

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG



BÁO CÁO ĐỒ ÁN CHUYÊN NGÀNH



Hệ thống quản lý tập trung thiết bị IoT công nghiệp
(The centralized management system
for Industrial IoT device)

Giảng viên hướng dẫn: ThS. Nguyễn Khánh Thuật

Môn học: NT114.P11 – Đồ Án Chuyên Ngành

21521017

Trần Đình Khôi

21520935

Phan Quốc Huy

TP. Hồ Chí Minh, tháng 12 năm 2024

LỜI CẢM ƠN

Lời đầu tiên, nhóm chúng em xin cảm ơn chân thành các Thầy/Cô khoa Mạng máy tính và Truyền thông dữ liệu đã truyền đạt những kiến thức quý báu trong suốt những năm học vừa qua.

Trong quá trình nghiên cứu, chúng em xin gửi lời cảm ơn chân thành đến Thầy Nguyễn Khánh Thuật, người đã định hướng và giúp đỡ, đóng góp ý kiến trong suốt quá trình lên ý tưởng và thực hiện đồ án này.

Ngoài ra, nhóm cũng xin cảm ơn các bạn/anh/chị tại phòng Lab E3.1 đã hỗ trợ và luôn tạo điều kiện tốt nhất cho nhóm có thể sử dụng cơ sở vật chất phục vụ cho khóa luận trong suốt thời gian vừa qua.

Một lần nữa chúng em xin chân thành cảm ơn!

TÓM TẮT NỘI DUNG

Đồ án trình bày về xây dựng một hệ thống theo dõi, quản lí tập trung các dữ liệu trên màn hình HMI của máy dệt tại Công Ty TNHH Sản Xuất Thương Mại Thái Anh – Tamatra, để quản lý các thông số như nhiệt độ các vùng của máy, thông số của sản phẩm được tạo ra.

ĐÓNG GÓP CÁ NHÂN

MSSV	Họ và tên	Đóng góp
21521017	Trần Đình Khôi	<ul style="list-style-type: none">• Lên ý tưởng cho hệ thống• Xây dựng hệ thống phần cứng• Lập trình mã nguồn cho các node và gateway• Xây dựng chương trình ứng dụng Winform• Viết báo cáo
21520935	Phan Quốc Huy	<ul style="list-style-type: none">• Xây dựng trang web quản lý dữ liệu• Viết chương trình xử lý hình ảnh• Viết báo cáo

Mục lục

LỜI CẢM ƠN	2
TÓM TẮT NỘI DUNG	3
ĐÓNG GÓP CÁ NHÂN	3
BẢNG HÌNH MINH HỌA	7
Chương 1. PHẦN MỞ ĐẦU	10
1.1. Lý do chọn đề tài	10
1.2. Timeline thực hiện đề tài	11
1.3. Mục tiêu nghiên cứu	12
Chương 2. LÝ THUYẾT TỔNG QUAN	13
2.1. Thiết bị	13
2.1.1. Raspberry Pi 4 Model B	13
2.1.2. Camera ELP 2MP	14
2.1.3. ESP32-Camera & Module camera OV2640	14
2.1.4. ESP32-WROOM32	16
2.1.5. Module NRF24L01	17
2.2. Công nghệ sử dụng	18
2.2.1. Truyền thông 2.4GHz	18

2.2.2.	MQTT	18
2.2.3.	Python và ứng dụng của Python.....	20
Chương 3. HIỆN THỰC ĐỀ TÀI		21
3.1.	Mô hình, phân tích kịch bản.....	21
3.1.1.	Node.....	22
3.1.6.1.	Node 1	22
3.1.6.2.	Node 2	24
3.1.2.	Gateway.....	26
3.1.3.	MQTT Broker	28
3.1.4.	Ứng dụng winform.....	33
3.1.5.	Mongo DB	38
3.1.6.	Server	42
3.1.6.1.	Chương trình xử lý hình ảnh	42
3.1.6.2.	Web hiển thị dữ liệu	47
3.1.6.3.	Chương trình xóa hình ảnh của Database	55
Chương 4. ĐÁNH GIÁ, HƯỚNG PHÁT TRIỂN		57
4.1.	Đánh giá.....	57
4.2.	Hướng phát triển	58
Chương 5. KẾT LUẬN		59
5.1.	Tóm tắt kết quả	59
5.2.	Đóng góp của đề tài.....	59

5.3. Nhận định và học hỏi	59
TÀI LIỆU THAM KHẢO	60

BẢNG HÌNH MINH HỌA

Hình 2. 1. Raspberry Pi4 model B	13
Hình 2. 2. Camera ELP 2MP.....	14
Hình 2. 3. ESP32-Camera và module camera OV2640	15
Hình 2. 4. ESP32-WROOM32	16
Hình 2. 5. Module NRF24L01	17
Hình 2. 6. Giới thiệu MQTT	19
Hình 3. 1. Mô hình hệ thống	21
Hình 3. 2. Thành phần Raspberry Pi 4 của Node 1	22
Hình 3. 3. Camera của Node 1 và màn hình máy dẹt	22
Hình 3. 4. Logic hoạt động của node 1	23
Hình 3. 5. Hình ảnh thực tế của node 2.....	24
Hình 3. 6. Logic hoạt động của node 2	25
Hình 3. 7. Hình ảnh mặt trước của gateway.....	26
Hình 3. 8. Hình ảnh mặt sau của gateway.....	27
Hình 3. 9. Logic hoạt động của gateway.....	27
Hình 3. 10. Mã giả mô tả cách hoạt động của hàm kết nối MongoDB.....	29
Hình 3. 11. Mã giả mô tả cách hoạt động của hàm on_connect	30
Hình 3. 12. Mã giả mô tả cách hoạt động của hàm on_message	31
Hình 3. 13. Mã giả mô tả cách hoạt động của hàm cấu hình mqtt client và kết nối	32
Hình 3. 14. Quy trình xử lý của chương trình.....	32

Hình 3. 15. Màn hình chính của ứng dụng.....	33
Hình 3. 16. Thiết lập kết nối đến Mongo DB.....	34
Hình 3. 17. Hộp thoại MongoConfigForm	34
Hình 3. 18. Nhập ID của camera cần cấu hình	35
Hình 3. 19. Màn hình ứng dụng khi tải ảnh thành công	36
Hình 3. 20. Thông báo khi ID nhập vào không có trong cơ sở dữ liệu.....	36
Hình 3. 21. Chọn vùng đọc và đặt tên.....	37
Hình 3. 22. Pop up thông báo thêm vùng chọn thành công	37
Hình 3. 23. Màn hình ứng dụng sau khi thêm vùng chọn thành công	38
Hình 3. 24. Server và Database lưu trữ dữ liệu của hệ thống	38
Hình 3. 25. Các hình ảnh được lưu trữ trong collection IMAGE	39
Hình 3. 26. Thông tin camera được lưu trữ trong bảng CAMERA_CONFIGURATION	40
Hình 3. 27. Các dữ liệu được lưu trữ trong bảng IMAGE_TEXT_DATA	41
Hình 3. 28. Mã giả mô tả cách hoạt động của hàm kiểm tra hình ảnh đã được xử lý ...	43
Hình 3. 29. Mã giả mô tả cách hoạt động của hàm đánh dấu ảnh đã được xử lý	43
Hình 3. 30. Mã giả mô tả cách hoạt động của hàm lưu kết quả OCR và MongoDB.....	44
Hình 3. 31. Mã giả mô tả cách hoạt động của hàm lấy ảnh mới nhất chưa được xử lý.	44
Hình 3. 32. Mã giả mô tả cách hoạt động của hàm phát hiện văn bản trong vùng cấu hình.....	45
Hình 3. 33. Mã giả mô tả cách hoạt động của hàm xử lý hình ảnh liên tục.....	46
Hình 3. 34. Giao diện chính của trang web quản lý dữ liệu.....	47

Hình 3. 35. Giao diện xem thông tin các vùng đọc của camera.....	48
Hình 3. 36. Giao diện xem dữ liệu từ hình ảnh của camera.....	50
Hình 3. 37. Bảng ReadValue hiển thị dữ liệu sau khi mapping với tên vùng chọn tạo bởi ứng dụng winform.....	52
Hình 3. 38. Bảng và biểu đồ Key hiển thị các dữ liệu mới nhất của một Key qua các ảnh mới nhất.....	54
Hình 3. 39. Dung lượng của các bảng trên database.....	56
Hình 3. 40. Mã giả của chương trình xóa ảnh trên database.....	56

Chương 1. PHẦN MỞ ĐẦU

1.1. Lý do chọn đề tài

Trong bối cảnh ngành công nghiệp sản xuất đang không ngừng đổi mới và hướng đến số hóa, việc tối ưu hóa quy trình quản lý và theo dõi dữ liệu từ các hệ thống sản xuất đóng vai trò then chốt. Công Ty TNHH Sản Xuất Thương Mại Thái Anh – Tamatra là một doanh nghiệp sản xuất dệt may, nơi các máy dệt hoạt động liên tục và tạo ra lượng dữ liệu lớn về các thông số vận hành như nhiệt độ, tốc độ, và chất lượng sản phẩm.

Hiện nay, việc giám sát và quản lý dữ liệu này vẫn còn thực hiện rời rạc, thiếu sự kết nối tập trung, dẫn đến nhiều khó khăn như:

- Khó khăn trong việc theo dõi và kiểm soát thông số vận hành: Các thông số quan trọng như nhiệt độ vùng máy, tốc độ máy dệt, hoặc chất lượng sản phẩm chưa được giám sát hiệu quả. Điều này có thể dẫn đến sai sót trong quá trình sản xuất, giảm chất lượng sản phẩm và tăng chi phí vận hành.
- Khả năng truy xuất dữ liệu hạn chế: Khi cần phân tích dữ liệu lịch sử để tìm nguyên nhân lỗi hoặc tối ưu hóa quy trình, việc thiếu một hệ thống lưu trữ tập trung khiến việc này trở nên phức tạp và mất nhiều thời gian.
- Chưa tối ưu hóa được quá trình ra quyết định: Với dữ liệu phân tán, các nhà quản lý gặp khó khăn trong việc đưa ra quyết định nhanh chóng và chính xác, đặc biệt khi cần ứng phó với các sự cố hoặc điều chỉnh quy trình sản xuất.
- **Đặc biệt là do hợp đồng của doanh nghiệp với nhà cung cấp thiết bị có một số điều khoản về việc không được can thiệp vào hệ thống của máy dệt.**

Với các thách thức trên, việc xây dựng một hệ thống theo dõi và quản lý tập trung dữ liệu từ màn hình HMI của máy dệt có thể cấu hình từ xa sẽ mang lại nhiều lợi ích thực tiễn:

- Tăng cường khả năng giám sát: Hệ thống giúp theo dõi liên tục các thông số quan trọng, cảnh báo khi có bất thường, từ đó giảm thiểu rủi ro và tổn thất trong sản xuất.
- Tự động hóa và số hóa dữ liệu: Dữ liệu được thu thập, lưu trữ, và phân tích tập trung, giúp tăng độ chính xác và giảm thiểu công sức quản lý thủ công.
- Hỗ trợ ra quyết định: Dữ liệu tập trung và được phân tích sẽ hỗ trợ ban quản lý trong việc đưa ra các quyết định kịp thời, tối ưu hóa quy trình sản xuất và nâng cao hiệu quả hoạt động.
- Hướng đến xu hướng công nghiệp 4.0: Đề tài không chỉ giải quyết bài toán cụ thể của công ty mà còn giúp doanh nghiệp bắt kịp xu hướng hiện đại hóa và số hóa trong ngành công nghiệp sản xuất.

