Đồng thời để nâng cao tối đa số lượng thiết bị trong mạng, 2 cổng RS485 trên board đều phải được khai thác, có thể xem như gateway quản lý 2 mạng Modbus riêng biệt.
- Gateway được kết nối đến IoT Platform thông qua MQTT, trong đó mỗi Gateway sẽ có địa chỉ ID định danh riêng trong mạng, đồng thời đảm bảo 2 luồng dữ liệu chính là downlink (gói tin yêu cầu từ IoT Server) và uplink (gói tin phản hồi yêu cầu, gói tin đồng bộ thông tin thiết bị và gói tin mang dữ liệu).
- Gateway phải có chế độ tùy chỉnh, bao gồm: các thông số liên quan đến địa chỉ IP của Gateway trong mạng (Static IP, Netmask, Default Gateway), Gateway ID, MQTT Server IP và các thông số để điều chỉnh cho mạng Modbus: Baudrate, Stopbit, Parity.
- Các cơ chế thông báo lỗi, đèn Led thông báo trạng thái đường truyền và lỗi trong quá trình chạy thực nghiệm.

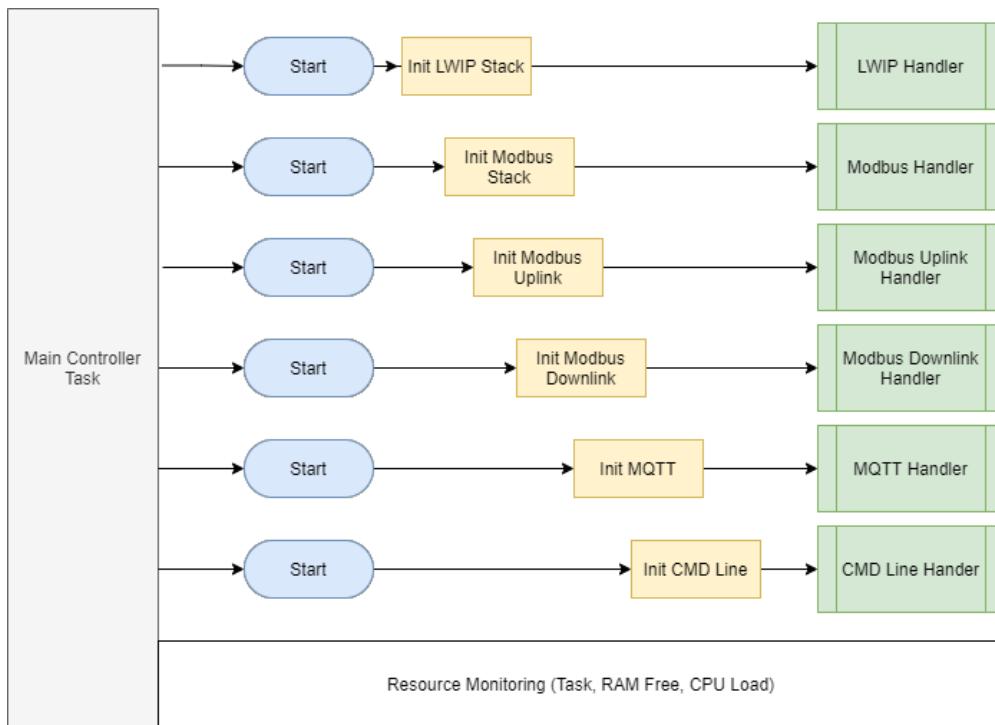
#### 4.2.1.2 Lưu đồ giải thuật

#### Quá trình khởi động IoT Gateway



Hình 4-2 Quá trình khởi động của gateway

Quá trình khởi động của Gateway: vi điều khiển STM32 sẽ cài đặt các thông số hoạt động cơ bản và khởi tạo các ngoại vi, sau đó sẽ tìm đến vùng nhớ lưu trữ thông số cài đặt (trên vùng nhớ Flash hoặc SD Card). Nếu các thông số đã được thiết đặt, Gateway sẽ tiến hành load các cài đặt và bắt đầu boot vào kernel của RTOS, ngược lại sẽ tiến hành load các thông số mặc định.



**Hình 4-3 Các khối nhiệm vụ của gateway**

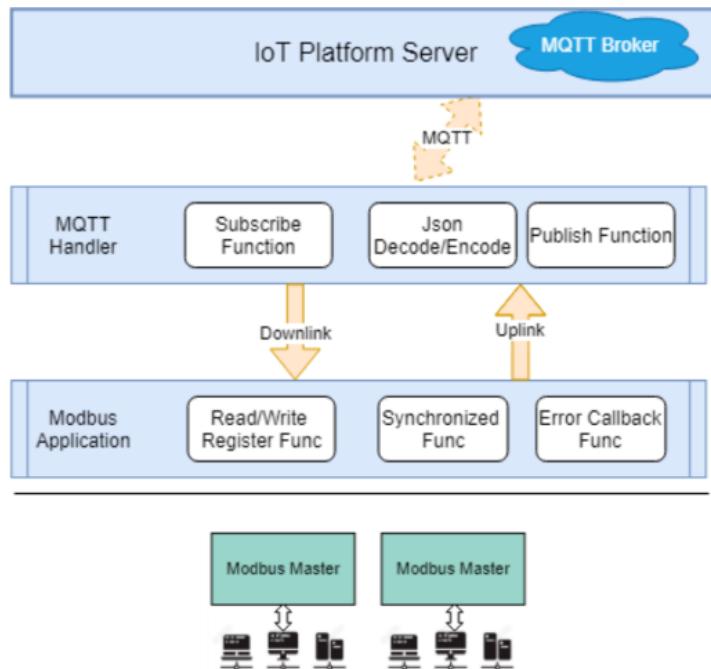
Sau khi boot thành công, Gateway sẽ tiến vào Main Controller Task – đây là task chính của Gateway, bao gồm các cơ chế tạo, quản lý các task sau này. Cũng trên Main Controller Task, các cơ chế monitoring tài nguyên hệ thống được triển khai và gửi lên người dùng thông qua Serial UART.

Để đảm bảo việc khởi động các giao thức và các ứng dụng một cách chính xác (MQTT chỉ chạy khi LWIP đã kết nối mạng, Modbus Uplink và Modbus Downlink sẽ chạy khi Modbus Stack sẵn sàng, ...), các Task phải được khởi tạo một cách có tuần tự. Các task mới chỉ được khởi động khi được cho phép bởi Main Controller Task cũng như phải phản hồi trạng thái khởi động để Main Controller Task quyết định các Task tiếp theo được khởi tạo.

*Modbus Stack:* Trong đề tài, Modbus Stack được triển khai dựa trên một thư Modbus có sẵn là FreeModbus, đây là một gói thư viện miễn phí, hỗ trợ các chế độ RTU/ASCII và TCP đối với các thiết bị nhưng kết nối được mạng, hỗ trợ nhiều nền tảng vi điều khiển nên việc porting không quá phức tạp, bên cạnh đó, FreeModbus còn được thiết kế để chạy trên các hệ điều hành thời gian thực thông dụng, việc triển khai FreeModbus không tốn nhiều tài nguyên (1 Timer và 1 Serial port),

300 bytes ram tuỳ theo các function được triển khai. Tuy nhiên, trong yêu cầu đặt ra, cần tuỳ biến thư viện để có thể chạy tổng cộng 2 master để có thể đạt được khoảng 500 thiết bị.

*Command Line:* Giao diện command line thông qua Serial để cài đặt các thông số cho Gateway



**Hình 4-4 Luồng dữ liệu của Gateway Modbus Serial**

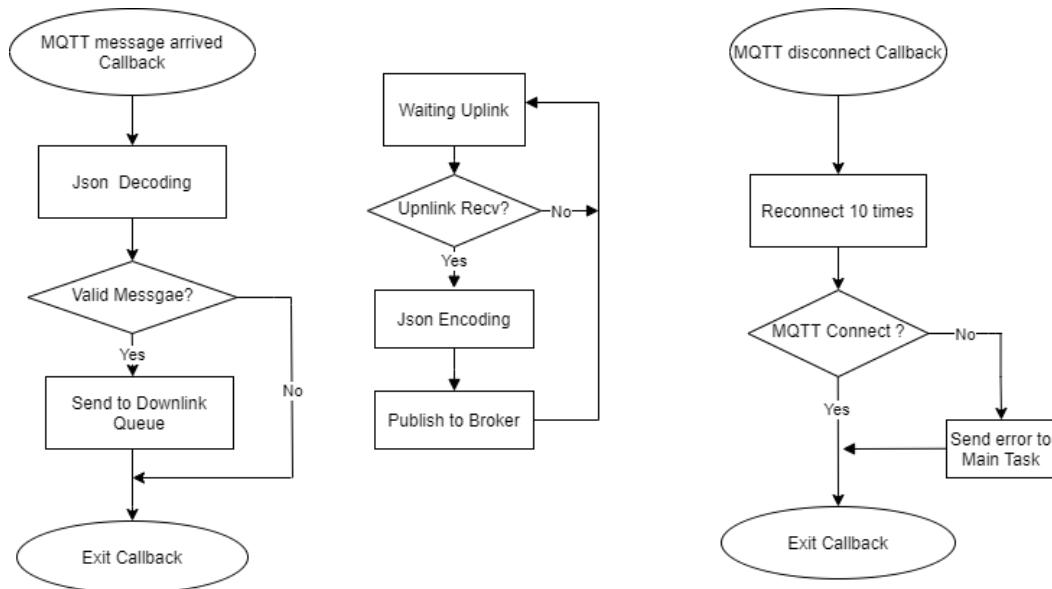
*MQTT Handler:* Trong quá trình khởi động, MQTT Handler sẽ khởi tạo dịch vụ MQTT dựa trên lớp TCP đã được cung cấp bởi thư viện LWIP khởi tạo trước đó. Sau đó, MQTT Handler sẽ tiến hành Subscribe vào Topic định danh trước (thông nhất với IoT Server và được lưu lại thông qua bộ nhớ SD), dịch vụ MQTT sẽ quản lý các gói tin giữa gateway và server, trong đê tài gói tin bao gồm 2 gói: Uplink packet từ Modbus Uplink để chuyển dữ liệu đọc được từ gateway đến server và Downlink packet từ phía server để yêu cầu gateway trả về các thông tin từ các thiết bị mà nó quản lý. Các gói tin được chuẩn hóa theo kiểu Json, gói từ uplink và downlink sẽ được gửi qua cơ chế hàng đợi queue trong FreeRTOS.

*Modbus Application:* quản lý 2 luồng dữ liệu trong mạng Modbus: Modbus Uplink và Modbus Downlink

- *Modbus Uplink:* Sau khi Modbus Stack đã sẵn sàng, Modbus Uplink sẽ được khởi tạo. Modbus Uplink phải chờ cờ cho phép từ người dùng để được phép truy cập vào tài nguyên modbus để có thể lần lượt đọc dữ liệu từ các thiết bị. Sau khi dữ liệu đọc thành công, Modbus Uplink sẽ gửi thông tin trả về từ thiết bị để chuyển dữ liệu qua luồng Uplink MQTT cho MQTT handler xử lí.
- *Modbus Downlink:* Sau khi Modbus Stack đã sẵn sàng, Modbus Downlink sẽ được khởi tạo. Modbus Downlink sẽ kiểm tra hàng đợi Queue từ khối MQTT handler và gọi các hàm cung cấp từ Modbus Stack để truy cập các thông tin liên quan đến giao thức modbus, bao gồm: Read Input Register (0x04), Read Holding Registers (0x03), Write Single Register (0x06), Read Coils (0x01), Write Single Coil (0x05). Các thông tin trả về sẽ được Modbus Stack trả về và gửi ngược lên đường Uplink packet, đồng thời sẽ báo lỗi sang Main Controller Task nếu có bất kỳ gói tin nào lỗi.

#### Quá trình xử lí các gói tin của MQTT Handler

Lưu đồ giải thuật chi tiết của MQTT Handler:



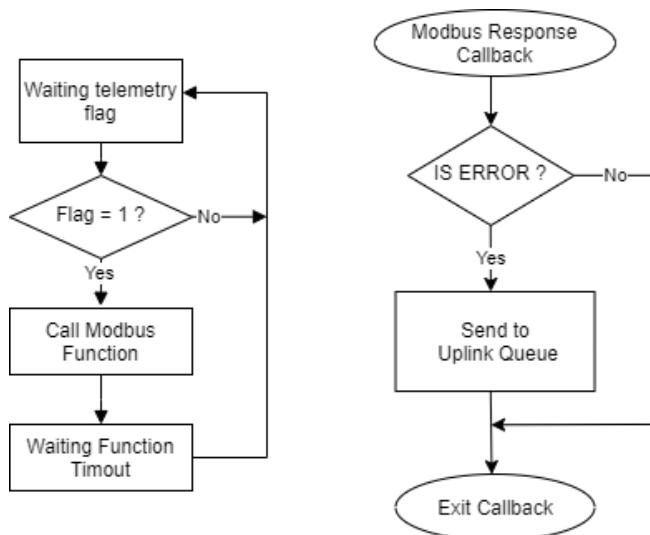
**Hình 4-5 Lưu đồ giải thuật của khối MQTT Handler**

Giải thích:

- Khởi MQTT Handler thiết lập kết nối đến Broker và đăng ký các Topic cần thiết. Từ đó tạo ra các luồng callback khi thỏa điều kiện: nhận được gói tin MQTT và lỗi trong kết nối MQTT.
- Khi nhận được dữ liệu từ luồng Modbus Uplink. Tùy theo đặc trưng của dữ liệu mà MQTT Handler sẽ có các phương thức khác nhau để đóng gói dữ liệu khác nhau, sau đó MQTT Handler gọi luồng MQTT Uplink để gửi dữ liệu lên MQTT Broker.
- Khi nhận được gói tin yêu cầu từ server sẽ được giải mã theo chuẩn JSON và kiểm tra tính hợp lệ. Nếu thỏa điều kiện, các yêu cầu của gói tin sẽ được lấy ra và gửi xuống tầng Modbus Downlink thông qua cơ chế hàng đợi của FreeRTOS.
- MQTT Handler sẽ chờ các phản hồi từ Modbus Downlink và gửi lên MQTT Broker.

#### Quá trình xử lý các gói tin của Modbus Uplink

Lưu đồ giải thuật chi tiết của Modbus Uplink:



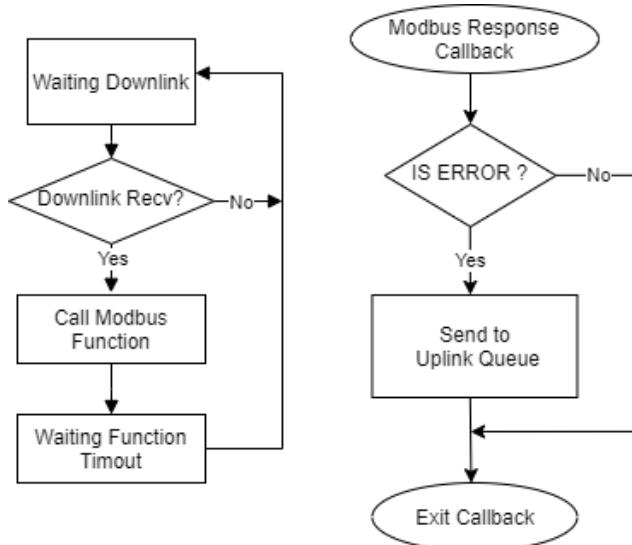
**Hình 4-6 Lưu đồ giải thuật của khối Modbus Uplink**

Giải thích:

- Modbus Uplink sẽ đợi cấp quyền sử dụng tài nguyên freemodbus từ người dùng, nếu nhận được yêu cầu đọc dữ liệu, nó sẽ gọi các hàm tương ứng để Modbus Stack xử lý.
- Modbus Stack phản hồi gói tin, dữ liệu sẽ được gửi lên MQTT Handler.

### Quá trình xử lý các gói tin của Modbus Downlink

Lưu đồ giải thuật chi tiết của Modbus Downlink:



**Hình 4-7 Lưu đồ giải thuật của khối Modbus Downlink**

Giai thích:

- Modbus Downlink sẽ đợi yêu cầu từ MQTT Handler qua cơ chế hàng đợi, nếu nhận được yêu cầu, nó sẽ gọi các hàm tương ứng để Modbus Stack xử lý.
- Khi Modbus Stack phản hồi gói tin, dữ liệu sẽ được gửi lên MQTT Handler.

### 4.2.2 Phần mềm trên máy tính

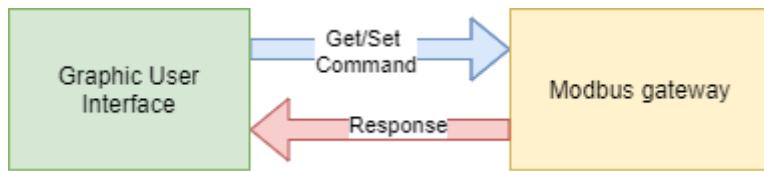
#### 4.2.2.1 Yêu cầu

- Cung cấp giao diện kết nối với Gateway thông qua cổng serial
- Cung cấp giao diện hiển thị thông tin và thay đổi các giá trị trên Gateway

#### 4.2.2.2 Lưu đồ giải thuật

Phần mềm đặt ra 2 luồng dữ liệu chính:

- Các lệnh Get, Set để lấy các thông tin từ Gateway hiển thị lên giao diện và gửi các thông tin từ giao diện xuống Gateway.
- Các phản hồi từ Gateway tương ứng.



**Hình 4-8 Luồng dữ liệu của phần mềm trên máy tính**

### 4.3 Xây dựng khái niệm Context Broker

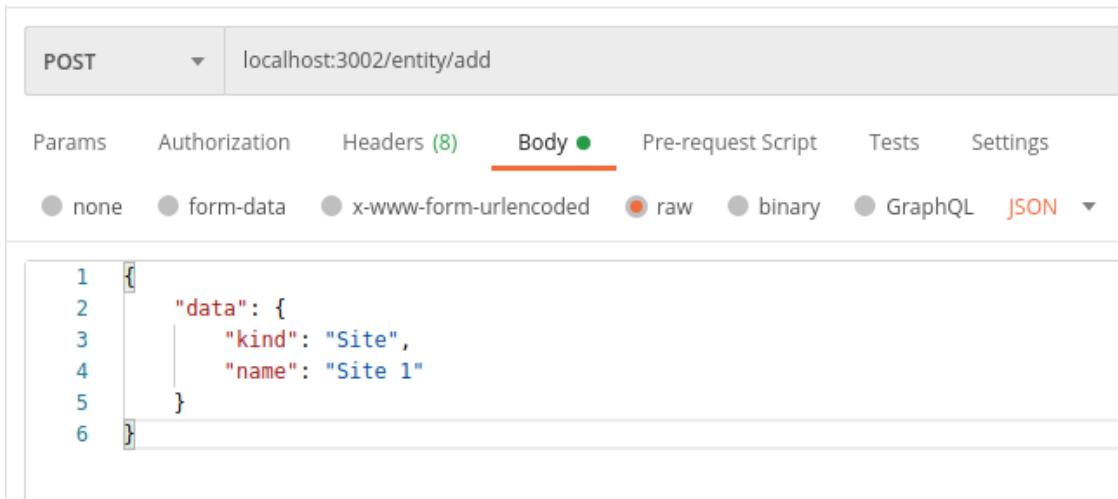
Context Broker là phần backend server được xây dựng trên nền tảng NodeJS. Context Broker được xây dựng như một dịch vụ cung cấp các API với các chức năng sau:

- Tạo các entity: Entity là một đối tượng ảo được tạo ra trên server đại diện cho một đối tượng thực trong hệ thống. Entity có thể là: Site (một đối tượng đại diện cho một trạm năng lượng mặt trời), Gateway (một gateway ảo được liên kết với gateway thực lớp vật lý về mặt dữ liệu. Gateway được tạo sẽ thuộc về một Site cụ thể), Device (một đối tượng ảo tượng trưng cho các thiết bị vật lý bên dưới. Device có thể là một Inverter, một Energy Meter, hay một Weather Station. Device cũng sẽ thuộc về một gateway cụ thể. Một gateway sẽ quản lý nhiều Device.)
- Cấu hình gateway: trước khi bắt đầu gửi dữ liệu, gateway cần phải gửi một gói tin provision để cấu hình. Gói tin provision sẽ chứa các thông tin về các device thuộc gateway đó và các channel của từng device.
- Lưu dữ liệu vào cơ sở dữ liệu: sau khi gateway gửi gói provision, gateway sẽ tiến hành thu thập dữ liệu từ các device và tiến hành gửi các gói tin telemetry. Đây là các gói tin dữ liệu. Context broker nhận dữ liệu và tiến hành lưu dữ liệu vào cơ sở dữ liệu.
- Cung cấp các phương thức truy cập dữ liệu: Context Broker cung cấp các phương thức trong việc lấy dữ liệu. Dữ liệu có thể được truy cập theo ngày, tháng, hoặc năm. Ngoài ra context broker còn cung cấp các bộ lọc giúp người dùng có thể lấy tất cả dữ liệu hoặc chỉ lấy các giá trị đầu cuối.

Dưới đây sẽ trình bày chi tiết các API được cung cấp bởi Context Broker:

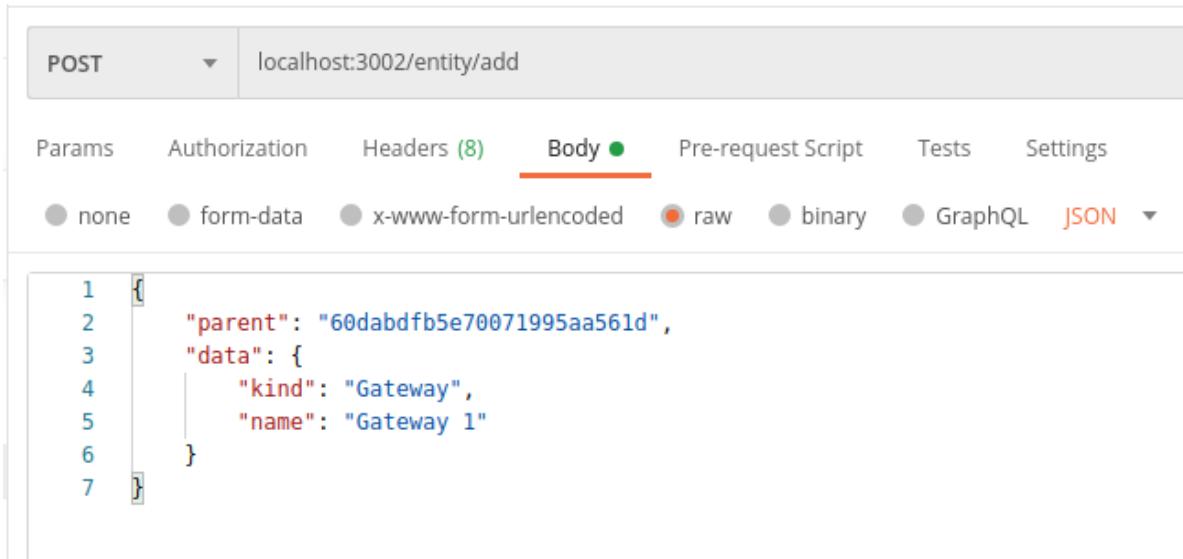
#### 4.3.1 API tạo Entity

Tạo entity là một phần tử gốc: API này giúp ta tạo ra một entity gốc không thuộc về một entity nào khác. Ta có thể thêm bất kỳ một thuộc tính nào của entity vào trường data.



Hình 4-9 API tạo entity là một phần tử gốc

Tạo entity là con của một entity khác: API này giúp ta tạo một entity là con trực tiếp của một entity khác. Trường parent sẽ là id của entity cha. Tương tự, ta cũng có khả năng thêm bất kỳ thuộc tính nào vào trong trường data.



Hình 4-10 API tạo entity là con của một Entity khác

### 4.3.2 API truy cập Entity

Truy cập entity bằng id: mỗi entity khi được tạo ra sẽ được gán cho một id, và ta có thể sử dụng id này để truy cập entity đó. Trường attrs sẽ chứa các thuộc tính mà ta truy cập của entity đó.

KEY	VALUE
id	60d1e3e782cf30796c5dd515
attrs	kind,name
Key	Value

**Hình 4-11 API truy cập entity bằng id**

Truy cập các entity là con trực tiếp của một Entity: API này giúp ta sẽ trả về tất cả các entity mà là con trực tiếp của một entity khác. Trường parent là id của entity cha.

