SIEMENS Vietnam - Digital Industries
33 Le Duan, Ben Nghe Ward, District 1, Ho Chi Minh City

SIEMENS

IoT2050 For Everyone - IoT2050FE

Người thực hiện: Lê Tiến Vinh
Nguyễn Minh Thành
Nguyễn Phúc Vinh

TP. Hồ Chí Minh, Tháng 1/2022

Contents

1 Tổng quan dự án

1.1 Tên dự án

IoT2050 FE - IoT2050 For Everyone

1.2 Bối cảnh

IOT2050 thuộc dòng thiết bị IOT Gateway chuẩn công nghiệp. Thiết bị được SIEMENS giới thiệu ra thị trường từ 09/2020 và đã được đón nhận tại nhiều nước trên thế giới. Khách hàng thiết bị hướng đến là công ty cung cấp dịch vụ Điện - Tự Động Hóa, nhà máy sản xuất có trang bị máy móc tự động hóa. Các thị trường khu vực châu Âu, châu Mỹ với mặt bằng công nghệ của người dùng tương đối tốt. Mức độ đón nhận IOT2050 trên các thị trường này cao hơn khu vực Đông Nam Á. Thống kê thu thập ý kiến từ các kỹ sư Tự động hóa tại thị trường Việt Nam, chúng ta có một số lý do dưới đây làm chậm quá trình áp dụng IOT2050 vào thực tế:

- IOT2050 là một máy tính nhúng tương tự như Raspberry, cho khả năng tùy biến cao nhưng đòi hỏi người dùng có kiến thức cơ bản về lập trình, troubleshooting, maintain code ... Phần lớn kỹ sư Tự động hóa ở Việt Nam thiếu kỹ năng này. Các trường đại học, cao đẳng, nghề ở Việt Nam chưa đào tạo các kỹ năng này trong chương trình kỹ sư Tự động hóa.
- Hơn 80% người dùng IOT2050 tại Việt Nam sử dụng các tính năng giống nhau như: kết nối IO (thiết bị PLC, biến tần, cảm biến, SQL Server ...), lưu trữ cục bộ, truyền tải dữ liệu về cloud, máy tính trung tâm, hiển thị dữ liệu lên giao diện web ... Các tính năng này có thể được "đóng gói" lại thành các source code hoàn chỉnh và có độ tùy biến cao. Việc này sẽ giúp người dùng dễ dàng tiếp cận thiết bị.
- Mặt bằng thu nhập của kỹ sư Tự động hóa tại Việt Nam trung bình thấp hơn kỹ sư Công nghệ thông tin. Các doanh nghiệp cung cấp dịch vụ, nhà máy không thể tuyển kỹ sư Công nghệ thông tin để thực hiện các dự án IOT công nghiệp. Kỹ sư tự động hóa phải làm việc với nhiều thiết bị, phần mềm, quy trình sản xuất ... Công việc đòi hỏi giao tiếp nhiều với khách hàng cũng làm mất lợi thế về thời gian nghiên cứu, học hỏi các công nghệ lập trình.

Từ các vấn đề trên, dự án IOT2050 FE - IOT2050 For Everyone được hình thành nhằm xây dựng một framework chung để sử dụng thiết bị và đáp ứng các nhu cầu:

- Kỹ sư tự động hóa dễ dàng sử dụng thiết bị để thực hiện các tính năng thông dụng.
- Kỹ sư công nghệ thông tin dễ dàng tích hợp các ứng dụng nâng cao (ngoài thông dụng) vào IOT2050.

1.3 Thông tin thiết bị

IOT2050 có 2 model 1GB RAM và 2GB RAM. Model 1GB RAM được ứng dụng rộng rãi hơn do nhu cầu tiết kiệm chi phí. Thông tin cấu hình chi tiết tại đây.

2 Yêu cầu cho hệ thống

2.1 Yêu cầu về dữ liệu

Trong một hệ thống sẽ gồm hai thành phần chính là: thiết bị (Device) và data point (Tag).

2.1.1 Thiết bị (Device)

IOT2050 có thể kết nối đến nhiều thiết bị, mỗi thiết bị có thể sử dụng một giao thức (Protocol) khác nhau. Các giao thức sử dụng có thể là: Modbus TCP/IP, Modbus RTU, OPC UA, MQTT,...

Các giao thức sử dụng 1 trong 2 loại đường truyền (Media): Ethernet hoặc Serial. Trong Serial có thể là: RS232, RS485, RS422. Tương ứng với mỗi dạng giao thức, thiết bị sẽ có các thuộc tính khác nhau.

a) Các giao thức công nghiệp truyền thống dùng Ethernet (Modbus TCP/IP, Profinet, EthernetIP,...)
Những thuộc tính chung cho các giao thức này như:

- Tên thiết bị
- Địa chỉ IP
- Port

Những thuộc tính riêng đối với các giao thức khác nhau như:

- Slave ID (chỉ với Modbus TCP/IP)
- Rack (chỉ với Profinet)
- Slot (chỉ với Profinet)
  ...

b) Các giao thức công nghiệp truyền thống dùng Serial (Modbus RTU, Profibus, CAN, ASCII,...)

- Tên thiết bị
- Baud rate
- Data bits (7 bits, 8 bits)
- Stop bit (1 Stop bit, 2 Stop bits)
- Parity (None Parity, Odd Parity, Even Parity)
- Slave ID (chỉ với Modbus RTU)

c) Các giao thức công nghiệp mới (OPC UA)

- Tên thiết bị
- End point (URL)
- User/Password
- SSL Certificate

d) Giao thức Message Queue cho IoT (MQTT)

- IP Address
- Port
- User/Password (MQTT hoặc OPC UA)
- SSL Certificate (MQTT hoặc OPC UA)
- QoS

2.1.2 Data point (Tag)

Mỗi thiết bị (Device) chứa một hoặc nhiều data point (Tag).

Tương ứng với mỗi dạng giao thức được dùng cho các thiết bị thì Tag của các thiết bị ứng với các giao thức đó cũng sẽ có các thuộc tính khác nhau.

a) Các giao thức công nghiệp truyền thống dùng Ethernet (Modbus TCP/IP, Profinet, EthernetIP,...)
   - Tên tag
   - Địa chỉ tag
   - Kiểu dữ liệu

b) Các giao thức công nghiệp truyền thống dùng Serial (Modbus RTU, Profibus,...)
   - Tên tag
   - Địa chỉ tag
   - Kiểu dữ liệu

c) Giao thức truyền thông công nghiệp mới (OPC UA)
   - Node ID
   - Node

d) Giao thức truyền thông Message Queue cho các ứng dụng IoT (MQTT)
   - Topic

2.2 Yêu cầu về chức năng

Các yêu cầu về chức năng của hệ thống được mô tả trong bảng bên dưới:

#	Tính năng	Mô tả
1	Quản lý các giao thức truyền thông trong công nghiệp	Tính năng quản lý giao thức truyền thông gồm có các yêu cầu như sau: • Hỗ trợ 3 giao thức truyền thông Modbus TCP/IP, Modbus RTU, OPC UA và một giao thức truyền thông điệp MQTT. • Người dùng là các developer có thể thêm các giao thức mới bằng cách thêm các thông tin cài đặt cho giao thức đó theo cấu trúc cơ sở dữ liệu được xây dựng sẵn bởi hệ thống và thông tin kiến trúc hệ thống từ documents.
2	Quản lý các thiết bị và data point	Tính năng quản lý các thiết bị và data point gồm có các yêu cầu như sau: • Người dùng có thể thêm mới thiết bị, các data point được sử dụng trong hệ thống của mình, thông qua nhập các thông tin cài đặt hoặc download và sau đó upload các thông tin cài đặt thông qua file .csv. Có quyền chỉnh sửa, hoặc xóa các thiết bị, data point. • Người dùng có thể quản lý được các thiết bị, data point nào đã được cài đặt thông tin hoặc chưa trong hệ thống. Bên cạnh đó là các data point nào được gắn với device nào. • Theo dõi số lượng data point được cài đặt trong hệ thống thông qua phân trang và tùy chỉnh số lượng hiển thị. • Người dùng có thể tạo ra số lượng thông tin cài đặt thiết bị thông qua số lượng thiết bị mà người dùng muốn tạo ra dưới dạng cùng một thông số cài đặt như nhau.
3	Điều khiển trạng thái thu thập dữ liệu từ các thiết bị ngoại biên	Tính năng này cho phép người dùng có thể bật hoặc tắt trạng thái thu thập dữ liệu từ các thiết bị ngoại biên thông qua nút nhấn trên giao diện cài đặt.
4	Truyền nhận dữ liệu thông qua giao thức truyền thông điệp MQTT	Tính năng truyền nhận dữ liệu thông qua giao thức message queue bao gồm các yêu cầu như sau: • Người dùng có thể thêm mới một gateway, cài đặt các thông số cho gateway đó, chỉnh sửa hoặc xóa các gateway không còn được sử dụng. • Người dùng có thể quản lý được các thiết bị nào, cũng như là các tag nào muốn thu thập dữ liệu tương ứng với từng thiết bị sẽ gửi thông tin dữ liệu thu thập được đến gateway đó. • Người dùng cũng có thể thay đổi thông tin nhận được từ gateway thông qua file JSON. • Người dùng có thể quản lý trạng thái hoạt động của các gateway có muốn gửi dữ liệu đi hoặc không thông qua nút nhấn trên giao diện cài đặt.

Table 1: *Danh sách các tính năng bắt buộc*

3 Kiến trúc hệ thống

3.1 Phân tích ngữ cảnh

Trong dự án này, thiết bị IOT2050 sẽ đóng vai trò như thiết bị điều khiển trung gian, đọc và gửi dữ liệu. Các kỹ sư tự động hoá sẽ được cung cấp một giao diện bởi IOT2050 và trực tiếp cấu hình, điều khiển luồng dữ liệu trên giao diện đó .Ngoài ra, hệ thống phải đạt được yêu cầu dễ mở rộng, để các kỹ sư IT có thể thêm các tính năng phát triển khác.

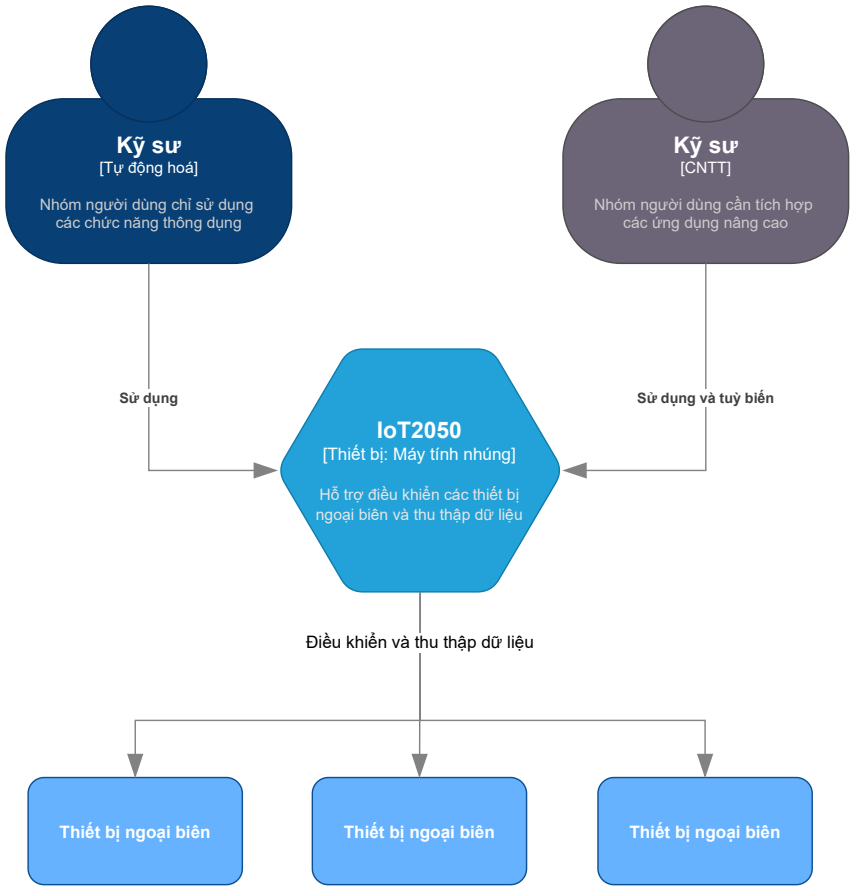

Figure 1: *Sơ đồ ngữ cảnh*

3.2 Các services trong hệ thống

Hệ thống gồm các services chính sau: Application, ConfigServer, nhóm service điều khiển/đọc data từ thiết bị (ModbusTCP, ModbusRTU, OPC UA), nhóm gateway service (MQTT Client) và Redis, ngoài ra còn có 1 service phụ là Logger. Chức năng từng service được mô tả như sau:

- **Application**: Cung cấp giao diện ứng dụng cho người dùng. Service này chỉ thực hiện một nhiệm vụ duy nhất là render file html.
- **ConfigServer**: Nhiệm vụ chính của service này là lưu các thông tin config thiết bị từ phía người dùng vào database.
- **Nhóm service điều khiển/đọc data từ thiết bị**: Trong công nghiệp hiện nay, có nhiều chuẩn giao tiếp giữa các thiết bị. Với mỗi chuẩn giao tiếp, nhóm hiện thực một service theo đặc trưng của chuẩn giao tiếp đó.
- **Nhóm gateway service**: Tương tự nhóm trên, tuy nhiên project hiện tại chỉ có 1 gateway là MQTT.

- **Redis**: Phục vụ cho việc giao tiếp giữa các service trong hệ thống. Redis là service có sẵn, tìm hiểu thêm tại đây.
- **Logger**: Service này chỉ sử dụng để ghi lại lịch sử hoạt động của hệ thống. Các log được phân loại theo tên service và thời gian xảy ra sự kiện.

Sơ đồ giao tiếp giữa các service được thể hiện trong hình sau:

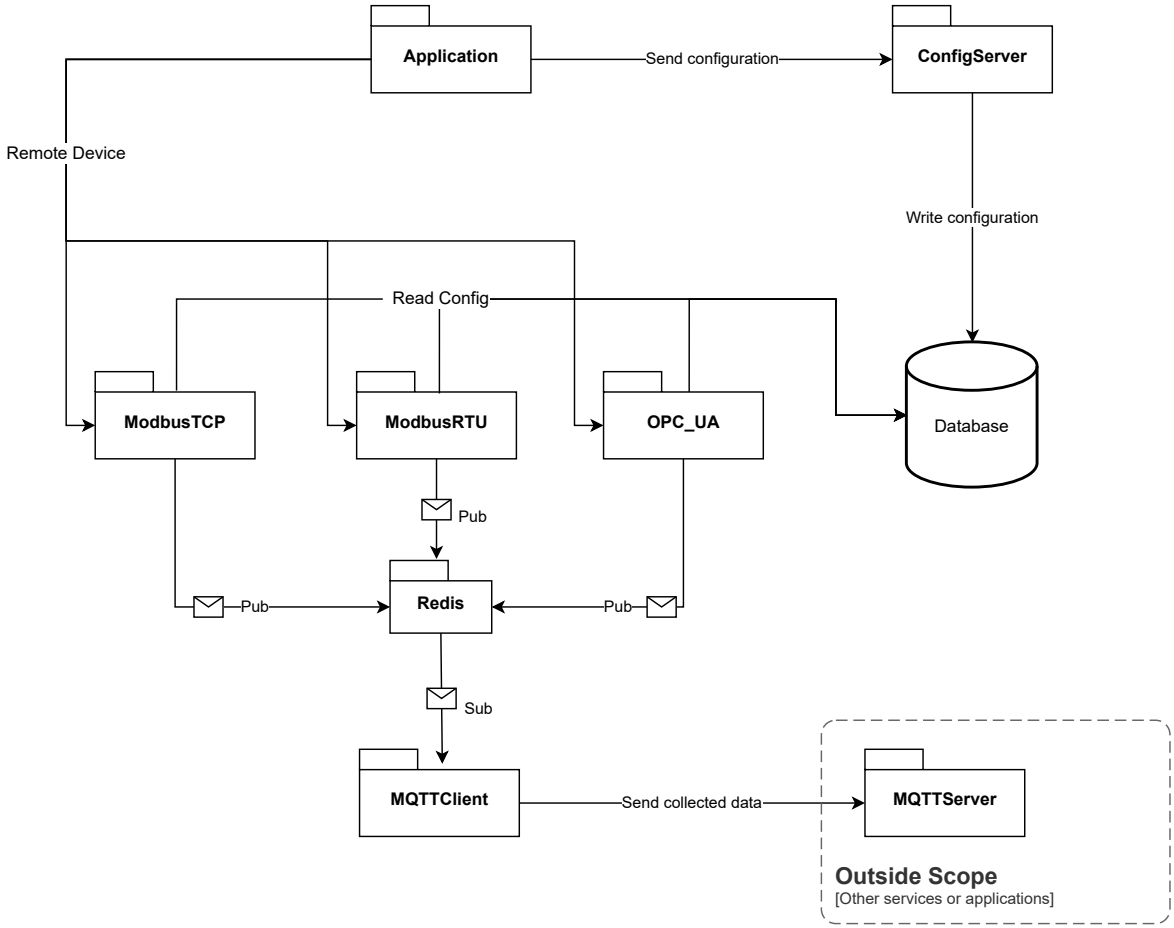

Figure 2: *Sơ đồ giao tiếp*

Ngoài ra, các service chính khi xảy ra sự cố, hoặc bất thường trong hoạt động hệ thống, đều gửi thông tin qua service Logger thông qua Redis.

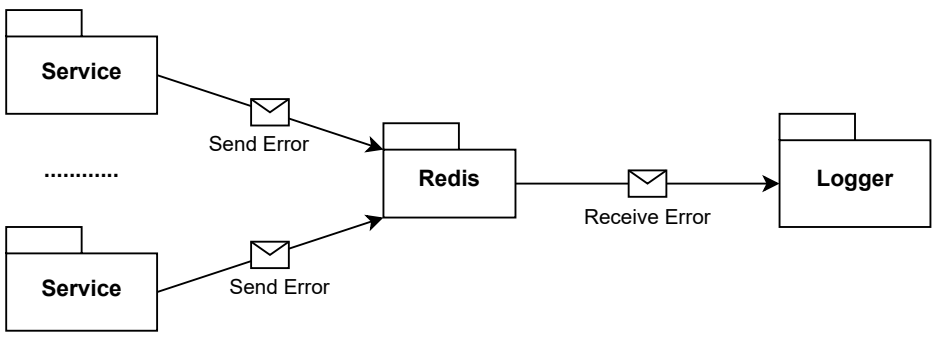

Figure 3: *Nhật ký hoạt động được gửi qua Logger*

Sau đây chúng ta sẽ đi sâu vào sơ đồ thiết kế của từng service riêng lẻ. Lưu ý rằng các thiết kế chỉ

nằm ở mức trừu tượng, giúp người đọc hình dung rõ hơn về logic của source code. Chi tiết về source code tại đây.

3.2.1 Application

Giao diện ứng dụng gồm 2 trang chính: Device và Gateway. Các trang đều cung cấp tính năng search item theo tên.

Trang Device được sử dụng để tạo mới/chỉnh sửa/xoá thiết bị. Trong một thiết bị, có thể thêm/xoá/sửa tag.

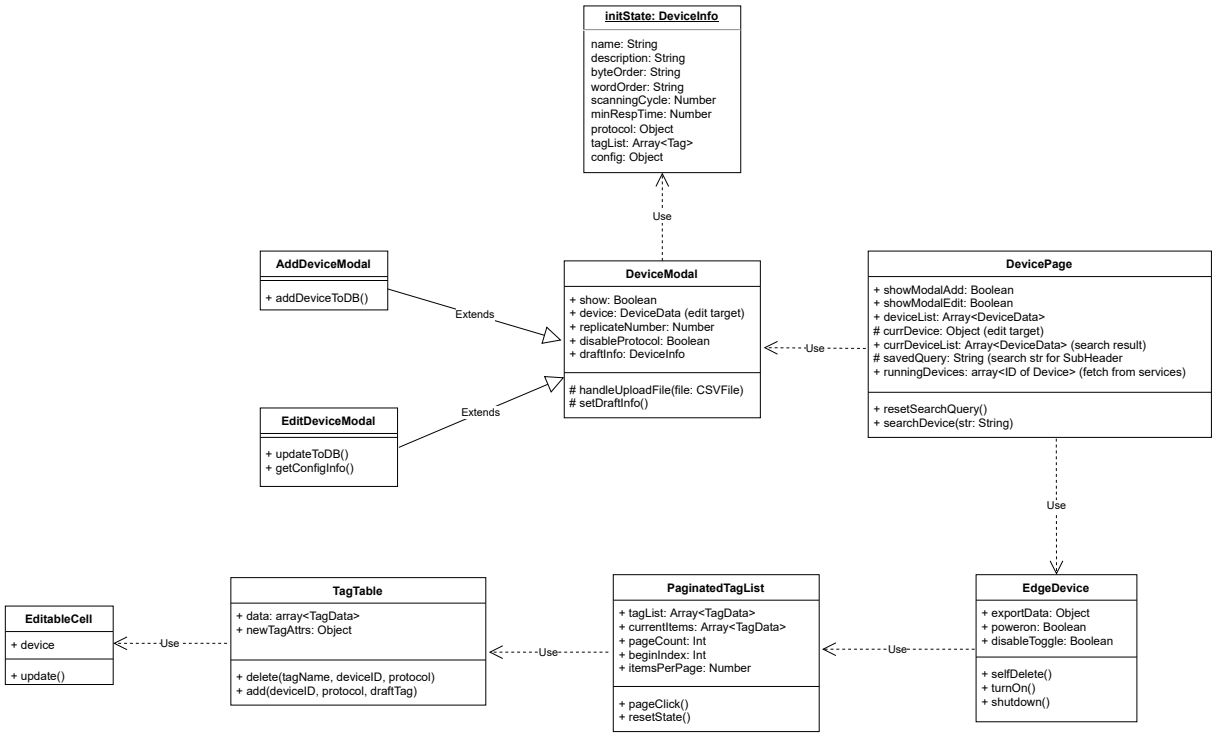

Figure 4: *Sơ đồ code trang Device*

Tương tự với trang Gateway, user có thể tạo/xoá/chỉnh sửa gateway. Ngoài ra, ở mỗi gateway, user có thể subscribe các device có sẵn. Với mỗi device, gateway được chọn subscribe các data point cần thiết.