Do đó, việc lựa chọn đề tài này không chỉ mang ý nghĩa thực tiễn đối với Công Ty TNHH Sản Xuất Thương Mại Thái Anh – Tamatra mà còn góp phần đưa kiến thức về IOT đã được học ứng dụng vào trong lĩnh vực dệt may nói riêng và ngành công nghiệp sản xuất nói chung.

1.2. Timeline thực hiện đề tài

Đề tài được thực hiện thông qua 5 giai đoạn chính với các mốc thời gian cụ thể sau:

Bảng 1. 1. Bảng timeline của đồ án

Giai đoạn	Công việc	Thời gian
1	Lên ý tưởng và chuẩn bị kiến thức	15/09/2024 – 20/09/2024
2	Phân tích thiết kế	21/09/2024 – 10/10/2024
3	Phát triển hệ thống	11/10/2024 – 5/12/2024

4	Kiểm thử	6/12/2024 – 10/12/2024
5	Hoàn Thành	11/12/2024 – 25/12/2024

1.3. Mục tiêu nghiên cứu

Với đề tài “Hệ thống quản lý tập trung thiết bị IoT công nghiệp”, mục tiêu chính là xây dựng một giải pháp hiệu quả và linh hoạt để quản lý tập trung các thiết bị theo dõi dữ liệu từ màn hình HMI trong môi trường công nghiệp.

Hệ thống sẽ tận dụng các thiết bị phần cứng hiện đại như Raspberry Pi, ESP32-Camera, và ESP32 để đảm bảo khả năng thu thập, xử lý, và truyền tải dữ liệu một cách chính xác và liên tục. Điểm nổi bật của hệ thống là khả năng cấu hình từ xa các thiết bị, giúp đơn giản hóa việc cài đặt, vận hành và bảo trì. Ngoài ra, hệ thống sẽ tích hợp các công nghệ truyền thông không dây tiên tiến như Wi-Fi và MQTT, kết hợp với module NRF24 để tạo ra một mạng lưới truyền thông ổn định, đáp ứng nhu cầu giao tiếp dữ liệu nhanh chóng và tiết kiệm năng lượng.

Hệ thống này không chỉ hỗ trợ việc giám sát các thông số quan trọng (như nhiệt độ, áp suất, hoặc các thông tin vận hành khác) mà còn đảm bảo tính linh hoạt khi mở rộng hoặc nâng cấp. Đồng thời, việc sử dụng các công nghệ hiện đại sẽ giúp tối ưu hóa hiệu suất vận hành, giảm thiểu thời gian chết của thiết bị, và hướng đến việc xây dựng một hệ thống quản lý phù hợp với xu hướng chuyển đổi số trong công nghiệp 4.0.

Thông qua đề tài này, nhóm nghiên cứu hy vọng tạo ra một nền tảng quản lý dữ liệu hiệu quả, dễ triển khai và tiết kiệm chi phí, đồng thời góp phần nâng cao hiệu quả quản lý sản xuất cho doanh nghiệp..

Chương 2. LÝ THUYẾT TỔNG QUAN

2.1. Thiết bị

2.1.1. Raspberry Pi 4 Model B

Raspberry Pi là một máy tính có kích thước nhỏ gọn, giá thành rẻ có thể cắm vào màn hình máy tính và sử dụng chuột, bàn phím tiêu chuẩn. Nó là một thiết bị nhỏ có khả năng cho phép mọi người khám phá máy tính và học cách lập trình bằng các ngôn ngữ như Scratch và Python. Nó có khả năng làm được hầu hết những gì chúng ta cần trên một máy tính thông thường như truy cập Internet, xem video, soạn thảo, chơi trò chơi, làm các server nhỏ trong gia đình,...

Hơn nữa, Raspberry Pi có khả năng tương tác với thế giới bên ngoài thông qua các chân điều khiển.



Hình 2. 1. Raspberry Pi4 model B

2.1.2. Camera ELP 2MP

Camera ELP 2MP là một trong những sản phẩm phổ biến trong lĩnh vực camera giám sát và các ứng dụng nhúng. Được thiết kế với nhiều tính năng nổi bật, camera này thường được sử dụng trong các dự án DIY, robot, và hệ thống giám sát.

Tính năng:

- Độ phân giải 2MP cho hình ảnh rõ nét, chi tiết.
- Giao tiếp USB hoặc MIPI giúp dễ dàng kết nối với các thiết bị khác như Raspberry Pi.
- Thích hợp cho nhiều ứng dụng như giám sát an ninh, hệ thống điều khiển từ xa.



Hình 2. 2. Camera ELP 2MP

2.1.3. ESP32-Camera & Module camera OV2640

2.1.3.1. ESP32-Camera:

ESP32-CAM^[1] có một module camera cỡ nhỏ có thể hoạt động như một hệ thống độc lập với kích thước 27x40.5x4.5mm và dòng ở chế độ deep sleep lên đến 6mA.

ESP32-CAM được đóng gói DIP-16 (Dual In-line Package) và có thể được lắp trực tiếp vào bo mạch chủ, cung cấp cho khách hàng chế độ kết nối với độ tin cậy cao, thuận tiện cho việc ứng dụng trong các thiết bị IoT khác nhau.



Hình 2. 3. ESP32-Camera và module camera OV2640

2.1.3.2. Module camera OV2640:

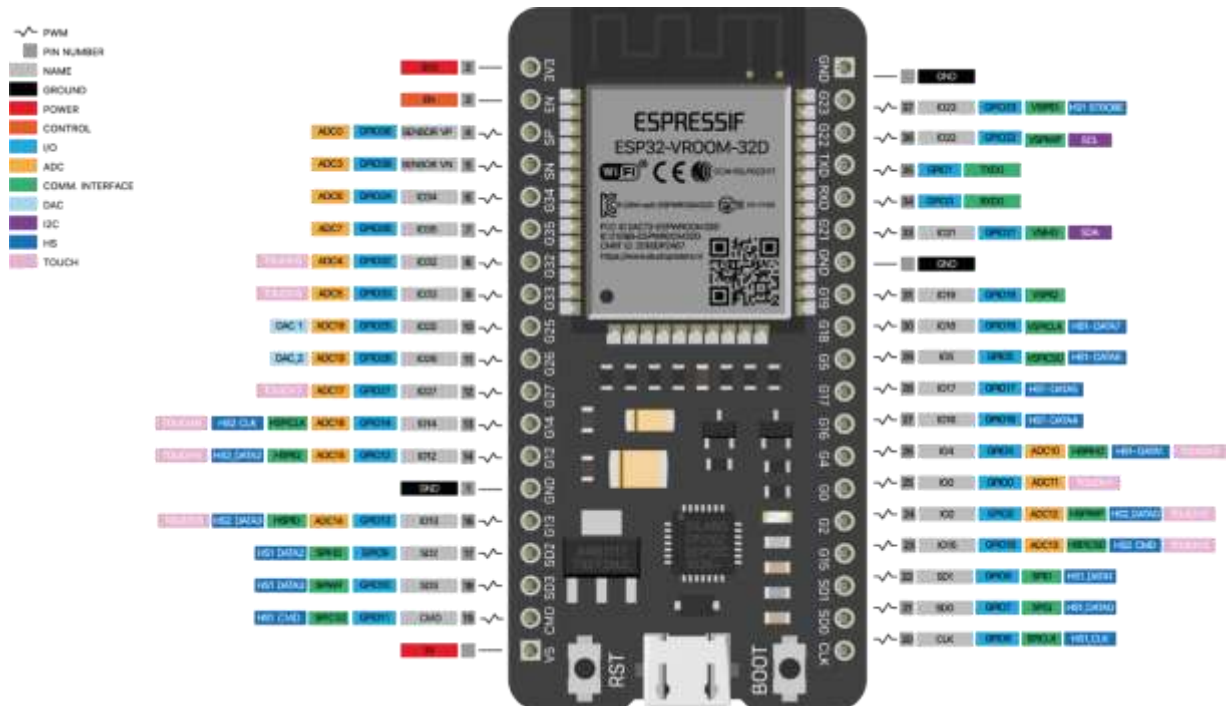
Mô-đun máy ảnh OV2640, sử dụng cảm biến CMOS 1/4 inch OV2640. Với độ nhạy, tính linh hoạt cao, hỗ trợ cho đầu ra JPEG. Và có thể hỗ trợ cân bằng trắng, màu sắc, độ bão hòa, độ tương phản và nhiều tham số khác được thiết lập để hỗ trợ định dạng JPEG / RGB565.

2.1.4. ESP32-WROOM32

ESP32-WROOM32^[2] là một module phát triển dựa trên vi điều khiển ESP32, cung cấp khả năng kết nối không dây mạnh mẽ và nhiều tính năng vượt trội, thích hợp cho các ứng dụng IoT (Internet of Things) và các dự án nhúng.

Tính năng:

- Tích hợp Wi-Fi và Bluetooth cho phép kết nối dễ dàng với các thiết bị và mạng.
- Hỗ trợ nhiều giao thức kết nối như HTTP, MQTT
- Hỗ trợ Arduino IDE và ESP-IDF



Hình 2. 4. ESP32-WROOM32

2.1.5. Module NRF24L01

Module NRF24L01^[3] là một module không dây sử dụng công nghệ truyền thông tần số vô tuyến 2.4 GHz, cho phép truyền tải dữ liệu giữa các thiết bị trong khoảng cách ngắn. Nó rất phổ biến trong các ứng dụng IoT và các dự án nhúng nhờ vào tính năng tiết kiệm năng lượng, dễ sử dụng và chi phí thấp.

Tính năng:

- Cho phép truyền tải dữ liệu giữa các module mà không cần dây dẫn
- Có thể kết nối với nhiều vi điều khiển như Arduino, ESP32, Raspberry Pi thông qua giao thức SPI.