KEY	VALUE
parent	60d1e3e782cf30796c5dd515
Key	Value

**Hình 4-12 API truy cập các entity là con trực tiếp của một entity khác**

Truy cập các entity cùng chung một gốc: API này sẽ trả về tất cả các entity của thuộc về một gốc. Trường ancestor sẽ là id của entity gốc.

KEY	VALUE
ancestor	60d1e3e782cf30796c5dd515
Key	Value

**Hình 4-13 API truy cập các Entity cùng chung một gốc**

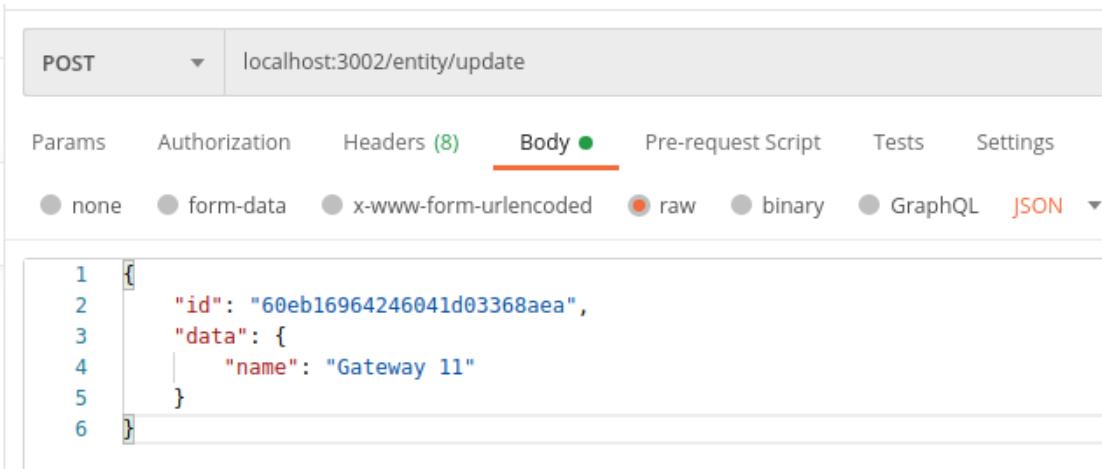
Truy cập các entity thông qua các thuộc tính của entity đó: API này sẽ trả về các entity có thuộc tính giống với thuộc tính ta muốn. Trong ví dụ dưới đây, API này sẽ trả về tất cả các entity mà có thuộc tính “kind” bằng “Inverter”.

KEY	VALUE
kind	Inverter
Key	Value

**Hình 4-14 API truy cập các entity thông thuộc tính của entity đó**

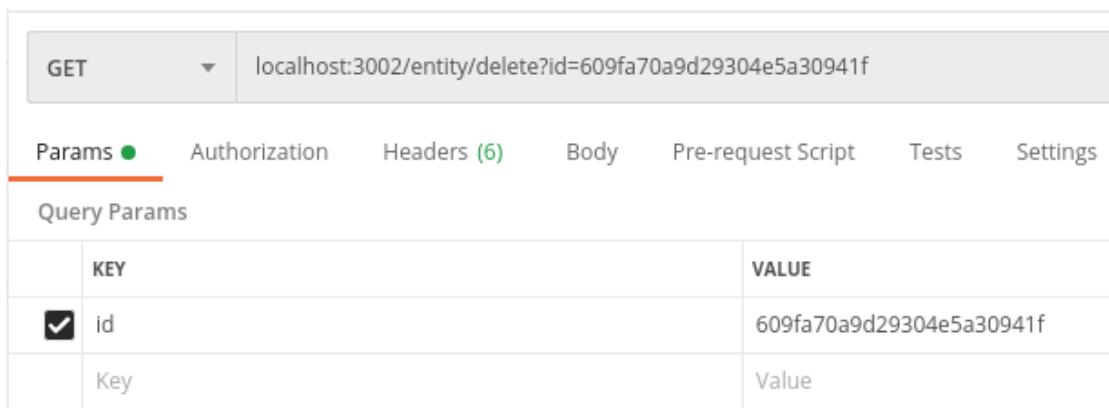
### 4.3.3 API cập nhật Entity

Cập nhật entity bằng id: sau khi tạo entity, chúng ta có thể thay đổi bất kỳ thuộc tính nào của entity đó, hoặc có thể thêm một thuộc tính mới cho entity đó. API này giúp ta cập nhật entity thông qua id của entity đó.

**Hình 4-15 API cập nhật entity thông qua id**

#### 4.3.4 API xoá Entity

Xoá entity bằng id: API này giúp ta xoá một entity thông id của entity đó.

**Hình 4-16 API xoá một entity bằng id**

Xoá các entity là con trực tiếp của một entity: API sẽ giúp ta xoá tất cả các entity là con trực tiếp của một entity khác. Trường parent là id của entity cha.

KEY	VALUE
parent	60dabdfb5e70071995aa561d
Key	Value

**Hình 4-17 API xoá các entity là con trực tiếp của một entity khác**

Xoá các entity cùng thuộc về một entity gốc: API này giúp ta xoá tất cả các entity cùng chung một gốc. Trường ancestor là id của entity gốc.

KEY	VALUE
ancestor	60867cddecdf3669ba29e39e
Key	Value

**Hình 4-18 API xoá các entity cùng chung một gốc**

Xoá các entity thông qua thuộc tính của entity: API này giúp ta xoá tất cả các entity mà có thuộc tính giống với thuộc tính mà ta muốn. Trong ví dụ dưới đây, API này sẽ xoá tất cả các entity mà có thuộc tính “kind” bằng “Inverter”.

GET    localhost:3002/entity/delete?kind=Inverter

Params ●    Authorization    Headers (6)    Body    Pre-request Script    Tests    Settings

Query Params

	KEY	VALUE
<input checked="" type="checkbox"/>	kind	Inverter
	Key	Value

**Hình 4-19 API xoá các entity thông qua thuộc tính**

#### 4.3.5 API cấu hình gateway

Kiểm tra trạng thái provision: Trước khi gateway gửi dữ liệu lên server, cần phải gửi gói tin provision trước để cấu hình gateway. Gateway sẽ có hai trạng thái provision: ended và pending. Chỉ khi trạng thái được chuyển sang pending thì context broker mới tiếp nhận gói tin provision từ phía gateway. API này sẽ trả về trạng thái provision của gateway. Trường entity sẽ là id của gateway.

GET    localhost:3002/provision/status?entity=60d1e3970340f939d390e8b6

Params ●    Authorization    Headers (6)    Body    Pre-request Script    Tests    Settings

Query Params

	KEY	VALUE
<input checked="" type="checkbox"/>	entity	60d1e3970340f939d390e8b6
	Key	Value

**Hình 4-20 API kiểm tra trạng thái provision**

Cho phép provision: API này sẽ chuyển trạng thái provision của gateway sang trạng thái pending. Lúc này, Context Broker mới tiếp nhận gói tính provision của gateway. Trường entity là id của gateway, trường timeout là thời gian mà trạng thái provision của gateway ở trạng thái pending. Sau thời gian timeout, context broker sẽ tự động chuyển trạng thái provision về trạng thái ended.

The screenshot shows the Postman interface with a GET request to `localhost:3002/provision/begin?entity=60d1e3970340f939d390e8b6&timeout=120`. The 'Params' tab is selected, displaying two query parameters: 'entity' with value `60d1e3970340f939d390e8b6` and 'timeout' with value `120`. There is also a 'Key' row with a 'Value' column.

KEY	VALUE
entity	<code>60d1e3970340f939d390e8b6</code>
timeout	<code>120</code>
Key	Value

**Hình 4-21 API cho phép provision**

Kết thúc provision: API này sẽ trạng thái provision của gateway sang trạng thái ended, kết thúc quá trình provision. Trường entity là id của gateway.

The screenshot shows the Postman interface with a GET request to `localhost:3002/provision/end?entity=60d1e3970340f939d390e8b6`. The 'Params' tab is selected, displaying one query parameter: 'entity' with value `60d1e3970340f939d390e8b6`. There is also a 'Key' row with a 'Value' column.

KEY	VALUE
entity	<code>60d1e3970340f939d390e8b6</code>
Key	Value

**Hình 4-22 API kết thúc provision**

Gửi gói provision: Sau khi cho phép provision, API này sẽ giúp gateway gửi gói tin provision chứa các thông tin cấu hình gateway. Trường entity là id của gateway. Trường data sẽ chứa các thông tin về device và các channel của device đó.

The screenshot shows a Postman interface with the following details:

- Method: POST
- URL: http://localhost:3002/provision
- Body tab is selected.
- JSON is selected as the raw data type.
- The body content is a JSON object representing a provision package:

```
1  [
2   "entity": "60f197c7bb3058a471acd852",
3   "data": [
4     {
5       "device_id": "01",
6       "device_name": "inverter01",
7       "device_kind": "inverter",
8       "device_channels": [
9         {
10          "channel_id": "01",
11          "channel_name": "Inverter_Operation_State",
12          "channel_type": "string"
13        },
14        {
15          "channel_id": "02",
16          "channel_name": "Inverter_Production",
17          "channel_type": "number"
18        },
19        {
20          "channel_id": "03",
21          "channel_name": "Inverter_Active_Power",
22          "channel_type": "number"
23        },
24        {
25          "channel_id": "04",
26          "channel_name": "Inverter_Reactive_Power",
27          "channel_type": "number"
28        }
      ]
    }
  ]
}
```

Hình 4-23 API gửi gói provision

Truy cập dữ liệu của gói tin provision: API này giúp ta truy cập các thông tin trong gói provision. Trường entity là id của gateway.

KEY	VALUE
entity	60eafa2f2207735e235abe8a
Key	Value

**Hình 4-24 API truy cập dữ liệu gói tin provision**

#### 4.3.6 API gửi dữ liệu

Gửi gói tin telemetry: sau khi gửi gói tin provision, gateway sẽ tiến hành gửi các gói tin telemetry chứa dữ liệu thu thập từ các device. Trường entity là id của gateway. Trường timestamp là thời điểm gateway đọc dữ liệu từ các device. Trường data sẽ chứa các dữ liệu thu thập được từ các device.

```

1 {
2   "entity": "60ea95bed8cb7119bee407da",
3   "timestamp": "2021-07-11T00:05:00.000Z",
4   "data": {
5     "01": {
6       "01": 111,
7       "02": 61,
8       "03": 111,
9       "04": 61
10    }
11  }
12 }
```

**Hình 4-25 API gửi gói tin telemetry**

### 4.3.7 API truy cập dữ liệu

Lấy dữ liệu records: API này giúp ta có thể truy cập dữ liệu các channel của các device thuộc về gateway.

- Trường id: đây là id của device mà chúng ta muốn truy cập dữ liệu
- Trường attrs: chứa tên các channel thuộc về device mà chúng ta muốn truy cập dữ liệu
- Trường interval: chứa interval mà chúng ta muốn truy cập. Trường này sẽ chứa một trong các giá trị sau : “year”, “month”, “day”, “hour”, “30m”, “15m”, “10m”, “minute”, “30s”, “15s”, “10s”, “second”.
- Trường filter: đây là trường chứa thông tin bộ lọc dữ liệu, thông qua trường này ta có thể truy cập tất cả dữ liệu, hoặc chỉ lấy các giá trị đầu cuối... Trường này có thể nhận một hoặc nhiều các giá trị sau: “all”, “first”, “last”, “max”, “min”, “avg”.
- Trường date: trường này sẽ chứa thông tin về thời gian ta muốn truy cập dữ liệu. Một số ví dụ về giá trị của trường này: “2021”, “2021,07”, “2021,07,11”.
- Trường from và to: hai trường này sẽ lần lượt chứa thông tin về mốc thời gian bắt đầu và mốc thời gian kết thúc của dữ liệu mà ta muốn truy cập.

KEY	VALUE
<input checked="" type="checkbox"/> id	60eb07a74246041d03368ae4
<input checked="" type="checkbox"/> attrs	InverterProduction
<input checked="" type="checkbox"/> interval	day
<input checked="" type="checkbox"/> filter	all
<input checked="" type="checkbox"/> date	2021,07,12
<input type="checkbox"/> from	2021,07,11
<input type="checkbox"/> to	2021,07,13
Key	Value

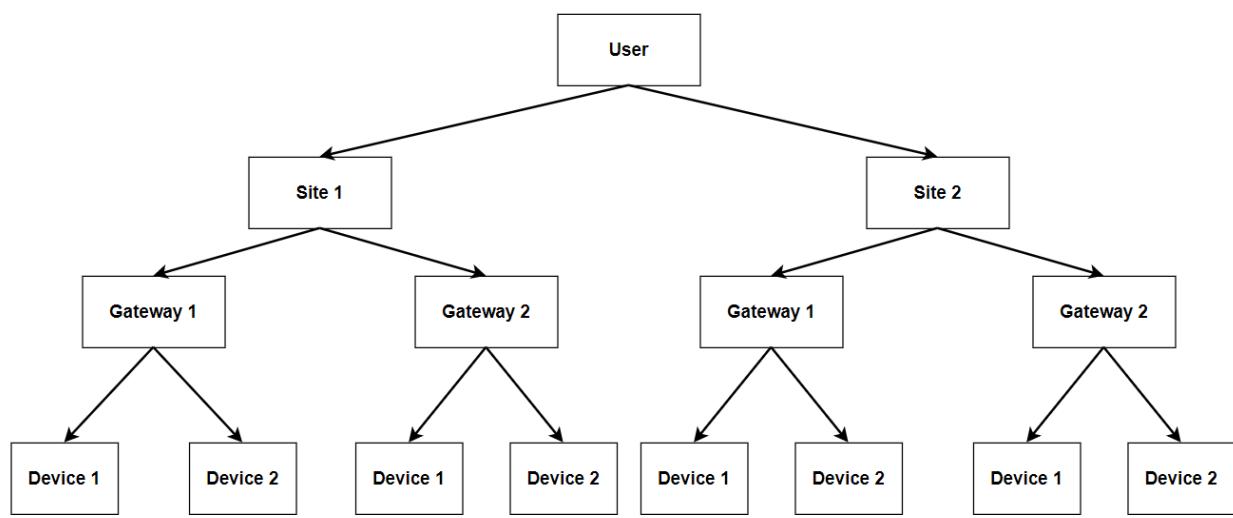
Hình 4-26 API truy cập dữ liệu

## 4.4 Xây dựng Web App

### 4.4.1 Mô hình quản lý và cách thức hoạt động của Web App

#### 4.4.1.1 Mô hình quản lý của Web App

Mô hình quản lý của web App:



**Hình 4-27 Mô hình quản lý Web App**

Trong đó:

- User: là tài khoản người dùng, một tài khoản có thể quản lý một hoặc nhiều Site.
- Site: là một đối tượng ảo tượng trưng cho một trạm năng lượng mặt trời. Site sẽ lưu trữ các thông tin về tên, địa điểm, chủ sở hữu, và công suất đỉnh của một trạm năng lượng mặt trời. Ứng dụng web có khả năng tạo ra và quản lý nhiều Site tương ứng với các trạm Solar khác nhau. Mỗi Site sẽ được phân bổ cho một tài khoản người dùng. Người dùng có thể truy cập các Site thuộc phân quyền của mình.
- Gateway: là một gateway ảo được liên kết với gateway thực lớp vật lý về mặt dữ liệu. Gateway được tạo sẽ thuộc về một Site cụ thể. Web App có khả năng tạo ra các gateway thuộc về một Site bất kỳ. Một Site có thể có nhiều Gateway. Khi gateway được tạo sẽ sinh ra một API Key, và API Key sẽ được cung cấp cho gateway thực. Với API Key gateway thực có khả năng gửi dữ liệu lên server bằng giao thức MQTT hoặc HTTP request.

- Device: là một đối tượng ảo tượng trưng cho các thiết bị vật lý bên dưới. Device có thể là một Inverter, một Energy Meter, hay một Weather Station. Device cũng sẽ thuộc về một gateway cụ thể. Một gateway sẽ quản lý nhiều Device.

#### 4.4.1.2 Quy trình hoạt động của Web App

Quy trình hoạt động của Web App:

- Bước 1: Admin tiến hành tạo một Site.
- Bước 2: Admin tiến hành tạo gateway. Khi gateway được tạo sẽ có một id duy nhất và id này được sử dụng như một API key được cung cấp cho gateway thực lớp vật lý.
- Bước 3: Admin kích hoạt cho phép cấu hình gateway ( cho phép server nhận gói tin provision của gateway thực lớp vật lý). Lúc này trạng thái provision của gateway được chuyển từ trạng thái “ended” sang trạng thái “pending”.
- Bước 4: Gateway thực lớp vật lý tiến hành gửi gói tin provision chứa thông tin cấu hình gateway ( bao gồm các thông tin về các device thuộc gateway và các channel của từng device).
- Bước 5: Web App nhận được gói tin provision. Admin tiến hành tạo các device và mapping các channel của device.
- Bước 6: Gateway thực lớp vật lý tiến hành gửi các gói tin Telemetry chứa các dữ liệu thu thập được từ các device theo một chu kỳ nhất định.
- Bước 7: Admin kiểm tra việc kết nối dữ liệu qua giao diện Web App.
- Bước 8: Admin tiến hành tạo tài khoản user, và bộ nhiệm quyền truy cập vào Site cho các tài khoản tương ứng.

#### 4.4.2 Xây dựng ứng dụng back-end hỗ trợ tính năng User Authentication (đăng ký, đăng nhập) và Authorization (phân quyền người dùng).

Hầu hết các ứng dụng web hiện nay, tính năng xác thực và phân quyền người dùng đều phải có. Ví dụ, khi tạo một website, đương nhiên ta cần phải xây dựng tính năng đăng ký, đăng nhập, phân quyền admin, member, ... Có một số kỹ thuật giúp ta xây dựng tính năng này, ví dụ: dùng Sessions, hoặc mới hơn là JWT.

#### *4.4.2.1 Sự khác nhau giữa Authentication và Authorization*

Authentication là quá trình hệ thống kiểm tra, xác định danh tính của người dùng hoặc một hệ thống khác đang truy cập vào hệ thống hiện tại. Hiểu nôm na, quá trình Authentication đi tìm câu trả lời cho câu hỏi: “Bạn là ai?”. Quá trình Authentication rất thông dụng, hầu hết các ứng dụng liên quan tới quản lý nội dung, tương tác với người dùng đều có. Hiện nay, authentication xác thực chủ yếu dựa trên hai thông tin: tên người dùng và mật khẩu.

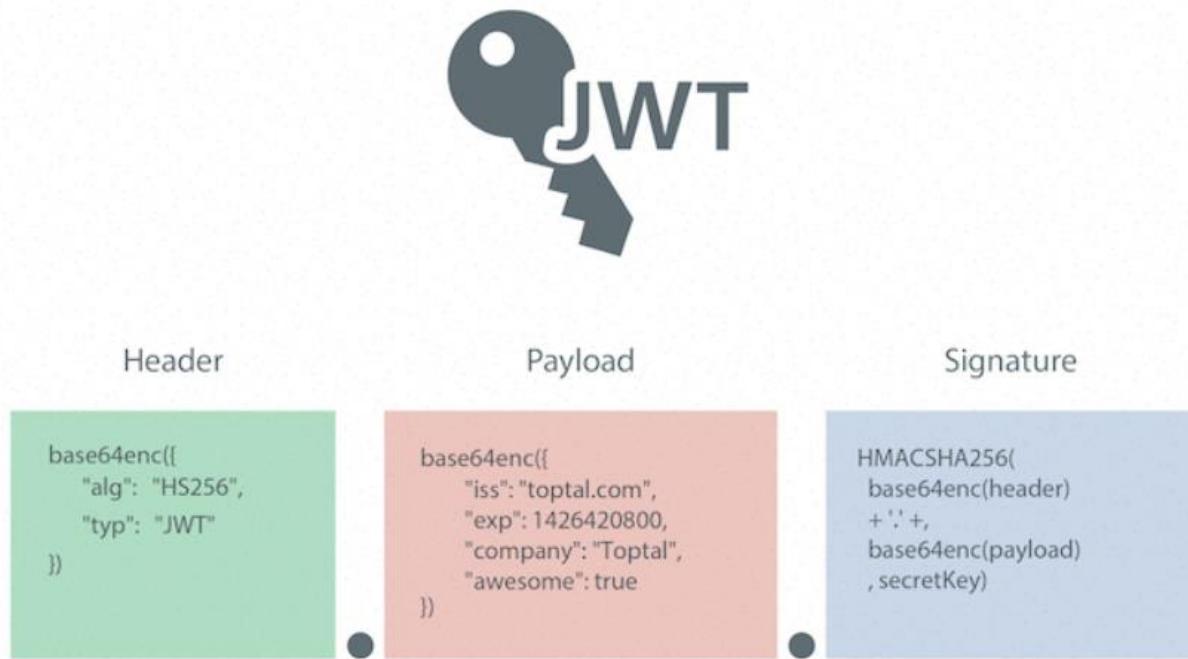
Tương tự, quá trình authorization để trả lời cho câu hỏi: “Bạn được phép làm gì?“. Về mặt kỹ thuật, quá trình authorization thường được thực hiện sau khi quá trình authentication kết thúc. Tức là, sau khi biết bạn là ai rồi thì bước tiếp theo xác định bạn được phép làm gì trong hệ thống.

#### *4.4.2.2 Token Based Authentication*

Token-based authentication là phương thức xác thực bằng chuỗi mã hóa. Một hệ thống sử dụng Token-based authentication cho phép người dùng nhập user/password (hoặc tương tự) để nhận về 1 chuỗi mã token. Mã này được sử dụng để "xác minh" quyền truy cập vào tài nguyên mà không cần phải cung cấp lại username/password nữa.

So với kỹ thuật xác thực dựa trên Session, bạn cần phải lưu Session vào Cookie. Lợi thế lớn nhất của Token-base authentication là lưu JSON Web Token (JWT) trên client như: Local Storage trên Browser, Keychain trong iOS app hay SharedPreferences trong ứng dụng Android, .v.v...

JWT là một phương tiện đại diện cho các yêu cầu chuyển giao giữa hai bên Client – Server, các thông tin trong chuỗi JWT được định dạng bằng JSON. Trong đó chuỗi Token phải có 2 phần là header, phần payload và phần signature được ngăn bằng dấu “.”.

**Hình 4-28 Cấu trúc chuỗi JWT**

JWT bao gồm 3 phần:

- Header: dùng để khai báo chữ ký và thuật toán mã hóa sẽ dùng cho token. Header là đối tượng JSON có định dạng như sau:

```
{
  "type" : "JWT",
  "alg" : "HS256"
}
```

Trong đoạn mã JSON, giá trị của khóa "typ" xác định rằng đối tượng là JWT và giá trị của khóa "alg" chỉ định thuật toán băm nào được sử dụng để tạo ra chữ ký (signature) cho JWT. Trong ví dụ của mình, thuật toán được sử dụng là HMAC-SHA256, thuật toán băm sử dụng secret key để tính toán chữ ký (signature).

- Payload: Phần thứ 2 của JWT là payload, nơi chứa các nội dung của thông tin . Thông tin truyền đi có thể là mô tả của một thực thể hoặc cũng có thể là các thông tin bổ sung thêm cho phần Header. Thành phần payload của JWT là dữ liệu được lưu trữ bên trong JWT (dữ

liệu này còn được gọi là "các xác nhận quyền sở hữu" của JWT), ví dụ: username, userid, author,...

