TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH

KHOA ĐIỆN – ĐIỆN TỬ

BỘ MÔN ĐIỀU KHIỂN TỰ ĐỘNG

NGUYỄN TRƯỜNG GIANG

ĐOÀN TRỌNG ĐỨC DUY

LUẬN VĂN TỐT NGHIỆP

XÂY DỰNG GIẢI PHÁP ĐIỀU KHIỂN VÀ GIÁM SÁT TỪ XA TÍCH HỢP GIAO THỨC OPC UA

KỸ SƯ NGÀNH KỸ THUẬT ĐIỀU KHIỂN & TỰ ĐỘNG HÓA

TP. HỒ CHÍ MINH, 2022

TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH

KHOA ĐIỆN – ĐIỆN TỬ

BỘ MÔN ĐIỀU KHIỂN TỰ ĐỘNG

NGUYỄN TRƯỜNG GIANG - 1711119

ĐOÀN TRỌNG ĐỨC DUY - 1810858

LUẬN VĂN TỐT NGHIỆP

XÂY DỰNG GIẢI PHÁP ĐIỀU KHIỂN VÀ GIÁM SÁT TỪ XA TÍCH HỢP GIAO THỨC OPC UA

BUILDING A REMOTE MONITORING AND CONTROL SOLUTION INTEGRATING WITH OPC UA

KỸ SƯ NGÀNH KỸ THUẬT ĐIỀU KHIỂN & TỰ ĐỘNG HÓA

GIẢNG VIÊN HƯỚNG DẪN

ThS. NGUYỄN ĐỨC HOÀNG

TP. HỒ CHÍ MINH, 2022

TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH

KHOA ĐIỆN – ĐIỆN TỬ

BỘ MÔN: ĐIỀU KHIỂN TỰ ĐỘNG

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

TP. HCM, ngày 20 tháng 5 năm 2022

NHẬN XÉT LUẬN VĂN TỐT NGHIỆP

CỦA CÁN BỘ HƯỚNG DẪN

Tên luận văn:

XÂY DỰNG GIẢI PHÁP ĐIỀU KHIỂN VÀ GIÁM SÁT TỪ XA TÍCH HỢP GIAO THỨC OPC UA

Nhóm Sinh viên thực hiện:		**Cán bộ hướng dẫn:**
Nguyễn Trường Giang	1711119	ThS. Nguyễn Đức Hoàng
Đoàn Trọng Đức Duy	1810858	ThS. Nguyễn Đức Hoàng

Đánh giá Luận văn

Về cuốn báo cáo:

Số trang	Số chương
Số bảng số liệu	Số hình vẽ
Số tài liệu tham khảo	Sản phẩm

Một số nhận xét về hình thức cuốn báo cáo:

Về nội dung luận văn:

Về tính ứng dụng:

Về thái độ làm việc của sinh viên:

<u>Đánh giá chung:</u> Luận văn đạt/không đạt yêu cầu của một luận văn tốt nghiệp kỹ sư, xếp loại Giỏi/ Khá/ Trung bình

<u>Điểm từng sinh viên:</u>

Nguyễn Trường Giang**:……/10**

Đoàn Trọng Đức Duy**:……/10**

Cán bộ hướng dẫn
(Ký tên và ghi rõ họ tên)

TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH

KHOA ĐIỆN – ĐIỆN TỬ

BỘ MÔN: ĐIỀU KHIỂN TỰ ĐỘNG

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

TP. HCM, ngày 20 tháng 5 năm 2022

NHẬN XÉT LUẬN VĂN TỐT NGHIỆP

CỦA CÁN BỘ PHẢN BIỆN

Tên luận văn:

XÂY DỰNG GIẢI PHÁP ĐIỀU KHIỂN VÀ GIÁM SÁT TỪ XA TÍCH HỢP GIAO THỨC OPC UA

Nhóm Sinh viên thực hiện: **Cán bộ hướng dẫn:**

Nguyễn Trường Giang 1711119

Đoàn Trọng Đức Duy 1810858

Đánh giá Luận văn

Về cuốn báo cáo:

Số trang Số chương

Số bảng số liệu Số hình vẽ

Số tài liệu tham khảo Sản phẩm

Một số nhận xét về hình thức cuốn báo cáo:

Về nội dung luận văn:

Về tính ứng dụng:

Về thái độ làm việc của sinh viên:

<u>Đánh giá chung:</u> Luận văn đạt/không đạt yêu cầu của một luận văn tốt nghiệp kỹ sư, xếp loại Giỏi/ Khá/ Trung bình

<u>Điểm từng sinh viên:</u>

Nguyễn Trường Giang**:……../10**

Đoàn Trọng Đức Duy**:……../10**

Cán bộ hướng dẫn
(Ký tên và ghi rõ họ tên)

TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH

KHOA ĐIỆN – ĐIỆN TỬ

BỘ MÔN: ĐIỀU KHIỂN TỰ ĐỘNG

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

TP. HCM, ngày 20 tháng 5 năm 2022

ĐỀ CƯƠNG CHI TIẾT

TÊN LUẬN VĂN: XÂY DỰNG GIẢI PHÁP ĐIỀU KHIỂN VÀ GIÁM SÁT TỪ XA TÍCH HỢP GIAO THỨC OPC UA
Cán bộ hướng dẫn:
Thời gian thực hiện: Từ ngày 03/01/2022 đến ngày 20/05/2022
Sinh viên thực hiện: **Nguyễn Trường Giang - 1711119** **Đoàn Trọng Đức Duy - 1810858**
Nội dung đề tài: ***Mục tiêu:*** *Xây dựng gateway IIoT và ứng dụng SCADA hỗ trợ giao thức OCP UA* ***Đối tượng****: Lập trình SCADA hỗ trợ giao thức OPC UA.* ***Phương pháp thực hiện:*** *Xây dựng OPC UA Server trên gateway là một máy tính nhúng Raspberry Pi để thu thập dữ liệu các trạm.* *Xây dựng OPC UA Client trên VPS để thu thập dữ liệu các trạm từ xa.* *Lập trình giao diện SCADA và các tính năng cơ bản của SCADA trên web và Android* *Thiết kế một trạm mô phỏng bằng các cảm biến nhiệt độ, độ ẩm, áp suất không khí kết nối tới Raspberry Pi*

Kết quả mong đợi: *Chương trình SCADA có thể giám sát và điều khiển được các trạm quan trắc qua giao thức OPC UA.*	
Kế hoạch thực hiện: *Nghiên cứu lý thuyết về giao thức OPC UA.* *Tìm hiểu giải các giải pháp để thực hiện dự án và các thư viện liên quan.* *Lập trình ứng dụng SCADA trên web và điện thoại di động dùng cơ sở dữ liệu MongoDB để lưu trữ dữ liệu* *Lập trình OPC UA Server trên Raspberry Pi và OPC UA Client trên VPS.* *Tìm hiểu lý thuyết và cách xây dựng một mạng riêng ảo VPN.* *Phát triển ứng dụng SCADA trên web bằng ngôn ngữ JavaScript, HTML và trên Android bằng ngôn ngữ Java.* *Mô phỏng dữ liệu các trạm trên OPC UA Server bằng hàm random.* *Xây dựng một trạm thực bằng các cảm biến nhiệt độ, áp suất, độ ẩm.*	
Xác nhận của Cán bộ hướng dẫn *(Ký tên và ghi rõ họ tên)*	TP. HCM, ngày 20 tháng 5 năm 2022 **Sinh viên** *(Ký tên và ghi rõ họ tên)*

DANH SÁCH HỘI ĐỒNG BẢO VỆ LUẬN VĂN

Hội đồng chấm luận văn tốt nghiệp, thành lập theo Quyết định số …… ngày …….. của Hiệu trưởng Trường Đại học Bách khoa TP.HCM.

1. …….. – Chủ tịch.
2. …….. – Thư ký.
3. …….. – Ủy viên.
4. …….. – Ủy viên.
5. …….. – Ủy viên.

LỜI CẢM ƠN

Lời đầu tiên, em xin gửi lời cảm ơn chân thành nhất đến thầy Nguyễn Đức Hoàng – giảng dạy tại Bộ môn Tự Động Hóa thuộc khoa Điện – Điện tử của Trường Đại Học Bách Khoa TP.HCM. Trong quá trình học tập và tìm hiểu Đồ án, em đã nhận được sự quan tâm giúp đỡ, hướng dẫn cách tận tình, tâm huyết của thầy. Thầy đã giúp em tích lũy thêm nhiều kiến thức chuyên sâu về chủ đề mà tụi em đang tìm hiểu. Thông qua bài báo cáo này, chúng em xin trình bày những gì mà chúng em đã tìm hiểu được và những kiến thức thầy truyền tải để gửi đến thầy.

Có lẽ kiến thức là vô hạn mà sự tiếp nhận kiến thức của mỗi người lại là hữu hạn. Do đó, trong quá trình hoàn thiện bài báo cáo, chắc chắn không tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự góp ý đến từ thầy để bài báo cáo của em được hoàn thiện hơn.

Chúng em xin chân thành cảm ơn.

MỤC LỤC

Chương 1. GIỚI THIỆU VỀ ĐỀ TÀI 2
1.1. Đặt vấn đề 2
1.2. Mục tiêu đề tài 3
1.3. Nhiệm vụ đề tài 3
1.4. Kiến trúc đề tài 5
Chương 2. LÝ THUYẾT VÀ CÁC PHƯƠNG PHÁP THỰC HIỆN 6
2.1. OPC UA 6
2.1.1. Tổng quan về giao thức OPC Classic 6
2.1.2. Tổng quan về giao thức OPC UA 8
2.1.3. Ưu điểm của OPC UA 11
2.1.4. Thư viện open62541 13
2.2. Mạng riêng ảo 14
2.2.1. Giới thiệu 14
2.2.2. Cách hoạt động 15
2.2.3. Lợi ích sử dụng VPN 16
2.2.4. Wireguard 17
2.3. Máy chủ ảo 19
2.3.1. Giới thiệu 19
2.3.2. Ưu điểm của VPS 20
2.3.3. Ứng dụng của VPS 20
Chương 3. THỰC HIỆN 21

3.1. Xây dựng gateway 21
3.1.1. Xây dựng OPC UA Server 21
3.1.2. Xây dựng VPN Client 27
3.2. Khởi tạo và phát triển ứng dụng trên VPS 28
3.2.1. Tổng quan kiến trúc hệ thống trên VPS 28
3.2.2. Xây dựng Wireguard server 29
3.2.3. Xây dựng OPC UA Client trên VPS 31
3.3. Xây dựng web server 36
3.3.1. Tổng quan 36
3.3.2. Các công cụ và giao thức sử dụng 37
3.3.3. Giao diện web 40
3.3.4. Xây dựng backend 44
3.4. Xây dựng ứng dụng Android 51
3.4.1. Kiến trúc phần mềm 51
3.4.2. Giao diện danh sách trạm 53
3.4.4. Giao diện trend dữ liệu lịch sử 64
3.4.5. Giao diện alarm thời gian thực 70
3.4.6. Giao diện thông tin alarm 74
Chương 4: ĐÁNH GIÁ KẾT QUẢ 78
4.1. Kết quả đạt được: 78
4.2. Các điểm thiếu sót: 78
4.3. Hướng phát triển của luận văn: 79

DANH MỤC HÌNH VẼ

Hình 1.1: Kiến trúc tổng quan của đề tài 5

Hình 2.1: Trao đổi dữ liệu khi chưa có OPC 6

Hình 2.2: Trao đổi dữ liệu dùng OPC 7

Hình 2.3: Cấu trúc điển hình của một hệ thống sử dụng OPC Classic 7

Hình 2.4: Cấu trúc các mức độ tự động ho 10

Hình 2.5: Mô hình hoạt động của giao thức OPC UA 12

Hình 2.6: Mô hình tổng quan VPN 14

Hình 2.7: Cách hoạt động của trao đổi khoá công khai 15

Hình 2.8: So sánh VPS hosting với dedicated và shared hosting 19

Hình 3.1: Cấu trúc thư mục OPC UA Server 21

Hình 3.2: Information model của trạm khí thải 22

Hình 3.3: Information model của trạm nước thải 23

Hình 3.4: Information model của trạm không khí 23

Hình 3.5: Information model của trạm nước ngầm 24

Hình 3.6: Tổng quan kiến trúc hệ thống trên VPS 28

Hình 3.7: Mở port trên VPS 30

Hình 3.8: Tổng quan về mô hình ứng dụng web 36

Hình 3.9: Cơ chế blocking I/O (trái) và non-blocking I/O (phải) 37

Hình 3.10: Mô hình MVC trong lập trình web 39

Hình 3.11: Giao diện login trên web 41

Hình 3.12: Giao diện màn hình Home trên web 41

Hình 3.13: Giao diện Trend khi chưa kết nối với OPC Server 42

Hình 3.14: Giao diện màn hình Trend trên web 43

Hình 3.15: Giao diện Alarm trên web 43

Hình 3.16: Sử dụng phương thức POST để submit 45

Hình 3.17: Submit biểu mẫu chứa username và password bằng jQuery 45

Hình 3.18: Sử dụng Google Maps API 47

Hình 3.19: Kiến trúc phần mềm giám sát trên Android 51

Hình 3.20: Giao diện đăng nhập trên Android 52

Hình 3.21: Giao diện danh sách các trạm trên Android 53

Hình 3.22: Code logic cho giao diện con của ViewPager2 60

Hình 3.23: Giao diện giám sát dữ liệu thời gian thực trên Android 61

Hình 3.24: Giao diện mobile trend khi có kết nối 64

Hình 3.25: Giao diện mobile trend khi bị mất kết nối 65

Hình 3.26: Giao diện danh sách alarm trên Android 70

Hình 3.27: Giao diện thông tin alarm trên Android 75

DANH MỤC TỪ VIẾT TẮT

API	Application Programming Interface
COM	Component Object Model
CSS	Cascading Style Sheets
DNP3	Distributed Network Protocol 3
GUI	Graphic User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IT	Information Technology
OPC	Open Platform Communication
OPC UA	OPC Unified Architecture
OT	Operational Technology
SCADA	Supervisory Control and Data Acquisition
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
VPN	Virtual Private Network
VPS	Virtual Private Server

TÓM TẮT LUẬN VĂN

Hệ thống SCADA đóng vai trò quan trọng trong việc giám sát và vận hành hệ thống tự động từ xa. Đặc biệt trong thời đại công nghiệp 4.0, thời đại vạn vật kết nối với nhau, việc điều khiển và giám sát không chỉ còn thực hiện tại nhà máy mà người vận hành có thể thực hiện được việc này tại bất kỳ đâu trên trái đất. Đó là lý do giao thức OPC UA ra đời.

Trong luận văn này, OPC UA Server sẽ được xây dựng trên một máy tính nhúng Raspberry Pi 4 (Gateway) để thu thập dữ liệu và điều khiển các cảm biến và các thiết bị trường trong trạm quan trắc môi trường. Các trạm quan trắc này thường cách xa nhau về mặt địa lý.

Sau đó, OPC UA Client được xây dựng trên một VPS để kết nối với các gateway từ xa thông qua mạng riêng ảo VPN sử dụng giao thức OPC UA.

Dữ liệu thu thập được sẽ tập trung trên VPS, từ đó quản trị viên sử dụng các ứng dụng web hay mobile có thể trực tiếp vận hành nhà máy từ xa một cách nhanh chóng và xử lý kịp thời các sự cố mà không cần có mặt tại nhà máy.

Phần mềm SCADA được lập trình trên cả nền tảng web và hệ điều hành Android với một số tính năng cơ bản bao gồm: vẽ trend, alarm, report excel.

Chương 1. GIỚI THIỆU VỀ ĐỀ TÀI

1.1. Đặt vấn đề

Trong thời đại công nghiệp hoá, hiện đại hoá phát triển mạnh mẽ hơn bao giờ hết, các phần mềm điều khiển giám sát và thu thập dữ liệu (SCADA) là không thể thể thiếu trong các quy trình sản xuất, các nhà máy, trong các lĩnh vực năng lượng, lĩnh vực quan trắc môi trường, … Các phần mềm SCADA hỗ trợ con người trong việc giám sát và điều khiển từ xa.

Bên cạnh đó, sự phát triển mạnh mẽ của khoa học, công nghệ đã dẫn đến việc ô nhiễm môi trường, hiệu ứng nhà kính ngày càng nghiêm trọng. Các nhà máy vận hành liên tục 24/7, các phương tiện sử dụng động cơ đốt trong đang lưu thông trên đường, … Những nguyên nhân trên đã làm ô nhiễm nguồn không khí, nguồn nước mặt, nước ngầm. Do đó, việc quan trắc các chỉ số môi trường trở nên quan trọng vì sự ô nhiễm môi trường ảnh hưởng trực tiếp đến sức khoẻ con người. Nhưng các trạm quan trắc thường cách xa nhau về mặt địa lý, dữ liệu bị bị phân tán đã gây khó khăn trong việc giám sát và thu thập dữ liệu.

Mặt khác, các phần mềm SCADA chính hãng thường có giá thành rất cao và thường không được sử dụng hết tính năng của nó. Thêm vào đó, các phần mềm này chỉ có thể được vận hành trên máy trạm trung tâm đặt tại một nơi cố định nào đó trong nhà máy và chỉ có thể được cài đặt trên hệ điều hành Window. Mà việc giám sát, thu thập dữ liệu và điều khiển từ xa một cách nhanh chóng, theo thời gian thực có thể mang lại hiệu suất kinh tế cho doanh nghiệp rất cao.

Vậy nên, việc phát triển một ứng dụng SCADA có thể tuỳ biến riêng và chạy được trên mọi thiết bị, chẳng hạn: laptop, điện thoại di động, máy tính bảng, … là rất cần thiết trong thời đại IoT. Trong luận văn này, nhóm sẽ phát triển một phần mềm SCADA có thể truy cập từ website và lẫn điện thoại di động chạy hệ điều hành Android. Ứng dụng giám sát này sử dụng giao thức OPC UA để thu

thập dữ liệu từ các trạm quan trắc và sử dụng được các tính năng cơ bản một cách hiệu quả cho hệ thống.

1.2. Mục tiêu đề tài

Xây dựng một OPC UA Server trên một máy tính nhúng để từ đó có thể thu thập dữ liệu từ các thiết bị trường không hỗ trợ giao thức OPC UA.

Tìm hiểu các thông số môi trường của các trạm quan trắc để có thể xây dựng một ứng dụng có thể đáp ứng yêu cầu của hệ thống quan trắc.

Nghiên cứu về các giải pháp để bảo mật thông tin từ OPC UA Server cho đến ứng dụng người dùng.

1.3. Nhiệm vụ đề tài

Tìm hiểu về hệ điều hành Linux.

Tìm hiểu về giao thức OPC UA, bao gồm OPC UA Client, OPC UA Server, Information Model.

Xây dựng Information Model cũng như mô phỏng dữ liệu cảm biến của từng loại trạm quan trắc.

Nghiên cứu lý thuyết về VPN cũng như cách hoạt động của nó. Từ đó xây dựng một VPN riêng tên là Wireguard để kết nối giữ OPC Server và OPC Client.

Tạo một VPS chạy hệ điều hành Linux trên Azure. Sau đó, sẽ xây dựng VPN Server, Web server cũng như là một OPC Client để thu thập dữ liệu từ xa của các trạm quan trắc.

Tìm hiểu về thư viện node-opcua, ngôn ngữ lập trình JavaScript lẫn các kiến thức có liên quan khác để xây dựng một NodeJS server chạy trên VPS.

Tạo một cluster trên cơ sở dữ liệu không quan hệ MongoDB dùng để lưu trữ thông tin người dùng và dữ liệu của các trạm.

Xây dựng các REST API để xử lý tác vụ đăng nhập, ký từ người dùng.

Sử dụng ngôn ngữ HTML và CSS để xây dựng giao diện website với các tính vẽ Trend, Alarm.