Hình 2. 5. Module NRF24L01

2.2. Công nghệ sử dụng

2.2.1. Truyền thông 2.4GHz

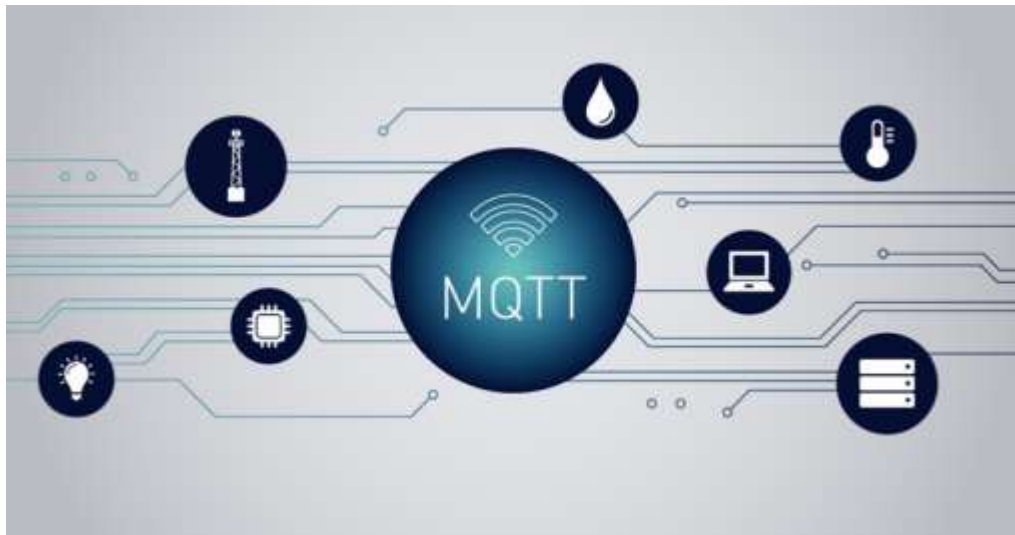
Truyền thông 2.4GHz là một dạng truyền thông không dây phổ biến sử dụng tần số 2.4 GHz cho nhiều ứng dụng khác nhau, từ mạng Wi-Fi đến các thiết bị IoT. Tần số này nằm trong băng tần ISM^[4] (Industrial, Scientific, and Medical)

Tính năng:

- Tần số 2.4 GHz nằm trong băng tần ISM, trải dài từ 2.4 GHz đến 2.5 GHz
- Phạm vi hoạt động từ 30m đến 100m. (Có thể xa hơn, tùy vào thiết bị sử dụng)
- Các tiêu chuẩn Wi-Fi như 802.11b/g/n có tốc độ từ 11 Mbps đến 600 Mbps.

2.2.2. MQTT

MQTT^[3] (Message Queuing Telemetry Transport) là một giao thức truyền tin nhắn nhẹ được sử dụng rộng rãi trong các ứng dụng Internet of Things (IoT). Giao thức này được thiết kế để truyền tin nhắn hiệu quả và đáng tin cậy giữa các thiết bị trong mạng IoT.



Hình 2. 6. Giới thiệu MQTT

2.2.2.1. Đặc điểm:

- Sử dụng gói tin nhỏ gọn, giúp giảm thiểu băng thông và thời gian truyền tải.
- Hỗ trợ nhiều mức độ xác nhận tin nhắn, giúp đảm bảo tin nhắn được truyền tải thành công.
- MQTT là giao thức mã nguồn mở, miễn phí và có thể được sử dụng bởi bất kỳ ai, cú pháp đơn giản và dễ hiểu, giúp cho việc triển khai và sử dụng dễ dàng.

2.2.2.2. Kiến trúc:

MQTT sử dụng mô hình publish-subscribe để truyền tin nhắn. Trong mô hình này, các thiết bị được chia thành hai loại:

- Publisher (Nhà xuất bản): Thiết bị gửi tin nhắn.
- Subscriber (Người đăng ký): Thiết bị nhận tin nhắn.

Publisher gửi tin nhắn đến một broker (trung gian). Broker lưu trữ tin nhắn và chuyển tiếp nó đến tất cả các Subscriber đã đăng ký với chủ đề tin nhắn.

2.2.2.3. Chủ đề (Topic):

Mỗi tin nhắn MQTT được gắn nhãn với một chủ đề. Chủ đề là một chuỗi ký tự xác định loại tin nhắn. Subscriber đăng ký với các chủ đề cụ thể để nhận tin nhắn quan tâm.

2.2.2.4. Cấp độ dịch vụ (QoS):

MQTT hỗ trợ ba cấp độ dịch vụ để đảm bảo tin nhắn được truyền tải thành công:

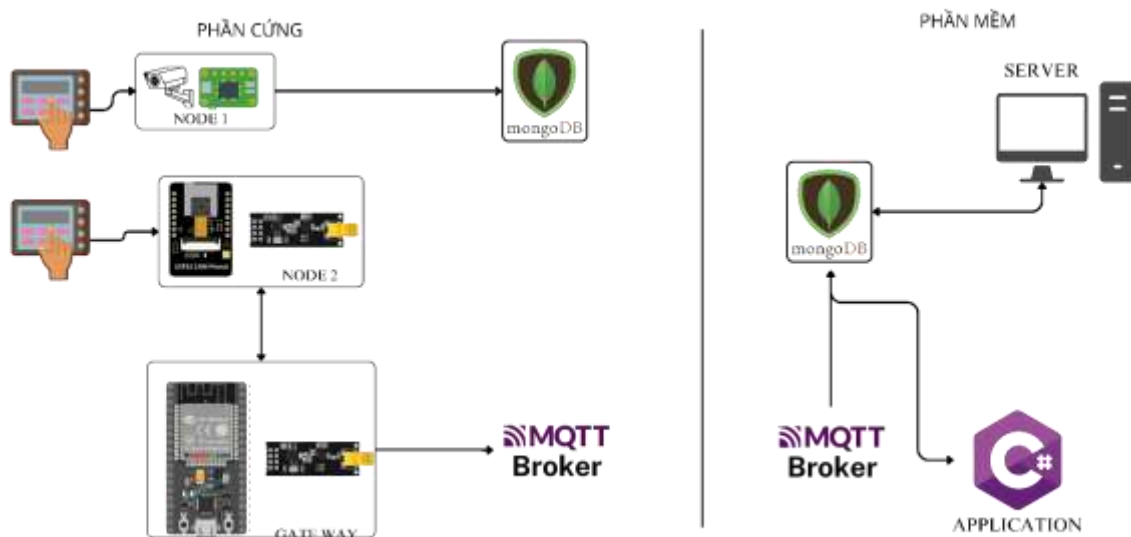
- QoS 0: Mức độ thấp nhất, không đảm bảo tin nhắn được truyền tải thành công.
- QoS 1: Đảm bảo tin nhắn được truyền tải ít nhất một lần.
- QoS 2: Đảm bảo tin nhắn được truyền tải chính xác một lần.

2.2.3. Python và ứng dụng của Python

Python là một ngôn ngữ đa năng, hỗ trợ nhiều phong cách lập trình như lập trình hướng đối tượng, lập trình câu lệnh, lập trình hàm và lập trình thủ tục. Python có nhiều thư viện và framework hữu ích.

Chương 3. HIỆN THỰC ĐỀ TÀI

3.1. Mô hình, phân tích kịch bản



Hình 3. 1. Mô hình hệ thống

Để dễ hiểu hệ thống hơn thì nhóm sẽ chia mô hình thành các phần nhỏ được mô tả chi tiết như ở dưới đây:

3.1.1. Node

3.1.6.1. Node 1



Hình 3. 2. Thành phần Raspberry Pi 4 của Node 1



Hình 3. 3. Camera của Node 1 và màn hình máy dệt

```
CONNECT to MongoDB using specified credentials;
SELECT database "AIOT_CAMERA" and collection "IMAGE";
INITIALIZE webcam;
IF webcam cannot be opened THEN
    DISPLAY error message: "Unable to open webcam"
    EXIT the program;
END
INITIALIZE variable "last_capture_time" with current time;
The program is running WHILE True Do
    READ a frame from the webcam;
    IF the frame is not received THEN
        DISPLAY error message: "Unable to read from webcam";
        EXIT the program;
    END
    IF 10 seconds have passed since the last capture THEN
        UPDATE "last_capture_time" to current time;
        CONVERT the frame to a Pillow image format;
        CONVERT the image to a Base64 string;
        CREATE a document with the following fields:
            - Camera ID;
            - Current UTC time;
            - Base64-encoded image;
        INSERT the document into the MongoDB collection;
        DISPLAY message: "Image uploaded to MongoDB";
    END
END
```

Hình 3. 4. Logic hoạt động của node 1

3.1.6.2. Node 2



Hình 3. 5. Hình ảnh thực tế của node 2


```

Continuously update network status WHILE True DO
  IF receive request from gateway THEN
    Create a header variable to store information about the packet;
    IF the request type is to capture an image THEN
      Call the captureImage() function to enable photo capture;
      Collect image size and data;
      Calculate totalPackets required to send the entire image data;
      FOR 0 TO totalPackets - 1 Do
        CREATE a Packet;
        SET node address and packet type (Data);
        SET packet number;
        Calculating data offset based on packet sequence number;
        Copy image data from offset into packet;
        Packet sending failed OR over the allowed number of attempts WHILE True DO
          SEND packet over network;
          IF sent successfully Then
            EXIT loop;
          END
          ELSE
            INCREMENT number of attempts;
          END
        END
        IF not sent successfully after 10 attempts Then
          SEND error packet to gateway
        END
      END
    END
  END
END

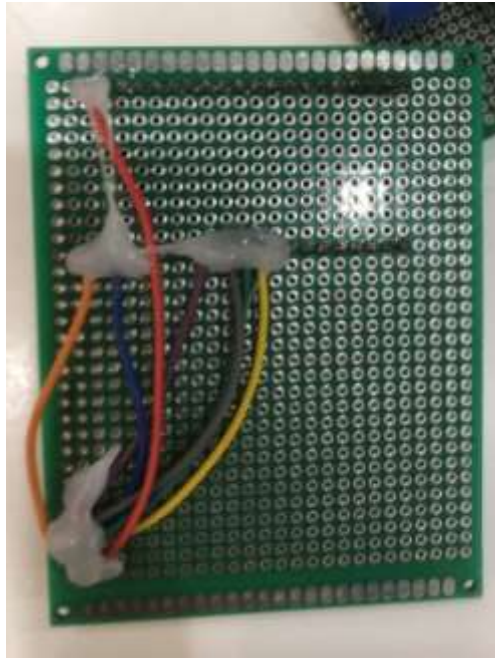
```