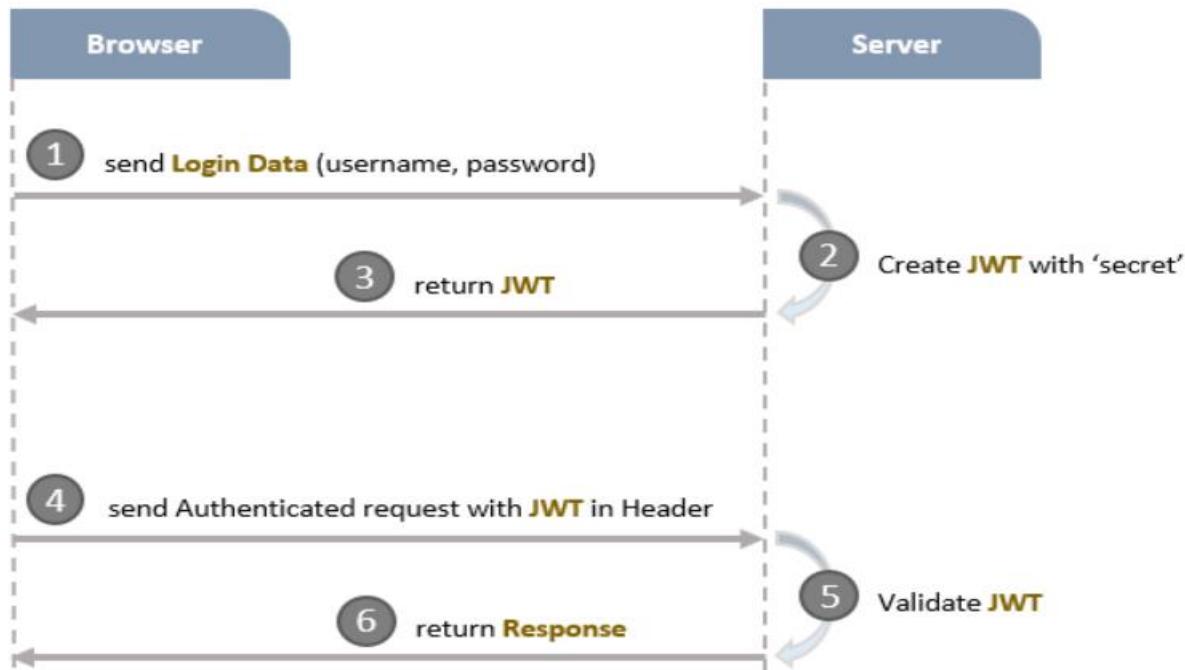
- Signature: Signature được tính bằng cách sử dụng thuật toán sau:

```
// signature algorithm
data = base64urlEncode( header ) + “.” + base64urlEncode( payload )
hashedData = hash( data, secret )
signature = base64urlEncode( hashedData )
```

**Hình 4-29 Thuật toán tạo ra signature trong chuỗi JWT**

Thuật toán này thực hiện bằng cách giải mã (header + payload). Thuật toán sau đó kết hợp các chuỗi được mã hóa kết quả cùng với dấu chấm(.) ở giữa chúng. Trong đoạn mã, chuỗi nội này được gán cho data. Chuỗi dữ liệu được băm bằng secret key sử dụng thuật toán băm được chỉ định trong tiêu đề JWT. Kết quả dữ liệu băm được gán cho hashedData. Dữ liệu băm này sau đó được mã hóa base64url để tạo ra chữ ký (signature) JWT.

Dưới đây là sơ đồ luồng hoạt động của JWT.



**Hình 4-30 Flow của authentication với JWT**

Nhìn vào sơ đồ, ta có thể thấy luồng đi như sau:

- User thực hiện login bằng cách gửi username và password lên phía server.
- Server tiếp nhận các dữ liệu mà User gửi lên để phục vụ cho việc xác thực người dùng. Trong trường hợp thành công, server sẽ tạo ra một JWT và trả về cho người dùng thông response.
- Người dùng nhập được JWT do server vừa mới trả về làm “chìa khoá” để thực hiện các request tiếp theo đối với server.
- Server trước khi thực hiện các yêu cầu được gọi từ phía User, sẽ verify JWT gửi lên. Nếu token hợp lệ, server sẽ thực hiện lệnh được yêu cầu.

#### *4.4.2.3 Tiến hành xây dựng ứng dụng Node.js với tính năng user authentication và authorization*

Sau khi tìm hiểu xong về lý thuyết, ta sẽ bắt đầu xây dựng một ứng dụng Node.js + Express với tính năng user authentication và authorization, trong đó:

- Người dùng có thể đăng ký tài khoản mới hoặc đăng nhập nếu đã có tài khoản
- Phân quyền tài khoản người dùng theo role. Với mỗi role, người dùng có quyền khác nhau để truy cập vào tài nguyên.

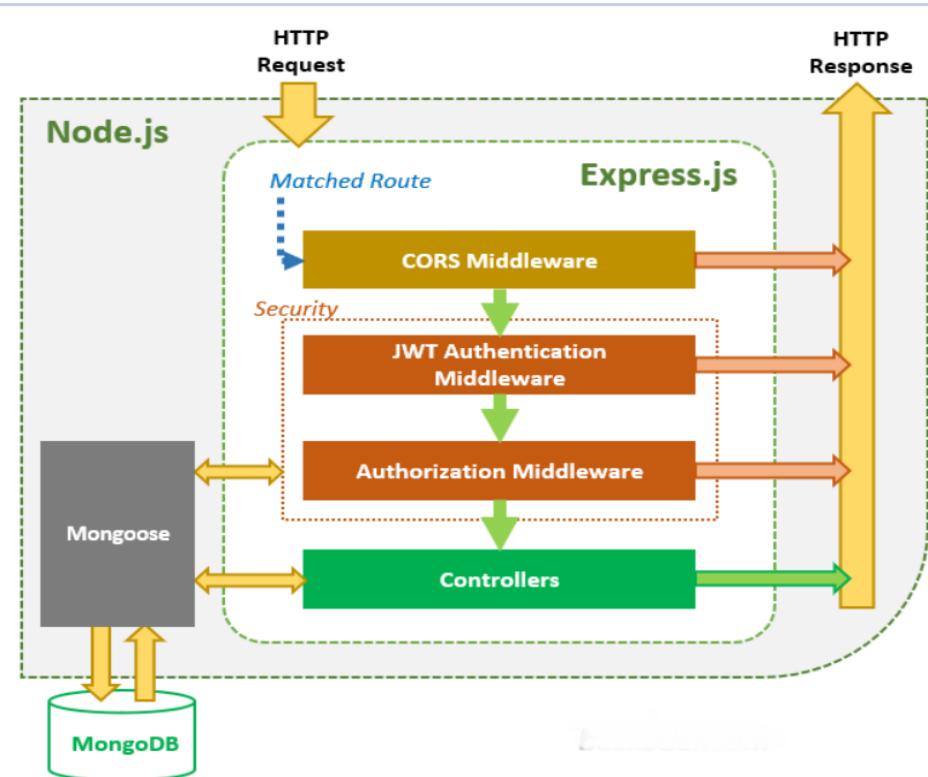
Dưới đây là danh sách những APIs cần thiết:

**Bảng 3 Danh sách những APIs cần thiết của web app**

Phương thức	APIs	Miêu tả công dụng
POST	/api/auth/signup	Đăng ký tài khoản mới
POST	/api/auth/signin	Đăng nhập
GET	/verifyToken	Kiểm tra token

GET	/get_user_infor	Được sử dụng bởi tài khoản admin, để lấy thông tin các user
POST	/delete_user	Được sử dụng bởi tài khoản admin, để xoá một user.
POST	/reset_password	Được sử dụng bởi tài khoản admin, cho phép thay đổi mật khẩu của user.
POST	/assign_site	Được sử dụng bởi tài khoản admin, cho phép admin bô nhiệm quyền truy cập một Site cho user
POST	/unassign_site	Được sử dụng bởi tài khoản admin, cho phép admin gỡ bỏ quyền truy cập một Site nào đó của user
POST	/update_password	Cho phép người dùng tự thay đổi mật khẩu.
POST	/reset_password_email	Cho phép người dùng reset lại mật khẩu qua email đã đăng ký trong trường hợp người dùng quên mật khẩu

Hình dưới đây mô tả tổng quan kiến trúc ứng dụng sử dụng Node.js + Express cho authentication & authorization.



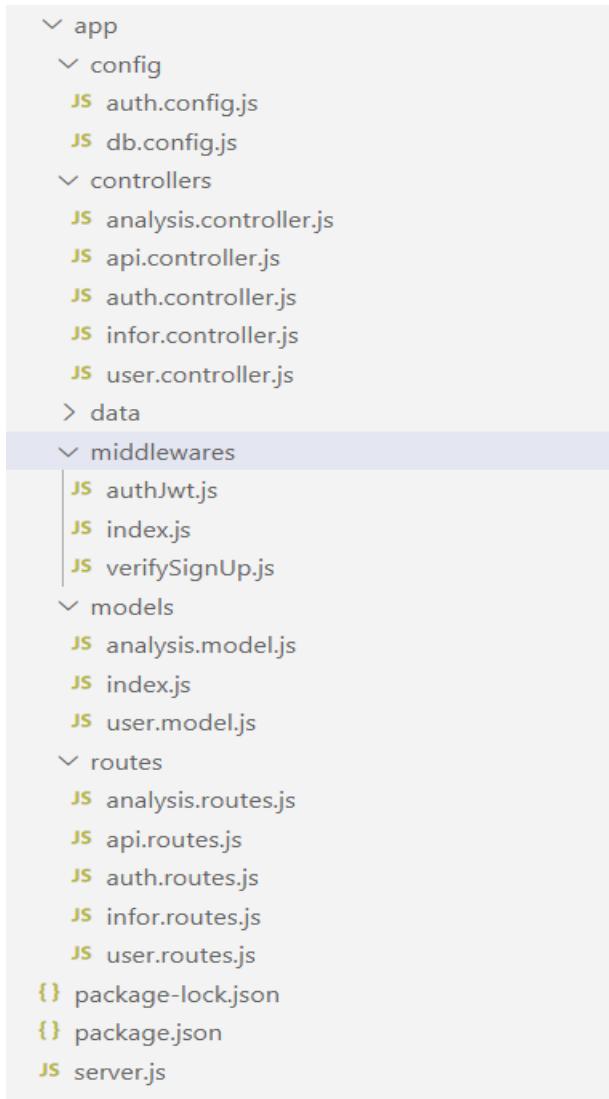
**Hình 4-31 Kiến trúc ứng dụng Node authentication với JWT**

Thông qua Express, các HTTP request hợp lệ và đúng với route đã thiết kế sẽ được kiểm tra bởi CORS Middleware trước khi vào Security layer.

Security layer bao gồm:

- JWT Authentication Middleware: có nhiệm vụ xác minh sign in, sign up và chuỗi token.
- Authorization Middleware: Kiểm tra role của người dùng đăng nhập với các thông tin được lưu trữ trong cơ sở dữ liệu.

Cấu trúc thư mục dự án:

**Hình 4-32 Cấu trúc thư mục ứng dụng**

Các bước tiến hành [15]:

❖ **Bước 1: Tạo dự án NodeJS**

Để bắt đầu, ta cần tạo mới dự án NodeJS. Khi tạo dự án mới xong, ta cần cài đặt thêm các thư viện cần thiết: express, cors, body-parser, mongoose, jsonwebtoken và bcryptjs.

Trong đó:

- Thư viện express: cung cấp các công cụ giúp tạo các APIs
- Thư viện body-parser: dùng để parse các request vào body object.

- Thư viện cors: cung cấp Express middleware dùng để bật tính năng CORS.
- Thư viện jsonwebtoken: hỗ trợ tạo ra các chuỗi JWT.
- Thư viện bcryptjs: giúp mã hoá mật khẩu người dùng.
- Thư viện mongoose: giúp ta làm việc với database MongoDB.

Nội dung package.json của dự án như sau:

```
① package.json > ...
1  {
2    "name": "authentication-authorization-jwt",
3    "version": "1.0.0",
4    "description": "create api for authentication and authorization users",
5    "main": "server.js",
6    "scripts": {
7      "start": "nodemon server.js",
8      "test": "echo \"Error: no test specified\" && exit 1"
9    },
10   "author": "Pham Thai Hoa",
11   "license": "ISC",
12   "dependencies": {
13     "axios": "^0.21.1",
14     "bcryptjs": "^2.4.3",
15     "body-parser": "^1.19.0",
16     "cors": "^2.8.5",
17     "crypto": "^1.0.1",
18     "express": "^4.17.1",
19     "jsonwebtoken": "^8.5.1",
20     "mongoose": "^5.12.1",
21     "nodemailer": "^6.5.0",
22     "nodemon": "^2.0.7",
23     "react-map-gl": "^6.1.16"
24   }
25 }
```

**Hình 4-33 Nội dung package.json của dự án**

#### ❖ Bước 2: Thiết lập Express web server

Trong thư mục gốc của dự án, ta tạo thêm tệp server.js có nội dung như sau:

*server.js*

```
const express = require("express");
```

```
const app = express();
app.use(function (req, res, next) {
  res.header("Access-Control-Allow-Origin", "*");
  res.header(
    "Access-Control-Allow-Headers",
    "Origin, X-Requested-With, Content-Type, Accept"
  );
  next();
});
app.use(express.json()); // for parsing application/json
app.use(express.urlencoded({ extended: true })); // for parsing application/x-www-form-urlencoded
const dbConfig = require("./app/config/db.config");
const db = require("./app/models");
// connect to database
db.mongoose
  .connect(`mongodb://${dbConfig.HOST}:${dbConfig.PORT}/${dbConfig.DB}`, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
  })
  .then(() => {
    console.log("Successfully connect to MongoDB.");
  })
  .catch((err) => {
    console.error("Connection error", err);
    process.exit();
  });
var path = require("path");

app.use("/", express.static("build"));
app.get("/dashboard*", function (req, res) {
  res.sendFile(path.join(__dirname, "build/index.html"), function (err) {
    if (err) {
      res.status(500).send(err);
    }
  });
});
app.get("/public*", function (req, res) {
  res.sendFile(path.join(__dirname, "build/index.html"), function (err) {
    if (err) {
      res.status(500).send(err);
    }
  });
});
require("./app/routes/auth.routes")(app);
```

```

require("./app/routes/user.routes")(app);
require("./app/routes/infor.routes")(app);
require("./app/routes/api.routes")(app);
require("./app/routes/analysis.routes")(app);
//set port, listen for requests
const PORT = process.env.PORT || 4000;
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}.`);
});

```

### ❖ Bước 3: Cấu hình kết nối MongoDB

Trong thư mục app, tạo riêng một thư mục mới, đặt tên là config. Thư mục này sẽ chứa tất cả các tệp liên quan tới cấu hình ứng dụng. Trong thư mục config, ta tạo tệp db.config.js để thêm các thông tin cài đặt cơ sở dữ liệu MongoDB cho ứng dụng:

*config/db.config.js*

```

module.exports ={
  HOST: "localhost",
  PORT: 27017,
  DB: "user_management"
}

```

### ❖ Bước 4: Định nghĩa Mongoose Model

Trong thư mục model, ta tiến hành tạo User model như sau:

*models/role.model.js*

```

const mongoose = require("mongoose");
const User = mongoose.model(
  "User",
  new mongoose.Schema({
    username: String,
    email: String,
    password: String,
    role: String,
    siteid: Array,
    resetToken: String,
    expireToken: Date,
  })
);
module.exports = User;

```

Mongoose Model này sẽ đại diện cho collection được tạo trong MongoDB. Khi bạn chạy chương trình, Mongoose sẽ tự động tạo ra collection có tên là: user.

#### ❖ **Bước 5: Cấu hình Auth Key**

Các hàm jsonwebtoken như: verify(), sign() sẽ cần tới một secret key để encode hay decode chuỗi token. Trong thư mục app/config, tạo thêm auth.config.js với nội dung sau:

*config/auth.config.js*

```
module.exports = {
  secret: "tell-me-your-secret"
};
```

#### ❖ **Bước 6: Tạo các hàm middleware**

Quá trình để verify một hành động trong SignUp, chúng ta cần làm 2 việc:

- Kiểm tra xem tên người dùng, email có bị trùng lặp trong DB hay không?
- Kiểm tra xem role đăng ký có hợp lại hay không?

Trong thư mục middleware, ta tạo một tệp verifySignUp.js có nội dung như sau:

*middlewares/verifySignUp.js*

```
const db = require("../models");
const User = db.user;
checkDuplicateUsernameOrEmail = (req, res, next) => {
  User.findOne({
    username: req.body.username,
  }).exec((err, user) => {
    if (err) {
      res.status(500).send({ message: err });
      return;
    }
    if (user) {
      res.status(400).send({ message: "Failed! Username is already in use!" });
      return;
    }
    User.findOne({
      email: req.body.email,
```

```

}).exec((err, user) => {
  if (err) {
    res.status(500).send({ message: err });
    return;
  }
  if (user) {
    res.status(400).send({ message: "Failed! Email is already in use!" });
    return;
  }
  next();
});
});
};

checkRolesExisted = (req, res, next) => {
  var role = req.body.role;
  if (role) {
    if (role != "admin" && role != "moderator" && role != "user") {
      res.status(400).send({ message: "Failed! Role does not exist!" });
      return;
    }
  }
  next();
};

const verifySignUp = {
  checkDuplicateUsernameOrEmail,
  checkRolesExisted,
};

module.exports = verifySignUp;

```

Để xử lý việc Authentication & Authorization, chúng ta cần tạo các hàm sau:

- Kiểm tra token có hợp lệ hay không? Chúng ta có thể lấy thông tin token trong trường x-access-token của Header HTTP, sau đó chuyển cho hàm verify() xử lý.
- Kiểm tra role đăng ký đã có role chưa hay là trống?

Trong thư mục middleware, ta tạo tệp authJwt.js với nội dung như sau:

### *middlewares/authJwt.js*

```

const jwt = require("jsonwebtoken");
const config = require("../config/auth.config");
const db = require("../models");

```

```

const User = db.user;
verifyToken = (req, res, next) => {
  let token = req.body.token || req.query.token;
  if (!token) {
    return res.status(403).send({ message: "No token provided!" });
  }
  jwt.verify(token, config.secret, (err, decoded) => {
    if (err) {
      return res.status(401).send({ message: "Unauthorized!" });
    }
    req.userId = decoded.id;
    next();
  });
};
isAdmin = (req, res, next) => {
  User.findById(req.userId).exec((err, user) => {
    if (err) {
      res.status(500).send({ message: err });
      return;
    }
    if (user) {
      if (user.role === "admin") {
        next();
        return;
      }
    }
    res.status(403).send({ message: "Required Admin Role!" });
    return;
  });
};
const authJwt = {
  verifyToken,
  isAdmin,
};
module.exports = authJwt;

```

### ❖ Bước 7: Tạo Controllers

Với phần authentication, chúng ta có 2 công việc chính cho tính năng authentication:

- Đăng ký: tạo người dùng mới và lưu trong cơ sở dữ liệu (với role mặc định là User nếu không chỉ định trước lúc đăng ký).

- Đăng nhập: Tìm username trong cơ sở dữ liệu. Nếu username tồn tại, so sánh password với password trong CSDL sử dụng. Nếu pass khớp, tạo token bằng jsonwebtoken rồi trả về client với thông tin user kèm theo access-token.

Trong thư mục controllers, ta tạo tệp authen.controller.js với nội dung như sau:

*controllers/auth.controller.js*

```
const config = require("../config/auth.config");
const db = require("../models");
const User = db.user;

var jwt = require("jsonwebtoken");
var bcrypt = require("bcryptjs");

exports.signup = (req, res) => {
  const user = new User({
    username: req.body.username,
    email: req.body.email,
    password: bcrypt.hashSync(req.body.password, 8),
    role: req.body.role ? req.body.role : "user",
    siteid: req.body.siteid,
  });
  user.save((err) => {
    if (err) {
      res.status(500).send({ message: err });
      return;
    } else {
      res.send({ message: "User was registered successfully!" });
    }
  });
};

exports.signin = (req, res) => {
  User.findOne({
    username: req.body.username,
  }).exec((err, user) => {
    if (err) {
      res.status(500).send({ message: err });
      return;
    }
    if (!user) {
      return res.status(404).send({ message: "User not found" });
    }
    var passwordIsValid = bcrypt.compareSync(req.body.password, user.password);
    if (!passwordIsValid) {
      return res.status(401).send({ message: "Invalid Password!" });
    }
    var token = jwt.sign({ id: user.id, user: user }, config.secret);
    res.json({ token: token, user: user });
  });
};
```

```

if (!passwordIsValid) {
  return res.status(401).send({
    accessToken: null,
    message: "Invalid Password!",
  });
}
var token = jwt.sign({ id: user.id }, config.secret, {
  expiresIn: 86400, //1 ngày
});
jwt.verify(token, config.secret, (err, decoded) => {
  if (err) {
    return res.status(401).send({ message: "Unauthorized!" });
  }
  res.status(200).send({
    user: {
      id: user._id,
      username: user.username,
      email: user.email,
      role: user.role,
      siteid: user.siteid,
    },
    accessToken: token,
    time_start: decoded.iat,
    time_expire: decoded.exp,
  });
});
});
});
};


```

Với phần authorization, ta có 5 APIs chỉ dành cho tài khoản admin và 2 APIs dành cho tất cả tài khoản:

- /get\_user\_infor: cho phép admin lấy tất cả các thông tin của user.
- /delete\_user: cho phép admin xoá tài khoản user.
- /reset\_password: cho phép admin reset mật khẩu của tài khoản user.
- /assgin\_site: cho phép admin bổ nhiệm quyền truy cập một Site cho user.
- /unassign\_site: cho phép admin gỡ bỏ quyền truy cập một Site của một user bất kỳ.
- /update\_password: cho phép tất cả các tài khoản có thể thay đổi mật khẩu.
- /reset\_password\_email: cho phép tất cả các tài khoản có thể reset lại mật khẩu thông qua email đã đăng ký trong trường hợp quên mật khẩu.