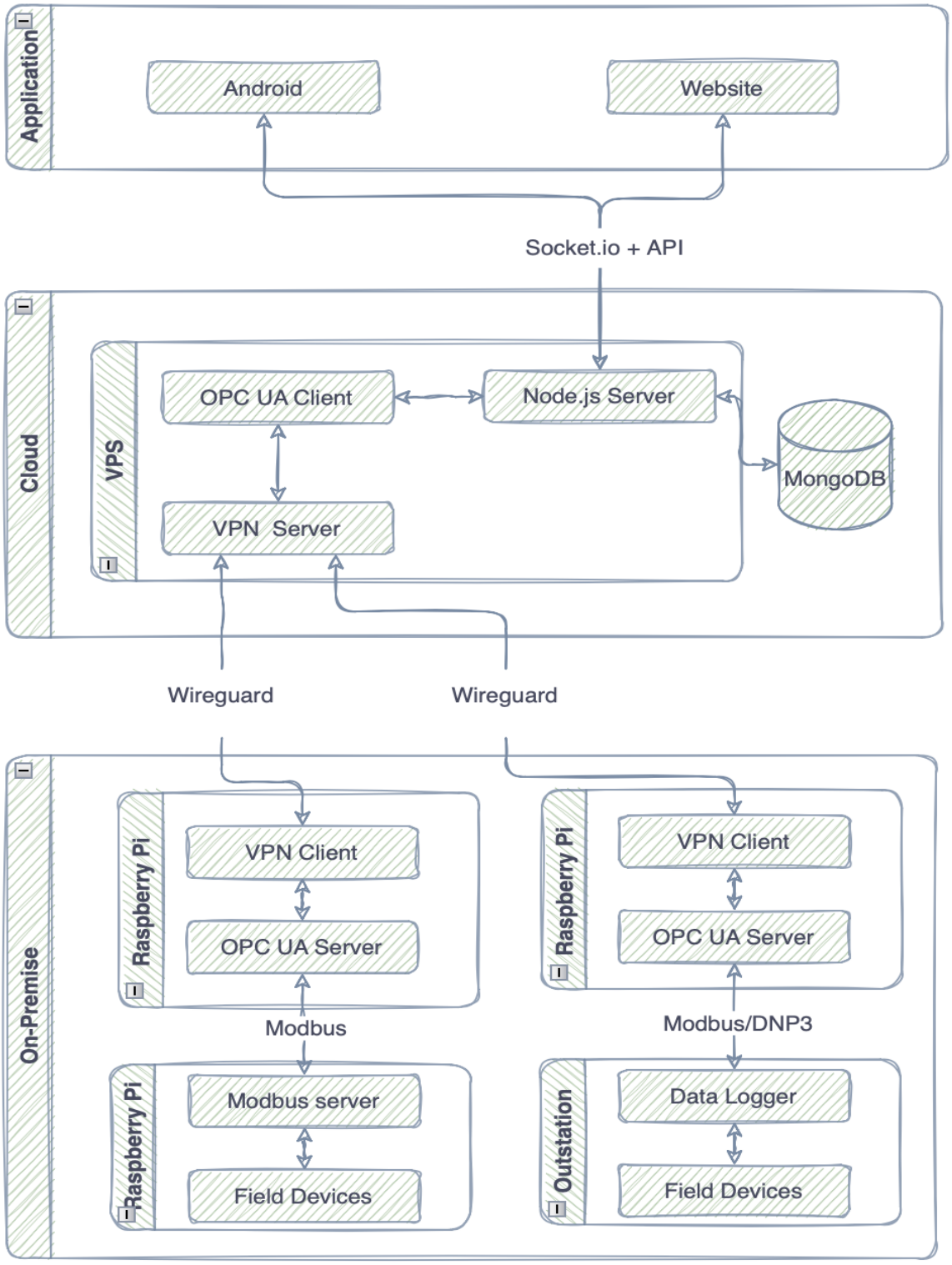
Xây dựng ứng dụng cho hệ điều hành Android bằng ngôn ngữ Java.

Mô phỏng các trạm quan trắc từ xa và kết nối các trạm đến server chạy trên VPS.

Thực hiện việc quan sát và điều khiển từ xa trên ứng dụng điện thoại và giao diện web.

Đánh giá kết quả thực hiện.

1.4. Kiến trúc đề tài



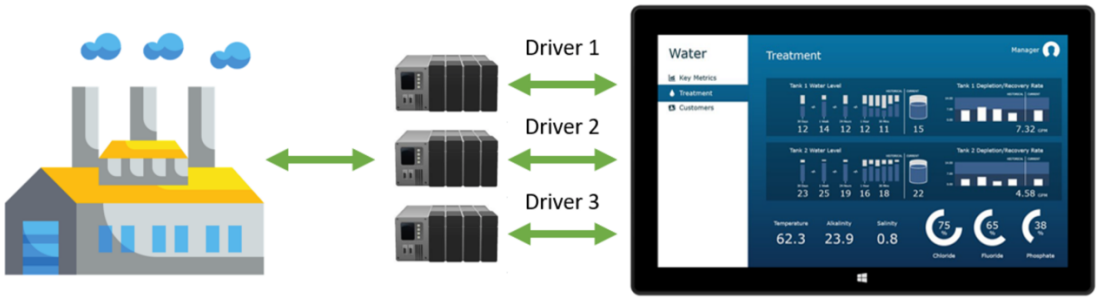
Hình 1.1: Kiến trúc tổng quan của đề tài

Chương 2. LÝ THUYẾT VÀ CÁC PHƯƠNG PHÁP THỰC HIỆN

2.1. OPC UA

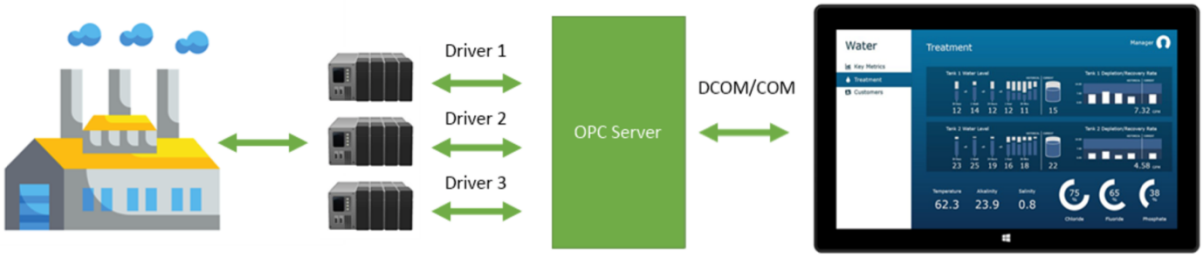
2.1.1. Tổng quan về giao thức OPC Classic

OPC (OLE cho Process Control) là một tiêu chuẩn truyền thông công nghiệp trong điều khiển quá trình, dựa trên công nghệ COM của Microsoft. Trong khoảng những năm 1990, khi cần tích hợp một thiết bị công nghiệp vào giao diện người dùng đồ hoạ (GUI) thì trước tiên ta cần tìm ra giao thức để có thể giao tiếp được. Điều này gây không ít khó khăn cho người lập trình HMI/SCADA do mỗi hãng sản xuất thường sử dụng những giao thức khác nhau.



Hình 2.1: Trao đổi dữ liệu khi chưa có OPC

Vì các phần mềm giám sát có giao diện điều khiển hầu như đều được phát triển trong môi trường Windows, nên người ta đã quyết định đơn giản hóa bằng cách giới thiệu một nền tảng giao tiếp mới, được gọi là OPC. Nó lần đầu tiên được phát hành vào năm 1996 bởi hiệp hội OPC Foundation. OPC là cụm từ được viết tắt của OLE for Process Control. Trong đó OLE được viết tắt bởi Object Linking and Embedding, là công nghệ cho phép nhúng và liên kết các tài liệu và các đối tượng khác nhau, dựa trên Công nghệ COM (Component Object Model) và DCOM (Distributed Component Object Model) của MicroSoft. Tóm lại, ưu điểm lớn nhất của OPC là giảm bớt nỗ lực kết nối giữa Controller với SCADA/HMI. Vì trong một hệ thống sản xuất, việc phải sử dụng nhiều loại Controller đến từ các hãng khác nhau là rất phổ biến.



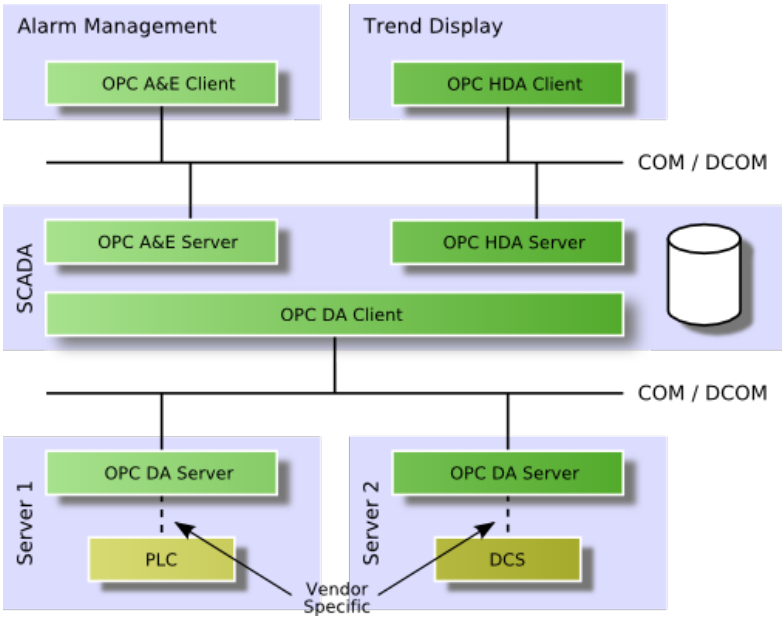
Hình 2.2: Trao đổi dữ liệu dùng OPC

Dữ liệu được giao tiếp giữa Client và Server trong OPC Classic được thực hiện theo những đặc tả khác nhau, tuỳ thuộc vào mục đích sử dụng, bao gồm 3 đặc tả chính:

- OPC DA (Data Access): cho phép Client đọc, ghi và hiển thị dữ liệu theo thời gian thực.

- OPC HDA (Historical Data Access): được sử dụng để tiếp cận những dữ liệu đã được lưu trữ.

- OPC A&E (Alarms and Events): quản lý những thông báo về Alarm và Event.



Hình 2.3: Cấu trúc điển hình của một hệ thống sử dụng OPC Classic

- **Nhược điểm của OPC Classic:**

Hai nhược điểm chính của OPC Classic là sự phụ thuộc vào nền tảng Windows và các vấn đề liên quan đến DCOM khi sử dụng giao tiếp từ xa giữa máy khách và máy chủ OPC. DCOM rất khó để cấu hình, có thời gian chờ rất lâu và không thể định cấu hình, cũng như không thể được sử dụng cho giao tiếp internet vì các vấn đề với tường lửa (firewall) của Windows.

+ Trong các nhà máy, OPC DA Server và SCADA thường chạy trên các máy tính Windows khác nhau và cũng ở các khu vực khác nhau, cách xa nhau. Các máy tính này bắt buộc phải sử dụng cùng một phiên bản Windows. Để truy cập vào Server, trên 2 máy tính phải đăng nhập cùng một username và password, phải dùng DCOM config để cấu hình một loạt các thông số không tường minh.

+ OPC DA chỉ chạy được trên hệ điều hành Windows và cũng không chạy được trên .Net, một công nghệ hiện đại hơn của Microsoft ra đời để thay thế COM. Chính điều này khiến việc nhúng OPC DA chạy trực tiếp trên PLC là một rào cản cực kỳ lớn.

+ Ngoài ra, OPC DA trao đổi dữ liệu bằng phương pháp truyền thống là "polling" (hỏi vòng), với một tần số lấy mẫu, client liên tục thực hiện việc gửi request tới server và chờ phản hồi, nếu dữ liệu không thực sự thay đổi đáng kể, điều này dẫn đến việc chiếm dụng phần lớn băng thông mạng.

2.1.2. Tổng quan về giao thức OPC UA

Vào năm 2008, phiên bản kế nhiệm của OPC Classic được ra đời, mang tên là OPC UA. OPC UA hoàn toàn khác so với đàn anh của mình là OPC DA. Chính vì vậy, OPC UA được hiểu bằng cụm từ Open Platform Communication (OPC) Unified Architecture (UA) nhằm tránh đi cụm từ đã không hợp thời OLE for Process Control của OPC Classic.

Sự ra đời của OPC UA đã khắc phục được tất cả các khuyết điểm của OPC Classic đã được đề cập ở trên. Tiêu biểu là:

- Đa nền tảng không ràng buộc với bất cứ hệ điều hành hay ngôn ngữ lập trình nào (Windows, Linux, Mac, Android,...).
- Là một giao thức tập trung mạnh vào kiến trúc object-oriented và service-oriented. Nó làm cho những mẫu dữ liệu thô khó hiểu trở nên giàu ngữ nghĩa hơn (meaningful and semantics data).
- Bao gồm luôn Historical Data Access, Alarms & Event. Trước đây các đặc tả này tồn tại riêng biệt: OPC HDA, OPC A&E.
- Có tính bảo mật cao, sử dụng nhiều lớp bảo mật: xác thực người dùng, ủy quyền, xác thực chứng chỉ, ký tên và mã hóa cho dữ liệu.
- Khả năng mở rộng mô hình thông tin dễ dàng.

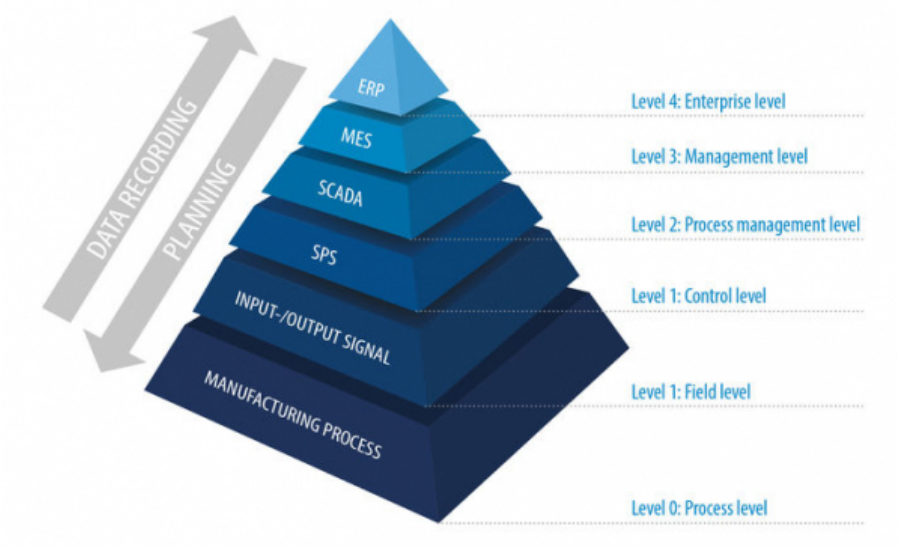
Có thể nói OPC UA dựa trên 3 trụ cột cơ bản: cơ chế truyền tải dữ liệu, gọi các phương thức từ xa và mô hình hóa mô tả dữ liệu.

Cơ chế truyền tải dữ liệu: OPC UA hỗ trợ cả giao thức nhị phân dựa trên TCP đảm bảo hiệu suất cao trong quá trình truyền và hai giao thức truyền thông web được sử dụng rộng rãi như HTTPS và SOAP.

Gọi các phương thức từ xa: đây là một công cụ đáng kinh ngạc để mở rộng tiềm năng của các thiết bị như PLC. Như chúng ta đã biết, PLC là bộ điều khiển có thể lập trình được sử dụng rộng rãi trong tự động hóa công nghiệp vì độ tin cậy và ổn định của nó. Mặt khác, các hệ thống PLC rất thô sơ và không cung cấp các khả năng tương tự như PC có thể cung cấp. Để rõ hơn hãy lấy một ví dụ, nếu chúng ta muốn PLC ghi vào cơ sở dữ liệu, chúng ta sẽ phải sử dụng PC làm trung gian và sau đó thực hiện các phương thức phức tạp để cho phép giao tiếp giữa PLC và PC. Nhưng OPC UA có thể mở rộng các tính năng của PLC, giúp việc gọi các phương thức từ xa trở nên đơn giản và tức thì. Do đó, PC có OPC UA Server có thể đưa ra nhiều phương thức (method call) phức tạp hơn, chẳng hạn như các chức năng đọc/ghi cơ sở dữ liệu, có thể được PLC kích hoạt khi cần thiết.

Mô hình hóa dữ liệu: OPC UA cung cấp một hệ thống mô hình hóa hoàn chỉnh cho phép mô tả dữ liệu của những biến có độ phức tạp cao. Đây chắc chắn là điểm mạnh quan trọng nhất của toàn bộ nền tảng giao tiếp vì nó có thể tạo ra các hệ thống trong đó việc trao đổi dữ liệu diễn ra với một cơ sở hạ tầng hoàn toàn phân tán.

Để hiểu rõ hơn ý nghĩa của OPC UA, chúng ta cần tìm hiểu khái niệm về kim tự tháp tự động hóa. Kim tự tháp tự động hóa là một cấu trúc mô tả các mức độ tự động hóa khác nhau trong một nhà máy sản xuất và là một ví dụ tuyệt vời về cách công nghệ được diễn giải trong ngành công nghiệp.



Hình 2.4: Cấu trúc các mức độ tự động ho

Tại đáy của kim tự tháp, chúng ta có cấp thiết bị trường. Điều này đại diện cho các thiết bị, cơ cấu chấp hành và cảm biến được lắp đặt trong dây chuyền sản xuất. Dữ liệu ở tầng này là "dữ liệu thô" (raw data), chúng được thu thập và đưa lên tầng phía trên.

Tầng tiếp theo được gọi là cấp điều khiển. Trong tầng này, thiết bị điều khiển của chúng ta sẽ là PLC, PID controller, IPC hoặc những board mạch công nghiệp, thiết bị nhúng công nghiệp,... Những thiết bị này chịu trách nhiệm thu thập dữ liệu cảm biến, tính toán ra quyết định điều khiển trực tiếp xuống thiết

bị chấp hành, và gửi dữ liệu lên tầng phía trên. Việc trao đổi dữ liệu ở lớp này nhất thiết phải đạt được tiêu chuẩn hard real-time.

Tầng thứ ba của kim tự tháp được gọi là cấp giám sát. Tại tầng này diễn ra việc thu thập dữ liệu thô được gửi lên từ tầng điều khiển để chuẩn hóa, vẽ đồ thị, cảnh báo,… và gửi xuống quyết định của người giám sát. Xét một ví dụ, dữ liệu thô chứa giá trị "300" được gửi lên cấp giám sát là của một cảm biến nhiệt độ (trong chiếc lò nung thứ 3 tại phân xưởng A), con số này cực kỳ khó hiểu. Tuy nhiên đối với OPC UA, dữ liệu từ cảm biến nhiệt được gửi lên tầng giám sát sẽ có thêm những "properties" như "°C" hay những "reference" để xác định nó là nút con của nút "Lò_nhiệt_3" và nằm trong namespace "Trạm_A".

2.1.3. Ưu điểm của OPC UA

So sánh với các giao thức truyền thông công nghiệp mở khác cũng dựa trên Ethernet như Modbus, DNP3,… OPC UA thể hiện ra được những ưu điểm vượt trội như sau:

+ Bảo mật

Một trong những lợi ích quan trọng nhất của việc loại bỏ sự phụ thuộc vào công nghệ COM / DCOM là các tính năng bảo mật được mở rộng. Các hệ thống OPC cổ điển dựa vào việc cấu hình khó khăn và phức tạp của DCOM để cung cấp bảo mật giữa các quá trình. Thông thường, các nhà cung cấp bỏ qua bước này trong giai đoạn thử nghiệm, dẫn đến việc cấu hình rất khó khăn cho người dùng.

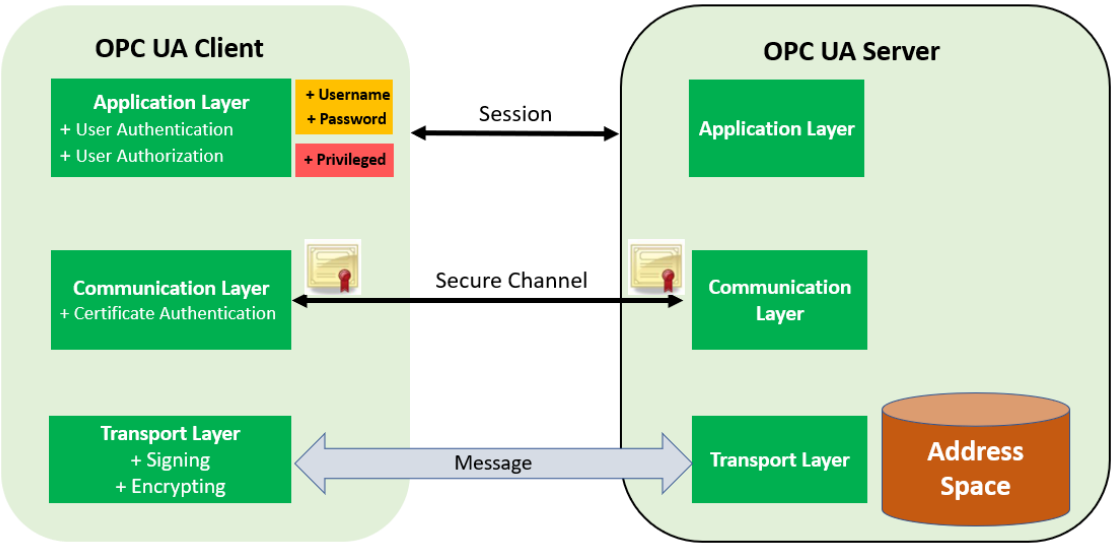
Đặc tả OPC UA dựa trên chứng chỉ tiêu chuẩn Web X.509 để cung cấp bảo mật ở các cấp độ khác nhau như:

+ Xác thực tài khoản người dùng.