Hình 3. 6. Logic hoạt động của node 2

3.1.2. Gateway



Hình 3. 7. Hình ảnh mặt trước của gateway



Hình 3. 8. Hình ảnh mặt sau của gateway

```

Initialize connection with WIFI and MQTT
Initialize addresses of the nodes
WHILE True Do
    IF the time to request sending is sufficient AND the image receiving process has not started yet THEN
        Update request submission time;
        Send photo request;
        Delete previously received image data;
    END
    IF there is a packet from NRF24 THEN
        IF it is an error package THEN
            Print an error message and restart esp;
        END
        IF it is an image information packet THEN
            Start receiving images;
            Reset received packet number;
            Remove old Base64 code;
        END
        IF it is an image data packet AND receive enough packages THEN
            Convert image data to Base64;
            Create topic based on node address;
            Send images via MQTT;
            Reset image receiving status = FALSE;
            Remove old Base64 code;
        END
    END
END
END

```

Hình 3. 9. Logic hoạt động của gateway

3.1.3. MQTT Broker

MQTT Broker là một thành phần quan trọng trong hệ thống, chịu trách nhiệm trung gian cho việc truyền tải dữ liệu giữa các node và gateway. Trong đồ án này, nhóm sử dụng Mosquitto để xây dựng MQTT Broker trên một máy tính chạy hệ điều hành Ubuntu. Mosquitto được lựa chọn vì tính ổn định, nhẹ nhàng và hỗ trợ đầy đủ giao thức MQTT, phù hợp với các ứng dụng IoT.

Các bước cài đặt Mosquitto trên ubuntu:

- Cập nhật hệ thống và cài đặt Mosquitto: Cập nhật danh sách gói và cài đặt Mosquitto từ kho chính thức:

```
sudo apt update && sudo apt install mosquitto mosquitto-clients -y
```

- Cấu hình Mosquitto: Tạo file cấu hình Mosquitto nếu chưa tồn tại:

```
sudo nano /etc/mosquitto/mosquitto.conf
```

- Thêm các dòng sau vào file *mosquitto.conf* để vô hiệu hóa chế độ ẩn danh và bảo vệ bằng mật khẩu:

```
allow_anonymous false
```

```
password_file /etc/mosquitto/passwd
```

- Tạo username/password bảo mật: Tạo tệp mật khẩu và thêm người dùng:

```
sudo mosquitto_passwd -c /etc/mosquitto/passwd <user>
```

- Khởi động lại dịch vụ Mosquitto: Áp dụng các thay đổi và đảm bảo Mosquitto chạy tự động:

```
sudo systemctl restart mosquitto
```

```
sudo systemctl enable mosquitto
```

Trên máy tính chạy Mosquitto, một chương trình (dịch vụ) Python được triển khai để:

- Lắng nghe các topic trên MQTT Broker: Dịch vụ đăng ký nhận tin nhắn từ các topic có tên bắt đầu bằng "image/", điều này cho phép dịch vụ nhận được hình ảnh từ nhiều camera khác nhau.
- Xử lý dữ liệu nhận được: Dữ liệu nhận từ MQTT broker được giả định là ảnh đã được mã hóa Base64. Dịch vụ không cần giải mã lại mà trực tiếp sử dụng dữ liệu Base64 này để lưu trữ. Thông tin ID camera được trích xuất từ tên của topic (dạng image/{idCam}), giúp phân biệt các hình ảnh từ các camera khác nhau.
- Lưu trữ thông tin vào MongoDB: Hình ảnh (dưới dạng Base64), cùng với thời gian nhận (UTC) và ID camera, được lưu trữ vào MongoDB. Mỗi tài liệu trong cơ sở dữ liệu MongoDB chứa các thông tin về camera và trạng thái xử lý của hình ảnh (chưa được xử lý).

Dưới đây là mô tả chi tiết về cách hoạt động của chương trình:

- Kết nối MongoDB: Hàm này kết nối đến MongoDB và chọn database cùng collection cần thiết để lưu trữ hình ảnh.

```
#1. Kết nối MongoDB
...

FUNCTION check_if_processed(image_id, db):
    SELECT the collection "IMAGE";
    FIND the document with _id = image_id and "processed" = True;
    RETURN True if document exists, otherwise False;
```

Hình 3. 10. Mã giả mô tả cách hoạt động của hàm kết nối MongoDB

- Hàm on_connect (kết nối thành công với MQTT broker): Hàm này được gọi khi MQTT client kết nối thành công với broker. Nó đăng ký để nhận các tin nhắn từ các topic bắt đầu bằng "image/".

```
#2. Hàm on_connect (kết nối thành công với MQTT broker)
...

FUNCTION on_connect(client, userdata, flags, rc):
    IF rc == 0:
        PRINT "Connected successfully"
        SUBSCRIBE to all topics starting with "image/#"
    ELSE:
        PRINT "Failed to connect, return code" + rc;

...
```

Hình 3. 11. Mã giả mô tả cách hoạt động của hàm on_connect

- Hàm `on_message` (xử lý tin nhắn nhận được): Hàm này được gọi khi có tin nhắn mới từ broker. Nó giải mã dữ liệu hình ảnh Base64 và lưu vào MongoDB.

```
#3. Hàm on_message (xử lý tin nhắn nhận được)
...

FUNCTION on_message(client, userdata, msg):
    PRINT "Message received on topic" + msg.topic;

    TRY:
        DECODE msg.payload (Base64 data) into image_base64;

        EXTRACT idCam from topic (last part after "image/");

        CREATE document with:
            - "idCam" = idCam
            - "thoi_gian_nhan_anh" = current UTC time
            - "image" = image_base64
            - "processed" = False

        INSERT document into MongoDB collection "IMAGE";
        PRINT "Document inserted with _id: " + document._id;
    CATCH Exception e:
        PRINT "Failed to process image or save to MongoDB: " + e;

    ...
```

Hình 3. 12. Mã giả mô tả cách hoạt động của hàm `on_message`

- Cấu hình mqtt client và kết nối: Hàm này cấu hình MQTT client, bao gồm thiết lập tên người dùng, mật khẩu và các callback để xử lý kết nối và nhận tin nhắn.

```
#4. Cấu hình mqtt client và kết nối
...
FUNCTION configure_and_connect_mqtt():
    CREATE MQTT client;

    SET username and password for the client;

    ASSIGN on_connect function as callback for connecting to the broker;
    ASSIGN on_message function as callback for receiving messages;

    CONNECT to MQTT broker at <broker_address> using port 1883;

    START the MQTT client loop to listen for and process messages continuously;
...

```

Hình 3. 13. Mã giả mô tả cách hoạt động của hàm cấu hình mqtt client và kết nối

- Mô tả quy trình xử lý:

```
#5. Quy trình xử lý
...
FUNCTION process_images():
    CONNECT to MongoDB using connect_to_mongodb();

    CONFIGURE MQTT client using configure_and_connect_mqtt();

    WHILE True:
        WAIT for incoming MQTT messages;

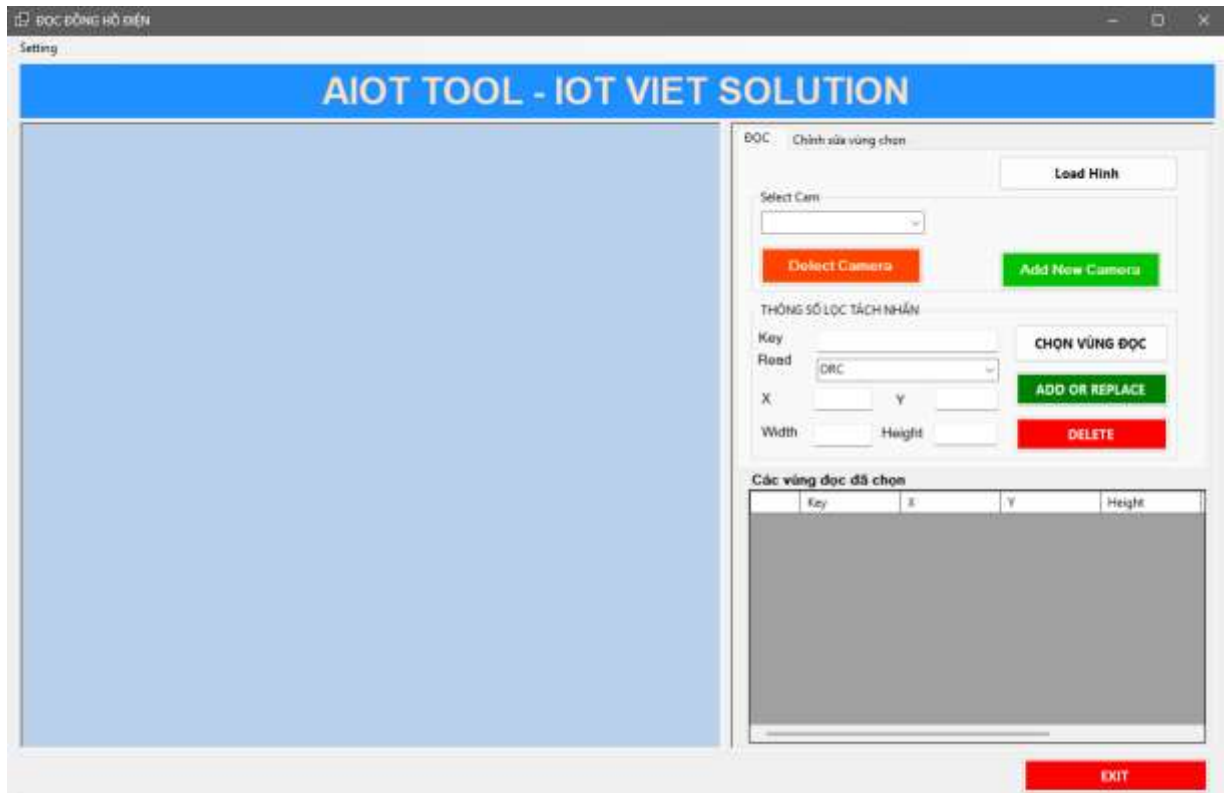
        WHEN a message is received:
            CALL on_message function to process the image and store data in MongoDB;
...

```

Hình 3. 14. Quy trình xử lý của chương trình

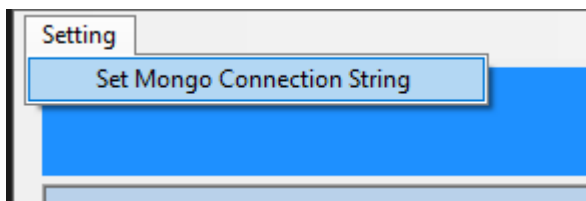
3.1.4. Ứng dụng winform

Khi chạy model xử lý ảnh trên server, thì model sẽ khoanh vùng các điểm có dữ liệu trên ảnh nên cần một ứng dụng có thể đặt tên cho các vùng chọn này. Ở đây nhóm xây dựng một ứng dụng winform bằng C# để đảm nhận nhiệm vụ trên.



