TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA ĐIỆN – ĐIỆN TỬ

BÙI QUANG HUY

Thiết Kế Giao Diện Và Lập Trình Hệ Thống IoT Thông Minh Cho Nuôi Trồng Tảo Giải Pháp Blue Cacbon

ĐỒ ÁN TỔNG HỢP

KỸ THUẬT ĐIỀU KHIỂN VÀ TỰ ĐỘNG HÓA

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2025

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA ĐIỆN – ĐIỆN TỬ

BÙI QUANG HUY - 42001172

Thiết Kế Giao Diện Và Lập Trình Hệ Thống IoT Thông Minh Cho Nuôi Trồng Tảo Giải Pháp Blue Cacbon

ĐỒ ÁN TỔNG HỢP

KỸ THUẬT ĐIỀU KHIỂN VÀ TỰ ĐỘNG HÓA

Người hướng dẫn
TS Đồng Sỹ Thiên Châu.

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2025

LỜI CẢM ƠN

Em viết lời cảm ơn này để bày tỏ sự biết ơn đến cô TS Đồng Sỹ Thiên Châu vì những kiến thức bổ ích mà các cô đã truyền đạt cho em trong xuyên suốt thời gian em được học tập dưới sự hướng dẫn của các cô.

Cô luôn mang đến các kiến thức mới mẽ, hấp dẫn, lôi cuốn và rất phù hợp với trình độ của em. Nhờ sự chuẩn bị chu đáo của cô, chúng em đã có cơ hộ tiếp nhận và phát triển kiến thức một cách tốt nhất. Dựa và những kiến thức mà các cô đã truyền đạt nên em mới có thể hoàn thiện được đồ án tốt nghiệp và những kỹ năng cần thiết của một sinh viên kỹ thuật cũng từ những kỹ năng này sẽ giúp em phát triển bản thân vào sự phát triển và từng bước hoàn thiện bản thân trong lương lai.

Em viết lời cảm ơn này để bày tỏ sự biết ơn đến cô TS Đồng Sỹ Thiên Châu vì những kiến thức bổ ích mà các cô đã truyền đạt cho em trong xuyên suốt thời gian em được học tập dưới sự hướng dẫn của các cô.

TP. Hồ Chí Minh, ngày 07 tháng 07 năm 2025

Tác giả

(ký tên và ghi rõ họ tên)

Công trình được hoàn thành tại Trường Đại học Tôn Đức Thắng
Cán bộ hướng dẫn khoa học TS Đồng Sỹ Thiên Châu.

Đồ án tốt nghiệp được bảo vệ tại **Hội đồng đánh giá Đồ án tốt nghiệp/tổng hợp của Trường Đại học Tôn Đức Thắng** vào ngày… /…/……

Xác nhận của Chủ tịch Hội đồng đánh giá Đồ án tốt nghiệp và Trưởng khoa quản lý chuyên ngành sau khi nhận Đồ án tốt nghiệp đã được sửa chữa (nếu có).

CHỦ TỊCH HỘI ĐỒNG **TRƯỞNG KHOA**

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của TS Đồng Sỹ Thiên Châu. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Đồ án tốt nghiệp/ tổng hợp còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Đồ án tốt nghiệp/ tổng hợp của mình. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 07 tháng 07 năm 2025

Tác giả

(ký tên và ghi rõ họ tên)

(Trang này dùng để đính kèm Nhiệm vụ Đồ án tốt nghiệp có chữ ký của Giảng viên hướng dẫn)

TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA ĐIỆN –ĐIỆN TỬ

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

LỊCH TRÌNH LÀM ĐỒ ÁN TỔNG HỢP

Họ tên sinh viên: Bùi Quang Huy
.....MSSV: 42001172
Tên đề tài: Thiết kế hệ thống IoT thông minh cho nuôi trồng tảo – Giải pháp Blue Cacbon

Tuần/Ngày	**Khối lượng**		**GVHD ký**
	Đã thực hiện	**Tiếp tục thực hiện**	
Tuần 1	Nhận đề tài đồ án tốt nghiệp.	Tìm hiểu và nghiên cứu về đề tài.	
Tuần 2	Tìm hiểu và nghiên cứu về đề tài.	Tìm hiểu quy trình nuôi tảo.	
Tuần 3	Tìm nguồn cung con giống và chất dinh dưỡng .	Nghiên cứu, tính toán và lên thiết kế hệ thống.	
Tuần 4	Tìm hiểu quy trình nuôi tảo.	Nghiên cứu, tính toán và lên thiết kế hệ thống.	
Tuần 5	Nghiên cứu, tính toán và lên thiết kế hệ thống.	Tìm hiểu các loại cảm biến phù hợp.	
Tuần 6	Tìm hiểu các loại cảm biến phù hợp.	Lựa chọn, thiết kế mô hình nuôi phù hợp.	
Tuần 7	Lựa chọn, thiết kế mô hình nuôi phù hợp.	Lập sơ đồ khối và lưu đồ giải thuật, lập trình	
Kiểm tra giữa kỳ	Đánh giá khối lượng hoàn thành……..% được tiếp tục/không tiếp tục thực hiện		

	ĐATN		
Tuần 8	Lập sơ đồ khối và lưu đồ giải thuật, lập trình	Thiết kế board mạch.	
Tuần 9	Thiết kế board mạch.	Thiết kế board mạch.	
Tuần 10	Thiết kế board mạch.	Chạy thử nghiệm hệ thống	
Tuần 11	Chạy thử nghiệm hệ thống	Chạy thử nghiệm hệ thống	
Tuần 12	Chạy thử nghiệm hệ thống	Phân tích, đánh giá kết quả thu được.	
Tuần 13	Phân tích, đánh giá kết quả thu được.	Viết báo cáo	
Tuần 14	Viết báo cáo	Viết báo cáo.	
Tuần 15	Viết báo cáo	Viết báo cáo	
Nộp Đồ án tốt nghiệp	Đã hoàn thành……..% Đồ án tốt nghiệp được bảo vệ/không được bảo vệ ĐATN		

THIẾT KẾ HỆ THỐNG IOT THÔNG MINH CHO NUÔI TRỒNG TẢO – GIẢI PHÁP BLUE CARBON

TÓM TẮT

Trong bối cảnh biến đổi khí hậu và ô nhiễm môi trường ngày càng nghiêm trọng, tảo đang được xem là một giải pháp sinh học tiềm năng giúp hấp thụ CO_2 và đóng góp vào quá trình giảm phát thải khí nhà kính. Từ định hướng đó, đề tài được xây dựng nhằm hỗ trợ việc theo dõi và quản lý quá trình nuôi trồng tảo một cách hiệu quả, tự động và tối ưu hơn.

Hệ thống được thiết kế tích hợp các cảm biến môi trường như cảm biến đo chất rắn hòa tan (TDS), cảm biến nhiệt độ, cảm biến pH và cảm biến oxy hòa tan (DO) nhằm thu thập các thông số quan trọng ảnh hưởng trực tiếp đến sự phát triển của tảo. Dữ liệu từ các cảm biến này sẽ được thu thập và truyền về vi điều khiển trung tâm (bằng ESP32 và Raspberry Pi), sau đó lưu trữ vào cơ sở dữ liệu để phục vụ phân tích và điều khiển.

Đặc biệt, hệ thống còn ứng dụng công nghệ xử lý hình ảnh thông minh từ camera và mô hình học sâu (Deep Learning) để nhận diện và phân tích mật độ tảo trong bể nuôi theo thời gian thực. Từ đó, hệ thống có thể phát hiện các dấu hiệu bất giúp người vận hành đưa ra các quyết định chính xác và kịp thời.

Kết quả của đề tài hứa hẹn sẽ góp phần vào việc hiện đại hóa hoạt động nuôi tảo theo hướng thông minh và bền vững, đồng thời đóng góp vào chiến lược phát triển kinh tế xanh – giảm phát thải carbon thông qua giải pháp Blue Carbon.

SMART IOT SYSTEM DESIGN FOR ALGAE FARMING – BLUE CARBON SOLUTION

ABSTRACT

In the context of increasingly severe climate change and environmental pollution, algae are being regarded as a promising biological solution for absorbing CO_2 and contributing to the reduction of greenhouse gas emissions. From that orientation, this project is developed to support the monitoring and management of algae cultivation in a more efficient, automated, and optimized manner.

The system is designed to integrate environmental sensors such as Total Dissolved Solids (TDS) sensor, temperature sensor, pH sensor, and Dissolved Oxygen (DO) sensor to collect key parameters directly affecting algae growth. Data from these sensors is gathered and transmitted to a central microcontroller (using ESP32 and Raspberry Pi), then stored in a database for analysis and control purposes.

Notably, the system also applies intelligent image processing technology using cameras and deep learning models to detect and analyze algae density in real-time. This enables the system to identify abnormal signs, helping operators make timely and accurate decisions.

The results of this project are expected to contribute to the modernization of algae farming in a smart and sustainable direction, while also supporting the green economic development strategy by reducing carbon emissions through the Blue Carbon solution.

MỤC LỤC

DANH MỤC BẢNG BIỂU

DANH MỤC HÌNH VẼ

DANH MỤC CÁC CHỮ VIẾT TẮT

TDS	Total Dissolved Solid
DO	Dissolved Oxygen
pH	Potentia Hydrogeni
IoT	Internet of Things

CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI

1.1 Giới thiệu về đề tài

Trong bối cảnh biến đổi khí hậu và ô nhiễm môi trường đang là những thách thức toàn cầu, các giải pháp sinh học thân thiện với môi trường ngày càng được chú trọng. Trong đó, tảo được xem là một sinh vật có tiềm năng lớn nhờ khả năng hấp thụ CO_2 nhanh, hiệu quả và khả năng thích nghi tốt với nhiều điều kiện sống. Nuôi trồng tảo không chỉ góp phần làm sạch môi trường mà còn mang lại giá trị kinh tế cao thông qua các sản phẩm như nhiên liệu sinh học, thực phẩm chức năng, dược phẩm và nguyên liệu công nghiệp.

Tuy nhiên, để đạt hiệu quả cao trong quá trình nuôi trồng tảo, cần có sự giám sát chặt chẽ và điều chỉnh linh hoạt các điều kiện môi trường như nhiệt độ, độ pH, nồng độ oxy hòa tan, hàm lượng chất rắn trong nước,... Việc áp dụng công nghệ Internet vạn vật (IoT) vào trong nông nghiệp – cụ thể là nuôi trồng tảo – là xu hướng tất yếu nhằm nâng cao tính tự động hóa, giảm chi phí lao động và tăng hiệu quả sản xuất.

Từ thực tiễn đó, đề tài "Thiết kế hệ thống IoT thông minh cho nuôi trồng tảo – Giải pháp Blue Carbon" được thực hiện với mục tiêu xây dựng một hệ thống giám sát và quản lý ao nuôi tảo theo thời gian thực. Hệ thống được tích hợp các cảm biến như cảm biến đo tổng chất rắn hòa tan (TDS), cảm biến nhiệt độ, cảm biến pH và cảm biến oxy hòa tan (DO), giúp thu thập các thông số môi trường quan trọng. Các dữ liệu này sẽ được vi điều khiển ESP32 và Raspberry Pi xử lý, lưu trữ vào cơ sở dữ liệu để phân tích và đưa ra cảnh báo hoặc tự động điều chỉnh khi cần thiết.

Ngoài ra, điểm nổi bật của hệ thống là khả năng ứng dụng camera và mô hình học sâu (Deep Learning) để xử lý hình ảnh bề mặt nước nuôi, từ đó phân tích mật độ tảo một cách thông minh. Tính năng này giúp phát hiện sớm các hiện tượng bất thường như tảo phát triển quá mức hoặc suy giảm mật độ sinh khối, giúp người vận hành kịp thời đưa ra các biện pháp xử lý hiệu quả.

Đề tài không chỉ mang lại ý nghĩa về mặt kỹ thuật và công nghệ, mà còn thể hiện rõ định hướng phát triển bền vững, xanh hóa sản xuất và đóng góp vào chiến lược giảm phát thải khí nhà kính. Giải pháp Blue Carbon mà đề tài hướng đến là một phần của xu hướng kinh tế xanh, sử dụng sinh khối thực vật dưới nước để hấp thụ và lưu trữ carbon, tạo ra giá trị môi trường và kinh tế đồng thời.

1.2 Mục đích nghiên cứu

Mục tiêu chính của đề tài là xây dựng một hệ thống giám sát và quản lý thông minh áp dụng công nghệ IoT và trí tuệ nhân tạo (AI) vào quá trình nuôi trồng tảo nhằm nâng cao hiệu quả sản xuất, đồng thời hướng đến mục tiêu phát triển bền vững và bảo vệ môi trường.

Hiện nay, nuôi trồng tảo không chỉ mang lại giá trị kinh tế trong các lĩnh vực như thực phẩm chức năng, dược phẩm, nhiên liệu sinh học,… mà còn được xem là giải pháp hấp thụ carbon hiệu quả – một phần của mô hình Blue Carbon, tức là sử dụng sinh khối thủy sinh để lưu trữ và giảm phát thải khí nhà kính CO_2. Tuy nhiên, việc nuôi tảo theo phương pháp truyền thống vẫn gặp nhiều hạn chế như phụ thuộc vào kinh nghiệm con người, thiếu khả năng giám sát liên tục và khó phát hiện kịp thời các yếu tố bất thường trong môi trường nước.

Từ thực tế, đề tài đặt ra các mục đích nghiên cứu cụ thể như sau:

- Thiết kế và triển khai một hệ thống IoT tích hợp cảm biến môi trường (bao gồm cảm biến nhiệt độ, pH, DO – oxy hòa tan, và TDS – tổng chất rắn hòa tan) nhằm thu thập dữ liệu thời gian thực từ môi trường nuôi trồng tảo. Hệ thống giúp giám sát liên tục các thông số ảnh hưởng trực tiếp đến sự phát triển của tảo.
- Xây dựng cơ sở dữ liệu và nền tảng lưu trữ cho phép người dùng theo dõi, truy xuất và phân tích dữ liệu từ xa thông qua vi điều khiển (ESP32 hoặc Raspberry Pi) kết nối với mạng Wi-Fi hoặc Internet.

- Ứng dụng xử lý ảnh và mô hình học sâu (Deep Learning) nhằm phân tích hình ảnh bề mặt bể tảo từ camera, phục vụ mục tiêu đánh giá mật độ tảo theo thời gian thực. Mục tiêu là có thể phát hiện sớm các hiện tượng như tảo phát triển quá mức (bloom), suy giảm mật độ sinh khối, hoặc sự thay đổi màu sắc bất thường – từ đó giúp người vận hành có thể xử lý kịp thời.
- Tối ưu hóa điều kiện nuôi tảo và tăng khả năng tự động hóa trong sản xuất bằng cách kết hợp giữa cảm biến và AI để ra quyết định hoặc cảnh báo tự động, từ đó giảm thiểu chi phí vận hành và nâng cao hiệu quả sản xuất.
- Góp phần vào xu hướng sản xuất xanh và phát triển bền vững, hỗ trợ mục tiêu giảm phát thải khí nhà kính thông qua giải pháp Blue Carbon – sử dụng sinh khối tảo để hấp thụ CO_2 một cách tự nhiên và hiệu quả.

Đề tài hướng đến việc phát triển một mô hình ứng dụng thực tiễn cao, có khả năng mở rộng và triển khai trong các hệ thống nuôi trồng tảo quy mô vừa và nhỏ, giúp hiện đại hóa sản xuất nông nghiệp – thủy sản theo hướng thông minh, bền vững và thân thiện với môi trường.

1.3 Đối tượng nghiên cứu

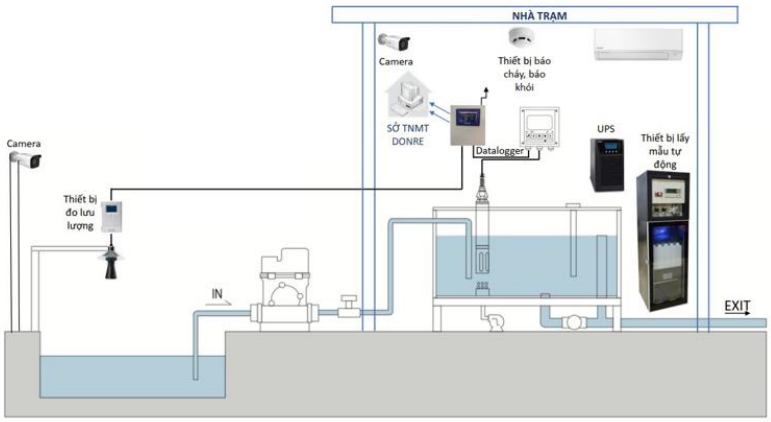
- **Môi trường nước nuôi tảo**: Các yếu tố cần giám sát bao gồm nhiệt độ, pH, nồng độ oxy hòa tan (DO), chất rắn hòa tan (TDS), và cường độ ánh sáng. Môi trường nước cần đảm bảo độ kiềm nhẹ (pH từ 8.5–9.5), nhiệt độ ổn định từ 30–35°C, oxy hòa tan ≥ 5 mg/L, TDS trong khoảng 1000–3000 ppm và đủ ánh sáng (2000–5000 lux) để đảm bảo hiệu suất quang hợp cho tảo. Những điều kiện này ảnh hưởng trực tiếp đến tốc độ sinh trưởng và mật độ sinh khối của vi tảo.

- **Vi tảo nghiên cứu – Spirulina:** Tảo xoắn xanh lam (Spirulina platensis) được chọn là đối tượng chính nhờ tốc độ phát triển nhanh, dễ nuôi, giàu protein (55–70%), và có khả năng hấp thụ CO_2 cao (~1.8–2.0 kg CO_2/kg sinh

khối). Ngoài ra, Spirulina còn phù hợp với môi trường nước có tính kiềm và thích hợp cho nuôi trồng ở bể hở. Các loài tảo khác như Chlorella hay Scenedesmus cũng có tiềm năng trong các ứng dụng Blue Carbon nhưng không phải trọng tâm của đề tài này.

Hình 1-1 Viên tảo sau khi thu hoạch

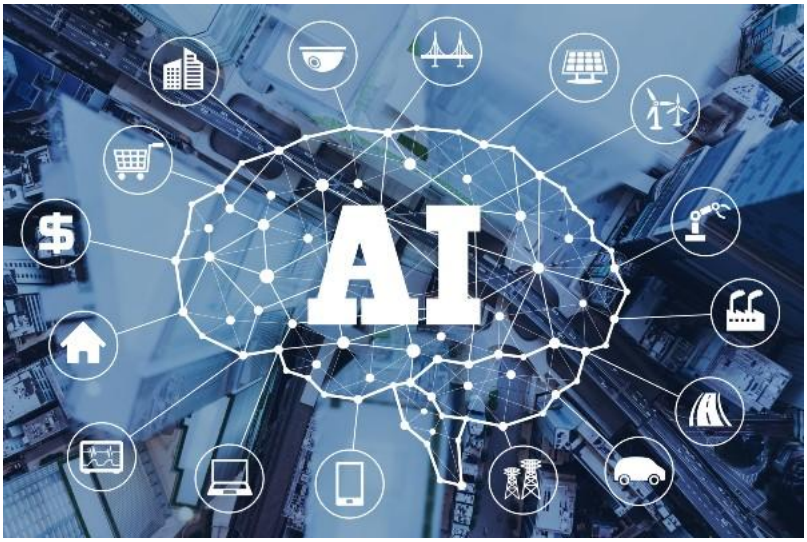
- **Hệ thống giám sát môi trường:** Bao gồm các cảm biến đo nhiệt độ , pH, DO, và TDS; kết nối với vi điều khiển ESP32 để thu thập dữ liệu thời gian thực. Dữ liệu được xử lý, lưu trữ và hiển thị qua Raspberry Pi với dashboard giám sát trực tuyến. Hệ thống cho phép người dùng giám sát từ xa và phát hiện sớm sự thay đổi bất thường trong môi trường nước.



Hình 1-2 Hệ thống giám sát

- **Ứng dụng trí tuệ nhân tạo (AI):** Camera gắn trong hệ thống (ESP32-CAM hoặc webcam) chụp ảnh bề mặt nước nuôi. Ảnh được xử lý bằng các thuật

toán học sâu (CNN, YOLO, hoặc OpenCV) để phân tích mật độ tảo, phát hiện các hiện tượng như tảo nở hoa, suy giảm sinh khối hay nổi váng. AI hỗ trợ cảnh báo sớm và ra quyết định kịp thời.



Hình 1-3 Mô hình học máy

- **Thiết bị và công nghệ hỗ trợ:** Sử dụng ESP32, Raspberry Pi 4, module Wi-Fi, mạch nguồn, LED chiếu sáng bổ sung, hệ thống truyền dữ liệu MQTT hoặc HTTP, và cơ sở dữ liệu (Firebase/MySQL). Hệ thống được thiết kế tối ưu cho hoạt động liên tục, tiêu thụ điện năng thấp và khả năng mở rộng linh hoạt.

- **Các yếu tố kinh tế và môi trường:** Hệ thống giúp giảm chi phí vận hành, tiết kiệm nhân công, tăng hiệu quả sản xuất sinh khối. Bên cạnh đó, việc nuôi tảo có khả năng hấp thụ CO_2 và tạo ra sinh khối hữu ích góp phần giảm phát thải khí nhà kính, thúc đẩy mô hình Blue Carbon và kinh tế tuần hoàn hướng đến phát triển bền vững.

1.4 Phạm vi nghiên cứu

- **Phạm vi không gian:**

Đề tài được thực hiện trong quy mô phòng thí nghiệm hoặc mô hình bể nuôi nhỏ tại khu vực giới hạn (dưới 10 m^2), sử dụng bể nhựa hoặc khung kính có thể tích từ 50–200 lít nước để nuôi trồng tảo Spirulina. Việc triển khai thử nghiệm và kiểm tra hệ thống được tiến hành trong điều kiện bán ngoài trời hoặc có chiếu sáng nhân tạo để đảm bảo tính kiểm soát.

Phạm vi kỹ thuật:

Hệ thống được thiết kế theo mô hình IoT cơ bản, sử dụng các cảm biến môi trường (nhiệt độ, pH, DO, TDS), vi điều khiển ESP32, máy tính nhúng Raspberry Pi, và camera giám sát. Phần mềm quản lý sử dụng giao thức Wi-Fi, MQTT, và nền tảng lưu trữ như Firebase hoặc MySQL. Mô hình AI xử lý ảnh chỉ áp dụng cho việc phân tích mật độ tảo qua ảnh tĩnh hoặc ảnh chụp định kỳ, chưa xử lý video liên tục theo thời gian thực.