+ Phân quyền người dùng.

+ Xác thực chứng chỉ.

+ Ký và mã hóa dữ liệu.



Hình 2.5: Mô hình hoạt động của giao thức OPC UA

Trong khi đó, những chuẩn truyền thông công nghiệp truyền thống như Modbus và DNP3 chỉ phù hợp khi được sử dụng ở cấp trường và cấp điều khiển. Chúng lộ rõ điểm yếu khi kết nối giữa những hệ thống OT và IT. Thông điệp của những giao thức này là rõ ràng và không được mã hóa. Giao tiếp giữa client và server hoàn toàn không có xác thực ở bất kỳ cấp độ nào.

+ Duyệt qua address space

Sau khi kết nối được thành lập, OPC UA có thể dễ dàng duyệt qua Root Folder để lấy được toàn bộ address space, nơi lưu trữ dữ liệu có cấu trúc của server. Ngược lại, với Modbus và DNP3, không có cách nào để làm được điều này ngoại trừ việc cấu hình thủ công. Những thanh ghi của Modbus hoặc DNP3 server sẽ hoàn toàn vô nghĩa với client, nếu như chúng chưa được cấu hình rằng chúng chứa dữ liệu.

+ Khả năng nhúng

+ Những sản phẩm, dự án, bộ công cụ phát triển phần mềm (Software Development Kit) phổ biến từ phiên bản thương mại đến bản mã nguồn mở miễn phí. Những khung thư viện OPC UA mã nguồn mở được viết trên nhiều ngôn ngữ

khác nhau, từ những cấp độ ứng dụng cao như JavaScript (node-opcua), Python (Free-Opcua), C# (UA .Net Standard) cho đến những cấp độ trên thiết bị nhúng có phần cứng hạn chế như C/C++ (open62541). Giao thức OPC UA được ứng dụng rộng rãi, hỗ trợ chính thức từ OPC Foundation, từ rất nhiều nhà sản xuất thiết bị, nhà phát triển phần mềm SCADA và đến từ cộng đồng kỹ sư tự động hóa.

2.1.4. Thư viện open62541

open62541 là một triển khai mã nguồn mở và miễn phí của OPC UA (OPC Unified Architecture) được viết bằng ngôn ngữ C99 và C ++ 98.

Thư viện có thể sử dụng được với tất cả các trình biên dịch chính và cung cấp các công cụ cần thiết để triển khai các máy khách và máy chủ OPC UA chuyên dụng.

Thư viện open62541 độc lập với nền tảng. Hỗ trợ giao tiếp máy khách/máy chủ OPC UA, subscription, call method và bảo mật (mã hóa).

open62541 hiện đã hỗ trợ rất nhiều cấu hình máy chủ với một số tính năng bổ sung cho thiết bị vi điều khiển nhúng như: ESP32, Arduino, Raspberry Pi,…. Các tệp nhị phân của máy chủ có thể có kích thước dưới 100kb, tùy thuộc vào mô hình thông tin.

- Giao tiếp:
  + Dùng giao thức nhị phân OPC UA.
  + Chunking (chia nhỏ các tin nhắn lớn).
  + Giao tiếp được mã hoá.
- Mô hình dữ liệu:
  + Hỗ trợ cho tất cả các loại node OPC UA (bao gồm node phương thức).
  + Thêm và xóa các nút và tham chiếu cũng trong thời gian chạy.
  + Kế thừa và khởi tạo đối tượng và các kiểu biến.
  + Kiểm soát truy cập cho từng node riêng lẻ.
- Tạo mã cho mô hình thông tin:

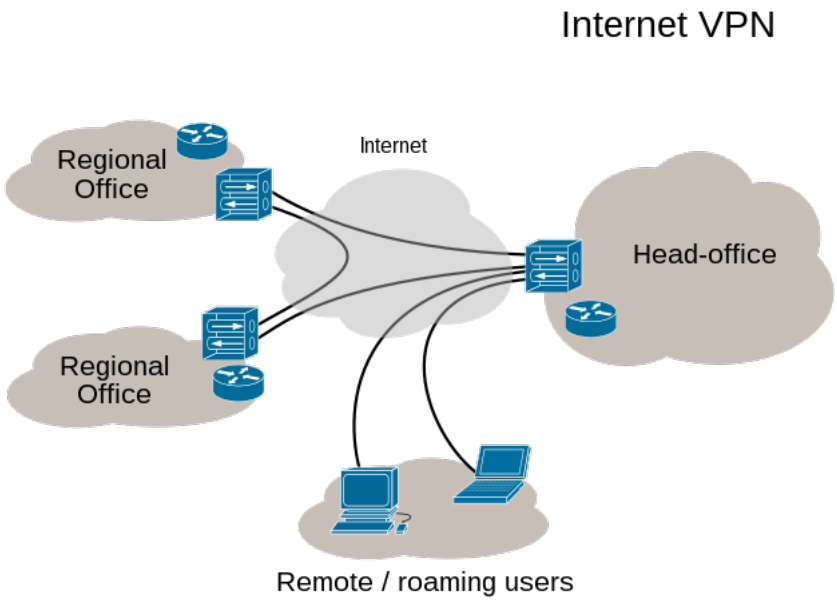
+ Hỗ trợ tạo các kiểu dữ liệu từ các định nghĩa XML chuẩn.

+ Hỗ trợ tạo các mô hình dữ liệu phía máy chủ (nodesets) từ các định nghĩa XML chuẩn.

2.2. Mạng riêng ảo

2.2.1. Giới thiệu

VPN (Virtual Private Network), là một công nghệ cho phép người dùng thiết lập một mạng nội bộ riêng từ xa thông qua Internet. VPN có thể được sử dụng để truy cập vào các website bị hạn chế truy cập về mặt địa lý, nó cũng có thể bảo vệ hoạt động duyệt web khỏi sự tấn công trên mạng công cộng bằng cách thiết lập một mạng riêng ảo.



Hình 2.6: Mô hình tổng quan VPN

(Nguồn: [1])

VPN là thành phần không thể thiếu đối với các doanh nghiệp. Họ có thể liên kết các chi nhánh từ xa khác nhau đến mạng chính của công ty. Từ đó cho phép nhân viên sử dụng các ứng dụng nội bộ của công ty. Đồng thời cho phép

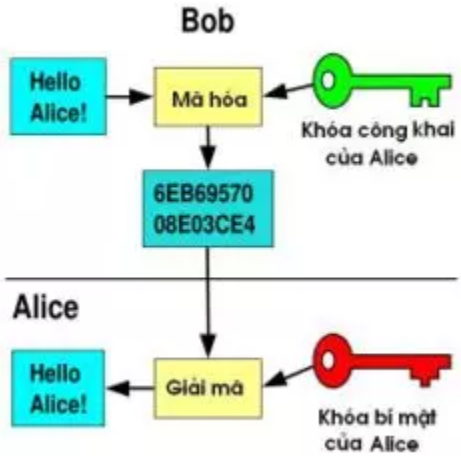
các cloud server trở thành một phần của mạng, tương tự như các máy chủ on-premises.

2.2.2. Cách hoạt động

Dữ liệu trao đổi giữa VPN Client và VPN Server sẽ đi trong một đường hầm (tunnel) riêng trên Internet. Dữ liệu đi trong tunnel sẽ được mã hoá và bảo mật nhiều lớp khác nhau. Tuỳ vào các nhà cung cấp VPN mà các phương thức bảo mật được dùng để mã hoá dữ liệu cũng khác nhau. Một trong các phương thức bảo mật nổi tiếng hiện nay có thể kể đến là IPSec (Internet Protocol Security), IKEv2 (Internet Key Exchange), SSTP (Secure Socket Tunneling Protocol), OpenVPN, ...Đa số trong các cách mã hoá này đều sử dụng cơ chế trao đổi mã khoá công khai.

Trao đổi mã khoá công khai

Sơ đồ của hệ mã công khai được cho ở hình sau:



Hình 2.7: Cách hoạt động của trao đổi khoá công khai

Hệ mã công khai sử dụng hai khóa có quan hệ toán học với nhau, tức là một khóa này được hình thành từ khóa kia: Người muốn nhận bản mã (Alice) tạo ra một khóa mật (private key) và từ khóa mật tính ra khóa công khai (public key) với một thủ tục không phức tạp, còn việc tìm khóa mật khi biết khóa công khai là bài toán khó giải được. Khóa công khai sẽ đưa đến cho người gởi bản tin (Bob)

qua kênh công cộng. Và bản tin được Bob mã hóa bằng khóa công cộng. Bản mã truyền đến Alice, và nó được giải mã bằng khóa mật.

Vậy nên, chỉ có người giữ khoá bí mật mới có thể giải mã được nội dung nhận được từ người gởi tin nhắn cho bạn.

2.2.3. Lợi ích sử dụng VPN

Truy cập Business Network từ xa: VPN thường được các du khách đi du lịch với mục đích kinh doanh (business traveler) sử dụng để truy cập mạng lưới kinh doanh của họ, bao gồm tất cả các nguồn tài nguyên mạng cục bộ. Các nguồn tài nguyên mạng cục bộ không được tiếp xúc trực tiếp với Internet để tăng cường tính bảo mật.

Truy cập Home Network từ xa : Ngoài ra bạn có thể thiết lập một VPN của riêng mình để truy cập khi đi du lịch. Điều này sẽ cho phép bạn truy cập Windows Remote Desktop thông qua Internet, tức là bạn sẽ được phép truy cập vào máy tính cá nhân của mình thông qua Internet, chia sẻ các tập tin, làm việc trên dữ liệu máy tính ở nhà và thậm chí là chơi game trên máy tính đó.

Ẩn hoạt động duyệt web từ mạng cục bộ và ISP (nhà cung cấp Internet) : Nếu đang sử dụng kết nối Wifi công cộng, và bạn duyệt web trên các trang web không phải HTTPS, khi đó các hoạt động của bạn sẽ được hiển thị với mọi người (nếu họ biết cách để xem hoạt động của bạn). Nếu muốn ẩn hoạt động duyệt web của mình để đảm bảo tính bảo mật, quyền riêng tư, bạn có thể kết nối với VPN. Mạng cục bộ sẽ chỉ nhìn thấy một kết nối VPN an toàn và duy nhất. Tất cả các traffic khác sẽ thông qua kết nối VPN. Và có thể sử dụng để bỏ qua việc giám sát của nhà cung cấp dịch vụ Internet (ISP) của bạn.

Truy cập các trang web bị chặn về mặt địa lý: Dù cho bạn là công dân sinh sống tại Hoa Kỳ, nhưng bạn đang đi du lịch tại một các quốc gia khác, không phải Hoa Kỳ và bạn muốn truy cập Netflix, Pandora hay Hulu thì điều này là không

thể. Tuy nhiên nếu kết nối với một VPN đặt tại Hoa Kỳ thì việc truy cập Netflix, Pandora hay Hulu lại là hoàn toàn có thể.

Sử dụng VPN để bỏ qua kiểm duyệt Internet.

2.2.4. Wireguard

(Nguồn: [2])

WireGuard là một giải pháp VPN (Virtual Private Network) tập trung vào bảo mật. Nó được biết đến ban đầu bởi tính đơn giản và dễ dàng sử dụng. WireGuard sử dụng các giao thức mã hoá và các thuật toán để bảo vệ dữ liệu. Ban đầu, WireGuard được phát triển dành cho Linux. Hiện nay, nó đã phổ biến đa nền tảng như: Windows, macOS, BSD, iOS và Android.

Một lý do khiến WireGuard trở nên phổ biến là vì nó giúp tăng cường sự tin tưởng vào khía cạnh công nghệ của mọi thứ. WireGuard có mã nguồn mở và chỉ bao gồm chưa đến 4000 dòng code, kích thước của nó chỉ bằng 1% so với các công nghệ cạnh tranh (OpenVPN/IPsec). Điều này có nghĩa là một người có kiến thức có thể hiểu được WireGuard tương đối nhanh chóng. Điều này thể hiện triết lý "bảo mật thông qua sự đơn giản" của WireGuard.

Cách thức mã hoá của Wireguard:

Đầu tiên, WireGuard loại trừ tính linh hoạt của mật mã. Đó là việc cung cấp các lựa chọn giữa thuật toán mã hóa, trao đổi key. Sở dĩ vì nó dẫn đến việc triển khai không an toàn với các công nghệ khác. Thay vào đó, giao thức của WireGuard sử dụng một lựa chọn các bản mã hiện đại. Do đó, các lựa chọn mật mã mặc định này đủ mạnh để người dùng không thể thay đổi hoặc cấu hình sai.

WireGuard sử dụng ChaCha20 để mã hóa đối xứng với Poly1305 cho message authentication. Sự kết hợp này hiệu quả hơn AES trên các kiến trúc CPU được nhúng mà không có khả năng tăng tốc phần cứng mật mã (cryptographic hardware acceleration); Curve25519 cho Diffie-Hellman (ECDH). Hay BLAKE2s để hashing, nhanh hơn SHA-3 và 1.5-RTT dựa trên nền tảng Noise. Ngoài ra, nó cũng bao gồm tính năng bảo vệ khỏi giả mạo, từ chối dịch vụ, và replay attack.

Trong giao thức WireGuard, mỗi peer được xác định với những peer khác thông qua các public key. Việc này tương tự với cách xác thực dựa trên khóa trong OpenSSH. Các public key cũng được sử dụng để thiết lập địa chỉ IP nào được chỉ định cho từng peer bên trong tunnel. Nó là một phần trong khái niệm mới được Wireguard gọi là "cryptokey routing".

Giao thức này không phản hồi với các packet mà nó không nhận ra. Vì vậy quá trình network scanning sẽ không cho thấy WireGuard đang chạy trên server. Hơn nữa, nó có thể hoạt động như cả server lẫn client cùng một lúc.

Hiệu suất của Wireguard:

Trên Linux, WireGuard hoạt động ở kernel space. Do đó, hiệu suất của nó cao hơn nhiều so với OpenVPN – trong userspace và virtual network interface driver. Điểm benchmark của WireGuard cho hiệu suất và tốc độ kết nối cao hơn OpenVPN đến 4 lần. Đồng thời, tốc độ của nó cũng cao hơn so với VPN dựa trên IPsec, trên cùng một phần cứng.

Wireguard tích hợp với Linux kernel:

Kể từ phiên bản 5.6 của Linux Kernel, phát hành vào ngày 29/3/2020, WireGuard đã chính thức là một trong những công nghệ được tích hợp mặc định. Đây cũng được xem là lần phát hành ổn định đầu tiên của WireGuard (phiên bản 1.0.0).

Các kế hoạch sau đó của Donenfeld liên quan đến việc hợp nhất một API mã hóa mới, đơn giản hơn vào Linux kernel, gọi là Zinc. Donefeld cho rằng Zinc sẽ giúp các developer thực hiện các thao tác mã hóa trong ứng dụng dễ dàng hơn. Việc này đã đánh dấu một bước ngoặt quan trọng trong quá trình upstream của WireGuard vào hệ thống Linux subsystem. Kể từ đó, đã có nhiều bước tiến nhanh chóng, đáng kể trước khi Zinc chính thức được phát hành ổn định.

2.3. Máy chủ ảo

2.3.1. Giới thiệu

(Nguồn: [3])

VPS là viết tắt của Virtual Private Server hay còn gọi là máy chủ riêng ảo.

Đây là dạng máy chủ được tạo ra bằng phương pháp phân chia một máy chủ vật lý thành nhiều máy chủ ảo khác nhau có tính năng tương tự như máy chủ riêng (dedicated server), chạy dưới dạng chia sẻ tài nguyên từ máy chủ vật lý ban đầu đó.

Server – Máy chủ là một máy tính mà máy chủ lưu trữ web của bạn lưu trữ các tệp và cơ sở dữ liệu cần thiết cho trang web. Bất cứ khi nào máy khách muốn truy cập trang web bạn đang sở hữu, trình duyệt của họ sẽ gửi một yêu cầu đến máy chủ của bạn và nó sẽ chuyển các tệp cần thiết qua internet. Dịch vụ lưu trữ VPS cung cấp cho bạn một máy chủ ảo mô phỏng một máy chủ vật lý. Tuy nhiên, trên thực tế thì máy được chia sẻ giữa một số người dùng với nhau.

Virtual – Công nghệ ảo hóa. Nhà cung cấp dịch vụ lưu trữ cài đặt một lớp ảo hóa trên hệ điều hành (OS) của máy chủ. Lớp này chia máy chủ thành các phân vùng và cho phép mỗi người dùng cài đặt hệ điều hành và phần mềm của riêng họ (Private).

Hình 2.8: So sánh VPS hosting với dedicated và shared hosting

Do đó, VPS – một máy chủ riêng ảo – vừa ảo vừa riêng tư do bạn có toàn quyền kiểm soát. Nó được tách biệt với những người dùng máy chủ khác ở cấp độ hệ điều hành. Trên thực tế, công nghệ VPS tương tự như việc tạo phân vùng trên máy tính của riêng bạn khi bạn muốn chạy nhiều hơn một hệ điều hành (Ví dụ: Windows và Linux) mà không cần khởi động lại.

Máy chủ ảo VPS thường được lựa chọn bởi các chủ sở hữu trang web có lưu lượng truy cập khá cao. Mức trung bình của lưu lượng truy cập vượt quá giới hạn của gói shared hosting. Nhưng vẫn không cần tài nguyên của một máy chủ chuyên dụng.

2.3.2. Ưu điểm của VPS

So với shared hosting, một số ưu điểm nổi bật của VPS có thể kể đến là:

- Nhanh hơn và đáng tin cậy hơn một máy chủ shared hosting.
- Vì các tài nguyên máy chủ như bộ nhớ hoặc sức mạnh xử lý được đảm bảo, nên không có sự dao động nào về tài nguyên có sẵn.
- Các vấn đề và sự gia tăng lưu lượng truy cập của những người dùng máy chủ khác không ảnh hưởng đến trang web của bạn.
- Bạn có quyền truy cập superuser (**root**) vào máy chủ của mình.
- Bạn có quyền riêng tư tốt hơn vì các tệp và cơ sở dữ liệu của bạn bị khóa khỏi những người dùng máy chủ khác.
- VPS là một dịch vụ dễ mở rộng quy mô. Khi trang web của bạn phát triển, bạn có thể dễ dàng nâng cấp tài nguyên máy chủ của mình (RAM, CPU, dung lượng đĩa, băng thông,…).

2.3.3. Ứng dụng của VPS

VPS được sử dụng rất rộng rãi trong doanh nghiệp lẫn những mô hình kinh doanh nhỏ lẻ. Một số ứng dụng của VPS có thể kể đến là: chạy máy chủ game, phát triển dự án, lưu trữ website, sử dụng cho hệ thống email doanh nghiệp, …

Trong luận văn này, VPS được sử dụng để chạy web server và OPC UA Client để thu thập dữ liệu của OPC UA Server từ xa.

Chương 3. THỰC HIỆN

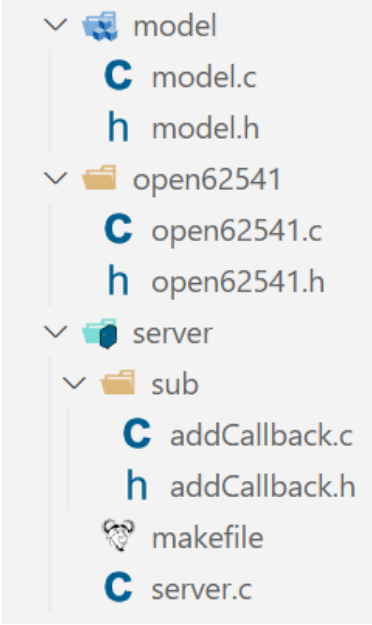
3.1. Xây dựng gateway

3.1.1. Xây dựng OPC UA Server

Hiện nay, có rất nhiều khung thư viện mã nguồn mở miễn phí được xây dựng trên đa dạng các ngôn ngữ lập trình khác nhau, và đến từ các bản chính thức của OPC Foundation hoặc đến từ cộng đồng. Trong luận văn, ta sẽ sử dụng thư viện open62541 được viết bằng ngôn ngữ C/C++ để xây dựng một OPC UA Server gọn nhẹ, hiệu suất cao trên một board máy tính nhúng Raspberry Pi 4 model B+.

Khi một chương trình OPC UA server chạy, nó sẽ phân bổ một vùng nhớ trên thiết bị để lưu giữ giá trị các Node được gọi là Address Space. Các node của server có được thông qua việc cấu hình một tệp XML được gọi là Information Model, sau đó biên dịch nó thành mã có cùng ngôn ngữ lập trình với khung thư viện sử dụng.