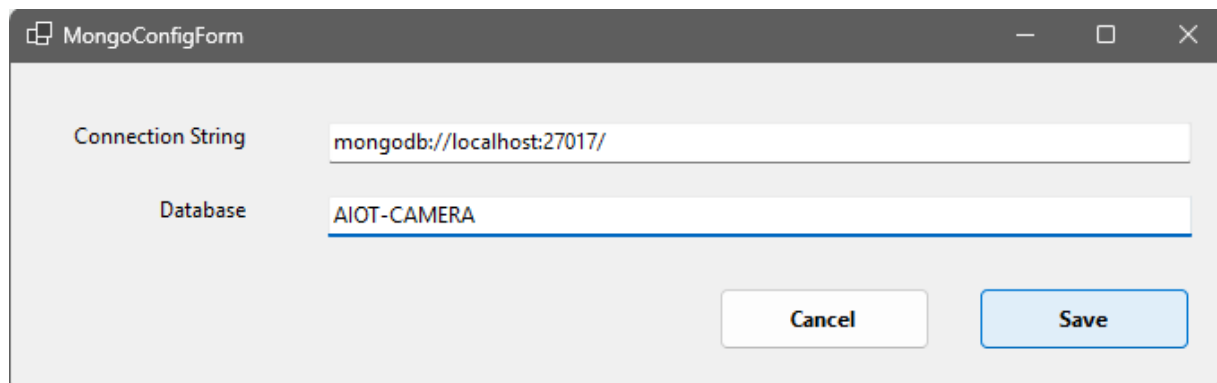
Hình 3. 15. Màn hình chính của ứng dụng

Khi chạy ứng dụng sẽ có màn hình như ở hình 3.8, cần thiết lập kết nối đến Mongo DB để lấy hình ảnh mới nhất của camera để tiến hành đặt tên cho vùng chọn. Trên màn hình của ứng dụng chọn **Setting** → **Set Mongo Connection String** như ở hình 3.9.



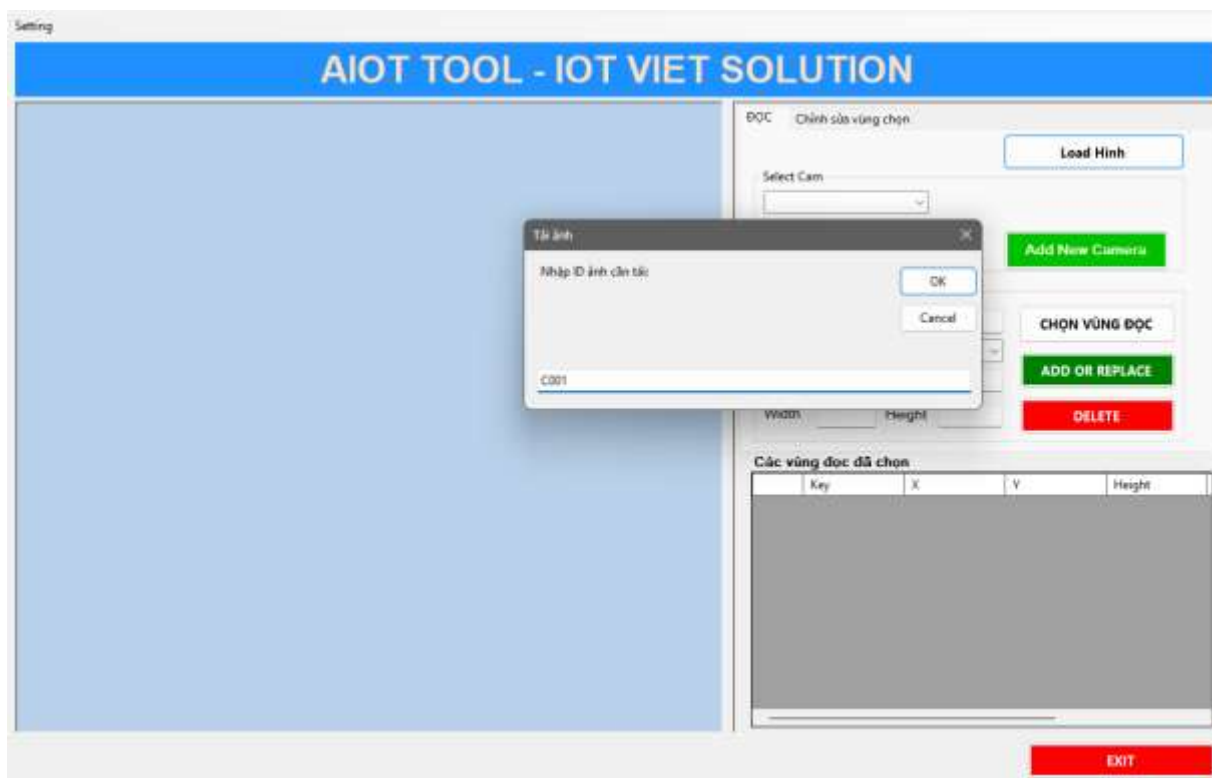
Hình 3. 16. Thiết lập kết nối đến Mongo DB

Sau đó trên hộp thoại **MongoConfigForm** nhập connection string và tên của database tương ứng trên Mongo DB, rồi chọn **Save**.

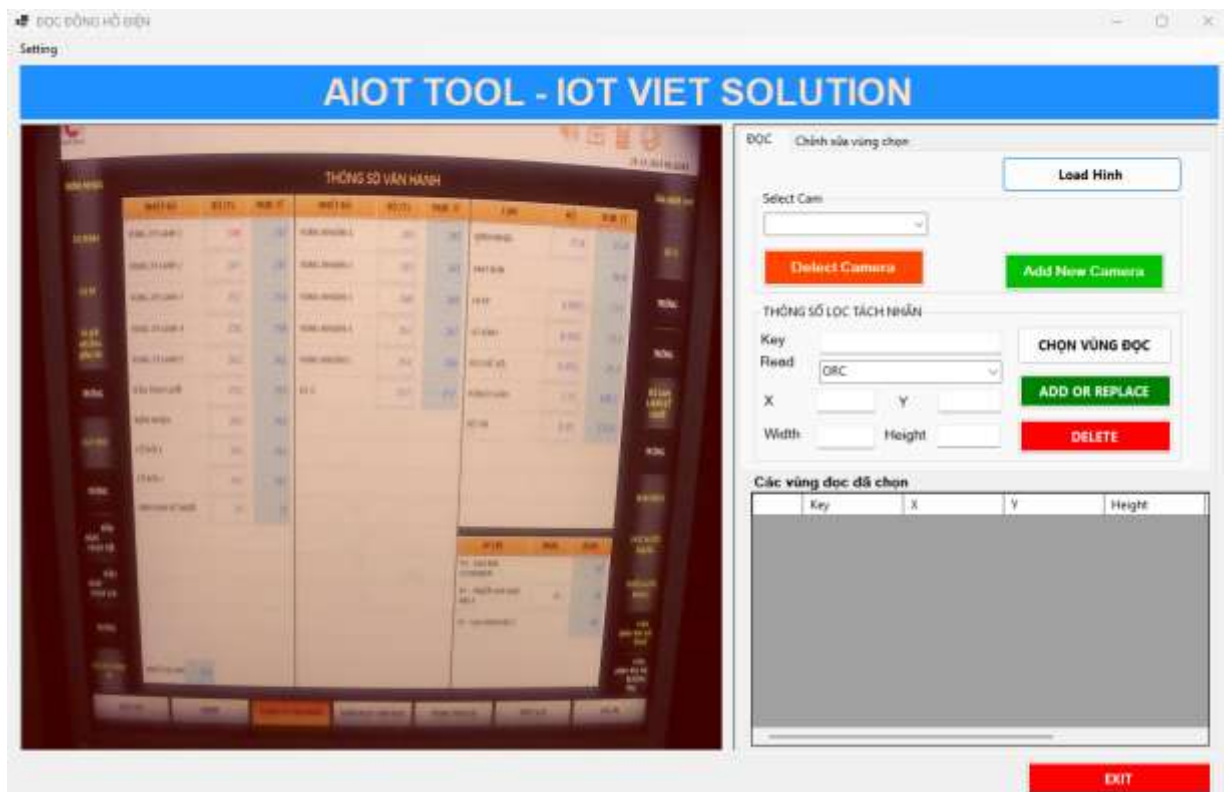


Hình 3. 17. Hộp thoại MongoConfigForm

Sau khi thiết lập kết nối đến Mongo DB, trên màn hình ứng dụng chọn nút **Load Hình**, rồi nhập ID của camera cần cấu hình vào hộp thoại và chọn **OK**.



Hình 3. 18. Nhập ID của camera cần cấu hình



Hình 3. 19. Màn hình ứng dụng khi tải ảnh thành công

Nếu ID nhập vào không khớp với bất kỳ ID của camera nào trên Mongo DB thì ứng dụng sẽ hiển thị thông báo như ở hình 3.13.



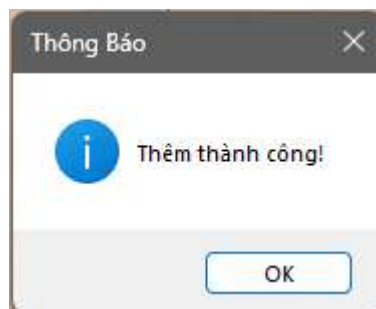
Hình 3. 20. Thông báo khi ID nhập vào không có trong cơ sở dữ liệu

Để tiến hành đặt tên cho các vùng chọn nhấn nút **CHỌN VÙNG ĐỌC**, sau đó chọn vùng dữ liệu cần đặt tên trên hình, rồi nhập tên của vùng chọn trong ô **Key** và nhấn nút **ADD OR REPLACE**.



Hình 3. 21. Chọn vùng đọc và đặt tên

Khi thêm vùng chọn thành công sẽ có pop up thông báo như ở hình 3.15, và trên hình ảnh cũng sẽ hiển thị vùng chọn đã thêm.