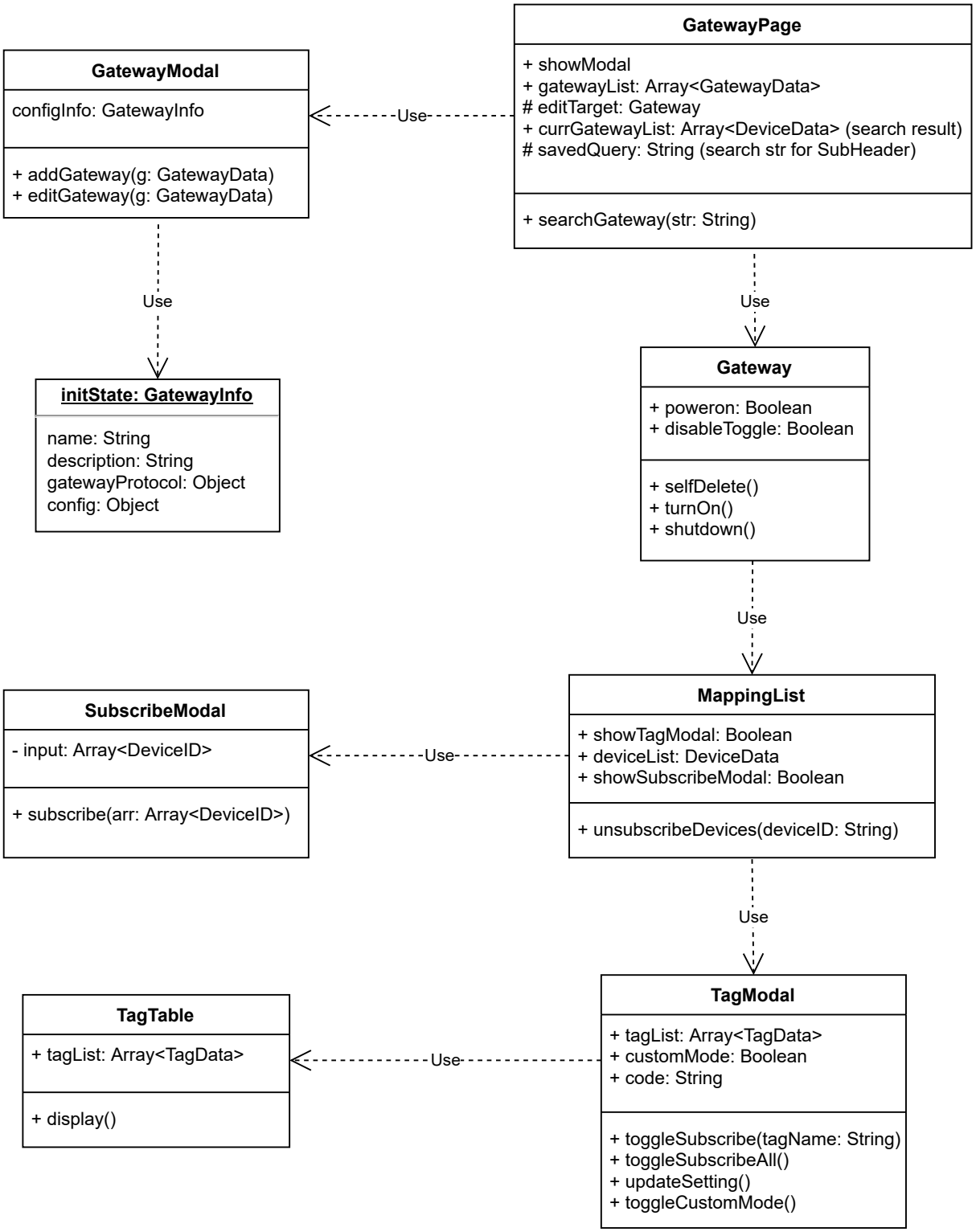

Figure 5: *Sơ đồ code trang Gateway*

3.2.2 Config Server

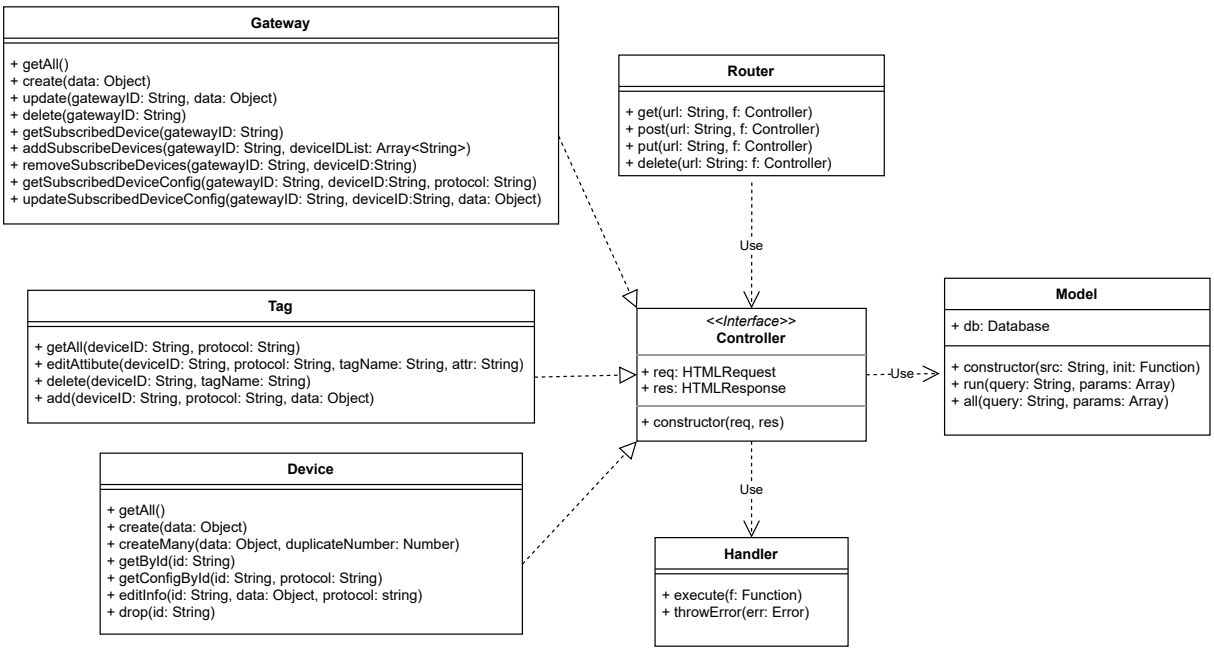

Figure 6: *Sơ đồ code Config Server*

3.2.3 Modbus TCP

Giao thức Modbus TCP giao thức hỗ trợ người dùng trong việc thu thập dữ liệu từ các thiết bị ngoại biên thông qua Ethernet.

Sơ đồ code cụ thể được trình bày như dưới đây.