Trong thư mục controllers, ta tạo tệp user.controller.js để xử lý cho các APIs trên với nội dung sau:

***controllers/user.controller.js***

```
exports.userInfor = (req, res) => {
  User.find().exec((err, users) => {
    if (err) {
      res.status(500).send({ message: err });
      return;
    } else {
      var userList = [];
      users.forEach(function (user) {
        if (user.role != "admin") {
          userList.push({
            id: user._id,
            username: user.username,
            email: user.email,
            role: user.role,
            siteid: user.siteid,
          });
        }
      });
      return res.send(userList);
    }
  });
};

exports.resetPasswordAdmin = (req, res) => {
  var user_id = req.body.userid || req.query.userid;
  User.findById(user_id).exec((err, userObj) => {
    if (err) {
      res.status(500).send({ message: err });
      return;
    } else {
      userObj.password = bcrypt.hashSync(req.body.newpassword, 8);
      userObj.save();
      return res.send({ message: "Reset password successfully !" });
    }
  });
};

exports.deleteUser = (req, res) => {
  var user_id = req.body.userid || req.query.userid;
  User.findById(user_id).exec((err, userObj) => {
    if (err) {
```

```
res.status(500).send({ message: err });
return;
} else {
  userObj.remove();
  res.send({ message: "user has been deleted" });
}
});
};

exports.assignSite = (req, res) => {
  var user_id = req.body.userid || req.query.userid;
  var site_id = req.body.siteid || req.query.siteid;
  User.findById(req.userId).exec((err, userObj) => {
    if (err) {
      res.status(500).send({ message: err });
      return;
    } else {
      var sign = 0;
      userObj.siteid.map((item, index) => {
        if (item == site_id) {
          sign = 1;
        }
      });
      if (sign != 1) {
        userObj.siteid.push(site_id);
        userObj.save();
      }
    }
  });
  if (user_id != null) {
    User.findById(user_id).exec((err, userObj) => {
      if (err) {
        res.status(500).send({ message: err });
        return;
      } else {
        var sign = 0;
        userObj.siteid.map((item, index) => {
          if (item == site_id) {
            sign = 1;
          }
        });
        if (sign != 1) {
          userObj.siteid.push(site_id);
          userObj.save();
          res.send({
            message: "site has been assigned to user!",
          });
        }
      }
    });
  }
};
```

```
        already: false,
    });
} else {
    res.send({
        message: "Site has been assigned to this account before",
        already: true,
    });
}
});
} else {
    res.send({ message: "site has been assigned to Admin!", already: true });
}
};

exports.unassignSite = (req, res) => {
    User.findById(req.userId).exec((err, userObj) => {
        if (err) {
            res.status(500).send({ message: err });
            return;
        } else {
            if (userObj.siteid.indexOf(req.body.siteid) != -1) {
                userObj.siteid.splice(userObj.siteid.indexOf(req.body.siteid), 1);
                userObj.save();
            }
        }
    });
    let account = req.body.account;
    if (account.length) {
        account.map((item, index) => {
            User.findOne({
                username: item,
            }).then((userObj) => {
                if (userObj) {
                    if (userObj.siteid.indexOf(req.body.siteid) != -1) {
                        userObj.siteid.splice(userObj.siteid.indexOf(req.body.siteid), 1);
                        userObj.save();
                    }
                }
            });
        });
        res.send({ message: "done" });
    }
};
};
```

❖ **Bước 8: Định nghĩa các Routes**

Khi một client gửi request tới web server sử dụng HTTP (GET, POST, PUT, DELETE) , chúng ta cần định nghĩa, xác định cách server tiếp nhận và phản hồi như thế nào. Đây chính là công dụng của các route. Chia các routes thành 2 nhóm: Authentication và Authorization

Trong thư mục routes, ta tạo tệp auth.routes.js với nội dung như sau:

#### *routes/auth.routes.js*

```
const { verifySignUp } = require("../middlewares");
const controller = require("../controllers/auth.controller");
module.exports = function (app) {
    app.use(function (req, res, next) {
        res.header(
            "Access-Control-Allow-Headers",
            "x-access-token, Origin, Content-Type, Accept"
        );
        next();
    });
    app.post(
        "/api/auth/signup",
        [
            verifySignUp.checkDuplicateUsernameOrEmail,
            verifySignUp.checkRolesExisted
        ],
        controller.signup
    );
    app.post("/api/auth/signin", controller.signin);
};
```

Trong thư mục routes, ta tạo tệp user.routes.js với nội dung như sau:

#### *routes/user.routes.js*

```
const { authJwt } = require("../middlewares");
const controller = require("../controllers/user.controller");
module.exports = function (app) {
    app.use(function (req, res, next) {
        res.header(
            "Access-Control-Allow-Headers",
            "x-access-token, Origin, Content-Type, Accept"
        );
        next();
    });
    app.get("/verifyToken", [authJwt.verifyToken, authJwt.isOnline]);
```

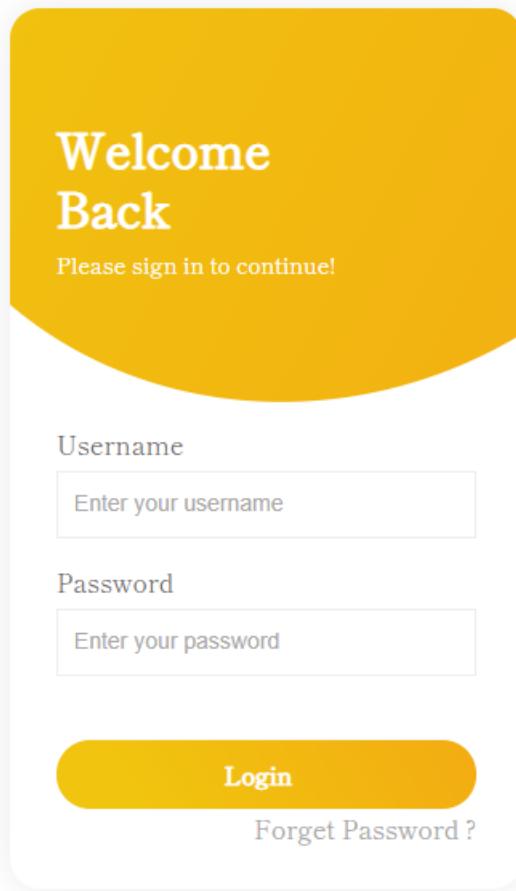
```
app.get(
  "/get_user_infor",
  [authJwt.verifyToken, authJwt.isAdmin],
  controller.userInfor
);
app.post(
  "/delete_user",
  [authJwt.verifyToken, authJwt.isAdmin],
  controller.deleteUser
);
app.post(
  "/assign_site",
  [authJwt.verifyToken, authJwt.isAdmin],
  controller.assignSite
);
app.post(
  "/unassign_site",
  [authJwt.verifyToken, authJwt.isAdmin],
  controller.unassignSite
);
app.post(
  "/update_password",
  [authJwt.verifyToken],
  controller.changePassword
);
app.post(
  "/reset_password",
  [authJwt.verifyToken, authJwt.isAdmin],
  controller.resetPasswordAdmin
);
app.post("/forget_password", controller.forgetPassword);
app.post("/reset_password_email", controller.resetPasswordEmail);
};
```

#### 4.4.3 Xây dựng giao diện Web App

##### 4.4.3.1 Xây dựng giao diện trang Login

Để có thể truy cập hệ thống người dùng cần phải tiến hành đăng nhập bằng cách sử dụng username và password đã đăng ký. Khi người dùng nhập username và password, server sẽ tiến hành kiểm tra username trong database, nếu tồn tại sẽ tiếp tục kiểm tra password có trùng khớp hay không. Nếu trùng khớp sẽ tự động chuyển đến trang giao diện ứng dụng. Trong trường hợp username không tồn tại hoặc password không trùng khớp, ứng dụng sẽ hiển thị thông báo cho người dùng.

Hình dưới đây là giao diện đăng nhập của ứng dụng:

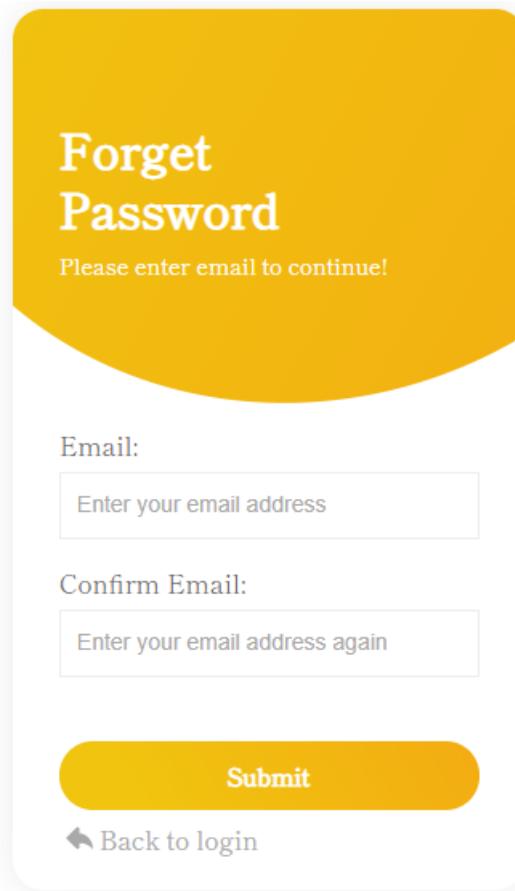


**Hình 4-34 Giao diện trang đăng nhập**

Trong trường hợp người dùng quên mật khẩu, người có thể reset lại mật khẩu thông qua email đã đăng ký bằng cách thực hiện các bước sau:

- Từ giao diện trang đăng nhập, người dùng click vào dòng chữ “Forget Password”.
- Trang web sẽ được chuyển đến giao diện mới, sau đó người dùng tiến hành nhập email đã đăng ký.
- Server sẽ tự động gửi vào email đã đăng ký một đường link, người dùng truy cập vào đường link này và tiến hành thay đổi mật khẩu.

Giao diện cho phép người dùng lấy lại mật khẩu thông qua email đã đăng ký:

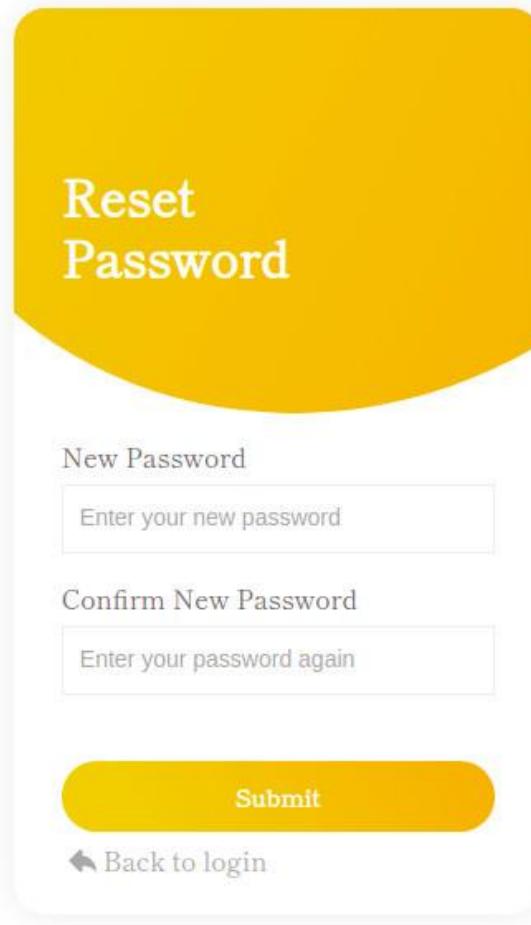


**Hình 4-35 Giao diện cho phép người dùng lấy lại mật khẩu thông qua email**

Email phản hồi từ yêu cầu reset password của người dùng:

A screenshot of an email inbox. On the left, there is a search bar and several icons. A single email message is visible, with the subject 'password reset' and the recipient 'hoapham140899@gmail.com'. The message was sent '2 minutes ago'. To the right of the inbox is the email content viewer. The subject is 'password reset'. The 'From' field is 'Example Team <from@example.com>' and the 'To' field is 'hoapham140899@gmail.com'. There is a 'Show Headers' link. Below the headers, there are tabs for 'HTML', 'HTML Source', 'Text', 'Raw', 'Spam Analysis', 'HTML Check', and 'Tech Info'. The 'HTML' tab is selected. The email body contains the text 'You requested for password reset' and 'click in this [link](#) to reset password'. At the bottom right of the email viewer, there are three small icons: a square, a circle, and a monitor.

Giao diện cho phép người dùng đặt lại mật khẩu mới:



**Hình 4-36 Giao diện cho phép người dùng đặt lại mật khẩu mới**

#### 4.4.3.2 Giao diện trang Site Management

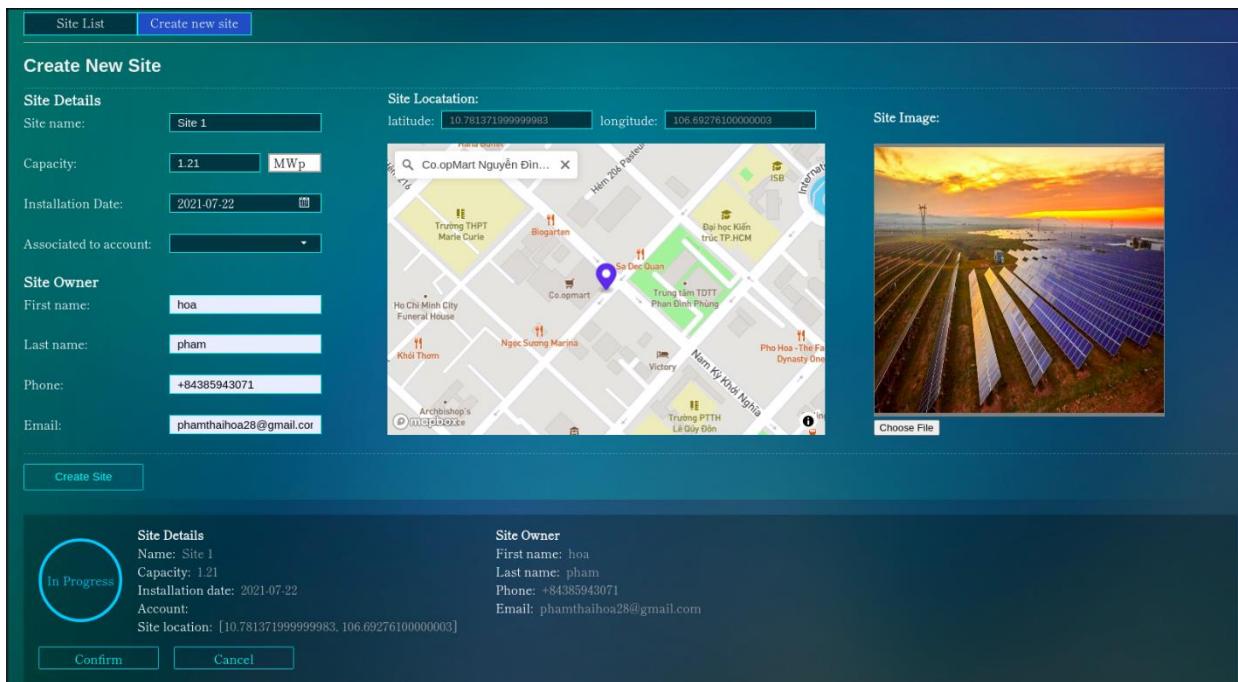
Khái niệm “Site”: Site ở đây được hiểu là một đối tượng ảo được tạo ra trên server đại diện cho một trạm năng lượng mặt trời. Trang Site Management chỉ được truy cập bởi tài khoản admin với chức năng quản lý các Site.

Các chức năng của trang Site Management:

- Tạo ra một Site mới với các thông tin về tên Site, Capacity, thông tin chủ sở hữu, tọa độ của Site và hình ảnh đại diện của site.
- Quản lý các Site đã tạo:
  - Xem thông tin chi tiết của Site
  - Xoá một Site bất kỳ

- Bổ nhiệm quyền truy cập Site cho một tài khoản user bất kỳ.
- Tạo các gateway ảo thuộc một Site bất kỳ và cung cấp API key cho gateway thực lớp vật lý
- Quản lý thông tin các gateway của từng Site
- Cho phép lấy thông tin cấu hình gateway được gửi lên từ gateway thực trong gói tin provision
- Cho phép tạo các device và tiến hành mapping các channel của từng device
- Quản lý thông tin các device của từng gateway

Giao diện tạo một Site mới:



**Hình 4-37 Giao diện tạo một Site mới**

Để tạo một Site mới ta thực hiện các bước sau:

- Truy cập vào trang Site Management
- Click vào tab Create new site
- Tiến hành điền đầy đủ các thông tin và bấm vào nút create site
- Kiểm tra lại thông tin của site và bấm vào nút Confirm để hoàn thành việc tạo site

Giao diện để quản lý các Site đã tạo:

The screenshot shows a web-based application for managing sites. At the top, there are tabs for 'Site List' and 'Create new site'. Below this is a search bar with placeholder text 'Search for names...' and a magnifying glass icon. A table lists two sites: 'Site 1' and 'Site 2'. Each row in the table includes columns for 'Name', 'Capacity (kWp)', 'Installation Date', and 'Action' (with an 'Edit' button). Below the table, a section titled 'Site Infor' displays detailed information for 'Site 1', including 'Site Details' (Name: Site 1, Capacity: 1.19, Installation date: 2021-07-16, Account: [redacted]) and 'Site Owner' (First name: hoa, Last name: pham, Phone: +84385943071, Email: phamthaihoa28@gmail.com). At the bottom of this section are five buttons: 'Add Gateway', 'Assign Account', 'Gateway Infor', 'Delete', and 'Cancel'.

**Hình 4-38 Giao diện quản lý các Site**

Để có thể xem thông tin các site đã tạo và tiến hành cài đặt cho các site ta thực hiện các bước sau:

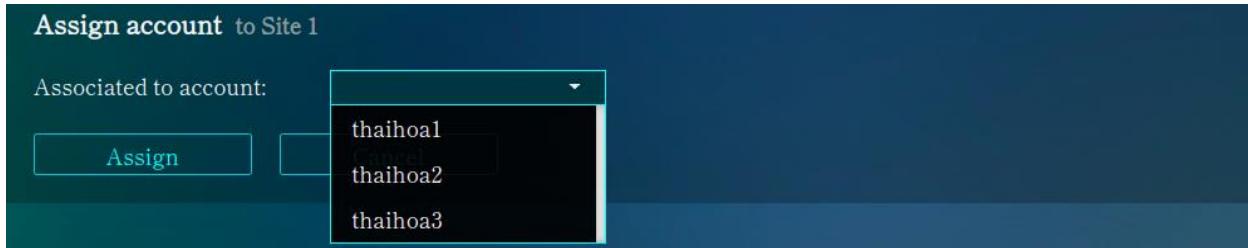
- Truy cập vào trang Site Management
- Click vào tab Site List
- Để có thể xem thông tin chi tiết của site và thực hiện các cấu hình cho site ta bấm vào nút Edit ứng với từng Site
- Ta có thể thực hiện các cấu hình cho site: tạo gateway mới thuộc site, xem thông tin của các gateway đã tạo, bổ nhiệm quyền truy cập site cho một tài khoản user, và xóa một site bất kỳ.

Giao diện để tạo một gateway thuộc một site bất kỳ:

The screenshot shows a modal dialog box titled 'Add a gateway to Site 1'. It has a text input field labeled 'Gateway name:' followed by a redacted text box. At the bottom of the dialog are two buttons: 'Create Gateway' and 'Cancel'.

**Hình 4-39 Giao diện tạo một gateway**

Giao diện để bổ nhiệm quyền truy cập site cho một tài khoản user:



**Hình 4-40 Giao diện bô nhiệm quyền truy cập site cho tài khoản user**

Giao diện quản lý các gateway thuộc một Site bất kỳ:

Gateway infor of Site 1				
Name	Id	Kind	Provision	Action
Gateway 1	60f971c4c345a00ecb57bc70	Gateway	ended	<button>Provision</button> <button>Mapping</button> <button>Device</button> <button>Delete</button>

**Hình 4-41 Giao diện quản lý các gateway thuộc một site**

Giao diện quản lý thông tin cấu hình của gateway:

Provision Data of Gateway 1				
Device Name	Device Id	Device Kind	Action	
inverter01	01	inverter	<button>Add Device</button>	
inverter02	02	inverter	<button>Add Device</button>	
inverter03	03	inverter	<button>Add Device</button>	
WeatherStation1	04	WeatherStation	<button>Add Device</button>	
EnergyMeter1	05	EnergyMeter	<button>Add Device</button>	
EnergyMeter2	06	EnergyMeter	<button>Add Device</button>	

**Hình 4-42 Giao diện quản lý thông tin cấu hình của gateway**

Giao diện để mapping channel và tạo device:

The screenshot shows the 'Add Device' interface. On the left, there's a 'Device Name' field set to 'EnergyMeter2' and a 'kind' dropdown set to 'EnergyMeter'. Below these are two input fields: 'key' and 'value', followed by an 'Ok' button. To the right, under 'Device Infor', is a list of mappings:

- Device Name: EnergyMeter2
- Device Kind: EnergyMeter
- Alias:
  - Active\_Generated\_Energy: ActiveGeneratedEnergy
  - Active\_Consumed\_Energy: ActiveConsumedEnergy
  - Reactive\_Generated\_Energy: ReactiveGeneratedEnergy
  - Reactive\_Consumed\_Energy: ReactiveConsumedEnergy
  - Energy\_Meter\_Production: EnergyMeterProduction
  - Active\_Power: ActivePower
  - Reactive\_Power: ReactivePower

Below this, there are 'Create Device' and 'Cancel' buttons.

**Hình 4-43 Giao diện mapping channel và tạo device**

Giao diện quản lý các device đã tạo:

The screenshot shows the 'Device List of Gateway 1' interface. It displays a table with columns: Device Name, Device Id, Device Kind, and Action (Delete). The data is as follows:

Device Name	Device Id	Device Kind	Action
inverter01	01	Inverter	Delete
inverter02	02	Inverter	Delete
inverter03	03	Inverter	Delete
EnergyMeter1	05	EnergyMeter	Delete
WeatherStation1	04	WeatherStation	Delete
EnergyMeter2	06	EnergyMeter	Delete

**Hình 4-44 Giao diện quản lý các device đã tạo**

#### 4.4.3.3 Giao diện trang Account Management

Trang Account Management chỉ được truy cập bởi tài khoản admin, cho phép Admin tạo mới các tài khoản user và quản lý các tài khoản đó.

Các chức năng của trang Account Management:

- Tạo mới một tài khoản với role là user
- Quản lý các tài khoản user đã tạo:
  - Xem thông tin chi tiết tài khoản
  - Xoá một tài khoản bất kỳ

- Reset password cho một tài khoản bất kỳ

Giao diện tạo mới một tài khoản user:

User List    Create new user

**Create new user**

Username: [REDACTED]

Email: [REDACTED]

Password: [REDACTED]

Confirm password: [REDACTED]

**Create**

**Hình 4-45 Giao diện tạo mới một tài khoản user**

Giao diện quản lý các tài khoản user:

User List    Create new user

ID	Username	Email	Role	Action
60e457e16093d9153a0673f9	thaihoa1	hoapham140899@gmail.com	user	Action
60f975a91a1d8c1107ffc11d	thaihoa2	phamthaihoa2@gmail.com	user	Action
60f975c31a1d8c1107ffc11e	thaihoa3	phamthaihoa3@gmail.com	user	Action

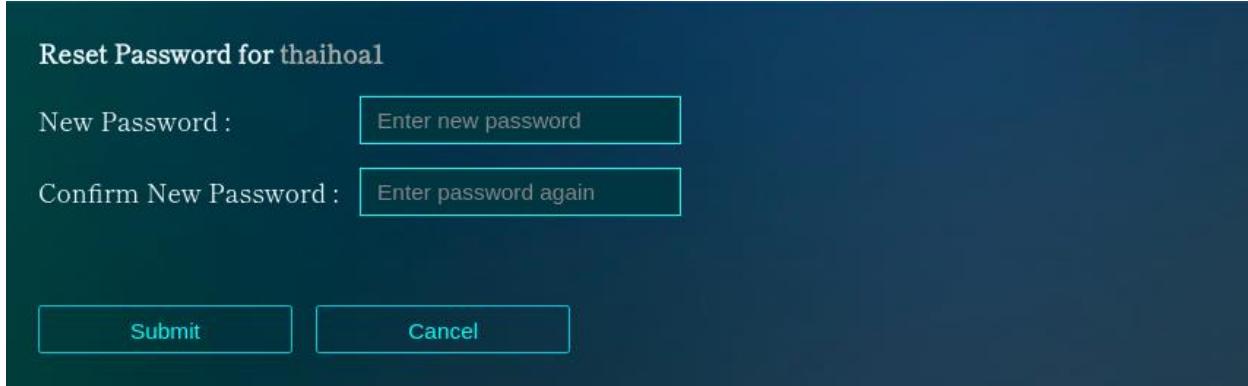
**User Infor**

ID: 60e457e16093d9153a0673f9  
 Username : thaihoa1  
 Email : hoapham140899@gmail.com  
 Role : user

**Reset Password**   **Delete**   **Cancel**

**Hình 4-46 Giao diện quản lý các tài khoản user**

Giao diện reset password cho tài khoản user:



**Hình 4-47 Giao diện reset password cho tài khoản user**

#### 4.4.3.4 Giao diện trang Datatype Editor

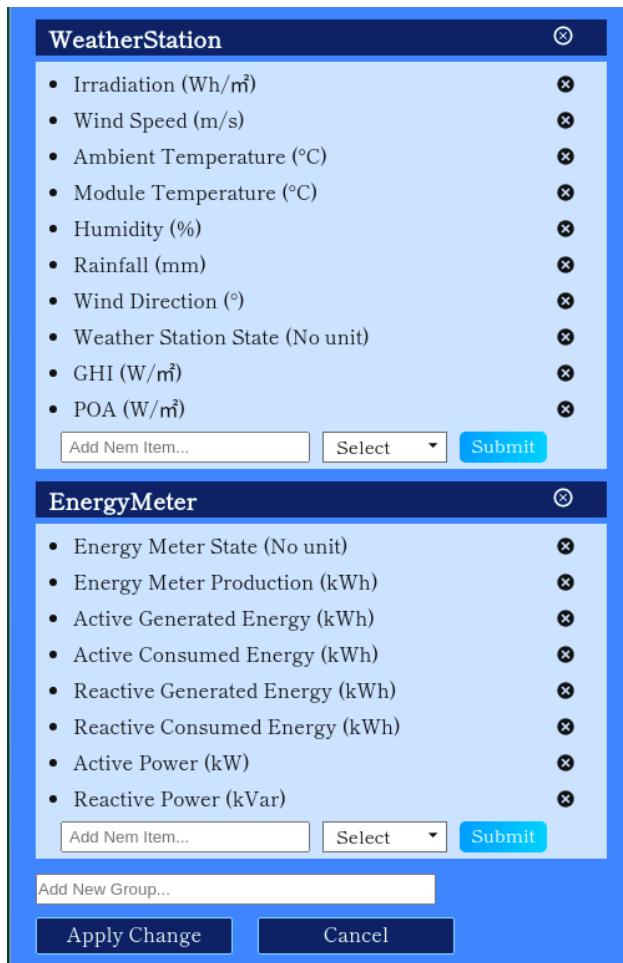
Trang Datatype Editor chỉ được truy cập bởi tài khoản Admin, cho phép admin tạo thêm các loại device và cấu hình các channel cho từng loại device.

