

DẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH



**Đồ án liên ngành (CO4041 - CO4029)**

---

**BÁO CÁO DỰ ÁN:**

**HỆ THỐNG QUẢN LÝ NÔNG TRẠI THÔNG MINH**

---

GVHD: PGS.TS. Thoại Nam  
ThS. Hoàng Lê Hải Thành  
CN. Trần Nguyễn Minh Duy

Sinh viên: Lê Hoàng Việt 2252903  
Chế Minh Đức 2210783  
Nguyễn Phú Quý 2212871

# Mục lục

<b>Danh sách hình vẽ</b>	<b>3</b>
<b>Danh sách bảng</b>	<b>3</b>
<b>Danh sách phân chia công việc</b>	<b>4</b>
<b>1                   Giới thiệu</b>	<b>5</b>
1.1   Bối cảnh và bài toán . . . . .	5
1.2   Tính cấp thiết của đề tài . . . . .	5
1.3   Tình hình nghiên cứu và các giải pháp hiện có . . . . .	5
1.4   Mục tiêu và câu hỏi nghiên cứu . . . . .	5
1.5   Phạm vi và giới hạn của đề tài . . . . .	6
<b>2                   Cơ sở lý thuyết</b>	<b>6</b>
2.1   Tổng quan về nông nghiệp thông minh (Smart Farm) . . . . .	6
2.1.1   Khái niệm và bối cảnh . . . . .	6
2.1.2   Các thành phần công nghệ cốt lõi . . . . .	6
2.1.3   Thách thức trong triển khai thực tế . . . . .	7
2.2   Kiến trúc hệ thống và Công nghệ nền tảng . . . . .	7
2.2.1   Kiến trúc tổng thể . . . . .	7
2.2.2   Nền tảng ứng dụng . . . . .	7
2.2.3   Hệ tầng Backend và Quy trình DevOps . . . . .	8
2.2.4   Tích hợp trí tuệ nhân tạo . . . . .	8
2.2.5   Phần cứng IoT và Giao thức truyền thông . . . . .	8
2.3   Bài toán phát hiện lỗi cảm biến trong hệ thống IoT . . . . .	8
2.3.1   Tổng quan về lỗi cảm biến trong môi trường IoT . . . . .	8
2.3.2   Các phương pháp tiếp cận hiện có và hạn chế . . . . .	9
2.3.3   Cơ sở lý thuyết của phương pháp Robust Feature Extractor (RFE) . . . . .	9
<b>3                   Khảo sát các giải pháp đã có</b>	<b>10</b>
3.1   Khảo sát các ứng dụng, sản phẩm hoặc hệ thống liên quan . . . . .	10
3.2   Khảo sát các mô hình, thuật toán, phương pháp . . . . .	10
3.3   Bảng tổng hợp và phân tích khoảng trống . . . . .	10
<b>4                   Phân tích yêu cầu</b>	<b>11</b>
4.1   Mục đích . . . . .	11
4.2   Phương pháp thu thập yêu cầu . . . . .	11
4.3   Kết quả thu thập yêu cầu . . . . .	11
4.4   Yêu cầu chức năng . . . . .	11
4.5   Yêu cầu phi chức năng . . . . .	12
4.6   Các mô hình phân tích UML . . . . .	13
4.6.1   Use case: Monitor Farm . . . . .	13
4.6.2   Use case: Monitor Device . . . . .	17
4.6.3   Use case: Monitor System Analytics . . . . .	21
4.6.4   Use case: Troubleshoot System . . . . .	23
4.6.5   Use case: Manage Reports . . . . .	25
<b>5                   Giải pháp đề xuất</b>	<b>27</b>
5.1   Mục tiêu của giải pháp . . . . .	27
5.1.1   Mục tiêu 1: Tự động hóa quá trình thu thập và xử lý dữ liệu môi trường . . . . .	27
5.1.2   Mục tiêu 2: Phát hiện và cảnh báo sớm các sự cố hệ thống . . . . .	27
5.1.3   Mục tiêu 3: Cung cấp giao diện quản lý trực quan và dễ sử dụng . . . . .	27
5.1.4   Mục tiêu 4: Đảm bảo khả năng mở rộng và hiệu năng cao . . . . .	27
5.1.5   Mục tiêu 5: Tối ưu hóa chi phí vận hành và bảo trì . . . . .	27
5.1.6   Mục tiêu 6: Đảm bảo tính bảo mật và độ tin cậy . . . . .	27
5.2   Thiết kế kiến trúc (Mức khái niệm) . . . . .	28



5.2.1	Phân hệ Quản lý Nông trại: . . . . .	28
5.2.2	Phân hệ IoT và Thu thập dữ liệu: . . . . .	29
5.2.3	Phân hệ Người dùng và Phân quyền: . . . . .	30
5.2.4	Phân hệ Kinh doanh . . . . .	31
5.3	Thiết kế Kiến trúc Hệ thống . . . . .	32
5.3.1	Kiến trúc Phân lớp (Layered Architecture) . . . . .	32
5.3.2	Kiến trúc Triển khai (Deployment Architecture) . . . . .	34
5.4	Dề xuất hiện thực và lựa chọn công nghệ . . . . .	35
5.4.1	Quản lý lược đồ dữ liệu cảm biến từ xa . . . . .	35
5.4.2	Dề xuất cải tiến cho thuật toán RFE . . . . .	37
5.5	Thiết kế các thành phần chính . . . . .	39
5.5.1	Software . . . . .	39
5.5.1.a	Quy trình Giám sát Dashboard (Analytics Flow): . . . . .	39
5.5.1.b	Quy trình Xử lý sự cố (Troubleshoot Flow): . . . . .	40
5.5.1.c	Quy trình quản lý Farm: . . . . .	41
5.5.1.d	Quy trình quản lý thiết bị: . . . . .	42
5.5.2	Firmware . . . . .	43
5.5.2.a	Activity Diagram . . . . .	43
5.5.2.b	Communication Diagram . . . . .	44
5.5.2.c	Component Diagram . . . . .	44
5.5.2.d	Sequence Diagram . . . . .	45
5.5.2.e	Timing Diagram . . . . .	46
5.5.2.f	State Machine Diagram . . . . .	47
5.6	Cách đánh giá giải pháp . . . . .	48
5.6.1	Dánh giá Hiệu năng Hệ thống (Performance Evaluation) . . . . .	48
5.6.2	Dánh giá Hiệu quả Thuật toán Phát hiện lỗi (Algorithmic Evaluation) . . . . .	48
5.6.3	Dánh giá Trải nghiệm và Chức năng (Functional & UX Evaluation) . . . . .	48
<b>6</b>	<b>Kế hoạch thực hiện</b>	<b>49</b>
6.1	Mục tiêu theo từng giai đoạn . . . . .	49
6.1.0.a	Firmware . . . . .	49
6.1.0.b	Software . . . . .	49
6.2	Lịch trình và mốc thời gian . . . . .	50
6.2.0.a	Firmware . . . . .	50
6.2.0.b	Software . . . . .	52
6.3	Rủi ro và phương án giảm thiểu . . . . .	53
6.3.0.a	Firmware . . . . .	53
6.3.0.b	Software . . . . .	54
<b>7</b>	<b>Kết luận</b>	<b>55</b>
7.1	Tóm tắt vấn đề và hướng tiếp cận . . . . .	55
7.2	Công việc đã hoàn thành trong giai đoạn Dồ án Chuyên ngành . . . . .	55
7.3	Định hướng thực hiện Dồ án Tốt nghiệp . . . . .	55
7.4	Bài học kinh nghiệm và Góc nhìn phản tư . . . . .	56
<b>Phụ lục A</b>	<b>Sơ đồ Cơ sở dữ liệu vật lý (Physical ERD)</b>	<b>57</b>
<b>Phụ lục B</b>	<b>Danh sách các API chính (API Specifications)</b>	<b>57</b>
<b>Phụ lục C</b>	<b>Danh sách linh kiện phần cứng (Bill of Materials)</b>	<b>58</b>
<b>Phụ lục D</b>	<b>Bảng so sánh các giải pháp tham khảo</b>	<b>59</b>
<b>Tài liệu tham khảo</b>		<b>60</b>

## Danh sách hình vẽ

1	Sơ đồ quản lý farm . . . . .	13
3	Sơ đồ luồng quản lý và tạo báo cáo . . . . .	21
5	Sơ đồ luồng quản lý và tạo báo cáo . . . . .	25
6	Sơ đồ thực thể quan hệ của phân hệ Quản lý Nông trại . . . . .	29
7	Sơ đồ thực thể quan hệ của phân hệ IoT và Thu thập dữ liệu . . . . .	30
8	Sơ đồ thực thể quan hệ của phân hệ Người dùng và Phân quyền . . . . .	31
9	Sơ đồ thực thể quan hệ của phân hệ Kinh doanh . . . . .	32
10	Sơ đồ Kiến trúc Phân lớp của Hệ thống . . . . .	33
11	Deployment Diagram . . . . .	34
12	Activity Diagram mô tả luồng giám sát và tương tác trên Dashboard . . . . .	39
13	Activity Diagram quy trình chẩn đoán và khắc phục sự cố hệ thống . . . . .	40
14	Activity Diagram mô tả luồng quản lý farm . . . . .	41
15	Activity Diagram mô tả luồng quản lý thiết bị . . . . .	42
16	Activity Diagram . . . . .	43
17	Communication Diagram . . . . .	44
18	Component Diagram . . . . .	44
19	Sequence Diagram . . . . .	45
20	Timing Diagram . . . . .	46
21	State Machine Diagram . . . . .	47
22	Sơ đồ thực thể chi tiết (Physical Data Model) . . . . .	57
23	Dặc tả API Module Quản lý Nông trại (Farms Endpoint) [Nguồn: Swagger UI Hệ thống]	58
24	Dặc tả API Module Quản lý Thiết bị (Devices Endpoint) [Nguồn: Swagger UI Hệ thống] .	58

## Danh sách bảng

1	Bảng phân chia công việc . . . . .	4
2	Các kịch bản đánh giá thuật toán phát hiện lỗi . . . . .	48
3	Bảng phân bổ lịch trình thực hiện đồ án 15 tuần . . . . .	50
5	Bảng phân tích rủi ro và phương án giám thiểu . . . . .	53
6	Phân tích rủi ro phần mềm và phương án xử lý . . . . .	54
7	Bill of Materials (BOM) . . . . .	59



## Danh sách phân chia công việc

STT	Họ tên	MSSV	Vai trò	Distribution
1	Lê Hoàng Việt	2252903	Backend	100%
1	Ché Minh Đức	2210783	Frontend	100%
1	Nguyễn Phú Quý	2212871	Firmware	100%

Bảng 1: Bảng phân chia công việc



## 1 Giới thiệu

### 1.1 Bối cảnh và bài toán

Trong xu thế nông nghiệp hiện đại, mô hình "Nông trại thông minh" (Smart Farm) đang trở thành tiêu chuẩn để đảm bảo năng suất và chất lượng nông sản thông qua việc ứng dụng công nghệ IoT và AI [1].

Tuy nhiên, thị trường hiện nay đang hình thành một nhóm đối tượng quản lý mới: các thương lái hoặc nhà đầu tư sở hữu nhiều nông trại (hoặc thuê lại để kinh doanh) nhưng không có chuyên môn sâu về công nghệ thông tin.

Mô hình kinh doanh của họ thường bao gồm việc quản lý chuỗi các nông trại phân tán hoặc cho khách hàng thuê ngắn hạn để trải nghiệm và canh tác. Từ thực tế này, bài toán đặt ra bao gồm:

- Sự phân mảnh thiết bị:** Các nông trại này thường tích hợp thiết bị IoT từ nhiều nhà cung cấp khác nhau, dẫn đến việc thiếu đồng bộ về giao thức và khó khăn trong quản lý tập trung [2].
- Rào cản kỹ thuật:** Người dùng (thương lái) gặp khó khăn khi phải thao tác trên nhiều ứng dụng rời rạc hoặc tự mình xử lý các sự cố kỹ thuật phức tạp.
- Yêu cầu về minh bạch chất lượng:** Khi kinh doanh dịch vụ, họ cần một công cụ tin cậy để chứng minh chất lượng môi trường canh tác (nhiệt độ, độ ẩm ổn định) cho khách hàng.

### 1.2 Tính cấp thiết của đề tài

Đề tài trở nên cấp thiết xuất phát từ nhu cầu thực tế của việc vận hành chuỗi nông trại quy mô thương mại và xu hướng chuyển đổi số mạnh mẽ tại Việt Nam:

- Xu hướng tắt yếu của chuyển đổi số:** Theo thống kê, tốc độ tăng trưởng GDP ngành nông nghiệp trong nửa đầu năm 2024 đạt 3.38%, mức cao nhất trong 5 năm qua, nhờ vào việc đẩy mạnh ứng dụng công nghệ cao [3]. Tuy nhiên, phần lớn các giải pháp hiện tại vẫn còn rời rạc, chưa tạo thành hệ sinh thái thống nhất.
- Nhu cầu quản lý tập trung (All-in-one):** Một trong những thách thức lớn nhất của nông nghiệp công nghệ cao hiện nay là sự thiếu liên kết chuỗi giá trị và sự manh mún trong quản lý [4]. Đối với các mô hình canh tác phân tán, việc thiếu một nền tảng quản lý hợp nhất dẫn đến lãng phí nguồn lực giám sát. Chủ đầu tư cần một giao diện duy nhất để quản lý hàng loạt nông trại.
- Độ tin cậy của dữ liệu cảm biến:** Các nghiên cứu gần đây chỉ ra rằng cảm biến IoT trong môi trường nông nghiệp khắc nghiệt thường xuyên gặp lỗi trôi số liệu (drift) hoặc mất kết nối. Nếu không phát hiện sớm bằng các thuật toán thông minh (Data-driven), dữ liệu sai lệch sẽ dẫn đến các quyết định sai lầm [5].

### 1.3 Tình hình nghiên cứu và các giải pháp hiện có

Hiện nay trên thị trường tồn tại hai nhóm giải pháp chính:

- Giải pháp trọn gói từ các hãng lớn (Israel, Nhật Bản):** Có độ ổn định cao nhưng chi phí rất đắt đỏ, hệ sinh thái đóng (không cho phép tích hợp thiết bị hãng khác), khó phù hợp với quy mô vừa và nhỏ tại Việt Nam [6].
- Giải pháp lắp ráp nhỏ lẻ (DIY):** Giá thành rẻ nhưng thiếu tính năng quản lý tập trung, giao diện sơ sài và đặc biệt là thiếu cơ chế cảnh báo lỗi thông minh.

**Khoảng trống nghiên cứu:** Chưa có nhiều giải pháp tập trung vào việc chuẩn hóa thiết bị đa nguồn kết hợp với thuật toán phát hiện lỗi cảm biến dành riêng cho phân khúc người dùng không chuyên kỹ thuật.

### 1.4 Mục tiêu và câu hỏi nghiên cứu

**Mục tiêu tổng quát:** Nghiên cứu và phát triển hệ thống quản lý tập trung và nhận diện lỗi cho các thiết bị IoT, hướng tới việc cung cấp giải pháp vận hành đơn giản, tin cậy cho các mô hình nông trại thông minh đa thiết bị.



### Mục tiêu cụ thể:

- Xây dựng kiến trúc hệ thống thống nhất có khả năng tích hợp thiết bị từ nhiều nguồn khác nhau.
- Phát triển module Sensor Fault Detection ứng dụng thuật toán để tự động phát hiện các lỗi thường (mất kết nối, dữ liệu sai lệch, trỗi cảm biến).
- Thiết kế giao diện người dùng (Dashboard) trực quan, thân thiện với đối tượng thương lái/chủ vườn, hỗ trợ giám sát đa nông trại và cảnh báo thời gian thực.

### Câu hỏi nghiên cứu:

- Làm thế nào để xây dựng một cơ chế định danh và quản lý thống nhất cho các thiết bị IoT đa dạng về giao thức?
- Thuật toán nào là tối ưu để phát hiện lỗi cảm biến trong môi trường dữ liệu thời gian thực với độ trễ thấp?
- Làm thế nào để thiết kế trải nghiệm người dùng tối giản hóa các thao tác kỹ thuật phức tạp?

## 1.5 Phạm vi và giới hạn của đề tài

### Phạm vi:

- *Đối tượng nghiên cứu:* Các thiết bị IoT cảm biến môi trường phổ biến và các thiết bị chấp hành cơ bản trong mô hình nhà kính.
- *Nền tảng công nghệ:* Hệ thống quản lý tập trung trên nền tảng Web (Web-based platform), sử dụng Cơ sở dữ liệu chuỗi thời gian (Time-series Database) tối ưu cho lưu trữ dữ liệu lớn IoT, và các giao thức truyền thông điệp nhẹ (Lightweight messaging protocols).
- *Triển khai:* Thủ nghiêm trên mô hình nông trại mẫu (Pilot testing) với dữ liệu mô phỏng và thực tế.

### Giới hạn:

- Hệ thống tập trung vào việc phát hiện lỗi phần cứng/cảm biến dựa trên phân tích dữ liệu, chưa bao gồm việc chẩn đoán sâu bệnh cây trồng bằng hình ảnh (Computer Vision).
- Các thuật toán phát hiện lỗi sẽ được kiểm nghiệm trên một số loại cảm biến đặc thù, có thể cần điều chỉnh lại khi áp dụng cho các loại cảm biến công nghiệp mới lạ.

## 2 Cơ sở lý thuyết

### 2.1 Tổng quan về nông nghiệp thông minh (Smart Farm)

#### 2.1.1 Khái niệm và bối cảnh

Nông nghiệp thông minh (Smart Farming) là việc ứng dụng các công nghệ tiên tiến của Công nghiệp 4.0 vào quy trình sản xuất nông nghiệp. Trọng tâm của mô hình này là khả năng thu thập, xử lý và phân tích dữ liệu thời gian thực từ các cảm biến nông nghiệp nhằm đưa ra các quyết định canh tác chính xác. Mục tiêu là tạo ra một khung làm việc (framework) bền vững và có khả năng mở rộng, giúp tích hợp các thiết bị IoT với nền tảng xử lý dữ liệu để hỗ trợ tối ưu hóa và cập nhật thông tin sản phẩm. Trong bối cảnh hiện nay, nông nghiệp thông minh không chỉ dừng lại ở tự động hóa sản xuất mà còn mở rộng sang việc kết nối thông minh giữa người nông dân và khách hàng, tạo ra chuỗi giá trị minh bạch từ trang trại đến bàn ăn.

#### 2.1.2 Các thành phần công nghệ cốt lõi

Dựa trên thực tế triển khai các giải pháp nông nghiệp hiện đại, một hệ thống Smart Farm tiêu chuẩn bao gồm các thành phần chính sau:

- **Hệ thống giám sát môi trường (Environmental Monitoring):** Sử dụng mạng lưới các cảm biến chuyên dụng để đo đạc các thông số vi khí hậu và môi trường đất/nước. Các thiết bị điển hình



bao gồm cảm biến độ ẩm đất, cảm biến nhiệt độ - độ ẩm (như DHT11), cảm biến ánh sáng, cảm biến mưa, và các cảm biến đo chất lượng dinh dưỡng như pH, EC/TDS.

- **Hệ tầng thu thập và xử lý dữ liệu (Data Infrastructure):** Dữ liệu từ cảm biến được thu thập liên tục và chuyển về các Nền tảng luồng dữ liệu (Data Streaming Platform - DSP). Tại đây, dữ liệu được xử lý để cung cấp thông tin chi tiết (actionable insights) cho người dùng cuối.
- **Nền tảng tương tác người dùng:** Các ứng dụng di động (ví dụ: Mini App trên nền tảng Zalo) đóng vai trò là giao diện điều khiển, cho phép nông dân truy cập dữ liệu trại thời gian thực, nhận cảnh báo và thực hiện các tác vụ như kích hoạt tưới tiêu từ xa. Đồng thời, công nghệ cũng hỗ trợ nông dân trong việc tiếp thị kỹ thuật số và quản lý nhật ký canh tác.

### 2.1.3 Thách thức trong triển khai thực tế

Mặc dù mang lại nhiều lợi ích, việc triển khai hệ thống IoT trong nông nghiệp đối mặt với những thách thức lớn về độ bền và độ tin cậy:

- **Điều kiện vận hành khắc nghiệt:** Các thiết bị IoT và bộ ghi dữ liệu (Datalogger) phải hoạt động liên tục ngoài trời, chịu tác động trực tiếp của thời tiết khắc nghiệt, đòi hỏi các tiêu chuẩn cao về khả năng chống nước, cách điện và bảo vệ cơ học.
- **Yêu cầu về độ ổn định dữ liệu:** Để đảm bảo hoạt động sản xuất an toàn và tin cậy, hệ thống cần duy trì khả năng thu thập dữ liệu ổn định và giảm thiểu nhiễu tín hiệu trong quá trình truyền thông.

## 2.2 Kiến trúc hệ thống và Công nghệ nền tảng

### 2.2.1 Kiến trúc tổng thể

Hệ thống Xanh Market được thiết kế theo mô hình phân tán, tích hợp chặt chẽ giữa các thiết bị IoT tại hiện trường, nền tảng xử lý dữ liệu dòng (Data Streaming Platform - DSP) và các ứng dụng người dùng cuối. Kiến trúc tổng thể bao gồm các thành phần chính:

- **Lớp thiết bị (Device Layer):** Bao gồm các trạm quan trắc (IoT Box) và hệ thống điều khiển tưới tiêu được triển khai thực tế tại nông trại (như Tomochan Farm). Các thiết bị này thu thập dữ liệu môi trường (nhiệt độ, độ ẩm, ánh sáng, EC/TDS) và gửi về hệ thống trung tâm.
- **Lớp xử lý và lưu trữ (Processing and Storage Layer):** Sử dụng Nền tảng luồng dữ liệu (DSP) để thu thập, xử lý và phân tích dữ liệu thời gian thực từ cảm biến. Hệ thống Backend được triển khai theo kiến trúc hiện đại, đảm bảo khả năng mở rộng và chịu lỗi.
- **Lớp ứng dụng (Application Layer):** Bao gồm cổng thông tin quản trị (CMS Portal) dành cho quản trị viên và các ứng dụng Zalo Mini App dành cho nông dân và khách hàng.

### 2.2.2 Nền tảng ứng dụng

Thay vì phát triển ứng dụng di động truyền thống (Native App), dự án lựa chọn nền tảng Zalo Mini App làm giao diện chính cho người dùng cuối với các ưu điểm vượt trội về khả năng tiếp cận thị trường Việt Nam:

- **Tận dụng lượng người dùng sẵn có:** Zalo là nền tảng chat phổ biến nhất Việt Nam, giúp ứng dụng tiếp cận ngay lập tức với nông dân và khách hàng mà không cần cài đặt phức tạp.
- **Định danh và xác thực:** Sử dụng cơ chế đăng nhập qua Zalo (Zalo Login), giúp hệ thống truy xuất được thông tin người dùng thực đã được xác minh, giảm thiểu rủi ro tài khoản ảo.
- **Thông báo tích hợp (Zalo Notification):** Hệ thống gửi cảnh báo và tin tức nông nghiệp trực tiếp qua tin nhắn Zalo, đảm bảo thông tin đến người dùng nhanh chóng và thuận tiện.
- **Công nghệ phát triển:** Zalo Mini App được xây dựng dựa trên các công nghệ web phổ biến (React + Vite), giúp tối ưu hóa quy trình phát triển và bảo trì.