Cấu trúc thư mục của OPC UA Server sẽ được xây dựng như sau:



Hình 3.1: Cấu trúc thư mục OPC UA Server

Trong đó:

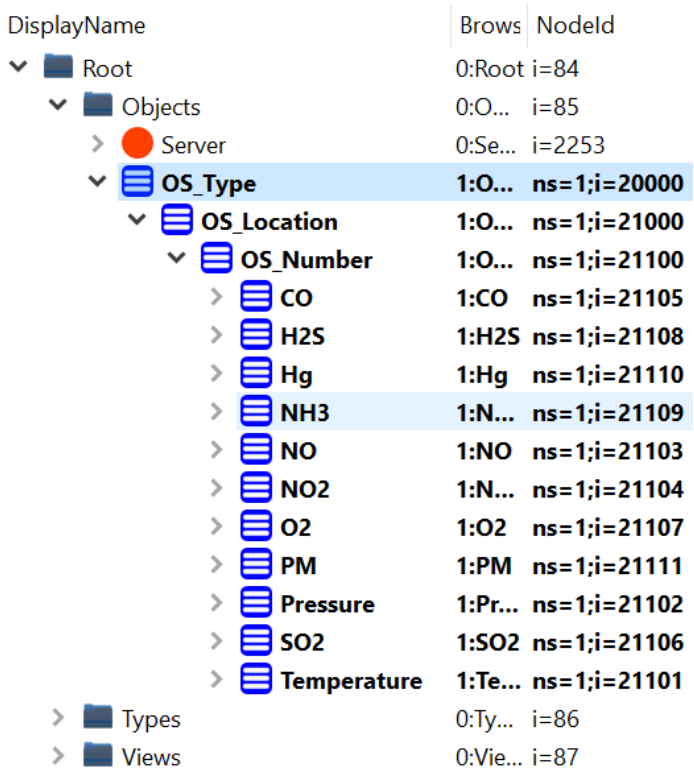
+ Thư mục model: sẽ chứa mã phục vụ cho việc khởi tạo Address Space.

+ Thư mục open62541: chứa mã của thư viện open62541 cung cấp API liên quan đến việc khởi tạo OPC UA server, được biên dịch theo hướng dẫn trên trang chủ của thư viện.

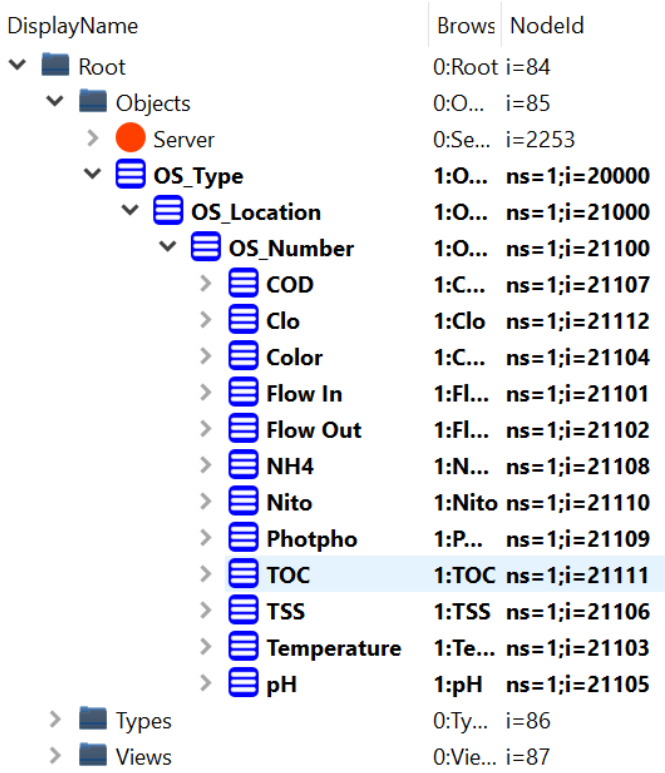
+ Thư mục server: chứa mã xây dựng chương trình chính OPC UA server.

+ Thư mục sub: chứa mã để thêm hàm callback cho các node.

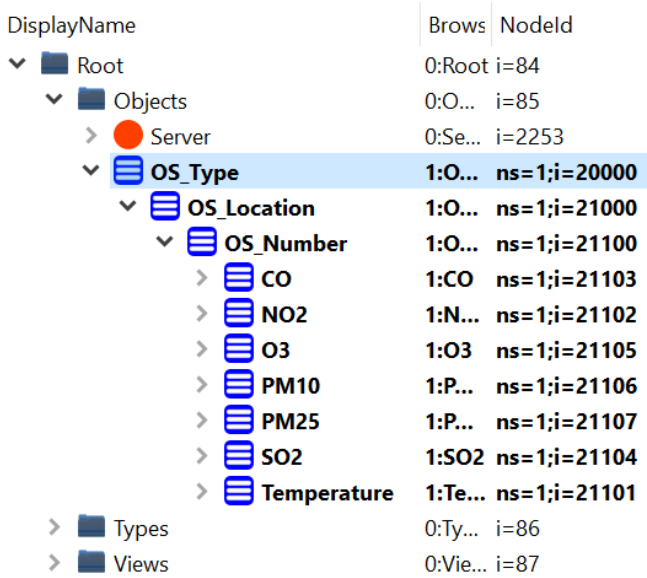
Tệp Information Model được thiết kế phải tuân theo một tệp xml tiêu chuẩn do OPC Foundation cung cấp, có tên là NodeSet2.xml. Có rất nhiều công cụ mô hình hóa có sẵn có thể được sử dụng. Trong đề tài, ta sẽ sử dụng một công cụ có tên là opcua-modeler của FreeOpcUa để tạo file XML Information Model.



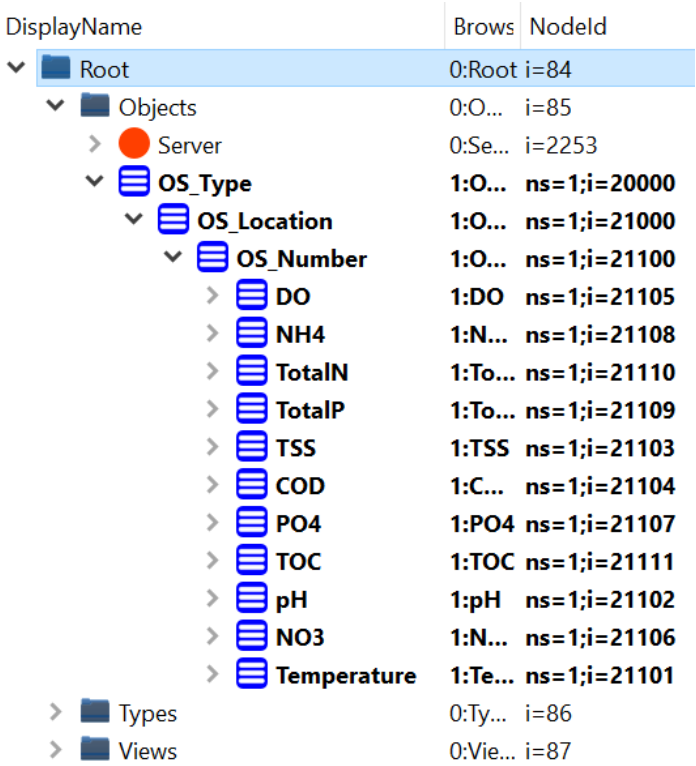
Hình 3.2: Information model của trạm khí thải



Hình 3.3: Information model của trạm nước thải



Hình 3.4: Information model của trạm không khí



Hình 3.5: Information model của trạm nước ngầm

Theo những hình trên, cấu trúc Information Model của một trạm quan trắc gồm có:

+ OS_Type: là một node có kiểu String giữ giá trị thể hiện kiểu của trạm quan trắc, ví dụ: khí thải, nước thải, nước mặt, không khí,...

+ OS_Location: là một node có kiểu String giữ giá trị thể hiện vị trí của trạm, ví dụ: KCN Long Thành, KCN Biên Hòa, Hồ Dầu Tiếng,...

+ OS_Num: là một node có kiểu String giữ giá trị thể hiện số thứ tự của trạm, ví dụ: trạm 1, trạm 2, trạm 3,...

+ Và các node còn lại là các node có kiểu Double giữ giá trị thể hiện thông số mà trạm cần quan trắc.

Đến đây, ta đã có các tệp XML chứa Information Model của các trạm. Tiếp theo, ta sẽ biên dịch những tệp XML này thành mã C/C++ để chuẩn bị cho việc

khởi tạo Address Space. Có rất nhiều công cụ biên dịch phổ biến, trong đề tài sẽ sử dụng công cụ nodeset_compiler của open62541 để biên dịch.

Những node được khởi tạo từ mã C/C++ của Information Model vẫn hoàn toàn chưa được liên kết đến một giá trị vật lý nào. Vì vậy, ta sẽ viết mã để tạo những hàm callback cho các node. Những hàm callback này có chức năng gán các giá trị vật lý vào các node mỗi khi nó được một client đọc.

```
void readTemperature(UA_Server *server,
const UA_NodeId *sessionId, void *sessionContext,
const UA_NodeId *nodeid, void *nodeContext,
const UA_NumericRange *range, const UA_DataValue *data)
{
  uint16_t register[1];
  modbus_read_registers(modbus_client, 0, 1,
register[0]);


  UA_Variant value;
  UA_Variant_setScalar(&value, &register[0],
&UA_TYPES[UA_TYPES_DOUBLE]);
  UA_Server_writeValue(server, UA_NODEID_NUMERIC(2,
21101), value);
}


void addCallback(UA_Server* server, UA_NodeId nodeid) {
  UA_ValueCallback callback;
  callback.onRead = &readTemperature;
  callback.onWrite = NULL;
  UA_Server_setVariableNode_valueCallback(server,
UA_NODEID_NUMERIC(2, 21101), callback);
}
```

Hàm readTemperature sẽ thực hiện đọc một thanh ghi modbus, sau đó định kiểu dữ liệu là double và ghi nó vào một Node có NodeID là “ns=2;i=21101”.

Tiếp theo, hàm addCallback sẽ gán địa chỉ của hàm readTemperature cho thuộc tính callback của Node để mỗi khi có yêu cầu read từ OPC UA client trên Node đó, nó sẽ thực hiện hàm readTemperature.

Tiếp theo, ta sẽ lập trình chương trình chính cho OPC UA server với các chức năng như sau:

```
// Initialize an Modbus TCP client instance
    mb = modbus_new_tcp(mb_ip, mb_port);
    modbus_connect(mb);
// Initialize an OPC UA server instance
    UA_Server *server = UA_Server_new();
// Initialize Address Space
    model(server);
// Add Callback Function
    addCallback(server, nodeId);
// Run OPC UA server
    retval = UA_Server_run(server, &running);
// Delete OPC UA server
    UA_Server_delete(server);
// Delete Modbus client
    modbus_close(mb_client);
    modbus_free(mb_client);
```

3.1.2. Xây dựng VPN Client

VPN được sử dụng để giúp các trạm quan trắc cách xa nhau về mặt địa lý và máy chủ ảo chạy trên Azure tham gia vào một mạng nội bộ riêng. Từ đó, OPC UA Client (trên VPS) và Server (trên Raspberry Pi) có thể nói chuyện với nhau thông qua giao thức OPC UA cho dù khoảng cách là bao xa, miễn là có kết nối Internet. Bên cạnh đó, dữ liệu được an toàn hơn nhờ vào việc VPN kết nhiều phương thức mã hoá để mã hoá thông tin.

Có rất nhiều cách để có một mạng riêng ảo, có thể mua từ nhà cung cấp dịch vụ VPN, cũng có thể sử dụng các thư viện được viết sẵn để thực thi mã lệnh trên máy chủ server. Trong luận văn này, nhóm sử dụng mã thư viện mã nguồn mở Wireguard để xây dựng VPN.

Xây dựng Wireguard Client:

Để xây dựng được Wireguard Client chỉ cần tải Wireguard về máy và một file cấu hình gồm public key server và private key client.

Tải Wireguard về máy chỉ cần chạy lệnh dòng lệnh này bằng terminal:

apt install wireguard -y

Import file config được tạo ra từ Wireguard server (chi tiết ở chương 3.2.2). File config này chứa các thông tin để kết nối đến wg server gồm: private key của client, địa chỉ IP nội bộ khi tham gia vào VPN, public key của server đang chạy Wireguard lẫn địa chỉ public IP và port.

```
[Interface]
PrivateKey = OPwwpAIPASnQxGhwBB/JHVXjhx6/+ApDbwRw0jPw+Wo=
Address = 10.66.66.2/32,fd42:42:42::2/128
DNS = 1.1.1.1,1.0.0.1
[Peer]
PublicKey = whWkeDIjbSirLAy5N+2d33N+kHnDn94IUzqhT/dtfCc=
PresharedKey =
6eEjhfz0lmumvHvyF0YKkH+Bir3wUBNeBgLkl8M3QbU=
Endpoint = 20.205.122.62:51820
```

Lưu ý rằng, file config này phải được lưu trong đường dẫn ./etc/wireguard của hệ điều hành Linux.

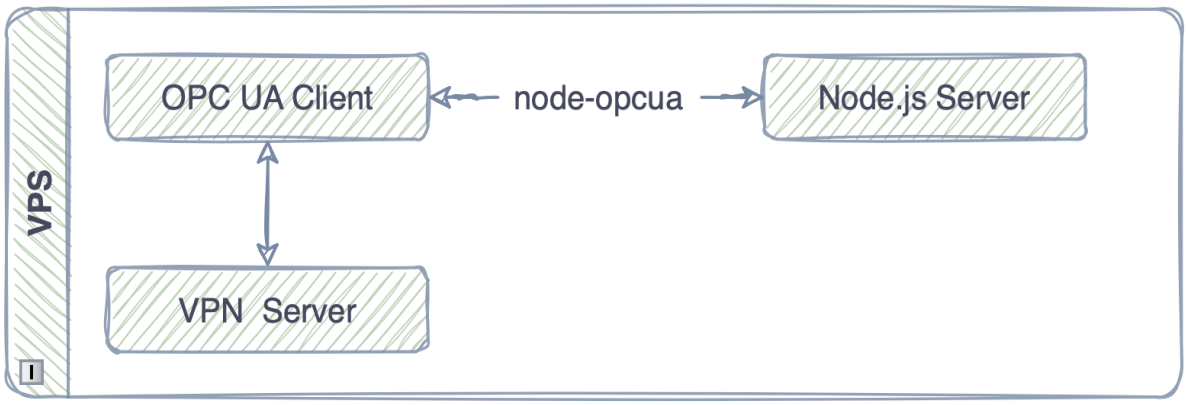
Sau khi đã cài đặt Wireguard và có file cấu hình từ server, việc còn lại chỉ đơn giản là chạy dòng lệnh sau:

wg-quick up <tên file config đã lưu>

3.2. Khởi tạo và phát triển ứng dụng trên VPS

3.2.1. Tổng quan kiến trúc hệ thống trên VPS

Với sự phát triển mạnh mẽ của cuộc cách mạng 4.0, việc kết nối vạn vật với nhau qua Internet trở nên hết sức quan trọng. Nên việc đưa dữ liệu lên VPS để xử lý và lưu vào cơ sở dữ liệu là điều tất yếu. Trong luận văn này, nhóm sử dụng VPS để thu thập dữ liệu của các trạm từ xa. Sau đó, xây dựng một web server bằng node.js để có thể theo dõi các chỉ số theo thời gian thực và điều khiển từ xa thông qua web client và ứng dụng android. Cơ sở dữ liệu trên cloud được sử dụng là MongoDB để lưu trữ thông tin quản trị viên và alarm.



Hình 3.6: Tổng quan kiến trúc hệ thống trên VPS

Kiến trúc ứng dụng trên VPS bao gồm:

- VPN Server: dùng để tạo kết nối an toàn tới các trạm từ xa.
- OPC Client: có chức năng thu thập dữ liệu từ các trạm.
- Node.js server: tạo REST API cho ứng dụng web mà mobile.

3.2.2. Xây dựng Wireguard server

Cài đặt Wireguard:

Tương tự như trên Wireguard Client thì trên server cũng phải cài đặt Wireguard trước tiên bằng cách sử dụng apt để cài đặt.

apt install wireguard

- Enable ip_forward cho Kernel:

```
echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf
sysctl -w net.ipv4.ip_forward=1
sysctl -p
```

Cấu hình Wireguard:

Wireguard hoạt động theo mô hình public_key/private_key. Do đó ta cần:

- Tạo cặp public/private key cho Server: Private key sẽ được lưu trữ bí mật trên server, public key sẽ gửi cho client.

```
cd /etc/wireguard/keys/

wg genkey | tee server_private.key | wg pubkey >
server_public.key
```

- Tạo cặp public/private key cho mỗi Client: Private key sẽ được giữ bởi client, public key sẽ được khai báo trên server

```
cd /etc/wireguard/keys/

wg genkey | tee client_private.key | wg pubkey >
client_public.key
```

Tạo server config: file này chứa các thông số cấu hình cho Wireguard server bao gồm:

- SERVER_IP: 20.205.122.62 – Địa chỉ IP public của server
- SERVER_PORT: 51820 – Port của VPN (Cần mở port này trên VPS để có thể sử dụng)

Inbound port rules | Outbound port rules | Application security groups | Load balancing

Network security group VPS-nsg (attached to network interface: vps455)
Impacts 0 subnets, 1 network interfaces

Add inbound port rule

Priority	Name	Port	Protocol	Source	Destination	Action
300	SSH	22	TCP	Any	Any	Allow
320	Port_51820	5182	UDP	Any	Any	Allow
330	Inport_3000	3000	Any	Any	Any	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

Hình 3.7: Mở port trên VPS

- Private key của server và public key của client vừa được tạo ra ở bước trên.

```
[Interface]
Address = 10.66.66.1/24,fd42:42:42::1/64
ListenPort = 51820
PrivateKey = IAM4BZ+pD9Z6rm4BXtJMF+0I9tz1hVd9R5S5zslaWVc=
PostUp = iptables -A FORWARD -i eth0 -o wg -j ACCEPT;
iptables -A FORWARD -i wg -j ACCEPT; ipt>
PostDown = iptables -D FORWARD -i eth0 -o wg -j ACCEPT;
iptables -D FORWARD -i wg -j ACCEPT; i>

### Client iPhone
[Peer]
PublicKey = NX/DaSls2TFJRUwwxR0JiOZ75o8H0e66OnP74vo5rG4=
PresharedKey = 6eEjhfz0lmumvHvyF0YKkH+Bir3wUBNeBgLkl8M3QbU=
AllowedIPs = 10.66.66.2/32,fd42:42:42::2/128

### Client Rasp
[Peer]
PublicKey = nA3/ZPGqkrKN0P5mDAyOcXSWi8DA+Y/YhgxD5PmyjRs=
PresharedKey = 10O8mzHNZxNvtmDXFhMvic3X4p/5mslN3/o3QGW4X38=
AllowedIPs = 10.66.66.3/32,fd42:42:42::3/128

### Client laptop
```

Sau khi tạo xong các file cấu hình cần thiết, khởi chạy dịch vụ Wireguard VPN bằng cách chạy lệnh:

wg-quick up wg0

Khi file config trên server có chứa public key của client và file config trên client cũng có public key của server thì chúng có thể kết nối với nhau và tạo thành một mạng riêng ảo.

3.2.3. Xây dựng OPC UA Client trên VPS

Sau khi thiết lập xong mạng riêng ảo thì ta đã có một kết nối an toàn từ các trạm cho đến VPS. Việc tiếp theo là cần xây dựng một OPC UA Client trên máy ảo để thu thập dữ liệu của các trạm từ xa thông qua giao thức OPC UA.

OPC UA Client trong luận văn này được xây dựng dựa trên thư viện mã nguồn mở node-opcua. Thư viện này được viết bằng ngôn ngữ JavaScript để thuận tiện cho việc thu thập dữ liệu các trạm và xây dựng web server bằng node.js.

Để tạo OPC UA Client, trước hết phải tải thư viện node-opcua từ npm:

npm install node-opcua –save

OPC UA Client thực hiện lần lượt các chức như sau để kết nối với OPC UA Server và thu thập dữ liệu. Hàm async.series được sử dụng để thực hiện các tác vụ theo thứ tự. Ví dụ, hàm connection sẽ được đảm bảo chạy trước hàm createSession.