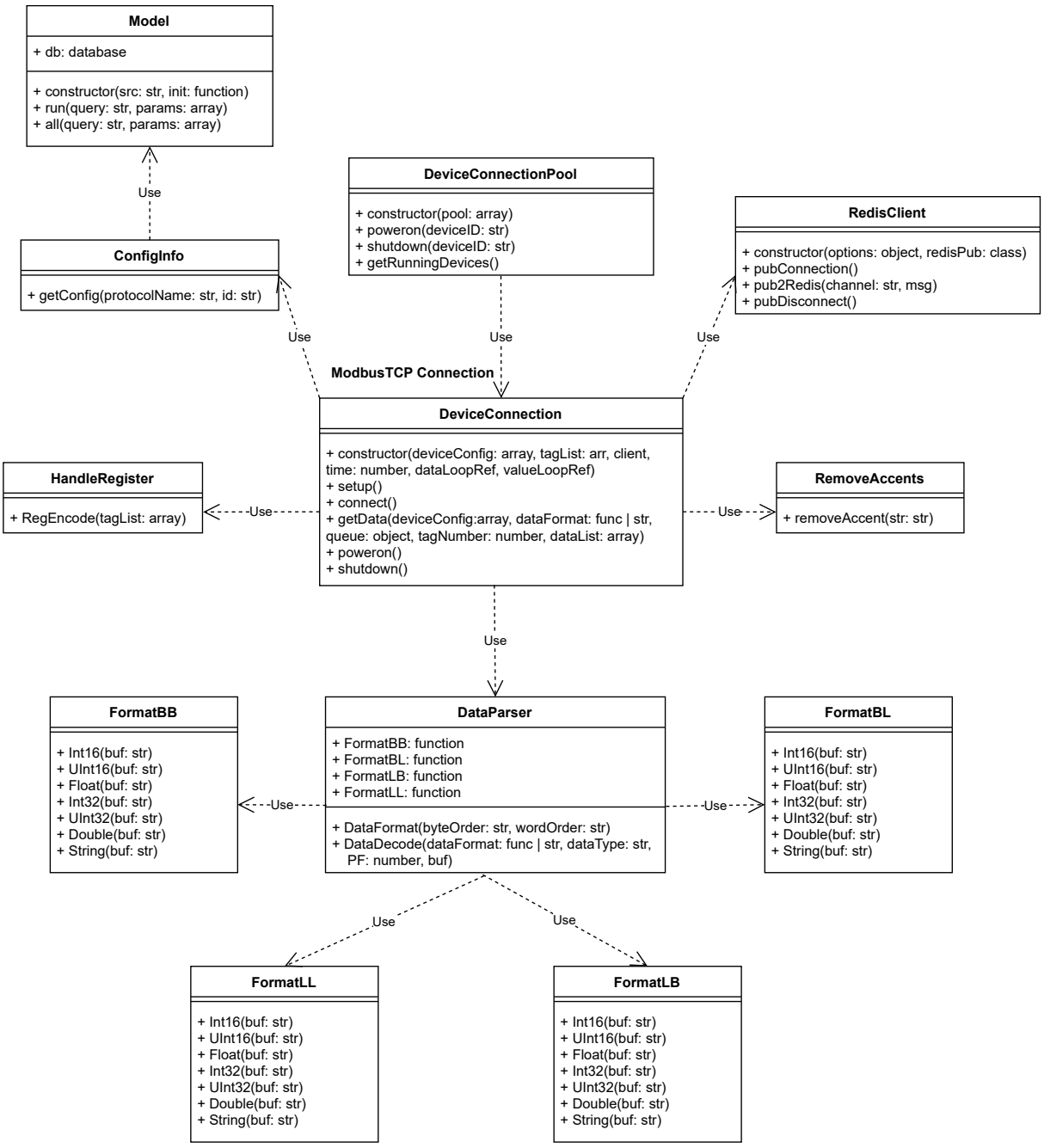

Figure 7: *Sơ đồ code ModbusTCP Client*

3.2.4 Modbus RTU

Tương tự với giao thức Modbus TCP giao thức Modbus RTU hỗ trợ người dùng trong việc thu thập dữ liệu từ các thiết bị ngoại biên với RS232 hoặc RS485. Sơ đồ code của giao thức Modbus RTU cũng tương tự với giao thức Modbus TCP chỉ khác nhau cách kết nối đến thiết bị.

Sơ đồ code cụ thể được trình bày như dưới đây.

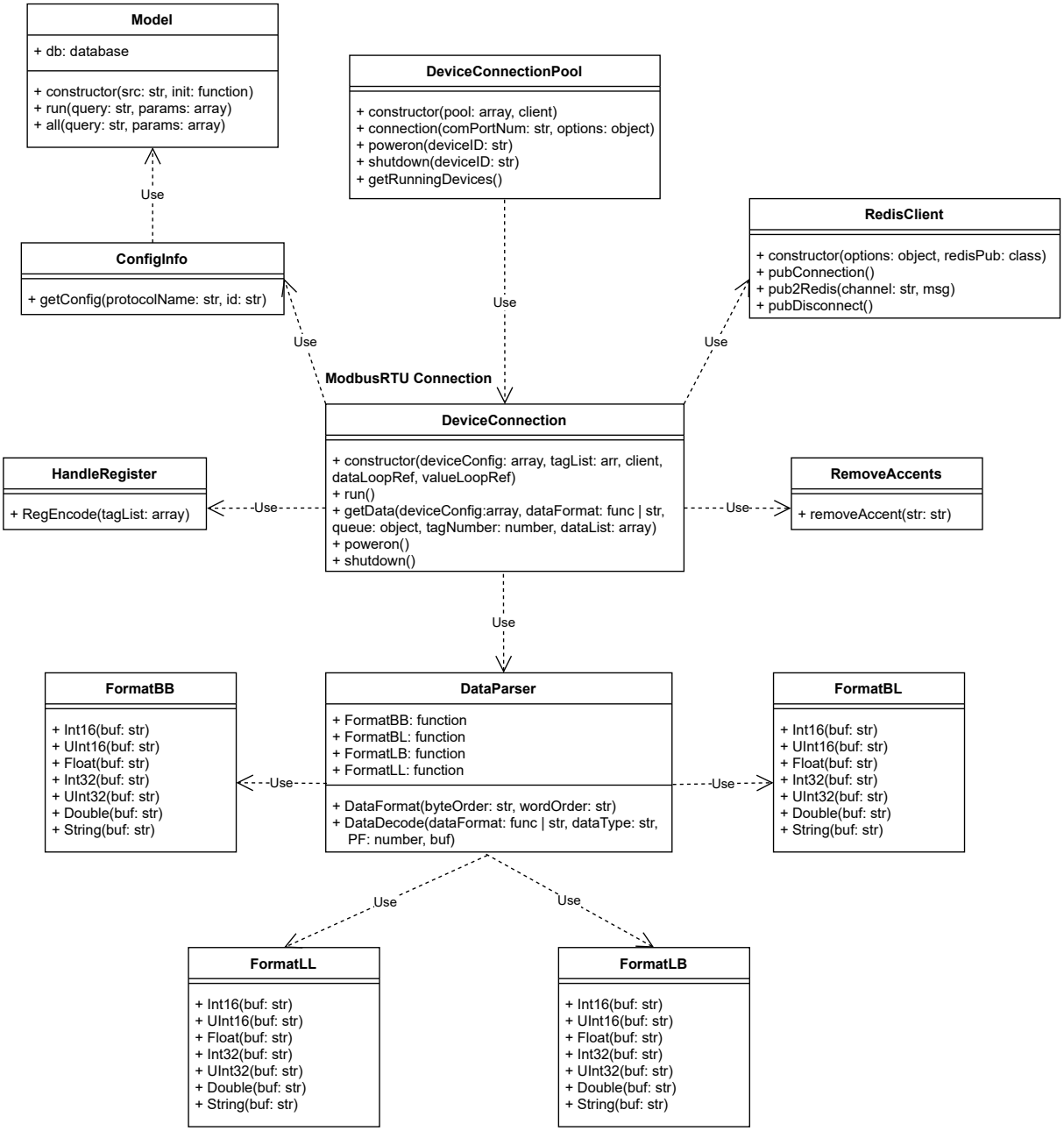

Figure 8: *Sơ đồ code ModbusRTU Client*

3.2.5 OPC UA

OPC UA là chuẩn đảm bảo kết nối mở, khả năng tương tác và độ bảo mật cho thiết bị. Ở đây xây dựng cho người dùng một OPC UA Client để kết nối đến thiết bị sử dụng OPC UA Server.

Sơ đồ code cụ thể được trình bày như dưới đây.