### 2.2.3 Hạ tầng Backend và Quy trình DevOps

Hệ thống Backend được xây dựng trên nền tảng điện toán đám mây riêng (Private Cloud) tại phòng thí nghiệm HPC Lab để đảm bảo tính bảo mật và chủ quyền dữ liệu. Quy trình triển khai áp dụng các tiêu chuẩn công nghiệp:

- **Áo hóa và Container hóa:** Sử dụng Docker để đóng gói các dịch vụ, đảm bảo môi trường vận hành đồng nhất giữa phát triển và sản xuất.
- **CI/CD Pipeline:** Tích hợp quy trình Tích hợp liên tục và Triển khai liên tục (CI/CD) để tự động hóa việc cập nhật phần mềm.
- **Kiểm soát chất lượng mã nguồn:** Sử dụng SonarQube để tự động quét và phân tích mã nguồn, phát hiện các lỗ hổng bảo mật và đảm bảo chất lượng code trước khi triển khai.
- **Giám sát hệ thống (Monitoring):** Sử dụng bộ công cụ Grafana và Prometheus để theo dõi hiệu năng server, tài nguyên hệ thống (CPU, RAM, Disk) và trạng thái các dịch vụ theo thời gian thực.

### 2.2.4 Tích hợp trí tuệ nhân tạo

Một điểm nhấn công nghệ của hệ thống là việc tích hợp mô hình ngôn ngữ lớn (LLM) để hỗ trợ nông dân tạo nội dung số:

- **Mô hình:** Sử dụng mô hình mã nguồn mở Llama 3.2 (3 tỷ tham số) từ Meta.
- **Triển khai:** Mô hình được chạy cục bộ (On-premise) thông qua Ollama trên máy chủ có GPU, đảm bảo dữ liệu của nông dân không bị gửi ra các dịch vụ bên thứ ba, tăng cường tính bảo mật.
- **Tự động hóa:** Quy trình tạo bài viết (Blog generation) được quản lý bởi công cụ n8n, kết nối giữa ứng dụng Zalo và mô hình AI để tự động soạn thảo các bài quảng bá nông trại dựa trên từ khóa đầu vào.

### 2.2.5 Phần cứng IoT và Giao thức truyền thông

Hệ thống phần cứng được thiết kế để hoạt động bền bỉ trong môi trường nông nghiệp:

- **Datalogger:** Được đặt trong một hộp điện được chế tạo riêng, được thiết kế để sử dụng ngoài trời trong nông nghiệp.
- **Cảm biến:** Hỗ trợ đa dạng các loại cảm biến từ cơ bản (DHT11, cảm biến mưa, ánh sáng) cho mô hình hộ gia đình đến các cảm biến chuyên dụng công nghiệp (pH, EC) cho trang trại lớn.
- **Giao thức:** Sử dụng giao chuẩn RS485 cho việc kết nối cảm biến nhằm giảm thiểu nhiễu và đảm bảo truyền tin ổn định trên khoảng cách xa trong trang trại.

## 2.3 Bài toán phát hiện lỗi cảm biến trong hệ thống IoT

### 2.3.1 Tổng quan về lỗi cảm biến trong môi trường IoT

Trong bối cảnh Công nghiệp 4.0, các hệ thống Internet vạn vật (IoT) đóng vai trò trung tâm trong việc thu thập dữ liệu thời gian thực cho các ứng dụng từ sản xuất thông minh đến thiết bị bay không người lái (drones). Tuy nhiên, đặc thù của các thiết bị IoT là thường xuyên phải hoạt động trong các môi trường khắc nghiệt (nhiệt độ cao, độ ẩm lớn, nhiễu điện từ). Điều này dẫn đến vấn đề suy giảm chất lượng dữ liệu, biểu hiện qua các dạng lỗi cảm biến phổ biến như:

- **Giá trị bất thường (Outliers/Spikes):** Các giá trị nhảy vọt đột ngột không phản ánh thực tế.
- **Trôi tín hiệu (Drift):** Giá trị đọc sai lệch dần theo thời gian so với giá trị thực.
- **Kẹt giá trị (Stuck-at):** Cảm biến trả về một giá trị không đổi trong thời gian dài.

Việc phát hiện sớm các lỗi này là tối quan trọng để đảm bảo độ tin cậy của toàn hệ thống.



### 2.3.2 Các phương pháp tiếp cận hiện có và hạn chế

Hiện nay, bài toán phát hiện lỗi cảm biến thường được giải quyết theo ba hướng chính, tuy nhiên mỗi hướng đều tồn tại những hạn chế khi áp dụng cho các thiết bị IoT có tài nguyên hạn chế (Low-power IoT devices):

- **Học máy cổ điển (Classic ML):** Thường gặp khó khăn trong việc nắm bắt các mẫu lỗi phức tạp hoặc phi tuyến tính trong chuỗi thời gian.
- **Học sâu (Deep Learning - DL):** Các mô hình như Autoencoder hay LSTM dù mạnh mẽ nhưng lại là những "hộp đen" (black-box), thiếu tính giải thích. Quan trọng hơn, chúng đòi hỏi tài nguyên tính toán và năng lượng lớn, không phù hợp để triển khai tại biên (Edge/Node).
- **Phương pháp dựa trên tương quan:** Hoạt động kém hiệu quả trong môi trường động và thường bỏ qua các thông tin nội tại của từng dòng dữ liệu riêng lẻ.

Từ những phân tích trên, đặt ra yêu cầu về một phương pháp trích xuất đặc trưng (Feature Extraction) chuyên biệt, vừa đảm bảo độ chính xác cao, vừa tối ưu hóa tài nguyên tính toán.

### 2.3.3 Cơ sở lý thuyết của phương pháp Robust Feature Extractor (RFE)

Để giải quyết bài toán trên, nghiên cứu đề xuất phương pháp luận Robust Feature Extractor (RFE). Thay vì tăng độ phức tạp của mô hình phân loại, RFE tập trung vào việc biến đổi dữ liệu thành các đặc trưng giàu thông tin dựa trên các nguyên lý thống kê và xử lý tín hiệu sau:

- **Phân tích động lực thời gian (Temporal Dynamics):** Dữ liệu cảm biến là dữ liệu chuỗi thời gian, do đó trạng thái hiện tại có quan hệ mật thiết với các trạng thái trước đó. RFE khai thác yếu tố này qua:
  - **Tốc độ thay đổi (Speed of Change):** Tính toán đạo hàm bậc nhất rời rạc của tín hiệu ( $value_t - value_{t-1}$ ) nhằm nắm bắt vận tốc và hướng biến thiên, giúp phát hiện các điểm gai bất thường.
  - **Bộ nhớ tạm thời (Temporal Memory):** Sử dụng các giá trị trễ (lagged values) để cung cấp ngữ cảnh ngắn hạn cho mô hình.
- **Thống kê cục bộ đa quy mô (Multi-Scale Local Statistics):** Một điểm dữ liệu cần được đánh giá trong ngữ cảnh của các điểm lân cận. RFE áp dụng kỹ thuật Cửa sổ trượt (Sliding Window) với nhiều kích thước khác nhau (ví dụ:  $w = 5, 10, 20$ ) để tính toán các tham số thống kê (trung bình, độ lệch chuẩn, độ xiên...).
  - Ý nghĩa: Giúp mô hình nhận diện hành vi của tín hiệu ở cả quy mô ngắn hạn (nhiều tức thời) và trung hạn (xu hướng trôi).
- **Phân tích biến động (Volatility Analysis):** Một đóng góp mới của RFE là việc áp dụng thống kê trượt trên chính đặc trưng "Tốc độ thay đổi".
  - Khi cảm biến gặp lỗi hoặc nhiễu loạn, độ lệch chuẩn của tốc độ thay đổi sẽ tăng cao. Đặc trưng này giúp lượng hóa sự "bất ổn" (instability) của tín hiệu một cách trực quan.
- **Làm tròn và khử nhiễu (Smoothing):** Sử dụng Trung bình động lũy thừa (EWMA - Exponentially Weighted Moving Average) để làm nổi bật xu hướng (trend) của dữ liệu và giảm thiểu tác động của nhiễu ngẫu nhiên, giúp phân biệt rõ hơn giữa nhiều môi trường và lỗi thực sự của thiết bị.



### 3 Khảo sát các giải pháp đã có

- 3.1 Khảo sát các ứng dụng, sản phẩm hoặc hệ thống liên quan
- 3.2 Khảo sát các mô hình, thuật toán, phương pháp
- 3.3 Bảng tổng hợp và phân tích khoảng trống



## 4 Phân tích yêu cầu

### 4.1 Mục đích

### 4.2 Phương pháp thu thập yêu cầu

### 4.3 Kết quả thu thập yêu cầu

### 4.4 Yêu cầu chức năng

Dựa trên phân tích hiện trạng và mô hình nghiệp vụ, các yêu cầu chức năng của hệ thống được chia thành các nhóm sau:

#### Nhóm 1: Tổng quan (Dashboard Overview)

**FR-03** Hệ thống phải hiển thị **thông số tổng quan** ngay khi đăng nhập, bao gồm: Tổng số người dùng, Số nông trại đang hoạt động (Active Farms), Số thiết bị đang kết nối (Connected Devices) và Thời gian hoạt động của hệ thống (System Uptime).

**FR-04** Hệ thống phải trực quan hóa dữ liệu hoạt động hàng tuần (Weekly Activity) dưới dạng **biểu đồ cột hoặc đường**, cho phép Admin so sánh nhanh số lượng người dùng mới, thiết bị mới và bài viết mới.

**FR-05** Hệ thống phải hiển thị danh sách **Cảnh báo hệ thống** (System Alerts) gần nhất (ví dụ: Nhiệt độ cao, Lịch bảo trì) ngay trên màn hình chính để Admin xử lý kịp thời.

**FR-06** Hệ thống phải cung cấp các nút **Thao tác nhanh** (Quick Actions) cho phép Admin tạo nhanh Người dùng mới, Nông trại mới hoặc Thêm thiết bị mới chỉ với 1 thao tác.

#### Nhóm 2: Quản lý Nông trại và Người dùng

**FR-07 (Quản lý Nông trại)** Hệ thống phải cho phép Admin thực hiện đầy đủ các chức năng (CRUD): Tạo mới, Xem chi tiết, Cập nhật thông tin và Xóa (hoặc vô hiệu hóa) một nông trại khỏi hệ thống.

**FR-08 (Quản lý Người dùng)** Hệ thống phải cho phép quản lý danh sách người dùng (Thương lái/Chủ vườn), bao gồm việc cấp phát tài khoản và phân quyền truy cập (Role-based Access Control) vào từng nông trại cụ thể.

**FR-09 (Gán thiết bị)** Hệ thống phải cho phép Admin thực hiện thao tác **gán (assign)** một hoặc nhiều thiết bị IoT cụ thể vào một nông trại đã định.

#### Nhóm 3: Quản lý và Giám sát Thiết bị

**FR-01 (Phát hiện lỗi)** Hệ thống phải có khả năng **tự động giám sát** dòng dữ liệu thời gian thực từ các cảm biến và phát hiện các trạng thái bất thường (ví dụ: mất kết nối quá 5 phút, giá trị vượt ngưỡng an toàn, hoặc dữ liệu bị "đóng băng").

**FR-02 (Cảnh báo)** Hệ thống phải **gửi cảnh báo tức thì** đến người quản trị (qua giao diện Dashboard, Email hoặc thông báo đẩy) ngay khi một lỗi cảm biến được xác nhận.

**FR-10 (Đăng ký thiết bị)** Hệ thống phải cho phép đăng ký thiết bị ESP32 mới vào hệ thống thông qua mã định danh duy nhất (Device ID/MAC Address).

**FR-11 (Giám sát trạng thái)** Hệ thống phải hiển thị trạng thái kết nối thời gian thực (Online/Offline) của từng thiết bị. Nếu thiết bị mất kết nối quá thời gian quy định (ví dụ: 5 phút), hệ thống phải tự động cập nhật trạng thái sang Offline.

**FR-12 (Điều khiển từ xa)** Hệ thống phải cho phép người dùng gửi lệnh điều khiển (Bật/Tắt) xuống các thiết bị chấp hành (Actuators) thông qua giao diện Web.



## 4.5 Yêu cầu phi chức năng

### NFR-01: Hiệu năng và Khả năng chịu tải

- **Môi trường triển khai:** Hệ thống phải hoạt động ổn định trên hạ tầng thử nghiệm bao gồm 01 thiết bị phần cứng thực (ESP32) kết hợp với mạng lưới thiết bị mô phỏng (Simulated Devices).
- **Khả năng chịu tải đồng thời:** Hệ thống phải duy trì kết nối và xử lý dữ liệu ổn định từ **50 thiết bị mô phỏng** gửi dữ liệu liên tục (tần suất 5 giây/bản tin) thông qua giao thức MQTT.
- **Độ trễ xử lý (Latency):**
  - Đối với thiết bị thực: Độ trễ hiển thị dữ liệu lên Dashboard **dưới 5 giây** (trong điều kiện mạng tiêu chuẩn).
  - Đối với thiết bị mô phỏng: Đảm bảo không xảy ra hiện tượng mất gói tin (packet loss) tại Message Broker khi tải đạt đỉnh.
- **Thời gian phản hồi Web:** Các thao tác truy xuất dữ liệu lịch sử hoặc tải danh sách thiết bị trên Web Admin phải hoàn tất trong vòng **dưới 2 giây**.

### NFR-02: Khả năng mở rộng

Hệ thống phải hỗ trợ việc thêm mới thiết bị hoặc nâng trại mà không cần tắt server để bảo trì (Zero-downtime scaling). Kiến trúc Microservices cho phép mở rộng độc lập module IoT Data Processor.

### NFR-03: Độ tin cậy và Tính sẵn sàng

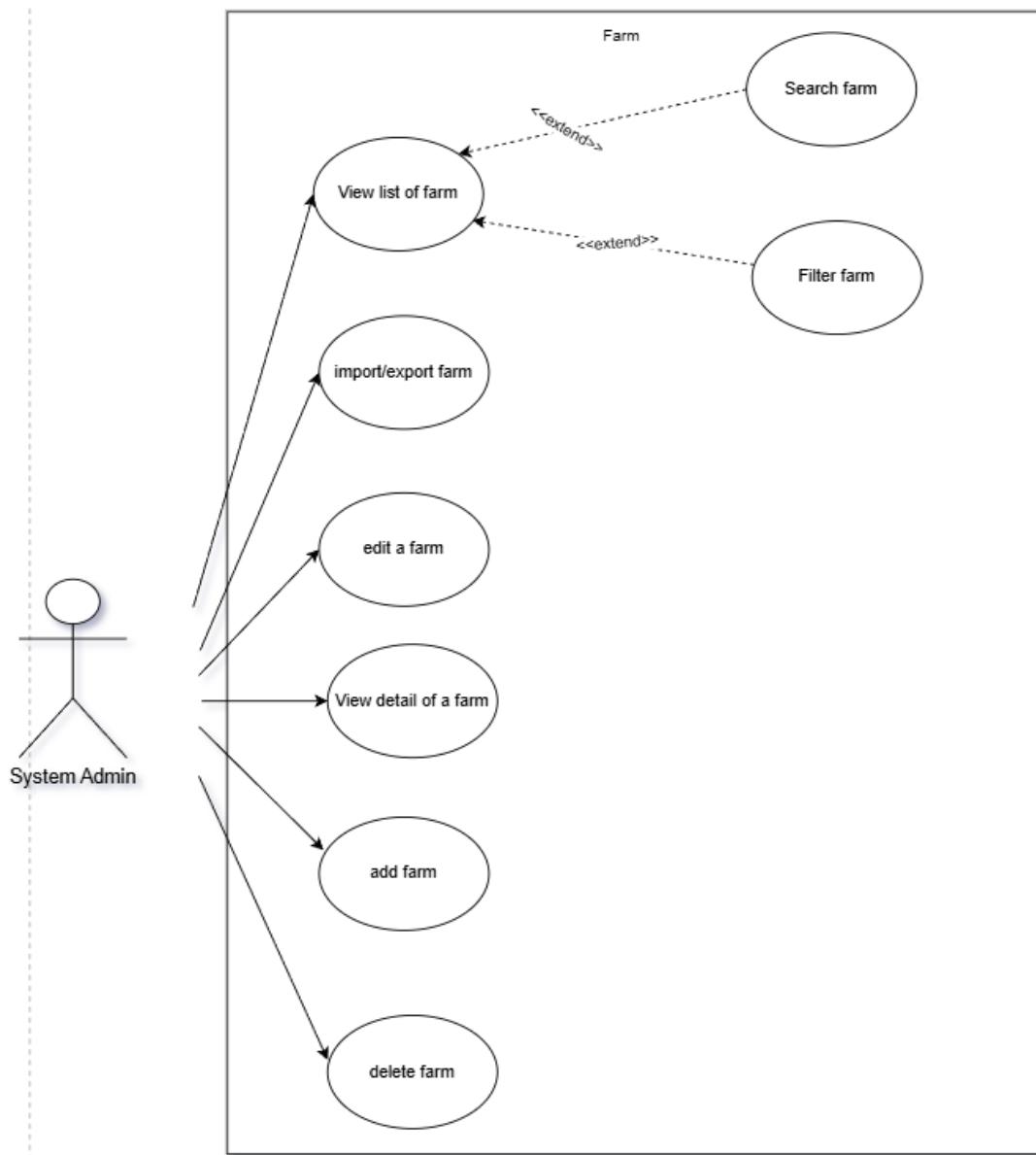
Hệ thống phải đảm bảo thời gian hoạt động (Uptime) đạt **99%**. Cơ chế **Auto-reconnect** phải hoạt động hiệu quả để Dashboard và Thiết bị tự động kết nối lại ngay khi đường truyền internet được khôi phục.

### NFR-04: Bảo mật

Dữ liệu truyền từ thiết bị về Server phải được mã hóa hoặc xác thực quyền truy cập. Mật khẩu người dùng phải được băm (hashing) trước khi lưu vào cơ sở dữ liệu.

## 4.6 Các mô hình phân tích UML

### 4.6.1 Use case: Monitor Farm



Hình 1: Sơ đồ quản lý farm



Mã số usecase	UC-01: Add Farm
Tên usecase	Tạo farm
Mô tả	Admin tạo 1 Farm
Actor	System Admin
Tiền điều kiện	Tài khoản Admin đã đăng nhập và có quyền truy cập module Farm.
Hậu điều kiện	Farm mới tạo được hiển thị trên danh sách farm
Trigger	Admin nhấn chọn menu “Farm” trên thanh điều hướng.
Luồng chính	<ol style="list-style-type: none"><li>Admin truy cập màn hình Farm.</li><li>Hệ thống kiểm tra quyền xem của Admin.</li><li>Admin chọn nút “Add Farm”.</li><li>Hệ thống hiển thị Form để nhập thông tin.</li><li>Admin điền thông tin cho farm rồi ấn nút “Save”</li><li>Hệ thống hiển thị cập nhật Farm mới vào danh sách</li></ol>
Quy tắc nghiệp vụ	<ul style="list-style-type: none"><li><b>BR-01 (Authorization):</b> Chỉ tài khoản có vai trò <b>System Admin</b> mới được phép tạo Farm.</li><li><b>BR-02 (Required Fields):</b> Các thông tin bắt buộc của Farm (ví dụ: Tên Farm, Vị trí) không được để trống.</li><li><b>BR-03 (Unique Farm Name):</b> Tên Farm phải là duy nhất và không được trùng lặp với các Farm đã tồn tại trong hệ thống.</li><li><b>BR-04 (Data Validation):</b> Dữ liệu nhập vào phải đúng định dạng và nằm trong phạm vi cho phép theo quy định của hệ thống.</li></ul>
Luồng thay thế / Mở rộng	<ul style="list-style-type: none"><li><b>E-01 (Cancel Create Farm):</b> Tại bước 4–5, Admin nhấn nút mũi tên “Back” ở góc trên bên trái hoặc nút “Cancel” cạnh nút “Save” → Hệ thống hủy thao tác tạo Farm và quay về màn hình danh sách Farm, không lưu dữ liệu.</li></ul>

Mã số usecase	UC-02: View Farm List
Tên usecase	Xem danh sách Farm
Mô tả	Admin xem danh sách các Farm có trong hệ thống
Actor	System Admin
Tiền điều kiện	Tài khoản Admin đã đăng nhập và có quyền truy cập module Farm
Hậu điều kiện	Danh sách Farm được hiển thị trên màn hình
Trigger	Admin nhấn chọn menu “Farm” trên thanh điều hướng
Luồng chính	<ol style="list-style-type: none"><li>Admin truy cập menu Farm.</li><li>Hệ thống kiểm tra quyền truy cập của Admin.</li><li>Hệ thống truy vấn dữ liệu Farm từ cơ sở dữ liệu.</li><li>Hệ thống hiển thị danh sách Farm.</li></ol>
Quy tắc nghiệp vụ	<ul style="list-style-type: none"><li><b>BR-01 (Authorization):</b> Chỉ tài khoản có vai trò <b>System Admin</b> mới được phép xem danh sách Farm.</li><li><b>BR-02 (Pagination):</b> Danh sách Farm được phân trang để đảm bảo hiệu năng hiển thị.</li></ul>



Mã số usecase	UC-05: Edit Farm
Tên usecase	Chỉnh sửa Farm
Mô tả	Admin chỉnh sửa thông tin Farm
Actor	System Admin
Tiền điều kiện	Farm tồn tại trong hệ thống
Hậu điều kiện	Thông tin Farm được cập nhật
Trigger	Admin chọn nút “Edit”
Luồng chính	<ol style="list-style-type: none"><li>Admin chọn Farm cần chỉnh sửa.</li><li>Hệ thống hiển thị form chỉnh sửa.</li><li>Admin cập nhật thông tin.</li><li>Admin nhấn “Save”.</li><li>Hệ thống lưu và cập nhật danh sách Farm.</li></ol>
Quy tắc nghiệp vụ	<ul style="list-style-type: none"><li><b>BR-01 (Authorization):</b> Chỉ System Admin được phép chỉnh sửa Farm.</li><li><b>BR-02 (Validation):</b> Dữ liệu chỉnh sửa phải hợp lệ.</li></ul>