- Connect to: Tạo kết nối thông qua địa chỉ URI.

```
// step 1 : connect to

          await client.connect(endpointUrl);
          console.log("connected !");
```

- createSession: Tạo phiên làm việc (session) để đảm bảo 2 bên luôn có kết nối với nhau.

```
// step 2 : createSession

     const session = await client.createSession();
     console.log("session created !");
```

- Browse: Trong session, OPC UA sẽ duyệt qua namespace hiện tại ở thư mục gốc để tìm các node.

```
 // step 3 : browse

     const browseResult = await
session.browse("RootFolder");
     console.log("references of RootFolder :");
     for(const reference of
browseResult.references) {
        console.log( "   -> ",
reference.browseName.toString());
     }
```

- Install subscription: Tạo một subscription để có thể theo dõi server. Nếu mất kết nối, client vẫn tiếp tục tạo session với server trong vòng 10s

```
// step 4: install a subscription and install a
monitored item for 10 seconds

const subscription =
ClientSubscription.create(session, {
    requestedPublishingInterval: 1000,
    requestedLifetimeCount:      100,
    requestedMaxKeepAliveCount:   10,
    maxNotificationsPerPublish:  100,
    publishingEnabled: true,
    priority: 10
});
```

- Install Monitor Items: Theo dõi giá trị node khi có sự thay đổi giá trị

```
// install monitored item
const itemToMonitor: ReadValueIdLike = {
    nodeId: "ns=3;s=Scalar_Simulation_Float",
    attributeId: AttributeIds.Value
};
const parameters: MonitoringParametersOptions = {
    samplingInterval: 100,
    discardOldest: true,
    queueSize: 10
};
const monitoredItem  = ClientMonitoredItem.create(
    subscription,
    itemToMonitor,
    parameters,
    TimestampsToReturn.Both
);
// Read the value on server when it changes
// Then, read value will be added to DataList array
monitoredItem.on("changed", (dataValue, index) => {
    DataList[OS_index][index].value =
dataValue.value.value.toFixed(2)
    DataList[OS_index][index].calculateAlarm
});

}
}
```

Class Infor cũng được khai báo trong OPC UA Client để tiện cho việc khai báo thông tin các trạm.

```
class Infor {
    constructor (endpoint, OS_Type, OS_Number, OS_Location){
        this.endpoint = endpoint;
        this.OS_Location = OS_Location;
        this.OS_Number = OS_Number;
        this.OS_Type = OS_Type
```

Class Infor cũng được sử dụng để thuận tiện cho việc quản lý dữ liệu cũng như xử lý tác vụ Alarm:

```
  class Data {
    severity = 0;

    constructor(indicator, value,  unit,  LOWLOW,  LOW,  HIGH,
HIGHHIGH,  DEAD_BAND, ) {
        this.indicator = indicator;
        this.value = value;
        this.unit = unit;
        this.severity = 0;
        this.startTime = 0;
        this.duration = 0;
        this.content = "";
        this.ACK = false;
        this.ackUser = "";
        this.ACKTime = 0;
        this.LOWLOW = this.c_LOWLOW =  LOWLOW;
```

Hàm trả về mức độ nguy hiểm của Alarm:

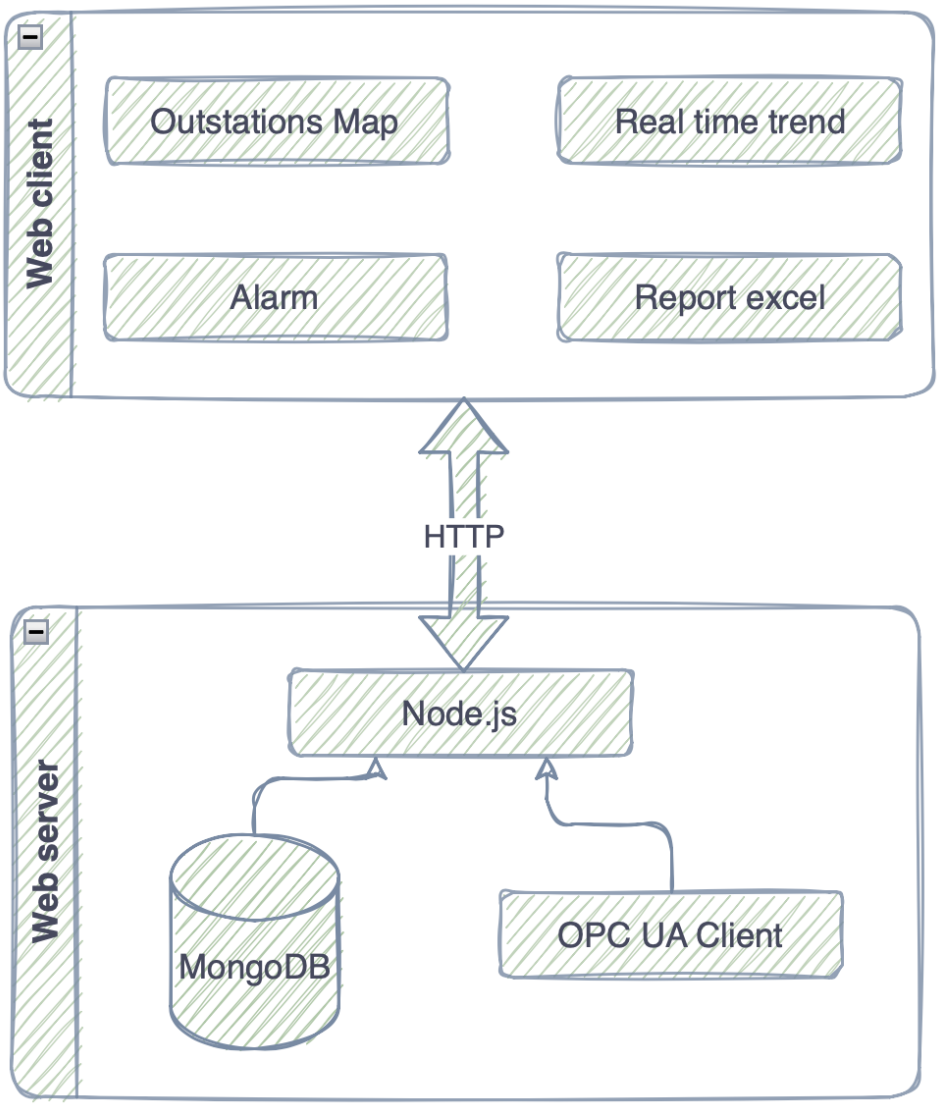
```
calculateSeverity() {
        if(this.value <= this.c_LOWLOW || this.value >= this.c_HIGHHIGH)
{
          this.severity = 2;
        } else if(this.value > this.c_LOWLOW && this.value <= this.c_LOW
|| this.value >= this.c_HIGH && this.value < this.c_HIGHHIGH) {
          this.severity = 1;
        } else this.severity = 0;
    }
```

Hàm trả về nội dung alarm gồm: tên của indicator và mức độ Alarm.

```
calculateContent() {
        if(this.value <= this.c_LOWLOW) {
          this.content = this.indicator + " is out of LOWLOW limit";
        } else if (this.value >= this.c_HIGHHIGH) {
          this.content = this.indicator + " is out of HIGHHIGH limit";
        } else if(this.value > this.c_LOWLOW && this.value <=
this.c_LOW) {
          this.content = this.indicator + " is out of LOW limit";
        } else if (this.value >= this.c_HIGH && this.value <
this.c_HIGHHIGH) {
          this.content = this.indicator + " is out of HIGH limit";
        } else
        this.content = this.indicator;
    }
```

3.3. Xây dựng web server

3.3.1. Tổng quan



Hình 3.8: Tổng quan về mô hình ứng dụng web

Trong thời đại công nghiệp 4.0, khi mọi thứ đã được số hoá và có sẵn trên khắp Internet. Chúng ta có thể truy cập các thông tin này bất kì đâu trên trên đất miễn là có Internet. Một trong những cách phổ biến là truy cập qua giao diện web.

Trong phần này, nhóm sẽ xây dựng một ứng dụng web server-side render bằng node.js và Express framework. Dữ liệu thu thập được từ OPC UA Client

được node.js xử lý và chìa các REST API bằng socket.io để giao diện web và ứng dụng mobile có thể truy cập được thông giao thức HTTP.

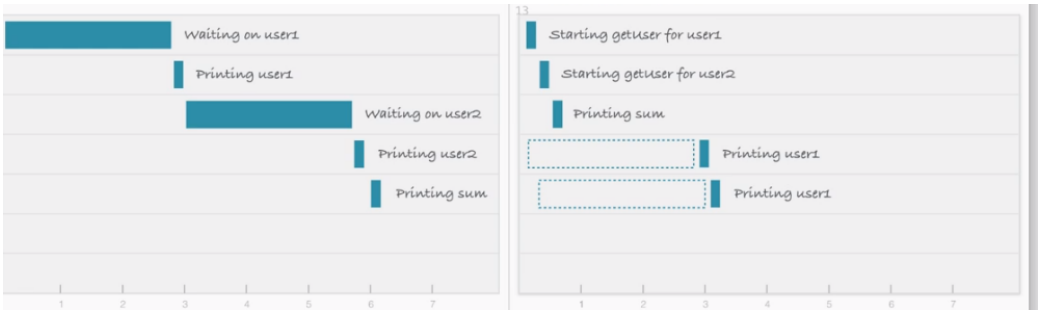
Phần node.js được xây dựng theo mô hình MVC để tiện trong việc xử lý code behind và render các website.

3.3.2. Các công cụ và giao thức sử dụng

Node.js là một môi trường thực thi JavaScript (JavaScript runtime environment) cho phép JavaScript có thể chạy ở phía server. Nó là một nền tảng được xây dựng trên "V8 Javascript engine" được viết bằng c++ và Javascript.

Điểm mạnh nhất của node.js là sử dụng cơ chế non-blocking I/O.

Xét một tình huống là chúng ta cần dữ liệu backend chi tiết cho user1 và user2 và sau đó hiển thị chúng trên màn hình hoặc bảng điều khiển. Để phản hồi cho lệnh này sẽ mất thời gian, nhưng cả 2 dữ liệu user có thể thực hiện độc lập và cùng lúc.



Hình 3.9: Cơ chế blocking I/O (trái) và non-blocking I/O (phải)

(Nguồn: [4])

Trong phương thức blocking, dữ liệu của user2 sẽ không bắt đầu cho đến khi dữ liệu của user1 xuất trên màn hình. Nếu đây là một server web, chúng ta phải bắt đầu một thread mới cho từng user mới. Nhưng JavaScript là đơn luồng (tính chất "xong cái này mới đến cái khác"). Và điều này làm JavaScript không phù hợp cho các tác vụ đa luồng.

Đó là lý do non-blocking xuất hiện. Mặt khác, sử dụng request non-blocking, bạn có thể request dữ liệu cho user2 mà không phải đợi phản hồi request từ user1. Bạn có thể bắt đầu cả 2 request song song. Non-blocking I/O loại bỏ nhu cầu đa luồng vì server có thể xử lý nhiều request cùng một lúc.

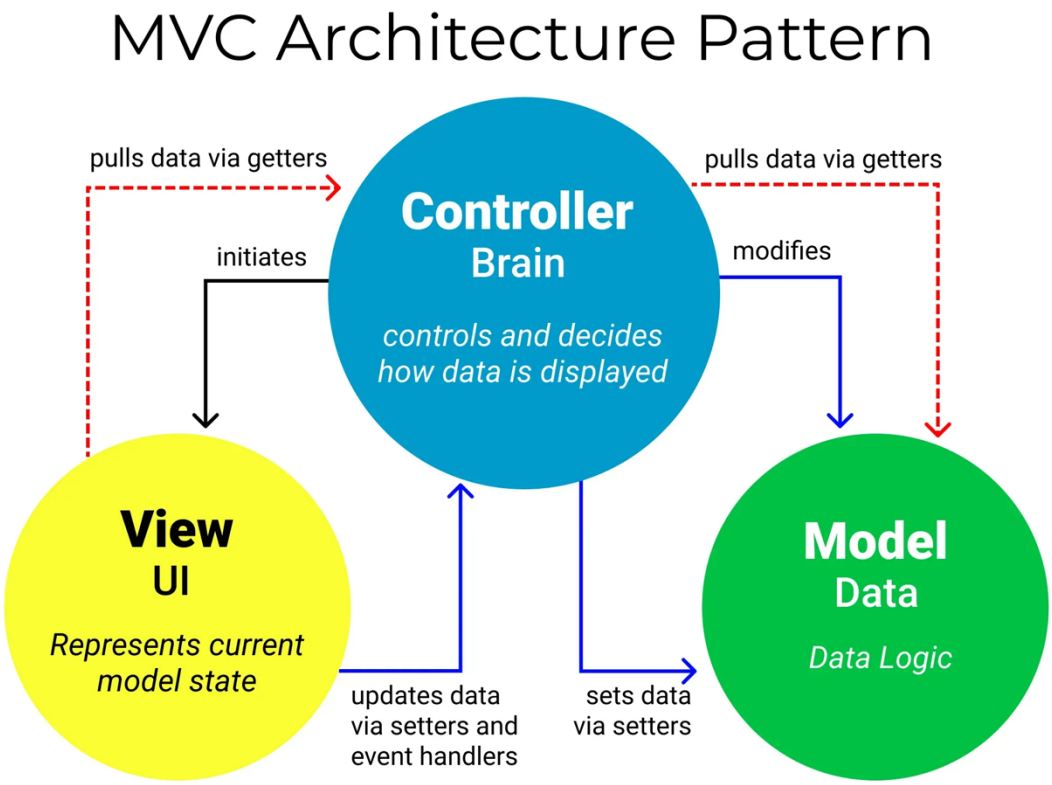
Express là một framework dành cho nodejs. Nó cung cấp cho chúng ta rất nhiều tính năng mạnh mẽ trên nền tảng web cũng như trên các ứng dụng di động. Express hỗ trợ các phương thức HTTP và middleware tạo ra một API vô cùng mạnh mẽ và dễ sử dụng. Có thể tổng hợp một số chức năng chính của express như sau:

- Thiết lập các lớp trung gian để trả về các HTTP request
- Định nghĩa router cho phép sử dụng với các hành động khác nhau dựa trên phương thức HTTP và URL
- Cho phép trả về các trang HTML dựa vào các tham số.

Mô hình MVC

MVC là viết tắt của cụm từ "Model-View-Controller". Đây là mô hình thiết kế sử dụng trong kỹ thuật phần mềm. MVC là một mẫu kiến trúc phần mềm để tạo lập giao diện người dùng trên máy tính. MVC chia thành ba phần được kết nối với nhau như tên gọi: Model (dữ liệu), View (giao diện) và Controller (bộ điều khiển).

Mô hình MVC gồm 3 loại chính là thành phần bên trong không thể thiếu khi áp dụng mô hình này:



Hình 3.10: Mô hình MVC trong lập trình web

(Nguồn: [5])

- Model: Là bộ phận có chức năng lưu trữ toàn bộ dữ liệu của ứng dụng và là cầu nối giữa 2 thành phần bên dưới là View và Controller. Một model là dữ liệu được sử dụng bởi chương trình. Đây có thể là cơ sở dữ liệu, hoặc file XML bình thường hay một đối tượng đơn giản. Chẳng hạn như biểu tượng hay là một nhân vật trong game.
- View: Đây là phần giao diện (theme) dành cho người sử dụng. View là phương tiện hiển thị các đối tượng trong một ứng dụng. Chẳng hạn như hiển thị một cửa sổ, nút hay văn bản trong một cửa sổ khác. Nó bao gồm bất cứ thứ gì mà người dùng có thể nhìn thấy được.

- Controller: Là bộ phận có nhiệm vụ xử lý các yêu cầu người dùng đưa đến thông qua View. Một controller bao gồm cả Model lẫn View. Nó nhận input và thực hiện các update tương ứng.

Socket.io

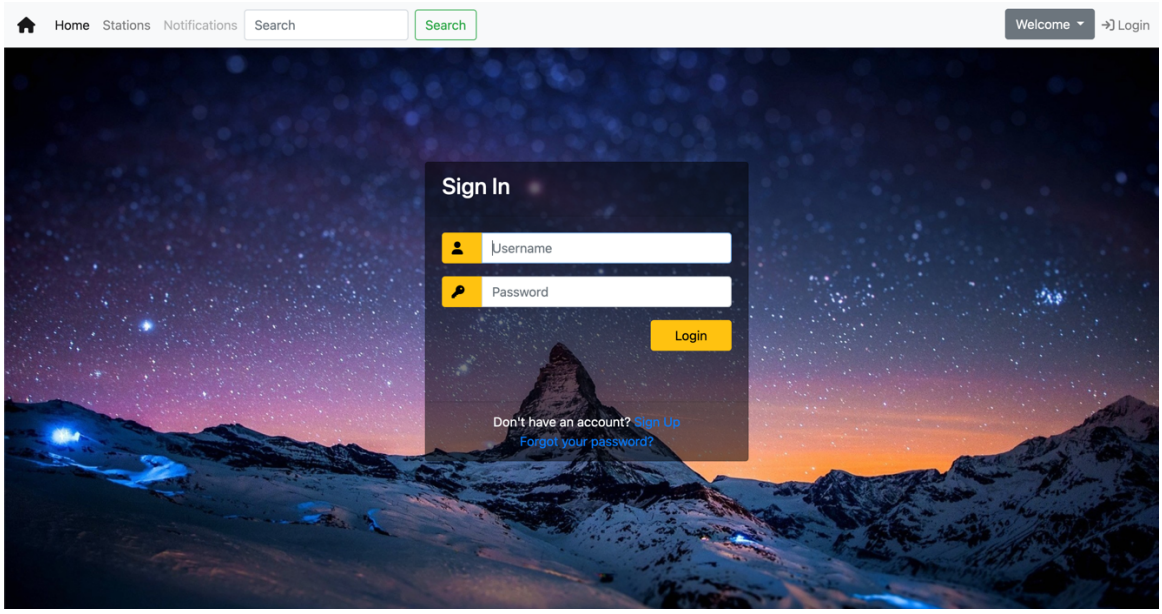
Để xây dựng một ứng dụng realtime cần sử dụng socketio. Socketio sẽ giúp các bên ở những địa điểm khác nhau kết nối với nhau, truyền dữ liệu ngay lập tức thông qua server trung gian. Socketio có thể được sử dụng trong nhiều ứng dụng như chat, game online, theo dõi đồ thị, …

Socketio không phải là một ngôn ngữ, mà chỉ là 1 công cụ giúp thực hiện những ứng dụng realtime. Vì thế, không thể sử dụng socket.io để thay thế hoàn toàn cho một ngôn ngữ, mà phải sử dụng kết hợp với một ngôn ngữ khác. Ngôn ngữ đó có thể là php, asp.net, nodejs,...

3.3.3. Giao diện web

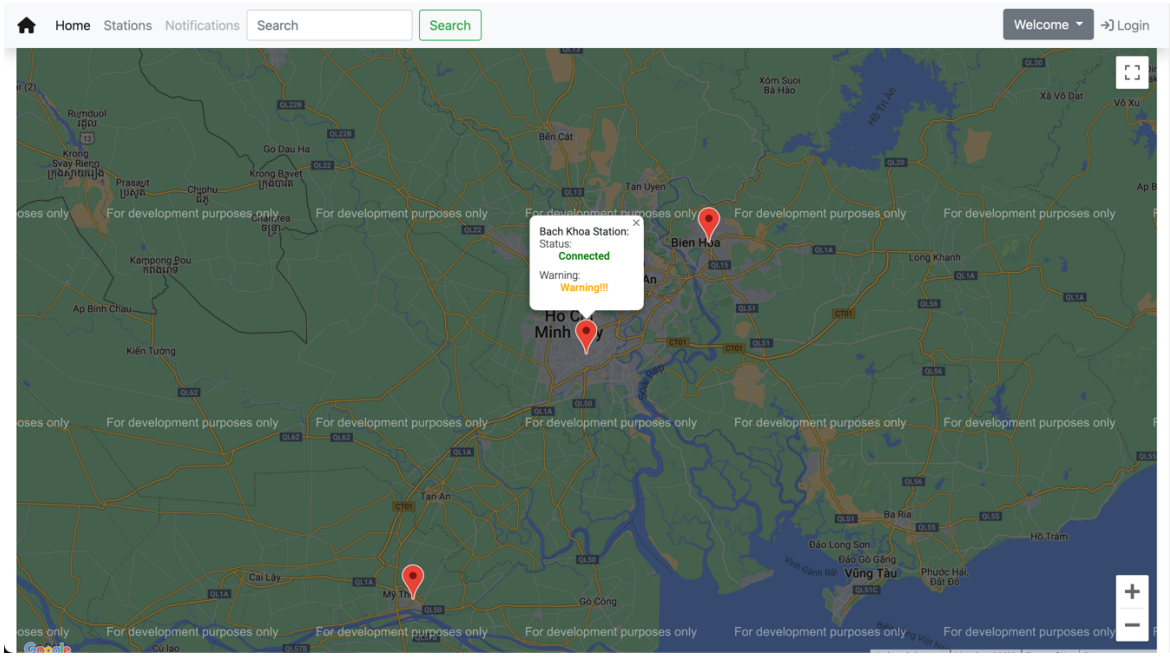
Phần giao diện ứng dụng web được thiết kế đơn giản, bao gồm: login, home, trend, alarm.

Login: màn hình đăng nhập vào ứng dụng web. Nếu cố tình truy cập vào web mà chưa đăng nhập thì màn hình đăng nhập sẽ xuất hiện đầu tiên. Sau khi đăng nhập thành công sẽ chuyển sang giao diện màn hình Home.



Hình 3.11: Giao diện login trên web

Home: hiển thị bản đồ có chứa vị trí các trạm nhằm thuận tiện cho việc giám sát các trạm từ xa.