Thông tin device và các channel của các device hiện có:

- Inverter:
  - Inverter Operation State
  - Inverter Production
  - Inverter Active Power
  - Inverter Reactive Power
  - Inverter Input Power
  - Inverter Efficiency
  - Inverter Mains Frequency
  - Inverter Internal Temperature
  - Inverter Power Factor
  - Inverter R Phase Current
  - Inverter S Phase Current
  - Inverter T Phase Current
  - Inverter R Phase Voltage
  - Inverter S Phase Voltage
  - Inverter T Phase Voltage

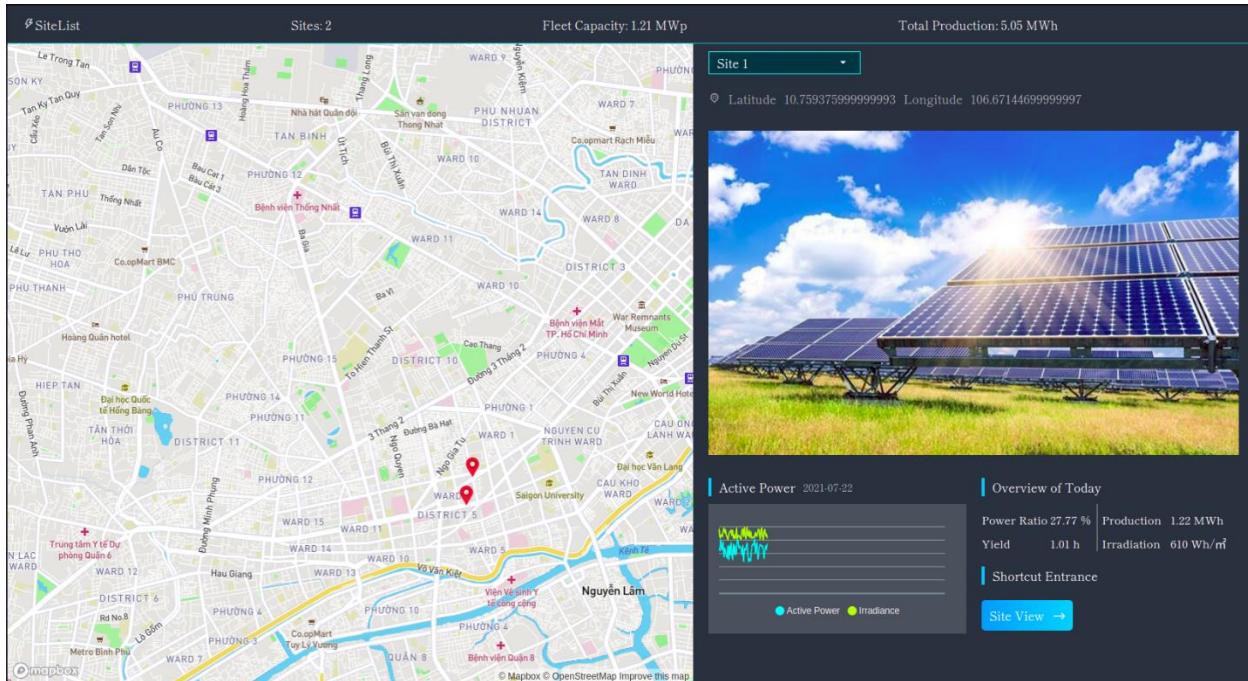
- Inverter Line Voltage L1L2
  - Inverter Line Voltage L2L3
  - Inverter Line Voltage L3L1
  - Inverter String Current
- Energy Meter:
- Active Generated Energy
  - Active Consumed Energy
  - Reactive Generated Energy
  - Reactive Consumed Energy
  - Energy Meter Production
  - Active Power
  - Reactive Power
- Weather Station:
- Irradiation
  - POA
  - GHI
  - Ambient Temperature
  - Module Temperature
  - Wind Speed
  - Wind Direction
  - Humidity
  - Rainfall

Giao diện của trang Datatype Editor:

**Hình 4-48 Giao diện trang Datatype Editor**

#### 4.4.3.5 Giao diện trang Fleetview

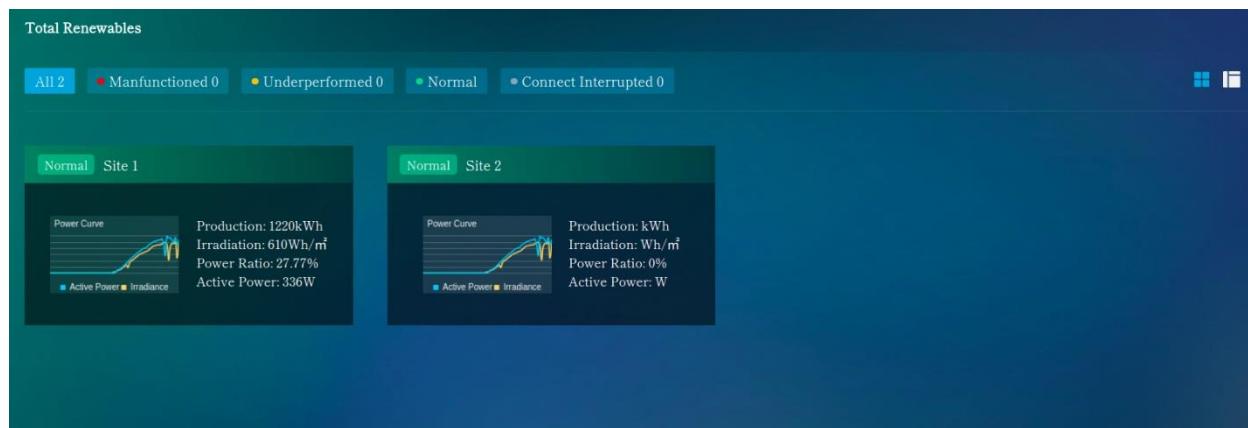
Fleetview là trang có thể được truy cập bởi cả tài khoản admin và user. Trang hiển thị vị trí tọa độ của các Site trên bản đồ thế giới. Bên cạnh đó, trang còn hiển thị một số thông tin tổng quan trong ngày của trạm năng lượng mặt trời như, cụ thể là các thông tin về : production, power ratio, yield, và irradiation.



**Hình 4-49 Giao diện trang Fleetview**

#### 4.4.3.6 Giao diện trang Site List

Site List là trang có thể truy cập bởi cả tài khoản admin và user. Trang sẽ hiển thông tin chi tiết tất cả các Site được bổ nhiệm với tài khoản dùng đăng nhập. Với tài khoản admin, mặc định sẽ hiển thị tất cả các Site đã được tạo. Trang sẽ có hai chế độ hiển thị: chế độ hiển thị dạng đối tượng và chế độ hiển thị dạng danh sách.



**Hình 4-50 Giao diện trang Site List chế độ đối tượng**

Total Renewables						
All 2	Manfunctioned 0	Underperformed 0	Normal	Connect Interrupted 0		
Search for names.. <input type="text"/>						<input type="button" value="Export"/>
Name	Operation State	Connection State	Production Today (kWh) ↑↑	Active Power (kW) ↑↑	Irradiance (W/m²) ↑↑	Irradiation (Wh/m²) ↑↑
Site 1	Normal	--	1220	336	418	610
Site 2	Normal	--	--	--	--	--

**Hình 4-51 Giao diện trang Site List chế độ danh sách**

#### 4.4.3.7 Giao diện trang Leaderboard

Leaderboard là trang có thể được truy cập bởi cả tài khoản admin và user. Các chức năng của trang bao gồm:

- Hiển thị thông tin về sản lượng điện lý thuyết (Budget Production) và sản lượng điện thực tế (Actual Production) của từng Site dưới dạng biểu đồ so sánh.
- Tính toán phần trăm mức độ hoàn thành sản lượng điện (Production Budget Completion Rate) của từng trạm điện. Cho phép xem thông tin chi tiết từng tháng bằng cách click vào “Details”.
- Hiển thị các thông tin về hiệu năng của trạm điện (bao gồm thông tin về Yield và PR) dưới dạng biểu đồ so sánh. Cho phép chọn các chế độ hiển thị theo ngày, tháng, hoặc năm.

Giao diện hiển thị của trang Leaderboard:



**Hình 4-52 Giao diện hiển thị trang Leaderboard**

Giao diện xem thông tin chi tiết Completion Rate:



**Hình 4-53 Giao diện xem thông tin chi tiết Production Completion Rate**

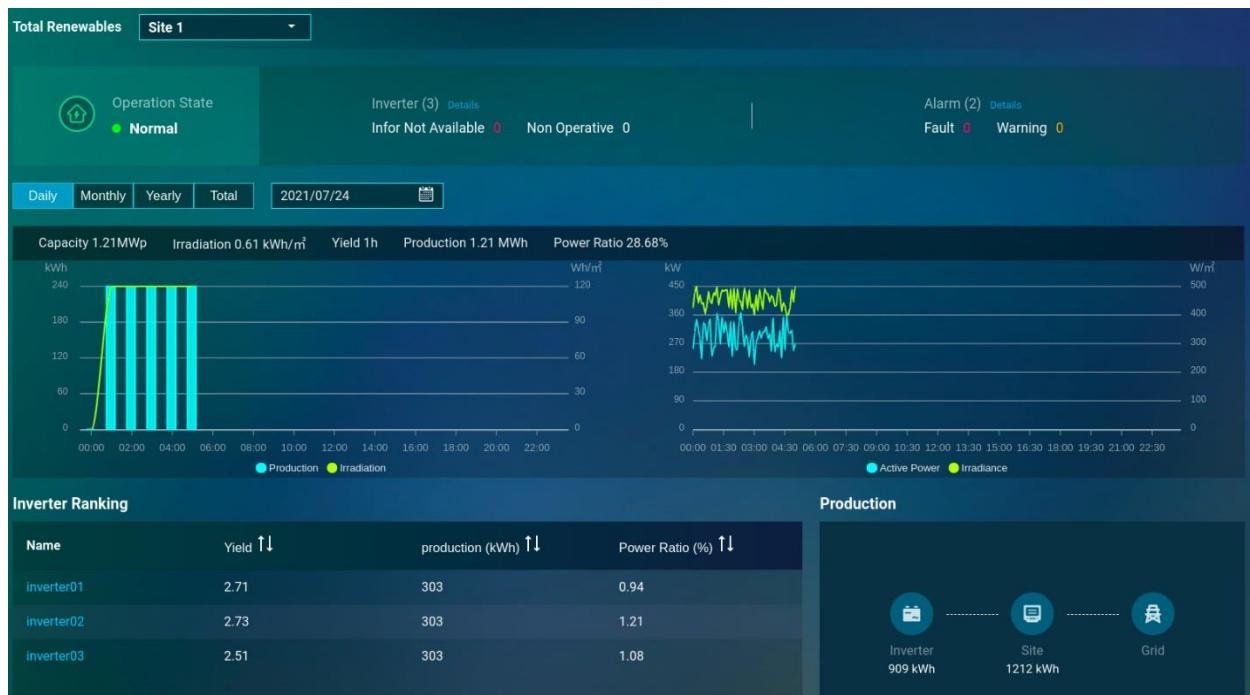
#### 4.4.3.8 Giao diện trang Site View

Site View là trang có thể được truy cập bởi cả tài khoản admin và user. Trang Site View cung cấp hình ảnh tóm tắt về hoạt động của một trạm điện, có thể cung cấp biểu đồ phân tích xu hướng, so sánh tổng thất điện năng tiêu thụ và giúp nhân viên giám sát và bảo trì nhanh chóng phát hiện và phân tích các sự cố của trạm điện và các thiết bị. Cho phép người dùng chọn các chế độ hiển thị thống kê theo ngày, tháng hoặc năm.

Các chức năng của trang Site View:

- Hiển thị trạng thái hoạt động của trạm điện, thông tin cảnh báo thiết bị.
- Hiển thị các thông tin cơ bản của trạm điện (bao gồm các thông tin về sản lượng điện, công suất, năng lượng bức xạ mặt trời và tỉ số công suất), cung cấp các biểu đồ xu hướng sản lượng điện, năng lượng bức xạ mặt trời và các phân tích Performance Ratio (PR) của trạm điện.
- Hiển thị xếp hạng các KPIs quan trọng của biến tần như: Production (kwh), Yield (h) và Power Ratio (%).
- Hiển thị so sánh sản lượng điện được tạo ra từ các biến tần và tổng sản lượng điện đo được từ các đồng hồ đo năng lượng (Energy Meter).

Giao diện hiển thị trang Site View ở chế độ ngày:



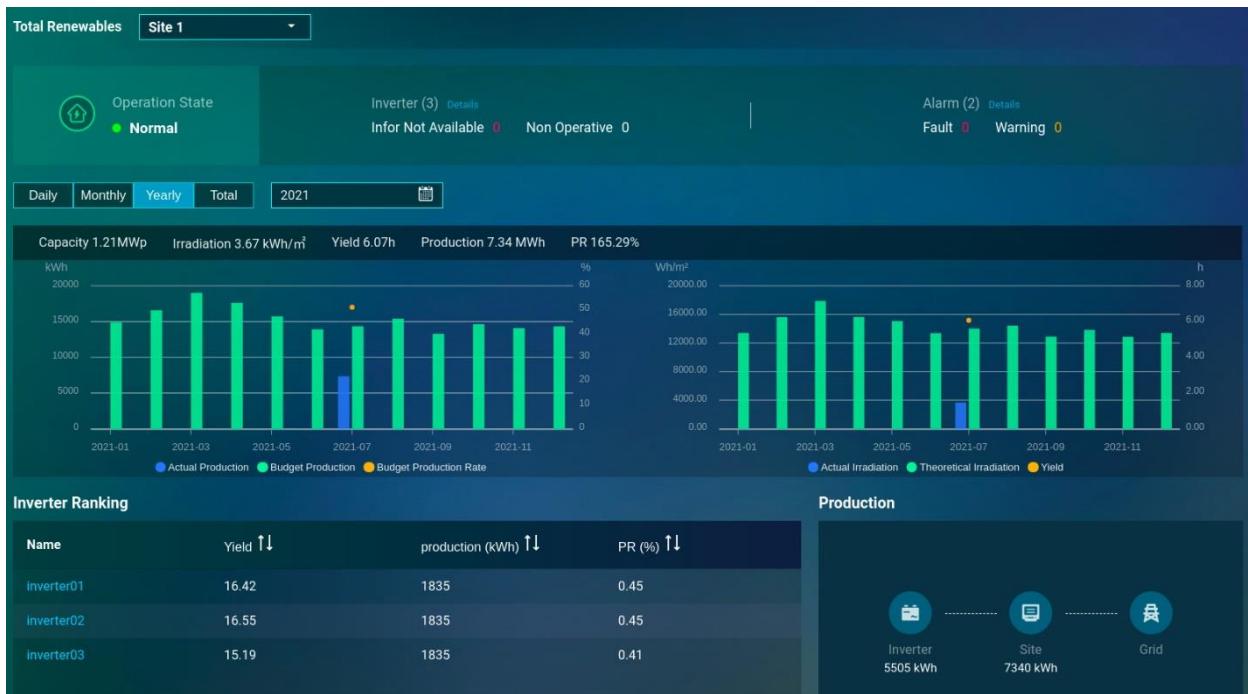
**Hình 4-54 Giao diện hiển thị trang Site View ở chế độ ngày**

Giao diện hiển thị trang Site View ở chế độ tháng:



**Hình 4-55 Giao diện hiển thị trang Site View ở chế độ tháng**

Giao diện hiển thị trang Site View ở chế độ năm:



**Hình 4-56 Giao diện hiển thị trang Site View ở chế độ năm**

#### 4.4.3.9 Giao diện trang Site KPI

Site KPI là trang có thể được truy cập bởi cả tài khoản admin và user. Trang Site KPI chủ yếu hiển thị các chỉ số KPI chính của trạm điện như độ chiếu xạ, sản lượng điện, PR ... Cho phép người dùng chọn các chế độ hiển thị theo ngày, tháng hoặc năm.

Các chức năng của trang Site KPI:

- Hiển thị các thông tin cơ bản của trạm điện như: tên trạm, công suất định (capacity), công suất hưu dụng ( active power), cường độ bức xạ mặt trời (Irradiance), và nhiệt độ (Temperature).
- Biểu đồ so sánh sản lượng điện (production), năng lượng bức xạ mặt trời (irradiation) và PR của trạm điện.
- Biểu đồ so sánh công suất hưu dụng (active power) và cường độ bức xạ mặt trời (Irradiance).
- Hiển thị bảng xếp hạng các KPIs quan trọng của biến tần bao gồm: Yield và Production.

- Hiển thị các thông tin về sản lượng lượng điện (Production), số giờ (Yield) và chỉ số phát thải CO<sub>2</sub> của trạm điện.
- Hiển thị so sánh tổng sản lượng điện được tạo ra từ các biến tần và tổng sản lượng điện đo được từ các đồng hồ đo năng lượng, đồng thời tính toán mức độ hao hụt.

Giao diện hiển thị trang Site KPI ở chế độ ngày:



Hình 4-57 Giao diện hiển thị trang Site KPI ở chế độ ngày

Giao diện hiển thị trang Site KPI ở chế độ tháng:



Hình 4-58 Giao diện hiển thị trang Site KPI ở chế độ tháng

Giao diện hiển thị trang Site KPI ở chế độ năm:



Hình 4-59 Giao diện hiển thị trang Site KPI ở chế độ năm

#### 4.4.3.10 Giao diện trang Device List

Device List là trang có thể được truy cập bởi cả tài khoản admin và user. Trang cung cấp danh sách các thiết bị và các chỉ số chính của từng thiết bị truy cập trong Site.

Các chức năng của trang Device List:

- Cho phép chọn loại thiết bị đã kết nối được hiển thị. Hiện hệ thống có 3 loại thiết bị bao gồm: Inverter, Energy Meter và Weather Station.
- Tóm quan nhanh về trạng thái và các chỉ số của từng thiết bị.
- Hỗ trợ tìm kiếm nhanh các thiết bị dựa theo tên thiết bị.
- Cho phép xếp hạng các thiết bị dựa theo các chỉ số của các thiết bị
- Cho phép xem thông tin chi tiết của từng thiết bị trong hệ thống, bằng cách click vào các thiết bị tương ứng.

Giao diện danh sách các Inverter: giao diện hiển thị danh sách các Inverter và các chỉ số của từng Inverter ( bao gồm State, Total Energy, Active Power, Input Power, Efficiency, Internal Temperature, ProductionToday, Yield Today).

Inverter Name	State	Syst. Diag.	Total Energy ↑	Active Power ↑	Input Power ↑	E
inverter01	Full Capacity	String Normal	1836	105	126	8%
inverter02	Full Capacity	String Normal	1836	134	108	9%
inverter03	Full Capacity	String Normal	1836	131	138	9%

Hình 4-60 Giao diện danh sách các Inverter

Giao diện danh sách các Energy Meter: giao diện hiển thị danh sách các Energy Meter và các chỉ số của từng Energy Meter( bao gồm State, Total Active Generated Energy, Total Active Consumed Energy, Total Reactive Generated Energy, Total Reactive Consumed Energy)

Meter Name	State	Type	Total Active Generated ↑	Total Active Consumed ↑	Total Reactive Generated ↑
EnergyMeter1	--	Energy Meter	158	171	148
EnergyMeter2	--	Energy Meter	196	133	103

**Hình 4-61 Giao diện danh sách các Energy Meter**

Giao diện hiển thị danh sách các Weather Station: giao diện hiển thi danh sách các Weather Station và các chỉ số của từng Weather Station (bao gồm State, POA, GHI, Ambient Temperature, Module Temperature, Humidity, Wind Speed, Wind Direction và Rainfall).

Weather Station Name	State	POA ↑	GHI ↑	Ambient Temp ↑	Module Temp 1 ↑	H
WeatherStation1	--	480	457	27	46	9.

**Hình 4-62 Giao diện danh sách các Weather Station**

Giao diện hiển thị chi tiết các thông số của một Inverter bao gồm:

- Một bảng hiển thị trạng thái hoạt động và các thông tin của Inverter bao gồm thông tin về Capacity, Internal Temperature và Efficiency.
- Một bảng hiển thị các channel về công suất, và đại lượng một chiều và xoay chiều của Inverter, cho phép chuyển đổi qua lại giữa các tab.
- Một bảng hiển thị các cảnh báo.
- Một biểu đồ so sánh sản lượng điện của Inverter và năng lượng bức xạ mặt trời, cho phép chọn các chế độ hiển thị theo ngày tháng hoặc năm.

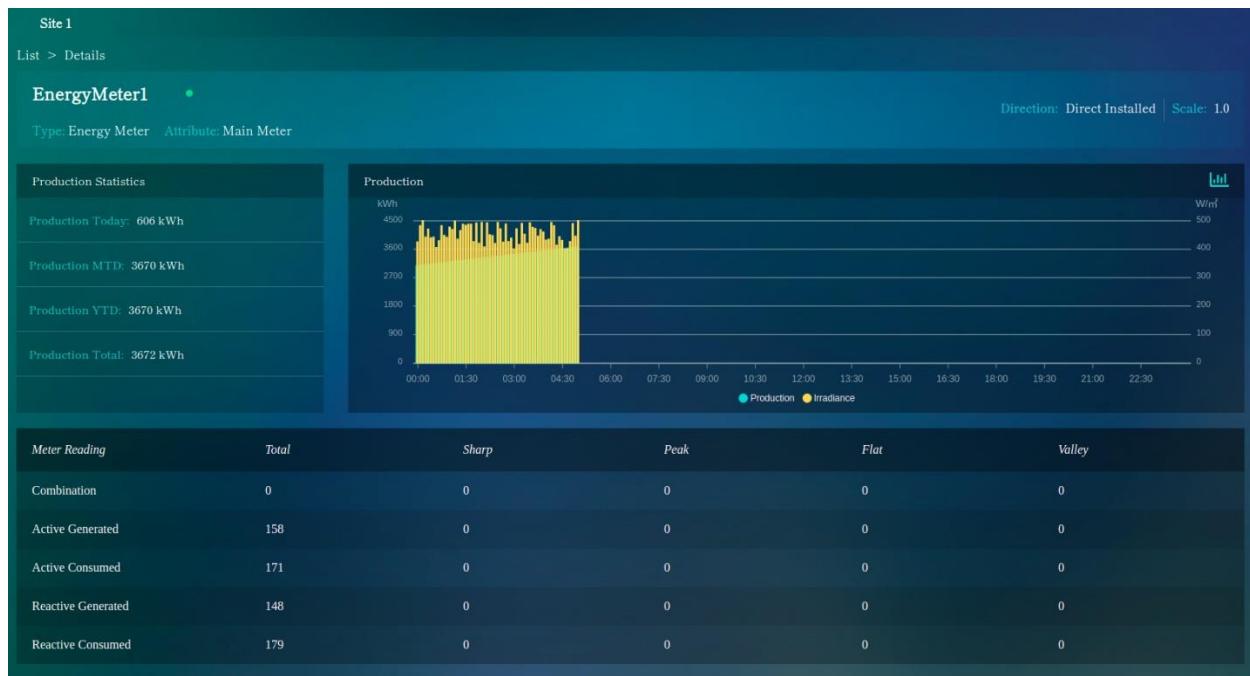
- Một biểu đồ vẽ các dòng String Current của Inverter và tính toán phần trăm Deviation của các dòng String này.
- Một bảng tổng hợp thông tin tất cả các channel của Inverter.



**Hình 4-63 Giao diện hiển thị chi tiết các thông số của một Inverter**

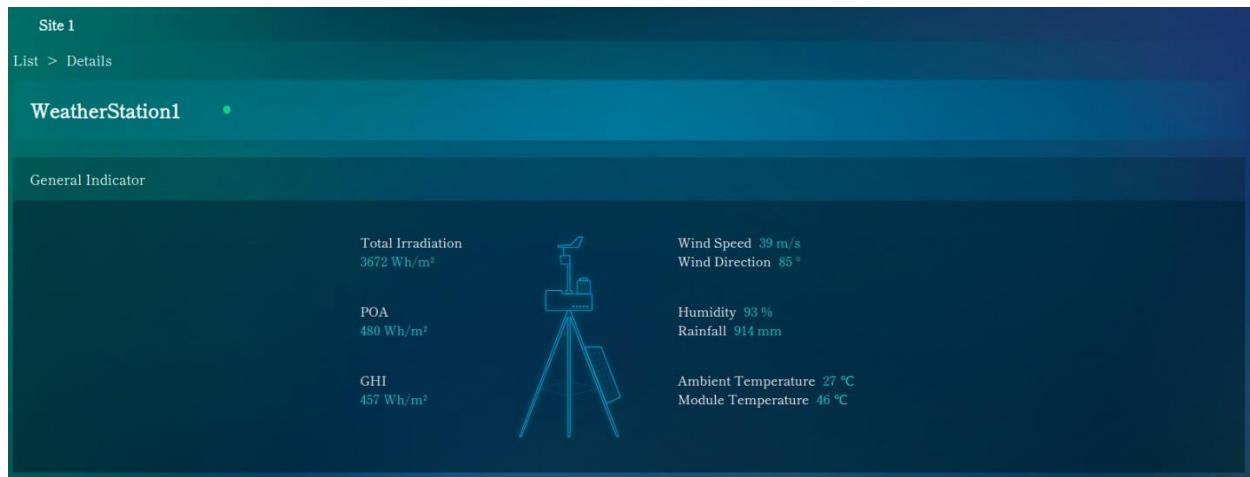
Giao diện hiển thị chi tiết các thông số của một Energy Meter bao gồm:

- Một bảng hiển thị thông tin trạng thái hoạt động của Energy Meter.
- Một bảng thống kê sản lượng điện đo được theo ngày, tháng, năm.
- Một biểu đồ so sánh lượng điện đo được (production) và cường độ bức xạ mặt trời (irradiance). Có hai chế độ hiển thị: biểu đồ cột và biểu đồ đường.
- Một bảng hiển thị chi tiết các channel của Energy Meter.