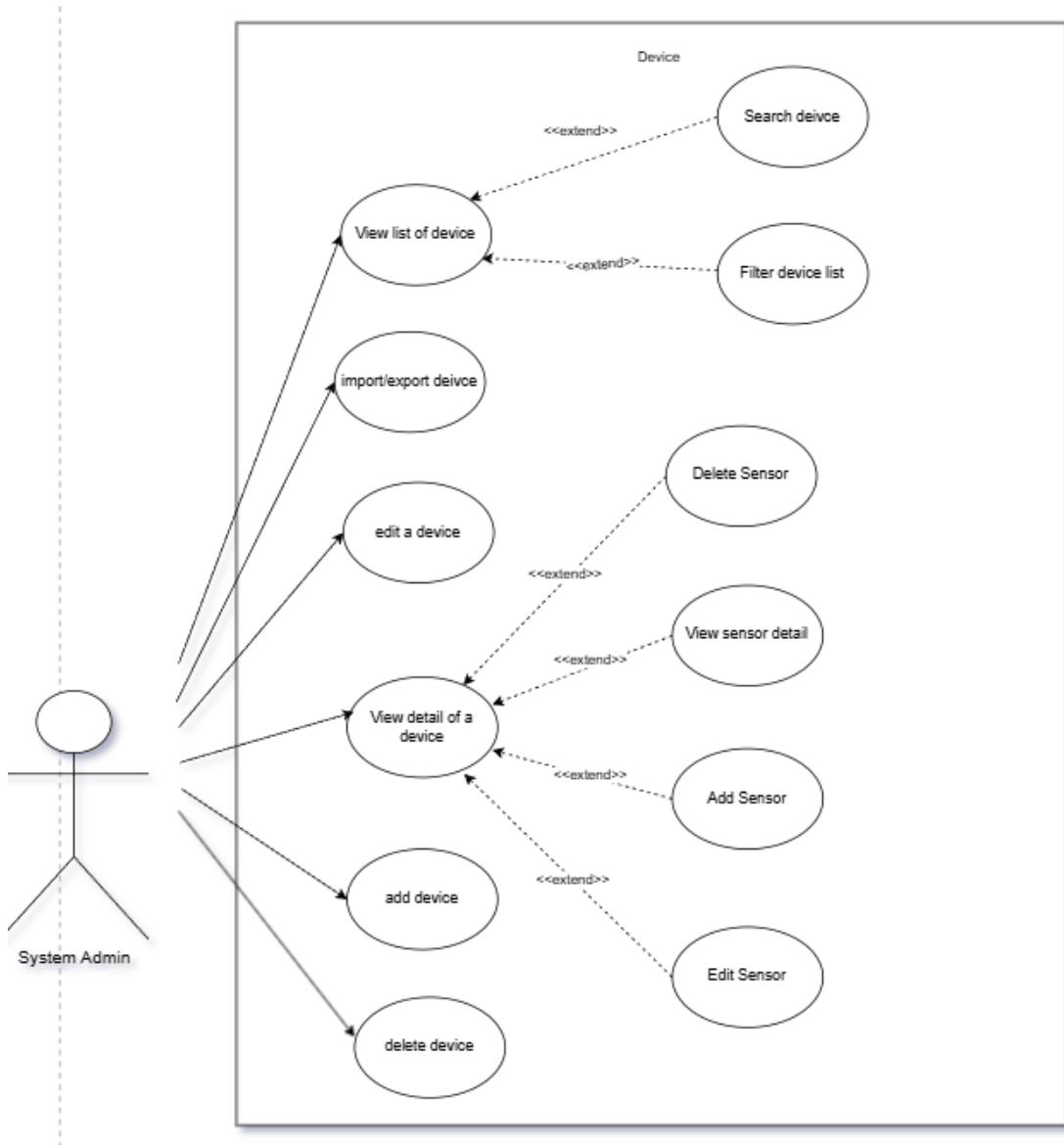
Mã số usecase	UC-06: Delete Farm
Tên usecase	Xóa Farm
Mô tả	Admin xóa Farm khỏi hệ thống
Actor	System Admin
Tiền điều kiện	Farm tồn tại
Hậu điều kiện	Farm bị xóa khỏi danh sách
Trigger	Admin chọn nút “Delete”
Luồng chính	<ol style="list-style-type: none"><li>Admin chọn Farm cần xóa.</li><li>Hệ thống hiển thị hộp thoại xác nhận.</li><li>Admin xác nhận xóa.</li><li>Hệ thống xóa Farm và cập nhật danh sách.</li></ol>
Quy tắc nghiệp vụ	<ul style="list-style-type: none"><li><b>BR-01 (Confirmation):</b> Phải xác nhận trước khi xóa.</li></ul>



Mã số usecase	UC-07: Import/Export Farm
Tên usecase	Import / Export Farm
Mô tả	Admin nhập hoặc xuất dữ liệu Farm
Actor	System Admin
Tiền điều kiện	Admin có quyền truy cập module Farm
Hậu điều kiện	Dữ liệu Farm được nhập hoặc xuất thành công
Trigger	Admin chọn chức năng Import hoặc Export
Luồng chính	<ol style="list-style-type: none"><li>Admin chọn Import hoặc Export.</li><li>Hệ thống hiển thị tùy chọn file.</li><li>Admin xác nhận thao tác.</li><li>Hệ thống xử lý dữ liệu.</li></ol>
Quy tắc nghiệp vụ	<ul style="list-style-type: none"><li><b>BR-01 (File Format):</b> Chỉ hỗ trợ định dạng cho phép (CSV, Excel).</li></ul>

Mã số usecase	UC-08: View Farm Detail
Tên usecase	Xem chi tiết Farm
Mô tả	Admin xem thông tin chi tiết của một Farm trong hệ thống
Actor	System Admin
Tiền điều kiện	<ul style="list-style-type: none"><li>Admin đã đăng nhập</li><li>Danh sách Farm đang được hiển thị</li></ul>
Hậu điều kiện	Thông tin chi tiết của Farm được hiển thị
Trigger	Admin nhấn chọn một Farm trong danh sách
Luồng chính	<ol style="list-style-type: none"><li>Admin xem danh sách Farm.</li><li>Admin nhấn chọn một Farm bất kỳ.</li><li>Hệ thống kiểm tra quyền truy cập của Admin.</li><li>Hệ thống truy vấn thông tin chi tiết của Farm.</li><li>Hệ thống hiển thị màn hình chi tiết Farm.</li></ol>
Quy tắc nghiệp vụ	<ul style="list-style-type: none"><li><b>BR-01 (Authorization):</b> Chỉ tài khoản có vai trò <b>System Admin</b> mới được phép xem chi tiết Farm.</li><li><b>BR-02 (Data Integrity):</b> Farm phải tồn tại trong hệ thống tại thời điểm truy vấn.</li></ul>
Luồng thay thế / Mở rộng	<ul style="list-style-type: none"><li><b>E-01 (Back to List):</b> Tại màn hình chi tiết, Admin nhấn nút mũi tên “Back” ở góc trên bên trái → Hệ thống quay về màn hình danh sách Farm.</li></ul>

#### 4.6.2 Use case: Monitor Device



(a) Sơ đồ quản lý thiết bị



Mã số usecase	UC-01: View device list
Tên usecase	Xem danh sách thiết bị
Mô tả	Admin xem danh sách các thiết bị trong hệ thống
Actor	System Admin
Tiền điều kiện	Admin đã đăng nhập và có quyền truy cập module Device
Hậu điều kiện	Danh sách thiết bị được hiển thị
Trigger	Admin chọn menu “Device”
Luồng chính	<ol style="list-style-type: none"><li>Admin truy cập module Device.</li><li>Hệ thống kiểm tra quyền truy cập.</li><li>Hệ thống hiển thị danh sách thiết bị.</li></ol>
Quy tắc nghiệp vụ	<ul style="list-style-type: none"><li><b>BR-01 (Authorization):</b> Chỉ System Admin được phép xem danh sách thiết bị.</li></ul>

Mã số usecase	UC-02: View device detail
Tên usecase	Xem chi tiết thiết bị
Mô tả	Admin xem thông tin chi tiết của một thiết bị
Actor	System Admin
Tiền điều kiện	Danh sách thiết bị đã được hiển thị
Hậu điều kiện	Thông tin chi tiết thiết bị được hiển thị
Trigger	Admin chọn một thiết bị trong danh sách
Luồng chính	<ol style="list-style-type: none"><li>Admin nhấn chọn một thiết bị.</li><li>Hệ thống tải dữ liệu chi tiết thiết bị.</li><li>Hệ thống hiển thị màn hình chi tiết thiết bị.</li></ol>
Luồng thay thế / Mở rộng	<ul style="list-style-type: none"><li><b>E-01 (Back):</b> Admin nhấn nút “Back” → hệ thống quay lại danh sách thiết bị.</li></ul>



Mã số usecase	UC-03: Add device
Tên usecase	Thêm thiết bị
Mô tả	Admin tạo mới một thiết bị
Actor	System Admin
Tiền điều kiện	Admin có quyền quản lý thiết bị
Hậu điều kiện	Thiết bị mới được lưu và hiển thị trong danh sách
Trigger	Admin nhấn nút “Add Device”
Luồng chính	<ol style="list-style-type: none"><li>Admin mở màn hình Device.</li><li>Admin chọn “Add Device”.</li><li>Hệ thống hiển thị form nhập thông tin.</li><li>Admin nhập thông tin và nhấn “Save”.</li><li>Hệ thống lưu thiết bị và cập nhật danh sách.</li></ol>
Quy tắc nghiệp vụ	<ul style="list-style-type: none"><li><b>BR-01:</b> Các trường bắt buộc không được để trống.</li></ul>

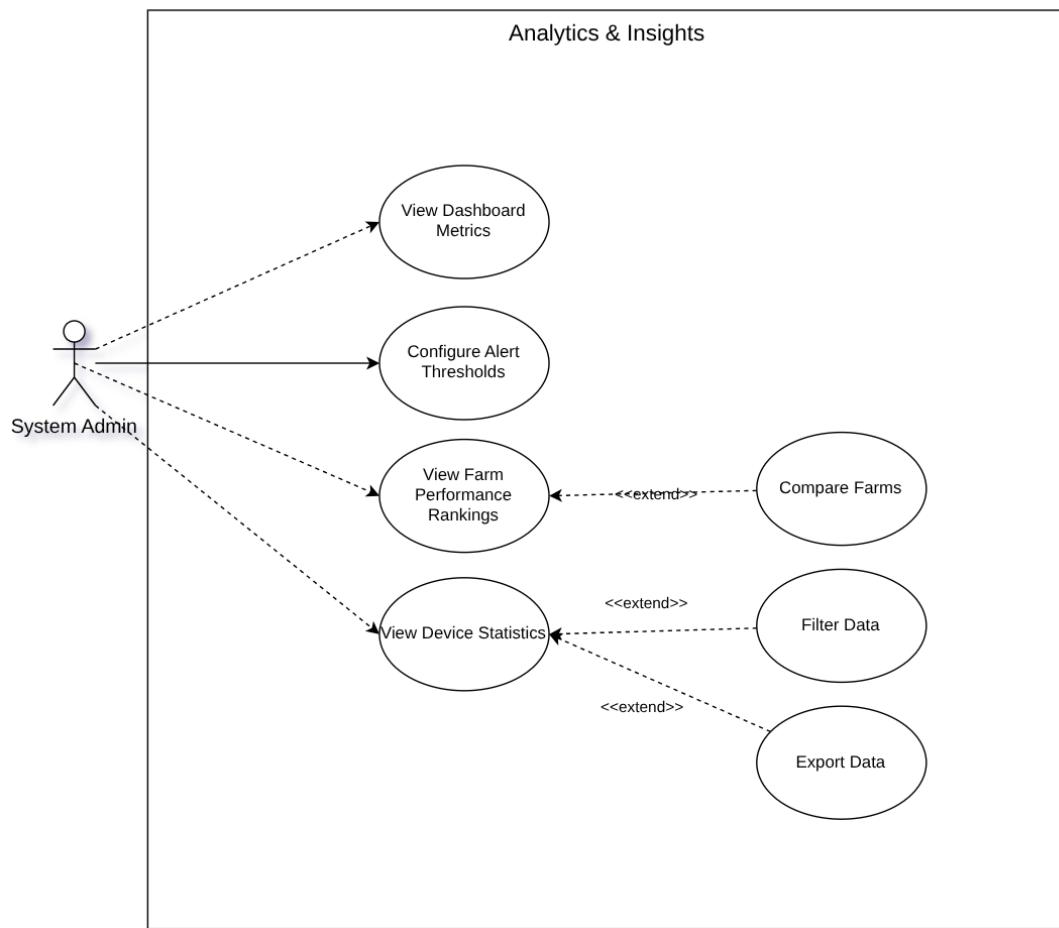
Mã số usecase	UC-04: Edit device
Tên usecase	Chỉnh sửa thiết bị
Mô tả	Admin cập nhật thông tin thiết bị
Actor	System Admin
Tiền điều kiện	Thiết bị đã tồn tại
Hậu điều kiện	Thông tin thiết bị được cập nhật
Trigger	Admin chọn “Edit” tại thiết bị
Luồng chính	<ol style="list-style-type: none"><li>Admin chọn thiết bị cần chỉnh sửa.</li><li>Hệ thống hiển thị form chỉnh sửa.</li><li>Admin cập nhật thông tin và nhấn “Save”.</li><li>Hệ thống lưu thay đổi.</li></ol>
Quy tắc nghiệp vụ	<ul style="list-style-type: none"><li><b>BR-01 (Authorization):</b> Chỉ tài khoản có vai trò <b>System Admin</b> mới được phép chỉnh sửa thiết bị.</li><li><b>BR-02 (Data Integrity):</b> Thiết bị phải tồn tại trong hệ thống tại thời điểm chỉnh sửa.</li><li><b>BR-03 (Data Validation):</b> Dữ liệu chỉnh sửa phải đúng định dạng và nằm trong phạm vi cho phép của hệ thống.</li></ul>
Luồng thay thế / Mở rộng	<ul style="list-style-type: none"><li><b>E-01 (Cancel Edit):</b> Tại bước 3–4, Admin nhấn nút “Cancel” hoặc “Back” → Hệ thống hủy thao tác chỉnh sửa và không lưu dữ liệu.</li></ul>



Mã số usecase	UC-05: Delete device
Tên usecase	Xóa thiết bị
Mô tả	Admin xóa thiết bị khỏi hệ thống
Actor	System Admin
Tiền điều kiện	Thiết bị tồn tại trong hệ thống
Hậu điều kiện	Thiết bị bị xóa khỏi danh sách
Trigger	Admin chọn “Delete” tại thiết bị
Luồng chính	<ol style="list-style-type: none"><li>1. Admin chọn thiết bị cần xóa.</li><li>2. Hệ thống hiển thị hộp thoại xác nhận.</li><li>3. Admin xác nhận xóa.</li><li>4. Hệ thống xóa thiết bị.</li></ol>
Quy tắc nghiệp vụ	<ul style="list-style-type: none"><li>• <b>BR-01 (Authorization):</b> Chỉ tài khoản có vai trò <b>System Admin</b> mới được phép xóa thiết bị.</li><li>• <b>BR-02 (Data Integrity):</b> Thiết bị phải tồn tại trong hệ thống tại thời điểm xóa.</li></ul>
Luồng thay thế / Mở rộng	<ul style="list-style-type: none"><li>• <b>E-01 (Cancel Delete):</b> Tại bước xác nhận, Admin chọn “Cancel” → Hệ thống hủy thao tác xóa thiết bị.</li><li>• <b>E-02 (Delete Not Allowed):</b> Nếu thiết bị còn sensor liên kết → Hệ thống hiển thị thông báo không thể xóa thiết bị.</li></ul>

Mã số usecase	UC-06: Import/Export device
Tên usecase	Import / Export thiết bị
Mô tả	Admin nhập hoặc xuất danh sách thiết bị
Actor	System Admin
Tiền điều kiện	Admin có quyền quản lý thiết bị
Hậu điều kiện	Dữ liệu thiết bị được nhập hoặc xuất thành công
Trigger	Admin chọn “Import/Export Device”
Luồng chính	<ol style="list-style-type: none"><li>1. Admin chọn chức năng Import hoặc Export.</li><li>2. Hệ thống xử lý dữ liệu.</li><li>3. Hệ thống thông báo kết quả.</li></ol>

#### 4.6.3 Use case: Monitor System Analytics

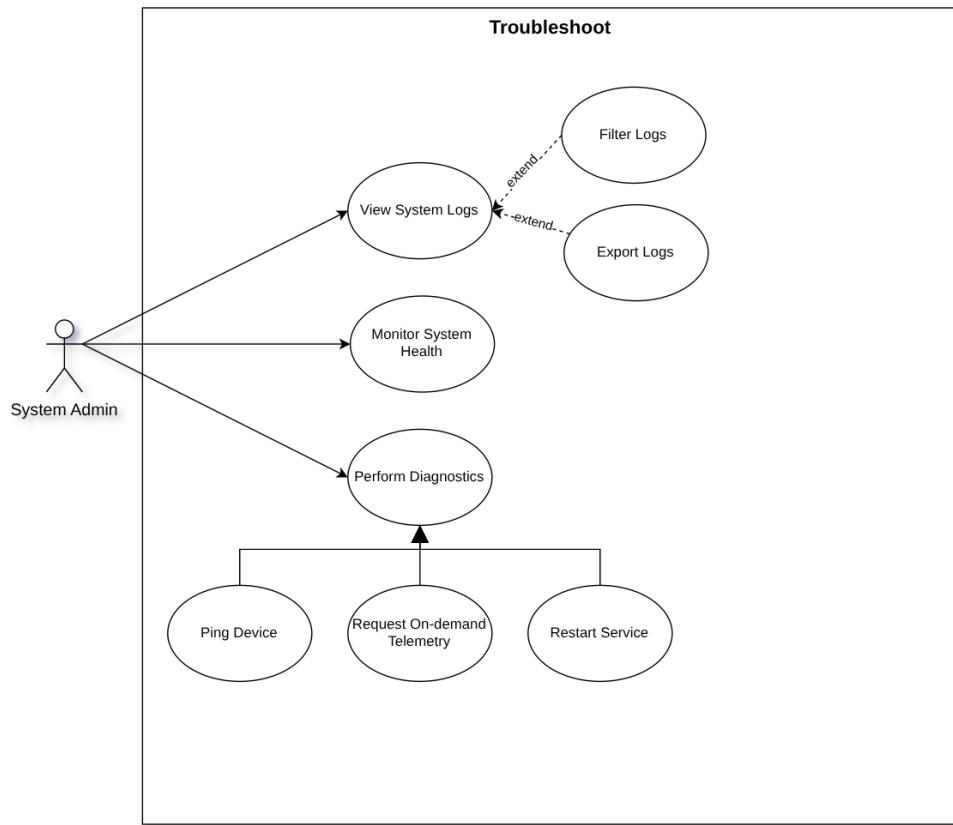


Hình 3: Sơ đồ luồng quản lý và tạo báo cáo



Mã số usecase	UC-01: Monitor System Analytics
Tên usecase	Giám sát Phân tích Hệ thống
Mô tả	Admin theo dõi tổng quan các chỉ số hoạt động, xếp hạng hiệu suất nông trại và thống kê thiết bị trên Dashboard.
Actor	System Admin
Tiền điều kiện	Tài khoản Admin đã đăng nhập và có quyền truy cập module Analytics.
Hậu điều kiện	Dữ liệu thống kê được hiển thị đầy đủ và cập nhật mới nhất.
Trigger	Admin nhấn chọn menu “Analytics & Insights” trên thanh điều hướng.
Luồng chính	<ol style="list-style-type: none"><li>Admin truy cập màn hình Analytics.</li><li>Hệ thống kiểm tra quyền xem của Admin.</li><li>Hệ thống tải dữ liệu tổng hợp từ cơ sở dữ liệu.</li><li>Hệ thống hiển thị Dashboard Metrics (doanh thu, sản lượng, nhiệt độ trung bình).</li><li>Hệ thống hiển thị Farm Performance Rankings (xếp hạng hiệu suất).</li><li>Hệ thống hiển thị Device Statistics (biểu đồ trạng thái Online/Offline).</li></ol>
Quy tắc nghiệp vụ	<ul style="list-style-type: none"><li><b>BR-01 (Data Latency):</b> Dữ liệu hiển thị trên Dashboard phải được cập nhật gần thời gian thực (Real-time), độ trễ tối đa không quá 30 giây.</li><li><b>BR-02 (Default View):</b> Mặc định hiển thị dữ liệu tổng hợp của toàn hệ thống trong 7 ngày gần nhất.</li><li><b>BR-03 (Access Control):</b> Chỉ tài khoản Admin có quyền “Manage” mới hiển thị nút cấu hình “Configure Alert Thresholds”.</li><li><b>BR-04 (Ranking Logic):</b> Xếp hạng nông trại dựa trên chỉ số KPI tổng hợp (tỷ lệ sản lượng / mức tiêu thụ năng lượng).</li></ul>
Luồng thay thế / Mở rộng	<ul style="list-style-type: none"><li><b>E-01 (Filter):</b> Tại bước 4-6, Admin chọn bộ lọc thời gian hoặc khu vực → Hệ thống truy vấn lại và cập nhật hiển thị.</li><li><b>E-02 (Export):</b> Admin bấm “Export” → Hệ thống kiểm tra định dạng file (PDF/CSV) và tiến hành tải xuống.</li><li><b>E-03 (Alert Config):</b> Admin thay đổi ngưỡng cảnh báo → Hệ thống lưu quy tắc mới vào cơ sở dữ liệu và áp dụng ngay lập tức.</li></ul>

#### 4.6.4 Use case: Troubleshoot System



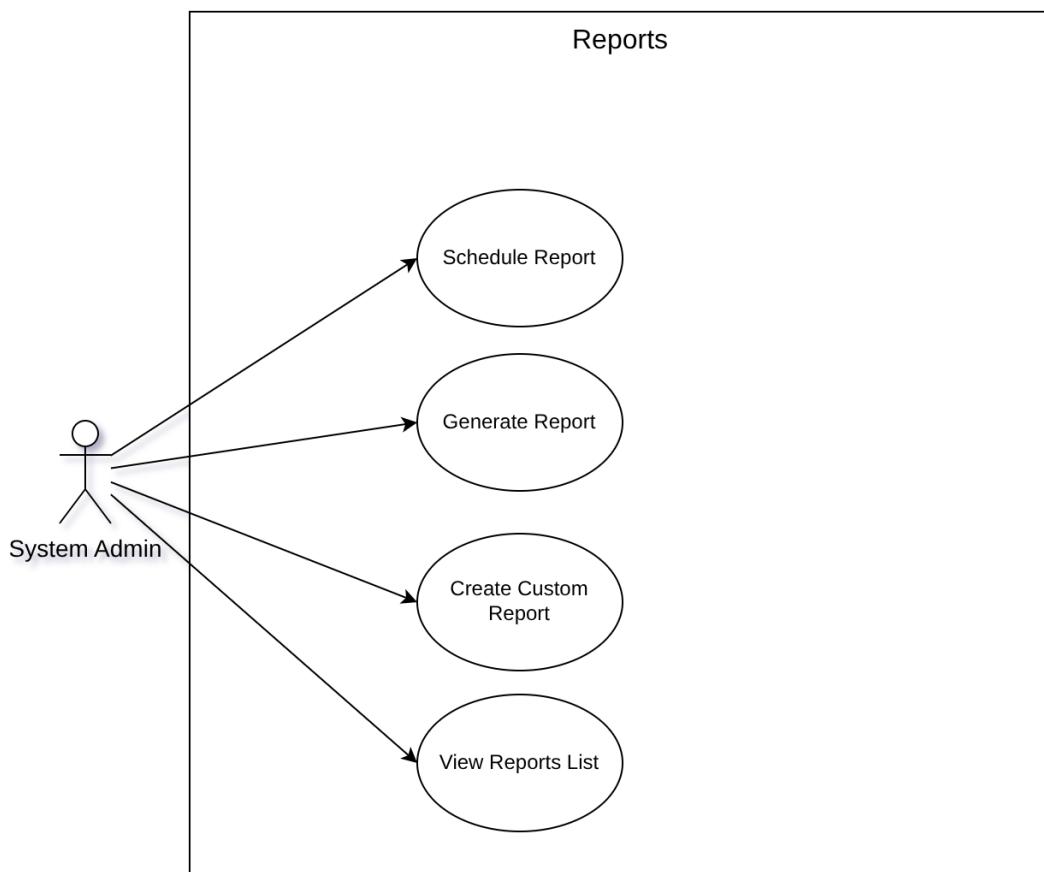
(a) Sơ đồ xử lý sự cố và theo dõi tình trạng hệ thống

Hình mô tả lần lượt quá trình kiểm tra dịch vụ, khắc phục từ xa, và cách truy xuất log, xem lịch sử cảnh báo để đối soát.