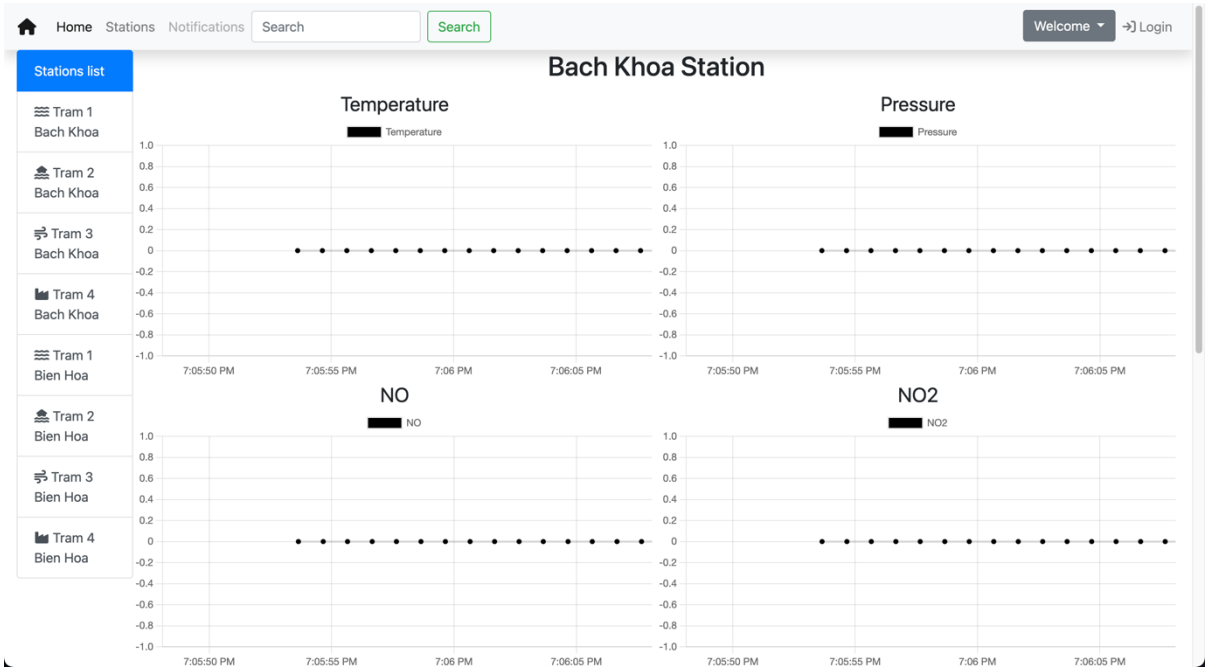
Hình 3.12: Giao diện màn hình Home trên web

Trên bản đồ có các điểm đánh dấu vị trí địa lý của các trạm. Khi nhấn vào các điểm đánh dấu trên bản đồ, thông tin của trạm sẽ hiển thị bao gồm: tên, trạng thái kết nối hay cảnh báo alarm.

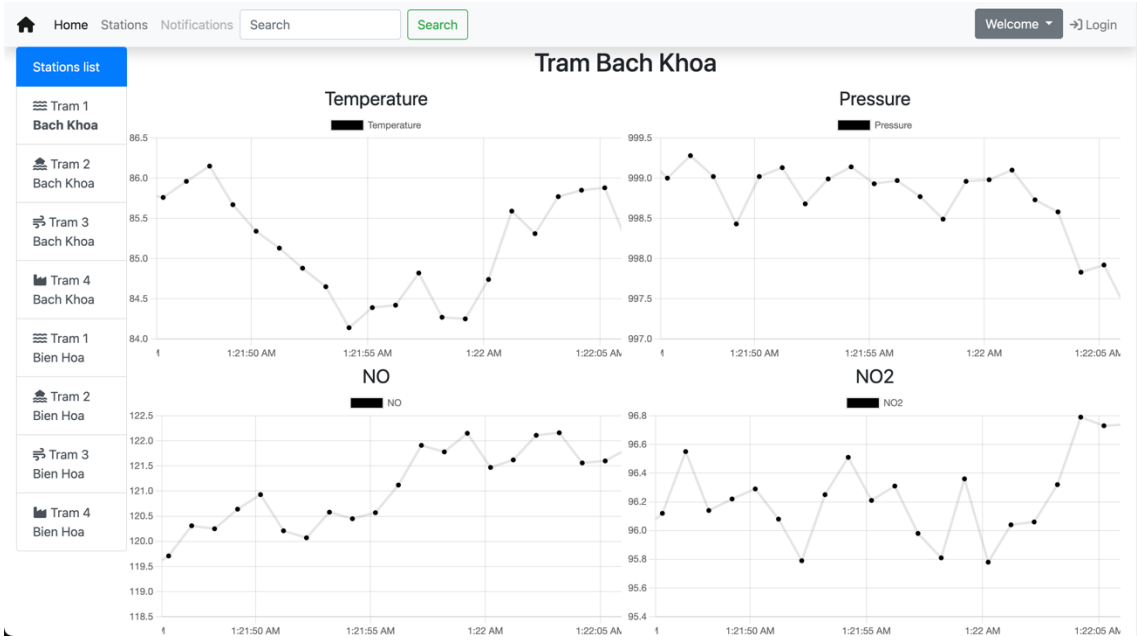
Nếu status là “Connected” nghĩa là OPC UA Client đã kết nối được tới các trạm quan trắc và ngược lại thì sẽ không xuất hiện trạng thái “Connected”.

Còn tín hiệu Warning để cảnh báo trong trạm đó có ít nhất một cảm biến đã vượt ngưỡng an toàn. Quản trị viên có thể mở sang trang Trend để biết chính xác loại cảm biến nào đã vượt ngưỡng hoặc chuyển sang trang alarm.

Trend: giao diện này có chức năng vẽ real time trend các thông số của từng trạm các nhau.



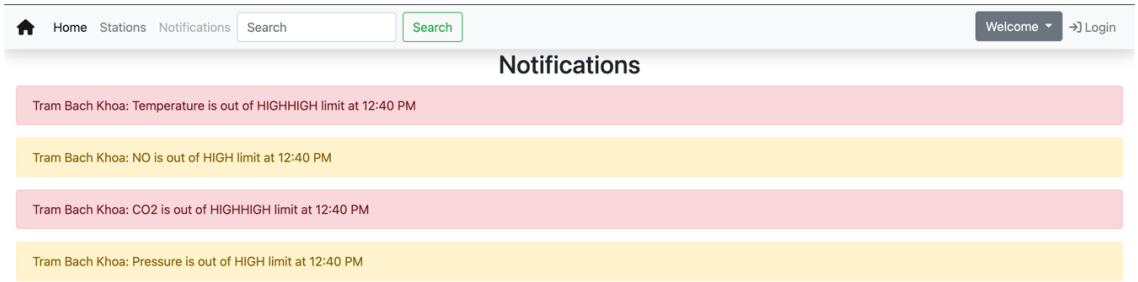
Hình 3.13: Giao diện Trend khi chưa kết nối với OPC Server



Hình 3.14: Giao diện màn hình Trend trên web

Dữ liệu thu thập được từ các trạm sau đó được vẽ theo thời gian thực nhờ vào socket.io. Khi bị mất kết nối tới OPC UA Server thì đồ thị sẽ đi ngang tại điểm cuối cùng mà nó nhận được dữ liệu. Người dùng có thể bấm sang trang Home để xác nhận chính xác trạng thái hiện tại của từng trạm.

Alarm: trang này sẽ hiển thị nội dung alarm của tất cả các trạm có trong hệ thống. Ta có thể xác nhận cảnh báo alarm (ACK alarm) khi đăng nhập dưới quyền quản trị viên.



Hình 3.15: Giao diện Alarm trên web

3.3.4. Xây dựng backend

Login

Trong phần login này, nhóm sử dụng phương thức POST để submit username và password về server bằng jQuery.

jQuery là một thư viện JavaScript đa tính năng, nhỏ gọn, nhanh, được tạo bởi John Resig vào năm 2006 với một phương châm hết sức ý nghĩa: Write less, do more - Viết ít hơn, làm nhiều hơn.

jQuery đơn giản hóa việc duyệt tài liệu HTML, xử lý sự kiện, hoạt ảnh và tương tác Ajax để phát triển web nhanh chóng. Các phân tích web đã chỉ ra rằng, jQuery là thư viện JavaScript được triển khai rộng rãi nhất.

jQuery là một bộ công cụ JavaScript được thiết kế để đơn giản hóa các tác vụ khác nhau bằng cách viết ít code hơn.

```
<div class="d-flex justify-content-center h-100">
    <div class="card" style="margin-top: 10%;">
        <div class="card-header">
            <h3>Sign In</h3>
        </div>
        <div class="card-body">
            <form method="POST" id="frmLogin" action="" >
                <div class="input-group form-group">
                    <div class="input-group-prepend">
                        <span class="input-group-text"><i class="fas fa-user"></i></span>
                    </div>
                    <input type="text" class="form-control" name="username" placeholder="Username" autofc

                </div>

                <div class="input-group form-group">
                    <div class="input-group-prepend">
                        <span class="input-group-text"><i class="fas fa-key"></i></span>
                    </div>
                    <input type="password" name="password" class="form-control" placeholder="Password" au
                </div>
                <div class="form-group">
                    <input type="submit" value="Login" class="btn float-right login_btn">
                </div>
```

Hình 3.16: Sử dụng phương thức POST để submit

Khi nút Login được nhấn thì biểu mẫu chứa thông tin gồm username và password được request lên server.

```
<script>
    $( "#frmLogin" ).on('submit', function( event ) {
        event.preventDefault();
        $("#frmLogin").off('submit').submit()
});
</script>
```

Hình 3.17: Submit biểu mẫu chứa username và password bằng jQuery

Server sau đó sẽ xác thực thông tin người dùng bằng cách tìm và xác username và password trong cơ sở dữ liệu MongoDB. Nếu thành công, bạn sẽ được trả về trang Home và sử dụng ứng dụng,.

```
LoginRoute.post('/', urlencodedParser, async function (req,
res) {
    const {username, password} = req.body
    if(!username || !password)
        return res
        .status(400)
        .json({status: false, message: 'Missing
username/password'})

    try {
        const user = await User.findOne({username})
        if (!user)
            return res.status(400).json({status: false,
message: "Invalid username"})

        // All good
        const unHashPass = argon2.verify(user.password,
password)
        if (!unHashPass){
            return res.status(400).json({status: false,
message: "Incorrect password"})
        } else {
                req.session.isAuthenticated = true;
                req.session.userAuth = user
                res.redirect('./home')
        }

    } catch (error) {
        return res.status(400).json({message: error})
    }
})
```

Home

Trong giao hiện bảo đồ này, nhóm sử dụng một API miễn phí do Google cung cấp. Google Maps API đã hỗ trợ nhiều ngôn ngữ khác nhau trên nhiều nền tảng. Để sử dụng chỉ cần lên trang chủ Google đăng kí tài khoản và Google sẽ cấp cho một key để sử dụng.

```
    function myMap() {
    var hcmut = {lat: 10.772895, lng: 106.659809}
    var bienhoa = {lat: 10.956918, lng: 106.865269}
    var mytho = {lat: 10.367031, lng: 106.368213}

    var map = new google.maps.Map(document.getElementById("googleMap"),{
        center: hcmut,
        zoom:10,
    })
    const marker = new google.maps.Marker({
        position: hcmut,
        map: map,
    })
    const marker2 = new google.maps.Marker({…
    const marker3 = new google.maps.Marker({…
    const infoWindow = new google.maps.InfoWindow({
        content: "<b>Bach Khoa Station:</b> <br> Status: <h2></h2> Warning: <h3></h3>",
        disableAutoPan: true,
    });
    const infoWindow2 = new google.maps.InfoWindow({…
    const infoWindow3 = new google.maps.InfoWindow({…

    marker.addListener("click", () => {
      infoWindow.open(map, marker);
    });
    marker2.addListener("click", ()=>{…
    marker3.addListener("click", ()=>{…
}
</script>
<script async defer src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAzvltrCjv8ibLJoTqfRtTH78thyf8h
```

Hình 3.18: Sử dụng Google Maps API

Tiếp theo chỉ là khai báo một map mới lẫn key vừa nhận được từ Gooogle. Sau đó thực hiện các tính năng cơ bản như tạo marker học pop-up inforWindow khi nhấn vào marker.

Trend

Trong phần Trend này, nhóm sử dụng thư viện vẽ chart tên là Chart.js. Đây là một trong những dự án mã nguồn mở giúp cho mọi người có thể vẽ những biểu đồ thể hiện số liệu trên website một cách dễ dàng và đẹp nhất. Dự án này hiện tại đã có đến hơn 41.000 stars và 2600 lượt commit trên Github và được cập nhật thường xuyên. 4 điểm mạnh nhất của Chart.js là: – Dự án mã nguồn mở: cả cộng đồng phát triển và khắc phục lỗi. – Tương thích tốt với HTML 5 cái này

gần như bắt buộc ở hiện tại – Hơn 8 kiểu biểu đồ phổ biến nhất hiện nay – Responsive: hiển thị đẹp nhất trên tất cả các thiết bị từ Desktop, Tablet, Mobile.

```
<script src="./socket.io/socket.io.js"></script>
<script>
        var socket = io()

        function updateValue() {
            socket.on("exchange-data", function(data){

                for (i=0; i<11; i++){
                     const indicator = [];
                     indicator[i] = data.data[0][i].indicator
                     $("#"+
indicator[i]).html(data.data[0][i].value)
                      }
            })
        }
        updateValue();

        $(document).ready(function(){
        });

</script>
```

Dữ liệu nhận được từ socket.io sẽ được biểu diễn lên chart theo thời gian thực. Việc sử dụng socket.io giúp cải thiện độ trễ từ các trạm đến giao diện người dùng.

```
<div style="font-size: x-large; text-align: center;"
id="text1">Temperature</div>
        <canvas id="myChart" style="width:100%;max-
width:700px"></canvas>
               <script>
            new Chart("myChart", {
            type: "line",
            data: {
                datasets: [{
                    label: "Temperature",
                    backgroundColor: "rgba(0,0,0,1.0)",
                    borderColor: "rgba(0,0,0,0.1)",
                    data: []
                }]
            },
            options: {
                legend: {display: false},
                scales: {
                    x: {
                        type: 'realtime'
                        realtime: {
                        duration: 20000,
                        refresh: 1000,
                        delay: 1000,
                        pause: false,
                        ttl: undefined,
                        frameRate: 30,
                        // a callback to update datasets
                        onRefresh: chart => {

chart.data.datasets.forEach(dataset => {
                                    dataset.data.push({
                                    x: Date.now(),
                                    y:
document.getElementById("Temperature").innerHTML
                                    })
                                });
                        }
                        }
```

Alarm:

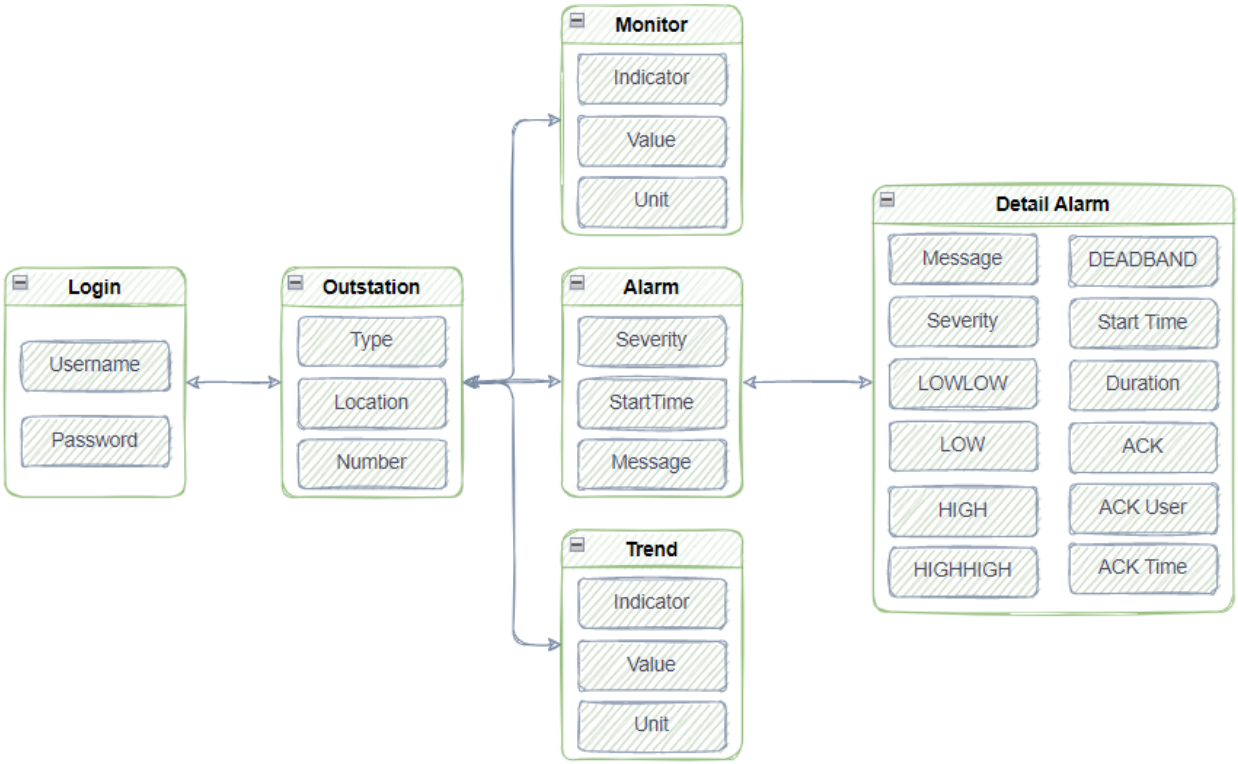
Thông tin về Alarm đã được xử lý phía OPC UA Client và trả về một chuỗi dạng JSON các chứa các thông tin về mức độ nguy hiểm lẫn nội dung alarm. Nên việc còn lại chỉ là thêm các cảnh báo vào mảng và render nó lên giao diện người dùng.

```
        function checkAlarm (){
              socket.on("exchange-data", function(data){
                var listAlarm = []
                for (i=0; i<11;i++){
                  listAlarm[i] = data.data[0][i]
                }

              function addAlarm(){
                     const alarmItem =
     document.createElement('li');
                alarmItem.textContent = listAlarm[1].content
                myList.appendChild(alarmItem)
              }
              })
        }
```

3.4. Xây dựng ứng dụng Android

3.4.1. Kiến trúc phần mềm

Ứng dụng di động giám sát các trạm quan trắc được xây dựng cho hệ điều hành Android sử dụng ngôn ngữ Java. Phần mềm có kiến trúc như sau:



Hình 3.19: Kiến trúc phần mềm giám sát trên Android

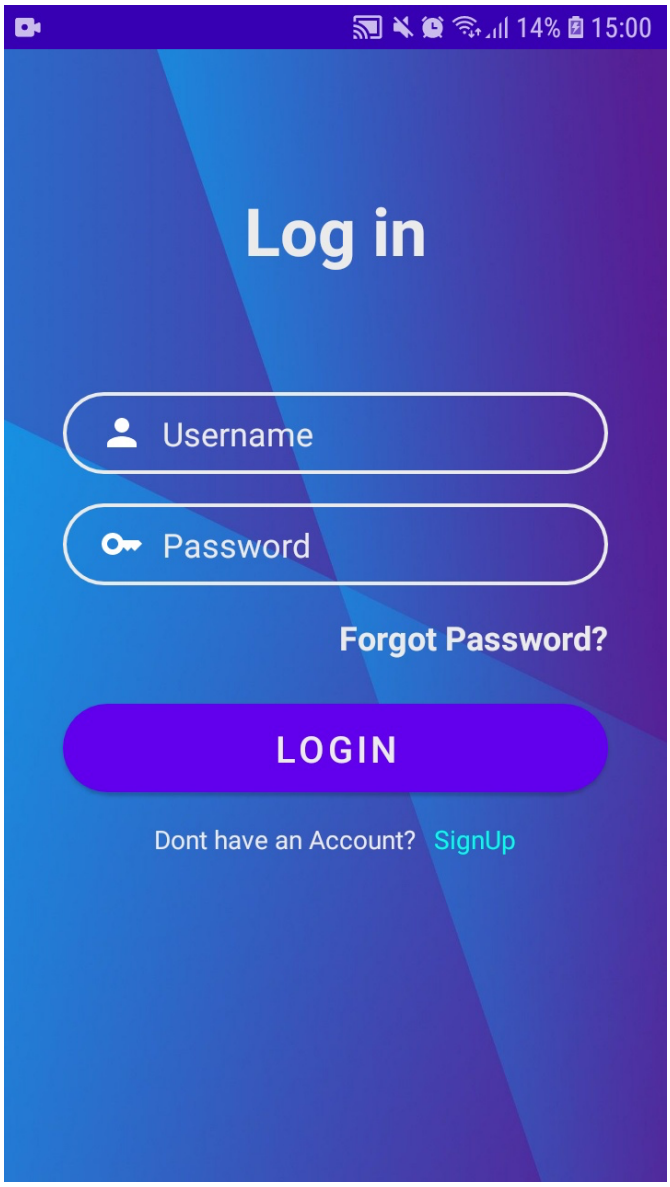
Theo sơ đồ trên, đầu tiên người dùng sẽ thực hiện đăng nhập với một giao diện Login. Sau khi đăng nhập thành công, phần mềm sẽ mở ra một giao diện danh sách các trạm quan trắc đồng thời thông tin của người đăng nhập cũng sẽ được chuyển vào theo để xử lý cho việc phân quyền. Trong luận văn chỉ áp dụng đơn giản 2 quyền, quyền user chỉ có thể đọc và quyền admin có thể thực hiện thao tác điều khiển (ghi dữ liệu) và xác nhận alarm.