Hình 3. 22. Pop up thông báo thêm vùng chọn thành công

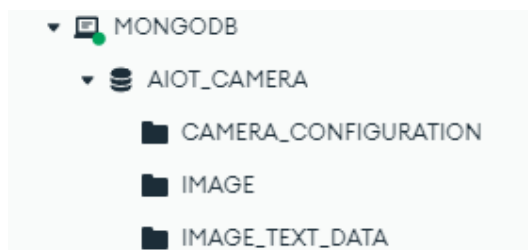


Hình 3. 23. Màn hình ứng dụng sau khi thêm vùng chọn thành công

Nếu muốn chỉnh sửa tên thì chỉ cần chọn Key ở trong ô Các vùng đọc đã chọn rồi thực hiện các bước tương tự việc thêm vùng chọn, hoặc nhấn nút **DELETE** để xóa.

3.1.5. Mongo DB

MongoDB^[5] là hệ quản trị cơ sở dữ liệu NoSQL mã nguồn mở, được sử dụng để lưu trữ dữ liệu dưới dạng document theo cấu trúc JSON linh hoạt. Trong hệ thống này, MongoDB đóng vai trò là nơi lưu trữ các hình ảnh và thông tin liên quan nhận được từ các thiết bị gửi lên hoặc thông qua MQTT Broker.



Hình 3. 24. Server và Database lưu trữ dữ liệu của hệ thống

Chức năng của MongoDB trong hệ thống:

- Bảng IMAGE: Lưu trữ các hình ảnh được gửi lên từ các node (mỗi hình ảnh được lưu trữ bao gồm một id của hình ảnh được tạo ngẫu nhiên bởi cơ sở dữ liệu, id của camera (hay node) chụp ảnh đó, thời gian nhận ảnh (là thời gian ảnh được chụp), nội dung hình ảnh được lưu trữ dưới dạng Base64, và một cờ để đánh dấu ảnh đã được xử lý hay chưa (được tạo mặc định là false, sau khi model xử lý ảnh xong thì cờ này sẽ được chuyển thành true)

```
_id: ObjectId('67578485ec23b1f86f291864')
idCam: "11323"
thoi_gian_nhan_anh: 2024-12-10T00:00:05.611+00:00
image: "/9j/4AAQSkZJRgABAQAAQABAAQ/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHR..."
processed: true
```

```
_id: ObjectId('6757848fec23b1f86f291865')
idCam: "11323"
thoi_gian_nhan_anh: 2024-12-10T00:00:15.677+00:00
image: "/9j/4AAQSkZJRgABAQAAQABAAQ/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHR..."
processed: true
```

```
_id: ObjectId('67578499ec23b1f86f291866')
idCam: "11323"
thoi_gian_nhan_anh: 2024-12-10T00:00:25.705+00:00
image: "/9j/4AAQSkZJRgABAQAAQABAAQ/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHR..."
processed: true
```

```
_id: ObjectId('675784a3ec23b1f86f291867')
idCam: "11323"
thoi_gian_nhan_anh: 2024-12-10T00:00:35.784+00:00
image: "/9j/4AAQSkZJRgABAQAAQABAAQ/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHR..."
processed: true
```

Hình 3. 25. Các hình ảnh được lưu trữ trong collection IMAGE

- Bảng CAMERA_CONFIGURATION: Lưu trữ thông tin cấu hình của các camera (node) bao gồm id camera, vị trí gắn camera, và thông tin các vùng chọn của camera.

```
_id: ObjectId('674553a28f7d8301f506d8c3')
IdCam : "11323"
Location : "NHA_MAY_SOI"
▼ ReadArea : Array (27)
  ▼ 0: Object
    Key : "XY_LANH_1"
    X : 624
    Y : 155
    Height : 48
    Width : 109
    ReadMode : "ORC"
  ▼ 1: Object
    Key : "XY_LANH_2"
    X : 621
    Y : 219
    Height : 50
    Width : 112
    ReadMode : "ORC"
  ▼ 2: Object
    Key : "XY_LANH_3"
    X : 619
    Y : 269
    Height : 43
    Width : 123
    ReadMode : "ORC"
  ▼ 3: Object
    Key : "XY_LANH_4"
    X : 627
    Y : 332
    Height : 46
    Width : 109
    ReadMode : "ORC"
  ▼ 4: Object
    Key : "XY_LANH_5"
    X : 619
    Y : 384
    Height : 44
    Width : 120
    ReadMode : "ORC"
  ▼ 5: Object
    Key : "DAU_THAY_LUOI"
    X : 627
    Y : 436
    Height : 43
    Width : 109
```

Hình 3. 26. Thông tin camera được lưu trữ trong bảng CAMERA_CONFIGURATION

- Bảng IMAGE_TEXT_DATA: Lưu trữ thông tin của các hình ảnh đã được xử lý id của ảnh trong bảng IMAGE, thời gian đọc ảnh, id camera, và giá trị đọc được từ các vùng chọn trong bảng CAMERA_CONFIGURATION.

```
_id: ObjectId('67599ef92695f2e09c021c88')
Id : "1733926649067"
IdImage : "67599edb63c00c9eedaaa789"
ReadingTime : "2024-12-11T21:17:21.105053+07:00"
CameraStation : "11323"
▼ RawValue : Array (27)
  ▼ 0: Object
    X : 786
    Y : 192
    Value : "295"
  ▼ 1: Object
    X : 789
    Y : 240
    Value : "250"
  ▼ 2: Object
    X : 791
    Y : 300
    Value : "255"
  ▼ 3: Object
    X : 783
    Y : 347
    Value : "250"
  ▼ 4: Object
    X : 792
    Y : 402
    Value : "260"
  ▼ 5: Object
    X : 785
    Y : 456
    Value : "250"
  ▼ 6: Object
    X : 792
    Y : 509
    Value : "260"
  ▼ 7: Object
    X : 0
    Y : 0
    Value : "0"
  ▼ 8: Object
    X : 791
    Y : 611
    Value : "260"
```

Hình 3. 27. Các dữ liệu được lưu trữ trong bảng IMAGE_TEXT_DATA

3.1.6. Server

3.1.6.1. Chương trình xử lý hình ảnh

Để xử lý hình ảnh đã được lưu trữ trên MongoDB, nhóm sử dụng một chương trình Python với thư viện PaddleOCR nhằm thực hiện OCR (nhận diện ký tự quang học) trên hình ảnh. PaddleOCR là một thư viện mã nguồn mở mạnh mẽ, được phát triển trên nền tảng PaddlePaddle. Thư viện này không chỉ hỗ trợ nhận diện văn bản từ hình ảnh mà còn cung cấp khả năng nhận diện đa ngôn ngữ (hơn 80 ngôn ngữ), bao gồm tiếng Việt.

Một điểm nổi bật của PaddleOCR là chức năng nhận diện văn bản theo góc độ (angle classification), giúp cải thiện đáng kể độ chính xác khi xử lý hình ảnh có văn bản bị nghiêng hoặc xoay. Thư viện cũng được tối ưu hóa để hoạt động trên cả CPU lẫn GPU, đảm bảo hiệu năng cao ngay cả trên các thiết bị phần cứng hạn chế.

Cùng với PaddleOCR, chương trình còn sử dụng một số thư viện cần thiết khác như:

- pymongo: Để kết nối và truy vấn cơ sở dữ liệu MongoDB, nơi lưu trữ các hình ảnh cần xử lý.
- cv2 (OpenCV): Để xử lý ảnh đầu vào, như cắt, thay đổi kích thước hoặc chuyển đổi định dạng ảnh.
- numpy: Để xử lý dữ liệu ảnh dưới dạng mảng số, hỗ trợ hiệu quả cho các tác vụ liên quan đến ma trận.
- base64: Để mã hóa và giải mã dữ liệu ảnh dưới dạng chuỗi Base64.
- datetime: Để quản lý và định dạng thời gian khi lưu trữ hoặc xử lý dữ liệu.

Việc kết hợp các thư viện này giúp đảm bảo chương trình hoạt động hiệu quả và chính xác trong việc xử lý hình ảnh cũng như lưu trữ và quản lý dữ liệu trong hệ thống.

Dưới đây là trình bày về logic xử lý dữ liệu của chương trình:

- Hàm kiểm tra ảnh đã được xử lý: Ban đầu chương trình sẽ quét bảng IMAGE và lấy ảnh mới nhất thông qua trường `thoi_gian_nhan_anh`. Sau đó sử dụng hàm kiểm tra để kiểm tra xem ảnh đã được xử lý chưa bằng cách tra cứu trường `processed` trong MongoDB.

```
#1. Kiểm Tra Ảnh Đã Được Xử Lý
...

FUNCTION check_if_processed(image_id, db):
    SELECT the collection "IMAGE";
    FIND the document with _id = image_id and "processed" = True;
    RETURN True if document exists, otherwise False;
...
```

Hình 3. 28. Mã giả mô tả cách hoạt động của hàm kiểm tra hình ảnh đã được xử lý

- Hàm đánh dấu ảnh là đã được xử lý: Sau khi xử lý một ảnh, hàm này sẽ cập nhật trường `processed` thành `True`.

```
#2. Đánh Dấu Ảnh Là Đã Được Xử Lý
...

FUNCTION mark_as_processed(image_id, db):
    SELECT the collection "IMAGE";
    UPDATE the document with _id = image_id:
        SET "processed" = True;
...
```

Hình 3. 29. Mã giả mô tả cách hoạt động của hàm đánh dấu ảnh đã được xử lý

- Hàm lưu kết quả OCR vào MongoDB: Hàm này lưu kết quả OCR vào MongoDB, bao gồm thông tin về ảnh và văn bản nhận dạng được.

```
#3. Lưu Kết Quả OCR Vào MongoDB
...

FUNCTION save_results_to_mongodb(results, collection, id_image, reading_time, raw_value, read_value, configuration):
    GET "IdCam" from the configuration or set "Unknown" as default;

    CREATE a document with the following fields:
    - "IdImage": id_image;
    - "ReadingTime": reading_time;
    - "CameraStation": IdCam;
    - "RawValue": raw_value;
    - "ReadValue": read_value;
    - "IsIoTsync": False;

    INSERT the document into the specified collection;
...
```

Hình 3. 30. Mã giả mô tả cách hoạt động của hàm lưu kết quả OCR và MongoDB

- Hàm lấy ảnh mới nhất chưa được xử lý: Hàm này truy vấn MongoDB để lấy ảnh mới nhất chưa được xử lý, đảm bảo xử lý theo thời gian.

```
#4. Lấy Ảnh Mới Nhất Chưa Được Xử Lý
...

FUNCTION get_latest_image(db, time_threshold):
    SELECT the collection "IMAGE";
    FIND one document where:
    - "thoi_gian_nhan_anh" > time_threshold;
    - "processed" is not True;

    SORT results by "thoi_gian_nhan_anh" descending;
    RETURN the newest document;
...
```