Mã số usecase	UC-02: Troubleshoot System
Tên usecase	Kiểm tra và xử lý sự cố
Mô tả	Quy trình xem tình trạng sức khỏe hệ thống, đọc log chi tiết và gửi lệnh khắc phục sự cố từ xa.
Actor	System Admin
Tiền điều kiện	Hệ thống phát hiện sự cố (Alert) hoặc Admin thực hiện kiểm tra định kỳ.
Hậu điều kiện	Nguyên nhân lỗi được xác định hoặc lệnh sửa chữa đã được gửi đi.
Trigger	Admin nhận được thông báo lỗi hoặc truy cập menu “Troubleshoot”.
Luồng chính	<ol style="list-style-type: none"><li>Admin truy cập trang Troubleshoot.</li><li>Hệ thống hiển thị đèn trạng thái (Health Indicators) của Server, API và Mạng.</li><li>Hệ thống truy xuất và hiển thị danh sách 50 dòng System Logs gần nhất.</li><li>Admin phân tích các chỉ số màu đỏ (Lỗi) và nội dung log để xác định nguyên nhân.</li></ol>
Quy tắc nghiệp vụ	<ul style="list-style-type: none"><li><b>BR-05 (Health Coding):</b> Trạng thái hệ thống phải được mã hóa màu: Xanh (Ôn định), Vàng (Cảnh báo tải &gt;80%), Đỏ (Lỗi/Ngừng hoạt động).</li><li><b>BR-06 (Log Privacy):</b> Các thông tin nhạy cảm trong log (như Password, API Key) phải được che dấu (masking) bằng ký tự ‘*****’ trước khi hiển thị.</li><li><b>BR-07 (Action Permission):</b> Chỉ tài khoản Super Admin mới có quyền thực thi các lệnh tác động hệ thống như “Restart Service”.</li></ul>
Luồng thay thế / Mở rộng	<ul style="list-style-type: none"><li><b>E-01 (Perform Diagnostics):</b> Admin chọn thiết bị lỗi → chọn hành động (Ping/Restart Service/Telemetry) → Hệ thống thực thi lệnh và trả về kết quả (Success/Timeout).</li><li><b>E-02 (Export Logs):</b> Admin bấm “Export Logs” → Hệ thống tổng hợp log lỗi thành file CSV/JSON và kích hoạt tải xuống.</li><li><b>E-03 (Filter Logs):</b> Admin lọc theo mức độ “Error” → Hệ thống ẩn các log thông thường, chỉ hiện log lỗi.</li></ul>

#### 4.6.5 Use case: Manage Reports



**Hình 5:** Sơ đồ luồng quản lý và tạo báo cáo

Hình thể hiện quy trình xem danh sách báo cáo, tạo mới, lập lịch hoặc tạo báo cáo tùy chỉnh và cập nhật lại danh sách.



Mã số usecase	UC-03: Manage Reports
Tên usecase	Quản lý và tạo báo cáo
Mô tả	Admin xem danh sách báo cáo đã lưu, tạo báo cáo mới tức thì hoặc lập lịch gửi báo cáo tự động.
Actor	System Admin
Tiền điều kiện	Admin đã đăng nhập thành công.
Hậu điều kiện	Danh sách báo cáo được cập nhật; file báo cáo được tạo ra.
Trigger	Admin truy cập trang “Reports”.
Luồng chính	<ol style="list-style-type: none"><li>Admin mở trang Reports.</li><li>Hệ thống truy vấn cơ sở dữ liệu báo cáo.</li><li>Hệ thống hiển thị danh sách Recent Reports (bao gồm: Tên, Ngày tạo, Người tạo, Loại báo cáo).</li><li>Admin xem thông tin hoặc tải về các báo cáo cũ.</li></ol>
Quy tắc nghiệp vụ	<ul style="list-style-type: none"><li><b>BR-08 (Retention Policy):</b> Báo cáo chỉ được lưu trữ trực tuyến trong 90 ngày. Sau thời gian này, dữ liệu sẽ được chuyển sang lưu trữ lạnh (Archive).</li><li><b>BR-09 (File Format):</b> Hệ thống phải hỗ trợ xuất báo cáo ra hai định dạng: PDF (để in ấn/trình ký) và Excel/CSV (để phân tích số liệu).</li><li><b>BR-10 (Schedule Limit):</b> Mỗi tài khoản Admin chỉ được thiết lập tối đa 5 lịch báo cáo tự động để đảm bảo hiệu năng hệ thống.</li></ul>
Luồng thay thế / Mở rộng	<ul style="list-style-type: none"><li><b>E-01 (Generate):</b> Admin bấm “Generate New” → Hệ thống thu thập dữ liệu hiện tại và tạo file báo cáo ngay lập tức.</li><li><b>E-02 (Schedule):</b> Admin bấm “Schedule” → Hệ thống hiển thị form chọn tần suất (hàng ngày/tuần) → Lưu lịch chạy tự động (Cron job).</li><li><b>E-03 (Custom Report):</b> Admin chọn các trường dữ liệu tùy chỉnh → Bấm “Create” → Hệ thống tạo báo cáo theo mẫu riêng.</li></ul>



## 5 Giải pháp đề xuất

### 5.1 Mục tiêu của giải pháp

Giải pháp được đề xuất nhằm xây dựng một hệ thống giám sát nông nghiệp thông minh dựa trên công nghệ Internet vạn vật (IoT), với mục tiêu chính là cung cấp một nền tảng quản lý toàn diện cho các nông trại hiện đại. Hệ thống được thiết kế để đáp ứng các mục tiêu cụ thể sau:

#### 5.1.1 Mục tiêu 1: Tự động hóa quá trình thu thập và xử lý dữ liệu môi trường

Hệ thống phải có khả năng tự động thu thập dữ liệu từ mạng lưới cảm biến phân tán (nhiệt độ, độ ẩm không khí, độ ẩm đất, ánh sáng) và xử lý chúng theo thời gian thực mà không cần sự can thiệp thủ công. Mục tiêu này nhằm giảm thiểu công sức quản lý của người nông dân, đồng thời đảm bảo tính liên tục và chính xác của dữ liệu giám sát.

#### 5.1.2 Mục tiêu 2: Phát hiện và cảnh báo sớm các sự cố hệ thống

Hệ thống phải tích hợp các cơ chế phát hiện lỗi tự động (như mất kết nối thiết bị, dữ liệu bất thường, hoặc giá trị vượt ngưỡng an toàn) và gửi cảnh báo tức thì đến người quản trị. Mục tiêu này giúp giảm thiểu thời gian phản ứng khi xảy ra sự cố, từ đó hạn chế thiệt hại về năng suất cây trồng và tài sản.

#### 5.1.3 Mục tiêu 3: Cung cấp giao diện quản lý trực quan và dễ sử dụng

Hệ thống phải cung cấp một Dashboard Web Admin với các tính năng trực quan hóa dữ liệu (biểu đồ, bảng thống kê) và các thao tác quản lý nhanh (Quick Actions), giúp người quản trị nắm bắt tình trạng hệ thống một cách nhanh chóng và thực hiện các tác vụ quản lý hiệu quả.

#### 5.1.4 Mục tiêu 4: Đảm bảo khả năng mở rộng và hiệu năng cao

Hệ thống phải được thiết kế với kiến trúc có khả năng mở rộng (Scalable Architecture), cho phép thêm mới thiết bị hoặc nông trại mà không cần tắt server để bảo trì. Đồng thời, hệ thống phải đảm bảo xử lý ổn định dữ liệu từ hàng chục thiết bị đồng thời với độ trễ thấp.

#### 5.1.5 Mục tiêu 5: Tối ưu hóa chi phí vận hành và bảo trì

Hệ thống phải sử dụng các công nghệ mã nguồn mở và kiến trúc tiết kiệm tài nguyên, giúp giảm thiểu chi phí triển khai và vận hành. Việc sử dụng các giao thức nhẹ như MQTT và cơ sở dữ liệu chuyên dụng cho Time-series (TimescaleDB) nhằm tối ưu hóa hiệu năng xử lý dữ liệu lớn.

#### 5.1.6 Mục tiêu 6: Đảm bảo tính bảo mật và độ tin cậy

Hệ thống phải tích hợp các cơ chế bảo mật đa lớp (xác thực người dùng, phân quyền truy cập, mã hóa dữ liệu) và đảm bảo thời gian hoạt động (Uptime) đạt 99%, với khả năng tự động kết nối lại khi mất kết nối mạng.



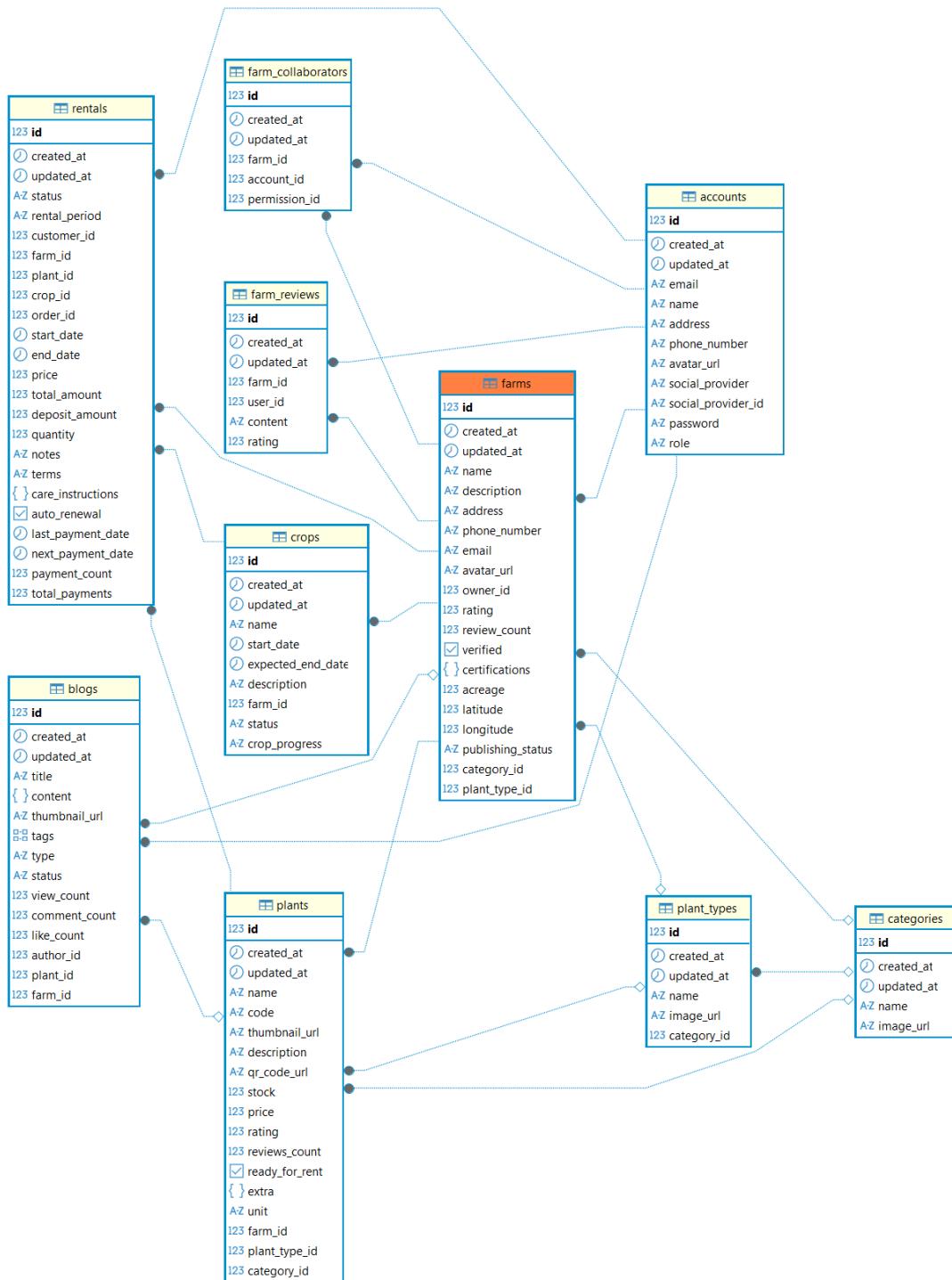
## 5.2 Thiết kế kiến trúc (Mức khái niệm)

Mô hình Quan hệ Thực thể (ERD) chi tiết của hệ thống được trình bày tại Phụ lục A. Thiết kế này đảm bảo tính toàn vẹn dữ liệu và khả năng mở rộng cho các nghiệp vụ giám sát nông nghiệp thông minh.

Hệ thống cơ sở dữ liệu được tổ chức thành các phân hệ chính như sau:

### 5.2.1 Phân hệ Quản lý Nông trại:

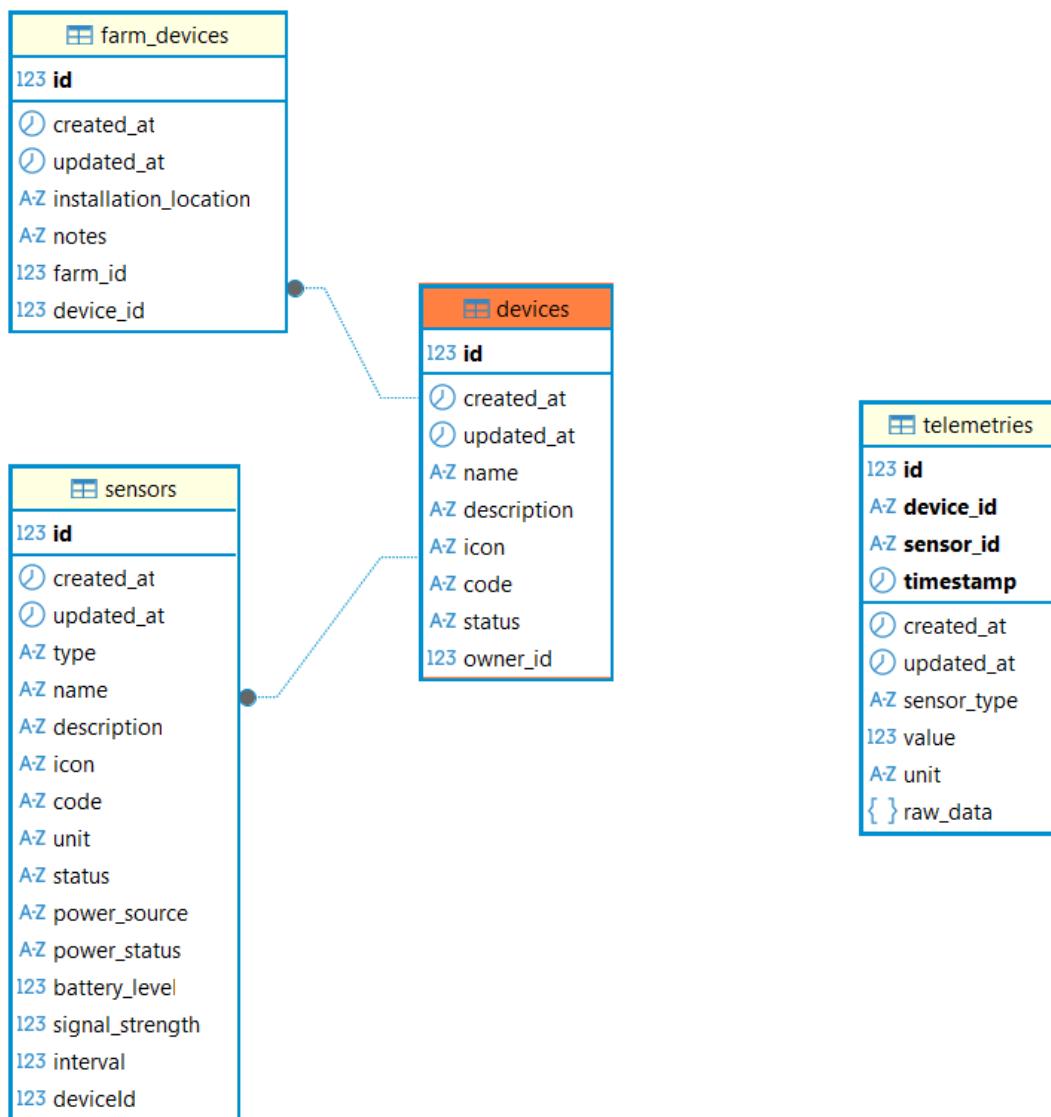
Đây là nhóm thực thể trung tâm, bao gồm bảng **farms** (lưu thông tin nông trại) liên kết 1-nhiều với **plants** (cây trồng) và **crops** (mùa vụ). Cấu trúc này cho phép quản lý chi tiết quy trình canh tác từ lúc xuồng giống đến khi thu hoạch, bao gồm cả việc theo dõi sức khỏe cây trồng qua bảng **plant\_health\_statuses**.



Hình 6: Sơ đồ thực thể quan hệ của phân hệ Quản lý Nông trại

### 5.2.2 Phân hệ IoT và Thu thập dữ liệu:

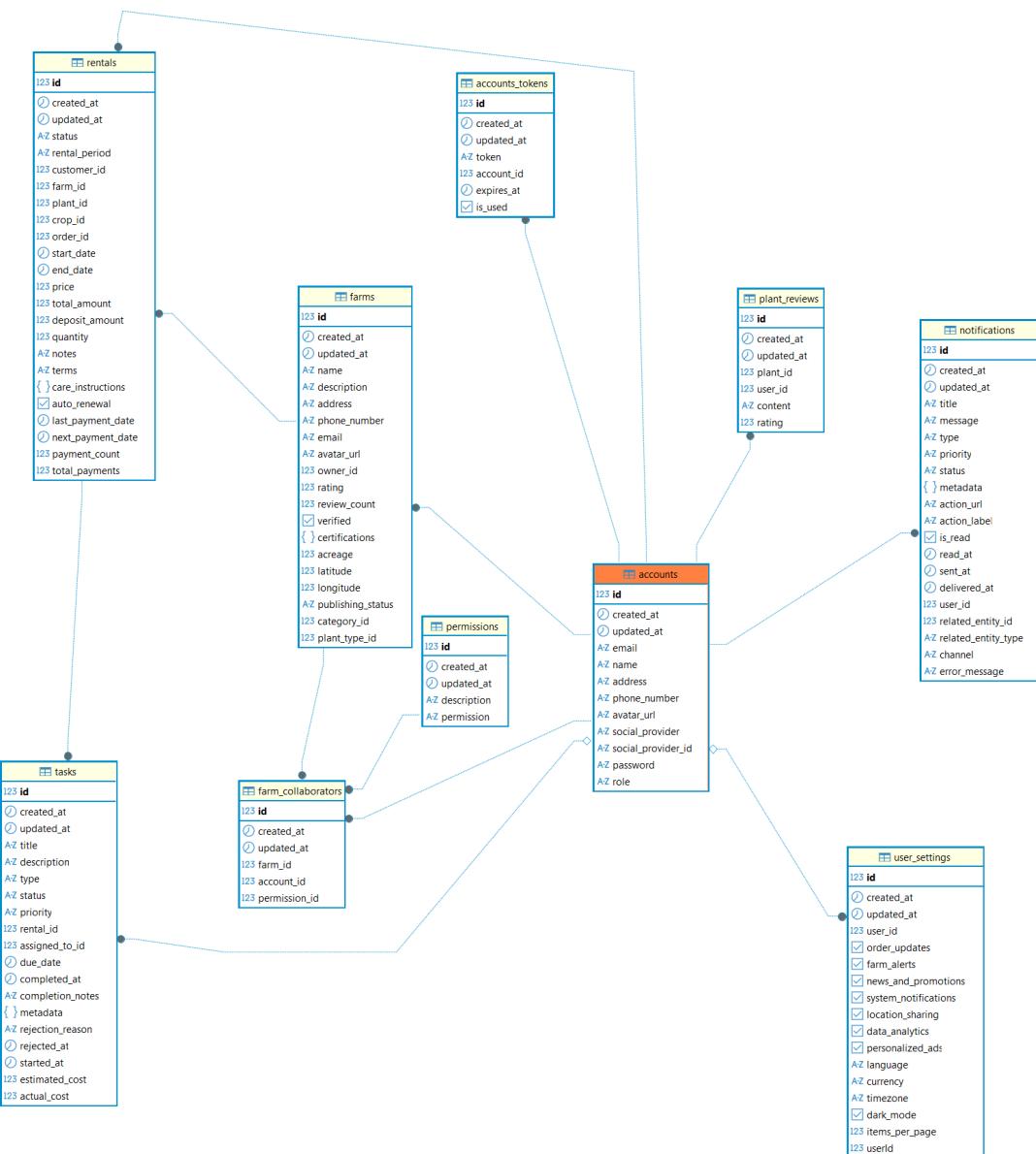
Các thiết bị phần cứng được quản lý qua bảng **devices** và **sensors**. Dữ liệu quan trọng nhất của hệ thống là **telemtries**, nơi lưu trữ hàng triệu bản ghi dữ liệu cảm biến (nhiệt độ, độ ẩm, ánh sáng) được gửi về liên tục. Quan hệ giữa **devices** và **farms** giúp xác định thiết bị nào đang hoạt động tại khu vực nào.



Hình 7: Sơ đồ thực thể quan hệ của phân hệ IoT và Thu thập dữ liệu

### 5.2.3 Phân hệ Người dùng và Phân quyền:

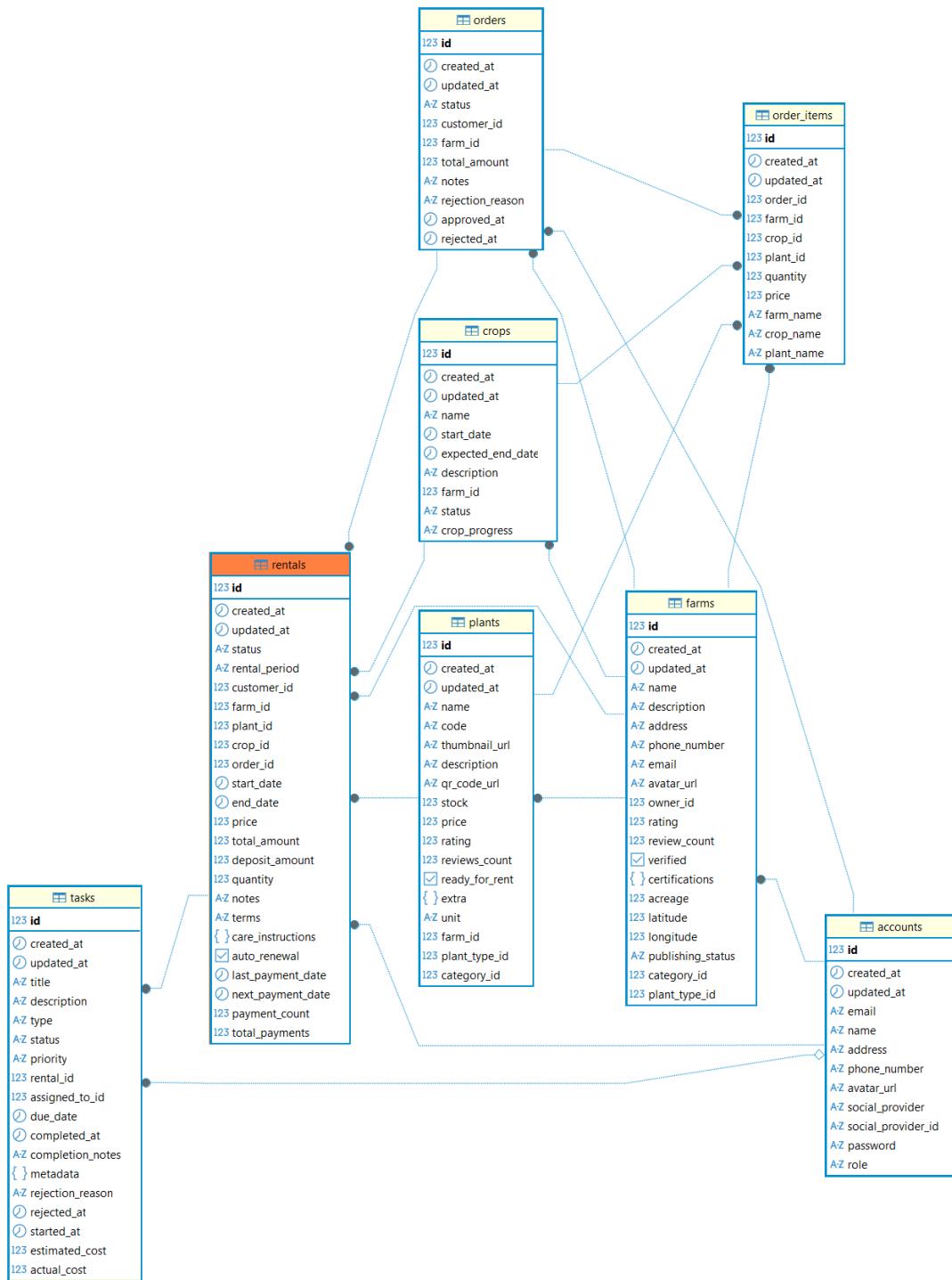
Hệ thống sử dụng bảng **accounts** để quản lý người dùng, kết hợp với **permissions** và **user\_settings** để thực hiện cơ chế phân quyền, đảm bảo chỉ Admin hoặc chủ nông trại mới có quyền truy cập các dữ liệu nhạy cảm hoặc thực hiện cấu hình thiết bị.