Phạm vi nghiên cứu chuyên môn:

Nghiên cứu tập trung vào giám sát và đánh giá điều kiện môi trường ảnh hưởng đến sự phát triển của tảo Spirulina, kết hợp xử lý tín hiệu cảm biến, phân tích hình ảnh bằng AI, và xây dựng dashboard hiển thị dữ liệu. Các nội dung sinh học như giống tảo, quy trình nuôi trồng, hay phân tích hóa sinh học sinh khối không phải là trọng tâm, mà chỉ được sử dụng ở mức tham khảo hỗ trợ cho thiết kế hệ thống.

Phạm vi ứng dụng:

Hệ thống được xây dựng thử nghiệm với quy mô nhỏ, phục vụ cho mục tiêu mô phỏng và chứng minh nguyên lý, chưa hướng đến ứng dụng đại trà hoặc triển khai thương mại hóa. Tuy nhiên, mô hình có thể được phát triển mở rộng cho các hệ thống nuôi tảo công nghiệp, đặc biệt trong các trại nuôi tảo quy mô vừa – nhỏ ứng dụng vào xử lý môi trường, hấp thụ CO_2, sản xuất sinh khối xanh, hoặc phục vụ nền kinh tế tuần hoàn – xanh – carbon thấp.

Dự kiến kết quả.

Đề tài dự kiến xây dựng thành công một hệ thống giám sát thông minh dành cho nuôi trồng tảo Spirulina, ứng dụng công nghệ IoT kết hợp với xử lý ảnh bằng trí tuệ nhân tạo. Hệ thống có khả năng thu thập, truyền và lưu trữ dữ liệu môi trường nước theo thời gian thực từ các cảm biến đo nhiệt độ, pH, DO và TDS. Đồng thời, hệ thống sử dụng camera và mô hình AI để phân tích mật độ tảo, hỗ trợ cảnh báo sớm khi có hiện tượng bất thường trong quá trình nuôi trồng.

Kết quả mong đợi là tạo ra một mô hình hoạt động ổn định, chi phí thấp, dễ mở rộng, có thể ứng dụng trong quy mô nhỏ hoặc tiền sản xuất. Ngoài ra, hệ thống sẽ cung cấp dữ liệu đầu vào có giá trị cho việc đánh giá hiệu quả hấp thụ CO_2 của tảo, từ đó góp phần định lượng tiềm năng Blue Carbon trong các mô hình nông nghiệp – sinh học bền vững.

Thông qua đó, đề tài không chỉ đóng vai trò hỗ trợ kỹ thuật trong quá trình nuôi tảo mà còn góp phần thúc đẩy xu hướng phát triển nông nghiệp thông minh, phục vụ chiến lược giảm phát thải carbon và phát triển kinh tế xanh trong tương lai.

CHƯƠNG 2. CÁC NGHIÊN CỨU LIÊN QUAN VÀ CƠ SỞ LÝ THUYẾT

2.1 Các nghiên cứu liên quan

Trong thời gian gần đây, các nghiên cứu ứng dụng công nghệ Internet vạn vật (IoT) và trí tuệ nhân tạo trong lĩnh vực nuôi trồng thủy sinh, đặc biệt là vi tảo, đã thu hút được nhiều sự quan tâm. Một trong những nghiên cứu đáng chú ý là của S. Vasisht và cộng sự (2019) với đề tài "IoT-Based Monitoring System for Algae Cultivation". Nhóm tác giả đã thiết kế một hệ thống sử dụng Arduino để đo pH và nhiệt độ, truyền dữ liệu qua Wi-Fi và hiển thị trên giao diện đơn giản. Tuy nhiên, nghiên cứu này mới chỉ tập trung vào phần cảm biến cơ bản, chưa có khả năng xử lý ảnh, lưu trữ dữ liệu lâu dài hay tích hợp phân tích thông minh.

Tại Việt Nam, Nguyễn Văn Lợi và cộng sự (2020) đã thực hiện đề tài "Thiết kế hệ thống theo dõi pH và nhiệt độ trong nuôi tảo Spirulina" tại Trường Đại học Bách Khoa TP.HCM. Hệ thống có khả năng theo dõi thông số môi trường thời gian thực và hiển thị trên website nội bộ. Mặc dù có tính thực tiễn cao, nhưng hệ thống vẫn còn thủ công trong điều khiển, thiếu khả năng xử lý ảnh hoặc đưa ra cảnh báo sớm dựa trên dữ liệu.

Ở hướng nghiên cứu xử lý ảnh, Y. Kose và cộng sự (2021) đã triển khai đề tài "AI-Powered Algae Detection Using Convolutional Neural Network", sử dụng mô hình học sâu (CNN) để phân loại các loại tảo dựa trên ảnh hiển vi. Đây là nghiên cứu có giá trị về mặt phân tích hình ảnh vi sinh học, song chưa gắn kết với hệ thống giám sát môi trường thực tế hay điều khiển nuôi trồng. Hơn nữa, việc sử dụng ảnh hiển vi chưa phù hợp với điều kiện ngoài trời hoặc quy mô sản xuất.

Ngoài ra, một số doanh nghiệp thương mại như AlgaeConnect (Hoa Kỳ, 2022) đã phát triển các hệ thống theo dõi môi trường bể nuôi tảo thông qua các cảm biến IoT

và nền tảng điều khiển từ xa. Tuy nhiên, các hệ thống này thường có chi phí cao, không mã nguồn mở, thiếu tính linh hoạt trong việc tùy chỉnh và mở rộng, và chưa tích hợp công nghệ xử lý ảnh bằng AI ngay tại thiết bị đầu cuối.

Từ các nghiên cứu nêu trên có thể thấy, phần lớn các hệ thống hiện tại vẫn đang phát triển độc lập theo từng hướng – hoặc là đo lường thông số môi trường, hoặc là xử lý ảnh, hoặc là điều khiển từ xa – mà chưa có nhiều mô hình tích hợp đồng bộ cả ba yếu tố quan trọng: IoT – xử lý ảnh AI – điều khiển thông minh. Ngoài ra, rất ít nghiên cứu đề cập đến vai trò của vi tảo trong hấp thụ CO_2 và định lượng hiệu quả sinh khối tảo phục vụ định hướng Blue Carbon. Chính vì vậy, đề tài này được thực hiện nhằm xây dựng một hệ thống thông minh, tích hợp các chức năng giám sát môi trường, phân tích mật độ tảo và cảnh báo bất thường, hướng đến một mô hình nuôi tảo hiệu quả – bền vững – hiện đại, góp phần vào chiến lược phát triển kinh tế xanh và giảm phát thải carbon.

2.2 Cơ sở lý thuyết

2.2.1 Cơ sở lý thuyết về hệ thống quan trắc

Hệ thống quan trắc môi trường trong mô hình nuôi tảo spirulina có vai trò thu thập và giám sát liên tục các thông số ảnh hưởng trực tiếp đến quá trình phát triển của tảo. Các thông số quan trọng bao gồm:

- Độ pH: Ảnh hưởng đến khả năng hấp thu dưỡng chất và tốc độ quang hợp của tảo. Spirulina phát triển tốt trong môi trường có pH từ 8.0 đến 11.0
- Nhiệt độ: Là yếu tố quyết định đến khả năng sinh trưởng và năng suất. Nhiệt độ lý tưởng trong khoảng 27 - 30°C.
- TDS (Total Dissolved Solids): Phản ánh nồng độ dưỡng chất hòa tan trong môi trường nuôi. Sau mỗi đợt thu hoạch, chỉ số này thường giảm, cần được theo dõi để điều chỉnh lượng dưỡng chất bổ sung.

- DO (Dissolved Oxygen): Là chỉ số đánh giá mức độ quang hợp và sức khỏe tổng thể của hệ sinh thái vi tảo. DO giảm mạnh có thể cảnh báo tình trạng phân hủy hữu cơ hoặc suy giảm tảo.

Để đo các thông số này, hệ thống sử dụng các cảm biến chuyên dụng:

- DO-S20: cảm biến tích hợp đo cùng lúc pH, DO và nhiệt độ, truyền dữ liệu qua giao thức RS485, có vỏ chống ăn mòn phù hợp cho môi trường nước.
- Cảm biến TDS (DFRobot Gravity): đo nồng độ chất rắn hòa tan, truyền dữ liệu qua UART.

Vi điều khiển ESP32 là trung tâm xử lý dữ liệu. Nó nhận tín hiệu từ các cảm biến, xử lý, hiển thị qua giao diện, đồng thời gửi lên cơ sở dữ liệu để lưu trữ và phân tích.

a) . Cảm biến DO S20 sensor

Trong hệ thống giám sát môi trường nước nuôi tảo, việc đo lường đồng thời các chỉ tiêu quan trọng như oxy hòa tan (DO), độ pH và nhiệt độ là thiết yếu để đảm bảo điều kiện sinh trưởng tối ưu cho vi tảo. Đề tài sử dụng cảm biến tích hợp DO-S20, là loại cảm biến đa chức năng có khả năng đo đồng thời ba thông số: DO, pH và nhiệt độ – giúp đơn giản hóa cấu trúc phần cứng, tiết kiệm chi phí và dễ dàng bảo trì hệ thống.

Hình 2-1 Cảm biến DO-so20

Cảm biến hoạt động dựa trên nguyên lý đo quang học không điện cực đối với chỉ tiêu DO, sử dụng hiện tượng dập tắt huỳnh quang để xác định nồng độ oxy hòa tan trong nước. Khi ánh sáng từ bộ phát chiếu vào điểm cảm biến (có chứa thuốc nhuộm huỳnh quang), oxy hòa tan trong môi trường sẽ ảnh hưởng đến tuổi thọ và cường độ phát quang của tín hiệu trả về. Thiết bị đo được sự thay đổi này và tính toán chính xác nồng độ DO mà không cần tiếp xúc điện cực, nhờ đó tránh hiện tượng ăn mòn, bám bẩn, và không cần hiệu chuẩn thường xuyên như cảm biến điện hóa.

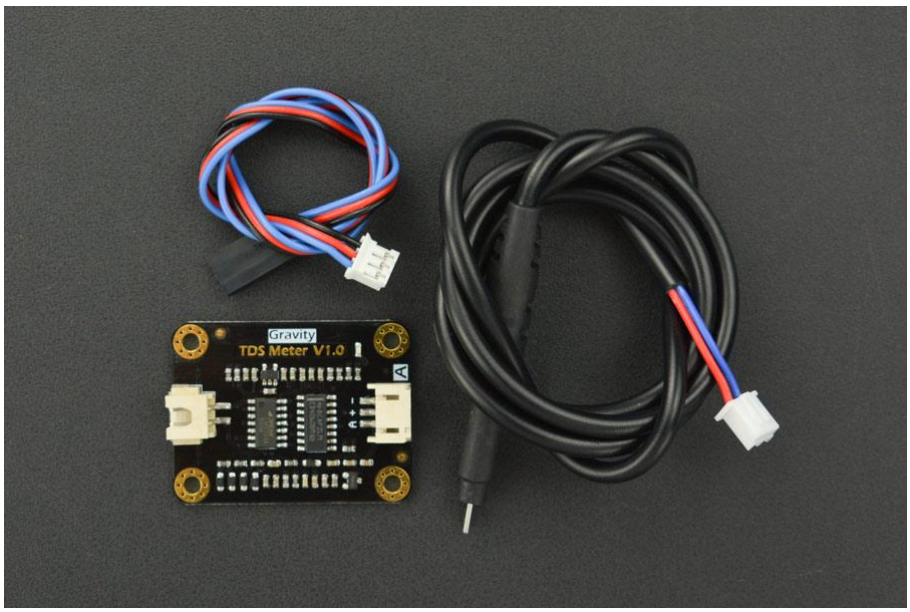
Đối với phép đo pH, cảm biến sử dụng công nghệ điện cực ion chọn lọc để xác định độ axit/bazơ trong môi trường nước – yếu tố ảnh hưởng lớn đến hoạt động sinh học của tảo và hiệu quả quang hợp. Nhiệt độ cũng được tích hợp để bù nhiệt cho hai phép đo còn lại, đảm bảo độ chính xác cao trong mọi điều kiện môi trường.

Cảm biến DO-S20 hỗ trợ giao tiếp RS485 – Modbus RTU, phù hợp với các hệ thống nhúng như ESP32 hoặc Raspberry Pi. Dữ liệu từ cảm biến được cập nhật theo

thời gian thực và truyền về trung tâm xử lý để hiển thị trên giao diện giám sát (InfluxDB hoặc Node-RED). Với độ chính xác cao, độ bền tốt và khả năng tích hợp đa chỉ tiêu, cảm biến DO-S20 là giải pháp tối ưu trong các mô hình nuôi tảo thông minh, giúp nâng cao hiệu quả vận hành và khả năng cảnh báo sớm.

b) Cảm biến TDS

TDS (Total Dissolved Solids) là chỉ số đo tổng lượng các chất rắn hòa tan trong nước, bao gồm khoáng chất, muối, kim loại và các hợp chất hữu cơ – những thành phần ảnh hưởng trực tiếp đến độ tinh khiết và chất lượng môi trường nước trong bể nuôi tảo. Trong quá trình nuôi trồng, sự tích tụ của dưỡng chất, phân tử hữu cơ hoặc tạp chất sẽ làm thay đổi nồng độ TDS, gây ảnh hưởng đến quá trình hấp thu chất dinh dưỡng, quang hợp và sinh trưởng của tảo.



Hình 2-2 Cảm biến TDS

Đề tài sử dụng cảm biến TDS loại Gravity TDS Sensor của hãng DFRobot, một thiết bị phổ biến trong các ứng dụng IoT, thủy canh và hệ thống tự động hóa môi trường nước. Cảm biến này hoạt động dựa trên nguyên lý đo điện trở suất (electrical conductivity – EC) của nước: khi các chất rắn hòa tan tăng lên, độ dẫn điện của

nước cũng tăng theo. Từ giá trị đo EC, hệ thống vi điều khiển (ESP32) sẽ tính toán và suy ra nồng độ TDS (theo ppm – parts per million).

Cảm biến được thiết kế dạng đầu dò điện cực đôi không phân cực, có khả năng đo trong dải 0–1000 ppm, điện áp hoạt động 3.3V–5V, tín hiệu ngõ ra dạng analog. Thiết bị tích hợp sẵn mạch khuếch đại tín hiệu và bộ lọc nhiễu, giúp tăng độ chính xác khi đo trong các môi trường có độ ẩm cao hoặc nhiều nhiễu điện.

Trong hệ thống đề tài, cảm biến TDS được kết nối với vi điều khiển ESP32 thông qua chân ADC (analog). Dữ liệu đo được xử lý tại vi điều khiển, truyền lên hệ thống lưu trữ (InfluxDB), hiển thị trên giao diện giám sát (Node-RED) và dùng làm cơ sở để điều chỉnh nồng độ dưỡng chất khi cần thiết. Nhờ đó, người vận hành có thể đánh giá và duy trì mức TDS tối ưu cho sinh trưởng của tảo, đồng thời phát hiện các bất thường như thừa hoặc thiếu chất dinh dưỡng trong bể nuôi.

2.2.2 Cơ sở lý thuyết về hệ thống bổ sung dưỡng chất

Trong hệ thống nuôi trồng tảo, đặc biệt là Spirulina – một loài vi tảo giàu protein và có khả năng hấp thụ CO_2 cao, môi trường nước cần đảm bảo đủ các dưỡng chất thiết yếu để duy trì khả năng quang hợp, sinh trưởng và nhân sinh khối. Việc bổ sung dinh dưỡng đúng cách là yếu tố then chốt để đạt năng suất cao và đảm bảo sự ổn định của hệ sinh thái nuôi.

a) Thành phần dưỡng chất thiết yếu cho tảo

Các nguyên tố cần thiết trong môi trường nuôi tảo được chia làm ba nhóm chính:

- Nguyên tố đa lượng (Macronutrients):
  - Nitơ (N): Cần thiết cho tổng hợp protein và chlorophyll.
  - Photpho (P): Tham gia vào năng lượng tế bào (ATP), DNA, RNA.
  - Kali (K): Điều hòa áp suất thẩm thấu, hoạt hóa enzyme.
- Nguyên tố trung lượng:
  - Magie (Mg^{2+}): Thành phần cấu tạo của chlorophyll.
  - Canxi (Ca^{2+}): Ổn định màng tế bào, hỗ trợ chuyển hóa.

- Lưu huỳnh (S): Cấu trúc axit amin và protein.

• Nguyên tố vi lượng (Micronutrients):

- Bao gồm: Sắt (Fe), Kẽm (Zn), Mangan (Mn), Đồng (Cu), Molypden (Mo),... có vai trò xúc tác và chuyển hóa enzyme.

Các dưỡng chất này thường được pha trộn theo công thức chuẩn và hòa tan trong nước để tạo thành dung dịch bổ sung định kỳ cho bể tảo.

b) Nguyên lý hoạt động của hệ thống bổ sung

Hệ thống sử dụng các cảm biến để theo dõi thông số môi trường:

- Cảm biến pH và TDS được gắn trực tiếp vào bể nuôi để đo nồng độ ion và độ kiềm.
- Khi các chỉ số giảm xuống dưới ngưỡng cho phép (ví dụ: TDS thấp hoặc pH lệch), bộ điều khiển ESP32 sẽ kích hoạt bơm điện tử mini để bổ sung dung dịch dinh dưỡng đã được pha sẵn.
- Quy trình này có thể hoạt động bán tự động hoặc hoàn toàn tự động nhờ thiết lập điều kiện logic trên nền tảng Node-RED.

c) Vai trò và lợi ích của hệ thống bổ sung thông minh

- Giúp tảo phát triển ổn định, hạn chế hiện tượng chết do thiếu dinh dưỡng.
- Tối ưu chi phí vận hành bằng cách định lượng hợp lý, tránh lãng phí.
- Tăng khả năng quang hợp → cải thiện hiệu suất hấp thụ CO_2.
- Hạn chế sai sót thủ công, nhờ giám sát và tự động hóa qua cảm biến

2.2.3 Cơ sở lý thuyết về trí tuệ nhân tạo và học máy trong giám sát môi trường

a) Khái quát về trí tuệ nhân tạo và học máy

Trí tuệ nhân tạo (Artificial Intelligence – AI) là lĩnh vực nghiên cứu và phát triển các hệ thống có khả năng thực hiện những nhiệm vụ mà trước đây chỉ con người mới làm được như suy luận, học hỏi, phân tích và ra quyết định. Một nhánh quan trọng của AI là học máy (Machine Learning – ML), cho phép các hệ thống học từ dữ liệu quá khứ để đưa ra dự đoán hoặc phân loại dữ liệu mới.

Học máy không cần được lập trình chi tiết cho từng trường hợp cụ thể, mà thay vào đó sử dụng mô hình toán học để học các quy luật tiềm ẩn trong dữ liệu. Quá trình học bao gồm:

- Huấn luyện (training): mô hình học từ dữ liệu đầu vào và đầu ra tương ứng.
- Kiểm thử (validation): đánh giá độ chính xác của mô hình.
- Dự đoán (inference): áp dụng mô hình cho dữ liệu mới.

b) Một số khái niệm và công thức cơ bản

- Hàm mất mát (Loss Function): là tiêu chí để đo lường sai số giữa đầu ra dự đoán và giá trị thực tế. Mục tiêu của học máy là tối thiểu hóa hàm mất mát. Một ví dụ phổ biến là Mean Squared Error (MSE):

$$\mathcal{L} = \frac{1}{n}\sum_{i=0}^{n} (yi - \widehat{Y}_i)^2$$

[1]

Trong đó:

- Y_i: giá trị thực tế (nhãn).
- $\widehat{Y}_i$: giá trị mô hình dự đoán.
- n: số lượng mẫu.

- Hàm ReLU (viết tắt của Rectified Linear Unit) là một hàm phi tuyến đơn giản và phổ biến nhất hiện nay trong các mô hình học sâu (Deep Learning), đặc biệt là mạng nơ-ron tích chập (CNN).

  Nó được định nghĩa bởi công thức:

$$f(x) = \begin{cases} 0, nếu\ x < 0 \\ x, nếu\ x \geq 0 \end{cases} hay\ gọn\ hơn: f(x) = max(0, x)$$

[1]

=> ReLU hoạt động như một bộ lọc thông tin, giúp mạng học được các đặc trưng quan trọng bằng cách bỏ qua (zero out) các giá trị không cần thiết (âm), trong khi giữ lại (linear) các giá trị có ý nghĩa.

- Hàm Softmax (phân loại đa lớp): dùng ở tầng đầu ra để đưa về xác suất:

$$P(y_i = k) = \frac{e^{zk}}{\sum_{i=1}^{K} e^{zj}}$$ [2]

Trong đó:

- $P(y_i = K)$: là xác suất mẫu dữ liệu i thuộc lớp thứ k.
- Zk: là đầu ra (score, logit) chưa chuẩn hóa của mô hình tương ứng với lớp k.
- K: là tổng số lớp cần phân loại.
- $\sum_{i=1}^{K} e^{zj}$: là tổng của tất cả các giá trị mũ hóa để chuẩn hóa thành xác suất (tổng này luôn ≥ 1).

c) Ứng dụng trong hệ thống giám sát nuôi tảo

Trong đề tài này, học máy được ứng dụng vào hai nhóm nhiệm vụ chính, kết hợp với dữ liệu thu thập từ camera và cảm biến môi trường:

- Phát hiện lỗi hệ thống (hỏng bơm sục khí):

Trong hệ thống giám sát nuôi tảo, ESP32-S3-WROOM-1U đóng vai trò là vi điều khiển trung tâm. Thiết bị thu thập dữ liệu từ cảm biến TDS (analog) và DO-S20 (qua RS485 → UART), sau đó xử lý tín hiệu và so sánh với ngưỡng cảnh báo. Khi cần thiết, ESP32 điều khiển các thiết bị chấp hành như relay bơm sục khí hoặc bơm dưỡng chất. Đồng thời, dữ liệu được truyền về Raspberry Pi 4 thông qua giao thức MQTT, giúp hệ thống giám sát hoạt động tự động và theo thời gian thực.

- Dự đoán sức khỏe của tảo:

Bên cạnh việc phát hiện bọt khí, hệ thống còn phân tích hình ảnh để nhận diện sự thay đổi màu sắc của nước và mức độ trong suốt của bề mặt bể nuôi. Các yếu tố này là chỉ báo quan trọng phản ánh mật độ sinh khối, tình trạng phát triển của tảo hoặc các dấu hiệu suy yếu như thiếu dưỡng chất, mất cân bằng môi trường. Dựa trên kết quả phân tích, hệ thống có thể đưa ra các gợi ý điều chỉnh như: bổ sung thêm dưỡng chất, thay đổi cường độ chiếu sáng, hoặc tăng thời lượng sục khí nhằm ổn định điều

kiện nuôi và tối ưu hóa tốc độ sinh trưởng của tảo. Việc kết hợp dữ liệu cảm biến và ảnh trực quan giúp nâng cao tính chính xác trong đánh giá và ra quyết định tự động.