Trong giao diện danh sách trạm, khi người dùng chọn vào 1 trạm, phần mềm sẽ mở ra một giao diện có thể trượt theo chiều ngang với 3 màn hình: Monitor, Trend và Alarm. Đồng thời sẽ gửi thông tin trạm mà người dùng đã

chọn như: loại trạm (khí thải, nước thải,...), vị trí trạm (KCN Long Thành, KCN Biên Hòa,...) và số thứ tự của trạm (Trạm 1, Trạm 2,...) đến những giao diện tiếp theo xử lý những dữ liệu của trạm đó.

Giao diện đăng nhập

Trước khi vào được màn hình chính của ứng dụng, ta cần phải đăng nhập với username và password đã được đăng ký và lưu trong Database.



Hình 3.20: Giao diện đăng nhập trên Android

3.4.2. Giao diện danh sách trạm

Hình 3.21: Giao diện danh sách các trạm trên Android

Giao diện danh sách trạm gồm một thanh toolbar và một danh sách các trạm quan trắc.

Trong đó, thanh toolbar gồm:

+ Một nút nhấn back: khi nhấn nút nhấn này sẽ thoát ứng dụng.

+ Một thanh tìm kiếm: để lọc và tìm kiếm các trạm quan trắc.

+ Một nút nhấn có icon account: gồm 2 chức năng là xem thông tin người dùng hiện tại và log out ra khỏi tài khoản, sau đó trở về giao diện Log in.

Các trạm trong danh sách bao gồm:

+ Một icon thể hiện loại trạm, ví dụ như: khí thải, nước thải, nước mặt,...

+ Hai đoạn text: thể hiện thông tin số thứ tự và vị trí của trạm.

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/black"
    android:orientation="vertical"
    >

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/tb_outstation"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/black"
        app:contentInsetLeft="0dp"
        app:contentInsetStart="0dp"
        app:contentInsetStartWithNavigation="0dp"

android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
        app:title="OUTSTATION"/>

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rcv_OutstationList"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginStart="30dp"
        android:layout_marginEnd="20dp"/>
</LinearLayout>
```

Phần view của giao diện danh sách trạm nằm trong file *activity_main.xml,* chứa 2 widget là Toolbar và RecyclerView. RecyclerView là một viewgroup có chức năng hiển thị một list các dữ liệu tương tự như ListView, tuy nhiên ưu điểm nổi bật của RecyclerView là nó sẽ chỉ thực hiện phần code behind tương ứng với từng view con trong list khi nó được trượt tới và hiển thị trên màn hình. Ưu điểm này từ đó làm tăng hiệu năng của ứng dụng, giúp giảm bớt tính toán cho những thành phần view con không được hiển thị.

Để tùy chỉnh thanh công cụ toolbar, ta sẽ tạo một file *menu_search_outstation.xml* trong thư mục menu bao gồm 2 item là Search và Account.

```
<menu
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item android:id="@+id/menu_search"
        android:title="Search"
        android:icon="@drawable/ic_search"
        app:showAsAction="always"

app:actionViewClass="androidx.appcompat.widget.SearchView"/>
    <item android:id="@+id/menu_account"
        android:icon="@drawable/ic_account"
        android:title="Account"
        app:showAsAction="ifRoom"/>
</menu>
```

Tiếp theo, ta sẽ thiết kế layout cho dữ liệu mà RecyclerView sẽ hiển thị. Layout này nằm trong thư mục *layout_outstation_list.xml*, bao gồm một ImageView là icon thể hiện loại trạm và 2 TextView thể hiện số thứ tự và vị trí của trạm. Ta cần thêm thuộc tính *android:id* để có thể truy cập trong phần code behind.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center_vertical"
    android:orientation="horizontal">

    <ImageView
        android:id="@+id/rcvimg_icon"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:src="@drawable/ic_khithai" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="20dp"
        android:orientation="vertical">

        <TextView
            android:id="@+id/rcvtv_outstationNum"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:maxWidth="110dp"
            android:text="Tram 1"
            android:textColor="@color/teal_200"
            android:textSize="18sp" />

        <TextView
            android:id="@+id/rcvtv_location"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="KCN Long Thanh"
            android:textColor="@color/teal_200"
            android:textSize="22sp" />
    </LinearLayout>
</LinearLayout>
```

Để binding dữ liệu lên RecyclerView, ta cần tạo một class Outstation cho đối tượng mà ta cần hiển thị và một OutstationAdapter tương ứng.

```
public class Outstation implements Serializable {

    private int id;
    private int resourceId;
    private String rcv_outstationNumber;
    private String rcv_location;
```

Class Outstation cần được implements một interface là Serializable để có thể bắt được sự kiện khi một item trong danh sách các trạm được nhấn.

Phần code behind xử lý logic cho giao diện danh sách trạm nằm trong *MainActivity.java*. Đầu tiên, ta sẽ mở một kết nối websocket đến VPS và tạo một topic để nhận danh sách các trạm mà VPS gửi tới, sau đó gán giá trị vào một danh sách các instance của class Outstation mà ta đã tạo trước đó.

```java
private Socket mSocket; {
    try {
        mSocket = IO.socket("http://20.205.122.62:3000");
    } catch (URISyntaxException e) {}

}

Emitter.Listener readListOutstation = new Emitter.Listener()
{
@Override
public void call(final Object... args) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            JSONObject data = (JSONObject) args[0];

            try {
                JSONArray inforList = data.getJSONArray("infor");
                int soTram = inforList.length();

                if(outstationApdater.getItemCount() == soTram) return;

                for(int i=0; i<soTram; i++) {
                    if(outstationApdater.getItemCount() != soTram) {
                        outstationApdater.getmListOutstation().add(new
                                        Outstation());
                    }

                    Outstation jvData;
                    jvData = outstationApdater.getOutstationByIndex(i);
                    JSONObject os = inforList.getJSONObject(i);
                    jvData.setId(os.getInt("id"));

                    if (os.getInt("OS_Type") == 0) {
                        jvData.setResourceId(R.drawable.ic_khithai);
                    } else if (os.getInt("OS_Type") == 1) {
                        jvData.setResourceId(R.drawable.ic_nuocthai);
                    } else if (os.getInt("OS_Type") == 2) {
                        jvData.setResourceId(R.drawable.ic_khongkhi);
                    } else if (os.getInt("OS_Type") == 3) {
                        jvData.setResourceId(R.drawable.ic_nuocmat);
                    }


                    jvData.setRcv_outstationNumber(
                              os.getString("OS_Number"));
                    jvData.setRcv_location(os.getString("OS_Location"));
                }
                outstationApdater.notifyDataSetChanged();

            } catch (JSONException e) {
                return;
            }
        }
    });
}
};
```

Tiếp theo, ta sẽ ánh xạ View tới 2 widget Toolbar và RecyclerView với android:id tương ứng bằng hàm findViewById.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    tb_outstation = findViewById(R.id.tb_outstation);
    rcv_outstation = findViewById(R.id.rcv_OutstationList);
```

Xử lý sự kiện khi người dùng nhấn vào một trạm trong danh sách được thực hiện bằng cách gọi hàm *onClickGoToDetail*.

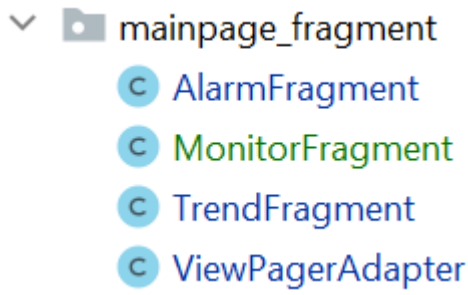
```
private void onClickGoToDetail(Outstation outstation) {
    Intent intent = new Intent(this, MonitorActivity.class);
    Bundle bundle = new Bundle();
    bundle.putSerializable(
          "MainActi_To_MonitorActi_Outstation", outstation);
    intent.putExtras(bundle);
    startActivity(intent);
}
```

Chỉ có một cách duy nhất để truyền dữ liệu giữa các Activity trong lập trình Android, đó là sử dụng Intent. Dữ liệu được đóng gói vào trong Intent và được lấy ra khỏi Intent theo các cặp dữ liệu dạng key-value. Trong đó, key là một chuỗi định danh độc nhất và value sẽ là object mà chúng ta cần truyền. Ở đây ta sẽ sử dụng cơ chế này để truyền đối tượng outstation mà được người dùng nhấn trong danh sách các trạm đến một activity có tên là MonitorActivity.

MonitorActivity là một activity bao gồm các thành phần như: Toolbar, ViewPager2 và BottomNavigationView.

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MonitorActivity">

    <androidx.appcompat.widget.Toolbar ...>

    <androidx.viewpager2.widget.ViewPager2 ...>

    <com.google.android.material.bottomnavigation.
         BottomNavigationView ...>

</RelativeLayout>
```

ViewPager2 kết hợp với BottomNavigationView sẽ cung cấp khả năng lướt để thay đổi giữa các giao diện khác nhau theo chiều ngang.



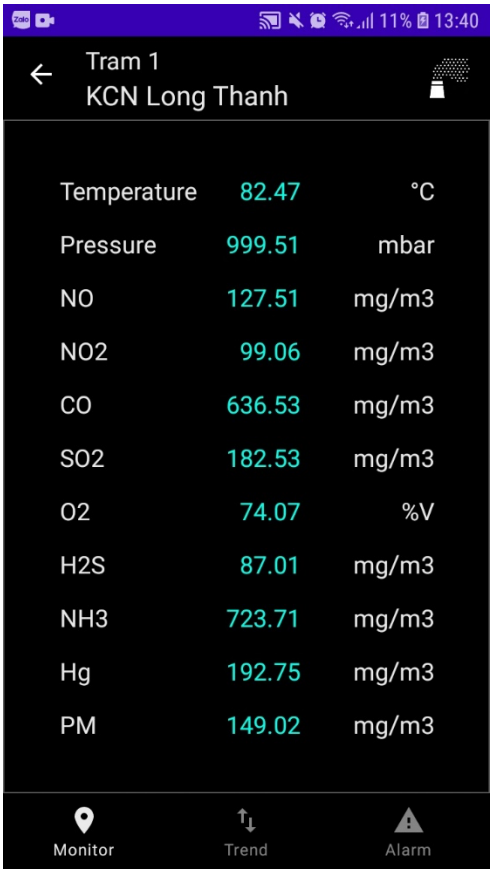
Hình 3.22: Code logic cho giao diện con của ViewPager2

ViewPager2 chịu trách nhiệm thay thế fragment tương ứng vào giao diện của MonitorActivity mỗi khi nó bắt được sự kiện lướt sang trái hoặc sang phải. Fragment chịu trách nhiệm quản lý một không gian màn hình, nhiều khi không gian này cũng chính là toàn màn hình. Và không gian màn hình đó của Fragment phải nằm trong một Activity nào đó. Một Activity có thể có nhiều Fragment, có khi nhiều Fragment của Activity đó cùng nhau hiển thị lên một màn hình, cũng có khi chúng luân phiên hiển thị nếu như mỗi chúng đều chiếm cả không gian màn hình. Và một ý nữa, một Fragment nào đó cũng có thể được khai báo và sử

dụng bởi nhiều Activity khác nhau. Ở đây, ViewPager2 sẽ quản lý giao diện được hiển thị cho 3 fragment: MonitorFragment, TrendFragment và AlarmFragment.

3.4.3. Giao diện giám sát dữ liệu thời gian thực

Phần giao diện giám sát dữ liệu thời gian thực nằm trong file *fragment_monitor.xml*, nó chỉ bao gồm một RecyclerView để hiển thị danh sách những thông số của trạm.



Hình 3.23: Giao diện giám sát dữ liệu thời gian thực trên Android

Để binding dữ liệu lên RecyclerView, ta cần tạo một class Data cho đối tượng mà ta cần hiển thị và một MonitorAdapter tương ứng.

```java
public class Data implements Serializable {

    private int id;
    private String indicator;
    private float value;
    private String unit;
    private float LOWLOW;
    private float LOW;
    private float HIGH;
    private float HIGHHIGH;
    private float DEAD_BAND;
    private int severity;
    private int pre_severity;
    private long startTime;
    private long duration;
    private long pre_duration;
    private String content;
    private boolean ACK;
    private String ackUser;
    private long ACKTime;
```

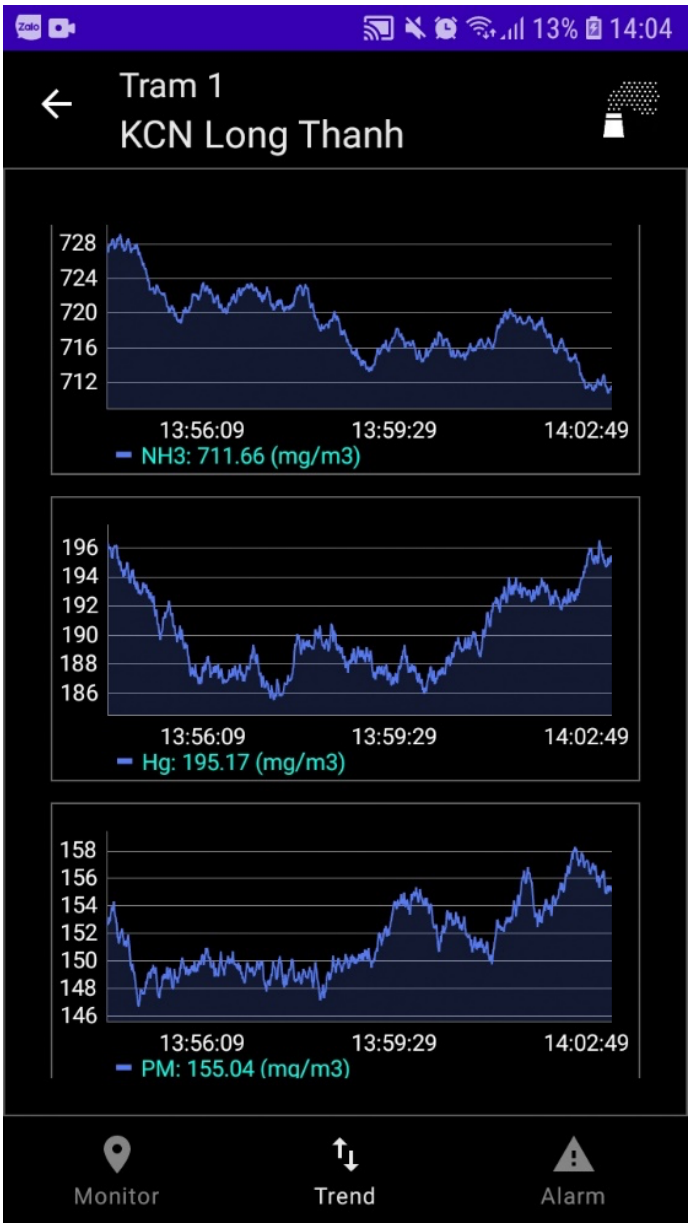
Class Data bao gồm các thuộc tính như hình trên và các hàm constructor, getter, setter tương ứng.

Phần code behind của giao diện giám sát dữ liệu thời gian thực sẽ nằm trong file *MonitorFragment.java*. Đầu tiên, ta sẽ tạo một topic cho websocket client để nhận dữ liệu thông số của trạm quan trắc tương ứng mà ta đã chọn ở giao diện danh sách trạm quan trắc. Gán dữ liệu nhận được vào một danh sách các instance của class Data.

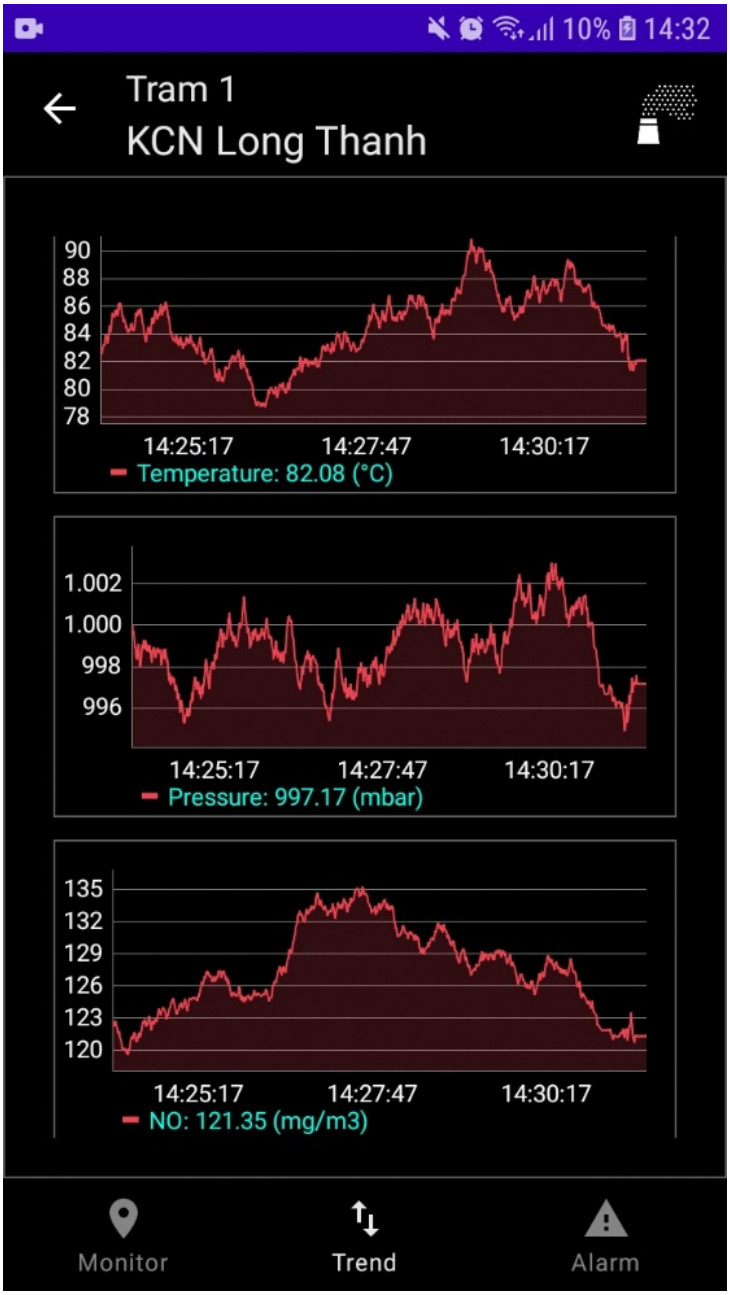
```
JSONObject data = (JSONObject) args[0];
try {
    JSONArray dataList = data.getJSONArray("data");
    JSONArray indicatorList = dataList.getJSONArray(outstation.getId());
    int chisoCount = indicatorList.length();
    DataApdater dataApdater = (DataApdater)        rcv_Datalist.getAdapter();
    if(dataApdater ==null) return;
    for(int i=0; i<chisoCount; i++) {
        if(dataApdater.getItemCount() != chisoCount) {
            dataApdater.getListData().add(new Data());
        }
        Data jvData;
        jvData = dataApdater.getDataByIndex(i);

        JSONObject indicator =
         indicatorList.getJSONObject(i);
        jvData.setId(i);
        jvData.setIndicator(

                 indicator.getString("indicator"));
        jvData.setValue((float)
         indicator.getDouble("value"));
        jvData.setUnit(indicator.getString("unit"));
        jvData.setLOWLOW((float)
         indicator.getDouble("LOWLOW"));
        jvData.setLOW((float) indicator.getDouble("LOW"));
        jvData.setHIGH((float) indicator.getDouble("HIGH"));
        jvData.setHIGHHIGH((float)
         indicator.getDouble("HIGHHIGH"));
        jvData.setDEAD_BAND((float)
         indicator.getDouble("DEAD_BAND"));
        jvData.setSeverity(indicator.getInt("severity"));
        jvData.setPre_severity(

                 indicator.getInt("pre_severity"));
        jvData.setStartTime(indicator.getLong("startTime"));
        jvData.setDuration(indicator.getLong("duration"));
        jvData.setPre_duration(

                 indicator.getLong("pre_duration"));
        jvData.setContent(indicator.getString("content"));
        jvData.setACK(indicator.getBoolean("ACK"));
        jvData.setAckUser(indicator.getString("ackUser"));
        jvData.setACKTime(indicator.getLong("ACKTime"));
    }
```

3.4.4. Giao diện trend dữ liệu lịch sử

Trong quá trình giám sát dữ liệu, nếu chỉ giám sát dữ liệu thời gian thực thì ta sẽ không thể biết được xu hướng thay đổi hoặc đôi khi ta sẽ bỏ qua những thời điểm dữ liệu vượt ra khỏi vùng an toàn, bởi vì ta không phải lúc nào cũng giám sát dữ liệu trong toàn bộ thời gian của mình. Giao diện trend sẽ giúp ta theo dõi được sự thay đổi của dữ liệu trong lịch sử.