Hình 8: Sơ đồ thực thể quan hệ của phân hệ Người dùng và Phân quyền

#### 5.2.4 Phân hệ Kinh doanh

Các nghiệp vụ kinh doanh như thuê thiết bị, đặt hàng, thanh toán được quản lý thông qua bảng **rentals** và **orders**.



Hình 9: Sơ đồ thực thể quan hệ của phân hệ Kinh doanh

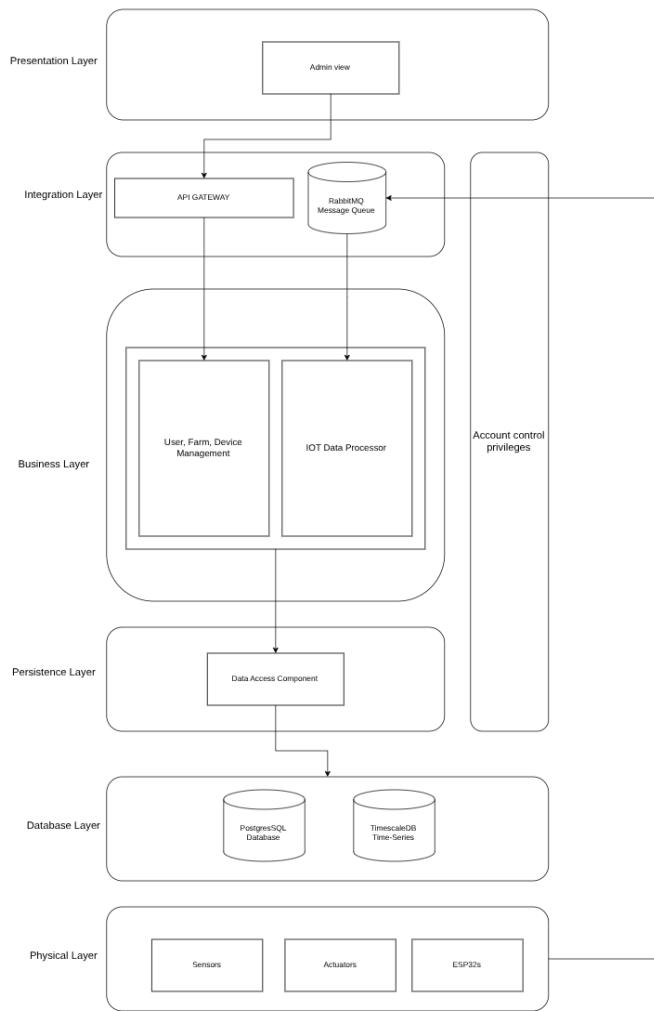
Và các bảng khác được trình bày tại Phụ lục A.

### 5.3 Thiết kế Kiến trúc Hệ thống

#### 5.3.1 Kiến trúc Phân lớp (Layered Architecture)

Kiến trúc hệ thống được thiết kế theo mô hình Phân lớp (Layered Architecture) kết hợp với hướng dịch vụ (Service-oriented). Việc phân chia này giúp tách biệt các mối quan tâm (Separation of Concerns), dễ dàng bảo trì và mở rộng độc lập từng thành phần.

Sơ đồ dưới đây minh họa các tầng logic và luồng dữ liệu trong hệ thống:



**Hình 10: Sơ đồ Kiến trúc Phân lớp của Hệ thống**

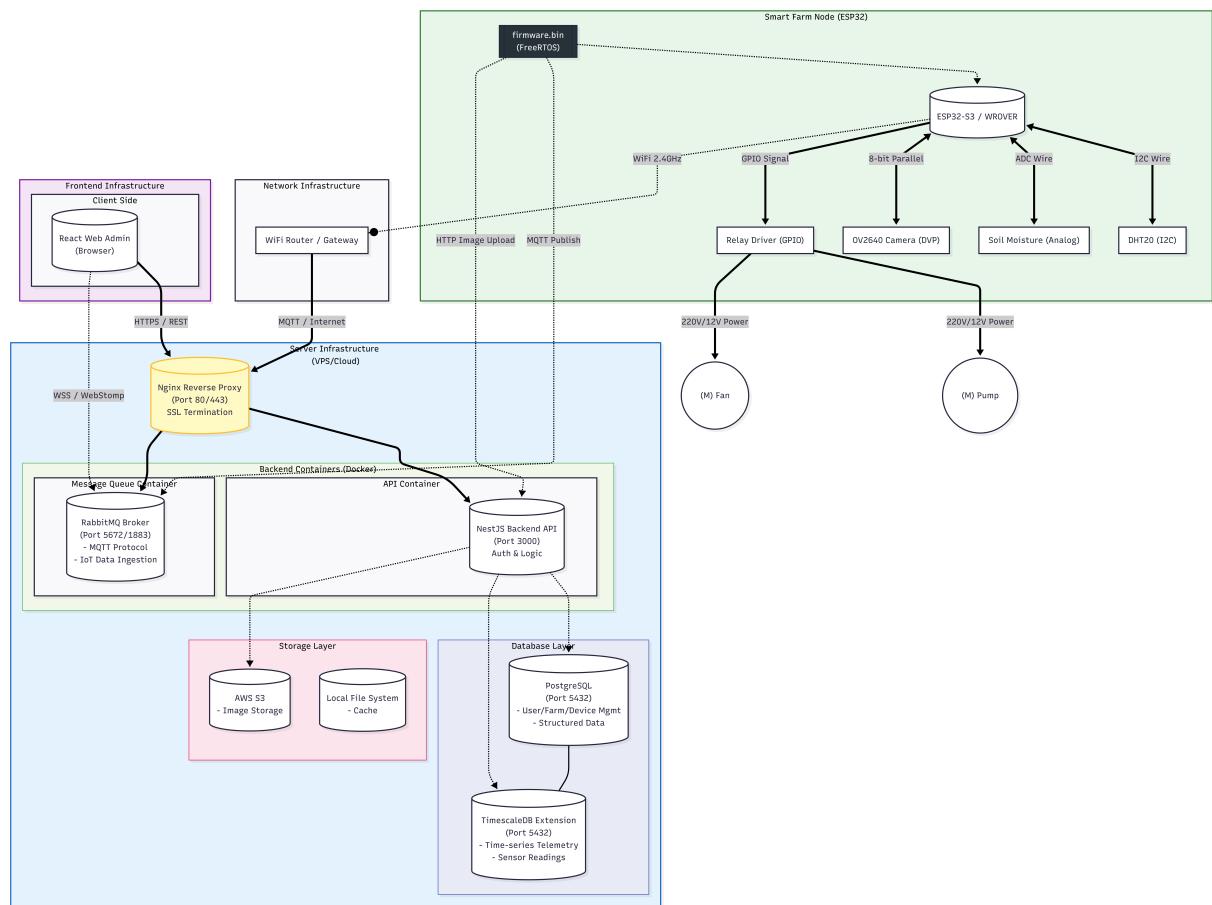
Hệ thống được tổ chức thành 6 tầng chính và 1 module bảo mật xuyên suốt:

- **Presentation Layer (Tầng Giao diện):** Là điểm tiếp xúc với người dùng cuối (System Admin). Tầng này gửi các yêu cầu HTTP đến hệ thống thông qua giao diện Web Admin để thực hiện các tác vụ quản lý.
- **Integration Layer (Tầng Tích hợp):** Dóng vai trò là cổng vào duy nhất (Entry Point) và điều phối luồng dữ liệu, bao gồm 2 thành phần:
  - *API Gateway:* Tiếp nhận và định tuyến các yêu cầu RESTful từ Web Admin xuống các dịch vụ nghiệp vụ tương ứng.
  - *RabbitMQ Message Queue:* Dóng vai trò Broker trung gian, tiếp nhận hàng loạt dữ liệu bất đồng bộ từ các thiết bị IoT (Sensors/ESP32), giúp giảm tải cho hệ thống xử lý chính (Decoupling).
- **Business Layer (Tầng Nghiệp vụ):** Nơi chứa toàn bộ logic xử lý của hệ thống:
  - *User & Farm Management:* Xử lý các logic về quản lý người dùng, nông trại, và cấu hình thiết bị.

- **IoT Data Processor:** Service chuyên biệt (Worker) để tiêu thụ dữ liệu từ RabbitMQ, xử lý tính toán, cảnh báo và chuẩn hóa dữ liệu trước khi lưu trữ.
- **Persistence Layer (Tầng Truy xuất Dữ liệu):** Cung cấp lớp trùu tượng hóa (Abstraction) để giao tiếp với cơ sở dữ liệu (Data Access Component), giúp tách biệt logic nghiệp vụ khỏi các câu lệnh truy vấn SQL cụ thể.
- **Database Layer (Tầng Dữ liệu):** Sử dụng chiến lược lưu trữ đa mô hình (Polyglot Persistence):
  - *PostgreSQL:* Lưu trữ dữ liệu quan hệ có cấu trúc (Users, Farms, Devices).
  - *TimescaleDB:* Cơ sở dữ liệu chuyên dụng cho Time-series để lưu trữ hàng triệu bản ghi nhật ký cảm biến (Telemetries) với hiệu năng cao.
  - *AWS S3:* Lưu trữ các file tĩnh (hình ảnh cây trồng, log files).
- **Physical Layer (Tầng Vật lý):** Bao gồm các thiết bị phần cứng (ESP32, Sensors, Actuators) thu thập dữ liệu môi trường và thực thi các lệnh điều khiển từ server.

**5.3.1.1 Module Bảo mật (Cross-cutting Concern):** Thành phần *Account Control Privileges* bao quát từ tầng Integration xuống tầng Business, đảm bảo mọi yêu cầu đi qua API Gateway đều được xác thực (Authentication) và phân quyền (Authorization) chặt chẽ trước khi được xử lý.

### 5.3.2 Kiến trúc Triển khai (Deployment Architecture)



Hình 11: Deployment Diagram

Sơ đồ triển khai (Hình 11) mô tả cách thức ánh xạ các thành phần phần mềm vào hạ tầng phần cứng vật lý và môi trường mạng. Hệ thống được triển khai theo mô hình tập trung (Centralized Cloud Architecture) kết hợp với các thiết bị biên (Edge Devices).



Các thành phần triển khai chính bao gồm:

- **Field Node (Nút thiết bị tại nông trại):**

- *Vi điều khiển:* Sử dụng ESP32-S3 chạy Firmware dựa trên hệ điều hành thời gian thực FreeRTOS, đảm bảo khả năng xử lý đa nhiệm (đọc cảm biến và gửi mạng song song).
- *Kết nối vật lý:* Các cảm biến (DHT20, Soil Moisture) và camera (OV2640) được kết nối trực tiếp qua các giao tiếp I2C, Analog và DVP. Các thiết bị chấp hành (Máy bơm, Quạt) được điều khiển qua Module Relay.

- **Server Infrastructure (Hệ thống Backend):** Toàn bộ hệ thống Backend được đóng gói và triển khai trên nền tảng Docker, giúp đảm bảo tính nhất quán giữa môi trường phát triển và vận hành (Production). Hệ thống bao gồm các Container chính:

- *Nginx Reverse Proxy:* Đóng vai trò lớp bảo mật đầu tiên, tiếp nhận mọi yêu cầu từ Internet (Port 80/443), chấm dứt SSL (SSL Termination) và điều phối traffic vào các dịch vụ bên trong.
- *Application Container:* Chạy NestJS Backend để xử lý API và logic nghiệp vụ.
- *Message Broker Container:* Chạy RabbitMQ để tiếp nhận dữ liệu IoT qua giao thức MQTT.
- *Database Cluster:* Chạy PostgreSQL tích hợp Extension TimescaleDB, phục vụ lưu trữ dữ liệu lai (Hybrid Storage).

- **Cơ chế Giao tiếp (Communication Protocols):**

- *Device - Server:* Sử dụng giao thức **MQTT** qua WiFi/4G để truyền tải dữ liệu cảm biến (nhiệt, độ ẩm) và giao thức **HTTP** để tải lên hình ảnh (dữ liệu lớn).
- *Client - Server:* Web Admin giao tiếp với hệ thống thông qua **HTTPS** (bảo mật) và **WebSocket** (để nhận cập nhật dữ liệu thời gian thực trên Dashboard).

## 5.4 Đề xuất hiện thực và lựa chọn công nghệ

### 5.4.1 Quản lý lược đồ dữ liệu cảm biến từ xa

- **Mục tiêu:** đảm bảo rằng mọi dữ liệu (telemetry) gửi từ hàng ngàn cảm biến về máy chủ đều:

- **Thống nhất (Consistent):** Dữ liệu luôn tuân theo một định dạng chuẩn.
- **Chất lượng (Valid):** Dữ liệu không bị sai, thiếu hoặc lỗi định dạng.
- **Dễ dàng mở rộng (Extensible):** Dễ dàng thêm cảm biến mới hoặc phiên bản firmware mới mà không làm sập hệ thống.
- **Dễ bảo trì (Maintainable):** Khi schema thay đổi (ví dụ: thêm cảm biến đo độ pH), máy chủ biết cách xử lý phiên bản cũ và mới.

- **Đề xuất cấu trúc:** Sử dụng định dạng JSON vì tính linh hoạt và dễ tiếp cận. Cấu trúc được chia thành 3 schema chính:

- **Schema chung (device\_base):** Tất cả các gói tin (*packet*) đều phải chứa các thông tin cơ bản này để định tuyến và nhận diện.

- \* `device_id(String)`: Mã định danh thiết bị.
  - \* `timestamp(int64)`: Thời gian gửi dữ liệu.
  - \* `schema_id(String)`: Tên của schema mà gói tin này đang sử dụng.
  - \* `schema_version(String)`: Phiên bản của schema.

- **Schema 1 (env\_data):** Dữ liệu môi trường.

```
{  
    "device_id": "env-sensor-zone-a-01",  
    "timestamp": 1678886400000,  
    "schema_id": "env_data",  
}
```



```
"schema_version": "v1.1",
"data": {
    "temperature_celsius": 28.5,
    "humidity_percent": 75.2
}
```

- **Schema 2 (camera\_data):** Dữ liệu hình ảnh từ camera.

```
{
    "device_id": "cam-zone-a-01",
    "timestamp": 1678887000000,
    "schema_id": "camera_event",
    "schema_version": "v1.0",
    "data": {
        "image_url": "https://storage.server.com/images/12345.jpg",
        "file_size": 51200
    }
}
```

- **Schema 3 (device\_health):** Dữ liệu tình trạng thiết bị.

```
{
    "device_id": "cam-zone-a-01",
    "timestamp": 1678886500000,
    "schema_id": "device_health",
    "schema_version": "v1.0",
    "data": {
        "status": "online", // "online", "offline", "error"
        "battery_percent": 85.0, // If battery-powered
        "uptime_seconds": 3600,
        "error_code": 0 // 0 = OK, optional error codes
    }
}
```

- **Đề xuất hệ thống quản lý:** Cách máy chủ biết env\_data v1.0 và v1.1 khác gì nhau:

- **Kho lưu trữ Schema (Schema Registry):**

- \* Là một cơ sở dữ liệu (hoặc một Git repository) chứa các tệp tin định nghĩa schema.
- \* Sẽ có các tệp như là "env\_data\_v1.0.0.json", "camera\_event\_v1.0.0.json",....
- \* Ưu điểm: Cả team phát triển thiết bị (firmware) và team phát triển phần mềm (software) đều nhìn vào đây để làm việc.

- **Quản lý Phiên bản (Versioning):** Sử dụng Semantic Versioning (vMAJOR.MINOR.PATCH).

- \* PATCH (v1.0.1): Sửa lỗi nhỏ, không ảnh hưởng cấu trúc.
- \* MINOR (v1.1.0): Thêm trường dữ liệu mới, vẫn tương thích ngược.
- \* MAJOR (v2.0.0): Thay đổi lớn, không tương thích ngược.

- **Xác thực Schema (Schema Validation)**

- \* Dữ liệu từ cảm biến gửi đến (ví dụ: qua MQTT Broker).
- \* Một dịch vụ "Ingestor" (bộ tiếp nhận) sẽ đọc gói tin.
- \* Nó thấy schema\_id: "env\_data" và schema\_version: "v1.1".



- \* Nó lập tức tra cứu trong Schema Registry để lấy tệp định nghĩa `env_data_v1.1.json`.
- \* Nó dùng tệp định nghĩa này để xác thực (validate) gói tin nhận được.
  - Nếu hợp lệ: Đẩy dữ liệu vào database (ví dụ: TimeScaleDB, InfluxDB).
  - Nếu không hợp lệ: Gói tin bị loại bỏ và gửi cảnh báo (ví dụ: "Thiết bị 'env-sensor-zone-a-01' đang gửi dữ liệu rác!").

- **Ưu điểm của đề xuất này:**

- **Ngăn chặn dữ liệu rác:** Hệ thống tự động loại bỏ dữ liệu sai định dạng ngay từ đầu vào.
- **Gỡ lỗi dễ dàng:** Biết chính xác thiết bị nào đang gửi sai phiên bản schema.
- **Đề dang mở rộng:** Khi muốn thêm cảm biến độ ẩm đất (v1.1), các thiết bị v1.0 cũ vẫn hoạt động bình thường song song với các thiết bị v1.1 mới.
- **Tính độc lập:** Xử lý camera (dữ liệu nặng) riêng biệt với telemetry (dữ liệu nhẹ) giúp hệ thống nhanh và ổn định.

#### 5.4.2 Đề xuất cải tiến cho thuật toán RFE

- **Cốt lõi thuật toán RFE:**

- Phương pháp RFE trong bài báo chủ yếu tập trung vào dữ liệu của một cảm biến đơn lẻ và tránh đặc trưng dựa trên tương quan để giảm chi phí tính toán.
- Tuy nhiên, trong một nhà kính, các cảm biến không hoạt động độc lập. Nhiệt độ ở mọi điểm phải tương quan với nhau.

- **Đề xuất cải tiến:** Đặc trưng tương quan không gian.

- Nếu một cảm biến báo nhiệt độ 50 độ C trong khi 10 cảm biến xung quanh nó báo 25 độ C thì cảm biến đó chắc chắn bị lỗi.
- Ý tưởng:  $\Delta \text{valueER} = \text{value\_A} - \text{avr}(\text{all\_other\_sensors\_value})$  (So một cảm biến với giá trị trung bình của tất cả các cảm biến).
- Khó khăn: khó để phát hiện ngưỡng nào quyết định cảm biến bị lỗi hay cảm biến bình thường

- **Khả năng tích hợp vào dự án:**

- RFE nguyên bản:

- \* **Ưu điểm:**

- Hiệu quả tính toán rất cao: Thuật toán được thiết kế cho các hệ thống IoT năng lượng thấp. Các phép toán rất nhanh và nhẹ. Hoàn toàn phù hợp để chạy trên một gateway tại nhà kính mà không cần phần cứng mạnh.

- Phát hiện lỗi cục bộ tốt: RFE rất phù hợp trong việc phát hiện các lỗi truyền thông của một cảm biến khi nó hoạt động sai so với chính nó như:

1. Giá trị bị kẹt (Stuck): Cảm biến luôn báo 25 độ C. RFE sẽ phát hiện ra vì đặc trưng tốc độ thay đổi và độ biến động sẽ bằng 0.
2. Lỗi đột biến (Spike): Giá trị nhảy vọt bất thường. RFE sẽ phát hiện qua tốc độ thay đổi.
3. Lỗi trôi (Drift): Giá trị từ từ sai lệch. RFE sẽ phát hiện qua các đặc trưng "xu hướng" (EWMA).

- \* **Nhược điểm:**

- RFE nguyên bản chủ động tránh các đặc trưng dựa trên tương quan để giảm chi phí. Do đó, nó không thể trả lời câu hỏi: "Cảm biến này đang báo giá trị có hợp lý so với các cảm biến xung quanh nó không?".



- Nếu một cảm biến nhiệt độ bị lỗi và báo giá trị 20°C (một giá trị hợp lệ) một cách ổn định, trong khi cả nhà kính đang là 35°C, RFE nguyên bản sẽ không phát hiện được lỗi này. Nó chỉ thấy một tín hiệu ổn định và cho là "bình thường".

\* Kết luận:

- Khả năng tích hợp cao.
- Hiệu quả về tài nguyên.
- Giải quyết phần lớn các lỗi phần cứng cơ bản của từng cảm biến riêng lẻ.

– RFE cải tiến với đặc trưng tương quan không gian:

\* Ưu điểm:

- Khắc phục điểm yếu lớn nhất: Bằng cách thêm đặc trưng "Tương quan không gian" (Delta = Giá trị cảm biến - Trung bình của mạng lưới), hệ thống giờ đây đã có "nhận thức về bối cảnh". Nó có thể phát hiện "lỗi logic" (cảm biến báo 20°C trong khi cả nhà kính là 35°C).
- Tăng độ chính xác: Việc bổ sung bối cảnh không gian cung cấp cho mô hình một bức tranh hoàn chỉnh hơn. Điều này sẽ làm tăng đáng kể độ chính xác phát hiện lỗi.
- Phù hợp với nông nghiệp: Mọi trường nhà kính có tính tương quan cao. Phiên bản cải tiến khai thác được cả hai yếu tố này.

\* Nhược điểm:

- Tăng chi phí tính toán: Để tính đặc trưng "tương quan không gian", hệ thống phải thu thập dữ liệu từ nhiều cảm biến rồi mới thực hiện phép tính. Điều này nặng hơn một chút so với RFE gốc.
- Độ phức tạp của luồng dữ liệu tăng lên: Cần một bước gom dữ liệu trước khi chạy kỹ thuật đặc trưng, thay vì xử lý song song từng cảm biến.