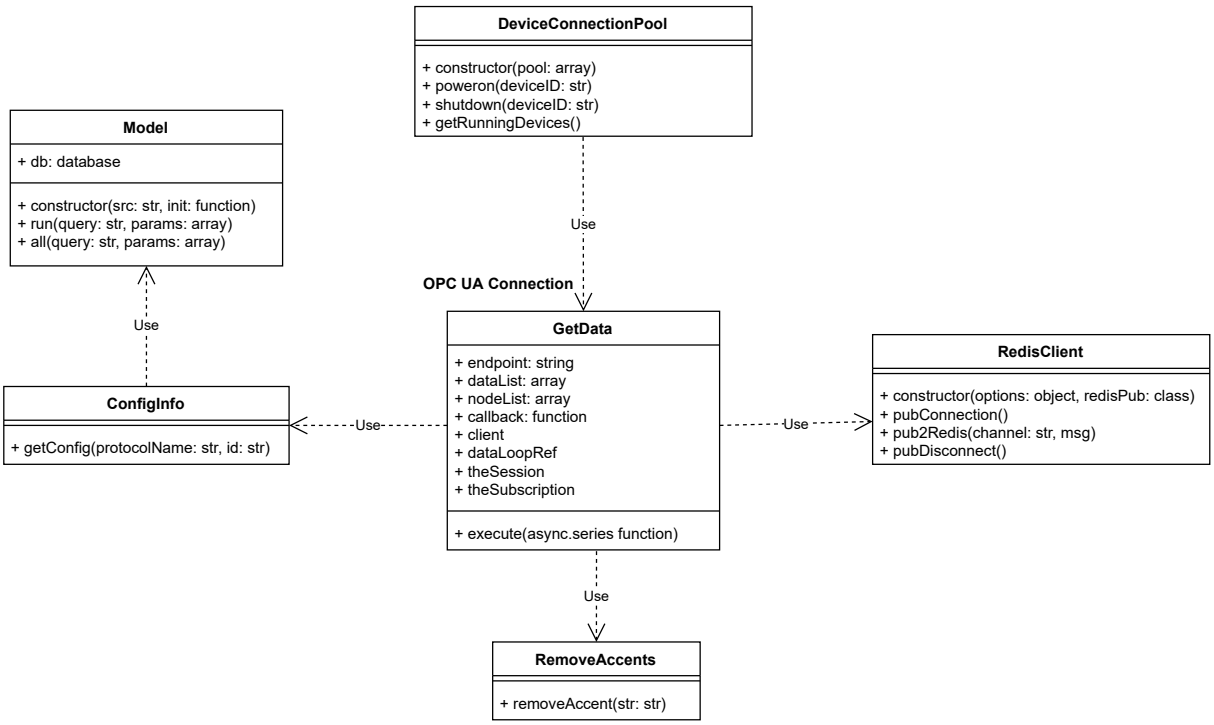

Figure 9: *Sơ đồ code OPC UA Client*

3.2.6 MQTT Client

Giao thức MQTT được dùng rộng rãi trong công nghiệp vì độ tin cậy cao, băng thông thấp và mạng lưới không ổn định. Ở đây xây dựng cho người dùng một MQTT Client để người dùng có thể gửi dữ liệu thu thập được từ các thiết bị đến các Broker mà người dùng mong muốn. Từ đó người dùng có thể nhận và sử dụng những dữ liệu này cho mục đích mong muốn.

Sơ đồ code cụ thể được trình bày như dưới đây.

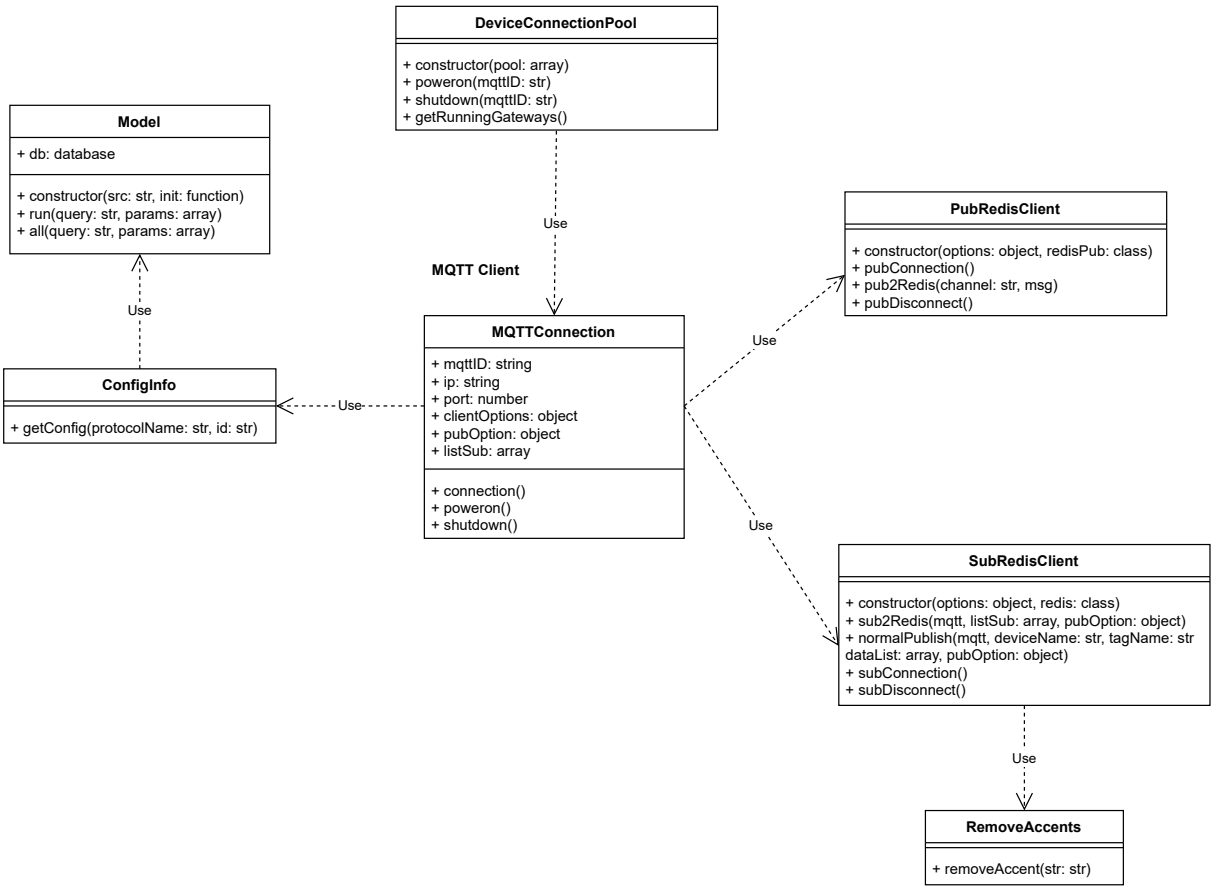

Figure 10: *Sơ đồ code MQTT Client*

3.2.7 Logger

Những thông tin về hoạt động của hệ thống từ các service đang hoạt động như: info, warning, error đều được ghi nhận trong log file. Người dùng có thể dùng để kiểm tra, chỉnh sửa những sự cố xảy ra khi hệ thống hoạt động.

Sơ đồ code cụ thể được trình bày như dưới đây.