**Hình 4-64 Giao diện hiển thị chi tiết các thông số của Energy Meter**

Giao diện hiển thị chi tiết các thông số của một Weather Station:



**Hình 4-65 Giao diện hiển thị chi tiết các thông số của Weather Station**

#### 4.4.3.11 Giao diện trang Charting Tool

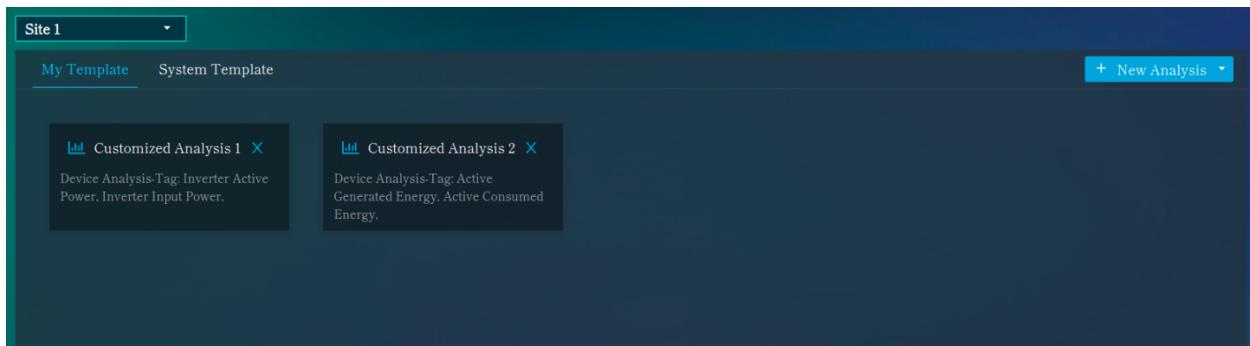
Trang Charting Tool là trang có thể được truy cập bởi cả tài khoản admin và user. Trang cung cấp công cụ để phân tích và hiển thị các channel của các Device thuộc một trạm bất kỳ dưới dạng các biểu đồ.

Các chức năng chính của trang Charting Tool:

- Cho phép tạo mới các phân tích.
- Cho phép lựa chọn các channel của các device cần phân tích.
- Cho phép lựa chọn dạng biểu đồ phân tích: biểu đồ đường hoặc cột.
- Hiển thị dữ liệu phân tích dưới dạng biểu đồ và dạng bảng.
- Cho phép chọn các mốc thời gian phân tích: Today, Last 3 days, Last 7 days, Last 30 days hoặc chọn khoảng thời gian bất kỳ.
- Cho phép người dùng lưu lại hoặc xoá các phân tích.
- Xuất dữ liệu phân tích dưới dạng file CSV.

Giao diện chính của trang Charting Tool gồm:

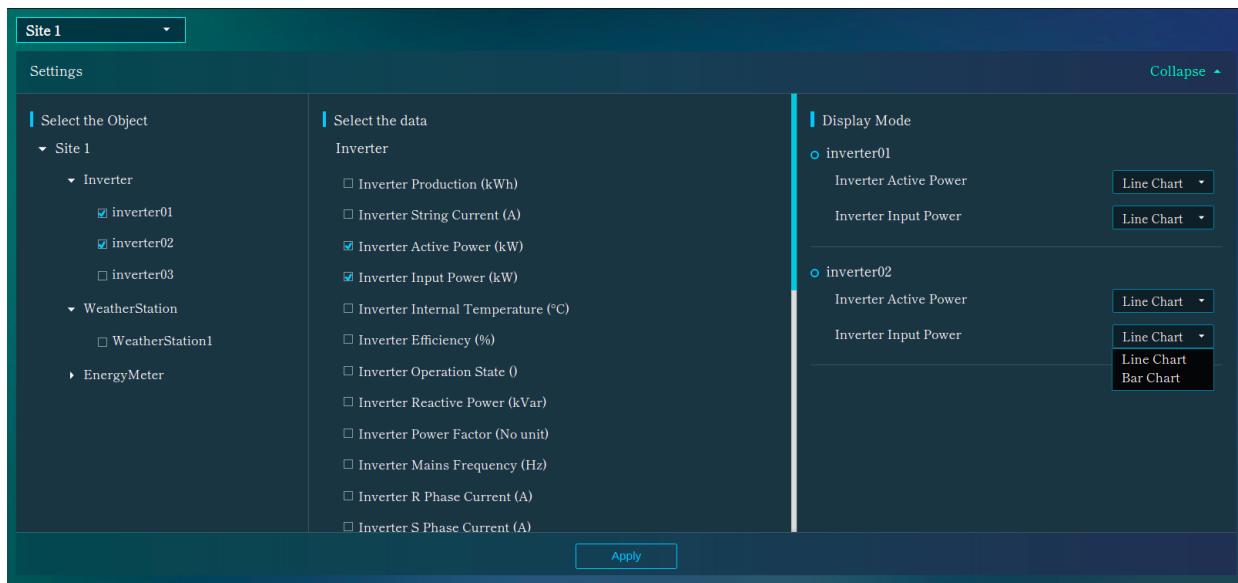
- Danh sách các phân tích mà người dùng đã lưu trước đó, cho phép người dùng xoá một phân tích bất kỳ bằng cách bấm vào biểu tượng “X” tương ứng.
- Nút “New Analysis” cho phép người dùng tạo mới các phân tích.



**Hình 4-66 Giao diện chính của trang Charting Tool**

Giao diện cấu hình các phân tích bao gồm:

- Cột thứ nhất: cho phép người dùng chọn lựa các device cần phân tích.
- Cột thứ hai: cho phép người dùng chọn lựa các channel ứng với các device cần phân tích.
- Cột thứ ba: cho phép người dùng chọn lựa dạng biểu đồ để phân tích.

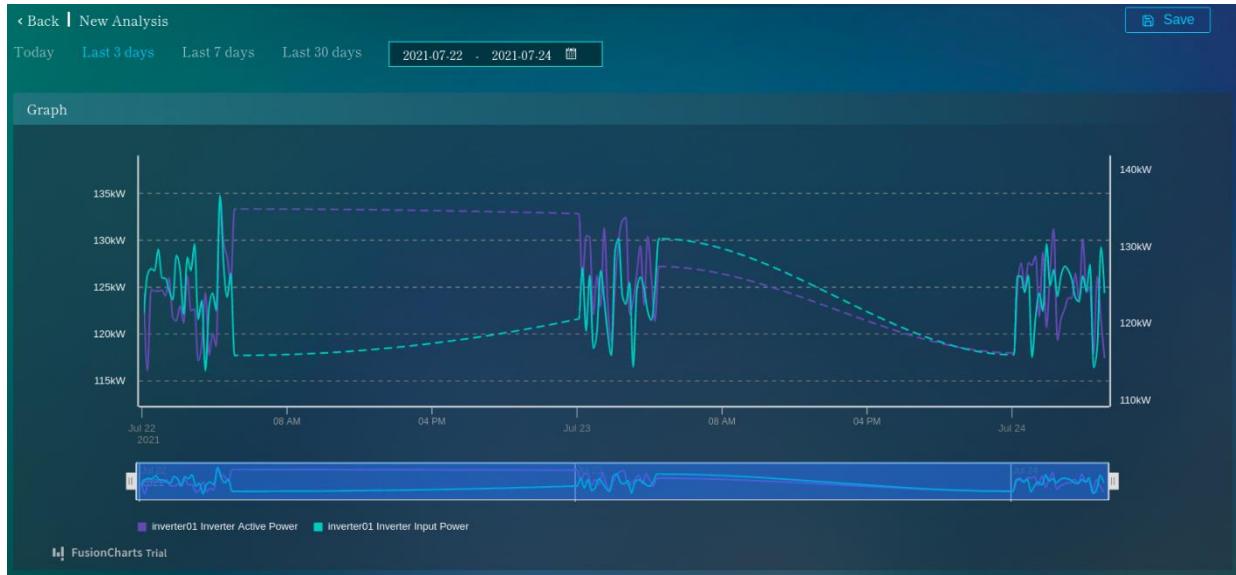


**Hình 4-67 Giao diện cấu hình các phân tích của trang charting tool**

Giao diện hiển thị dữ liệu phân tích dưới dạng biểu đồ:



**Hình 4-68 Giao diện hiển thị dữ liệu phân tích dưới dạng biểu đồ của trang Charting tool**



**Hình 4-69 Giao diện hiển thị dữ liệu phân tích dạng biểu đồ của trang Charting Tool**

Giao diện hiển thị dữ liệu phân tích dạng bảng:

Data		
Date	inverter01 Inverter Active Power	inverter01 Inverter Input Power
2021-07-24 00:00:00	147	112
2021-07-24 00:01:00	100	129
2021-07-24 00:02:00	113	100
2021-07-24 00:03:00	124	120
2021-07-24 00:04:00	119	114
2021-07-24 00:05:00	102	103
2021-07-24 00:06:00	118	108
2021-07-24 00:07:00	111	113
2021-07-24 00:08:00	113	107
2021-07-24 00:09:00	133	146
2021-07-24 00:10:00	134	118
2021-07-24 00:11:00	102	121

Rows per page: 100 ▾

**Hình 4-70 Giao diện hiển thị dữ liệu phân tích dạng bảng của trang Charting Tool**

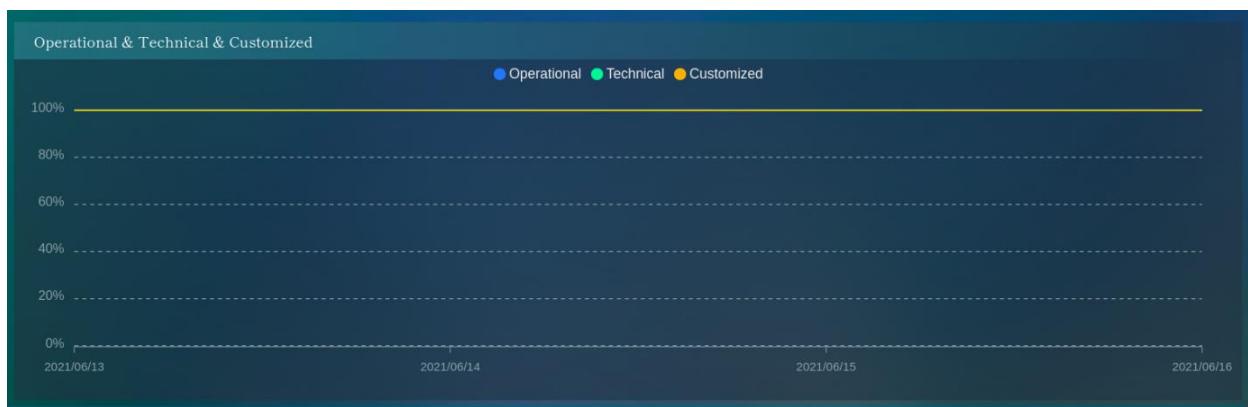
#### 4.4.3.12 Giao diện trang Availability

Trang Availability là trang có thể được truy cập bởi cả tài khoản Admin và user. Trang hiển thị các thông kê thời gian hoạt động, sửa chữa, bảo trì của một trạm điện. Trang hiện tại chỉ mới hoàn thiện phần giao diện, việc phân tích kết nối dữ liệu đang trong quá trình triển.

Giao diện trang Availability:



**Hình 4-71 Giao diện trang Availability**



**Hình 4-72 Giao diện trang Availability**

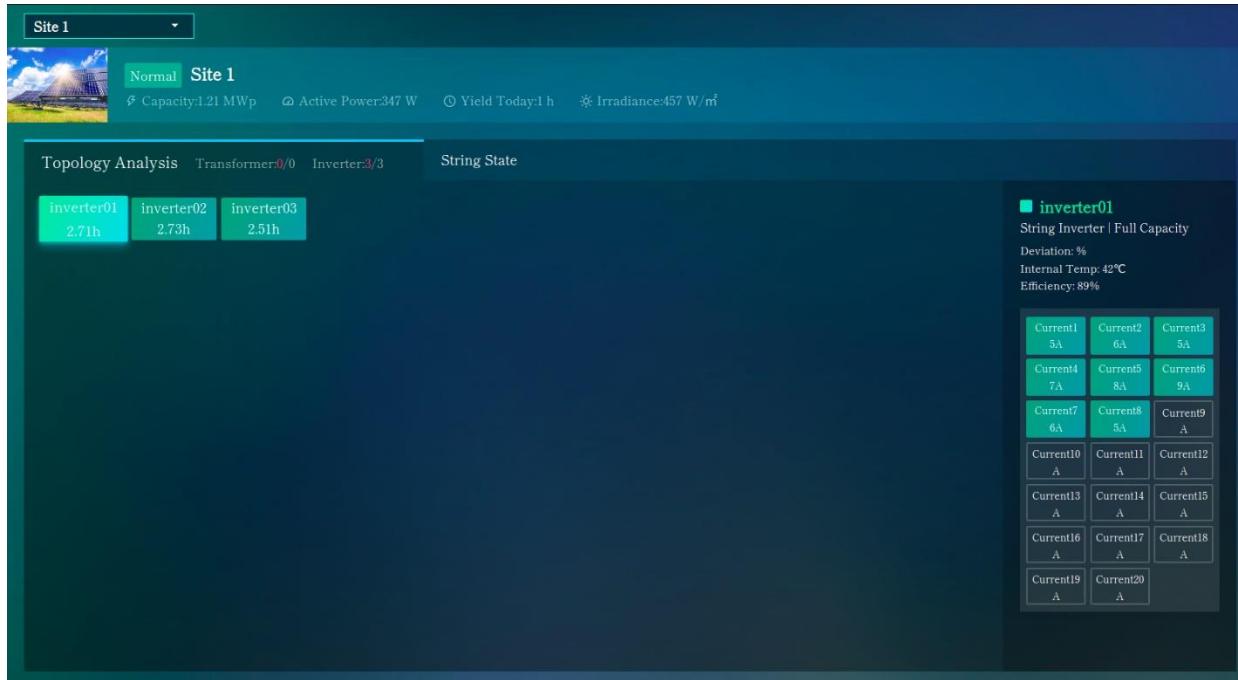
#### 4.4.3.13 Giao diện Topology Analysis

Topology Analysis là trang có thể được truy cập bởi cả tài khoản admin và user. Trang hiển thị chi tiết về các dòng String Current của các Inverter.

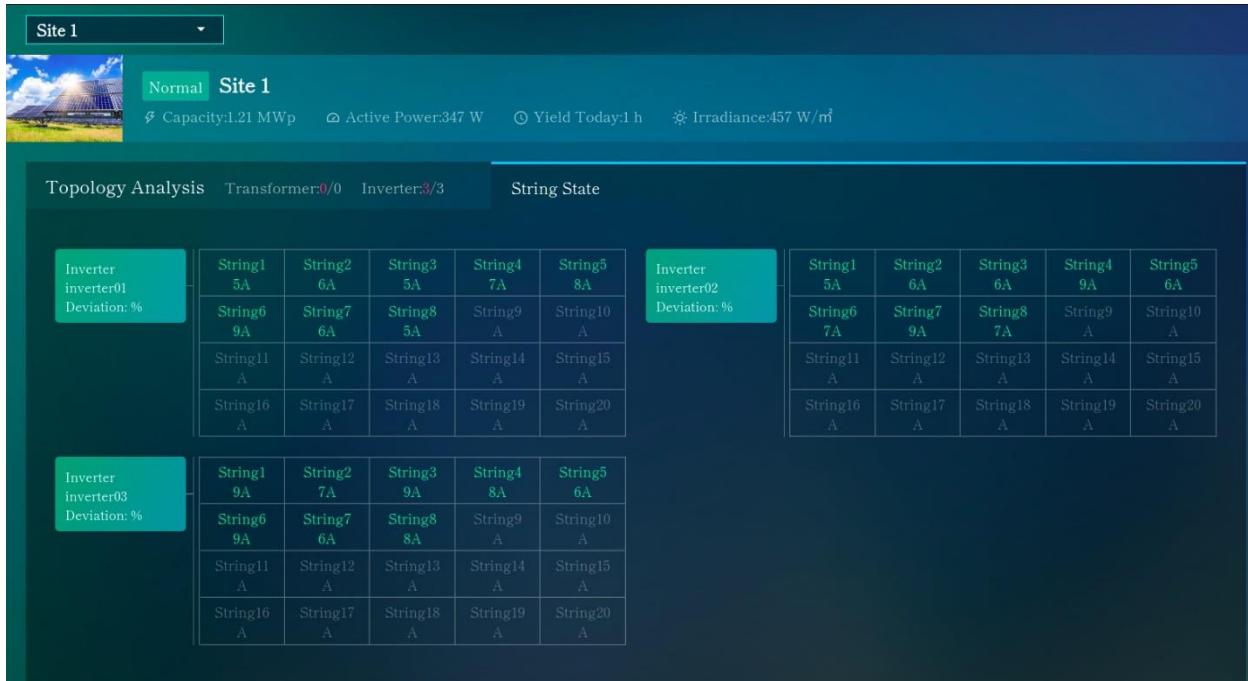
Các chức năng chính của trang:

- Hiển thị các thông tin cơ bản của trạm điện: trạng thái, capacity, active power, yield, irradiance.
- Hiển thị danh sách các Inverter với các thông số.
- Hiển thị trạng thái và giá trị của các dòng String Current của các Inverter.

Giao diện trang Topology Analysis:



Hình 4-73 Giao diện trang Topology Analysis

**Hình 4-74 Giao diện trang Topology Analysis**

#### 4.4.3.14 Giao diện trang Budget Production Input

Budget Production Input là trang có thể được truy cập bởi cả tài khoản admin và user.

Các chức năng của trang:

- Hiển thị sản lượng điện theo lý thuyết (budget production) và sản lượng điện thực tế (actual production) dưới dạng biểu đồ so sánh.
- Tính toán phần trăm mức độ hoàn thành sản lượng điện (Production Budget Completion Rate).
- Cho phép chọn năm để hiển thị.
- Cho phép nhập vào các giá trị sản lượng điện lý thuyết theo từng tháng với năm tương ứng.
- Cho phép chỉnh sửa hoặc xoá các giá trị sản lượng điện lý thuyết đã nhập.

Giao diện trang Budget Production Input:



**Hình 4-75 Giao diện trang Budget Production Input**

Giao diện nhập dữ liệu sản lượng điện lý thuyết:

The screenshot shows a modal dialog titled "Add Record". At the top left is a "Year" dropdown set to "2021". Below it is a grid for entering monthly electricity production data. The grid has two columns for each month, labeled "Jan" through "Dec", with "kWh" units indicated next to the input fields. The grid is as follows:

	Jan	kWh	Feb	kWh	
Mar	[ ]	kWh	Apr	[ ]	kWh
May	[ ]	kWh	Jun	[ ]	kWh
Jul	[ ]	kWh	Aug	[ ]	kWh
Sep	[ ]	kWh	Oct	[ ]	kWh
Nov	[ ]	kWh	Dec	[ ]	kWh

At the bottom right of the modal are "Cancel" and "OK" buttons.

**Hình 4-76 Giao diện nhập dữ liệu sản lượng điện lý thuyết**

#### 4.4.3.15 Giao diện trang Budget Insolation Input

Budget Insolation Input là trang có thể được truy cập bởi tài khoản admin và user.

Các chức năng của trang:

- Hiển thị năng lượng bức xạ mặt trời lý thuyết (Theoretical Irradiation) và năng lượng bức xạ mặt trời được đo (Actual Irradiation) dưới dạng biểu đồ so sánh.
- Cho phép chọn năm để hiển thị.
- Cho phép nhập vào các giá trị năng lượng bức xạ mặt trời lý thuyết theo từng tháng với năm tương ứng.
- Cho phép chỉnh sửa hoặc xoá các giá trị năng lượng bức xạ mặt trời lý thuyết đã nhập.

Giao diện trang Budget Insolation Input:

**Hình 4-77 Giao diện trang Budget Insolation Input**

Giao diện nhập các giá trị năng lượng bức xạ mặt trời lý thuyết:

Add Record

Year

Month	Value	Unit	Month	Value	Unit
Jan	<input type="text"/>	Wh/m <sup>2</sup>	Feb	<input type="text"/>	Wh/m <sup>2</sup>
Mar	<input type="text"/>	Wh/m <sup>2</sup>	Apr	<input type="text"/>	Wh/m <sup>2</sup>
May	<input type="text"/>	Wh/m <sup>2</sup>	Jun	<input type="text"/>	Wh/m <sup>2</sup>
Jul	<input type="text"/>	Wh/m <sup>2</sup>	Aug	<input type="text"/>	Wh/m <sup>2</sup>
Sep	<input type="text"/>	Wh/m <sup>2</sup>	Oct	<input type="text"/>	Wh/m <sup>2</sup>
Nov	<input type="text"/>	Wh/m <sup>2</sup>	Dec	<input type="text"/>	Wh/m <sup>2</sup>

**Hình 4-78 Giao diện nhập giá trị năng lượng bức xạ mặt trời lý thuyết**

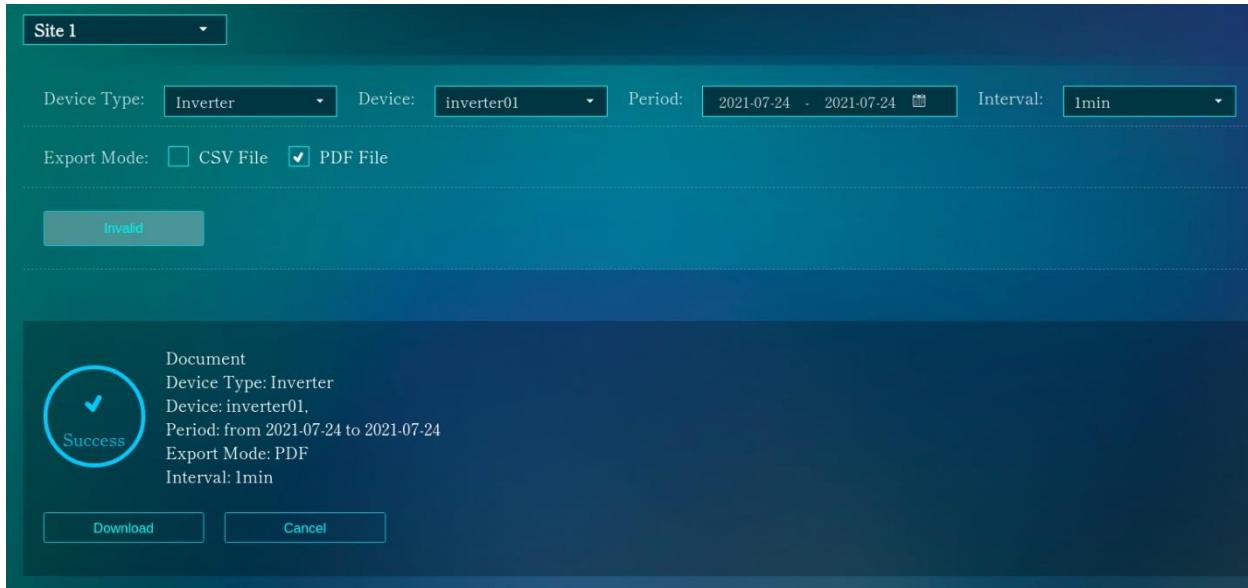
#### 4.4.3.16 Giao diện trang Data Report

Data Report là trang có thể được truy cập bởi cả tài khoản admin và user. Trang cung cấp khả năng xuất các báo cáo về dữ liệu.