Hình 3. 31. Mã giả mô tả cách hoạt động của hàm lấy ảnh mới nhất chưa được xử lý

- Hàm phát hiện văn bản trong vùng cấu hình: Hàm này sử dụng PaddleOCR để phát hiện văn bản trong các vùng đã cấu hình trước của ảnh.

```
#5. Phát Hiện Văn Bản Trong Vùng Cấu Hình
...
FUNCTION detect_text_in_configured_areas(image, read_areas):
    INITIALIZE PaddleOCR with "use_angle_cls" and language "en";

    INITIALIZE empty lists for results and raw_value;

    FOR each area in read_areas:
        EXTRACT "Key", "X", "Y", "Height", "Width" from the area;

        CROP the image based on "X", "Y", "Height", "Width";
        RESIZE the cropped image to double its size;

        PERFORM OCR on the cropped image;

        IF OCR results are found:
            FOR each detected text line:
                EXTRACT the text and its bounding box;
                ADJUST bounding box coordinates to the original image;

                CALCULATE the center (X, Y) of the text box;

                ADD the detected text and center to the results and raw_value lists;
            ELSE:
                APPEND default values ("0") for empty areas to results and raw_value;

    RETURN results and raw_value;
...
```

Hình 3. 32. Mã giả mô tả cách hoạt động của hàm phát hiện văn bản trong vùng cấu hình

- Hàm xử lý hình ảnh liên tục: Hàm này thực hiện vòng lặp liên tục để lấy ảnh mới, xử lý và lưu kết quả OCR.

```
#6. Xử Lý Ảnh Liên Tục
...

FUNCTION process_images_continuously():
    CONNECT to MongoDB at "localhost:27003";
    SELECT the database "AIOT_CAMERA";

    GET the latest image document from the "IMAGE" collection;
    SET time_threshold to the "thoi_gian_nhan_anh" of the latest image;

    WHILE True:
        TRY:
            CALL get_latest_image() to retrieve the newest unprocessed image after time_threshold;

            IF new_image is found:
                EXTRACT "id_image" and update time_threshold;

                GET camera configuration from "CAMERA_CONFIGURATION" based on the "idCam" of the image;
                CONTINUE if configuration or "ReadArea" is missing;

                DECODE Base64 image data to an OpenCV image;

                IF image has already been processed:
                    CONTINUE to the next iteration;

                MARK the image as processed in MongoDB;

                CALL detect_text_in_configured_areas() to perform OCR;
                SAVE the OCR results into the "IMAGE_TEXT_DATA" collection;

            ELSE:
                WAIT for 30 seconds;

        CATCH any exception:
            WAIT for 30 seconds;
    ...
```

Hình 3. 33. Mã giả mô tả cách hoạt động của hàm xử lý hình ảnh liên tục

- Luồng Chương Trình:
 - Kết Nối MongoDB: Chương trình kết nối với cơ sở dữ liệu MongoDB và các collection liên quan.
 - Xử Lý Ảnh Liên Tục:
 - Chương trình lấy ảnh mới nhất chưa được xử lý.
 - Lấy cấu hình vùng đọc ảnh từ MongoDB.

- Giải mã dữ liệu ảnh Base64 và thực hiện nhận dạng văn bản qua PaddleOCR.
- Đánh dấu ảnh đã được xử lý và lưu kết quả vào MongoDB.
- Phát Hiện Văn Bản: Dùng PaddleOCR để nhận diện văn bản từ các vùng ảnh đã chỉ định.
- Vòng Lặp Liên Tục: Chương trình luôn kiểm tra và xử lý ảnh mới, đảm bảo hệ thống hoạt động liên tục.

3.1.6.2. Web hiển thị dữ liệu



The screenshot shows a web browser window displaying a web application titled "QUẢN LÝ CAMERA". The application features a table with four columns: "STT", "Camera", "Vị trí", and "Thông tin cấu hình". There are two rows of data in the table. Each row has a button next to the "Camera" and "Thông tin cấu hình" columns.

STT	Camera	Vị trí	Thông tin cấu hình
1.		NHA_MAY_90I	
2.		NHA_MAY_90I	

Hình 3. 34. Giao diện chính của trang web quản lý dữ liệu

Ở màn hình chính hiển thị bảng các camera được quản lý trong hệ thống, với các cột chính bao gồm:

- Số thứ tự camera.
- Tên Camera: Gồm các nút chuyển hướng đến các bảng hình ảnh đã xử lý trên Camera tương ứng.
- Vị trí đặt Camera.
- Thông tin cấu hình của Camera: Gồm các nút chuyển hướng đến các bảng thông tin cấu Hình Camera.

Quy trình xử lý dữ liệu trong bảng:

- API Endpoint: Định nghĩa API /cameras được định nghĩa trong file app.js và trả về dữ liệu các camera đang quản lý từ MongoDB.
- Fetching Data: Dữ liệu của Camera được lấy về từ API /cameras trong mã nguồn script.js bằng hàm fetchCameraConfigurations().
- Rendering Data: Sau khi lấy dữ liệu về, hàm fetchCameraConfigurations() sẽ tạo các hàng (<tr>) và ô (<td>) tương ứng với dữ liệu Camera và thêm vào bảng.

THÔNG TIN CẤU HÌNH CAMERA 11323

STT	Key	X	Y	Height	Width	ReadMode
1	XY_LANH_1	186	150	46	109	ORC
2	XY_LANH_2	178	210	53	117	ORC
3	XY_LANH_3	186	269	50	104	ORC
4	XY_LANH_4	186	326	41	104	ORC
5	XY_LANH_5	188	374	48	106	ORC
6	DAU_THAY_LUOI	188	429	46	101	ORC
7	BOM_NHUA	183	483	44	112	ORC
8	CO_NOI_1	180	536	50	106	ORC
9	CO_NOI_2	186	591	44	101	ORC
10	BE_NUOC	186	643	37	101	ORC
11	VUNG_KHUON_1	662	158	50	112	ORC
12	VUNG_KHUON_2	646	217	50	120	ORC
13	VUNG_KHUON_3	646	276	51	117	ORC
14	VUNG_KHUON_4	646	329	43	112	ORC
15	VUNG_KHUON_5	638	383	43	117	ORC

Hình 3. 35. Giao diện xem thông tin các vùng đọc của camera



Ở bảng thông tin cấu hình Camera gồm các thông tin sau:

- Số thứ tự các Key.
- Tên Key, là các vùng đọc được đánh giấu của camera.
- Các cột X, Y, Height, Width xác định tọa độ của vùng đọc và chiều dài chiều rộng để xác định vùng đọc trên màn hình.
- ReadMode tương ứng với từng Key.

Quy trình xử lý dữ liệu trong bảng:

- API Endpoint: Định nghĩa API /cameras được định nghĩa trong file app.js và trả về dữ liệu các camera đang quản lý từ MongoDB.
- Fetching Data: Dữ liệu của Camera được lấy về từ API /cameras trong mã nguồn script.js bằng hàm fetchCameraConfigurations(), sau đó sẽ truyền dữ liệu nhận được là config vào hàm viewConfiguration(config) để thực hiện bước Rendering Data.
- Rendering Data: Sau khi lấy dữ liệu về, hàm viewConfiguration(config) sẽ tạo các hàng (<tr>) và ô (<td>) tương ứng với dữ liệu Camera và thêm vào bảng.

HÌNH ẢNH ĐÃ XỬ LÝ 11323

STT	Thời gian	Read Value	Ảnh
1	1/13/2025, 9:06:41 AM	XEM	
2	1/13/2025, 9:06:04 AM	XEM	
3	1/13/2025, 9:05:26 AM	XEM	
4	1/13/2025, 9:04:50 AM	XEM	
5	1/13/2025, 9:04:42 AM	XEM	
6	1/13/2025, 9:04:04 AM	XEM	
7	1/13/2025, 9:03:27 AM	XEM	
8	1/13/2025, 9:02:50 AM	XEM	
9	1/13/2025, 9:02:42 AM	XEM	
10	1/13/2025, 9:02:04 AM	XEM	

Trước
1
/ 5279
Sau

Hình 3. 36. Giao diện xem dữ liệu từ hình ảnh của camera

Tiếp theo là bảng lưu dữ liệu về hình ảnh đã gửi từ Camera được chia thành nhiều trang với các thông tin bao gồm:

- Số thứ tự các bức ảnh được chụp từ camera.
- Thời gian ảnh chụp được từ camera, được sắp xếp từ mới nhất đến cũ nhất.

- ReadValue: Gồm 1 nút chuyển hướng đến bảng hiển thị các value của từng Key của Camera.
- Ảnh: Hiển thị ảnh thực tế chụp được, có thể xem chi tiết khi chọn vào từng bức ảnh.
- Số trang: mỗi trang hiển thị được 10 ảnh, số trang phụ thuộc vào số lượng ảnh đã được nhận.

Quy trình xử lý dữ liệu trong bảng:

- API Endpoint: Định nghĩa API /processed/:idCam được định nghĩa trong file app.js và trả về dữ liệu ảnh đã xử lý từ MongoDB.
- Fetching Data: Dữ liệu được lấy từ API /processed/:idCam bằng hàm fetchImageProcessed trong script.js.
- Rendering Data: Sau khi lấy được dữ liệu từ API, hàm displayPage sẽ tạo các hàng (<tr>) và các ô (<td>) tương ứng với dữ liệu ảnh đã xử lý và thêm vào bảng.