Hình 3.24: Giao diện mobile trend khi có kết nối



Hình 3.25: Giao diện mobile trend khi bị mất kết nối

Giao diện trend chỉ bao gồm một ListView chứa danh sách trend của các chỉ số trạm quan trắc. Ở đây ta sử dụng ListView mà không sử dụng RecyclerView bởi vì ta luôn cần cập nhật dữ liệu cho những view con dù chúng không được hiển thị trên màn hình. Giao diện trend có tích hợp dữ liệu thời gian thực trong đồ thị để tiện theo dõi và chức năng đổi sang màu đỏ để nhận biết khi có sự cố mất kết nối với server.

Có rất nhiều thư viện hỗ trợ vẽ chart trên Android, trong luận văn sẽ sử dụng thư viện MPAndroidChart với hơn 35 ngàn lượt đánh giá sao trên github. Đầu tiên, trong file TrendAdapter.java ta sẽ viết một hàm khởi tạo cho những trend trong danh sách.

```
private void initLineChart(LineChart chart, Data object, Context
context) {
    // enable description text
    chart.getDescription().setEnabled(false);
    // enable touch gestures
    chart.setTouchEnabled(false);
    // enable scaling and dragging
    chart.setDragEnabled(false);
    chart.setScaleEnabled(true);
    chart.setDrawGridBackground(false);
    chart.setDoubleTapToZoomEnabled(false);
    chart.setPinchZoom(false);
    LineData data = new LineData();
    data.setValueTextColor(Color.WHITE);
    // add empty data
    chart.setData(data);
    // get the legend (only possible after setting data)
    Legend l = chart.getLegend();
    // modify the legend ...
    l.setForm(Legend.LegendForm.LINE);
    l.setTextColor(ContextCompat.getColor(context,
        R.color.teal_200));
    l.setTextSize(12f);
    XAxis xl = chart.getXAxis();
    xl.setTextColor(Color.WHITE);
    xl.setTextSize(12f);
    xl.setDrawGridLines(false);
    xl.setAvoidFirstLastClipping(true);
    xl.setPosition(XAxis.XAxisPosition.BOTTOM);
    xl.setLabelCount(3);
    xl.setCenterAxisLabels(true);
    xl.setGranularity(1f);
    xl.setEnabled(true);
    YAxis leftAxis = chart.getAxisLeft();
    leftAxis.setTextColor(Color.WHITE);
    leftAxis.setTextSize(12f);
    leftAxis.setDrawGridLines(true);

    YAxis rightAxis = chart.getAxisRight();
    rightAxis.setEnabled(false);
}
```

Để vẽ được những điểm dữ liệu lên đồ thị, ta cần viết một hàm addEntry nhận đầu vào là đối tượng chart nó sẽ vẽ lên và một đối tượng Data để cung cấp dữ liệu. Đối tượng Data sẽ được lấy từ danh sách những đối tượng Data trong MonitorFragment, đây cũng chính là nguồn data-source chứa topic lấy dữ liệu từ VPS.

```
private void addEntry(LineChart chart, Data object) {
```

Trong hàm addEntry, đầu tiên ta sẽ xử lý để lấy được thời điểm chart bắt đầu vẽ dữ liệu, sau đó xử lý để hiện thị thời gian lên trục X của đồ thị.

```
if(set.getEntryCount() == 0) {
    START_TIME = System.currentTimeMillis();
    XAxis xl = chart.getXAxis();
    xl.setValueFormatter(new ValueFormatter() {
        @Override
        public String getFormattedValue(float value) {
                return (new SimpleDateFormat("HH:mm:ss",
Locale.ENGLISH).format(new Date(START_TIME + (long) value *
DELAY_TIME)));
        }
    });
```

Tiếp theo ta sẽ xử lý nếu số lượng điểm dữ liệu trên đồ thị nhiều hơn một giá trị định trước là VISIBLE_NUM thì ta sẽ xóa điểm dữ liệu đầu tiên để giữ cho số lượng điểm dữ liệu trên đồ thị luôn là một hằng số. Mục đích của việc này là để tránh tràn bộ nhớ cho thiết bị di động.

```
if(set.getEntryCount() > VISIBLE_NUM) {
    set.removeFirst();
    for(int i = 0; i < set.getEntryCount(); i++) {
        Entry entry = set.getEntryForIndex(i);
        entry.setX(entry.getX() -1);
    }
```

Cuối cùng ta sẽ lấy dữ liệu từ đối tượng Data, thêm nó vào danh sách điểm dữ liệu của đồ thị và gọi hàm xác nhận đã có sự thay đổi trong danh sách trend để TrendAdapter có thể cập nhật dữ liệu lên giao diện.

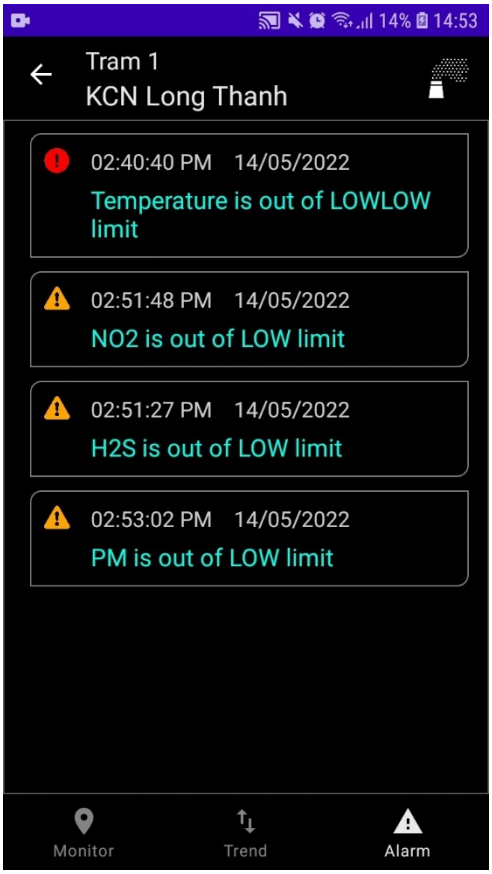
```
float y_value = object.getValue();
data.addEntry(new Entry(set.getEntryCount(), y_value), 0);
set.setLabel(object.getIndicator() + ": " +
Float.toString(object.getValue()) + " (" + object.getUnit()
+ ")");
data.notifyDataChanged();
chart.notifyDataSetChanged();
chart.setVisibleXRangeMaximum(VISIBLE_NUM);
chart.moveViewToX(data.getEntryCount());
```

Sau khi đã viết xong hàm addEntry, ta sẽ tạo một luồng mới và gọi hàm trong luồng đó mỗi 500ms. Ta cần hủy luồng mỗi khi thoát khỏi giao diện trend để tránh tràn bộ nhớ.

```
thread = new Thread(new Runnable() {
   @Override
       public void run() {
          while (true) {
             addEntry(chart, data);
             try {
                 Thread.sleep(DELAY_TIME);
             } catch (InterruptedException e) {
                 e.printStackTrace();}}
      }
});
@Override
public void onDestroy() {
    super.onDestroy();
    if (thread != null) {
        thread.interrupt();}}
```

3.4.5. Giao diện alarm thời gian thực

Giao diện alarm thời gian thực sẽ cảnh báo cho người dùng biết chỉ số nào đang bị vượt quá các mức ngưỡng được đặt trước. Có nhiều vấn đề xoay quanh việc quản lý alarm như: bắt được những alarm chưa được xác nhận nhưng đã bị mất đi, giữ được thông tin user đã xác nhận alarm, sử dụng deadband để tránh "spam" alarm liên tục,...



Hình 3.26: Giao diện danh sách alarm trên Android

Giao diện alarm thời gian thực cung cấp một danh sách những alarm với thông tin như: mức độ cảnh báo (alert, alarm,...), thời gian alarm bắt đầu và thông báo của alarm.

Đầu tiên, ta sẽ thiết kế layout cho những phần tử trong danh sách alarm.

```
        <ImageView
             android:id="@+id/img_alarm_icon"
             android:layout_width="20dp"
             android:layout_height="20dp"
             android:layout_alignParentStart="true"
             android:layout_marginStart="10dp"
             android:layout_marginTop="10dp"
             android:adjustViewBounds="true"
             android:maxWidth="45dp"
             android:maxHeight="45dp"
             android:scaleType="fitCenter"
             android:src="@drawable/ic_alert" />
        <TextView
            android:id="@+id/tv_alarm_datetime"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="15dp"
            android:layout_marginTop="10dp"
            android:layout_marginBottom="5dp"
            android:layout_toEndOf="@id/img_alarm_icon"
            android:text="dd/MM/yyyy hh:mm:ss aa"
            android:textColor="@color/white_700"
            android:textSize="16sp" />
        <TextView
            android:id="@+id/tv_alarm_content"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/tv_alarm_datetime"
            android:layout_marginStart="15dp"
            android:layout_marginBottom="10dp"
            android:layout_toEndOf="@+id/img_alarm_icon"
            android:text="Pressure exceeded HIGH limit!!!"
            android:textColor="@color/teal_200"
            android:textSize="18sp" />
```

Phần xử lý mức độ alarm, thời điểm bắt đầu xảy ra alarm cũng như khoảng thời gian alarm đã xảy ra, đã được xử lý ở VPS. Nhiệm vụ của chúng ta là lấy danh sách những chỉ số từ data-source là MonitorFragment và lọc ra một danh sách những chỉ số nào đang bị alarm và hiển thị lên giao diện.

Đầu tiên, ta sẽ duyệt qua danh sách alarm cũ, để bắt được những alarm chưa được xác nhận những đã tự mất đi và sau đó gửi chúng về lại VPS để tiến hành ghi nhận vào database.

```
for(int i=0; i<alarmDataList.size(); i++) {
    Data data = alarmDataList.get(i);
    if(data.getSeverity() != data.getPre_severity() &&
(data.getPre_severity() == 1 || data.getPre_severity() == 2)) {
        JSONObject object = new JSONObject();
        try {
            object.put("indicator", data.getIndicator());
            object.put("value", data.getValue());
            object.put("unit", data.getUnit());
            object.put("LOWLOW", data.getLOWLOW());
            object.put("LOW", data.getLOW());
            object.put("HIGH", data.getHIGH());
            object.put("HIGHHIGH", data.getHIGHHIGH());
            object.put("DEAD_BAND", data.getDEAD_BAND());
            object.put("severity", data.getPre_severity());
            object.put("startTime", data.getStartTime());
            object.put("duration", data.getPre_duration());
            object.put("content", data.getContent());
            object.put("ACK", data.getACK());
            object.put("ackUser", data.getAckUser());
            object.put("ACKTime", data.getACKTime());
        } catch (JSONException e) {
            e.printStackTrace();
        }
        mSocket.emit("alarmLog", object);
    }
}
```

Sau đó ta sẽ lọc từ danh sách những chỉ số của MonitorFragment những chỉ số nào đang có mức độ cảnh báo là alert (severity=2) và alarm (severity=1). Tuy nhiên danh sách alarm sẽ không hiển thị những chỉ số đã được xác nhận ACK để tránh tràn danh sách.

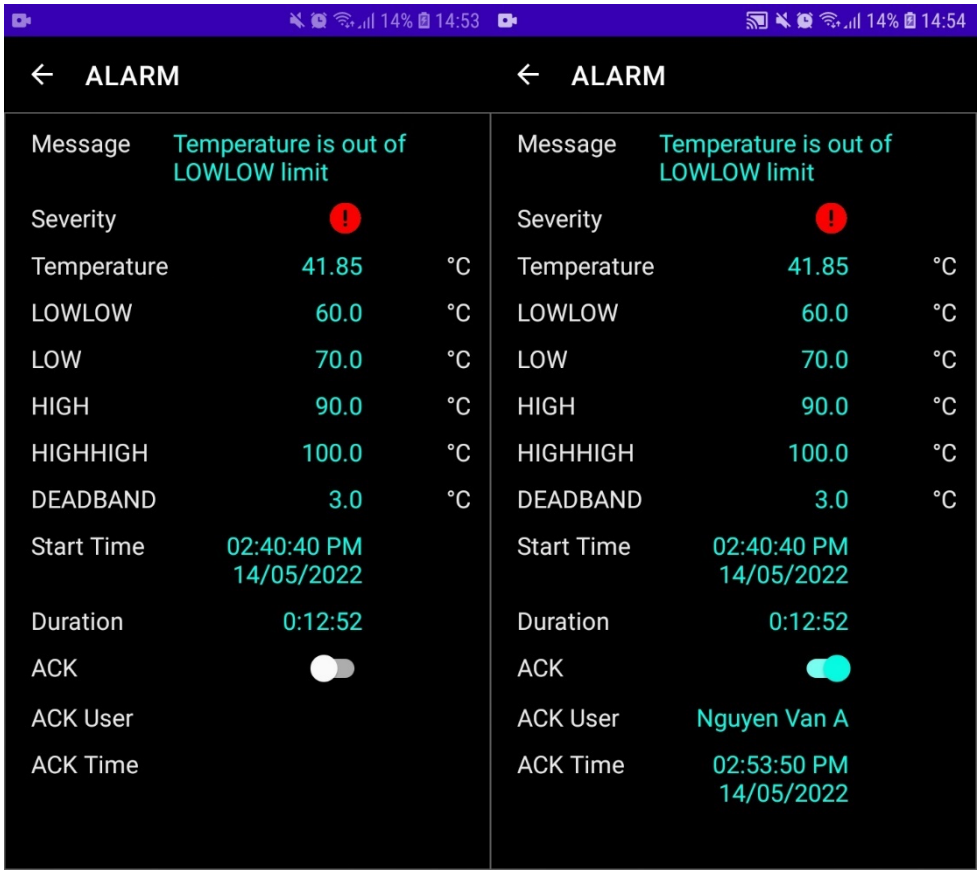
```
alarmDataList.clear();

for(int i=0; i< dataList.size(); i++) {
    if(!dataList.get(i).getACK() &&
                        (dataList.get(i).getSeverity() == 1 ||
                        dataList.get(i).getSeverity() == 2)) {

       alarmDataList.add(dataList.get(i));
    }
}
```

Cuối cùng, ta sẽ xử lý để bắt sự kiện nhấn vào một alarm trong danh sách, sẽ mở ra giao diện Alarm Detail.

```
private void ocClickGoToAlarmDetail(Data data) {
    Intent intent = new Intent(getContext(),
                            DetailAlarmActivity.class);
    Bundle bundle = new Bundle();
    bundle.putSerializable(

       "AlarmFrg_To_DetailAlarmActi_Data", data);
```

3.4.6. Giao diện thông tin alarm

Giao diện thông tin alarm sẽ cung cấp những thông tin chi tiết hơn về các cảnh báo như: giá trị chỉ số gây ra alarm, các mức ngưỡng, khoảng thời gian alarm đã xảy ra.



Hình 3.27: Giao diện thông tin alarm trên Android

Ngoài ra, giao diện còn cung cấp một switch để người dùng được phân quyền admin có thể xác nhận alarm. Sau khi xác nhận, giao diện sẽ hiển thị thông tin người dùng đã xác nhận và thời điểm xác nhận, đồng thời sẽ vô hiệu hóa nút nhấn xác nhận.

```
if(user.role == "user") {
    ackSwitch.setClickable == false;
} else {
    ackSwitch.setClickable == true;
}

ackSwitch.setOnCheckedChangeListener(new
       CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton
              compoundButton, boolean isChecked) {
        if(isChecked && user.role == "admin") {
            data.setACK(true);
            data.setAckUser(user.name);
            data.setACKTime(System.currentTimeMillis());
            tv_da_ackUser.setText(data.getAckUser());
            tv_da_ackTime.setText(new
              SimpleDateFormat("hh:mm:ss aa\ndd/MM/yyyy",
              Locale.ENGLISH).format(new
              Date(data.getACKTime())));
            ackSwitch.setClickable(false);
```

Cùng lúc nó sẽ gửi lại thông tin đối tượng Data đã được xác nhận cho VPS để ghi nhận vào database.

```
JSONObject object = new JSONObject();
try {
    object.put("indicator", data.getIndicator());
    object.put("value", data.getValue());
    object.put("unit", data.getUnit());
    object.put("LOWLOW", data.getLOWLOW());
    object.put("LOW", data.getLOW());
    object.put("HIGH", data.getHIGH());
    object.put("HIGHHIGH", data.getHIGHHIGH());
    object.put("DEAD_BAND", data.getDEAD_BAND());
    object.put("severity", data.getSeverity());
    object.put("startTime", data.getStartTime());
    object.put("duration", data.getDuration());
    object.put("content", data.getContent());
    object.put("ACK", data.getACK());
    object.put("ackUser", data.getAckUser());
    object.put("ACKTime", data.getACKTime());
} catch (JSONException e) {
    e.printStackTrace();
}
mSocket.emit("alarmLog", object);
```

Chương 4. ĐÁNH GIÁ KẾT QUẢ

4.1. Kết quả đạt được:

- Tìm hiểu được các giải pháp thu thập dữ liệu từ các trạm quan trắc cách xa nhau về mặt địa lý sử dụng công nghệ mạng riêng ảo.
- Xây dựng được các gateway trên Raspberry Pi bao gồm: một Modbus Driver để thu thập dữ liệu từ các thiết bị trường và một OPC UA Server để truyền dữ liệu lên Azure Cloud Computing thông qua kết nối VPN.
- Khởi tạo và xây dựng được một VPS trên Azure chạy OPC UA Client để thu thập dữ liệu tập trung về VPS.
- Xây dựng được REST API để cung cấp cho ứng dụng di động và web.
- Phát triển được ứng dụng di động trên Android viết bằng ngôn ngữ Java. Ứng dụng này thực hiện một số tính năng cơ bản bao gồm: real-time trend, alarm, login.
- Xây dựng được ứng dụng web sử dụng Node.js, và Express framework với những tính năng như của mobile app.
- Sau khi tham khảo luận văn các khoá trước nhận thấy đa phần các đề tài đều sử dụng cloud database để lưu trữ và chia sẽ dữ liệu đến các ứng dụng. Nhóm nhận thấy khi dữ liệu trong database có kích thước lớn thì sẽ làm tăng độ trễ khi truy vấn dữ liệu. Từ đó, nhóm quyết định xây dựng hệ thống sử dụng VPN và thư viện socket.io để truyền thẳng dữ liệu từ server trên VPS đến client mà không cần lưu vào database.

4.2. Các điểm thiếu sót:

- Một số tính năng còn thiếu trên ứng dụng người chẳng hạn như: report excel, …
- Chưa khai thác được vào các tính năng bảo mật của giao thức OPC UA.

- Chưa thu thập được dữ liệu thực tế từ các thiết bị trường nhằm kiểm tra hiệu suất, các tính năng của hệ thống.

4.3. Hướng phát triển của luận văn:

- Khai thác thêm tính năng bảo mật của giao thức OPC UA.
- Hoàn thiện các tính năng còn thiếu sót, phát triển thêm một số tính năng khác.
- Xây dựng hệ thống sử dụng các giao thức IoT khác, ví dụ như MQTT Sparkplug B.

TÀI LIỆU THAM KHẢO

[1] Wikipedia, “Virtual Private Network”. [Trực tuyến]. Địa chỉ: https://en.wikipedia.org/wiki/Virtual_private_network [Truy cập 18/5/2022].

[2] Nguyễn Hưng, “Wireguard là gì? Mọi thứ cần biết về Wireguard VPN”, 2021. [Trực tuyến]. Địa chỉ: https://vietnix.vn/wireguard-la-gi/ [Truy cập 18/5/2022].

[3] Nguyễn Hưng, “VPS là gì? VPS dùng để làm gì?”, 2022. [Trực tuyến]. Địa chỉ: https://vietnix.vn/vps-la-gi/ [Truy cập 18/5/2022].

[4] Priyesh Patel, “What is exactly Node.js?”, 2018. [Trực tuyến]. Địa chỉ: https://www.freecodecamp.org/news/what-exactly-is-node-js-ae36e97449f5 [Truy cập 18/5/2022].

[5] Nguyễn Hưng, “Tìm hiểu mô hình MVC là gì? Ví dụ về cách sử dụng mô hình MVC”, 2021. [Trực tuyến]. Địa chỉ: https://vietnix.vn/tim-hieu-mo-hinh-mvc-la-gi/ [Truy cập 18/5/2022].