2.3 CÁC THIẾT BỊ CHÍNH TRONG MẠCH ĐIỀU KHIỂN

2.3.1 ESP32-S3-WROOM-1U

ESP32-S3-WROOM-1U là một module vi điều khiển tích hợp hiệu năng cao của hãng Espressif Systems, được xây dựng dựa trên SoC ESP32-S3. Module này hỗ trợ đầy đủ kết nối không dây Wi-Fi 802.11 b/g/n và Bluetooth 5 (LE), tích hợp sẵn các chức năng tăng tốc AI thông qua vector instructions, và đặc biệt là có thiết kế không tích hợp ăng-ten PCB, thay vào đó sử dụng ăng-ten ngoài qua đầu nối u.FL – phù hợp với môi trường công nghiệp có yêu cầu tín hiệu ổn định hoặc đặt thiết bị trong vỏ kim loại.

Hình 2-3 Vi điều khiển ESP32

Thông số kỹ thuật nổi bật:

- CPU: Dual-core Xtensa® LX7, 240 MHz
- RAM: 512 KB SRAM và PSRAM 8 MB
- Flash: 8 MB - SPI flash

- Nguồn hoạt động: 3.0 V – 5 V
- Wi-Fi: IEEE 802.11 b/g/n (2.4 GHz)
- Bluetooth® 5.0 LE với chế độ long-range và Mesh
- I/O: Hơn 40 chân GPIO, hỗ trợ UART, SPI, I2C, ADC (12-bit), DAC (8-bit), PWM
- Hỗ trợ AI acceleration, phù hợp triển khai TensorFlow Lite micro
- Giao tiếp USB OTG (USB 1.1)

Môi trường lập trình:

- ESP-IDF (Espressif IoT Development Framework
- Arduino IDE, PlatformIO
- MicroPython

Trong hệ thống giám sát nuôi tảo, ESP32-S3-WROOM-1U đóng vai trò là vi điều khiển trung tâm. Thiết bị thu thập dữ liệu từ cảm biến TDS (analog) và DO-S20 (qua RS485 → UART), sau đó xử lý tín hiệu và so sánh với ngưỡng cảnh báo. Khi cần thiết, ESP32 điều khiển các thiết bị chấp hành như relay bơm sục khí hoặc bơm dưỡng chất. Đồng thời, dữ liệu được truyền về Raspberry Pi 4 thông qua giao thức MQTT, giúp hệ thống giám sát hoạt động tự động và theo thời gian thực.

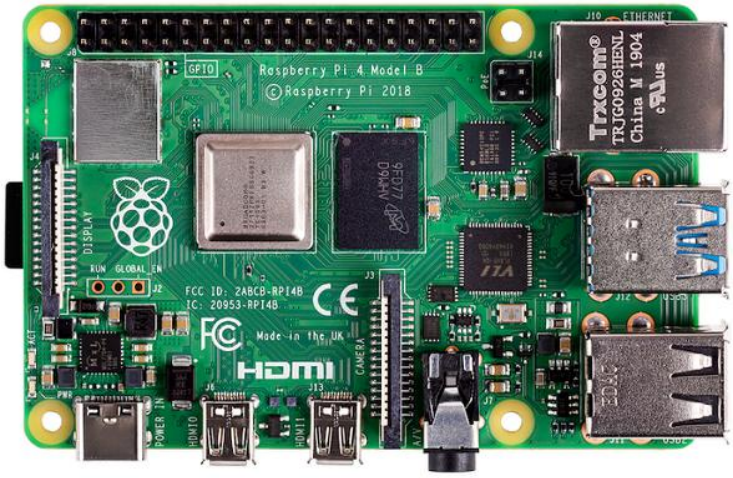
2.3.2 Raspberry Pi 4

Raspberry Pi 4 là một dòng máy tính nhúng đơn bo mạch (Single Board Computer – SBC) được phát triển bởi tổ chức Raspberry Pi Foundation, với mục tiêu ban đầu là hỗ trợ giáo dục và phổ cập tin học. Tuy nhiên, nhờ hiệu năng cao, chi phí thấp và khả năng lập trình linh hoạt, Raspberry Pi 4 hiện đã trở thành một nền tảng phổ biến trong nhiều ứng dụng kỹ thuật số, đặc biệt là trong các hệ thống nhúng và Internet of Things (IoT).

Về phần cứng, Raspberry Pi 4 được trang bị bộ vi xử lý Broadcom BCM2711, tích hợp 4 nhân ARM Cortex-A72 (64-bit) với tốc độ xung nhịp 1.5 GHz, cho phép xử

lý đa nhiệm hiệu quả. Thiết bị có các tùy chọn bộ nhớ RAM gồm 2 GB, 4 GB hoặc 8 GB LPDDR4, hỗ trợ lưu trữ qua thẻ microSD hoặc ổ cứng ngoài qua cổng USB 3.0. Các cổng giao tiếp chính bao gồm Ethernet Gigabit, Wi-Fi 802.11ac, Bluetooth 5.0, cùng hệ thống 40 chân GPIO cho phép mở rộng kết nối với nhiều thiết bị ngoại vi.

Về phần mềm, Raspberry Pi 4 sử dụng hệ điều hành mã nguồn mở Raspberry Pi OS (dựa trên Debian Linux), hỗ trợ đầy đủ các ngôn ngữ lập trình như Python, C/C++, Node.js,… Đồng thời, nền tảng này còn cho phép triển khai các dịch vụ máy chủ như Mosquitto (MQTT broker), InfluxDB (cơ sở dữ liệu thời gian thực), và Node-RED (giao diện điều khiển trực quan) – những công cụ phổ biến trong hệ sinh thái IoT hiện nay.



Hình 2-4 Máy tính nhúng Raspberry Pi 4

Trong hệ thống giám sát nuôi tảo, Raspberry Pi 4 đóng vai trò như một máy chủ trung tâm (local server), thực hiện các chức năng chính như:

- Tiếp nhận dữ liệu từ vi điều khiển ESP32 thông qua giao thức MQTT, với vai trò là subscriber hoặc broker.

- Lưu trữ dữ liệu đo đạc từ các cảm biến (pH, nhiệt độ, DO, TDS) vào cơ sở dữ liệu InfluxDB, giúp đảm bảo tính toàn vẹn và khả năng truy xuất thời gian thực.
- Hiển thị dữ liệu qua giao diện Node-RED Dashboard, cho phép người dùng theo dõi trực tuyến các chỉ số môi trường.
- Ghi nhận hình ảnh từ camera USB định kỳ, phục vụ xử lý ảnh và phân tích sức khỏe tảo bằng thị giác máy tính.
- Hỗ trợ huấn luyện và triển khai các mô hình học máy (Machine Learning), phục vụ đánh giá tình trạng hệ thống hoặc tự động hóa ra quyết định điều khiển.

Việc tích hợp Raspberry Pi 4 trong hệ thống không chỉ giúp đơn giản hóa kiến trúc phần cứng mà còn tăng tính mở rộng và khả năng tùy biến, đáp ứng tốt yêu cầu của một hệ thống IoT thông minh trong lĩnh vực nuôi trồng thuỷ sinh.

2.3.3 IC nguồn LM2596

LM2596 là một vi mạch chuyển đổi DC-DC dạng buck (giảm áp), thuộc nhóm bộ ổn áp xung hiệu suất cao, được sử dụng phổ biến trong các mạch điện tử cần hạ áp từ nguồn đầu vào cao xuống mức điện áp thấp hơn, ổn định và có thể điều chỉnh được.

Vi mạch này có khả năng hoạt động với điện áp đầu vào từ 4V đến 40V DC, và cung cấp điện áp đầu ra có thể điều chỉnh trong khoảng 1.25V đến 37V DC, với dòng tải tối đa lên đến 2A. Nhờ sử dụng kỹ thuật chuyển mạch ở tần số 150 kHz, LM2596 cho hiệu suất cao (thường đạt trên 80%), giảm tổn hao năng lượng so với các bộ ổn áp tuyến tính.

Hình 2-5 IC ổn áp LM2576

LM2596 thường được tích hợp sẵn vào module có biến trở điều chỉnh đầu ra, tụ lọc và diode Schottky, giúp việc tích hợp vào hệ thống trở nên đơn giản. Thiết bị này còn được trang bị các chức năng bảo vệ như:

- Bảo vệ quá dòng
- Bảo vệ quá nhiệt
- Bảo vệ ngắn mạch đầu ra

Trong hệ thống giám sát nuôi tảo, LM2596 được sử dụng để hạ áp từ nguồn 12 V DC xuống 5 V DC nhằm cung cấp điện ổn định cho các thiết bị nhạy cảm như vi điều khiển ESP32-S3, module relay, cảm biến TDS và các mạch điều khiển logic. Nhờ khả năng cấp dòng ổn định, IC này đảm bảo hoạt động liên tục và an toàn của hệ thống trong điều kiện môi trường khắc nghiệt.

2.3.4 RS485

RS485 (Recommended Standard 485) là một chuẩn truyền thông nối tiếp công nghiệp, được thiết kế để truyền dữ liệu một cách hiệu quả và tin cậy trong môi trường có nhiễu điện từ cao. Chuẩn này được sử dụng rộng rãi trong các hệ thống điều khiển giám sát, tự động hóa công nghiệp, và đặc biệt phổ biến trong các giao thức như Modbus RTU, Profibus, SCADA,…

RS485 sử dụng phương pháp truyền tín hiệu vi sai (differential signaling) qua hai dây tín hiệu A và B. Thay vì truyền tín hiệu logic theo mức điện áp tuyệt đối như chuẩn UART thông thường, RS485 truyền tín hiệu dựa trên hiệu điện thế giữa hai dây, giúp tăng cường khả năng chống nhiễu và duy trì tính toàn vẹn dữ liệu ở khoảng cách xa.

- Thông số kỹ thuật đặc trưng:

– Kiểu truyền thông: Half-duplex (hai chiều nhưng không đồng thời), vi sai

– Cấu hình đường truyền: 2 dây chính (A và B), thêm GND nếu cần đồng bộ

– Khoảng cách truyền tối đa: lên đến 1200 mét ở tốc độ thấp (~100 kbps)

– Tốc độ truyền tối đa: lên đến 10 Mbps (ở khoảng cách ngắn ≤ 15 m)

– Số thiết bị kết nối: tối đa 32 thiết bị truyền/nhận trên một đường bus (mở rộng đến 128 hoặc hơn với các driver hiện đại)

- Tín hiệu logic:

– Logic “1”: $V_A - V_B > +200$ mV

– Logic “0”: $V_A - V_B < -200$ mV

- Ưu điểm của RS485:

– Truyền dữ liệu khoảng cách xa mà không suy hao

– Khả năng chống nhiễu cao, thích hợp trong môi trường có EMI lớn

– Hỗ trợ truyền thông đa điểm (multi-drop), tiết kiệm dây dẫn

– Đơn giản, dễ triển khai với chi phí thấp

Trong đề tài, chuẩn RS485 được sử dụng để kết nối cảm biến DO-S20 (tích hợp đo DO, pH, nhiệt độ) với vi điều khiển ESP32-S3, thông qua module chuyển đổi MAX485. Nhờ đó, dữ liệu môi trường được truyền đi ổn định và chính xác, đảm bảo khả năng giám sát liên tục và tin cậy cho hệ thống nuôi tảo.

2.4 Các phần mềm phục vụ thiết kế.

2.4.1 Phần mềm Autodesk Fusion 360

Autodesk Fusion 360 là phần mềm thiết kế 3D theo mô hình tham số và tích hợp toàn diện các công cụ từ thiết kế cơ khí, mô phỏng, gia công CNC đến kết xuất và cộng tác trực tuyến. Fusion 360 được phát triển bởi hãng Autodesk, hoạt động đa nền tảng trên cả Windows và macOS, với khả năng đồng bộ hóa đám mây (cloud-based), giúp người dùng dễ dàng truy cập, chia sẻ và làm việc nhóm.

Nhờ giao diện hiện đại, trực quan và tích hợp nhiều module thiết kế trong một môi trường duy nhất, Fusion 360 đang ngày càng được ứng dụng rộng rãi trong các lĩnh vực như cơ khí chính xác, sản xuất thiết bị điện tử, chế tạo mô hình mẫu, thiết kế công nghiệp và các hệ thống điều khiển tự động.

Hình 2-6 Phần mền thiết kế Fusion 360

Các tính năng chính của Fusion 360:

- Thiết kế 3D (Solid Modeling, Surface Modeling): Hỗ trợ dựng mô hình chi tiết từ bản sketch 2D thông qua các lệnh như Extrude, Revolve, Loft, Sweep, cùng khả năng tạo hình tự do bằng các công cụ Form.
- Lắp ráp (Assembly): Tạo tổ hợp từ nhiều chi tiết, kiểm tra mối liên kết (Joint), ràng buộc chuyển động, sự va chạm và khớp cơ khí giữa các bộ phận.

- Tạo bản vẽ kỹ thuật (Drawing): Xuất bản vẽ 2D tự động từ mô hình 3D, kèm đầy đủ kích thước, dung sai, ghi chú, mặt cắt và ký hiệu kỹ thuật theo tiêu chuẩn.
- Phân tích mô phỏng (Simulation & FEA): Mô phỏng ứng suất, biến dạng, mô men xoắn, tải nhiệt, dao động, giúp đánh giá độ bền cấu trúc trước khi chế tạo thực tế.
- Gia công CAM (Manufacture): Tích hợp module CAM giúp lập trình gia công CNC 2D/3D trực tiếp từ mô hình 3D, bao gồm tiện, phay, khoan,...
- Thiết kế tham số (Parametric Design): Cho phép gán ràng buộc kích thước, tạo mối liên hệ giữa các thành phần – tự động cập nhật mô hình khi thay đổi giá trị.
- Kết xuất (Rendering) & tạo file sản xuất: Hỗ trợ xuất file STEP, STL, IGES, DXF, DWG... và kết xuất hình ảnh 3D chân thực với vật liệu và ánh sáng thực tế.

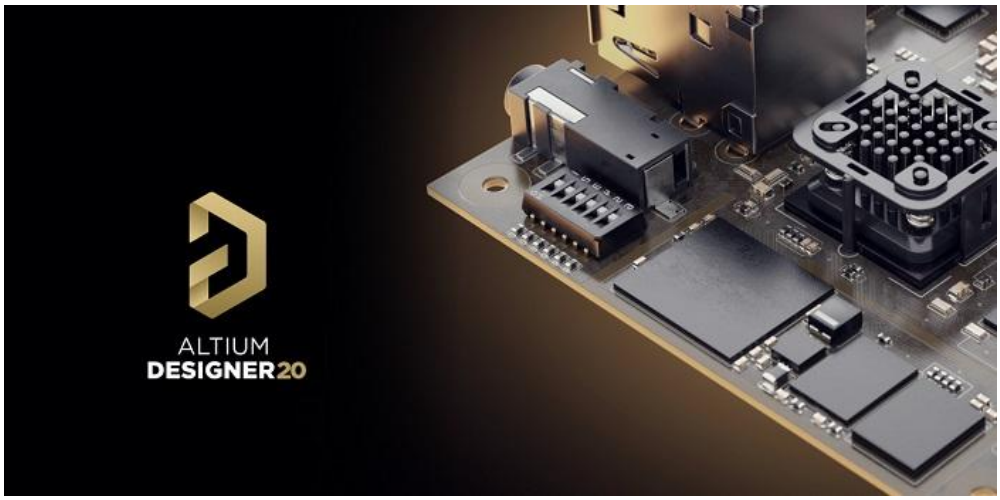
Trong đề tài, Fusion 360 được sử dụng để thiết kế mô hình 3D của khung vỏ máy quan trắc và hộp bảo vệ Raspberry Pi 4. Việc mô hình hóa này giúp đảm bảo các linh kiện được lắp đặt chính xác, gọn gàng và an toàn trong quá trình vận hành. Thiết kế bao gồm các chi tiết như khe thoát dây, lỗ bắt vít, khe gió và nắp tháo lắp thuận tiện. Sau khi hoàn tất, bản vẽ được xuất ra dưới dạng file DXF phục vụ cắt mica bằng laser hoặc STL để in 3D.

2.4.2 Phần mềm Altium Designer

Altium Designer là phần mềm thiết kế mạch điện tử tích hợp (EDA – Electronic Design Automation) do hãng Altium Limited phát triển. Đây là một trong những công cụ vẽ mạch điện tử chuyên nghiệp và mạnh mẽ nhất hiện nay, được sử dụng rộng rãi trong nghiên cứu, phát triển sản phẩm và sản xuất công nghiệp.

Altium Designer cho phép thiết kế toàn bộ quy trình phát triển mạch điện tử trên một nền tảng duy nhất – từ vẽ sơ đồ nguyên lý (schematic), thiết kế mạch in PCB, mô phỏng mạch điện, đến xuất bản vẽ chế tạo và kiểm tra thiết kế (DRC).

Với giao diện trực quan, thư viện linh kiện phong phú và khả năng tích hợp mạnh mẽ với các công cụ mô phỏng, Altium Designer hỗ trợ tốt cho cả sinh viên, kỹ sư và doanh nghiệp trong việc tạo ra các sản phẩm điện tử chất lượng cao, từ thiết bị tiêu dùng đến các hệ thống điều khiển công nghiệp, IoT, và thiết bị y tế.



Hình 2-7 Phần mền thiết kế mạch Altium

Các tính năng chính của Altium Designer:

- Vẽ sơ đồ nguyên lý (Schematic): Hỗ trợ xây dựng mạch điện logic một cách trực quan, dễ dàng liên kết các linh kiện.
- Thiết kế mạch in PCB: Tạo bố cục mạch in từ 1 lớp đến nhiều lớp, hỗ trợ định tuyến tự động và thủ công.
- Kiểm tra thiết kế tự động (DRC/ERC): Phát hiện lỗi kết nối, lỗi nguyên tắc thiết kế trước khi đưa vào sản xuất.
- Tạo thư viện linh kiện: Cho phép tạo, quản lý và sử dụng các linh kiện điện tử tùy chỉnh theo nhu cầu.
- Mô phỏng mạch điện: Hỗ trợ mô phỏng mạch số và tương tự để kiểm tra hoạt động trước khi sản xuất.

- Xuất file sản xuất (Gerber, BOM,...): Dễ dàng tạo file Gerber, file khoan, và danh sách linh kiện (BOM) cho chế tạo mạch.
- Tích hợp 3D và kiểm tra cơ khí: Hiển thị mô hình 3D của bo mạch, kiểm tra kích thước, tương thích với vỏ hoặc khung cơ khí.

Altium Designer đóng vai trò là phần mềm chính để thiết kế mạch điện tử, phục vụ cho việc thu thập dữ liệu từ cảm biến, xử lý tín hiệu, cấp nguồn và giao tiếp giữa các thiết bị trong hệ thống nuôi tảo. Các ứng dụng cụ thể bao gồm:

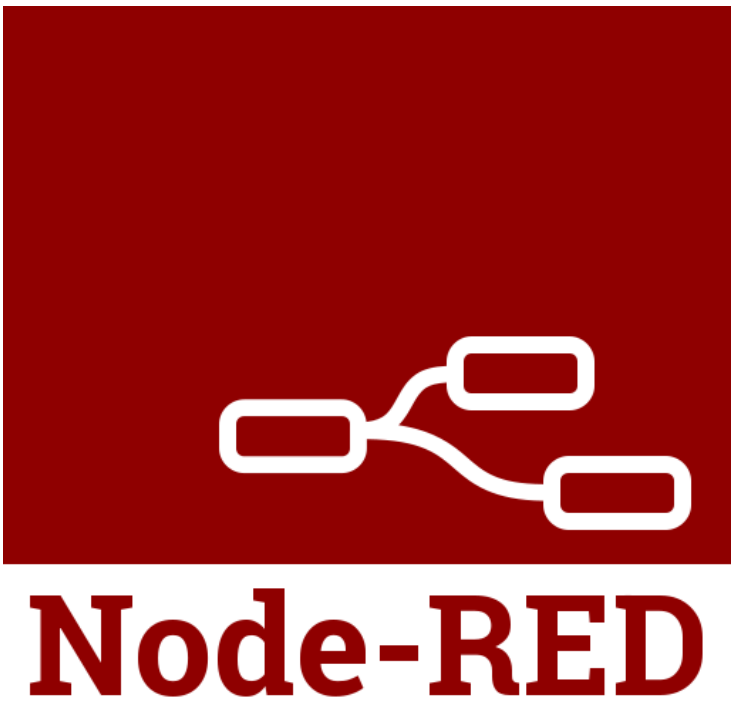
- Thiết kế mạch điều khiển trung tâm với ESP32-S3: Sử dụng Altium để vẽ sơ đồ nguyên lý (schematic) của mạch điều khiển chính dựa trên vi điều khiển ESP32-S3 WROOM-32. Bố trí bo mạch in (PCB layout) với các chân I/O kết nối đến cảm biến, UART, camera và nguồn.
- Thiết kế mạch giao tiếp cảm biến công nghiệp: Thiết kế mạch chuyển đổi RS485-TTL sử dụng IC MAX485, dùng để đọc dữ liệu từ cảm biến DO-S20 (đo DO, pH, nhiệt độ). Đảm bảo các tín hiệu từ cảm biến được đưa về ESP32 một cách ổn định, chống nhiễu.
- Thiết kế mạch kết nối cảm biến TDS: Thiết kế mạch đọc tín hiệu từ cảm biến TDS Gravity qua giao tiếp analog hoặc UART. Tích hợp thêm bộ lọc RC để loại bỏ nhiễu tín hiệu trước khi đưa vào vi điều khiển.
- Thiết kế mạch nguồn cho toàn hệ thống: Sử dụng IC LM2596 để tạo bộ nguồn ổn định 3.3V và 5V, cấp cho ESP32, cảm biến và các module truyền thông. Thiết kế mạch bảo vệ nguồn, chống ngược cực, chống quá dòng.
- Thiết kế giao tiếp giữa ESP32 và Raspberry Pi 4: Vẽ sơ đồ kết nối giao tiếp UART giữa ESP32 và Raspberry Pi 4 để truyền tín hiệu trigger từ vi điều khiển lên máy chủ, phục vụ xử lý ảnh và lưu trữ dữ liệu.

2.4.3 Phần mềm Node-RED.

Node-RED là một phần mềm mã nguồn mở do IBM phát triển, dùng để xây dựng các ứng dụng Internet of Things (IoT) thông qua phương pháp lập trình trực quan

dạng luồng (Flow-based programming). Thay vì phải viết code phức tạp, người dùng chỉ cần kéo – thả các khối chức năng (node) để tạo ra các luồng xử lý dữ liệu (flow), giúp tiết kiệm thời gian và giảm độ phức tạp trong triển khai hệ thống.