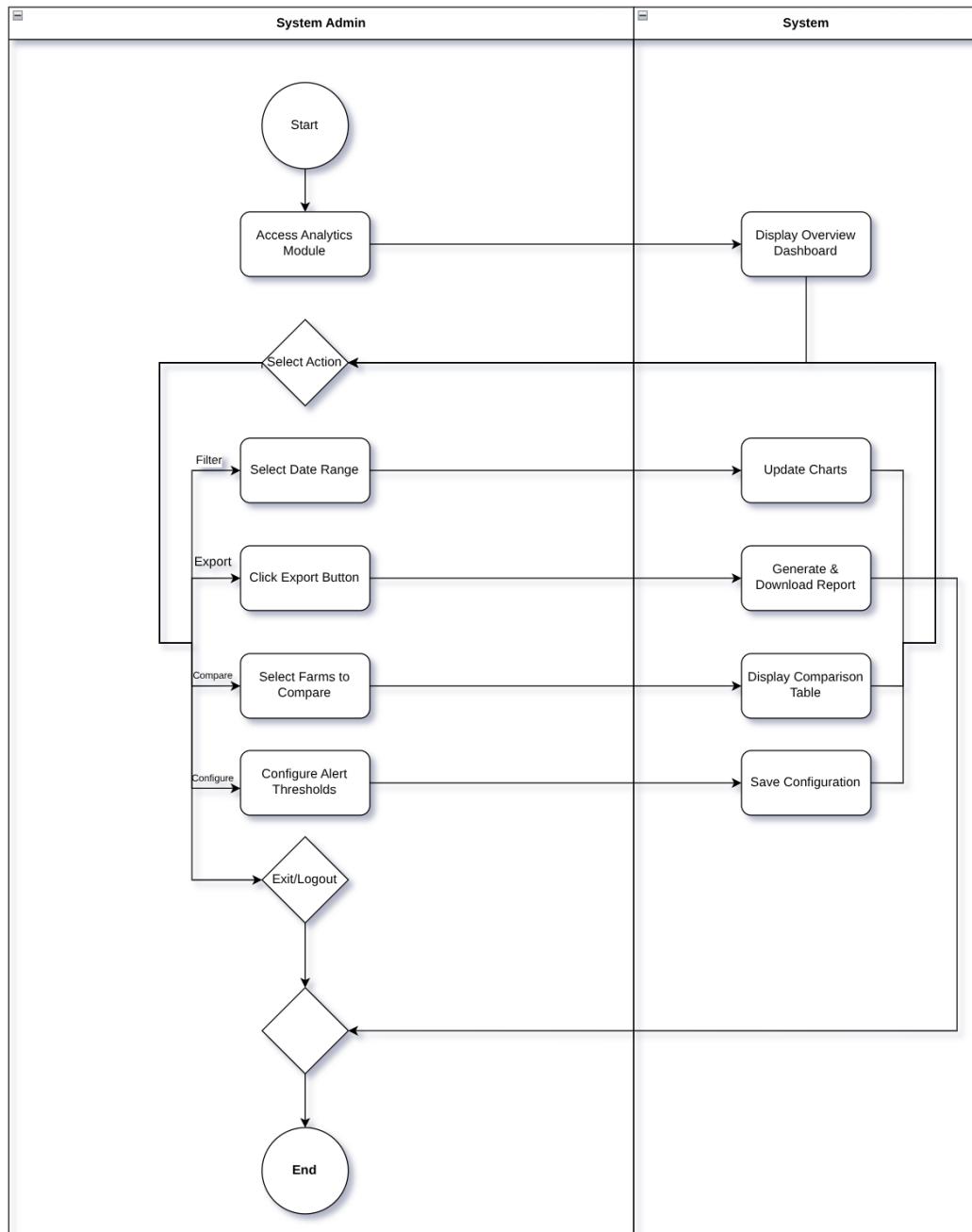
\* Kết luận:

- Khả năng tích hợp cao.
- Tăng nhẹ về chi phí tính toán.
- Độ chính xác và khả năng phát hiện lỗi logic vượt trội.

## 5.5 Thiết kế các thành phần chính

### 5.5.1 Software

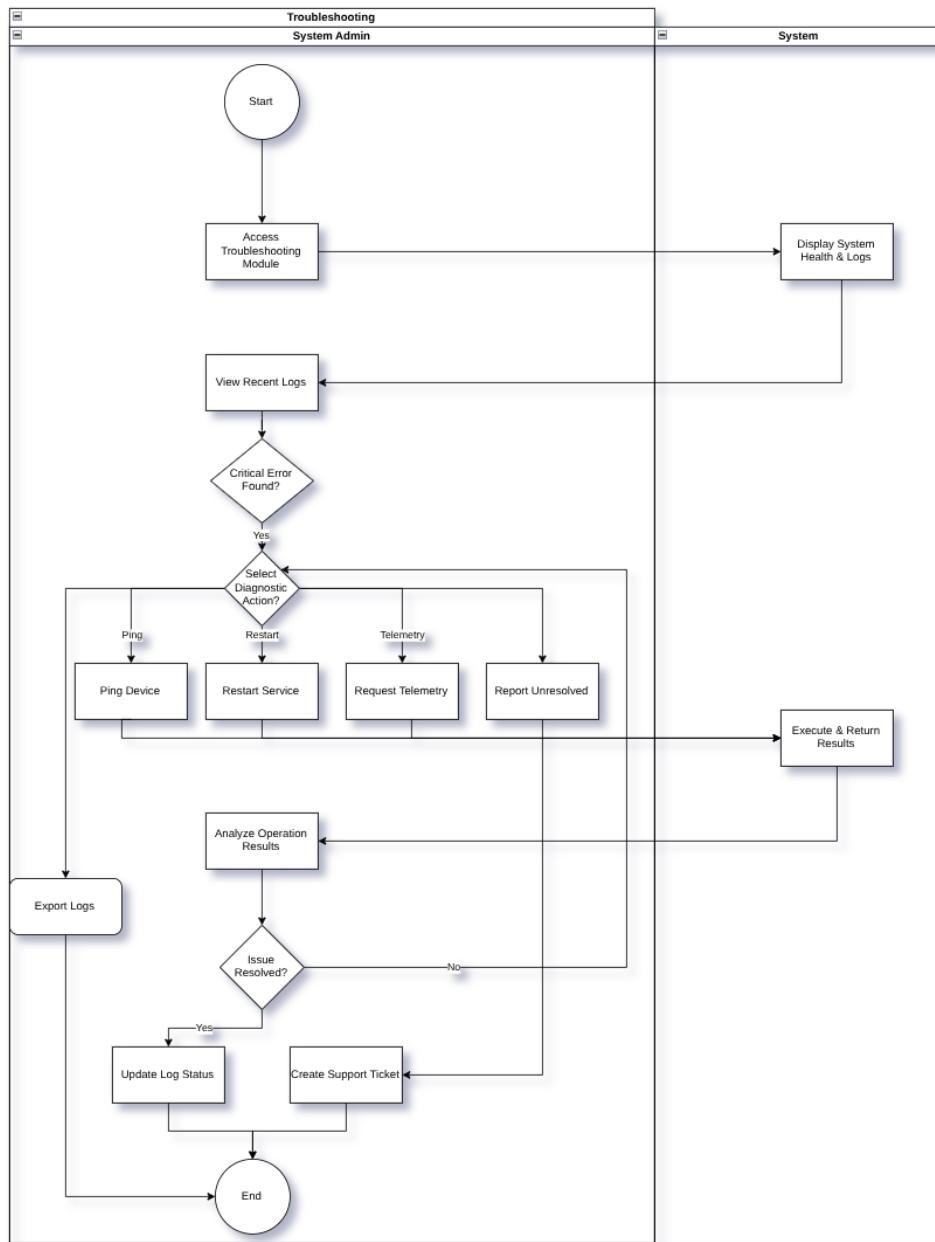
#### 5.5.1.a Quy trình Giám sát Dashboard (Analytics Flow):



Hình 12: Activity Diagram mô tả luồng giám sát và tương tác trên Dashboard

Sơ đồ thể hiện tính tương tác liên tục: Admin có thể thực hiện nhiều tác vụ (Lọc, Xuất, Cấu hình) và quay lại màn hình chính mà không bị ngắt quãng phiên làm việc.

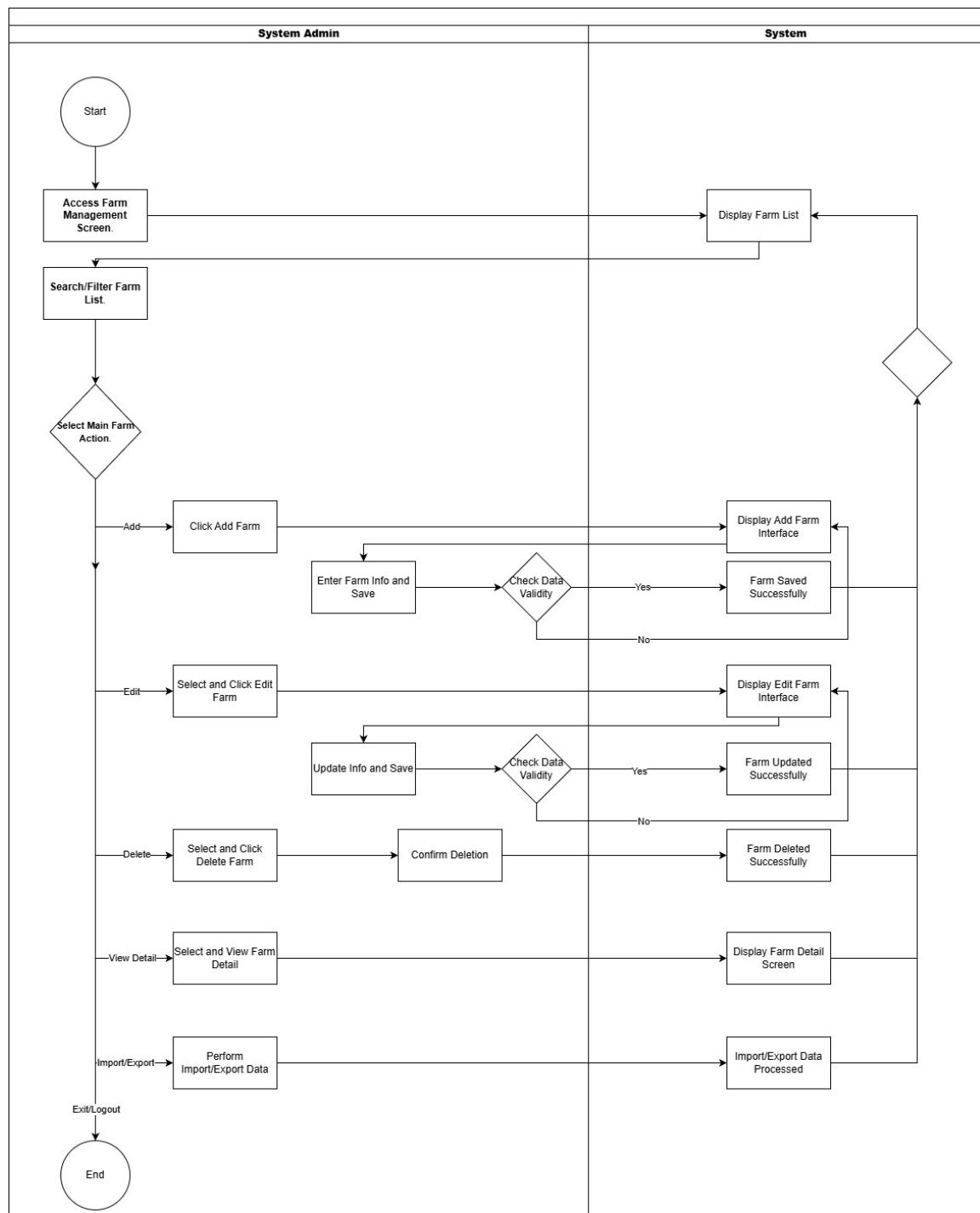
### 5.5.1.b Quy trình Xử lý sự cố (Troubleshoot Flow):



**Hình 13:** Activity Diagram quy trình chẩn đoán và khắc phục sự cố hệ thống

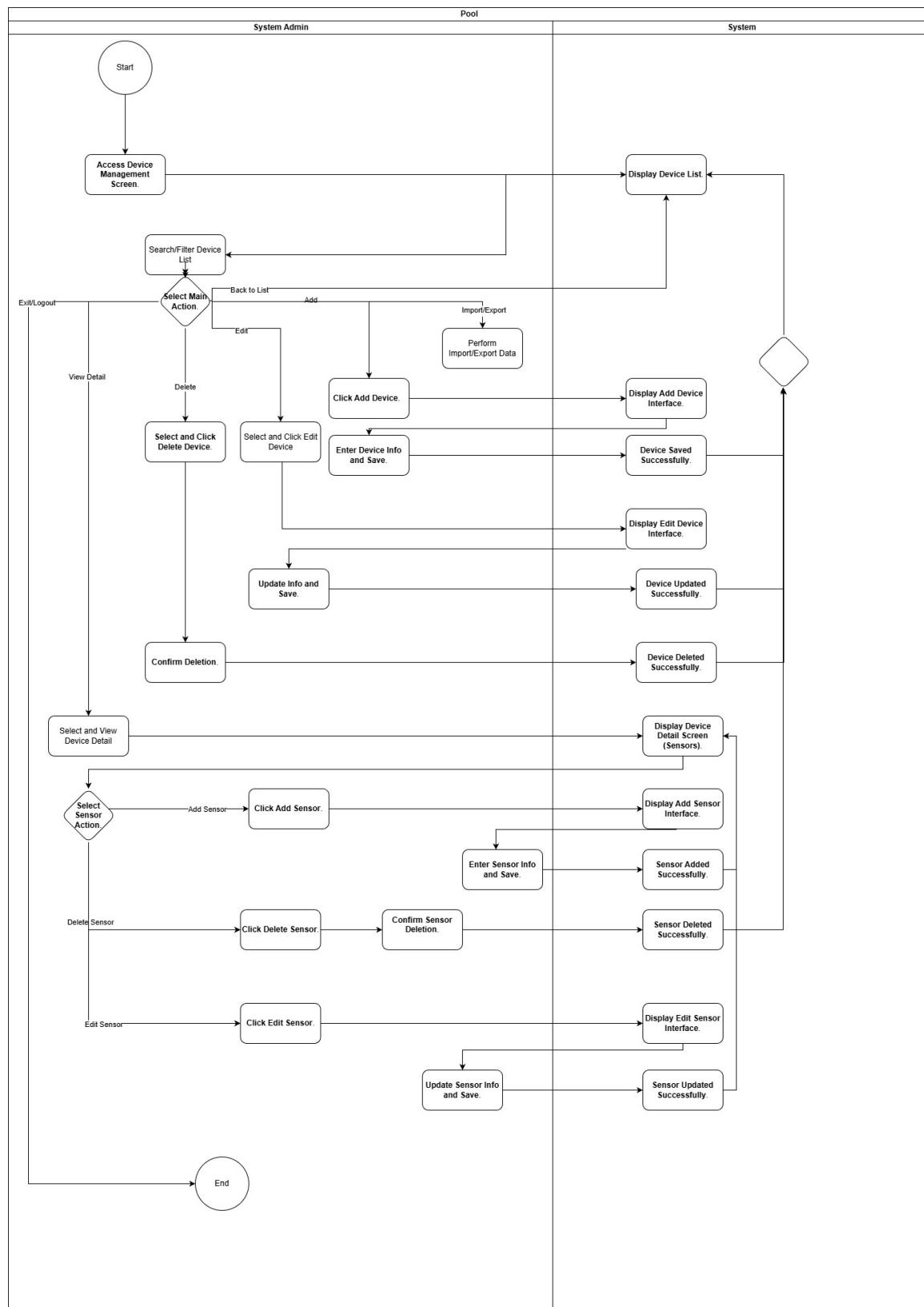
Quy trình đảm bảo mọi sự cố đều có điểm kết thúc rõ ràng: hoặc là được giải quyết (Resolved), hoặc là được chuyển tiếp thành vé hỗ trợ (Support Ticket) để đội ngũ kỹ thuật can thiệp sâu hơn.

### 5.5.1.c Quy trình quản lý Farm:



Hình 14: Activity Diagram mô tả luồng quản lý farm

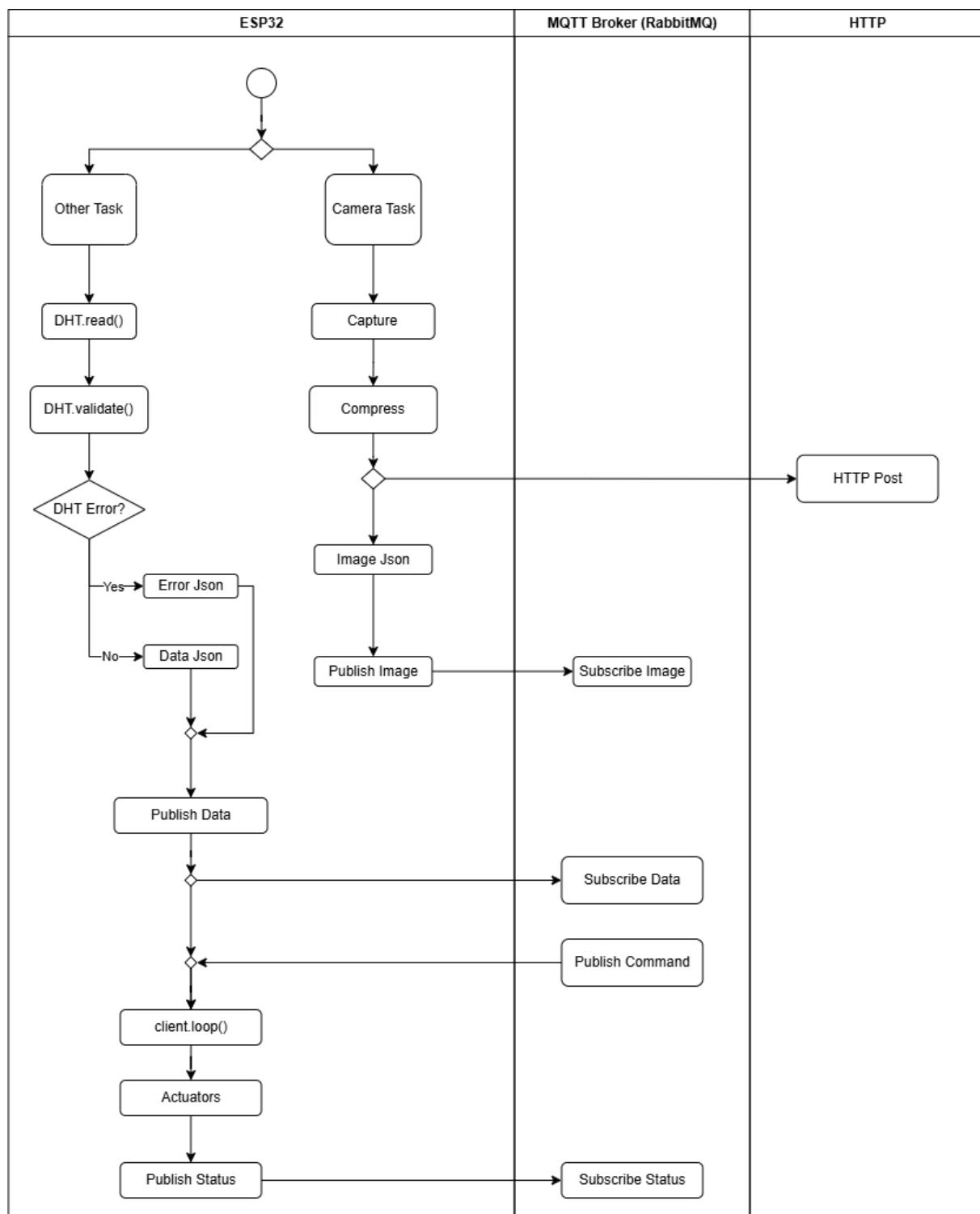
#### 5.5.1.d Quy trình quản lý thiết bị:



Hình 15: Activity Diagram mô tả luồng quản lý thiết bị

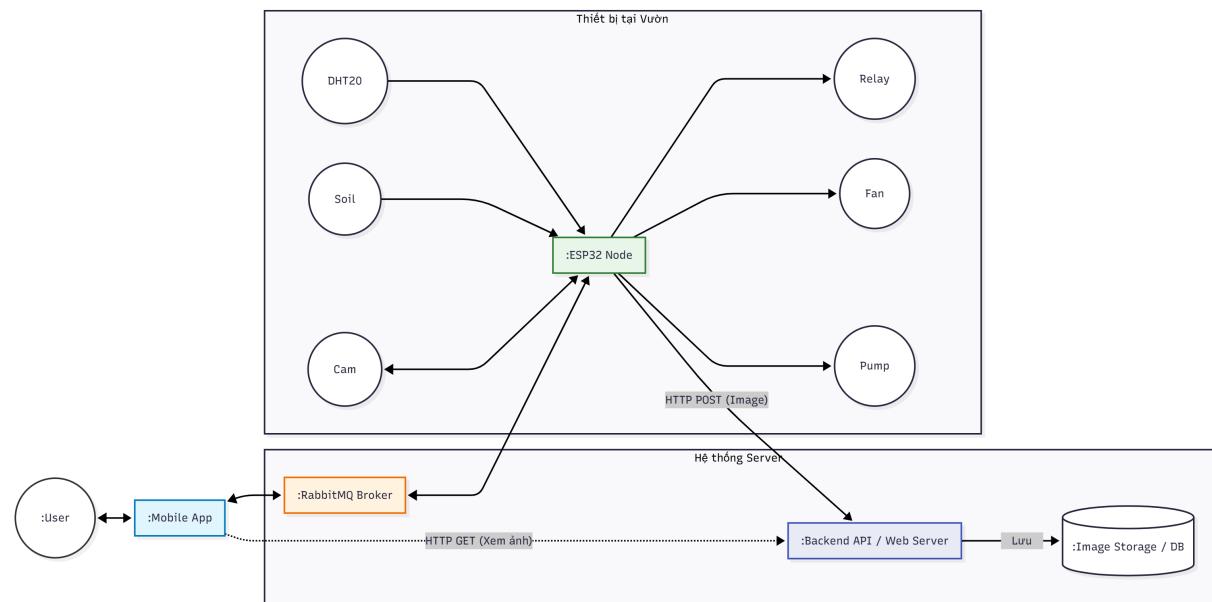
### 5.5.2 Firmware

#### 5.5.2.a Activity Diagram



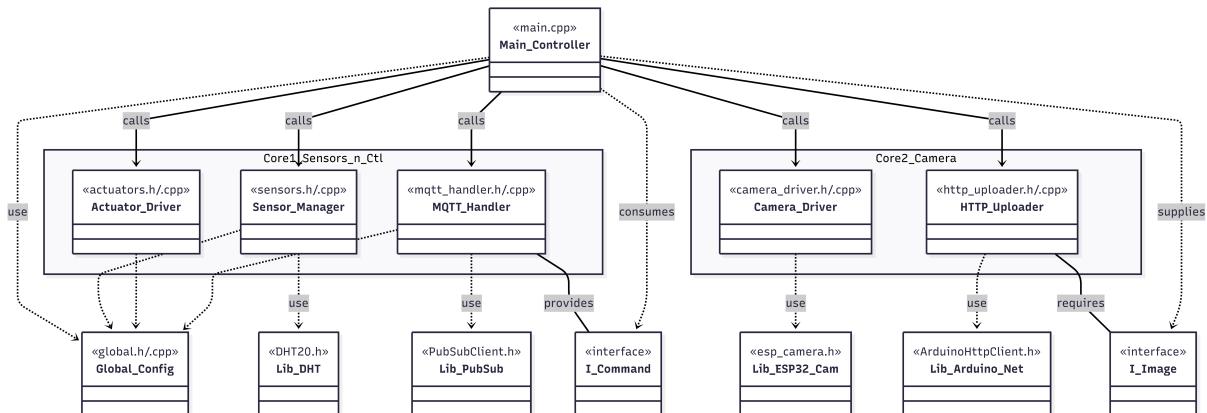
Hình 16: Activity Diagram

### 5.5.2.b Communication Diagram



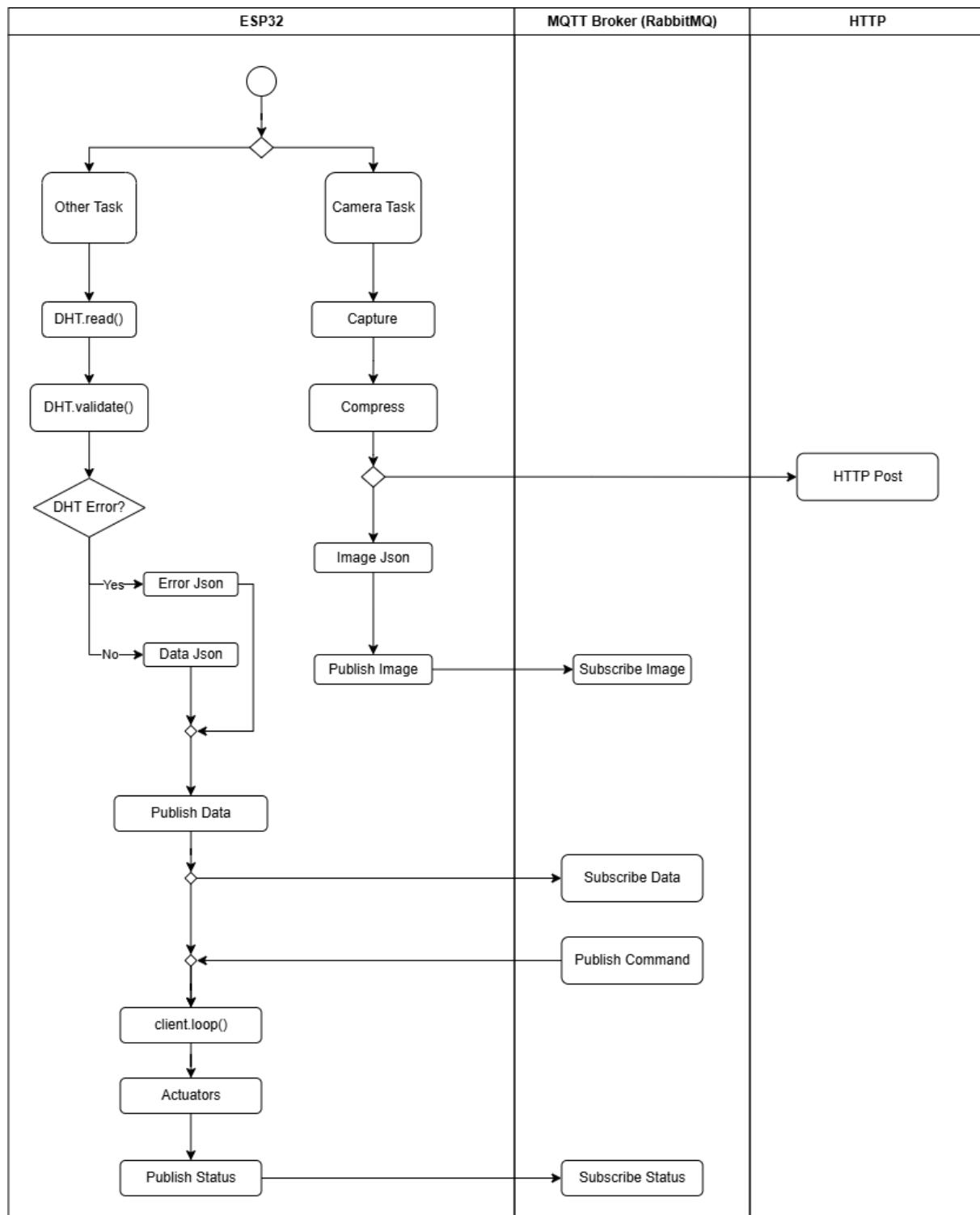
Hình 17: Communication Diagram

### 5.5.2.c Component Diagram



Hình 18: Component Diagram

#### 5.5.2.d Sequence Diagram



Hình 19: Sequence Diagram

### 5.5.2.e Timing Diagram

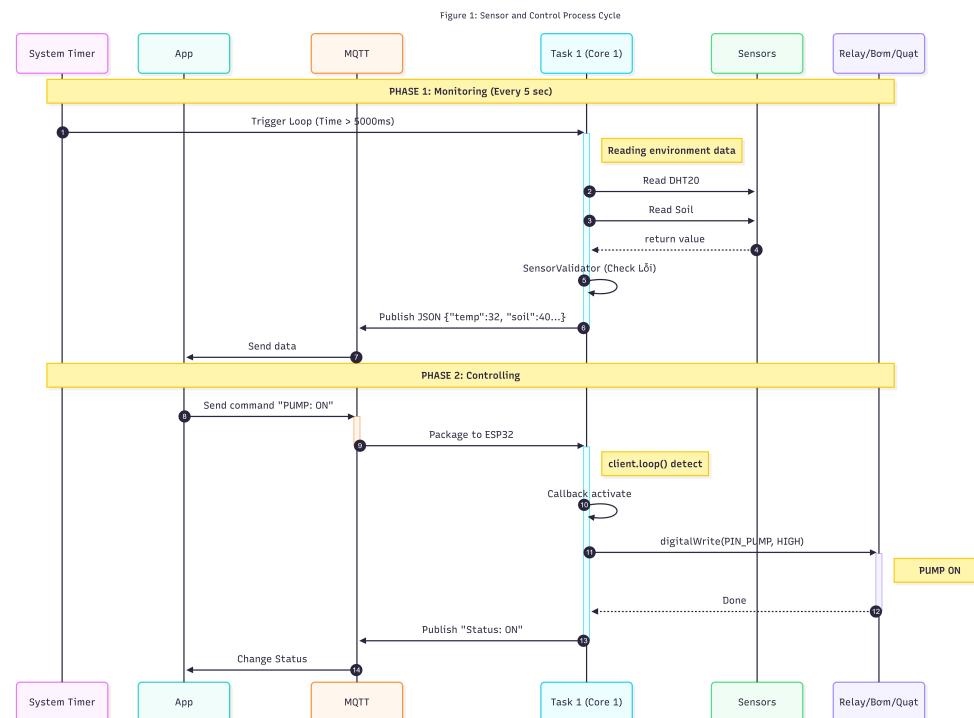
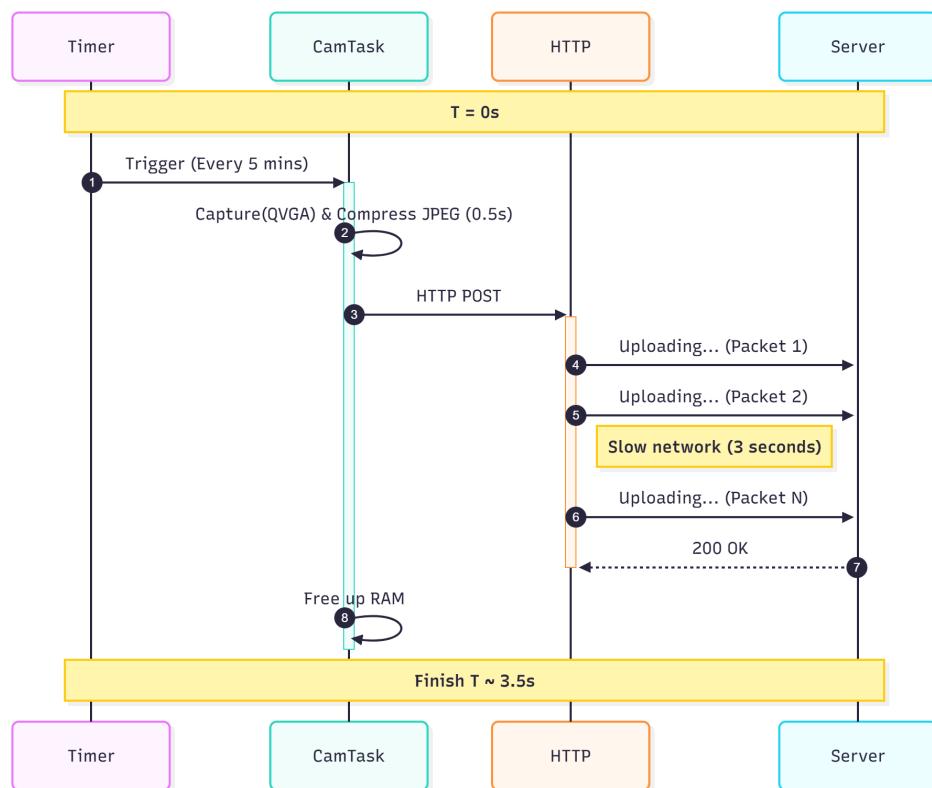
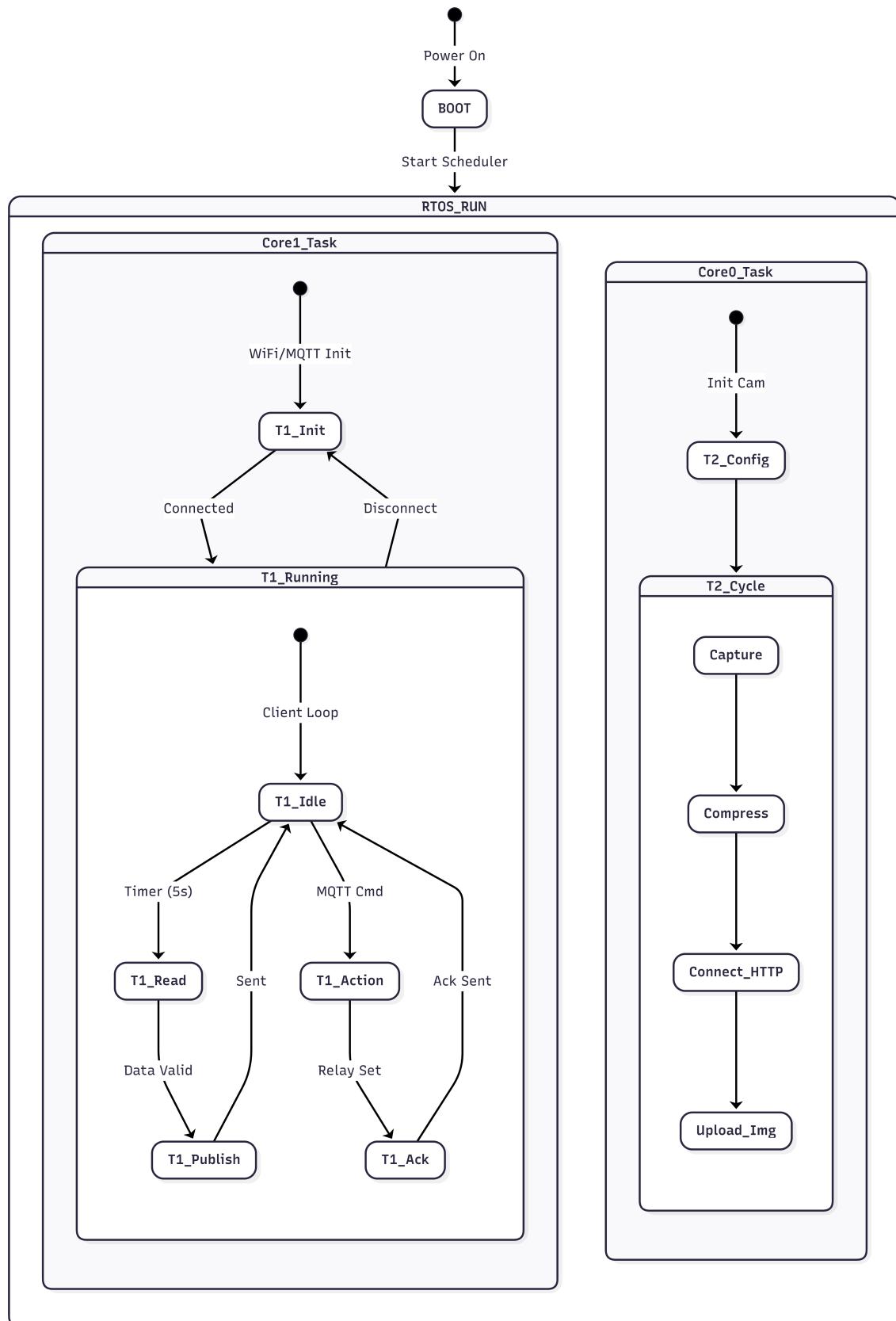


Figure 2: Camera Process Cycle



Hình 20: Timing Diagram

### 5.5.2.f State Machine Diagram



Hình 21: State Machine Diagram



## 5.6 Cách đánh giá giải pháp

Để đảm bảo tính khoa học và thực tiễn của đề tài, nhóm nghiên cứu đề xuất phương pháp đánh giá hệ thống dựa trên 3 trụ cột chính: Hiệu năng hệ thống (System Performance), Độ chính xác của thuật toán (Algorithmic Accuracy) và Trải nghiệm người dùng (User Experience).

### 5.6.1 Đánh giá Hiệu năng Hệ thống (Performance Evaluation)

Phần này tập trung vào khả năng chịu tải và độ ổn định của hạ tầng IoT (RabbitMQ, NestJS, TimescaleDB). Nhóm sẽ sử dụng các công cụ kiểm thử tự động (như JMeter hoặc Script mô phỏng) để giả lập tải.

- Kích bản thử nghiệm:** Giả lập 50 thiết bị ảo gửi dữ liệu liên tục với tần suất 5 giây/gói tin trong thời gian 60 phút.
- Các chỉ số đo lường (Metrics):**
  - Throughput (Thông lượng):* Số lượng bản tin xử lý được trên giây (msg/s).
  - End-to-End Latency:* Thời gian trễ từ khi thiết bị gửi dữ liệu đến khi hiển thị trên Dashboard (Mục tiêu: < 5 giây).
  - Resource Usage:* Mức tiêu thụ CPU và RAM của Server (Docker Containers) khi chịu tải đỉnh.
  - Packet Loss Rate:* Tỷ lệ gói tin bị mất tại Message Broker.

### 5.6.2 Đánh giá Hiệu quả Thuật toán Phát hiện lỗi (Algorithmic Evaluation)

Đối với module *Sensor Fault Detection*, nhóm sẽ đánh giá dựa trên phương pháp "Tiêm lỗi" (Fault Injection) - chủ động tạo ra các dữ liệu sai lệch để kiểm tra khả năng phát hiện của hệ thống.

Bảng 2: Các kịch bản đánh giá thuật toán phát hiện lỗi

STT	Loại lỗi giả lập	Tiêu chí Đạt (Pass Criteria)
1	<b>Mất kết nối (Hard Fault):</b> Ngắt nguồn thiết bị đột ngột.	Hệ thống cập nhật trạng thái "Offline" và gửi cảnh báo trong vòng 3 chu kỳ gửi tin (khoảng 15s).
2	<b>Dữ liệu bất thường (Outlier):</b> Gửi giá trị nhiệt độ đột biến (ví dụ: 100°C).	Hệ thống phát hiện giá trị vượt ngưỡng (Threshold) và ghi nhận cảnh báo.
3	<b>Dữ liệu đóng băng (Stuck):</b> Gửi liên tục một giá trị không đổi trong 30 phút.	Hệ thống phát hiện phương sai (Variance) bằng 0 và cảnh báo lỗi cảm biến bị kẹt.

### 5.6.3 Đánh giá Trải nghiệm và Chức năng (Functional & UX Evaluation)

Đánh giá mức độ hoàn thiện của sản phẩm đối với người dùng cuối (System Admin/Thương lái) thông qua kiểm thử chấp nhận (UAT - User Acceptance Testing).

- Phương pháp:** Thực hiện danh sách kiểm tra (Checklist) các chức năng nghiệp vụ cốt lõi trên giao diện Web Admin.
- Các chỉ số đánh giá:**
  - Tỷ lệ hoàn thành tác vụ (Task Completion Rate):* Người dùng có thực hiện được các thao tác (Thêm Farm, Gán thiết bị, Xem báo cáo) mà không gặp lỗi không?
  - Thời gian phản hồi giao diện (UI Response Time):* Các thao tác chuyển trang, load dữ liệu lịch sử phải hoàn tất dưới 2 giây.
  - Tính trực quan:* Biểu đồ dữ liệu và các cảnh báo lỗi có dễ hiểu và dễ nhận biết hay không.



## 6 Kế hoạch thực hiện

### 6.1 Mục tiêu theo từng giai đoạn

#### 6.1.0.a Firmware

- **Giai đoạn 1:** Nghiên cứu và Chuẩn bị (Tuần 1 - Tuần 4)
  - Nghiên cứu lý thuyết về kiến trúc ESP32 (Dual-core), giao thức Camera DVP, MQTT và FreeRTOS.
  - Lựa chọn và mua sắm linh kiện phần cứng (ESP32-S3/Wrover, OV2640, DHT20, Relay, cảm biến đất).
  - Thiết kế sơ đồ nguyên lý và lắp ráp mạch thử nghiệm trên Breadboard.
- **Giai đoạn 2:** Phát triển và Tích hợp hệ thống (Tuần 5 - Tuần 11)
  - Phần cứng: Hoàn thiện mạch in (PCB) hoặc mạch hàn thủ công chắc chắn.
  - Firmware: Viết chương trình điều khiển trung tâm sử dụng FreeRTOS. Tách tác vụ xử lý ảnh (Core 0) và tác vụ điều khiển/kết nối (Core 1).
  - Kết nối: Xây dựng giao thức truyền nhận JSON qua MQTT và HTTP Streaming.
- **Giai đoạn 3:** Kiểm thử, Tối ưu và Viết báo cáo (Tuần 12 - Tuần 15)
  - Chạy thử nghiệm hệ thống trong môi trường thực tế (vườn mô phỏng).
  - Đo đặc thông số: Độ trễ (latency), độ ổn định WiFi, nhiệt độ chip.
  - Viết báo cáo thuyết minh và chuẩn bị slide bảo vệ.

#### 6.1.0.b Software

- **Giai đoạn 1:** Khởi tạo nền tảng và Cơ sở dữ liệu (Tuần 1 - Tuần 4)
  - Thiết lập môi trường phát triển (Docker, Git).
  - Xây dựng cấu trúc Database: Quan hệ (PostgreSQL) cho quản lý User/Farm và Chuỗi thời gian (TimescaleDB) cho dữ liệu cảm biến.
  - Xây dựng module xác thực (Authentication) và phân quyền (RBAC) làm nền tảng bảo mật.
- **Giai đoạn 2:** Phát triển Backend Core và Tích hợp IoT (Tuần 5 - Tuần 9)
  - Phát triển các API quản lý nghiệp vụ (Farm, Device Management).
  - Xây dựng IoT Data Processor: Dịch vụ tiêu thụ dữ liệu từ RabbitMQ, xử lý Validate Schema (JSON) và lưu trữ vào TimescaleDB.
  - Triển khai cơ chế cảnh báo thời gian thực (Real-time Alert).
- **Giai đoạn 3:** Phát triển Frontend và Dashboard (Tuần 10 - Tuần 13)
  - Xây dựng giao diện người dùng (Web Admin).
  - Trực quan hóa dữ liệu: Biểu đồ nhiệt độ/độ ẩm, thống kê trạng thái thiết bị.
  - Tích hợp module Báo cáo (Report) và Xử lý sự cố (Troubleshoot).
- **Giai đoạn 4:** Kiểm thử, Tối ưu và Triển khai (Tuần 14 - Tuần 15)
  - Kiểm thử tải (Load Testing) với hàng nghìn request giả lập.
  - Tối ưu truy vấn Database (Indexing, Compression).
  - Đóng gói (Containerization) và viết tài liệu hướng dẫn sử dụng.



## 6.2 Lịch trình và mốc thời gian

### 6.2.0.a Firmware

Bảng 3: Bảng phân bổ lịch trình thực hiện đồ án 15 tuần

Tuần	Công việc chi tiết	Kết quả bàn giao
1-2	<ul style="list-style-type: none"><li>Nghiên cứu Datasheet ESP32, OV2640.</li><li>Tìm hiểu thư viện ArduinoJson, PubSubClient.</li><li>Đặt mua linh kiện phần cứng.</li></ul>	<ul style="list-style-type: none"><li>Chương 1 (Cơ sở lý thuyết).</li><li>Danh sách linh kiện đầy đủ.</li></ul>
3	<ul style="list-style-type: none"><li>Thiết kế sơ đồ khối và sơ đồ nguyên lý (Schematic).</li><li>Kiểm tra hoạt động riêng lẻ của module cảm biến, Relay.</li></ul>	<ul style="list-style-type: none"><li>Bản vẽ sơ đồ nguyên lý.</li><li>Code kiểm thử đơn giản.</li></ul>
4	<ul style="list-style-type: none"><li>Lắp ráp phần cứng hoàn chỉnh (trên mạch in hoặc đúc lõi).</li><li>Kiểm tra nguồn, đo đặc chống nhiễu.</li></ul>	<ul style="list-style-type: none"><li>Mô hình phần cứng thô.</li><li>Chương 2 (Thiết kế phần cứng).</li></ul>
5-6	<ul style="list-style-type: none"><li><b>Cấu hình FreeRTOS:</b> Tạo Task, Queue, Semaphore.</li><li>Lập trình đọc cảm biến (DHT20, Đất) và điều khiển Relay.</li></ul>	<ul style="list-style-type: none"><li>Code khung sườn (Skeleton code).</li><li>Dữ liệu hiển thị trên Serial Monitor.</li></ul>
7-8	<ul style="list-style-type: none"><li><b>Xử lý Camera (Trọng tâm):</b> Cấu hình giao thức DVP, lấy dữ liệu ảnh JPEG.</li><li>Lập trình Web Server để stream ảnh qua HTTP.</li></ul>	<ul style="list-style-type: none"><li>Hình ảnh hiển thị trên trình duyệt.</li><li>Frame rate đạt mức ổn định (&gt;10fps).</li></ul>
9	<ul style="list-style-type: none"><li><b>Kết nối IoT:</b> Lập trình giao thức MQTT.</li><li>Đóng gói dữ liệu dạng JSON.</li></ul>	<ul style="list-style-type: none"><li>Gửi thành công JSON lên Broker.</li><li>Nhận lệnh điều khiển từ xa.</li></ul>
10	<ul style="list-style-type: none"><li>Tích hợp hệ thống: Ghép module Camera + Cảm biến + MQTT.</li><li>Xử lý xung đột tài nguyên giữa 2 nhân (Core 0/1).</li></ul>	<ul style="list-style-type: none"><li>Firmware hoàn chỉnh v1.0.</li><li>Chương 3 (Thiết kế phần mềm).</li></ul>
11	<ul style="list-style-type: none"><li>Xây dựng Dashboard điều khiển (Web app hoặc Mobile app đơn giản).</li></ul>	<ul style="list-style-type: none"><li>Giao diện người dùng (UI) hoạt động.</li></ul>
12	<ul style="list-style-type: none"><li>Chạy thử nghiệm hệ thống liên tục (Stress test).</li><li>Tinh chỉnh ngưỡng cảm biến, tối ưu bộ nhớ.</li></ul>	<ul style="list-style-type: none"><li>Số liệu thực nghiệm.</li><li>Bảng đánh giá độ ổn định.</li></ul>

Tiếp tục ở trang sau...