READ VALUE

STT	Key	Value	X	Y
1	XY_LANH_1	231	346	203
2	XY_LANH_2	242	352	254
3	XY_LANH_3	248	343	306
4	XY_LANH_4	255	341	359
5	XY_LANH_5	259	338	417
6	DAU_THAY_LUOI	250	337	468
7	BOM_NHUA	260	342	520
8	CO_NOI_1	260	345	570
9	CO_NOI_2	260	340	617
10	BE_NUOC	37	352	666
11	VUNG_KHUON_1	255	816	204
12	VUNG_KHUON_2	255	824	256
13	VUNG_KHUON_3	255	816	308
14	VUNG_KHUON_4	256	807	364
15	VUNG_KHUON_5	255	809	419

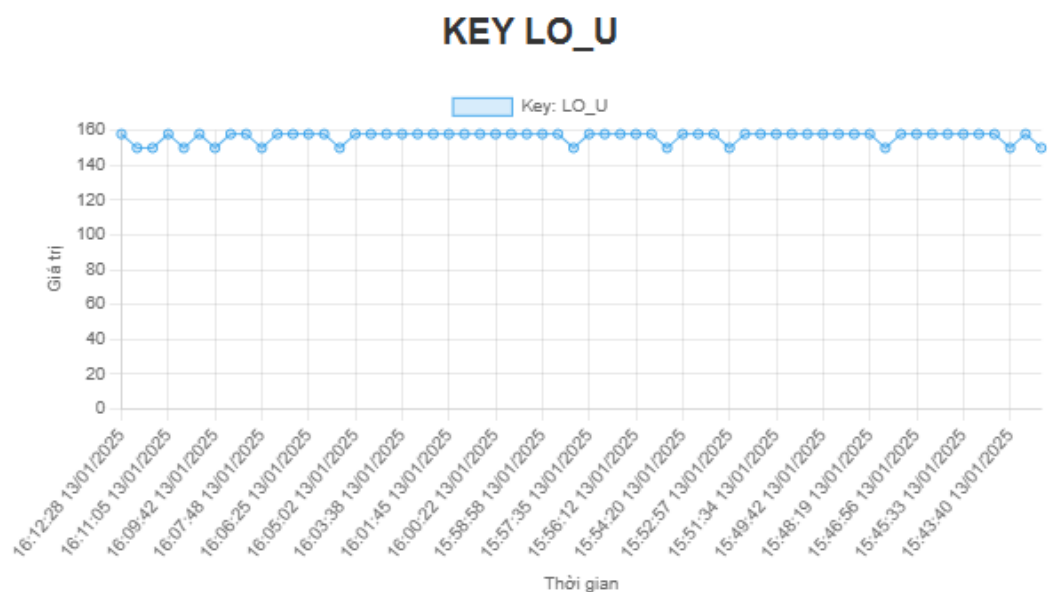
Hình 3. 37. Bảng ReadValue hiển thị dữ liệu sau khi mapping với tên vùng chọn tạo bởi ứng dụng winform

Bảng ReadValue bao gồm các thông tin về Key của từng ảnh nhận được bao gồm:

- Số thứ tự các Key đọc được từ ảnh.
- Tên Key: Đồng thời cũng là nút chuyển hướng đến bảng các giá trị gần nhất của Key đó.
- Giá trị của từng Key của ảnh chụp được.
- Trục tọa độ X,Y của vùng đọc KEY.

Quy trình xử lý dữ liệu trong bảng:

- API Endpoint: Định nghĩa API /processed/:idCam được định nghĩa trong file app.js và trả về dữ liệu ảnh đã xử lý từ MongoDB.
- Fetching Data: Dữ liệu của Camera được lấy về từ API /processed/:idCam bằng hàm fetchImageProcessed(idCam) trong script.js, sau đó sẽ truyền dữ liệu nhận được là config vào hàm viewReadValue(image.ReadValue) để thực hiện bước Rendering Data.
- Rendering Data: Sau khi lấy dữ liệu về, hàm viewReadValue(image.ReadValue) sẽ tạo các hàng (<tr>) và ô (<td>) tương ứng với dữ liệu Camera và thêm vào bảng.



Thời gian	Giá trị
1/13/2025, 4:12:28 PM	158
1/13/2025, 4:11:51 PM	150
1/13/2025, 4:11:43 PM	150
1/13/2025, 4:11:05 PM	158
1/13/2025, 4:10:27 PM	150
1/13/2025, 4:09:49 PM	158
1/13/2025, 4:09:42 PM	150
1/13/2025, 4:09:04 PM	158
1/13/2025, 4:08:26 PM	158
1/13/2025, 4:07:48 PM	150
1/13/2025, 4:07:41 PM	158
1/13/2025, 4:07:03 PM	158

Hình 3. 38. Bảng và biểu đồ Key hiển thị các dữ liệu mới nhất của một Key qua các ảnh mới nhất.

Bảng và biểu đồ Key hiển thị các thông tin của các Key được đọc từ các ảnh mới nhất bao gồm các thông tin như là:

- Thời gian Key được đọc, tức là thời gian nhận được ảnh tương ứng.
- Giá trị của Key tại thời điểm chụp ảnh.

Quy trình xử lý dữ liệu trong biểu đồ và bảng:

- API Endpoint: Định nghĩa API `/key-data/:idCam/:key` để lấy dữ liệu liên quan đến key cụ thể từ camera.
- Fetching Data: Hàm `showKeyData(key)` trong file `script.js` sẽ gọi API `/key-data/:idCam/:key` để lấy dữ liệu liên quan đến key cụ thể từ camera.
- Rendering Data: Hàm `showKeyData(key)` tạo các hàng và ô tương ứng với dữ liệu key và thêm vào bảng, đồng thời sử dụng thư viện `Chart.js` để vẽ biểu đồ bằng các dữ liệu lấy được từ API.

3.1.6.3. Chương trình xóa hình ảnh của Database

Do chu kì chụp ảnh của các node là 30 giây một lần và có nhiều node như vậy nên lượng dữ liệu trên database là rất lớn, nên cần một chương trình để xóa bớt các ảnh đã được xử lý để giảm dung lượng lưu trữ. Ở đây nhóm sử dụng một chương trình python để xóa các ảnh đã chụp từ 1 tuần trước, chỉ giữ lại dữ liệu đã xử lý.

CAMERA_CONFIGURATION				
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.45 kB	1	2.29 kB	1	20.48 kB

IMAGE				
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
17.59 GB	74 K	234.98 kB	2	2.95 MB

IMAGE_TEXT_DATA				
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
16.44 MB	36 K	2.74 kB	2	1.83 MB

Hình 3. 39. Dung lượng của các bảng trên database

```

BEGIN
    CONNECT to MongoDB using pymongo.MongoClient with connection string
    SELECT database "<database_name>"
    SELECT collection "<collection_name>"

    GET current time in UTC timezone
    CALCULATE one_week_ago as current time minus 1 one_week_ago
    CREATE query to find documents where "thoi_gian_nhan_anh" is less than one_week_ago

    DELETE documents in "<collection_name>" collection that match the query
END

```

Hình 3. 40. Mã giả của chương trình xóa ảnh trên database

Khi chạy chương trình sẽ kết nối đến bảng IMAGE của database, sau đó sử dụng trường “thoi_gian_nhan_anh” để xác định thời gian chụp của ảnh, nếu ảnh nào đã chụp từ một tuần trước thì xóa đi.

Chương 4. ĐÁNH GIÁ, HƯỚNG PHÁT TRIỂN

4.1. Đánh giá

Hệ thống đã được triển khai thành công và mang lại những kết quả tích cực. Sau đây là đánh giá chi tiết về hệ thống:

4.1.1. Hiệu suất vận hành:

Hệ thống đã đáp ứng được yêu cầu giám sát liên tục các thông số quan trọng được hiển thị trên màn hình của máy dẹt.

Tính đồng bộ cao giữa các node, gateway và server.

Hình ảnh từ các camera được xử lý nhanh chóng và đưa vào lưu trữ an toàn.

4.1.2. Điểm mạnh:

Dễ cài đặt và vận hành: Hệ thống có khả năng cấu hình từ xa (raspberry pi có thể cấu hình từ xa bằng SSH), được tích hợp các giao diện người dùng trực quan.

Tự động hoá: Toàn bộ dữ liệu được thu thập, phân tích, và lưu trữ tự động.

Tính khả mở: Hệ thống sử dụng các thiết bị và giao thức phổ biến như ESP32, Raspberry Pi, MQTT, giúp dễ dàng mở rộng hoặc nâng cấp.

4.1.3. Hạn chế:

Thời gian xử lý hình ảnh trong những trường hợp lượng dữ liệu lớn có thể bị chậm.

Các node đòi hỏi điều kiện môi trường vận hành tốt như kết nối Wi-Fi ổn định, và tín hiệu của module NRF24L01 chưa thật sự ổn định.

Chưa tích hợp nhiều công nghệ AI tự động hoá việc xử lý sự cố.

Trang web hiển thị dữ liệu chưa được tối ưu, khi tải lượng lớn dữ liệu sẽ bị chậm.

4.2. Hướng phát triển

4.2.1. Cải tiến hiệu năng:

Tối ưu hoá code xử lý hình ảnh để giảm thời gian xử lý.

Tích hợp GPU và các thư viện mạnh như TensorFlow để tăng tốc việc nhận diện hình ảnh.

4.2.2. Tích hợp AI và Machine Learning:

Phát triển các mô hình phân tích dữ liệu để dự báo xu hướng hoặc phát hiện sự cố sát sao hơn.

Đồng bộ với các hệ thống quản lý tích hợp khác trong doanh nghiệp.

4.2.3. Mở rộng khả năng:

Xây dựng web với khả năng hiển thị dữ liệu bằng các biểu đồ giúp người dùng dễ quản lý hơn.

Chương 5. KẾT LUẬN

5.1. Tóm tắt kết quả

Sau khi thực hiện, đề tài "Hệ thống quản lý tập trung thiết bị IoT công nghiệp" đã hoàn thành các mục tiêu đề ra:

- Xây dựng hệ thống thu thập dữ liệu tự động từ các thiết bị trong môi trường sản xuất.
- Đáp ứng tính đồng bộ, dễ cấu hình và vận hành.
- Tích hợp các công nghệ hiện đại như MQTT, MongoDB, PaddleOCR để xử lý và phân tích dữ liệu hiệu quả.

5.2. Đóng góp của đề tài

Đề tài cung cấp một hệ thống quản lý linh hoạt cho doanh nghiệp trong ngành sản xuất, góp phần tự động hoá và số hóa quy trình.

Tạo ra nền tảng hữu ích cho các nghiên cứu tiếp theo, có thể được áp dụng trong nhiều ngành công nghiệp khác nhau.

5.3. Nhận định và học hỏi

Quá trình thực hiện đề tài giúp nhóm hiểu sâu hơn về các giao thức truyền thông, công nghệ IoT và kỹ năng triển khai thực tế.

Tăng cường kỹ năng làm việc nhóm, quản lý dự án và xử lý sự cố.

TÀI LIỆU THAM KHẢO

- [1] “AI Thinker ESP32-CAM”, PlatformIO. [Trực tuyến]. Available: <https://docs.platformio.org/en/latest/boards/espressif32/esp32cam.html#ai-thinker-esp32-cam>
- [2] “ESP32-S3-WROOM-1ESP32-S3-WROOM-1U”, Espressif. [Trực tuyến]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf
- [3] “Optimized High Speed Driver for nRF24L01(+) 2.4GHz Wireless Transceiver”, Doxygen, 10 1 2025. [Trực tuyến]. Available: <https://nrf24.github.io/RF24/>
- [4] “ISM Bands Around the World”, Mark Harris, 28 9 2021 [Trực tuyến]. Available: <https://resources.altium.com/p/ism-bands-around-world>
- [5] “MongoDB là gì? Định nghĩa đầy đủ và chi tiết nhất về MongoDB”. [Trực tuyến]. Available: <https://topdev.vn/blog/mongodb-la-gi/>