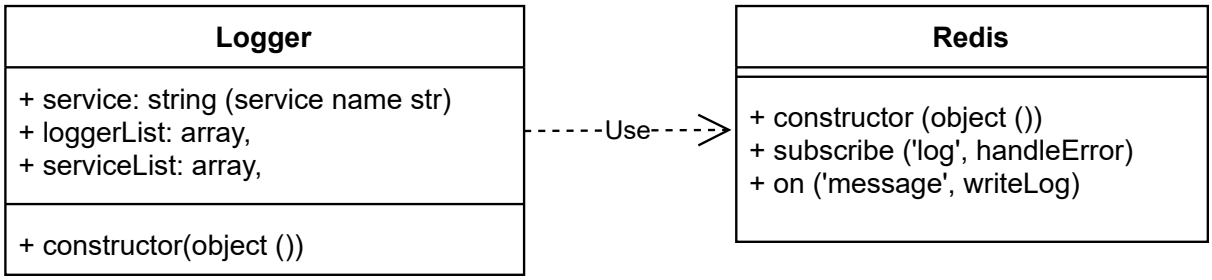

Figure 11: *Sơ đồ code Logger*

3.3 Kết nối mạng

Với scope hiện tại là nhà máy, phạm vi truy cập của hệ thống là mạng LAN. Bất kỳ máy tính cùng mạng LAN với thiết bị IOT2050 đều có thể truy cập được ứng dụng bằng cách nhập url có định dạng như sau: http://<Địa chỉ IP của IOT2050>:<Cổng kết nối>. Ví dụ: http://192.168.137.153:3000.

Dữ liệu được IOT2050 đọc từ các thiết bị khác sẽ được gửi ra bên ngoài thông qua gateway (Ứng dụng cung cấp chức năng config gateway cho người dùng)

4 Cơ sở dữ liệu

4.1 Phân tích

Dựa trên mô tả trong phần data requirements, dữ liệu trong dự án được chia thành hai nhóm:

- Dữ liệu về các thiết bị, tag, gateway và thông số cài đặt trên từng thực thể (Nhóm 1)
- Dữ liệu realtime đọc từ các thiết bị ngoại biên (Nhóm 2)

Dữ liệu thuộc nhóm 1 được thiết kế sao cho có khả năng mở rộng. Với mỗiloại protocol khác nhau, ta có thông tin dữ liệu của tag và cấu hình thiết bị khác nhau.

Dữ liệu thuộc nhóm 2 sẽ được cung cấp cho các bên thứ ba sử dụng thông qua API. Dữ liệu thuộc nhóm này chỉ được lưu trữ tạm thời, dự phòng trong tình huống xảy ra sự cố đường truyền mạng khiến client không nhận được dữ liệu. Khi client kết nối lại, dữ liệu dự phòng sẽ được gửi kèm với dữ liệu hiện tại. Do tính chất tạm thời và phi cấu trúc, ta không cần xây dựng thiết kế cho nhóm dữ liệu này.

4.2 Thiết kế lược đồ ý niệm

Dự án sử dụng lược đồ thực thể - mối liên kết (Entity Relationship Diagram - ERD) để biểu diễn lược đồ ý niệm. Điểm mạnh của lược đồ ERD là gần gũi với ngôn ngữ tự nhiên, không yêu cầu hiểu biết về công nghệ, nhưng lại đáp ứng được các quy luật ánh xạ sang thiết kế luận lý cho CSDL. Sau đây là lược đồ ERD tổng quát:

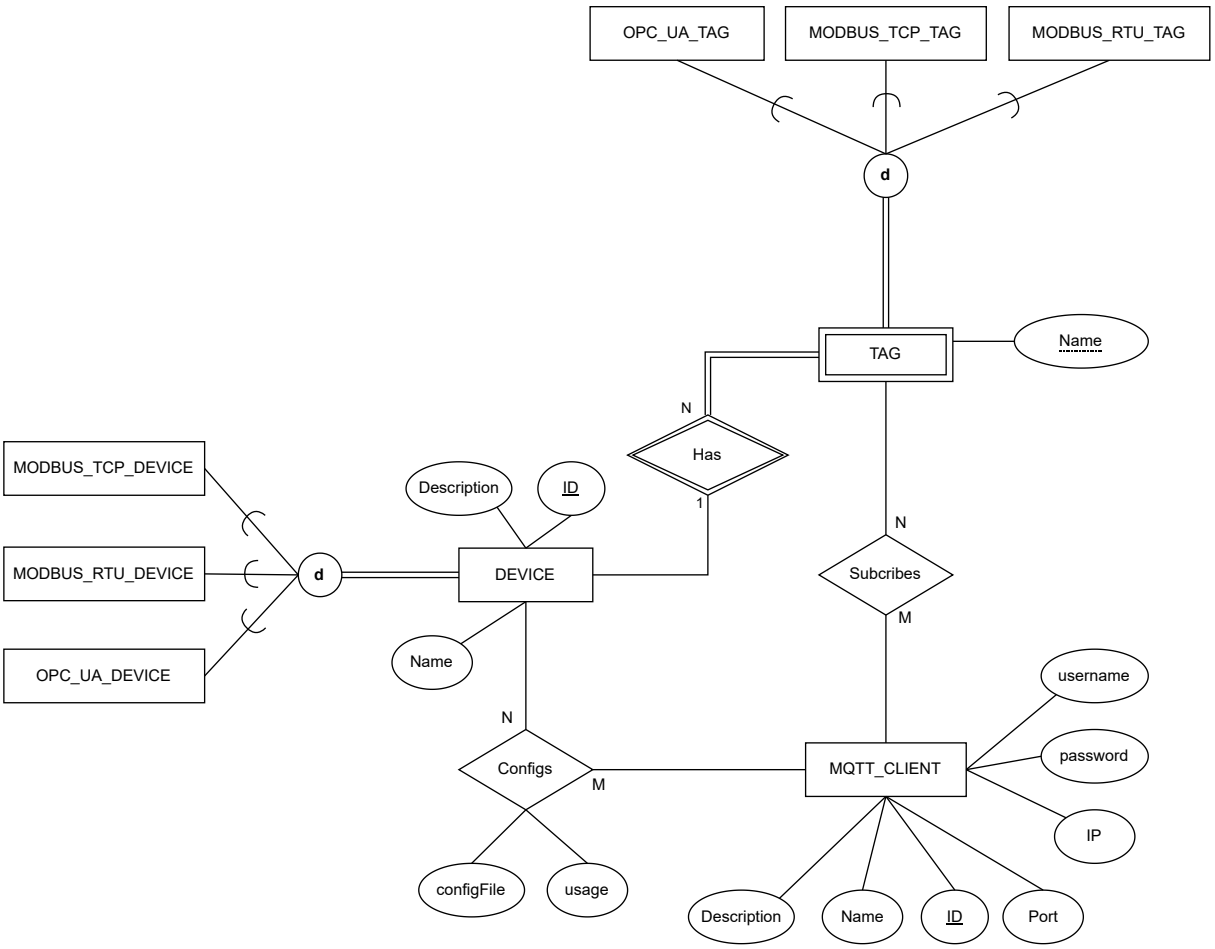

Figure 12: *Thiết kế luận lý tổng quát*

Ta có mô tả về lược đồ ERD (Entity Relationship Diagram) như sau:

- Thực thể Device có thể chia làm 3 loại, tuỳ vào dạng protocol mà Device sử dụng (Modbus TCP, Modbus RTU, v.v). Mỗi loại protocol sẽ có thuộc tính cấu hình khác nhau. Các Device được phân biệt bởi ID.
- Thực thể Tag (hoặc Data Point) không thể tồn tại độc lập, mà được xác định dựa vào Device tương ứng. Tương tự với Device, Tag cũng có những thuộc tính khác nhau tuỳ vào loại Protocol. Một thực thể Tag được phân biệt bởi tên và ID của Device mà nó được gắn vào.
- Thực thể MQTT_Client chính là một gateway. Hiện tại gateway chỉ sử dụng một loại protocol là MQTT_Client nên dự án sử dụng thực thể MQTT_Client đại diện cho một gateway. Trong tương lai, gateway có thể sử dụng thêm nhiều loại protocol. Các developer tiếp tục phát triển có thể phân loại gateway giống như cách phân loại thực thể Device.
- Một MQTT_Client có thể chỉ subscribes những Tag (data point) cần thiết từ một Device. Ngược lại, một Tag cũng có thể được subscribe bởi nhiều MQTT_Client khác nhau. Mối quan hệ giữa MQTT_Client và Tag là mối quan hệ N-N.
- Ngoài cách subscribe trực tiếp các tag, MQTT_Client còn có thể sử dụng một file code bằng Javascript để config format data được gửi lên từ Device theo ý muốn. Mối quan hệ Configs giữa MQTT_Client và Device là N-N.