Node-RED được viết bằng JavaScript (chạy trên nền Node.js), tương thích tốt với nhiều nền tảng như Windows, Linux, Raspberry Pi,... Đặc biệt, phần mềm rất phù hợp với các hệ thống IoT, nơi cần giao tiếp với thiết bị, cảm biến, xử lý và trực quan hóa dữ liệu.



Hình 2-8 Giao diện Node- RED

Các tính năng chính của Node-RED:

- Lập trình kéo-thả dễ sử dụng: Cho phép thiết kế luồng xử lý logic mà không cần lập trình chuyên sâu. Các node được kết nối với nhau thành sơ đồ xử lý rõ ràng, trực quan.
- Giao tiếp đa dạng với thiết bị IoT: Node-RED hỗ trợ nhiều giao thức như MQTT, HTTP, Modbus, TCP/IP,..., giúp dễ dàng giao tiếp với ESP32, Raspberry Pi, và các cảm biến công nghiệp.

- Tích hợp cơ sở dữ liệu: Dễ dàng kết nối với InfluxDB, MySQL, Firebase,... để lưu trữ dữ liệu theo thời gian hoặc truy xuất lịch sử.
- Tạo giao diện giám sát (Dashboard): Cung cấp các widget như biểu đồ, đồng hồ, thanh trạng thái,... để người dùng theo dõi trực tiếp giá trị cảm biến trên trình duyệt web.
- Tự động hóa và cảnh báo: Có thể thiết lập các luồng xử lý để gửi email cảnh báo, bật tắt thiết bị, hoặc kích hoạt các hành động khi giá trị vượt ngưỡng.
- Cộng đồng hỗ trợ mạnh: Có sẵn hàng nghìn node mở rộng cho nhiều mục đích khác nhau: AI, điều khiển thiết bị, kết nối đám mây, gửi tin nhắn,...

Node-RED giữ vai trò trung tâm điều phối và hiển thị dữ liệu trong hệ thống:

- Kết nối với ESP32 qua giao thức MQTT hoặc UART, nhận dữ liệu từ các cảm biến như pH, nhiệt độ, TDS, DO,...
- Lưu trữ dữ liệu cảm biến vào InfluxDB thông qua node ghi dữ liệu.
- Hiển thị giao diện Dashboard cho người dùng trên trình duyệt web (PC hoặc điện thoại) – bao gồm biểu đồ, chỉ số, trạng thái,...
- Tự động gửi cảnh báo qua email nếu phát hiện các giá trị bất thường (ví dụ pH vượt ngưỡng hoặc mất tín hiệu cảm biến).
- Xử lý ảnh gián tiếp bằng cách tiếp nhận tín hiệu từ Raspberry Pi hoặc trigger mô hình học máy khi cần.

2.4.4 Phần mềm InfluxDB

InfluxDB là một phần mềm cơ sở dữ liệu chuyên biệt cho dữ liệu dạng chuỗi thời gian (time-series), được phát triển bởi công ty InfluxData. Đây là công cụ rất phổ biến trong các hệ thống IoT, giám sát cảm biến, tự động hóa công nghiệp và phân tích dữ liệu thời gian thực.

Khác với các hệ quản trị cơ sở dữ liệu truyền thống (như MySQL, PostgreSQL), InfluxDB được tối ưu hóa để lưu trữ, truy vấn và xử lý dữ liệu gắn theo dấu thời

gian – ví dụ như: nhiệt độ, độ ẩm, pH, TDS, nồng độ DO,... được ghi lại liên tục từ các cảm biến.

Hình 2-9 Cơ sở dữ liệu influxdb

Các tính năng chính của InfluxDB:

- Lưu trữ dữ liệu theo thời gian thực: InfluxDB cho phép ghi nhận hàng triệu dữ liệu cảm biến theo từng thời điểm cụ thể, đảm bảo độ chính xác cao và tốc độ ghi nhanh.
- Truy vấn mạnh mẽ với ngôn ngữ InfluxQL: Cho phép người dùng lọc, thống kê, tính toán trung bình, cực trị, tốc độ thay đổi,... từ dữ liệu cảm biến theo từng mốc thời gian.
- Tích hợp dễ dàng với hệ thống IoT: Hỗ trợ giao tiếp với các nền tảng như Node-RED, MQTT, Telegraf, Python,... nên dễ kết nối với các thiết bị như ESP32, Raspberry Pi.
- Lưu trữ hiệu quả và nhẹ: Có thể chạy trực tiếp trên các thiết bị nhỏ như Raspberry Pi mà vẫn đảm bảo hiệu suất cao trong lưu trữ và truy xuất dữ liệu.
- Hỗ trợ trực quan hóa dữ liệu: Kết hợp tốt với Node-RED Dashboard hoặc phần mềm đồ thị như Grafana để tạo ra các biểu đồ theo thời gian thực.

Phần mềm InfluxDB đóng vai trò trung tâm lưu trữ dữ liệu thu thập được từ hệ thống cảm biến và vi điều khiển. Cụ thể:

- ESP32 gửi dữ liệu từ các cảm biến (nhiệt độ, pH, TDS, DO) qua giao thức MQTT về Raspberry Pi, nơi đang cài đặt InfluxDB.
- Dữ liệu được ghi lại theo từng mốc thời gian thực, giúp theo dõi biến động môi trường nuôi tảo.
- Người dùng có thể truy vấn và hiển thị biểu đồ thông qua Node-RED Dashboard để theo dõi quá trình nuôi tảo một cách trực quan.

Ngoài ra, các dữ liệu lịch sử được lưu giữ có thể được dùng làm đầu vào để huấn luyện mô hình học máy, phục vụ dự đoán sức khỏe tảo hoặc phát hiện bất thường.

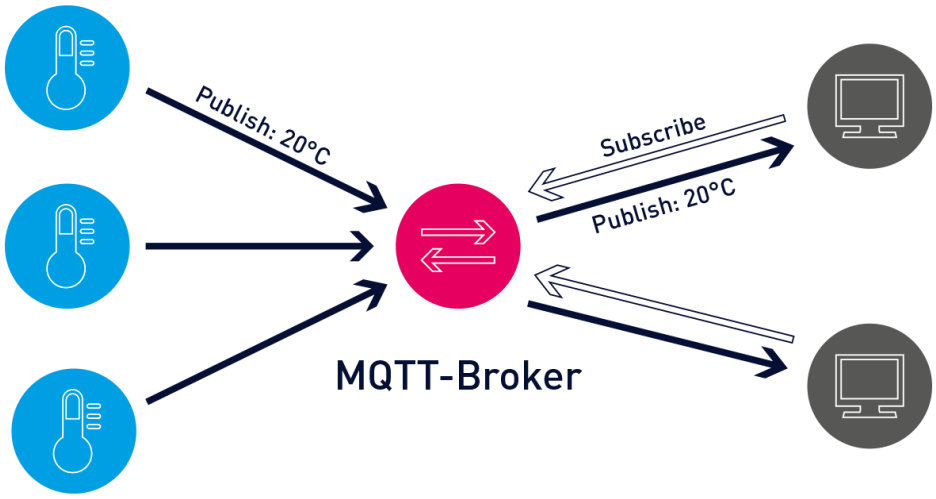
2.5 Các phương thức truyền dữ liệu trong hệ thống

2.5.1 Truyền thông MQTT

MQTT (Message Queuing Telemetry Transport) là một giao thức truyền thông nhẹ, hiệu quả và phổ biến trong các hệ thống Internet of Things (IoT). MQTT được thiết kế để thu thập dữ liệu từ nhiều thiết bị và gửi chúng đến máy chủ giám sát một cách nhanh chóng, ổn định và tiêu tốn ít băng thông.

Giao thức này hoạt động dựa trên mô hình "Publish – Subscribe" (công bố – đăng ký) thông qua một máy chủ trung gian gọi là MQTT Broker. Trong đó:

- Các thiết bị như ESP32 sẽ gửi dữ liệu cảm biến (publish) theo từng "chủ đề" (topic).
- Các thiết bị hoặc phần mềm như Node-RED hoặc InfluxDB sẽ đăng ký nhận dữ liệu từ các chủ đề đó (subscribe).



Hình 2-10 Mô tả giao tiếp giữa MQTT – Broker

Đặc điểm nổi bật của MQTT:

- Băng thông thấp: Tin nhắn gọn nhẹ, phù hợp cho mạng không dây hoặc thiết bị tài nguyên hạn chế.
- Độ tin cậy cao: Hỗ trợ xác nhận tin nhắn với nhiều mức chất lượng dịch vụ (QoS), đảm bảo không mất dữ liệu.
- Thời gian thực: Đáp ứng nhanh, độ trễ thấp, phù hợp cho các ứng dụng yêu cầu phản hồi nhanh.
- Chi phí thấp và dễ triển khai: Dễ cài đặt trên thiết bị nhỏ như Raspberry Pi, hỗ trợ đa nền tảng.
- Khả năng mở rộng tốt: Dễ dàng thêm thiết bị mới vào hệ thống chỉ bằng cách cấu hình đúng topic.

Ứng dụng MQTT trong đề tài:

- Truyền dữ liệu cảm biến (nhiệt độ, pH, TDS, DO) từ ESP32 về máy chủ Raspberry Pi.
- Giao tiếp giữa ESP32 và Node-RED, giúp hiển thị dữ liệu real-time và lưu trữ vào InfluxDB.

- Gửi tín hiệu xử lý ảnh, hoặc cảnh báo lỗi khi hệ thống phát hiện sự cố như mất sục khí, sai lệch thông số, v.v.

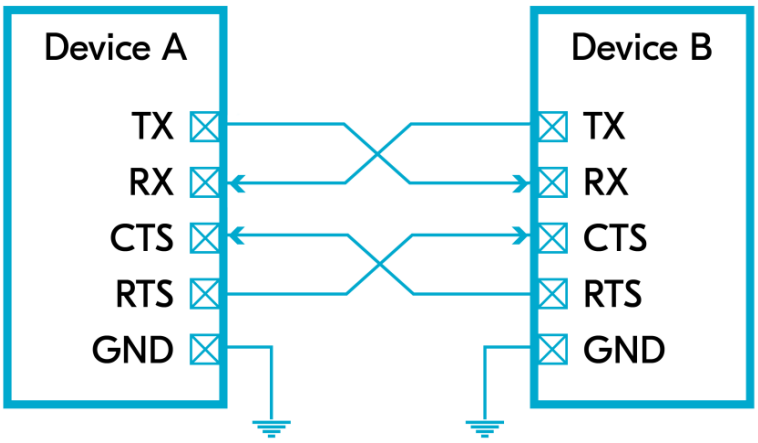
Nhờ MQTT, hệ thống đạt được hiệu suất truyền thông cao, tiết kiệm băng thông, đồng thời đảm bảo dữ liệu được truyền tải chính xác và ổn định theo thời gian thực.

2.5.2 Giao tiếp UART

UART (Universal Asynchronous Receiver – Transmitter) là một chuẩn truyền thông nối tiếp đơn giản, phổ biến, cho phép hai thiết bị trao đổi dữ liệu theo kiểu nối tiếp không đồng bộ. UART hoạt động thông qua hai đường dây chính:

- TX (Transmit): Dây truyền dữ liệu.
- RX (Receive): Dây nhận dữ liệu.

UART không yêu cầu đồng hồ xung (clock) chung giữa hai thiết bị, do đó dễ triển khai trong các hệ thống nhúng như ESP32, Raspberry Pi,…



Hình 2-11 Chuyển giao tiếp UART

Đặc điểm của chuẩn UART:

- Truyền nối tiếp từng bit theo thời gian, thường ở tốc độ từ 9600 đến 115200 bps.
- Chỉ dùng 2 dây chính: TX và RX (ngoài ra có thể có GND).
- Không đồng bộ, nên dễ sử dụng nhưng yêu cầu hai thiết bị cấu hình đúng baudrate (tốc độ truyền).

- Chỉ giao tiếp giữa hai thiết bị (point-to-point) – không hỗ trợ đa điểm như RS485.

Ứng dụng trong hệ thống:

- ESP32 và Raspberry Pi được kết nối trực tiếp qua chuẩn UART, dùng để truyền các tín hiệu điều khiển hoặc khởi động luồng xử lý ảnh.
- Là cầu nối trung gian cho dữ liệu từ cảm biến đến hệ thống lưu trữ/phân tích.
- Có thể được sử dụng để cấu hình cảm biến, gửi lệnh từ người dùng tới vi điều khiển.

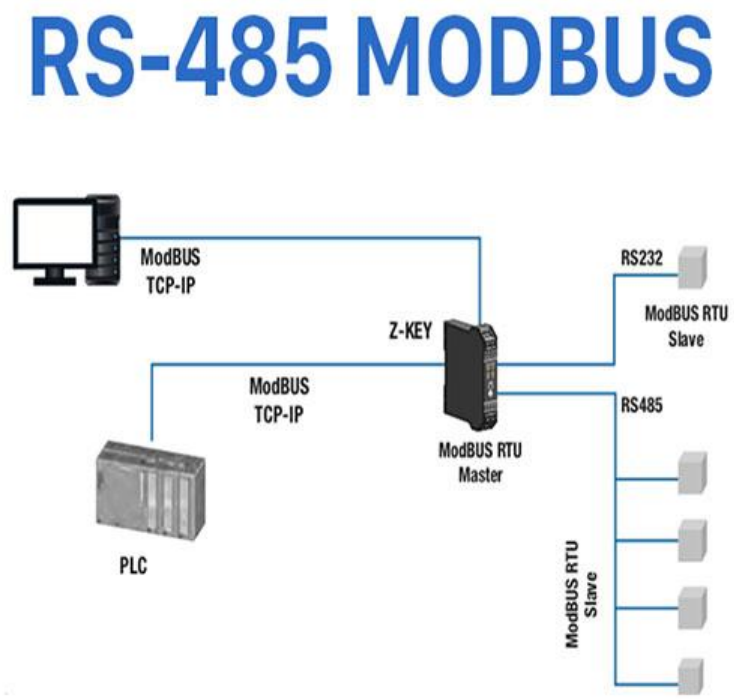
Ưu điểm:

- Dễ sử dụng và phổ biến trong hệ thống nhúng.
- Tiết kiệm dây và tài nguyên phần cứng.
- Hoạt động ổn định ở khoảng cách ngắn (dưới 5 mét).

2.5.3 Truyền thông Modbus

Modbus là một giao thức truyền thông công nghiệp phổ biến được thiết kế để trao đổi dữ liệu giữa các thiết bị điều khiển và giám sát như PLC, cảm biến, thiết bị đo lường, và các bộ điều khiển nhúng. Giao thức này được phát triển lần đầu tiên vào năm 1979 bởi hãng Modicon (nay thuộc Schneider Electric), và nhanh chóng trở thành tiêu chuẩn mở được ứng dụng rộng rãi trong lĩnh vực tự động hóa và điều khiển công nghiệp.

Modbus nổi bật nhờ cấu trúc dữ liệu đơn giản, dễ cài đặt và tích hợp với các hệ thống vi điều khiển, hệ thống nhúng hoặc SCADA. Giao thức này cho phép một thiết bị chủ (master) giao tiếp với một hoặc nhiều thiết bị tớ (slave) thông qua việc gửi và nhận các khung dữ liệu (data frame).



Hình 2-12 Chuẩn giao tiếp RS-485 MODBUS

Modbus hỗ trợ nhiều chuẩn truyền dẫn, trong đó hai dạng chính là:

- Modbus RTU (Remote Terminal Unit): hoạt động trên đường truyền nối tiếp (serial) như RS485 hoặc RS232, dữ liệu được mã hóa nhị phân để tăng hiệu quả truyền tải.
- Modbus TCP: hoạt động trên nền tảng mạng Ethernet, cho phép truyền dữ liệu thông qua giao thức TCP/IP – phù hợp với các hệ thống giám sát hiện đại.

Trong hệ thống nuôi tảo thông minh, Modbus RTU qua chuẩn RS485 thường được sử dụng để giao tiếp với các cảm biến công nghiệp như cảm biến DO, pH, hoặc các bộ điều khiển rời nhờ khả năng truyền dữ liệu ổn định, khoảng cách xa (lên đến hàng trăm mét) và khả năng kết nối nhiều thiết bị trên cùng một bus truyền.

Ưu điểm của Modbus bao gồm:

- Dễ tích hợp với các nền tảng vi điều khiển (như ESP32, STM32,...)
- Tương thích với nhiều loại thiết bị công nghiệp tiêu chuẩn

- Cấu trúc khung dữ liệu đơn giản, xử lý nhanh
- Có thể truyền nhiều loại dữ liệu (số nguyên, số thực, bit trạng thái,...)

Với đặc điểm đó, Modbus là một giao thức lý tưởng để kết nối các cảm biến và thiết bị ngoại vi trong hệ thống IoT giám sát nuôi tảo, giúp đảm bảo luồng dữ liệu ổn định, chính xác và có thể mở rộng linh hoạt.

2.6 Các thiết bị chấp hành trong hệ thống

2.6.1 Máy bơm dưỡng chất và van xả

Máy bơm 365 DC 12V là một loại bơm ly tâm mini được sử dụng phổ biến trong các ứng dụng bơm nước hoặc chất lỏng ở quy mô nhỏ. Với thiết kế nhỏ gọn, trọng lượng nhẹ, chi phí hợp lý và khả năng hoạt động ổn định ở nguồn 12VDC, thiết bị này là lựa chọn phù hợp cho các hệ thống điện tử điều khiển tự động yêu cầu dòng chảy ổn định và liên tục.

Hình 2-13 Bơm chất dinh dưỡng 12v(DC)

Thông số kỹ thuật cơ bản của bơm 365 12VDC:

- Điện áp hoạt động: 12VDC
- Dòng tiêu thụ không tải: ~0.23 A
- Lưu lượng dòng chảy: 2 – 3 lít/phút
- Áp suất đầu ra: 1 – 2.5 kg/cm²
- Độ sâu hút: 1 – 2.5 mét
- Đầu vào/đầu ra: đường kính ngoài 8 mm

- Tuổi thọ thiết kế: khoảng 2 – 3 năm (trong điều kiện làm việc bình thường)
- Trọng lượng: ~111 g
- Loại động cơ: chổi than DC loại 365

Trong hệ thống giám sát và quản lý nuôi tảo thông minh, máy bơm đóng vai trò trung tâm trong quá trình bổ sung dưỡng chất vào bể nuôi. Khi nồng độ TDS trong môi trường thấp hơn ngưỡng cho phép (do thu hoạch, bay hơi hoặc hấp thu dinh dưỡng), vi điều khiển sẽ điều khiển bơm hoạt động để đưa dung dịch dinh dưỡng từ bình chứa vào hệ thống nuôi. Kết hợp với van điện từ, quá trình cấp bổ sung diễn ra tự động, chính xác và tối ưu.

Việc sử dụng bơm 365 mang lại nhiều ưu điểm đáng kể cho hệ thống: thiết bị có khả năng hoạt động ổn định với điện áp thấp, dễ dàng điều khiển bằng relay thông qua vi điều khiển ESP32, cho phép tích hợp mạch điều khiển đơn giản. Ngoài ra, với lưu lượng vừa phải và áp suất đủ lớn, bơm đáp ứng tốt yêu cầu bơm dưỡng chất ở cự ly gần mà không gây xáo trộn mạnh dòng nước trong bể. Thiết bị cũng có sẵn trên thị trường, dễ thay thế, sửa chữa và phù hợp với định hướng phát triển hệ thống tiết kiệm và bền vững trong quy mô thí nghiệm hoặc thương mại nhỏ.

2.6.2 Máy Sủi Oxy RESUN ACO 002 25W

Trong hệ thống nuôi tảo, oxy hòa tan (DO – Dissolved Oxygen) đóng vai trò then chốt trong việc duy trì hoạt động sống và tăng trưởng của tế bào tảo. Đặc biệt đối với tảo Spirulina – một loài tảo hiếu khí – việc cung cấp đầy đủ oxy là điều kiện tiên quyết giúp quá trình quang hợp và sinh khối diễn ra ổn định. Máy sục khí RESUN ACO 002 25W là thiết bị chấp hành chính giúp tạo oxy hòa tan trong bể nuôi bằng cách bơm khí vào nước thông qua các đầu sủi, tạo thành các bọt khí mịn khuếch tán nhanh.

Hình 2-14 Bơm sục khí 220v(AC)

Vai trò chính:

- Tăng hàm lượng DO trong nước, hỗ trợ tảo phát triển tốt.
- Tạo dòng lưu chuyển nhẹ trong bể, giúp tảo không bị lắng và phân bố đều ánh sáng.
- Hỗ trợ quá trình trao đổi khí CO_2 – O_2 trong môi trường nuôi.
- Góp phần phát hiện lỗi hệ thống khi xử lý ảnh không phát hiện bọt khí

Thông số kỹ thuật:

- Loại thiết bị: RESUN ACO 002 25W
- Công suất định mức: 25W
- Lưu lượng khí: 40 L/phút
- Điện áp hoạt động: 220-240V/50-60Hz
- Số đầu khí: 6 đầu khí
- Ống khí: 3 mét ống dẫn loại Φ8mm
- Cơ chế hoạt động: Piston rung điện từ (Electromagnetic Pump)
- Độ ồn: < 40 dB
- Thương hiệu: Trung Quốc

Máy sục khí được lựa chọn trong hệ thống có khả năng cung cấp oxy ổn định và hoạt động liên tục 24/7, phù hợp với đặc thù nuôi tảo yêu cầu lưu lượng khí đều và lâu dài. Thiết bị có thiết kế vỏ cách nhiệt, độ bền cao, ít bị nóng khi vận hành kéo

dài. Với công suất thấp nhưng hiệu suất cao, máy giúp tiết kiệm điện năng đáng kể. Ngoài ra, cấu tạo đơn giản, dễ lắp đặt và bảo trì, cùng với khả năng vận hành êm ái, hạn chế rung động, giúp duy trì ổn định môi trường nuôi mà không ảnh hưởng đến sinh trưởng của tảo.

2.6.3 Đèn LED phổ ấm hỗ trợ hoạt động quang hợp ban đêm

Trong hệ thống nuôi tảo, ánh sáng là yếu tố thiết yếu để đảm bảo quá trình quang hợp diễn ra hiệu quả. Vào ban đêm hoặc trong điều kiện thiếu ánh sáng tự nhiên, cần sử dụng nguồn sáng nhân tạo để hỗ trợ quá trình tổng hợp chất hữu cơ trong tế bào tảo, từ đó duy trì và thúc đẩy sinh trưởng.

Hệ thống sử dụng đèn LED dây ánh sáng ấm (Warm White – 3000K) nhằm mô phỏng phổ ánh sáng mặt trời ở vùng đỏ, hỗ trợ tốt cho quá trình quang hợp. Loại đèn này phù hợp với đặc tính của loài tảo Spirulina, vốn nhạy với vùng phổ đỏ và vàng – những bước sóng có khả năng thâm nhập sâu vào môi trường nước.