Bảng 3 – tiếp theo từ trang trước

Tuần	Công việc chi tiết	Kết quả bàn giao
13	<ul style="list-style-type: none"><li>Tổng hợp số liệu, vẽ biểu đồ kết quả.</li><li>Viết chương Kết quả và Thảo luận.</li></ul>	<ul style="list-style-type: none"><li>Chương 4 (Kết quả thực nghiệm).</li></ul>
14	<ul style="list-style-type: none"><li>Hoàn thiện toàn bộ báo cáo thuyết minh.</li><li>Chỉnh sửa định dạng, trích dẫn tài liệu tham khảo.</li></ul>	<ul style="list-style-type: none"><li>Bản thảo báo cáo hoàn chỉnh.</li></ul>
15	<ul style="list-style-type: none"><li>Soạn thảo Slide thuyết trình.</li><li>Quay video demo và luyện tập bảo vệ.</li></ul>	<ul style="list-style-type: none"><li>Slide Powerpoint.</li><li>Video Demo sản phẩm.</li></ul>



### 6.2.0.b Software

Tuần	Công việc chi tiết	Kết quả bàn giao
1-2	<b>Khởi tạo nền tảng:</b> - Thiết kế ERD vật lý cho PostgreSQL và TimescaleDB. - Cài đặt Docker (Postgres, RabbitMQ). - Khởi tạo cấu trúc dự án NestJS (Monorepo).	- File docker-compose.yml. - Script khởi tạo Database.
3-4	<b>Quản lý người dùng (Auth):</b> - Phát triển API Đăng ký, Đăng nhập (JWT). - Xây dựng cơ chế phân quyền RBAC (Admin/User).	- API Documentation (Swagger). - Sơ đồ phân quyền hoạt động.
5-6	<b>Quản lý Farm &amp; Device:</b> - API CRUD Nông trại, Khu vực. - API Đăng ký thiết bị và gán vào Farm.	- Các API quản lý chính hoàn thiện. - Dữ liệu lưu trữ thành công vào Postgres.
7-8	<b>Tích hợp IoT (Core):</b> - Cấu hình RabbitMQ Consumer trong NestJS. - Xử lý Validate JSON Schema đầu vào. - Lưu trữ dữ liệu cảm biến vào TimescaleDB.	- Dữ liệu từ Queue được lưu vào bảng Hypertable. - Log được các gói tin lõi định dạng.
9	<b>Cảnh báo thời gian thực:</b> - Xây dựng logic so sánh ngưỡng (Threshold). - Tích hợp Socket.io để đẩy thông báo lên Web.	- Cảnh báo hiển thị ngay lập tức khi vượt ngưỡng.
10	<b>Frontend - Cơ bản:</b> - Dụng khung Web Admin (React). - Giao diện Login, Quản lý Farm/Device.	- Giao diện quản lý cơ bản hoạt động.
11-12	<b>Frontend - Nâng cao:</b> - Vẽ biểu đồ (Charts) nhiệt độ/độ ẩm. - Trang Troubleshoot và Xuất báo cáo.	- Dashboard hiển thị trực quan. - Chức năng Export PDF/Excel.
13	<b>Tích hợp hệ thống:</b> - Ghép nối ESP32 (thực/ảo) - Backend - Web. - Kiểm thử luồng dữ liệu E2E.	- Hệ thống hoạt động thông suốt từ cảm biến lên Web.
14	<b>Tối ưu hóa:</b> - Indexing và Materialized Views cho DB. - Cấu hình nén dữ liệu TimescaleDB.	- Báo cáo hiệu năng hệ thống (Response time).
15	<b>Đóng gói &amp; Báo cáo:</b> - Deploy lên Server thực tế. - Viết tài liệu hướng dẫn sử dụng.	- Link Demo sản phẩm. - Slide thuyết trình.



### 6.3 Rủi ro và phương án giảm thiểu

#### 6.3.0.a Firmware

Bảng 5: Bảng phân tích rủi ro và phương án giảm thiểu

STT	Rủi ro (Risk)	Mức độ	Phương án giảm thiểu
1	<b>Hỏng hóc phần cứng:</b> Cháy ESP32 hoặc Camera do đấu sai nguồn/ngắn mạch.	Cao	<ul style="list-style-type: none"><li>Mua dự phòng 01 bộ linh kiện.</li><li>Kiểm tra kĩ mạch bằng VOM trước khi cấp nguồn.</li></ul>
2	<b>Tràn bộ nhớ (Stack Overflow):</b> Do xử lý ảnh JPEG lớn trên FreeRTOS.	Cao	<ul style="list-style-type: none"><li>Sử dụng hàm: uxTaskGetStackHighWaterMark để giám sát RAM.</li><li>Cân nhắc sử dụng ESP32 có PSRAM.</li></ul>
3	<b>Nhiều tín hiệu:</b> Camera bị sọc hoặc cảm biến đất báo giá trị ảo.	Trung bình	<ul style="list-style-type: none"><li>Sử dụng tụ lọc nguồn 100uF và 100nF.</li><li>Đi dây tín hiệu ngắn gọn, tách biệt với dây động cơ.</li></ul>
4	<b>Độ trễ truyền ảnh cao:</b> Video bị giật, lag khi mạng WiFi yếu.	Trung bình	<ul style="list-style-type: none"><li>Giảm độ phân giải ảnh (VGA/QVGA) khi mạng kém.</li><li>Tách luồng gửi ảnh sang Core 0 độc lập.</li></ul>
5	<b>Chậm tiến độ:</b> Do vướng mắc thuật toán phức tạp.	Thấp	<ul style="list-style-type: none"><li>Ưu tiên xử lý phần Camera trước (phần khó nhất).</li><li>Tham khảo cộng đồng Open Source ESP32.</li></ul>



### 6.3.0.b Software

Bảng 6: Phân tích rủi ro phần mềm và phương án xử lý

STT	Rủi ro (Risk)	Mức độ	Phương án giảm thiểu
1	<b>Nghẽn cỗ chai Database:</b> Khi hàng trăm thiết bị gửi dữ liệu cùng lúc, việc Insert từng dòng gây quá tải.	Cao	<ul style="list-style-type: none"><li>Sử dụng kỹ thuật Batch Insert (gom 100-500 bản ghi/lần).</li><li>Tận dụng kiến trúc Hypertables của TimescaleDB để phân mảnh dữ liệu.</li></ul>
2	<b>Mất toàn vẹn dữ liệu (Data Inconsistency):</b> Lỗi xảy ra giữa chừng khi tạo Nông trại và gán Thiết bị.	Cao	<ul style="list-style-type: none"><li>Sử dụng Database Transaction (ACID) để đảm bảo "All or Nothing".</li><li>Cơ chế Rollback tự động khi có lỗi.</li></ul>
3	<b>Gián đoạn thời gian thực:</b> Kết nối WebSocket bị ngắt khiến Dashboard không cập nhật số liệu.	Trung bình	<ul style="list-style-type: none"><li>Cơ chế Heartbeat (Ping/Pong) để phát hiện mất kết nối.</li><li>Tự động kết nối lại (Auto-reconnect) ở phía Frontend React.</li></ul>
4	<b>Mất dữ liệu tại Broker:</b> RabbitMQ bị đầy hàng đợi (Queue overflow) nếu Backend xử lý chậm.	Trung bình	<ul style="list-style-type: none"><li>Cấu hình Message Acknowledgment (chỉ xóa tin khi đã xử lý xong).</li><li>Tăng số lượng Worker (Consumer) trong NestJS để xử lý song song.</li></ul>
5	<b>Dữ liệu rác (Invalid Schema):</b> Thiết bị gửi sai định dạng JSON làm lỗi parser hệ thống.	Cao	<ul style="list-style-type: none"><li>Triển khai lớp Validation (DTO) chặt chẽ tại đầu vào (Gateway).</li><li>Ghi log gói tin lỗi ra bảng riêng (Dead Letter Queue) để debug.</li></ul>
6	<b>Sự cố Vận hành (Deployment):</b> Docker Container bị crash do lỗi bộ nhớ hoặc lỗi runtime.	Cao	<ul style="list-style-type: none"><li>Cấu hình Docker Restart Policy (<code>restart: always</code>).</li><li>Thiết lập Health Check định kỳ cho các service.</li></ul>



## 7 Kết luận

### 7.1 Tóm tắt vấn đề và hướng tiếp cận

Dề tài "Hệ thống quản lý nông trại thông minh" được phát triển nhằm giải quyết bài toán thực tế về sự phân mảnh trong quản lý chuỗi nông trại của các thương lái và nhà đầu tư nông nghiệp. Hướng tiếp cận chủ đạo của nhóm là xây dựng một nền tảng quản lý tập trung (All-in-one), tích hợp khả năng giám sát dữ liệu thời gian thực từ thiết bị đa nguồn, đồng thời đơn giản hóa trải nghiệm vận hành cho đối tượng người dùng không chuyên về kỹ thuật.

### 7.2 Công việc đã hoàn thành trong giai đoạn Đồ án Chuyên ngành

Trong khuôn khổ giai đoạn thiết kế, nhóm đã hoàn thành các nhiệm vụ nền tảng sau:

- Khảo sát và Phân tích:** Đã thực hiện khảo sát các giải pháp hiện có, từ đó xác định rõ yêu cầu bài toán gồm các chức năng quản lý cốt lõi và các ràng buộc phi chức năng về hiệu năng, độ trễ.
- Thiết kế Kiến trúc tổng thể:** Đã đề xuất mô hình kiến trúc phân lớp (Layered Architecture) hoàn chỉnh, kết hợp sức mạnh của **NestJS** (Backend), **RabbitMQ** (Message Queuing) và **TimescaleDB** (Time-series Data). Đây là nền tảng vững chắc để đảm bảo khả năng mở rộng (Scalability) và tính ổn định khi số lượng thiết bị gia tăng.
- Thiết kế Chi tiết Module IoT:** Đã hoàn thiện thiết kế Firmware trên **ESP32** với cơ chế đa luồng (FreeRTOS) và phân tích kỹ lưỡng các rủi ro phần cứng (tràn bộ nhớ, nhiễu tín hiệu) cùng phương án giảm thiểu.
- Lập kế hoạch hiện thực:** Đã xây dựng lộ trình chi tiết cho giai đoạn Đồ án Tốt nghiệp, bao gồm các mốc thời gian kiểm thử và triển khai cụ thể cho từng thành viên.

### 7.3 Định hướng thực hiện Đồ án Tốt nghiệp

Giai đoạn tiếp theo sẽ tập trung vào việc hiện thực hóa bản thiết kế thành sản phẩm chạy thực tế:

#### 1. Hiện thực hóa Firmware & Phần cứng:

- Lập trình Firmware ESP32 sử dụng FreeRTOS, tích hợp đọc dữ liệu song song từ đa cảm biến (DHT20, độ ẩm đất) và Camera OV2640.
- Cài đặt và tinh chỉnh thuật toán phát hiện lỗi cảm biến (sử dụng phương pháp thống kê hoặc RFE) để tự động nhận diện các bất thường như trôi số liệu (Drift) hoặc mất tín hiệu.
- Tối ưu hóa giao thức MQTT để giảm thiểu độ trễ truyền tin.

#### 2. Xây dựng Hệ thống Backend & Frontend:

- Phát triển bộ RESTful API và các Worker xử lý dữ liệu IoT chuyên biệt thông qua RabbitMQ.
- Xây dựng Web Dashboard với ReactJS, tập trung vào tính năng trực quan hóa dữ liệu và hệ thống cảnh báo thời gian thực (Real-time Alerting).

#### 3. Kiểm thử và Tối ưu hóa:

- Thực hiện Load Testing với mạng lưới thiết bị mô phỏng quy mô vừa và lớn để đánh giá giới hạn chịu tải của hệ thống.
- Triển khai hệ thống tại mô hình thực nghiệm (Pilot Test) để thu thập dữ liệu thật, từ đó đánh giá độ chính xác của thuật toán phát hiện lỗi và độ ổn định của phần cứng.

#### 4. Các chỉ số kỹ thuật mục tiêu (Target Performance Metrics):

Dể đảm bảo tính định lượng cho việc đánh giá kết quả vào cuối kỳ, nhóm thiết lập các chỉ số KPI mục tiêu như sau:



Tiêu chí	Mô tả	Mục tiêu (Target)
<b>Độ trễ (Latency)</b>	Thời gian từ khi thiết bị gửi dữ liệu đến khi hiển thị trên Dashboard.	< 3 giây (với mạng 4G/WiFi tiêu chuẩn).
<b>Khả năng chịu tải</b>	Số lượng thiết bị gửi dữ liệu đồng thời mà không gây mất gói tin.	50 thiết bị (giả lập) với tần suất 5 giây/gói tin.
<b>Độ chính xác AI</b>	Tỷ lệ phát hiện đúng các lỗi cảm biến cơ bản (mất nguồn, gai dữ liệu).	> 85% trên tập dữ liệu kiểm thử.
<b>Tốc độ phản hồi Web</b>	Thời gian tải trang và hiển thị biểu đồ lịch sử.	< 2 giây (cho truy vấn 7 ngày gần nhất).
<b>Độ ổn định</b>	Thời gian hoạt động liên tục không lỗi (Uptime) trong môi trường thử nghiệm.	99% (trong 48 giờ chạy test liên tục).

## 7.4 Bài học kinh nghiệm và Góc nhìn phản tư

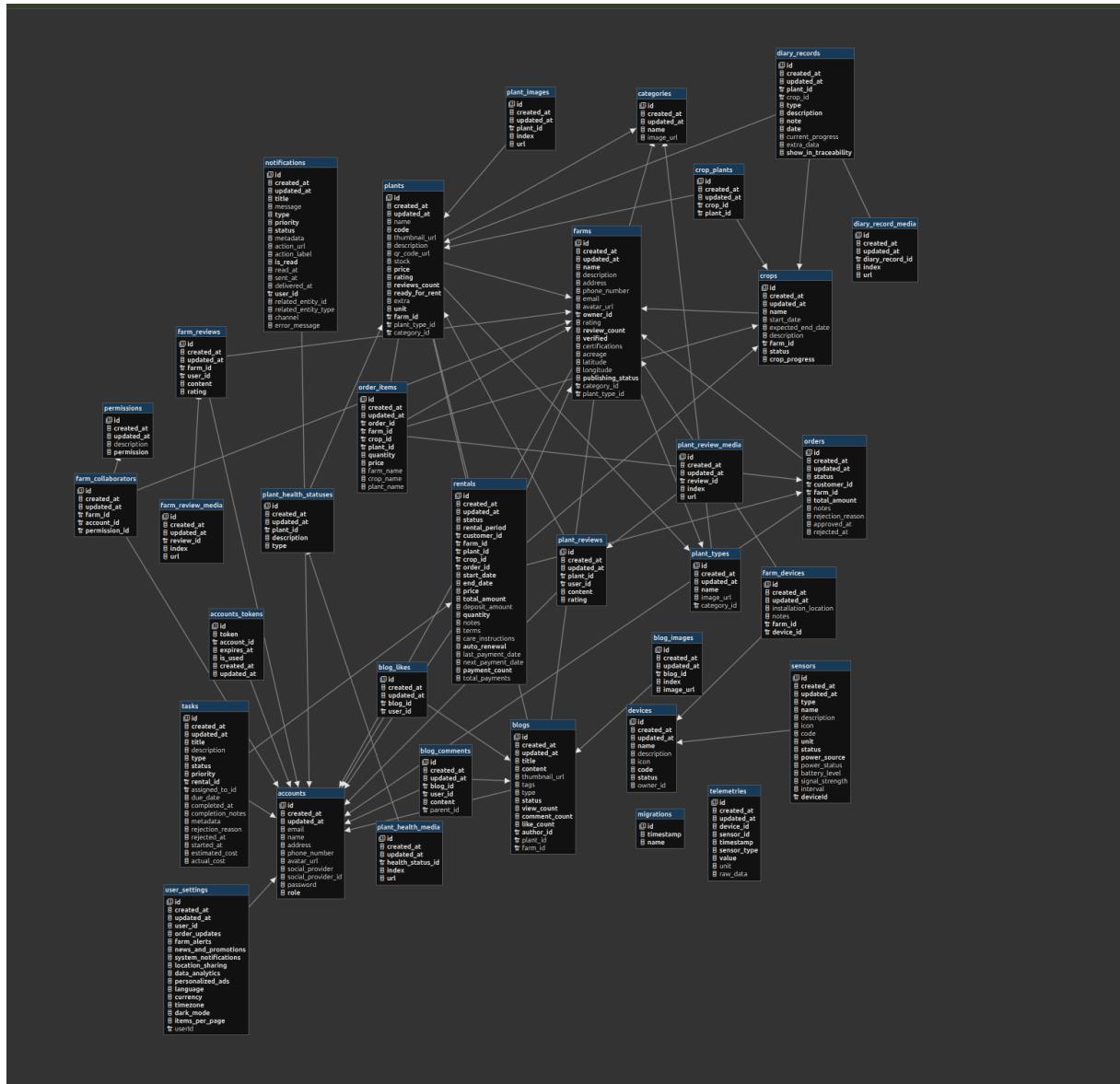
Qua quá trình nghiên cứu, nhóm nhận thức sâu sắc rằng thách thức lớn nhất của dự án IoT không chỉ nằm ở việc lập trình, mà ở việc \*\*thiết kế kiến trúc hệ thống\*\* sao cho linh hoạt và chịu lỗi tốt ngay từ đầu.

Nhóm đã rút ra bài học quan trọng về việc sử dụng cơ chế hàng đợi (Message Queue) để tách rời các thành phần hệ thống (Decoupling), giúp tránh nghẽn cổ chai khi lưu lượng dữ liệu tăng cao. Ngoài ra, việc phân tích rủi ro phần cứng ngay từ giai đoạn thiết kế sẽ giúp giảm thiểu đáng kể thời gian Debug và chi phí phát sinh khi triển khai thực tế.

Dự án này là bước đệm quan trọng, không chỉ mang giá trị học thuật mà còn hướng tới một sản phẩm có tính ứng dụng cao, góp phần giải quyết bài toán chuyển đổi số trong nông nghiệp một cách thiết thực.

## Phụ lục A Sơ đồ Cơ sở dữ liệu vật lý (Physical ERD)

Hình dưới đây mô tả chi tiết toàn bộ lược đồ cơ sở dữ liệu của hệ thống, bao gồm các bảng, các trường dữ liệu, kiểu dữ liệu và mối quan hệ giữa chúng.



Hình 22: Sơ đồ thực thể quan hệ chi tiết (Physical Data Model)

## Phụ lục B Danh sách các API chính (API Specifications)

Phần này cung cấp tài liệu kỹ thuật chi tiết về các API Endpoint đã được thiết kế cho hệ thống Backend, được trích xuất trực tiếp từ giao diện Swagger UI.

Các hình ảnh dưới đây mô tả rõ ràng phương thức HTTP (GET, POST, PUT, PATCH, DELETE), đường dẫn (URI) và mô tả chức năng ngắn gọn cho từng Endpoint hai module quan trọng nhất là Quản lý Nông trại (Farms) và Quản lý Thiết bị (Devices).



The screenshot shows the Swagger UI for the 'Farms' endpoint. It lists the following API operations:

- GET /api/v1/farms/{id}/plants
- POST /api/v1/farms/{id}/plants
- PATCH /api/v1/farms/{id}/plants/{plant\_id}
- GET /api/v1/farms/{id}/plants/{plant\_id}
- GET /api/v1/farms/{id}
- PATCH /api/v1/farms/{id}
- DELETE /api/v1/farms/{id}
- GET /api/v1/farms/{id}/devices
- GET /api/v1/farms
- POST /api/v1/farms
- POST /api/v1/farms/{id}/certifications
- GET /api/v1/farms/{id}/plants/{plant\_id}/diaries
- POST /api/v1/farms/{id}/plants/{plant\_id}/diaries
- POST /api/v1/farms/{id}/reviews

Hình 23: *Detailed description of the API module for Farm Management (Farms Endpoint) [Source: Swagger UI System]*

The screenshot shows the Swagger UI for the 'Devices' endpoint. It lists the following API operations:

- PUT /api/v1/notifications/{id}/status
- Devices
  - GET /api/v1/devices
  - POST /api/v1/devices
  - GET /api/v1/devices/{id}
  - PATCH /api/v1/devices/{id}
  - PUT /api/v1/devices/{id}
  - DELETE /api/v1/devices/{id}
  - POST /api/v1/devices/assign-to-farm
  - DELETE /api/v1/devices/{deviceId}/farms/{farmId}
  - GET /api/v1/devices/admin/all
  - GET /api/v1/devices/admin/{id}
  - PATCH /api/v1/devices/admin/{id}
  - PUT /api/v1/devices/admin/{id}
  - DELETE /api/v1/devices/admin/{id}
  - POST /api/v1/devices/admin

Hình 24: *Detailed description of the API module for Device Management (Devices Endpoint) [Source: Swagger UI System]*

Nhìn vào các hình trên, có thể thấy hệ thống đã thiết kế đầy đủ các thao tác CRUD (Tạo, Đọc, Cập nhật, Xóa) cho Nông trại và Thiết bị, cũng như các API nghiệp vụ đặc thù như gán thiết bị vào nông trại ('/assign-to-farm') hay lấy lịch sử dữ liệu cảm biến.

## Phụ lục C Danh sách linh kiện phần cứng (Bill of Materials)

Bảng liệt kê các linh kiện được sử dụng để xây dựng Node IoT:



Bảng 7: Bill of Materials (BOM)

STT	Tên linh kiện	Thông số kỹ thuật chính	Số lượng
1	ESP32-S3 WROOM	Dual-core 240MHz, WiFi/BLE	01
2	Cảm biến DHT20	Giao tiếp I2C, đo nhiệt độ/độ ẩm	01
3	Camera OV2640	2 Megapixel, giao tiếp DVP	01
4	Cảm biến độ ẩm đất	Capacitive (Điện dung), Analog	01
5	Relay Module	5VDC - 1 kênh, Opto cách ly	02
6	Module quạt	3.3VDC	01
7	Nguồn Adapter	5VDC	01

## Phụ lục D Bảng so sánh các giải pháp tham khảo

## References

- [1] A. D. Boursianis et al., ?Smart farming: Internet of Things technologies, challenges and future directions,? *Sensors*, vol. 22, no. 10, p. 3735, 2022.
- [2] M. Noura, M. Atiquzzaman, and M. Gaedke, ?Interoperability in IoT: Challenges and solutions,? In *Proceedings of the International Conference on IoT*, Springer, 2019, pp. 1–6.
- [3] Báo Chính Phủ, ?Thủ tướng: Ngành nông nghiệp phải tăng tốc, bứt phá, xuất khẩu 70 tỷ USD trong năm 2025,? *Cổng Thông tin điện tử Chính phủ*, 2024.
- [4] Vietnam Report, *Báo cáo tăng trưởng và 6 thách thức của nông nghiệp công nghệ cao Việt Nam*, 2024.
- [5] J. Smith and T. Nguyen, ?Intelligent Fault Detection in IoT-Based WSNs for Precision Agriculture Using Machine Learning,? *IEEE Internet of Things Journal*, vol. 12, no. 4, pp. 1023–1035, 2025.
- [6] P. Stapleton et al., ?A review of low-cost precision agriculture: sensors, monitoring and communication systems,? *Journal of Agricultural Science*, vol. 12, no. 3, pp. 55–70, 2023.
- [7] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, ?Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications,? *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.
- [8] NestJS Team, *NestJS Documentation: Introduction and Fundamentals*, 2024.
- [9] Microsoft, *TypeScript: Typed JavaScript at Any Scale*, 2024.
- [10] OASIS Open, *MQTT Version 5.0 Committee Specification*, 2019.
- [11] R. Fielding et al., ?Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing,? IETF, Request for Comments 7230, 2014.
- [12] The PostgreSQL Global Development Group, *PostgreSQL 16.0 Documentation*, 2024.
- [13] Timescale Inc., *TimescaleDB Documentation: Hypertables and Chunking*, Truy cập: 13/12/2025, 2024.
- [14] A. Bader et al., ?Time Series Databases: New Ways to Store and Access Data,? *Proceedings of the VLDB Endowment*, 2017.
- [15] Espressif Systems, *ESP32 Series Datasheet*, 2024.
- [16] N. Semiconductors, *I2C Bus Specification and User Manual*, 2021.
- [17] I. OmniVision Technologies, *OV2640 Camera Module Datasheet*, 2020.
- [18] E. Systems, *ESP32 Technical Reference Manual*, 2016.
- [19] FreeRTOS, *FreeRTOS Scheduler and Scheduling Policies*, 2022.
- [20] E. International, *The JSON Data Interchange Syntax (ECMA-404)*, 2017.
- [21] T. N. M. Duy, *Improving Feature Extraction for Sensor Fault Detection in Low-Power IoT Systems*, 2025.