Các chức năng của trang:

- Cho phép chọn loại device.
- Cho phép chọn các device tương ứng.
- Cho phép chọn khoảng thời gian cần xuất báo cáo.
- Cho phép chọn các interval tương ứng: 1min, 15min, 30min, 1hour, 1day.
- Cho phép chọn loại file để xuất báo cáo: CSV hoặc PDF.

Giao diện trang Data Report:

**Hình 4-79 Giao diện trang Data Report**

Giao diện file PDF Report:

### Data Report

<b>Id</b>	<b>Device</b>	<b>Inverter Production</b>	<b>Inverter Active Power</b>	<b>Time</b>
0	inverter01	1533	147	2021-07-24T00:00:00.000Z
1	inverter01	1534	100	2021-07-24T00:01:00.000Z
2	inverter01	1535	113	2021-07-24T00:02:00.000Z
3	inverter01	1536	124	2021-07-24T00:03:00.000Z
4	inverter01	1537	119	2021-07-24T00:04:00.000Z
5	inverter01	1538	102	2021-07-24T00:05:00.000Z
6	inverter01	1539	118	2021-07-24T00:06:00.000Z
7	inverter01	1540	111	2021-07-24T00:07:00.000Z
8	inverter01	1541	113	2021-07-24T00:08:00.000Z
9	inverter01	1542	133	2021-07-24T00:09:00.000Z
10	inverter01	1543	134	2021-07-24T00:10:00.000Z

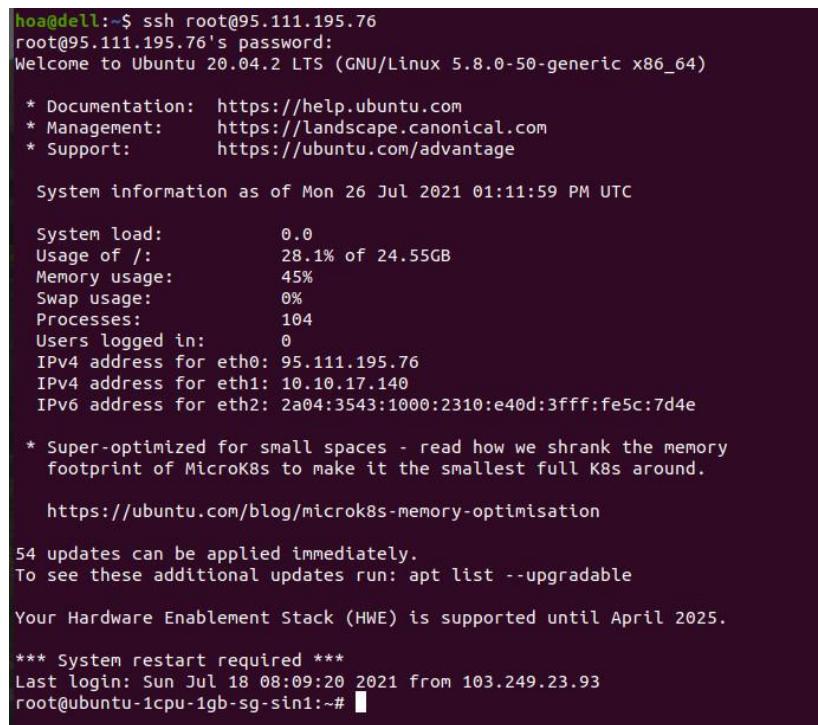
**Hình 4-80 Giao diện file PDF Report**

#### 4.4.4 Deploy Web App lên Linux Server

Các bước tiến hành Deploy web app:

- Bước 1: Truy cập vào server

Sau khi bạn tạo một Cloud VPS mới, thường thì nó sẽ ở trạng thái mặc định, tức là chưa có thêm bất cứ ứng dụng nào khác được cài thêm. Ngoài ra, tùy nhà cung cấp VPS mà bạn sẽ được cấp tài khoản **root** hoặc tài khoản user bình thường nhưng có quyền sudo. Bạn sẽ truy cập vào server thông qua giao thức SSH và sẽ cấu hình server bằng dòng lệnh trong một cửa sổ terminal.



```
hoa@dell:~$ ssh root@95.111.195.76
root@95.111.195.76's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.8.0-50-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 System information as of Mon 26 Jul 2021 01:11:59 PM UTC

 System load:          0.0
 Usage of /:           28.1% of 24.55GB
 Memory usage:         45%
 Swap usage:           0%
 Processes:            104
 Users logged in:     0
 IPv4 address for eth0: 95.111.195.76
 IPv4 address for eth1: 10.10.17.140
 IPv6 address for eth2: 2a04:3543:1000:2310:e40d:3fff:fe5c:7d4e

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

 https://ubuntu.com/blog/microk8s-memory-optimisation

 54 updates can be applied immediately.
 To see these additional updates run: apt list --upgradable

 Your Hardware Enablement Stack (HWE) is supported until April 2025.

 *** System restart required ***
 Last login: Sun Jul 18 08:09:20 2021 from 103.249.23.93
root@ubuntu-1cpu-1gb-sg-sin1:~#
```

Hình 4-81 Giao diện truy cập vào linux server

- Bước 2: Cài đặt các ứng dụng và các gói thư viện cần thiết trên Server

Sau khi truy cập vào server, chúng ta cần cài đặt cái ứng dụng và các thư viện cần thiết để chương trình web app có thể hoạt động, ví dụ như: Node, MongoDB, git, PM2, ...

- Bước 3: Đưa mã nguồn Web App lên server.
- Bước 4: Chạy các ứng dụng web sử dụng PM2

PM2 là một free open source, hiện đại, hiệu quả, cross-platform và quan trọng là nó free cho ứng dụng sử dụng Node.js với load balancer thích hợp. Nó hoạt động trên Linux, MacOS cũng như Windows. Nó hỗ trợ giám sát Application, quản lý hiệu quả các dịch vụ quy trình vi mô, chạy các ứng dụng ở chế độ cluster, khởi động và tắt Application nodejs một cách gọn gàng.

root@ubuntu-1cpu-1gb-sg-sin1:~# pm2 status						
id	name	mode	ø	status	cpu	memory
0	index	fork	0	online	0%	60.5mb
2	index	fork	0	online	0%	46.5mb
1	server	fork	0	online	0%	60.7mb

**Hình 4-82 Giao diện quản lý các ứng dụng NodeJs sử dụng PM2**

- Bước 5: Cấu hình Nginx làm Reverse Proxy Server.

## 5. KẾT QUẢ THỰC HIỆN

### 5.1 IoT Gateway

#### 5.1.1 Phần cứng

Dựa trên phần cứng có sẵn từ trước, các khôi chức năng cần thiết hoạt động bình thường, bao gồm:

- Các cổng RS485 cho giao thức Modbus.
- Khối Ethernet có thể kết nối vào mạng giúp board có thể truyền nhận dữ liệu qua ethernet
- Khối SD card hoạt động trên giao thức SDIO bình thường, trao đổi dữ liệu giữa STM32 và SD card không bị lỗi.

#### 5.1.2 Phần mềm

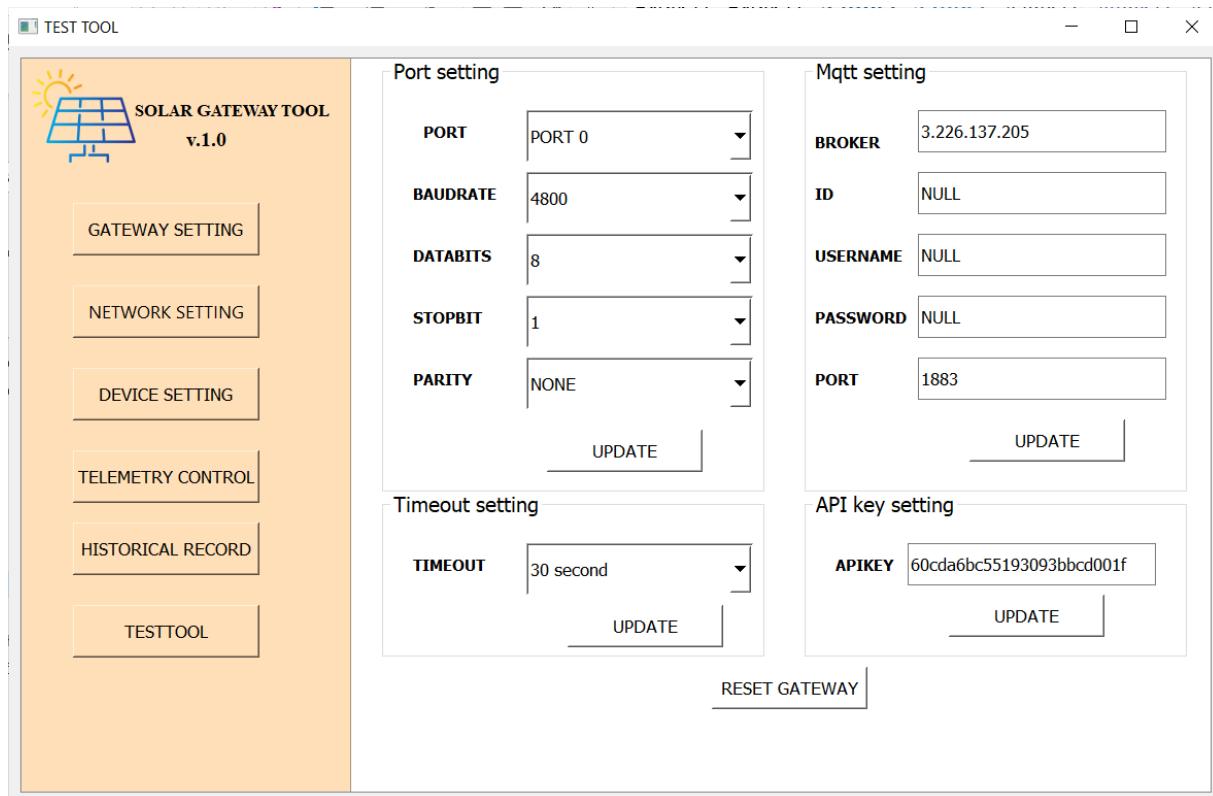
Các kết quả đạt được:

- Tích hợp được hệ điều hành thời gian thực FreeRTOS và thư viện LwIP lên Gateway giúp cho việc triển khai multitasking và sử dụng các dịch vụ liên quan đến mạng dễ dàng hơn.

- Tích hợp và tuỳ chỉnh FreeModbus ở chế độ 2 Master thành công lên nền tảng FreeRTOS.
- FreeModbus ở chế độ RTU, triển khai các Function chính: Read/Write Single/Multiple Register, Read/Write Coil Register, Read Input Register. Các cơ chế báo lỗi: lỗi gói tin, lỗi quá thời gian chờ, lỗi sai mã chức năng, ...
- Tích hợp thư viện MQTT, kết nối thành công với Broker MQTT ở các QoS khác nhau.
- Triển khai cầu chuyển đổi giữa giao thức MQTT và Modbus, bao gồm 2 luồng dữ liệu chính là Downlink Request từ phía Server và Uplink Response từ phía Gateway. Các gói tin được chuẩn hoá theo kiểu JSON để dễ dàng tương tác với IoT Platform Server.
- Giao diện Serial Configuration để có thể thay đổi các giá trị khi cài đặt. Các giá trị sẽ được lưu vào vùng nhớ Flash của vi điều khiển hoặc SD card trên board.

### 5.1.3 Ứng dụng Desktop

Phần mềm hoạt động trên máy tính chạy đúng theo các yêu cầu đặt ra.



Hình 5-1 Giao diện ứng dụng desktop

### 5.1.4 Các bài thử nghiệm, đánh giá IoT gateway

#### 5.1.4.1 Thủ nghiệm với mạng đơn thiết bị

##### Thực hiện trên thiết bị ảo

Trong phần này, gateway sẽ được kết nối phần mềm Modbus Slave giả lập thiết bị modbus thông qua cổng serial.

Các thông số của bài kiểm tra:

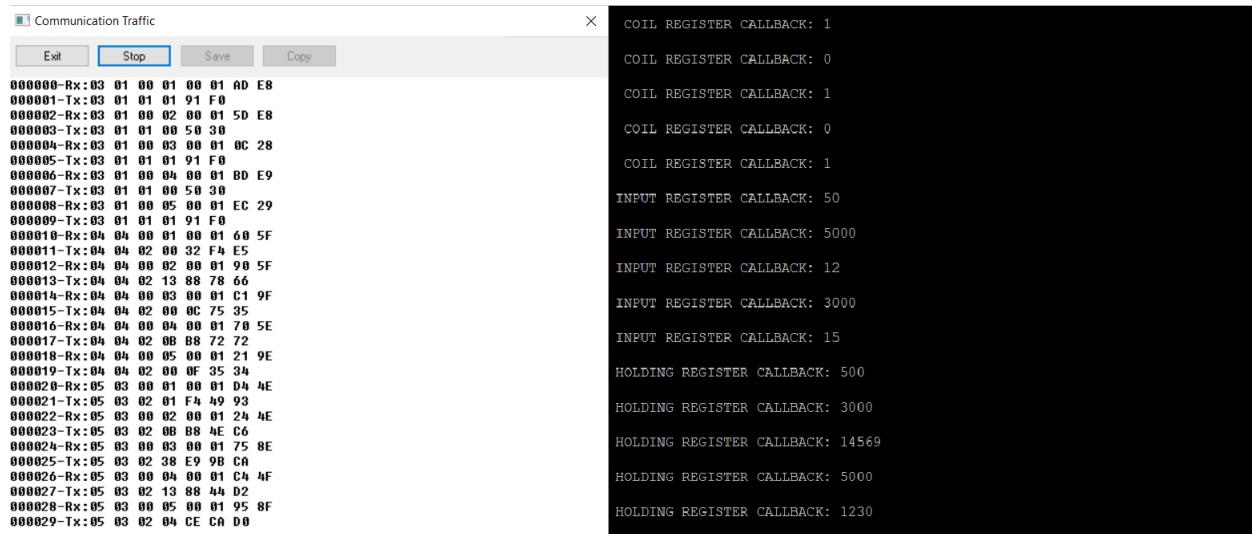
- Modbus Slave: RTU Mode, Serial 9600, 8, N1.
- Function: Read Holding Register(0x03), Read Coil Register(0x01), Read Input Register(0x04).

The figure consists of three separate windows, each showing a table of data for a Modbus Slave configuration:

- COIL\_SLAVE.mbs**: ID = 3; F = 01. The table has columns for Alias and Value. Row 3 has an alias of 1 and a value of 1, highlighted in blue.
- INPUT\_SLAVE...**: ID = 4; F = 04. The table has columns for Alias and Value. Row 9 has an alias of 45632 and a value of 45632, highlighted in blue.
- HOLDING\_SLAVE...**: ID = 5; F = 03. The table has columns for Alias and Value. Row 9 has an alias of 12364 and a value of 12364, highlighted in blue.

**Hình 5-2 Các thiết bị giả lập Modbus Slave (Từ trái sang: Coil Register Slave, Input Register Slave, Holding Register Slave)**

## Kết quả:



**Hình 5-3 View communication trên giao diện Modbus Slave (bên trái), giá trị đọc được từ các Modbus Slave (bên phải)**

## Giải thích:

- Gateway lần lượt đọc các thanh ghi từ 1 đến 5 của từng Slave.
- Giao diện View communication là luồng dữ liệu request – response giữa gateway và slave (Rx: là gói từ request từ Gateway, Tx: là gói tin response từ Slave)
- Giao diện terminal là giá trị cụ thể mà gateway đọc được.

## Kết quả:

- Gateway có thể giao tiếp với phần mềm giả lập Modbus Slave thông qua các function cơ bản của Modbus: Read/Write Single Holding Registers, Read Coil Register, Read Input Register.

## Thực hiện trên thiết bị thực

Trong phần này, Gateway sẽ được lần lượt kết nối với cảm biến nhiệt độ, độ ẩm SHT20.

Các thông số bài kiểm tra:

- Modbus Slave: SHT20, RTU Mode, Serial 9600, 8, N1.

- Function: Read Input Register (0x04)

```
INPUT REGISTER CALLBACK AT PORT 0: 312
INPUT REGISTER CALLBACK AT PORT 0: 763
INPUT REGISTER CALLBACK AT PORT 0: 312
INPUT REGISTER CALLBACK AT PORT 0: 750
INPUT REGISTER CALLBACK AT PORT 0: 312
INPUT REGISTER CALLBACK AT PORT 0: 746
INPUT REGISTER CALLBACK AT PORT 0: 313
INPUT REGISTER CALLBACK AT PORT 0: 742
INPUT REGISTER CALLBACK AT PORT 0: 313
INPUT REGISTER CALLBACK AT PORT 0: 742
INPUT REGISTER CALLBACK AT PORT 0: 314
INPUT REGISTER CALLBACK AT PORT 0: 737
INPUT REGISTER CALLBACK AT PORT 0: 314
INPUT REGISTER CALLBACK AT PORT 0: 736
INPUT REGISTER CALLBACK AT PORT 0: 314
INPUT REGISTER CALLBACK AT PORT 0: 735
INPUT REGISTER CALLBACK AT PORT 0: 314
INPUT REGISTER CALLBACK AT PORT 0: 738
```

**Hình 5-4 Giá trị đọc được từ SHT20 quan sát qua giao diện terminal của gateway**

**Giải thích:**

- các giá trị đọc được: 312 là giá trị nhiệt độ chưa qua xử lý scale, 763 là giá trị độ ẩm chưa qua xử lý scale.

**Nhận xét:**

- Gateway có thể giao tiếp với thiết bị Modbus thật thông qua các function: Read Input Registers.

*5.1.4.2 Thủ nghiệm với mạng nhiều thiết bị*

*Thực hiện trên thiết bị ảo*

Thực hiện đọc ghi lần lượt và liên tục tất cả thiết bị trong mạng Modbus. Các thông số bài kiểm tra:

- Modbus Slave: RTU Mode, Serial 9600, 8, N1.

- Function: Read Single/Multi Register (0x03). Write Single Register (0x06)

Trong phần này, gateway sẽ được kết nối với phần mềm Modbus Slave giả lập nhiều thiết bị modbus thông qua cổng serial. Các thanh ghi, kiểu dữ liệu được lấy từ tài liệu kỹ thuật được cấp từ nhà sản xuất, bao gồm giả lập các thiết bị: Inverter (Model: SUN2000 V200R002 [16] được cung cấp bởi Huawei), Energy Meter (Model: PM5300 [17] được cung cấp bởi Schneider Electric), Weather Station (Model: VSN800-14 [18] được cung cấp bởi Sunspec).

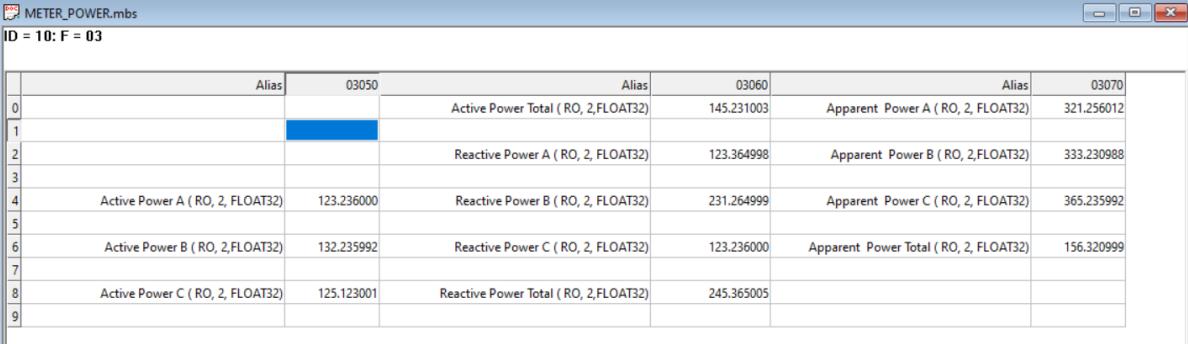


**Hình 5-5 Các thiết bị thực được giả lập dữ liệu (Inverter SUN2000 V200R002, Energy Meter PM5350, Weather Station VSN800-14)**

The screenshot shows a software interface titled "INVERTER\_Ud.mbs" with the identifier "ID = 9: F = 03". The table lists the following data:

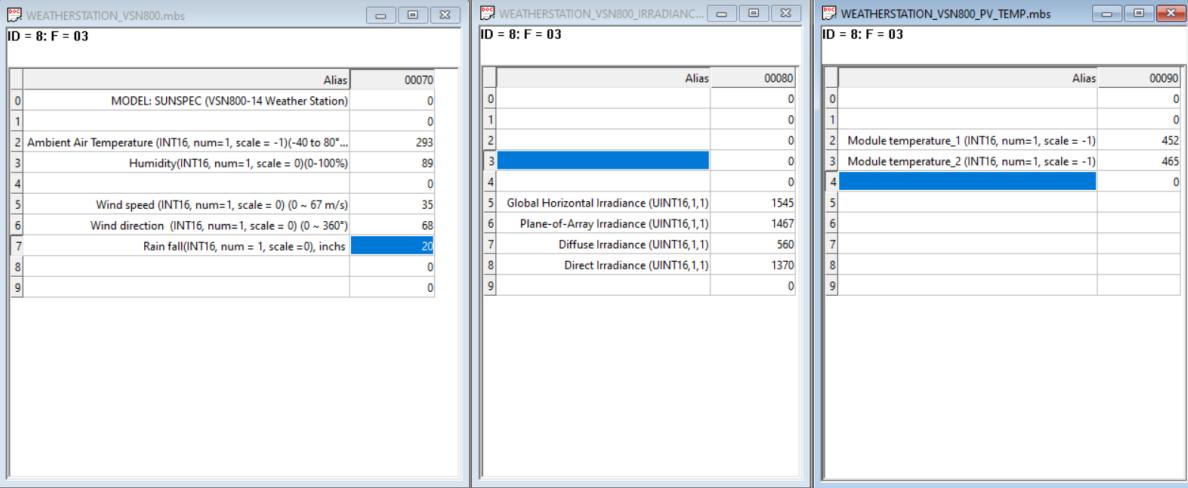
	Alias	Value	Alias	Value
0	U, I, Uab, Frequency, Temperature	32270	Ia(RO, U16, 10, 1)	20
1			Ib(RO, U16, 10, 1)	19
2			Ic(RO, U16, 10, 1)	18
3				
4	Uab (RO, U16, 10, 1)	3799		
5	Ubc(RO, U16, 10, 1)	3801		
6	Uca(RO, U16, 10, 1)	3789		
7	Ua(RO, U16, 10, 1)	2193		
8	Ub(RO, U16, 10, 1)	2186		
9	Uc(RO, U16, 10, 1)	2195		

**Hình 5-6 Inverter simulator**



The screenshot shows a software interface titled "METER\_POWER.mbs" with the identifier "ID = 10: F = 03". It displays a table with three columns of data:

	Alias	Value	Alias	Value	Alias	Value	
0		03050		Active Power Total ( RO, 2,FLOAT32)	145.231003	Apparent Power A ( RO, 2, FLOAT32)	321.256012
1				Reactive Power A ( RO, 2, FLOAT32)	123.364998	Apparent Power B ( RO, 2,FLOAT32)	333.230988
2				Reactive Power B ( RO, 2, FLOAT32)	231.264999	Apparent Power C ( RO, 2, FLOAT32)	365.235992
3				Reactive Power C ( RO, 2, FLOAT32)	123.236000	Apparent Power Total ( RO, 2, FLOAT32)	156.320999
4		Active Power A ( RO, 2, FLOAT32)	123.236000				
5							
6		Active Power B ( RO, 2,FLOAT32)	132.235992				
7							
8		Active Power C ( RO, 2, FLOAT32)	125.123001	Reactive Power Total ( RO, 2,FLOAT32)	245.365005		
9							

**Hình 5-7 Energy Meter simulator**


The screenshot shows three software windows side-by-side, all titled "WEATHERSTATION\_VSN800.mbs" with the identifier "ID = 8: F = 03".