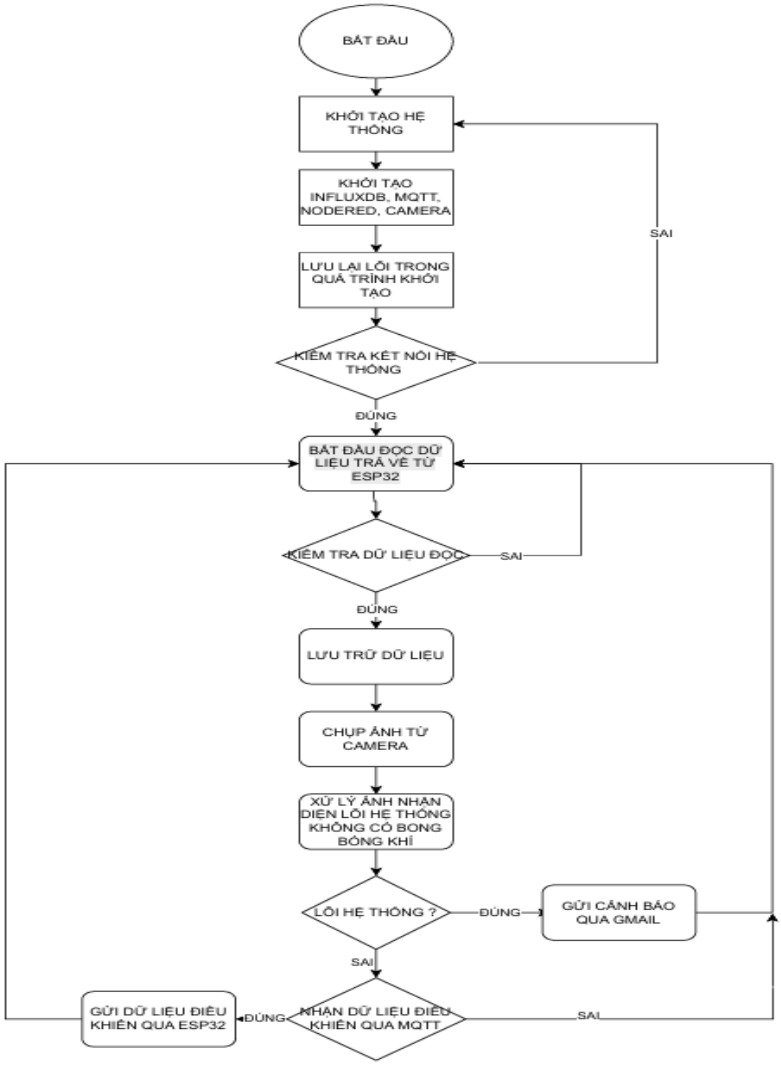
Thông số kỹ thuật của thiết bị:

- Model: LED Strip 2835 10W/m
- Nguồn vào: 24VDC
- Công suất tiêu thụ: 10W/m
- Chiều dài: 5m/roll
- Dòng điện: 2.08A
- Cường độ sáng: 600 lumen/m
- Loại LED: SMD 2835 – chip Seoul
- Nhiệt độ màu (CCT): 3000K – ánh sáng ấm (Warm White)
- Chỉ số hoàn màu (CRI): > 80
- Kích thước dây LED: Rộng 12mm, dày 4mm
- Lõi đồng: Dày 3oz = 105 micron
- Chống nước: IP65 – phù hợp môi trường ẩm
- Tiêu chuẩn an toàn: ROHS / EMC

CHƯƠNG 3. THIẾT KẾ GIAO DIỆN HỆ THỐNG GIÁM SÁT VÀ ĐIỀU KHIỂN .

3.1 THIẾT KẾ PHẦN MỀM

3.1.1 Lưu đồ giải thuật



Hình 3-1 Lưu đồ giải thuật hệ thống

.

Lưu đồ trong Hình 3.1.1 mô tả cách hệ thống hoạt động theo từng bước và cho thấy quá trình được lặp lại để hệ thống có thể chạy liên tục và ổn định. Quy trình xử lý được chia làm hai giai đoạn chính: giai đoạn khởi động hệ thống và giai đoạn hoạt động bình thường.

Trong giai đoạn đầu, hệ thống tiến hành khởi tạo các thành phần như InfluxDB, MQTT, Node-RED và camera. Nếu có lỗi xảy ra trong quá trình khởi tạo, hệ thống sẽ ghi lại lỗi và quay trở lại bước khởi tạo để thực hiện lại từ đầu. Sau khi khởi tạo xong, hệ thống kiểm tra các kết nối. Nếu các kết nối đúng và ổn định, hệ thống sẽ chuyển sang giai đoạn tiếp theo.

Khi hệ thống đã sẵn sàng, nó sẽ bắt đầu đọc dữ liệu từ ESP32. Dữ liệu sau khi đọc được kiểm tra tính hợp lệ, nếu đúng sẽ được lưu lại vào cơ sở dữ liệu. Đồng thời, hệ thống sẽ kích hoạt camera để chụp ảnh tại thời điểm đó, phục vụ cho việc phân tích ảnh.

Phần xử lý ảnh giúp phát hiện một số lỗi như không có bọt khí trong môi trường cần giám sát. Nếu phát hiện lỗi hệ thống, hệ thống sẽ gửi cảnh báo qua Gmail để người dùng kịp thời xử lý. Nếu không có lỗi, hệ thống sẽ kiểm tra xem có dữ liệu điều khiển nào gửi từ người dùng hoặc từ một hệ thống khác qua MQTT hay không. Nếu có, dữ liệu điều khiển sẽ được gửi xuống ESP32 để thực hiện các hành động như bật/tắt thiết bị.

Thiết kế theo kiểu chia nhánh và hoạt động theo vòng lặp như vậy giúp hệ thống có thể hoạt động ổn định, tự động, và có khả năng mở rộng thêm nhiều chức năng thông minh trong tương lai như cảnh báo sớm, theo dõi hành vi người dùng, hoặc kết nối với hệ thống giám sát trung tâm.

3.1.2 Xây dựng chương trình nhận diện hình ảnh lỗi hệ thống.

Trong hệ thống giám sát chất lượng nước tự động, chức năng nhận diện hình ảnh đóng vai trò then chốt nhằm đảm bảo khả năng phát hiện kịp thời các lỗi vận hành, đặc biệt là lỗi không có bọt khí, thường liên quan đến sự cố hệ thống cấp khí hoặc

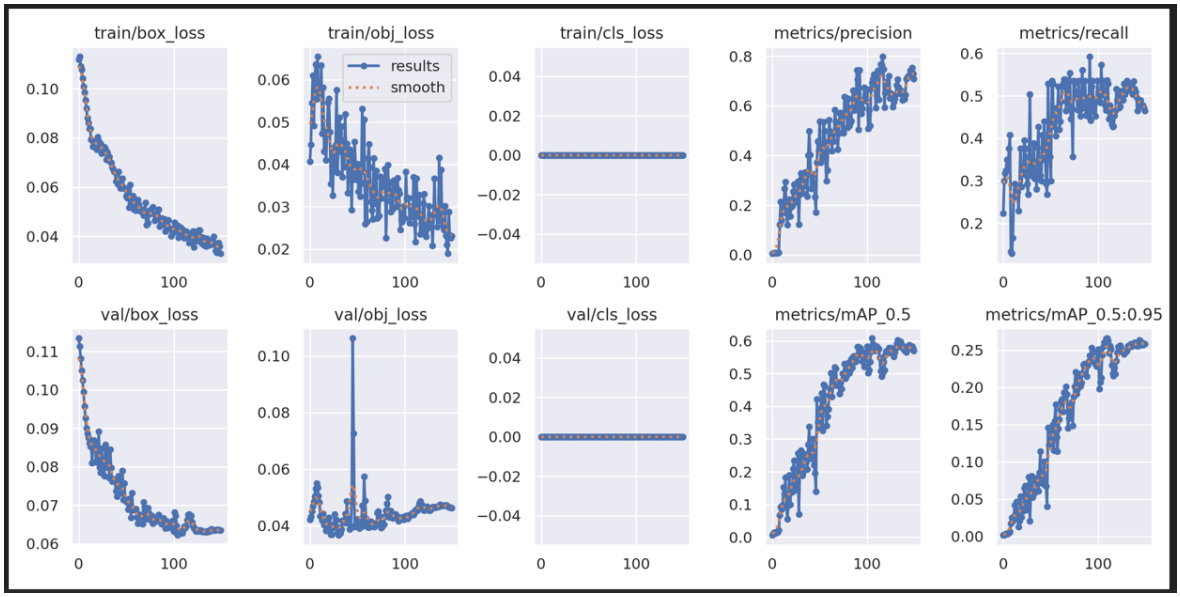
máy bơm khí ngừng hoạt động. Đặc biệt, hệ thống cần phải nhận diện chính xác trong các điều kiện môi trường ánh sáng và góc nhìn thay đổi, đây là các yếu tố có thể gây nhiễu quá trình nhận diện hình ảnh thông thường.

Để đáp ứng yêu cầu này, hệ thống đã lựa chọn mô hình YOLOv5 làm thuật toán chính cho quá trình nhận diện lỗi thông qua ảnh chụp từ camera giám sát. Mô hình YOLOv5 là một kiến trúc mạng CNN hiện đại được tối ưu cho việc phát hiện đối tượng theo cơ chế một bước (one-stage detection), cho phép xử lý nhanh, hiệu quả và phù hợp với bài toán thời gian thực trên các thiết bị có tài nguyên hạn chế. Cụ thể, YOLOv5 sử dụng backbone CSPDarknet kết hợp kỹ thuật Path Aggregation Network (PANet), giúp giảm thiểu đáng kể số lượng tham số và độ phức tạp tính toán nhưng vẫn duy trì khả năng trích xuất đặc trưng hiệu quả. Việc này giúp hệ thống nhận diện nhanh chóng và chính xác trạng thái bọt khí có hoặc không trong môi trường nước, đảm bảo hiệu năng cao ngay trên các thiết bị nhúng như Raspberry Pi.

Mục tiêu nhận diện ảnh trong hệ thống đặt ra là phát hiện đối tượng thuộc lớp chính là "Bubble" (bọt khí). Ngưỡng độ chính xác yêu cầu từ 50% trở lên nhằm đảm bảo khả năng vận hành tin cậy trong thực tế, nơi điều kiện ánh sáng và môi trường nước có thể thay đổi liên tục. Khi mô hình phát hiện không có đối tượng bọt khí hoặc độ tin cậy của đối tượng dưới ngưỡng đặt ra, hệ thống xác định xảy ra lỗi và ngay lập tức kích hoạt gửi cảnh báo.

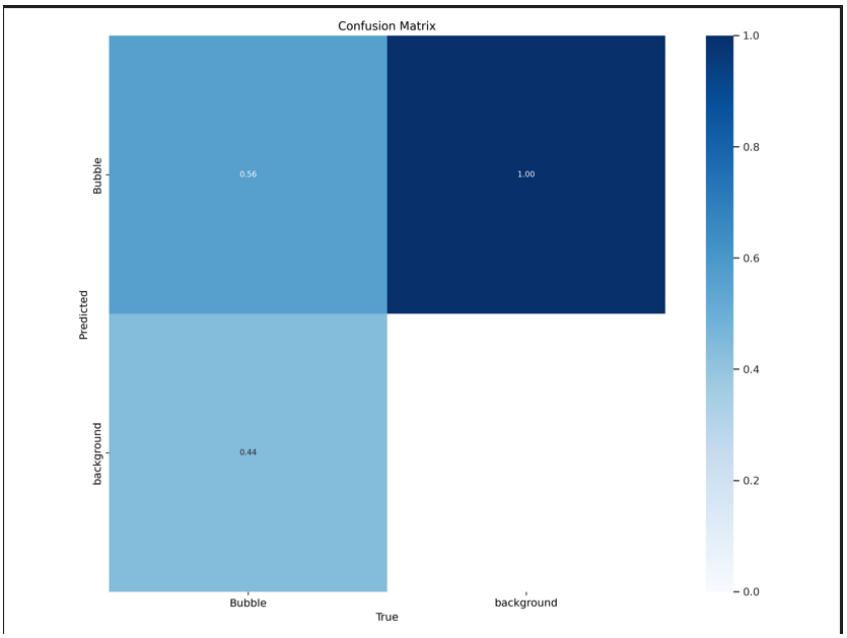
Quá trình huấn luyện mô hình YOLOv5 được thực hiện trên nền tảng Google Colab với bộ thư viện mã nguồn mở PyTorch, sử dụng tập dữ liệu hơn 400 hình ảnh được thu thập từ thực tế môi trường bể nước, các ảnh được gán nhãn theo hai lớp rõ ràng là "Bubble" và "background". Mô hình được huấn luyện trong 150 epoch, sử dụng thuật toán tối ưu hóa SGD với tốc độ học khởi tạo là 0.01, dữ liệu được chia theo tỷ lệ 80% cho huấn luyện và 20% cho kiểm thử. Kết quả huấn luyện được ghi nhận và theo dõi thông qua bộ công cụ tích hợp trong YOLOv5 và TensorBoard, từ đó xuất

đồ thị Learning Curve thể hiện sự hội tụ rõ ràng của mô hình trong quá trình huấn luyện.



Hình 3-2 Kết quả huấn luyện của mô hình

Hình 3.1 2 kết quả huấn luyện biểu diễn các đường cong mất mát (loss) và độ chính xác (precision, recall) cho thấy giá trị mất mát giảm ổn định dần theo số lượng epoch, trong khi các thông số đánh giá khác tăng lên rõ rệt. Sau khoảng hơn 100 epoch, độ chính xác và khả năng bao phủ (recall) trên tập kiểm thử đã ổn định, đạt giá trị mAP@0.5 khoảng 58%, đồng thời hàm mất mát duy trì ở mức thấp ổn định, cho thấy mô hình có khả năng học hiệu quả và không xuất hiện hiện tượng quá khớp (overfitting). Đồ thị Precision-Recall cũng thể hiện rõ khả năng cân bằng giữa độ chính xác và độ phủ của mô hình, đáp ứng tốt yêu cầu của bài toán.

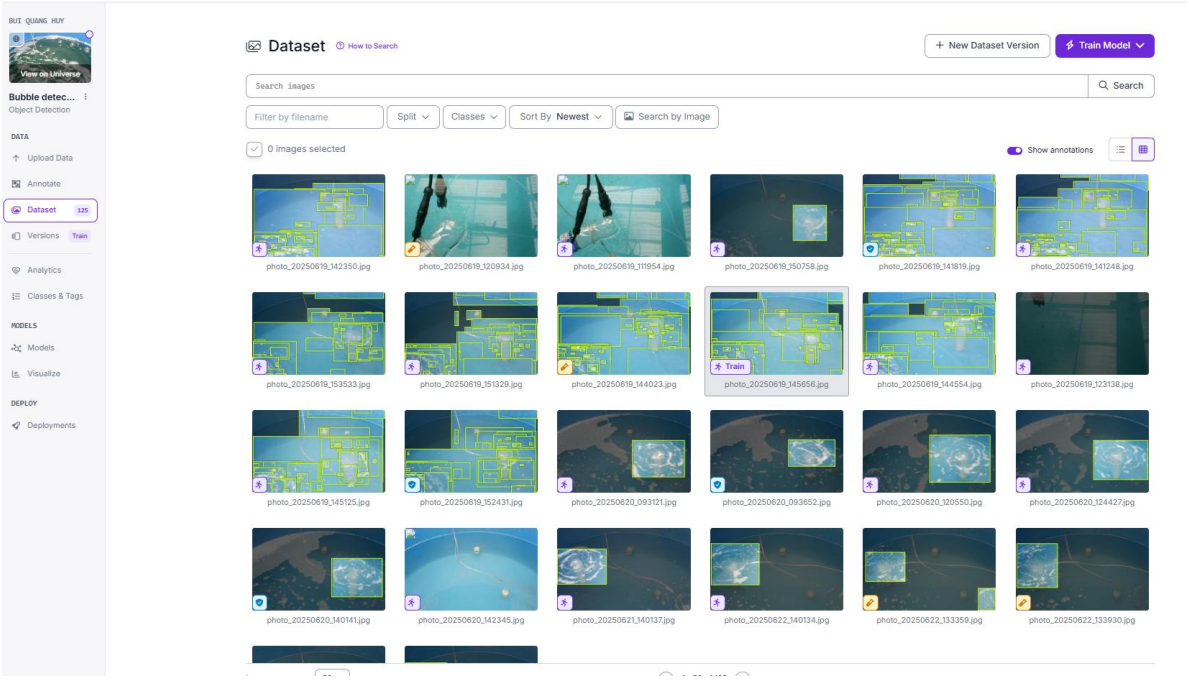


Hình 3-3 Ma trận nhầm lẫn

Hình 3.1-3 ma trận nhầm lẫn (confusion matrix) được xây dựng nhằm đánh giá cụ thể khả năng phân loại của mô hình, cho thấy tỉ lệ phân loại đúng lớp "Bubble" là khoảng 56%, còn lại 44% bị nhận diện nhầm sang background. Tuy nhiên, khả năng xác định đúng background đạt 100%, có nghĩa là không xuất hiện cảnh báo giả đối với ảnh nền không chứa bọt khí.

Hình 3-4 Kết quả nhận diện của mô hình trong thực tế.

Kết quả nhận diện hình ảnh lỗi hệ thống bằng mô hình YOLOv5 khi triển khai thực tế đạt độ chính xác cao và ổn định, với độ tin cậy 85% trên các ảnh đầu vào. Các khung giới hạn vùng bọt khí cũng được xác định chính xác và rõ ràng, thể hiện khả năng nhận diện tốt và ổn định của mô hình trong các điều kiện quan sát thực tế. Điều này khẳng định tính hiệu quả và độ tin cậy cao của mô hình YOLOv5 khi ứng dụng trong hệ thống giám sát chất lượng nước, đặc biệt trên các thiết bị nhúng có tài nguyên hạn chế như Raspberry Pi.



Hình 3-5 Gắn nhãn cho dữ liệu.

3.1.3 Xây dựng chương trình đọc cảm biến

Trong hệ thống giám sát chất lượng nước sử dụng ESP32, chức năng thu thập và xử lý dữ liệu cảm biến đóng vai trò trung tâm nhằm đảm bảo hệ thống có thể cung cấp thông tin chính xác về môi trường nước theo thời gian thực. Đặc biệt, hệ thống cần thu thập đồng thời nhiều thông số như nồng độ oxy hòa tan (DO), nhiệt độ, độ pH và tổng chất rắn hòa tan (TDS), sau đó tổng hợp và gửi lên máy chủ điều khiển để

hiển thị và giám sát. Việc đảm bảo độ chính xác và độ tin cậy trong việc đọc cảm biến, cũng như khả năng hoạt động ổn định trong thời gian dài là các yêu cầu then chốt khi triển khai thực tế ngoài hiện trường.

Để đáp ứng các yêu cầu trên, hệ thống sử dụng vi điều khiển ESP32 với chương trình được phát triển trên nền tảng Arduino Framework, đồng thời tận dụng khả năng xử lý song song của ESP32 bằng cách triển khai nhiệm vụ đọc cảm biến dưới dạng tác vụ độc lập (FreeRTOS task). Trong chương trình, các giao thức giao tiếp như UART, I2C được khai báo rõ ràng để kết nối với từng loại cảm biến, bao gồm, 2 Cảm biến DO S-20 giao tiếp qua UART (Serial2) sử dụng giao thức Modbus RTU, Module ADC ADS1115 đọc tín hiệu analog từ đầu dò TDS thông qua giao tiếp I2C, Dữ liệu đo được định dạng theo chuẩn JSON và gửi qua UART (Serial1) đến máy tính nhúng Raspberry Pi.

Quá trình đọc cảm biến được chia thành hai bước chính. Đầu tiên là giai đoạn khởi động và ổn định tín hiệu: hệ thống cấp nguồn cho cảm biến thông qua chân GPIO16 và chờ một khoảng thời gian 30 giây để đảm bảo cảm biến hoạt động ổn định. Sau đó, chương trình gửi một chuỗi lệnh định dạng sẵn (MSG_READ) theo chuẩn Modbus RTU đến cảm biến DO. Dữ liệu phản hồi sẽ được phân tích để trích xuất ba thông số quan trọng: nhiệt độ, nồng độ DO và giá trị pH. Nếu phản hồi hợp lệ (kiểm tra byte đầu và độ dài gói tin), giá trị đo được lưu vào biến và được cập nhật cho chu kỳ hiện tại.

Để đảm bảo hoạt động ổn định và tránh tình trạng treo hệ thống, watchdog timer được kích hoạt với thời gian timeout 10 giây, và được làm mới (esp_task_wdt_reset) trong mỗi vòng lặp tác vụ cảm biến cũng như trong vòng lặp loop(). Việc này giúp bảo vệ hệ thống khỏi các lỗi phần mềm kéo dài thời gian phản hồi.

Tần suất cập nhật dữ liệu được cấu hình là 5 phút mỗi lần đọc, đảm bảo vừa đủ để theo dõi sự thay đổi của môi trường nước, đồng thời tiết kiệm năng lượng và giảm tần suất giao tiếp không cần thiết. Toàn bộ chương trình được thiết kế tối ưu cho các

hệ thống nhúng với tài nguyên hạn chế, nhưng vẫn đảm bảo đủ độ chính xác, độ ổn định và khả năng mở rộng trong các ứng dụng thực tế.

3.1.4 Xây dựng chương trình nhận dữ liệu

Chương trình nhận dữ liệu được phát triển bằng ngôn ngữ Python, thực hiện chức năng chính là đọc và xử lý các dữ liệu gửi từ hệ thống cảm biến, đồng thời lưu trữ dữ liệu đo đạc vào cơ sở dữ liệu InfluxDB và chụp ảnh hiện trường bằng camera để phục vụ phân tích và giám sát hệ thống.

Chương trình sử dụng thư viện pySerial để giao tiếp với cảm biến qua cổng Serial, cấu hình ở tốc độ baud rate 115200 trên cổng /dev/ttyUSB0. Dữ liệu nhận được có định dạng JSON chứa các thông số đo đạc quan trọng như nhiệt độ, pH, nồng độ oxy hòa tan và TDS. Khi nhận dữ liệu, chương trình tiến hành phân tích và kiểm tra tính hợp lệ của gói tin JSON. Trong trường hợp dữ liệu JSON hợp lệ, các giá trị này sẽ được ghi lại vào cơ sở dữ liệu InfluxDB thông qua thư viện influxdb-client với các thông tin cấu hình như URL truy cập (http://localhost:8086), token xác thực và bucket lưu trữ (env_spirulina_data). Để đảm bảo tính ổn định và chính xác trong việc ghi dữ liệu, chương trình bổ sung thêm bước xác minh bằng cách truy vấn lại dữ liệu vừa gửi lên, đảm bảo dữ liệu thực sự đã được ghi vào InfluxDB. Kết quả các bước này đều được ghi lại đầy đủ vào file nhật ký hệ thống (startup.log) với dấu thời gian cụ thể, hỗ trợ việc theo dõi, phân tích và khắc phục sự cố sau này.