Dưới đây là mô tả Attribute cụ thể của các loại Device và Tag:

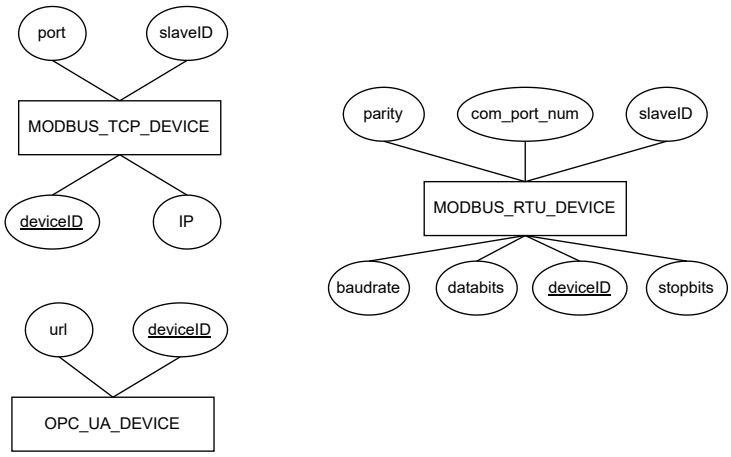

Figure 13: *Thuộc tính các loại Device*

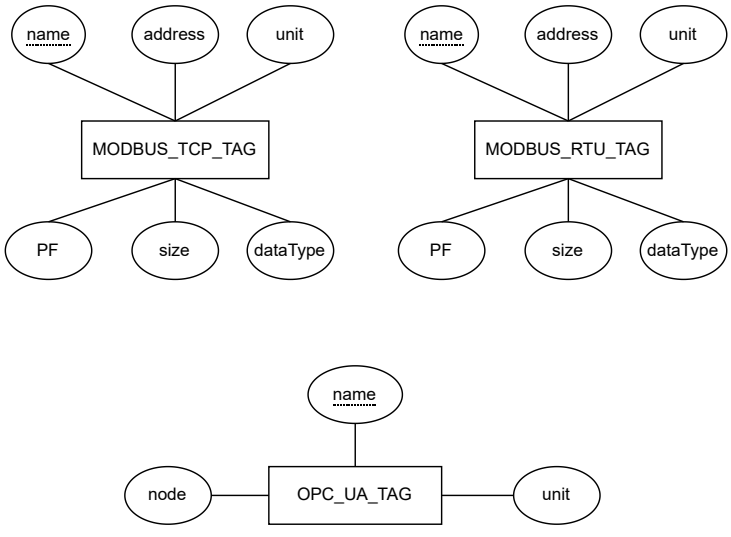

Figure 14: *Thuộc tính các loại Tag*

4.3 Thiết kế luận lý

Áp dụng các quy luật ánh xạ từ lược đồ ERD, ta có thiết kế luận lý như sau:

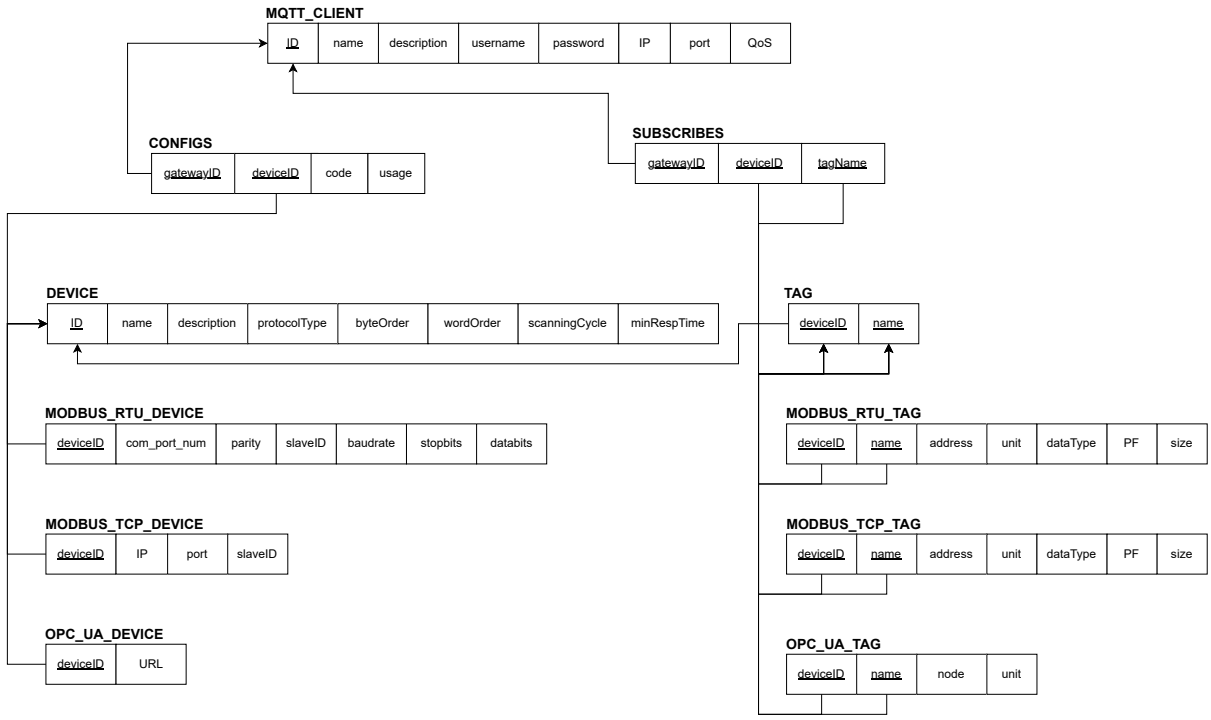

Figure 15: *Thiết kế luận lý*

Có thể coi mỗi một mảng trong sơ đồ là một table trong SQL, các ô chính là các cột trong bảng của nó. Ký hiệu gạch chân là primary key, các mũi tên nối giữa các bảng chính là quan hệ references giữa các bảng sử dụng foreign key.

5 Deployment

Dự án sử dụng công nghệ Docker để deploy và vận hành hệ thống. Các services được đóng gói thành các image độc lập, giao tiếp với nhau bởi docker network. Các service đều thống nhất sử dụng chung một database. Chi tiết mô tả tham khảo tại repository của dự án.

5.1 Deployment cho user

Các image của dự án đã được publish trên Docker Hub. Người dùng chỉ cần pull các image về và chạy bằng file start-service.yml.

5.2 Deployment cho developer

Phần này sẽ hướng dẫn developer rebuild các image sau khi có cập nhật mới trong các version kế tiếp của dự án. Yêu cầu kiến thức về Docker: **Container**, **Image**, **Network**, **Volume**.

Source code có 2 mode: production và development. Developer sẽ code trên môi trường development (môi trường mặc định). Để build và publish các image, developer phải chuyển môi trường sang production bằng câu lệnh đã hướng dẫn trong file README.md của repo.

Lưu ý rằng các image phải được build trên kiến trúc phù hợp với kiến trúc của thiết bị IOT2050 (ví dụ: linux/arm64/v8).