- Left Window:** Alias 00070
 

	Alias	Value
0	MODEL: SUNSPEC (VSN800-14 Weather Station)	0
1		0
2	Ambient Air Temperature (INT16, num=1, scale = -1)(-40 to 80°...)	293
3	Humidity(INT16, num=1, scale = 0)(0-100%)	89
4		0
5	Wind speed (INT16, num=1, scale = 0) (0 ~ 67 m/s)	35
6	Wind direction (INT16, num=1, scale = 0) (0 ~ 360°)	68
7	Rain fall(INT16, num = 1, scale = 0), inches	20
8		0
9		0
- Middle Window:** Alias 00080
 

	Alias	Value
0		0
1		0
2		0
3		0
4		0
5	Global Horizontal Irradiance (UINT16,1,1)	1545
6	Plane-of-Array Irradiance (UINT16,1,1)	1467
7	Diffuse Irradiance (UINT16,1,1)	560
8	Direct Irradiance (UINT16,1,1)	1370
9		0
- Right Window:** Alias 00090
 

	Alias	Value
0		0
1		0
2	Module temperature_1 (INT16, num=1, scale = -1)	452
3	Module temperature_2 (INT16, num=1, scale = -1)	465
4		0
5		
6		
7		
8		
9		

**Hình 5-8 Weather station simulator****Bảng 4 Bảng thống kê kết quả phản hồi của Modbus Slave trong mạng đa thiết bị**

Tổng số channel	Tổng số gói đọc	Số gói lỗi	Loại lỗi
93	1000	3	Timeout
93	5000	9	Timeout
93	10000	23	Timeout
93	50000	84	Timeout

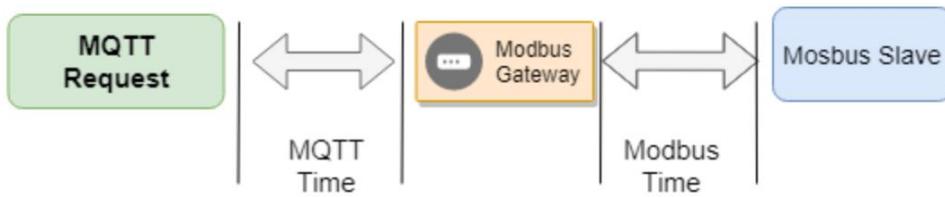
## Kết quả:

- Các thiết bị hoạt động bình thường và không xung đột lẫn nhau, tuy nhiên vẫn có các gói bị lỗi

## Nhận xét:

- Qua các quá trình kiểm tra và tìm hiểu, các gói tin quá thời gian không phản hồi do bị lỗi thông qua đường truyền và bị loại khỏi hàng đợi bởi FreeModbus; các đường bus kết nối tiếp xúc chưa tốt gây ra lỗi ở vài gói tin.

### 5.1.4.3 Thủ nghiệm tốc độ tối đa



**Hình 5-9 Mô hình thời gian gói tin đi trong mạng Modbus.**

Mô hình thời gian trễ của luồng dữ liệu:

- MQTT Time: Thời gian của giao thức MQTT, bao gồm thời gian gói tin Downlink Request được gửi đến Gateway thông qua trung gian Broker và thời gian gói tin Uplink Response phản hồi đến nơi yêu cầu. Trong bài kiểm tra, gói tin kiểm tra sẽ không được gửi xuống thiết bị Slave mà sẽ phản hồi ngay lập tức về Server khi đi đến Gateway.
- Modbus Time: Thời gian của giao thức Modbus từ khi bắt đầu nhận Request được yêu cầu đến khi trả về gói tin phản hồi từ Slave. Trong bài kiểm tra, thời gian này được tính bằng thời gian tổng gói tin đi trong mạng trừ MQTT Time.

Các thông số trong bài kiểm tra:

- Mạng LAN: 100Mbps, Router Huawei HG8045A.
- MQTT Broker: Mosquitto broker on cloud (<https://mosquito.org>)
- MQTT Client: Gateway Modbus with LAN8720A at 100Mbps

- Modbus: RTU Mode, Serial: 9600, 8, N, 1.

Thời gian MQTT Time thu được:

**Bảng 5 Bảng thống kê về thời gian trễ của giao thức MQTT**

No	QoS	Tổng số gói	Số gói lỗi	Thời gian trung bình một gói tin (ms)
1	0	100	0	156,63
2	1	100	0	218,31
3	2	100	0	344,01

**Nhận xét:** thời gian trễ cao, việc đặt QoS lên 1 và 2 làm tăng đáng kể độ trễ. Bên cạnh đó, broker trên cloud cũng gây ra độ trễ nhất định.

Thời gian tổng của gói tin đi trong mạng:

**Bảng 6 Bảng thống kê tổng thời gian gói tin đi trong mạng Modbus và MQTT**

No	QoS	Tổng số gói	Số gói lỗi	Thời gian trung bình một gói tin (ms)
1	0	100	0	241,09
2	0	100	0	247,09
3	0	100	0	243,87

## 5.2 Web App

### 5.2.1 Kết quả đạt được

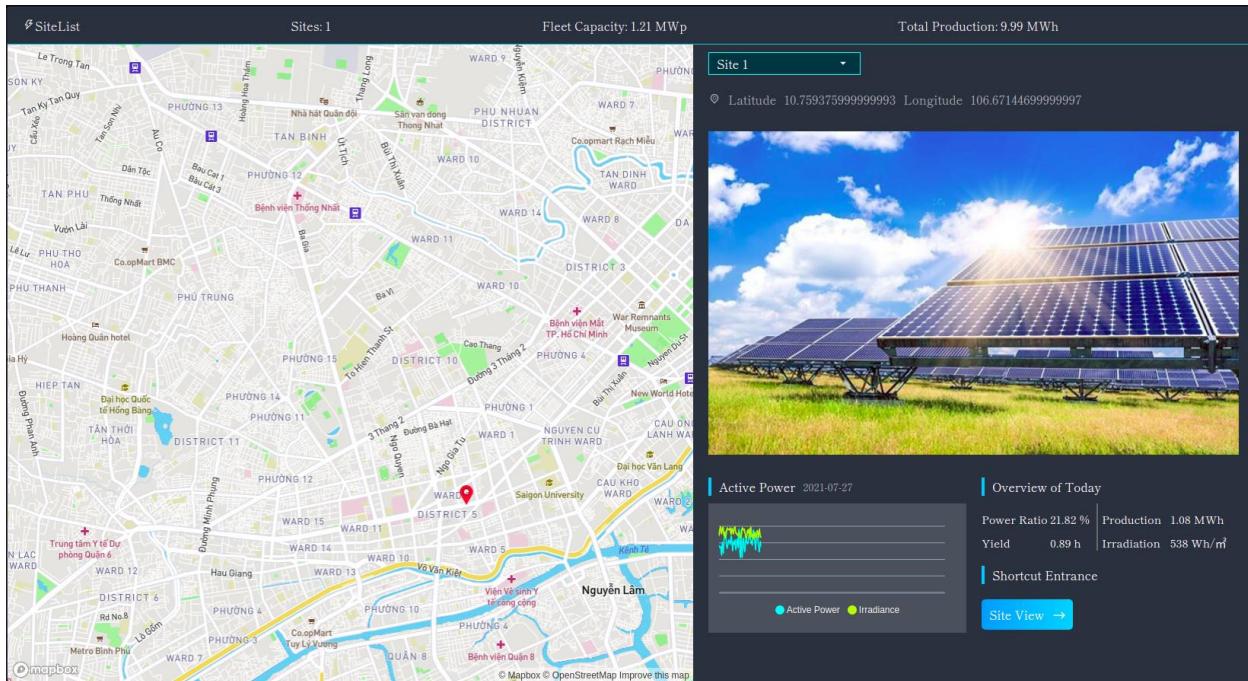
Các kết quả đạt được:

- Xây dựng thành công một ứng dụng web có các chức năng đăng nhập, đăng xuất, tạo các tài khoản với các role admin hoặc user, cho phép quản lý và phân quyền người dùng.
- Xây dựng hoàn chỉnh giao diện các trang giúp trực quan hóa dữ liệu thu thập được từ các hệ thống năng lượng điện mặt trời. Cho phép hiển thị dữ liệu dưới dạng các biểu đồ phân tích, thực hiện một vài tính toán đơn giản, thực hiện các thống kê dữ liệu theo ngày, tháng, năm, cho phép xuất các báo cáo dữ liệu theo yêu cầu.
- Mô hình quản lý hệ thống theo các cấp: Site, gateway, device. Cho phép thực hiện các cấu hình linh hoạt, tuỳ biến, dễ mở rộng, có khả năng quản lý multi-site ( quản lý đồng thời nhiều hệ thống điện mặt trời).
- Kết nối thành công trong việc nhận dữ liệu từ các gateway và thiết bị thực.

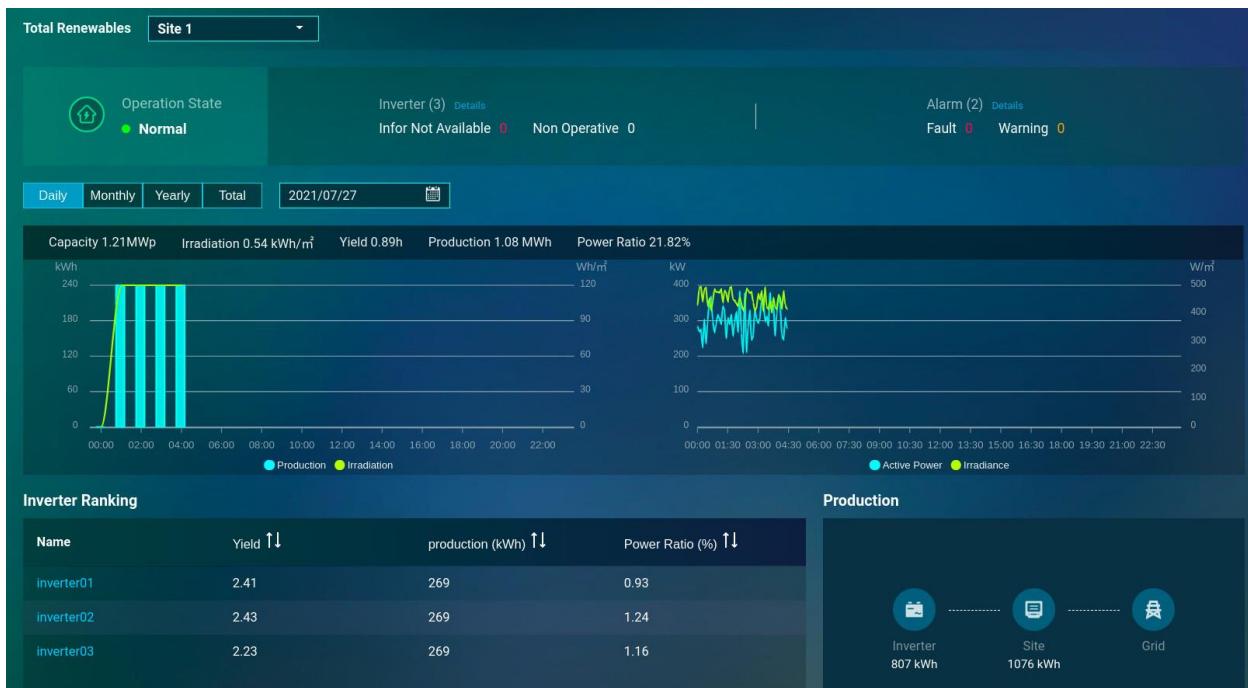
### 5.2.2 Một số hình ảnh của Web App

Hình ảnh sau đây nhận được bằng cách tiến hành mô phỏng hoạt động của một trạm năng lượng điện mặt trời, bao gồm các thiết bị sau:

- 3 Inverter
- 2 Energy Meter
- 1 Weather Station



Hình 5-10 Giao diện trang Fleetview



Hình 5-11 Giao diện trang Site View



Hình 5-12 Giao diện trang Site KPI

**Site 1**

**Inverter**   **Meter**   **Weather Station**

Search for names..

Inverter Name	State	Syst. Diag.	Total Energy ↑	Active Power ↑	Input Power ↑	Efficiency ↑	I
inverter01	Full Capacity	String Normal	2498	104	128	90	4
inverter02	Full Capacity	String Normal	2498	137	107	97	4
inverter03	Full Capacity	String Normal	2498	140	111	91	4

Prev | 1 | Next | 10/Page

Hình 5-13 Giao diện trang Device List

## 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

### 6.1 Kết luận

Nhóm sinh viên tóm tắt những điều rút ra được từ kết quả đề tài, những kinh nghiệm có được sau khi thực hiện đề tài. Sau gần 5 tháng thực hiện đề tài, nhóm sinh viên đã rút ra được những kinh nghiệm quý giá trong quá trình và thiết kế, thực hiện một hệ thống.

- Trong quá trình tìm hiểu lý thuyết, phải xác định rõ lý thuyết cần đọc để tránh lan man trong quá trình tìm hiểu.
- Quá trình phát triển phần mềm tiết kiệm thời gian hơn bởi vì phát triển trên 1 development board đã có sẵn. Hầu hết chương trình phần mềm sử dụng trong đề tài được phát triển dựa trên các ví dụ của nhà cung cấp đưa ra hoặc mã nguồn mở miễn phí trên Internet. Qua đó, sinh viên có thể trau dồi khả năng vận dụng những thành quả của người đi trước để lại trong quá trình phát triển phần mềm để xây dựng sản phẩm cho riêng mình, hay nói cách khác là đặt mình ở vị thế đang đứng trên vai “người khổng lồ”, cần biết tận dụng lợi thế này để tiết kiệm thời gian so với làm toàn bộ từ đầu. Tuy nhiên, việc vận dụng cũng phải dựa trên sự am hiểu lý thuyết.
- Cuối cùng, đây cũng là cơ hội để sinh viên rèn luyện khả năng viết báo cáo, có thể nói là một kỹ năng cần thiết trong tương lai.

#### 6.1.1 Ưu điểm

##### Đối với IoT Gateway:

- Tích hợp thành công các giao thức đã đặt ra vào hệ thống.
- Triển khai thành công giao thức Modbus góp phần làm đơn giản hóa các quá trình quản lý dữ liệu trên mạng có dây nhiều thiết bị cũng như tương tác được với các thiết bị hỗ trợ chuẩn Modbus bên ngoài. Có thể tùy biến số lượng thiết bị Modbus mà gateway quản lý.

##### Đối với ứng dụng Web:

- Xây dựng thành công các chức năng cơ bản cần có của một ứng dụng web: đăng nhập, đăng xuất, tạo tài khoản với các role khác nhau, cho phép quản lý và phân quyền người dùng.
- Xây dựng các giao diện quản lý đa dạng, đầy đủ, đáp ứng khả năng trực quan hoá dữ liệu của một hệ thống năng lượng mặt trời.
- Mô hình quản lý hệ thống theo các cấp: Site, Gateway, Device. Cho phép thực hiện các cấu hình linh hoạt, dễ dàng mở rộng, có khả năng quản lý multi-site.
- Việc lưu trữ dữ và truy cập dữ liệu được thực hiện một cách trọn vẹn.

### 6.1.2 Nhược điểm

Một vài mục tiêu đặt ra của đề tài vẫn chưa được hoàn thiện.

#### Đối với IoT Gateway

- Chưa phát triển board mạch gateway hoàn chỉnh, ảnh hưởng đến các giao thức khi triển khai trên development board (các đường kết nối thông qua bus chưa ổn định).
- Modbus: chỉ triển khai Modbus RTU và các chức năng đọc, ghi cơ bản nhất. Tuy có thể đáp ứng được các yêu cầu trong mạng thực tế nhưng vẫn chưa đủ hỗ trợ toàn bộ các chức năng của giao thức này.
- Việc xử lý lớp giao thức MQTT trên nền tảng vi điều khiển vẫn còn chậm do ảnh hưởng đến tốc độ của các Gateway Modbus.

#### Đối với ứng dụng Web:

- Ứng dụng web được xây dựng theo mô hình request – response, việc hiển thị dữ liệu được cập nhật bằng cách liên tục thực hiện các request đến server theo một interval nhất định. Việc này dẫn đến việc cập nhật dữ liệu của ứng dụng chưa thực sự realtime như các ứng dụng được xây dựng theo các mô hình websocket.
- Việc thực hiện nhiều request đến server, dẫn đến hoạt động của ứng dụng có thể bị chậm khi số lượng truy cập nhiều.
- Ứng dụng web, hiện chỉ thực hiện được các tính toán đơn giản, chưa có khả năng thực hiện các tính toán phức tạp.

- Phần giao diện ứng dụng được xây dựng chưa tương thích với tất cả các thiết bị có độ phân giải màn hình khác nhau.

## 6.2 Hướng phát triển

### 6.2.1 Đối với IoT Gateway

#### Về phần cứng:

- Mở rộng các cổng quản lý thiết bị Modbus của gateway để quản lý được nhiều thiết bị hơn.
- Tích hợp thêm module SIM 4G (triển khai ở dạng LTE-M) để gateway có thể giao tiếp với cloud khi mất kết nối mạng Ethernet bị mất.
- Tiến hành vẽ board mạch hoàn chỉnh với các chức năng đã đề ra và các chức năng mới để hoàn thiện về phần cứng của gateway.

#### Về phần mềm:

- Khắc phục những lỗi phần mềm.
- Phát triển Modbus TCP để gateway có thể giao tiếp với các thiết bị cùng giao thức này.
- Tối ưu bộ nhớ để gateway có thể chứa nhiều thông tin của thiết bị hơn.
- Phát triển các chức năng hỗ trợ chẩn đoán hệ thống tại biên.
- Phát triển thêm các tính năng của ứng dụng Desktop để người dùng dễ quản lý gateway (các cơ chế quản lý thiết bị dưới gateway, giao diện tương thích hơn với người dùng...)

### 6.2.2 Đối với ứng dụng web

- Có thể chuyển đổi ứng dụng hoạt động theo mô hình websocket để việc hiển thị dữ liệu có thể đáp ứng realtime hoàn toàn.
- Xây dựng thêm các khả năng tính toán phức tạp hơn.
- Tiếp tục hoàn thiện thêm phần giao diện, cung cấp khả năng đáp ứng responsive khi truy cập bằng cách thiết bị có độ phân giải màn hình khác nhau.
- Xây dựng thêm các rule engine để có thể cung cấp khả năng thực hiện các alarm.

## 7. TÀI LIỆU THAM KHẢO

[1] solaredge.com, “Solaredge Monitoring Platform,” [Trực tuyến].

Available: <https://www.solaredge.com/products/pv-monitoring#/>

[2] azure.microsoft.com, “Azure for Energy,” [Trực tuyến].

Available: <https://azure.microsoft.com/en-us/industries/energy/>

[3] ats.com.vn, “Solar Plant Control and Monitoring Solution,” [Trực tuyến]

Available: <https://ats.com.vn/solar-plant-control-monitoring/>

[4] mese.vn, “Giải pháp IoT cho hệ thống quản lý năng lượng mặt trời,” [Trực tuyến]

Available: <https://mese.vn/vi/author/admin>

[5] "Modbus," Modicon, 16 July 2021. [Trực tuyến].

Available: <https://en.wikipedia.org/wiki/Modbus/>

[6] tapit.vn, “Tìm hiểu về khung truyền Modbus RTU,” [Trực tuyến].

Available: <https://tapit.vn/khung-truyen-modbus-rtu/>

[7] “mqtt.org,” [Trực tuyến]. Available: <https://mqtt.org/>

[8] tapit.vn, “Tìm hiểu về định dạng một số gói tin MQTT,” [Trực tuyến].

Available: <https://tapit.vn/tim-hieu-dinh-dang-mot-so-goi-tin-mqtt/>

[9] “FreeRTOS™ Real-time operating system for microcontrollers,” Real Time Engineers

Ltd, 2021. [Trực tuyến]. Available: <https://www.freertos.org/>

[10] bizfly.vn, “Thiết kế web app và 12 bước thiết kế Web application tối ưu cho doanh nghiệp”, [Trực tuyến]

Available: <https://bizfly.vn/techblog/thiet-ke-web-app.html>

[11] tuhocict.com, “Mô hình hoạt động của ứng dụng web”, [Trực tuyến]

Available: <https://tuhocict.com/ung-dung-web-va-php/>

[12] VNTALKING, “Lập trình React Thật Đơn Giản”.

[13] freeTuts.net, “NodeJS là gì? Đặc tính và các framework NodeJS phổ biến”, [Trực tuyến]

Available: <https://freetuts.net/nodejs-la-gi-584.html>

[14] viblo.asia, “Tìm hiểu về MongoDB”, [Trực tuyến]

Available: <https://viblo.asia/p/tim-hieu-ve-mongodb-4P856ajGIY3>

[15] vntalking.com, “Node.js + MongoDB: Authentication và Authorization sử dụng JWT”, [Trực tuyến]

Available: <https://vntalking.com/node-authentication-va-authorization-su-dung-jwt.html>

[16] solar.huawei.com, “Smart Energy Controller SUN2000-2/3/4/5KTL-L1,” [Trực tuyến]

Available: <https://solar.huawei.com/hi-IN/download?p=%2Fmedia%2FSolar%2Fattachment%2Fpdf%2Fapac%2Fdatasheet%2FSUN2000-2-5KTL-L1.pdf>

[17] se.com, “PM5350 Power and Energy meter,” [Trực tuyến]

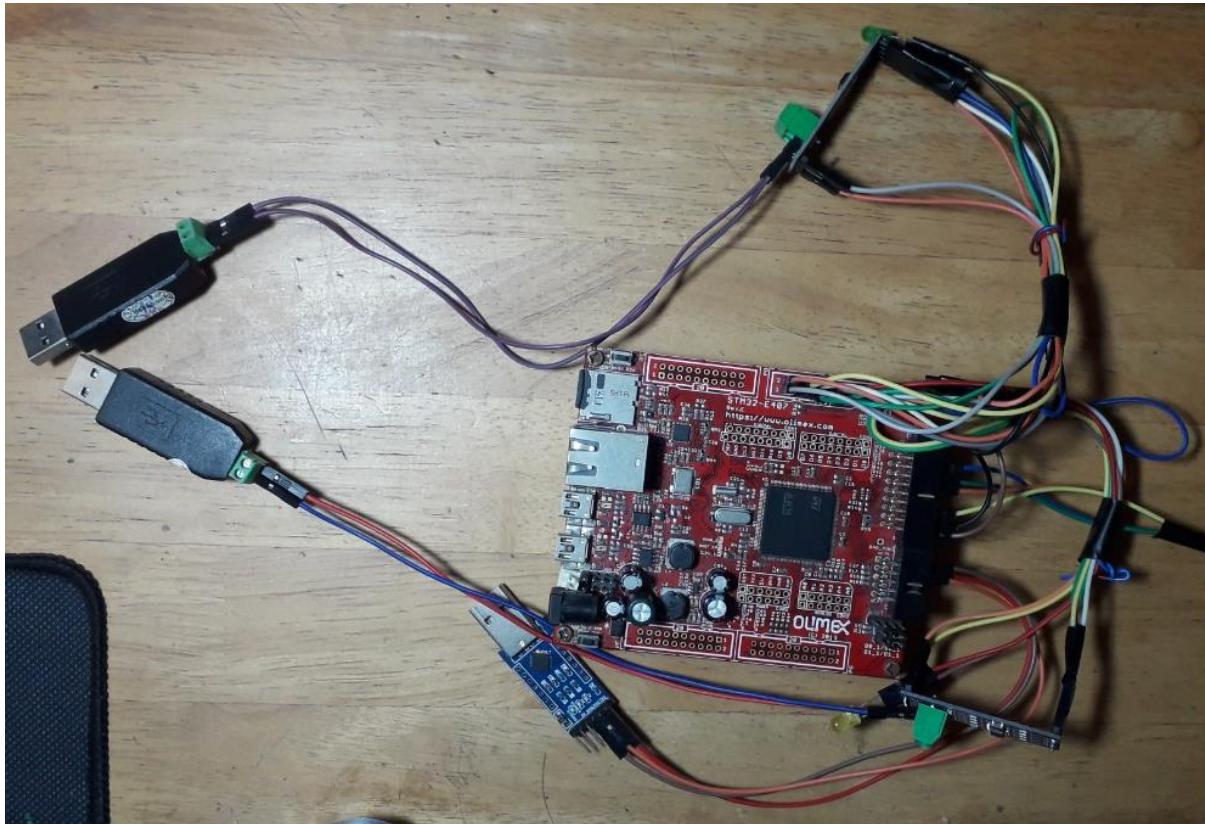
Available: <https://www.se.com/vn/vi/product/METSEPM5350/pm5350-power-%26-energy-meter-with-thd%2C-alarming/>

[18] new.abb.com, “VSN800-14 weather station,” [Trực tuyến]

Available: <https://new.abb.com/>

## 8. PHỤ LỤC

### 8.1 Mô hình demo hệ thống



Hình 8-1 Các kết nối trên gateway sau khi hoàn thành