Ngoài việc xử lý và lưu trữ dữ liệu cảm biến, chương trình còn tích hợp thêm chức năng giám sát hình ảnh. Camera được quản lý bởi thư viện OpenCV và được cấu hình tự động khởi động khi chương trình bắt đầu. Mỗi khi nhận được dữ liệu cảm biến hợp lệ, hệ thống sẽ chụp một bức ảnh hiện trạng tại thời điểm đó. Ảnh chụp được lưu trữ dưới dạng file ảnh .jpg vào thư mục /home/huy/images, tên file được tạo ra tự động theo thời gian thực (photo_<timestamp>.jpg) giúp thuận tiện cho việc quản lý và truy xuất về sau. Trong quá trình vận hành, nếu chương trình phát hiện

lỗi khi truy cập hoặc đọc dữ liệu từ camera liên tiếp quá 5 lần, hệ thống sẽ tự động giải phóng và khởi động lại camera để đảm bảo khả năng hoạt động ổn định lâu dài. Cấu trúc lập trình theo dạng vòng lặp vô hạn giúp hệ thống luôn trong trạng thái sẵn sàng nhận dữ liệu liên tục theo thời gian thực, đồng thời kèm theo đầy đủ các cơ chế kiểm tra, xử lý lỗi để đảm bảo hệ thống có thể tự phục hồi và duy trì hoạt động ổn định ngay cả khi gặp các tình huống bất thường. Việc kết hợp giữa xử lý dữ liệu cảm biến, lưu trữ và chụp ảnh hiện trường theo thời gian thực đã tạo nên một nền tảng giám sát toàn diện, hiệu quả, giúp đảm bảo an toàn và độ tin cậy cao khi triển khai thực tế.

3.1.5 Cấu trúc và phương thức giao tiếp

Hệ thống giám sát chất lượng nước nuôi tảo được tích hợp nhiều thành phần phần cứng và phần mềm, liên kết chặt chẽ nhằm đảm bảo khả năng theo dõi, phát hiện và cảnh báo kịp thời các sự cố trong quá trình vận hành. Để các thành phần này phối hợp nhịp nhàng và đạt hiệu quả cao, việc thiết kế kiến trúc giao tiếp truyền dữ liệu giữa các thiết bị phải đảm bảo tính hợp lý, đồng bộ, đáp ứng được yêu cầu về độ trễ thấp và độ tin cậy cao.

Cấu trúc giao tiếp trong hệ thống bao gồm: giao tiếp giữa ESP32 và các cảm biến như cảm biến DO S-20, cảm biến pH và TDS thông qua các giao thức UART và I2C; giao tiếp giữa ESP32 và vi điều khiển xử lý trung tâm (ví dụ như Raspberry Pi hoặc máy chủ nội bộ) thông qua giao thức UART hoặc MQTT tùy theo thiết kế hệ thống; giao tiếp giữa ESP32 và camera hoặc module điều khiển relay, các đầu ra điều khiển khác được lập trình thông qua các chân GPIO. Ngoài ra, dữ liệu sau khi xử lý có thể được truyền lên máy chủ để lưu trữ trong cơ sở dữ liệu InfluxDB, đồng thời hiển thị trên nền tảng Node-RED hoặc dashboard web.

3.2 XÂY DỰNG GIAO DIỆN ĐIỀU KHIỂN VÀ LƯU TRỮ

3.2.1 Thiết kế giao diện người dùng bằng NodeRed

Trong hệ thống giám sát chất lượng nước, Node-RED được sử dụng làm công cụ xây dựng giao diện người dùng trực quan, giúp người vận hành có thể dễ dàng theo dõi dữ liệu cảm biến, điều khiển thiết bị và nhận cảnh báo lỗi theo thời gian thực. Giao diện này được triển khai trên tab "Home" trong Node-RED Dashboard, gồm nhiều thành phần được bố trí logic, phục vụ cả mục tiêu giám sát và điều khiển.

Về hiển thị dữ liệu, hệ thống truy vấn dữ liệu từ cơ sở dữ liệu InfluxDB trong khoảng thời gian 24 giờ gần nhất. Dữ liệu các trường quan trọng như oxy hòa tan , pH, nhiệt độ và TDS được xử lý bằng chuỗi node function và biểu diễn dưới dạng biểu đồ đường hoặc đồng hồ đo . Cụ thể, mỗi loại dữ liệu được hiển thị bằng một gauge riêng biệt với ngưỡng màu cảnh báo, giúp người dùng có thể đánh giá nhanh trạng thái hệ thống.

Về chức năng điều khiển, Node-RED cung cấp các nút nhấn (UI button) tương ứng cho từng thiết bị như motor bơm pH, motor sục khí 1 và 2, và đèn chiếu sáng. Các nút ON/OFF gửi dữ liệu MQTT tương ứng đến ESP32 thông qua các topic như STATUS_MOTOR_PH,STATUS_MOTOR_AIR_1,STATUS_AIR_2,STATUS_LIGH. Các trạng thái thiết bị sau khi được bật/tắt sẽ được phản hồi lại qua MQTT và thể hiện trực tiếp bằng đèn LED trên Dashboard , sử dụng màu đỏ và xanh lá để biểu thị trạng thái tắt và bật tương ứng.

Ngoài ra, hệ thống cũng tích hợp biểu mẫu nhập liệu cho phép người dùng thiết lập ngưỡng cảnh báo pH cao và thấp. Các giá trị thiết lập sẽ được lưu trữ trong bộ nhớ luồng và hiển thị trực tiếp trên giao diện. Khi phát hiện giá trị pH vượt ngưỡng hoặc xảy ra sự cố trong hệ thống như không có bọt khí, Node-RED sẽ tự động gửi cảnh báo qua email đến người quản trị để kịp thời xử lý.

Tổng thể, luồng xử lý dữ liệu trong hệ thống được xây dựng theo hướng linh hoạt và dễ mở rộng, kết nối chặt chẽ giữa các thành phần truyền thông, lưu trữ và hiển thị. Mọi dữ liệu đo được, tín hiệu điều khiển và thông báo đều được xử lý theo thời gian thực, giúp hệ thống phản ứng nhanh trước các tình huống bất thường. Giao diện người dùng đóng vai trò trung tâm giám sát và điều phối, đảm bảo quá trình vận hành được theo dõi chặt chẽ, minh bạch và tin cậy.

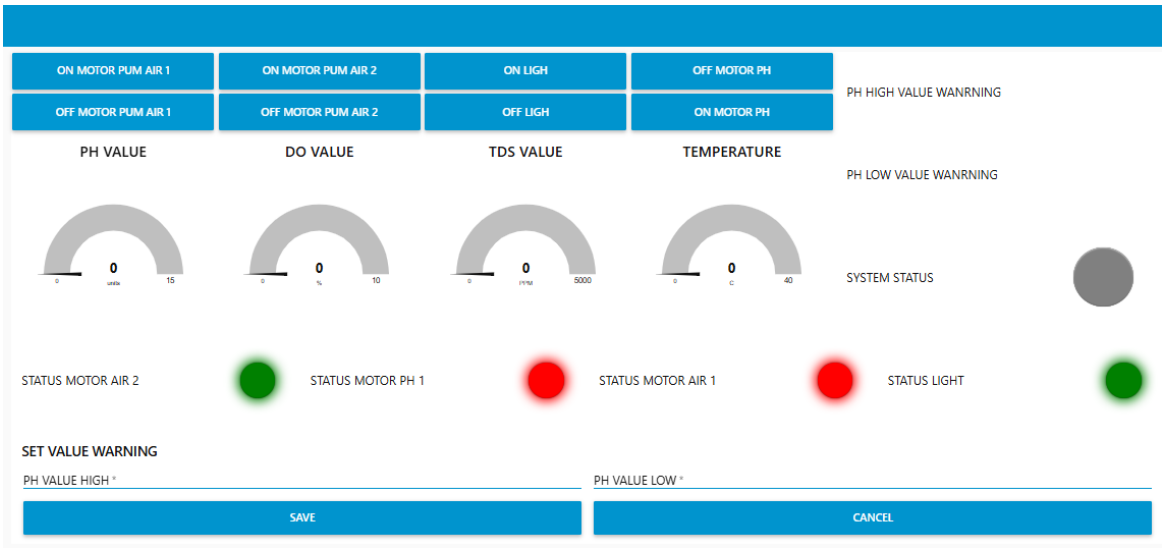
3.2.2 Thiết kế cơ sở dữ liệu InFluxDB

Hệ thống giám sát chất lượng môi trường sử dụng InfluxDB làm nền tảng lưu trữ dữ liệu cảm biến theo thời gian thực. Với đặc tính tối ưu cho dữ liệu chuỗi thời gian, InfluxDB cho phép ghi nhận và truy vấn dữ liệu liên tục, đáp ứng yêu cầu về tốc độ và độ tin cậy trong các ứng dụng IoT. Cấu trúc cơ sở dữ liệu được tổ chức đơn giản, trực quan, giúp việc lưu trữ, truy xuất và hiển thị dữ liệu trở nên linh hoạt và hiệu quả.

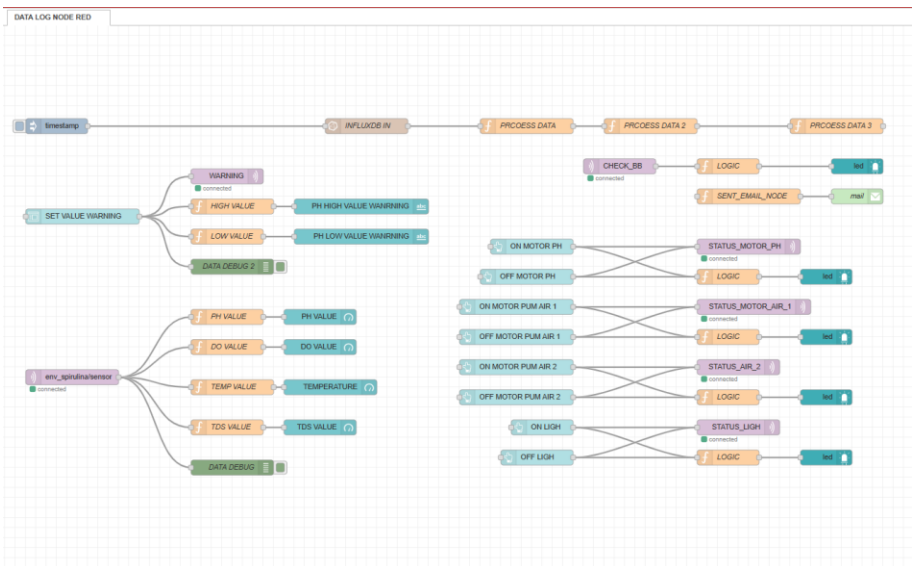
Dữ liệu được truyền từ ESP32 về hệ thống trung tâm dưới dạng các gói JSON chứa thông tin đo đạc gồm giá trị pH, nồng độ oxy hòa tan, nhiệt độ và chỉ số TDS. Sau đó, dữ liệu được ghi trực tiếp vào InfluxDB thông qua API. Mỗi bản ghi được gắn dấu thời gian chính xác, đồng thời lưu trữ dưới dạng các trường số liệu trong bảng đo có tên là “sensor_data”.

Quá trình ghi dữ liệu được xác minh lại thông qua truy vấn kiểm tra mẫu trên cơ sở dữ liệu, giúp đảm bảo tính toàn vẹn và độ tin cậy. Hệ thống còn tích hợp cơ chế ghi nhật ký (log) các hoạt động lưu trữ và cảnh báo khi có lỗi phát sinh trong quá trình giao tiếp với InfluxDB.

Việc sử dụng InfluxDB không chỉ mang lại hiệu quả cao trong việc xử lý dữ liệu cảm biến mà còn cho phép tích hợp mở rộng với các công cụ giám sát như Node-RED và các nền tảng hiển thị như Grafana trong tương lai. Đây là một nền tảng vững chắc để xây dựng hệ thống giám sát thông minh, ổn định và có khả năng mở rộng cho nhiều ứng dụng thực tiễn khác.



Hình 3-6 Giao diện điều khiển bằng NodeRed



Hình 3-7 Kết nối các tính năng hiển thị và điều khiển

Hình 3-8 Dữ liệu chất lượng nước từ bể nuôi

Hình 3-9 Giá trị đọc được từ cảm biến từ ESP32

CHƯƠNG 4. GIA CÔNG MÔ HÌNH VÀ TRIỂN KHAI HỆ THỐNG NUÔI TẢO

4.1 GIA CÔNG MÔ HÌNH

4.1.1 Đặt gia công PCB tại xưởng

Sau khi hoàn thiện thiết kế sơ đồ nguyên lý và bố trí linh kiện trên phần mềm Altium Designer, nhóm tiến hành xuất bộ file sản xuất PCB bao gồm: tệp Gerber, tệp khoan (Drill), và sơ đồ định vị lớp (Layer Stackup). Các tệp được nén lại thành một gói duy nhất định dạng .zip để gửi đến nhà sản xuất.

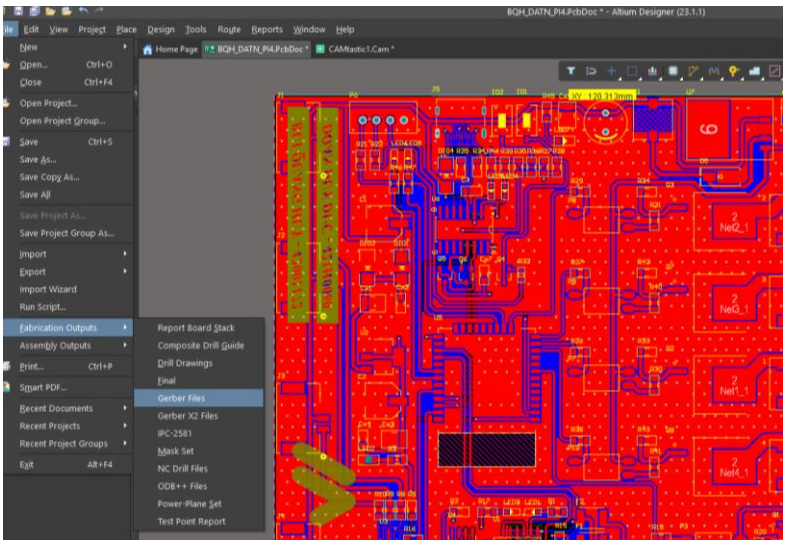
Để đảm bảo chất lượng gia công và tối ưu chi phí, nhóm lựa chọn dịch vụ sản xuất mạch in từ nền tảng trực tuyến JLCPCB. Đây là một trong các nhà cung cấp dịch vụ gia công PCB phổ biến toàn cầu, hỗ trợ đặt hàng trực tuyến, xem trước tệp Gerber, tùy chọn thông số kỹ thuật linh hoạt và thời gian sản xuất ngắn.

Quy trình đặt hàng được thực hiện theo các bước như sau:

a) Chuẩn bị file sản xuất

Từ giao diện thiết kế của Altium Designer, nhóm sử dụng chức năng Fabrication Outputs để xuất file Gerber và file khoan. Cấu hình các lớp bao gồm:

- Top layer, Bottom layer
- Top/Bottom Soldermask
- Top/Bottom Silkscreen
- Mechanical Layer
- Drill drawing



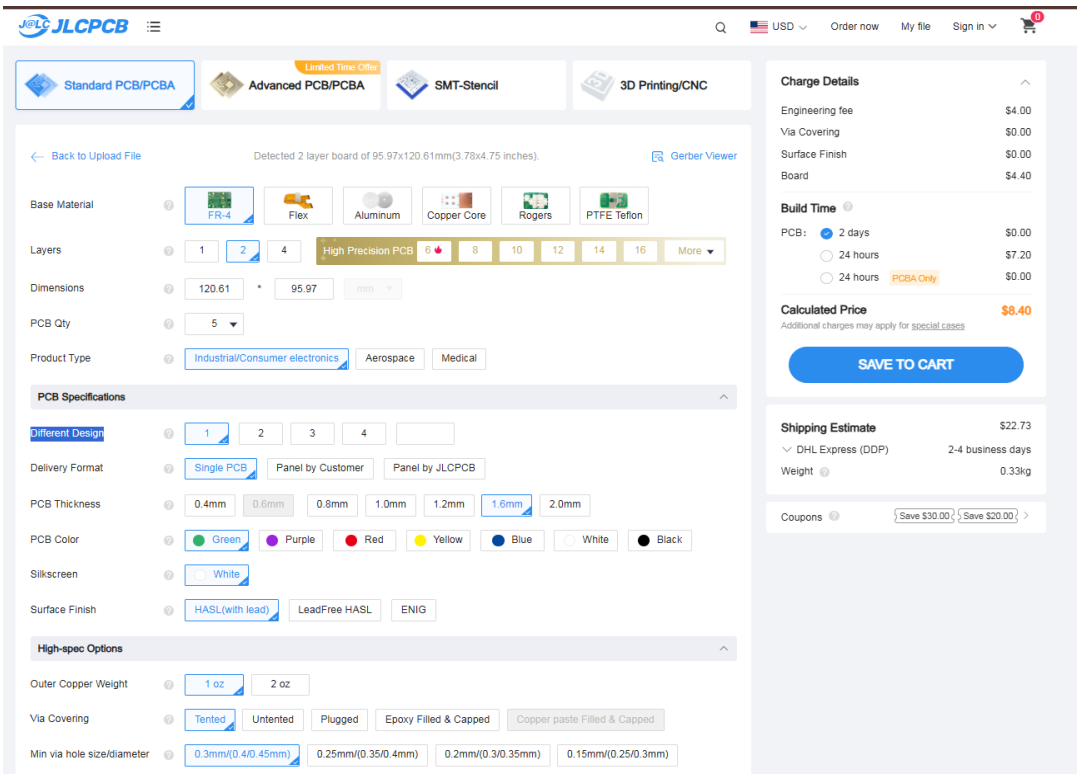
Sau khi kiểm tra, toàn bộ tệp tin được nén thành một tệp .zip theo đúng định dạng yêu cầu của nhà sản xuất.

b) Tải tệp và cấu hình đơn hàng trên nền tảng JLCPCB

Truy cập trang https://jlcpcb.com và chọn tính năng "Order Now". Sau đó, tệp .zip được tải lên hệ thống và hiển thị bản xem trước (Gerber Viewer) để kiểm tra trực quan sơ đồ mạch trước khi đặt hàng.

Các thông số kỹ thuật của mạch được cấu hình như sau:

- Số lớp : 2 layers
- Độ dày bo mạch: 1.6 mm
- Độ dày lớp đồng: 1 oz (35 µm)
- Màu soldermask: Xanh lá (Green)
- Xử lý bề mặt: HASL (Hot Air Solder Leveling)
- Kích thước mạch (D×R):Theo thiết kế thực tế (~121 mm × 9.6 mm)
- Số lượng đặt: 5 bản



Hình 4-1 Lựa chọn cấu hình mạch in

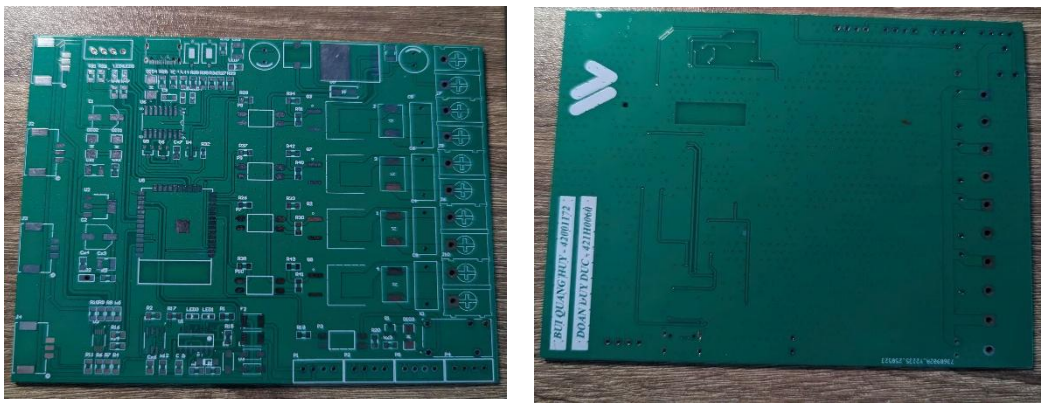
Sau khi xác nhận thông tin cấu hình, nhóm kiểm tra lại định tuyến các lớp mạch, đường tín hiệu, lỗ khoan và lớp in chữ để tránh sai sót trước khi chuyển sang bước thanh toán.

c) Đặt hàng, theo dõi và kiểm tra thành phẩm

Sau khi hoàn tất cấu hình, nhóm tiến hành thanh toán qua PayPal hoặc thẻ tín dụng quốc tế. Đơn vị vận chuyển được lựa chọn là DHL để đảm bảo thời gian giao hàng nhanh (trong khoảng 5–7 ngày làm việc). Hệ thống cung cấp mã theo dõi đơn hàng để cập nhật trạng thái vận chuyển.

Sau khi nhận PCB thành phẩm, nhóm tiến hành kiểm tra sơ bộ các yếu tố kỹ thuật như:

- Độ sắc nét của lớp đồng và lớp in chữ
- Vị trí lỗ khoan đúng theo thiết kế
- Không có dấu hiệu bong tróc lớp soldermask hoặc chập mạch



Hình 4-2 Mạch in thực tế.

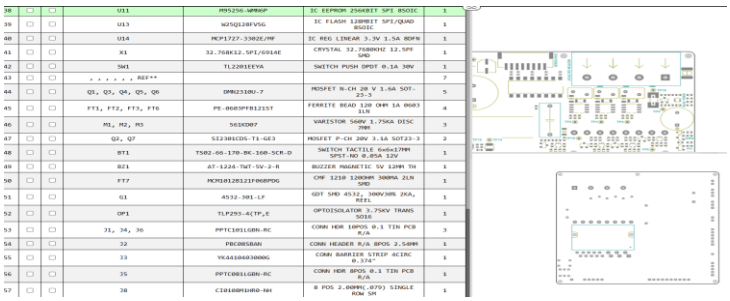
Bo mạch sau khi kiểm tra được xác nhận đạt yêu cầu và sẵn sàng cho giai đoạn hàn linh kiện.

4.1.2 Hàn linh kiện

a) Chuẩn bị linh kiện và thiết bị

Sau khi nhận được mạch in PCB đã gia công hoàn chỉnh, công đoạn tiếp theo là chuẩn bị đầy đủ các linh kiện điện tử để tiến hành lắp ráp mạch. Việc chuẩn bị linh kiện là bước quan trọng nhằm đảm bảo quá trình hàn diễn ra liên tục, chính xác, tránh thiếu sót hoặc sai sót gây ảnh hưởng đến hiệu năng mạch.

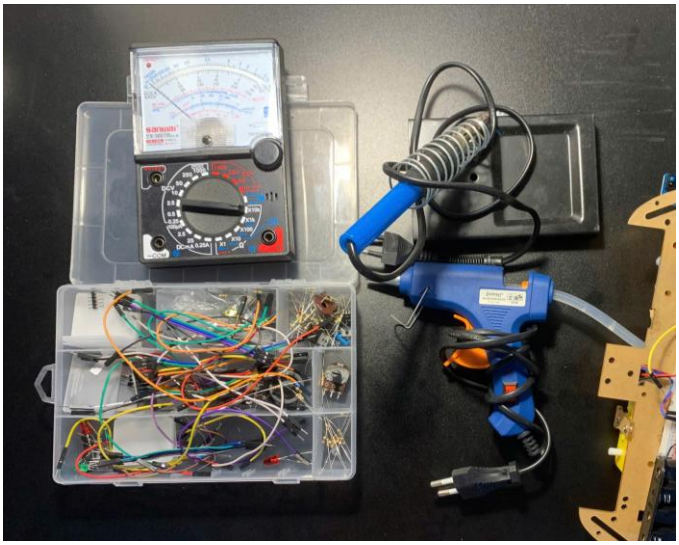
Trước tiên, nhóm tiến hành đối chiếu danh sách BOM (Bill of Materials) với sơ đồ bố trí linh kiện (PCB Layout) để xác định chính xác các loại linh kiện cần sử dụng, bao gồm: điện trở, tụ điện, diode, transistor, IC, vi điều khiển, header, jack nguồn, module giao tiếp và các đầu nối (terminal block). Mỗi loại linh kiện được sắp xếp vào các hộp riêng hoặc được dán nhãn để dễ nhận diện.



Hình 4-3 Danh sách linh kiện.

Linh kiện được kiểm tra sơ bộ bằng đồng hồ vạn năng để xác định đúng giá trị và tình trạng hoạt động trước khi hàn. Một số linh kiện như tụ hóa, diode, LED,… có phân cực, vì vậy cần kiểm tra kỹ chiều lắp đặt để tránh lắp ngược gây hư hỏng hoặc đoản mạch sau khi cấp nguồn.

Bên cạnh đó, nhóm chuẩn bị các dụng cụ và vật tư cần thiết cho quá trình hàn, bao gồm: mỏ hàn nhiệt độ ổn định (60–80 W), thiếc hàn, nhíp, giấy lau, keo tản nhiệt băng keo cách điện và bảng gắn linh kiện.



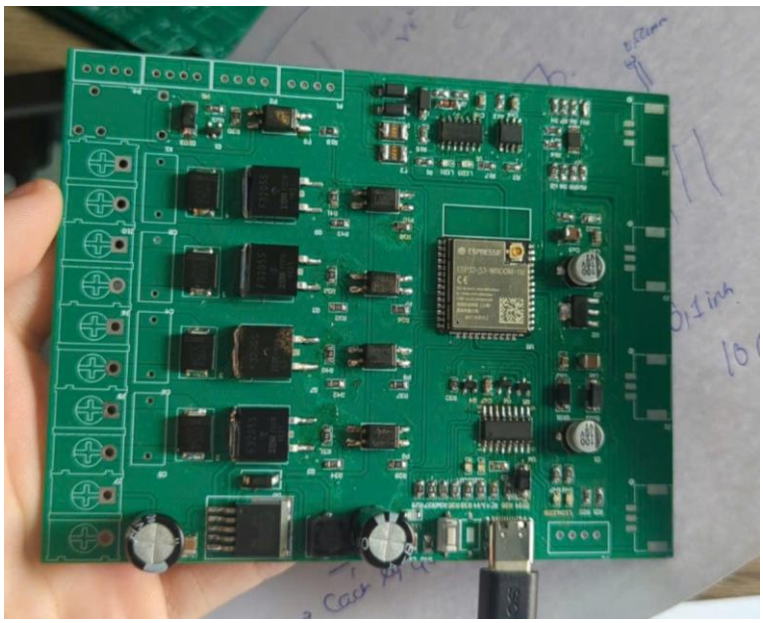
Hình 4-4 Chuẩn bị linh kiện và thiết bị

Cuối cùng, vị trí làm việc được bố trí ngăn nắp, có đèn chiếu sáng rõ ràng và mặt bàn chống tĩnh điện để đảm bảo an toàn cho các linh kiện bán dẫn nhạy cảm như vi điều khiển, IC logic. Việc chuẩn bị đầy đủ và cẩn thận giúp giảm thời gian hàn, tăng độ chính xác và hạn chế các lỗi kỹ thuật trong quá trình lắp ráp thủ công.

b) Hàn linh kiện vào mạch

Sau khi chuẩn bị đầy đủ linh kiện và dụng cụ, nhóm tiến hành hàn thủ công các linh kiện lên bo mạch. Quá trình hàn tuân theo nguyên tắc từ linh kiện thấp đến cao, từ trung tâm ra ngoài, nhằm thuận tiện thao tác và đảm bảo bố trí chính xác.

Các linh kiện như điện trở, tụ gốm, diode được hàn trước, tiếp theo là IC, MOSFET, relay, jack nguồn và các chân header. Trong quá trình hàn, mỏ hàn được điều chỉnh ở mức nhiệt phù hợp (khoảng 320–350 °C), đảm bảo chân hàn bám chắc, không dư thiếc và không gây chập mạch. Đối với các linh kiện phân cực như diode, tụ hóa, LED…, nhóm kiểm tra chiều lắp đặt cẩn thận trước khi cố định.



Hình 4-5 Mạch hoàn thiện linh kiện

Sau khi hoàn tất, bo mạch được vệ sinh bằng cồn isopropyl để loại bỏ dư flux và kiểm tra bằng đồng hồ đo thông mạch. Các vị trí hàn được rà soát kỹ để phát hiện lỗi như hàn nguội, nối tắt hoặc pad chưa đủ thiếc. Mạch sau đó sẵn sàng cho bước cấp nguồn và kiểm thử chức năng.

4.1.3 Lắp ráp hoàn thiện thiết bị.

Sau khi hàn linh kiện và kiểm tra chức năng từng phần tử mạch, quá trình lắp ráp hoàn thiện thiết bị được tiến hành. Bo mạch điều khiển chính, bao gồm ESP32-S3, các module cảm biến (TDS, RS485), relay và mạch nguồn LM2596 được cố định chắc chắn vào hộp kỹ thuật thông qua chân đế nhựa và ốc cách điện. Các dây tín

hiệu và dây nguồn được bó gọn bằng dây rút và ống gen nhằm đảm bảo tính thẩm mỹ và giảm nhiễu tín hiệu.

Mạch được lắp kèm với Raspberry Pi 4, ổ cứng lưu trữ, camera USB và các kết nối giao tiếp như cổng nguồn, cổng USB, RS485... Tất cả được định tuyến và gắn cố định trong hộp, đảm bảo không bị xê dịch trong quá trình vận hành. Sau cùng, hộp được đóng kín, kiểm tra lại các điểm tiếp xúc và tiến hành cấp nguồn thử nghiệm để đảm bảo toàn bộ hệ thống hoạt động ổn định trước khi triển khai ra môi trường thực tế.

Hình 4-6 Hoàn thiện thiết bị

4.2 TRIỂN KHAI MÔ HÌNH HỆ THỐNG NUÔI TẢO

4.2.1 Điều kiện triển khai mô hình

Mô hình được xây dựng trong không gian nhà kho khép kín, mái lợp bằng tôn nhựa xuyên sáng để tận dụng ánh sáng tự nhiên, đồng thời hạn chế cường độ bức xạ quá cao vào thời điểm trưa vốn đặc trưng của mùa hè ở Việt Nam. Không gian khép kín giúp kiểm soát tốt hơn các yếu tố nhiệt độ môi trường và giảm thiểu tác động từ gió, bụi hoặc sinh vật ngoại lai.

Hình 4-7 Tham khảo điều kiện nuôi.

Bể nuôi sử dụng là thùng nhựa tròn dung tích 200 lít. Nguồn nước sử dụng là nước máy đã lắng lọc và được khử clo. Dưỡng chất được pha thủ công dựa trên công thức chuẩn hóa với các thành phần như $NaHCO_3$, K_2HPO_4, $MgSO_4$ và vi lượng, đảm bảo pH và độ khoáng thích hợp cho giống tảo Spirulina platensis sinh trưởng.

4.2.2 Lắp đặt và cấu hình hệ thống kỹ thuật

Hệ thống phần cứng bao gồm:

- Vi điều khiển ESP32-S3 kết nối các cảm biến (DO, pH, nhiệt độ, TDS).
- Cảm biến DO-S20 giao tiếp RS485; cảm biến TDS analog của DFRobot.
- Bộ điều khiển relay để điều khiển máy sục khí, bơm, đèn chiếu sáng.

- Camera chụp ảnh bề mặt bể định kỳ 5 phút/lần, kết nối Raspberry Pi.
- Dữ liệu được truyền về Raspberry Pi qua giao thức MQTT, hiển thị qua Node-RED và lưu vào InfluxDB.

Các cảm biến được cố định tại vị trí trung tâm bể với độ sâu tiêu chuẩn 15–20 cm để đảm bảo đo đúng vùng phân bố sinh khối tảo. Máy sục khí và đèn chiếu sáng 3000K được lập trình bật/tắt theo lịch cố định.

4.2.3 Quy trình nuôi thử nghiệm

Giai đoạn đầu tiên là vận hành hệ thống trong điều kiện không có tảo, nhằm kiểm tra sự ổn định của các kết nối, độ chính xác của dữ liệu và khả năng hoạt động liên tục của hệ thống. Sau một tuần thử nghiệm, dữ liệu từ các cảm biến cho thấy độ nhiễu thấp, kết nối ổn định, không có lỗi truyền thông.
Sau đó, bể được vệ sinh lại, bổ sung dưỡng chất và tiến hành thả giống tảo Spirulina. Mật độ tảo ban đầu khoảng 0,1 g/L. Trong suốt quá trình nuôi, hệ thống hoạt động liên tục 24/7 và thu thập dữ liệu theo thời gian thực.

Dữ liệu được ghi nhận theo 3 giai đoạn chính:

- Trước khi thả giống (nước sạch): Ghi nhận thông số nền như pH, DO, TDS trong môi trường chưa có sinh khối.
- Giai đoạn sinh trưởng: Các thông số thay đổi theo thời gian do quá trình quang hợp và tiêu thụ khoáng chất.
- Giai đoạn xảy ra sự cố: Một số thời điểm ghi nhận hiện tượng tảo chết cục bộ (DO giảm mạnh, pH dao động), giúp xác minh tính hữu ích của hệ thống trong cảnh báo sớm.

Hệ thống camera kết hợp xử lý ảnh (YOLOv5) cho phép nhận biết mật độ bọt khí và màu sắc bề mặt, phục vụ phát hiện bất thường hoặc đánh giá sức khỏe tảo.

4.2.4 Đánh giá sơ bộ

Mô hình thử nghiệm bước đầu đã chứng minh được tính ổn định và hiệu quả vận hành của hệ thống kỹ thuật trong điều kiện nuôi tảo thực tế. Dữ liệu thu nhận từ các cảm biến (pH, DO, TDS, nhiệt độ) có độ chính xác cao, phản ánh đúng các biến động môi trường xảy ra trong suốt quá trình sinh trưởng của tảo. Hệ thống vận hành hoàn toàn tự động, cho phép giám sát liên tục mà không cần can thiệp thủ công, góp phần giảm đáng kể khối lượng công việc giám sát truyền thống.

Bên cạnh dữ liệu định lượng, hệ thống còn tích hợp camera chụp ảnh định kỳ và xử lý ảnh bằng mô hình học máy, hỗ trợ phát hiện các dấu hiệu bất thường như thay đổi màu sắc bề mặt hoặc giảm mật độ bọt khí – những chỉ số liên quan trực tiếp đến tình trạng sức khỏe của sinh khối tảo. Việc kết hợp giữa dữ liệu cảm biến và thông tin hình ảnh đã tạo thành một hệ giám sát toàn diện, hỗ trợ phân tích đa chiều và có khả năng mở rộng cho các ứng dụng dự báo.

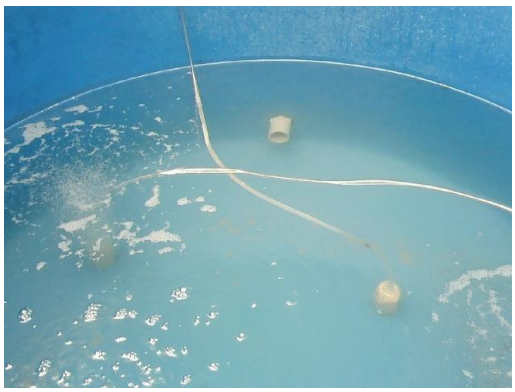
Kết quả triển khai cho thấy hệ thống có tiềm năng ứng dụng cao trong các mô hình nuôi tảo quy mô vừa và nhỏ. Với chi phí triển khai hợp lý, mức độ tự động hóa cao và khả năng tích hợp trí tuệ nhân tạo, hệ thống phù hợp với xu hướng hiện đại hóa công nghệ nuôi trồng và phát triển các giải pháp xanh bền vững trong nông nghiệp công nghệ cao.

CHƯƠNG 5. KIỂM ĐỊNH VÀ KẾT QUẢ

5.1 KẾT QUẢ THỰC NGHIỆM

a) Quá trình nuôi và vận hành hệ thống

Mô hình nuôi tảo Spirulina được triển khai trong một bể nhựa kín dung tích 200 lít, đặt trong nhà kho mái tôn lấy sáng tự nhiên. Môi trường được chuẩn bị bằng cách pha sẵn dung dịch dưỡng chất theo công thức chuẩn ($NaHCO_3$, Urea, $MgSO_4$...) với nguồn nước lọc sạch, sau đó điều chỉnh pH và TDS phù hợp (pH ~7.0, TDS ~1,000 ppm) trước khi tiến hành thả giống ở hình 5-1.



Hình 5-1 Ngày đầu tiên thả nuôi giống tảo.

Trong suốt 15 ngày nuôi liên tục, hệ thống vận hành ổn định với các thành phần:

- Sục khí liên tục 24/7 bằng bơm sục khí và đá sủi, đảm bảo tảo luôn luân chuyển và hấp thụ được dưỡng chất.
- Cảm biến đo pH, DO, TDS, nhiệt độ truyền dữ liệu qua ESP32 về Raspberry Pi.
- Dữ liệu hiển thị real-time trên dashboard Node-RED, đồng thời lưu vào InfluxDB.

- Hệ thống bổ sung dưỡng chất tự động được kích hoạt nếu TDS hoặc pH giảm ngoài ngưỡng cài đặt.

Hình 5-2 Sau 15 ngày thả giống tảo.

Hình 5-2 thể hiện kết quả giám sát cho thấy môi trường ổn định, tảo sinh trưởng đều, nước chuyển từ màu xanh nhạt sang xanh đậm rõ rệt, độ trong giảm dần phản ánh sự gia tăng mật độ sinh khối.

b) Quy trình lọc và thu tảo sinh khối

Sau 15 ngày, hệ thống tạm dừng để tiến hành lọc và thu sinh khối:

- Sử dụng lưới lọc chuyên dụng 50–100 µm, tiến hành lọc thủ công bằng van xả đáy.
- Nước được dẫn qua lưới lọc, giữ lại phần tảo có kích thước lớn và mật độ cao.
- Lượng nước sau lọc (~70–75%) được bơm tuần hoàn trở lại bể, nhằm tái sử dụng môi trường, giữ lại tảo chưa đạt chất lượng và làm nền cho đợt nuôi tiếp theo.

c) Đánh giá chất lượng tảo sau lọc

Tảo được phân chia rõ rệt theo hai mức chất lượng:

- Tảo đạt chuẩn: màu xanh đậm, sợi dài, lắng nhanh – chiếm khoảng 25–30% tổng thể. Đây là phần được giữ lại để sấy hoặc làm mẫu phân tích.
- Tảo chưa đạt: màu nhạt, sợi ngắn, phân tán – thường là sinh khối non, chưa đủ chu kỳ. Phần này được giữ lại trong nước tuần hoàn để tiếp tục phát triển.

Ngoài ra, hệ thống camera và AI hỗ trợ ghi nhận màu nước thay đổi dần theo thời gian, hỗ trợ đánh giá trực quan độ phát triển và quyết định thời điểm thu tảo hợp lý.

d) Ưu điểm và kết luận

Việc tái sử dụng nước và sinh khối non không chỉ giúp tiết kiệm nguồn tài nguyên mà còn hỗ trợ duy trì giống tảo khỏe, không cần mua mới, đồng thời rút ngắn chu kỳ thiết lập mỗi mẻ. Điều này góp phần vào mục tiêu phát triển bền vững và kinh tế tuần hoàn trong mô hình nuôi tảo.

5.1.1 Kết quả phát hiện bọt khí bằng mô hình học máy

Trong hệ thống giám sát nuôi tảo, việc đảm bảo hoạt động ổn định của máy sục khí là điều tối quan trọng. Để tự động hóa giám sát tình trạng này, đề tài đã ứng dụng thị giác máy tính kết hợp mô hình học sâu – cụ thể là YOLOv5 – để nhận diện bọt khí xuất hiện trên bề mặt bể nuôi. Mô hình được huấn luyện từ tập dữ liệu ảnh thực tế được thu thập trong quá trình triển khai mô hình (Hình 5-3).

a) Quy trình thực hiện:

- Hình ảnh được camera gắn trên bể chụp lại theo chu kỳ 5 phút/lần, sau đó truyền về Raspberry Pi 4.
- Ảnh đầu vào được đưa vào mô hình YOLOv5 đã huấn luyện.
- Mô hình trả về kết quả gồm vị trí (bounding box) và độ tin cậy (confidence score) của vùng được nhận diện là bọt khí.

Hình 5-3 Kết quả trả về từ mô hình học máy.

b) Kết quả định lượng:

Sau quá trình huấn luyện mô hình YOLOv5 để phát hiện bọt khí trong hệ thống nuôi tảo, kết quả kiểm thử trên tập dữ liệu cho thấy hiệu suất như sau:

Chỉ số	Giá trị	Diễn giải
Precision (P)	0.589	Khoảng 58.9% vùng dự đoán là bọt khí là đúng.
Recall (R)	0.529	Mô hình phát hiện được 52.9% số vùng có bọt khí thực sự tồn tại.
mAP50	0.580	Trung bình độ chính xác ở ngưỡng IOU ≥ 0.5 khá tốt với dữ liệu thực tế.
mAP50-95	0.265	Chỉ số mAP toàn diện, đánh giá chất lượng phân biệt mô hình ở nhiều IOU.
Số ảnh kiểm thử	24	Số ảnh được dùng để đánh giá hiệu năng mô hình sau khi huấn luyện.
Tổng số đối tượng	157	Tổng số nhãn "Bubble" trong tập validation.

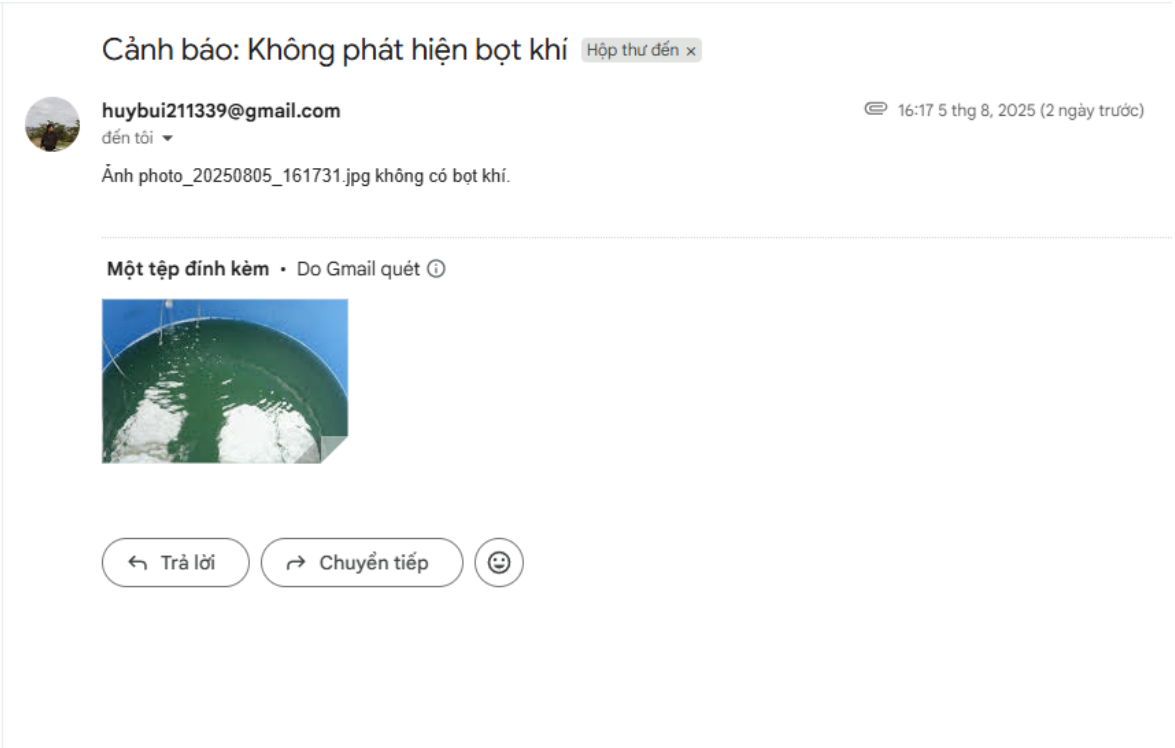
Bảng 5-1 Bảng kết quả huấn luyện mô hình.

c) Kết quả vận hành:

Trong quá trình triển khai thực tế, mô hình phát hiện bọt khí đã cho thấy khả năng hoạt động ổn định trong điều kiện ánh sáng tự nhiên trong nhà. Các tình huống được mô phỏng và kiểm nghiệm bao gồm cả trạng thái hoạt động bình thường và sự cố hệ thống.

- **Phát hiện bất thường**

Ở Hình 5 4 khi camera ghi nhận hình ảnh không xuất hiện vùng nào được phân loại là "Bubble" với độ tin cậy vượt ngưỡng cấu hình. hệ thống sẽ đánh giá khả năng cao là máy sục khí đang gặp sự cố hoặc ngừng hoạt động. Trong trường hợp này, một cảnh báo tức thời được gửi tự động đến người quản lý thông qua email để kịp thời xử lý.



Hình 5-4 Cảnh báo về email.

- **Nhận diện trạng thái bình thường**

Ở trạng thái hoạt động bình thường, mô hình đã phát hiện chính xác nhiều vùng chứa bọt khí tại vị trí đầu ra khí từ máy sục. Các khung phát hiện (bounding boxes) có độ tin cậy cao, dao động từ 0.75 đến 0.90, phản ánh khả năng nhận diện tốt trong điều kiện quan sát thuận lợi.

- **Đánh giá tổng thể**

Ưu điểm:

- Hệ thống hoạt động hoàn toàn tự động, không yêu cầu can thiệp thủ công trong quá trình giám sát.
- Có khả năng phát hiện sớm sự cố sục khí – một yếu tố quan trọng ảnh hưởng trực tiếp đến hàm lượng oxy hòa tan và sự phát triển của tảo.
- Tích hợp tốt với hệ thống IoT, hỗ trợ cảnh báo thời gian thực.

Hạn chế:

- Độ chính xác có thể suy giảm khi xuất hiện các yếu tố gây nhiễu như bóng đổ mạnh, nước bị mờ, hoặc bọt khí nhỏ, phân bố không đồng đều.
- Chưa xử lý tốt ánh sáng yếu hoặc hiện tượng phản chiếu từ thành bể.
- Khả năng mở rộng:
- Việc mở rộng tập dữ liệu huấn luyện, bổ sung ảnh từ nhiều điều kiện môi trường và cải thiện gán nhãn sẽ giúp tăng độ chính xác.

CHƯƠNG 6. KẾT LUẬN

6.1 Tổng kết đề tài

Sau quá trình nghiên cứu nghiêm túc, triển khai thực tiễn và thử nghiệm nhiều lần, đề tài "Thiết kế hệ thống IoT thông minh cho nuôi trồng tảo – Giải pháp Blue Carbon" đã đạt được toàn bộ mục tiêu đề ra ban đầu. Hệ thống được thiết kế hoàn chỉnh, tích hợp chặt chẽ giữa các thành phần phần cứng, phần mềm, truyền thông và thị giác máy tính.

Hệ thống có khả năng giám sát liên tục các chỉ số môi trường quan trọng như pH, nhiệt độ, DO và TDS; truyền dữ liệu theo thời gian thực thông qua giao thức MQTT đến máy chủ nội bộ (Raspberry Pi), lưu trữ bằng InfluxDB, hiển thị trực quan qua Node-RED Dashboard, đồng thời kết hợp mô hình học sâu (YOLOv5) để phân tích hình ảnh và phát hiện sự cố như mất bọt khí – dấu hiệu cảnh báo sớm cho việc hư hỏng máy sục khí.

Bên cạnh hệ thống kỹ thuật, nhóm thực hiện cũng đã triển khai mô hình nuôi tảo thực nghiệm, theo dõi trong suốt chu trình 15 ngày, thu thập dữ liệu và đánh giá tính ổn định, khả năng mở rộng của mô hình.

Có thể khẳng định, đề tài đã xây dựng thành công một mô hình giám sát môi trường thông minh có khả năng ứng dụng cao trong thực tiễn, đồng thời thể hiện được tinh thần tiên phong trong việc đưa công nghệ IoT và AI phục vụ cho lĩnh vực nông nghiệp xanh – một xu hướng phát triển tất yếu của thế kỷ XXI.

6.2 Hạn chế

Mặc dù đạt được kết quả tích cực, đề tài vẫn còn một số hạn chế nhất định:

- Mô hình học sâu YOLOv5 hoạt động tốt trong điều kiện ánh sáng tiêu chuẩn, nhưng gặp khó khăn khi ánh sáng yếu hoặc có phản xạ mặt nước.

- Cảm biến TDS và DO cần hiệu chuẩn thủ công sau thời gian dài sử dụng.
- Cảnh báo hệ thống mới dừng ở mức ngưỡng tĩnh, chưa kết hợp thuật toán điều khiển phản hồi tự động.
- Hệ thống chưa được thử nghiệm ngoài trời hoặc trong điều kiện khắc nghiệt, nên độ ổn định lâu dài chưa được kiểm chứng toàn diện.

6.3 Định hướng phát triển

Đề tài mở ra một hướng nghiên cứu rất tiềm năng, có thể tiếp tục phát triển theo các định hướng sau:

- Tối ưu và nâng cấp mô hình AI: Huấn luyện lại với tập dữ liệu phong phú hơn, bao gồm ảnh đêm, ảnh mờ, phản chiếu mạnh, giúp mô hình trở nên linh hoạt và mạnh mẽ hơn trong nhiều điều kiện môi trường.
- Tích hợp thêm cảm biến môi trường nâng cao như đo ánh sáng, CO_2, NH_3, độ đục để nâng cấp hệ thống thành một trạm giám sát môi trường thủy sinh toàn diện.
- Tự động hóa điều khiển bằng thuật toán ra quyết định thông minh: hệ thống không chỉ cảnh báo mà còn có thể tự động bơm dưỡng chất, điều chỉnh đèn quang hợp, hoặc tối ưu chu kỳ sục khí theo trạng thái thực tế.
- Chuyển đổi thành thiết bị thương mại hóa, đóng gói thành module gọn nhẹ, có thể triển khai tại các hợp tác xã, cơ sở sản xuất tảo vừa và nhỏ, giúp phổ cập hóa công nghệ vào đời sống thực tiễn.
- Kết nối lên Cloud: đưa dữ liệu lên nền tảng như Firebase, Azure IoT Hub hoặc AWS IoT để hỗ trợ phân tích quy mô lớn và quản lý từ xa.

CHƯƠNG 7. TÀI LIỆU THAM KHẢO

Tiếng Anh

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective. MIT Press.*

Nielsen, M. (2015). *Neural Networks and Deep Learning.*

PHỤ LỤC A

Phụ lục 1: Code đọc cảm biến ESP32.

```
#include <Arduino.h>
#include "ADS1X15.h"
#include "esp_task_wdt.h"
#include <ArduinoJson.h>

#define RXD2 36
#define TXD2 37

// IO điều khiển
#define IO_MOTOR_PH     4
#define IO_MOTOR_AIR_1  5
#define IO_AIR_2        6
#define IO_LIGHT        7

#define VREF 5.0
#define SCOUNT 30

ADS1115 ADS(0x48);
byte ByteArrayDO[250];
byte MSG_READ[] = {0x01, 0x03, 0x00, 0x00, 0x00, 0x2A, 0xC4, 0x15};

// Biến cảm biến
float oxi_ = 0.0;
float temperature = 0.0;
float decimalValue = 0.0;
```

```
float tdsValue = 0;
float adc1 = 0, f = 1.0;

TaskHandle_t sensorTaskHandle;

String inputString = "";

// Gửi chuỗi JSON dữ liệu cảm biến
void sendJSON()
{
  char jsonBuffer[256];
  snprintf(jsonBuffer, sizeof(jsonBuffer),
          "{\"oxi\":%.2f,\"ph\":%.3f,\"temp\":%.2f,\"tss\":%.3f}",
          oxi_, decimalValue, temperature, tdsValue);
  Serial.println(jsonBuffer);
}

// Gửi yêu cầu đến cảm biến DO (MODBUS)
void sendRequest(const byte *request, size_t len)
{
  for (size_t i = 0; i < len; ++i)
  {
    Serial2.write(MSG_READ[i]);
    Serial2.flush();
  }
}

// Nhận phản hồi từ cảm biến DO
```

```
void receiveResponse(byte *byteArray, int maxLen)
{
  int index = 0;
  unsigned long start = millis();
  while ((millis() - start < 1000) && index < maxLen)
  {
    if (Serial2.available())
    {
      byteArray[index++] = Serial2.read();
    }
  }
}

// Xử lý dữ liệu DO nhận được
bool processDOData()
{
  for (int i = 0; i < 5; i++)
  {
    sendRequest(MSG_READ, sizeof(MSG_READ));
    receiveResponse(ByteArrayDO, sizeof(ByteArrayDO));
    if (ByteArrayDO[0] == 0x01 && ByteArrayDO[1] == 0x03 &&
ByteArrayDO[2] == 0x54)
    {
      word oxi = ((ByteArrayDO[43] << 8) & 0xFF00) | ByteArrayDO[44];
      word temp = (ByteArrayDO[37] << 8) | ByteArrayDO[38];
      word phval = (ByteArrayDO[79] << 8) | ByteArrayDO[80];

      temperature = float(temp) * 0.01;
```

```
      decimalValue = float(phval) * 0.001;
      oxi_ = oxi * 0.01;
      return true;
    }
    vTaskDelay(1000 / portTICK_PERIOD_MS);
  }
  return false;
}

// Task đo cảm biến định kỳ
void sensor_task(void *param)
{
  for (;;)
  {
    esp_task_wdt_reset();

    digitalWrite(16, 1);
    vTaskDelay(30000 / portTICK_PERIOD_MS); // bật nguồn cảm biến

    bool ok = processDOData();
    digitalWrite(16, 0);

    int16_t val_1 = ADS.readADC(1);
    adc1 = val_1 * f;
    float compensationCoefficient = 1.0 + 0.02 * (temperature - 25.0);
    float compensationVolatge = adc1 / compensationCoefficient;
    tdsValue = (133.42 * pow(compensationVolatge, 3) - 255.86 *
pow(compensationVolatge, 2) + 857.39 * compensationVolatge) * 0.5;
```

```
    if (ok)
    {
      sendJSON();
    }

    vTaskDelay(30000 / portTICK_PERIOD_MS); // chờ 30s nữa trước lần đo tiếp
theo
  }
}

// Xử lý lệnh JSON từ Serial và phản hồi IO đang bật
void handleJsonCommand(const String &input)
{
  StaticJsonDocument<128> doc;
  DeserializationError err = deserializeJson(doc, input);

  if (err)
  {
    Serial.print("[LỖI] JSON không hợp lệ: ");
    Serial.println(err.c_str());
    return;
  }

  String ioOn = "";

  if (doc.containsKey("STATUS_MOTOR_PH"))
  {
```

```
  bool on = doc["STATUS_MOTOR_PH"].as<String>() == "1";
  digitalWrite(IO_MOTOR_PH, on ? LOW : HIGH);
  if (on) ioOn += "IO_MOTOR_PH, ";
}

if (doc.containsKey("STATUS_MOTOR_AIR_1"))
{
  bool on = doc["STATUS_MOTOR_AIR_1"].as<String>() == "1";
  digitalWrite(IO_MOTOR_AIR_1, on ? LOW : HIGH);
  if (on) ioOn += "IO_MOTOR_AIR_1, ";
}

if (doc.containsKey("STATUS_AIR_2"))
{
  bool on = doc["STATUS_AIR_2"].as<String>() == "1";
  digitalWrite(IO_AIR_2, on ? LOW : HIGH);
  if (on) ioOn += "IO_AIR_2, ";
}

if (doc.containsKey("STATUS_LIGH"))
{
  bool on = doc["STATUS_LIGH"].as<String>() == "1";
  digitalWrite(IO_LIGHT, on ? LOW : HIGH);
  if (on) ioOn += "IO_LIGHT, ";
}

 Serial.println("[INFO] Đã xử lý lệnh điều khiển IO.");
```

```
  if (ioOn.length() > 0)
  {
    ioOn.remove(ioOn.length() - 2); // xóa dấu ", " cuối
    Serial.print("[TRẠNG THÁI IO] IO đã bật: ");
    Serial.println(ioOn);
  }
  else
  {
    Serial.println("[TRẠNG THÁI IO] Không có IO nào đang bật.");
  }
}

void setup()
{
  Serial.begin(9600);
  Serial2.begin(9600, SERIAL_8N1, RXD2, TXD2);

  // GPIO cấp nguồn cho cảm biến DO
  pinMode(16, OUTPUT);
  digitalWrite(16, 1);

  // IO điều khiển
  pinMode(IO_MOTOR_PH, OUTPUT);
  pinMode(IO_MOTOR_AIR_1, OUTPUT);
  pinMode(IO_AIR_2, OUTPUT);
  pinMode(IO_LIGHT, OUTPUT);

  // Mặc định tất cả IO tắt (HIGH)
```

```
  digitalWrite(IO_MOTOR_PH, HIGH);
  digitalWrite(IO_MOTOR_AIR_1, HIGH);
  digitalWrite(IO_AIR_2, HIGH);
  digitalWrite(IO_LIGHT, HIGH);

  Wire.begin(42, 2, 100000);
  ADS.begin();
  ADS.setGain(0);
  f = ADS.toVoltage(1);

  esp_task_wdt_init(10, true);
  esp_task_wdt_add(NULL);

  xTaskCreatePinnedToCore(
     sensor_task,
     "SensorTask",
     4096,
     NULL,
     1,
     &sensorTaskHandle,
     1);
}

void loop()
{
  esp_task_wdt_reset();

  // Đọc chuỗi JSON từ Serial
```

```
  while (Serial.available())
  {
   char c = Serial.read();
   if (c == '\n')
   {
    inputString.trim();
    if (!inputString.isEmpty())
     handleJsonCommand(inputString);
    inputString = "";
   }
   else
   {
    inputString += c;
   }
  }

  delay(100);
}
```

Phụ lục 2 Chương trình xử lý ảnh và đọc dữ liệu Raspberry Pi

```
import os
import json
import time
import cv2
import torch
import mimetypes
import smtplib
from email.message import EmailMessage
```

```
from datetime import datetime
import serial
import paho.mqtt.client as mqtt
from influxdb_client import InfluxDBClient, Point
from influxdb_client.client.write_api import SYNCHRONOUS

# ========== CẤU HÌNH ==========
INFLUX_URL = "http://localhost:8086"
INFLUX_TOKEN =
"lstiyaIlNxYkz4Kqb4J12xKiKEgZ62EZqyX6BL_cFfWdp3FjYm9U-
Fy57bZIZT1TRTaF2CqY3gaf7zbiBlChqw=="
INFLUX_ORG = "my-org"
INFLUX_BUCKET = "env_spirulina_data"

MQTT_BROKER = "localhost"
MQTT_PORT = 1883
MQTT_TOPIC_PUB = "env_spirulina/sensor"
MQTT_TOPIC_CHECK = "CHECK_BB"

IMAGE_DIR = "/home/huy/images"
YOLO_MODEL = "/home/huy/Desktop/python_code/best.pt"
YOLO_OUTPUT_DIR = "/home/huy/Desktop/python_code/image_dt_save"

EMAIL_SENDER = "huybui211339@gmail.com"
EMAIL_PASSWORD = "pvbh kcfj muyl kwyo"
EMAIL_RECEIVER = "huybui211339@gmail.com"

LOG_FILE = "/home/huy/startup.log"
```

```
SENSOR_TIMEOUT = 120  # 2 phút
DETECT_INTERVAL = 120  # 2 phút

os.makedirs(IMAGE_DIR, exist_ok=True)
os.makedirs(YOLO_OUTPUT_DIR, exist_ok=True)

# ========== HÀM LOG ==========
def log(msg):
    now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    message = f"[{now}] {msg}"
    with open(LOG_FILE, "a", encoding="utf-8") as f:
        f.write(message + "\n")
    print(message)

# ========== HÀM GỬI EMAIL ==========
def send_email(subject, content, image_path=None):
    try:
        msg = EmailMessage()
        msg["Subject"] = subject
        msg["From"] = EMAIL_SENDER
        msg["To"] = EMAIL_RECEIVER
        msg.set_content(content)
        if image_path and os.path.exists(image_path):
            with open(image_path, "rb") as img_file:
                img_data = img_file.read()
                mime_type, _ = mimetypes.guess_type(image_path)
                maintype, subtype = mime_type.split("/") if mime_type else
("application", "octet-stream")
```

```
            msg.add_attachment(img_data, maintype=maintype, subtype=subtype,
filename=os.path.basename(image_path))
        with smtplib.SMTP_SSL("smtp.gmail.com", 465) as smtp:
            smtp.login(EMAIL_SENDER, EMAIL_PASSWORD)
            smtp.send_message(msg)
        log(f"[EMAIL] Đã gửi cảnh báo: {subject}")
    except Exception as e:
        log(f"[ERROR] Gửi email thất bại: {e}")

# ========== CAMERA ==========
def init_camera(max_index=5):
    for i in range(max_index):
        cam = cv2.VideoCapture(i)
        if cam.isOpened():
            log(f"Đã mở camera tại index {i}")
            return cam
        cam.release()
    log("Không tìm thấy camera khả dụng.")
    send_email("Cảnh báo: Lỗi Camera", "Không tìm thấy camera khả dụng.")
    return None

def capture_image():
    cam = init_camera()
    if not cam:
        return None
    ret, frame = cam.read()
    cam.release()
    if not ret:
```

```
        send_email("Cảnh báo: Lỗi Camera", "Không thể chụp ảnh từ camera.")
    return frame if ret else None

# ========== YOLO ==========
log("[YOLO] Đang tải mô hình...")
model = torch.hub.load('ultralytics/yolov5', 'custom', path=YOLO_MODEL,
force_reload=False)
model.conf = 0.6
log("[YOLO] Mô hình YOLO đã sẵn sàng!")

def detect_latest_image():
    images = [f for f in os.listdir(IMAGE_DIR) if f.endswith(".jpg")]
    if not images:
        log("[YOLO] Không có ảnh nào trong thư mục.")
        return

    latest_image = max(images, key=lambda x:
os.path.getmtime(os.path.join(IMAGE_DIR, x)))
    latest_path = os.path.join(IMAGE_DIR, latest_image)

    log(f"[YOLO] Xử lý ảnh mới nhất: {latest_path}")
    results = model(latest_path)
    df = results.pandas().xyxy[0]
    bubbles = df[df["name"] == "bubble"]

    results.render()
    result_frame = cv2.cvtColor(results.ims[0], cv2.COLOR_RGB2BGR)
```

```
    processed_path = os.path.join(YOLO_OUTPUT_DIR,
f"processed_{latest_image}")
    cv2.imwrite(processed_path, result_frame)

    if len(bubbles) > 0:
        log("[PHÁT HIỆN] Có bọt khí trong ảnh.")
        if mqtt_client:
            mqtt_client.publish(MQTT_TOPIC_CHECK, "1")
    else:
        log("[CẢNH BÁO] Không phát hiện bọt khí trong ảnh.")
        send_email("Cảnh báo: Không phát hiện bọt khí", f"Ảnh {latest_image} không
có bọt khí.", processed_path)
        if mqtt_client:
            mqtt_client.publish(MQTT_TOPIC_CHECK, "0")

# ========== MQTT CALLBACK ==========
def on_message(client, userdata, msg):
    try:
        if ser:
            topic = msg.topic.strip()
            value = msg.payload.decode().strip()
            print(f"[MQTT] Nhận: {topic} = {value}")
            valid_keys = ["STATUS_MOTOR_PH", "STATUS_MOTOR_AIR_1",
"STATUS_AIR_2", "STATUS_LIGH"]
            if topic in valid_keys and value in ["0", "1"]:
                json_cmd = json.dumps({topic: value}) + "\n"
                ser.write(json_cmd.encode())
                print(f"[SERIAL] Gửi: {json_cmd.strip()}")
```

```
    except Exception as e:
        log(f"[MQTT ERROR] {e}")
        send_email("Cảnh báo: Lỗi MQTT", f"MQTT gặp lỗi: {e}")

# ========== KẾT NỐI SERIAL ==========
try:
    ser = serial.Serial("/dev/ttyUSB0", 9600, timeout=1)
    log("Serial đã mở.")
except Exception as e:
    log(f"Lỗi mở serial: {e}")
    send_email("Cảnh báo: Lỗi Serial", f"Lỗi mở serial: {e}")
    ser = None

# ========== KẾT NỐI INFLUXDB ==========
try:
    influx = InfluxDBClient(url=INFLUX_URL, token=INFLUX_TOKEN,
org=INFLUX_ORG)
    write_api = influx.write_api(write_options=SYNCHRONOUS)
    log("Kết nối InfluxDB thành công.")
except Exception as e:
    log(f"Lỗi InfluxDB: {e}")
    send_email("Cảnh báo: Lỗi InfluxDB", f"Lỗi InfluxDB: {e}")
    influx = None
    write_api = None

# ========== KẾT NỐI MQTT ==========
try:
    mqtt_client = mqtt.Client()
```

```
    mqtt_client.on_message = on_message
    mqtt_client.connect(MQTT_BROKER, MQTT_PORT)
    for t in ["STATUS_MOTOR_PH", "STATUS_MOTOR_AIR_1",
"STATUS_AIR_2", "STATUS_LIGH"]:
        mqtt_client.subscribe(t)
    mqtt_client.loop_start()
    log("MQTT đã kết nối và subscribe các topic.")
except Exception as e:
    log(f"Lỗi MQTT: {e}")
    send_email("Cảnh báo: Lỗi MQTT", f"Lỗi MQTT: {e}")
    mqtt_client = None

last_sensor_time = time.time()
last_detect_time = time.time()
log("Hệ thống đã khởi động.")

# ========== VÒNG LẶP CHÍNH ==========
while True:
    try:
        if ser:
            line = ser.readline().decode('utf-8').strip()
            if not line:
                if time.time() - last_sensor_time > SENSOR_TIMEOUT:
                    log("[CẢNH BÁO] Không nhận được dữ liệu cảm biến trong 10
phút.")
                    send_email("Cảnh báo: Lỗi cảm biến", "Không nhận được dữ liệu cảm
biến trong 10 phút.")
                    last_sensor_time = time.time()
```

```
            time.sleep(0.1)
        else:
            try:
                if line.startswith("{'") or line.startswith("[{"):
                    line = line.replace("'", '"')
                data = json.loads(line)
                print(f"[SENSOR] {data}")
                last_sensor_time = time.time()
            except json.JSONDecodeError:
                continue

            if write_api:
                point = Point("sensor_data")
                for k, v in data.items():
                    point = point.field(k, v)
                point = point.time(time.time_ns())
                write_api.write(bucket=INFLUX_BUCKET, org=INFLUX_ORG,
record=point)

            if mqtt_client:
                mqtt_client.publish(MQTT_TOPIC_PUB, json.dumps(data))

            frame = capture_image()
            if frame is not None:
                timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
                image_name = f"photo_{timestamp}.jpg"
                image_path = os.path.join(IMAGE_DIR, image_name)
                cv2.imwrite(image_path, frame)
```

```
            log(f"[CAMERA] Đã lưu ảnh: {image_path}")
        else:
            log("[CAMERA ERROR] Không thể chụp ảnh.")
            send_email("Cảnh báo: Lỗi Camera", "Không thể chụp ảnh từ
camera.")

    if time.time() - last_detect_time >= DETECT_INTERVAL:
        detect_latest_image()
        last_detect_time = time.time()

  except Exception as e:
    log(f"[LOOP ERROR] {e}")
    send_email("Cảnh báo: Lỗi hệ thống", f"Lỗi trong vòng lặp chính: {e}")
    time.sleep(1)
```