ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

ĐỒ ÁN TỐT NGHIỆP

NGÀNH: KỸ THUẬT ĐIỀU KHIỂN VÀ TỰ ĐỘNG HÓA

ĐỀ TÀI:

NGHIÊN CỨU VÀ THIẾT KẾ HỆ THỐNG TỰ ĐỘNG GIÁM SÁT CHẤT LƯỢNG NƯỚC , CHIẾU SÁNG VÀ AN TOÀN THÔNG MINH TẠI HỒ BƠI

Người hướng dẫn: **TS. NGUYỄN THỊ KIM TRÚC**
Sinh viên thực hiện:
1. HỒ TUẤN PHONG – MSSV: 105200375 – LỚP: 20TDHCLC1
2. TRẦN GIA LINH – MSSV: 105200370 – LỚP: 20TDHCLC1

Đà Nẵng, 6/2025

TÓM TẮT

Tên đề tài: *Nghiên cứu và thiết kế hệ thống tự động hóa hồ bơi với giám sát chất lượng nước, chiếu sáng và an toàn thông minh*

Sinh viên thực hiện: Hồ Tuấn Phong Số thẻ SV: 105200375

Trần Gia Linh Số thẻ SV: 105200370

Lớp: 20TDHCLC1

Đồ án "Nghiên cứu và thiết kế hệ thống tự động hóa hồ bơi với giám sát chất lượng nước, chiếu sáng và an toàn thông minh" nhằm xây dựng một hệ thống hồ bơi thông minh có khả năng vận hành tự động và giám sát từ xa. Qua khảo sát thực tế tại khu resort Marriott, nhóm nhận thấy hệ thống vận hành hồ bơi hiện nay vẫn chủ yếu sử dụng phương pháp thủ công hoặc bán tự động, dẫn đến nhiều bất cập như khó kiểm soát chất lượng nước theo thời gian thực, không phát hiện kịp thời các tình huống đuối nước và thiếu khả năng tích hợp giám sát tập trung.

Trên cơ sở đó, nhóm đề xuất giải pháp tích hợp giữa bộ điều khiển lập trình PLC Siemens S7-1200, máy tính nhúng Raspberry Pi 5, hệ thống cảm biến đo pH, Clo dư và mực nước, cùng với mô hình trí tuệ nhân tạo (YOLOv11) để phát hiện đuối nước qua camera giám sát. Giao diện Web giám sát được thiết kế bằng Flask Webserver, cho phép người dùng theo dõi trạng thái thiết bị, điều khiển từ xa và nhận cảnh báo thời gian thực.

Hệ thống sau khi mô phỏng và kiểm thử cho thấy tính hiệu quả cao trong vận hành, khả năng phản hồi nhanh, nâng cao độ an toàn và thuận tiện trong quản lý. Đồng thời, giải pháp có tính mở rộng linh hoạt và tiềm năng ứng dụng vào các khu vực công cộng hoặc hồ bơi thương mại trong tương lai.

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

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

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

TT	Họ tên sinh viên	Số thẻ SV	Lớp	Ngành
1	Hồ Tuấn Phong	105200375	20TDHCLC1	Kỹ thuật Điều khiển và Tự động hóa
2	Trần Gia Linh	105200370	20TDHCLC1	Kỹ thuật Điều khiển và Tự động hóa

1. *Tên đề tài đồ án:*

   Nghiên cứu và thiết kế hệ thống tự động hóa hồ bơi với giám sát chất lượng nước, chiếu sáng và an toàn thông minh

2. *Đề tài thuộc diện:* ☐*Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện*
3. *Các số liệu và dữ liệu ban đầu:*
   - *Sơ đồ nguyên lý*
   - *Sơ đồ mặt bằng, bản vẽ thi công công trình*
4. *Nội dung các phần thuyết minh và tính toán:*

a. Phần chung:

TT	Họ tên sinh viên	Nội dung
1	Hồ Tuấn Phong	- Tìm hiểu tổng quan hệ thống - Tìm hiểu các thiết bị và quy trình vận hành của hệ thống xử lý nước, chiếu sáng - Thiết lập lưu đồ thuật toán, xây dựng sơ đồ phân cấp - Thiết lập các chức năng của hệ thống - Đánh giá kết quả thực hiện và rút ra kết luận
2	Trần Gia Linh	

b. Phần riêng:

TT	Họ tên sinh viên	Nội dung
1	Hồ Tuấn Phong	- Tìm hiểu và thiết lập kết nối Raspberry Pi với PLC, Camera IP và Raspberry Pi - Thiết kế giao diện giám sát, điều khiển Webserver - Xây dựng mô hình học sâu và triển khai trên Raspberry Pi
2	Trần Gia Linh	- Tìm hiểu và thiết lập kết nối các cảm biến với PLC

		- Lập trình PLC - Thiết kế giao diện HMI

5. Các bản vẽ, đồ thị (ghi rõ các loại và kích thước bản vẽ) :

a. Phần chung:

TT	Họ tên sinh viên	Nội dung

b. Phần riêng:

TT	Họ tên sinh viên	Nội dung
1	Trần Gia Linh	- Sơ đồ đấu nối - Bản vẽ tủ điện

6. Họ tên ngườihướng dẫn:	*Phần/ Nội dung:*
TS. Nguyễn Thị Kim Trúc	- Hướng đi của đề tài. - Giải pháp điều khiển. - Hướng dẫn thuyết minh báo cáo đề tài

7. Ngày giao nhiệm vụ đồ án: 17/2/2025

8. Ngày hoàn thành đồ án: 17/6/2025

Đà Nẵng, ngày 17 tháng 6 năm 2025

Trưởng Bộ môn Tự động hóa **Người hướng dẫn**

TS. Giáp Quang Huy

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

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

PHIẾU KIỂM SOÁT TIẾN ĐỘ LÀM ĐỒ ÁN TỐT NGHIỆP

(Phiếu dành cho người hướng dẫn/sinh viên)

Họ tên sinh viên: Hồ Tuấn Phong Số thẻ SV: 105200375

Trần Gia Linh Số thẻ SV: 105200370

Tên đề tài ĐATN: Nghiên cứu và thiết kế hệ thống tự động hóa hồ bơi với giám sát chất lượng nước, chiếu sáng và an toàn thông minh

Họ tên người HD: TS. Nguyễn Thị Kim Trúc Đơn vị: Khoa Điện

Tuần	Ngày	Khối lượng		GVHD ký tên
		đã thực hiện (%)	tiếp tục thực hiện (%)	
1	17/02/2025	Nhận đề tài	Tìm hiểu tổng quan đề tài	
2	24/02/2025	Xác định mục tiêu, phạm vi, đối tượng và phương pháp nghiên cứu	Tìm hiểu các yêu cầu của đề tài, nghiên cứu sơ đồ nguyên lý	
3	03/03/2025	Xây dựng quy trình công nghệ cho hệ thống	Viết báo cáo thuyết minh chương 1	
4	10/03/2025	Duyệt lần 1: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
5	17/03/2025	Hoàn thành báo cáo thuyết minh chương một	Tìm hiểu các thiết bị có trong hệ thống	
6	24/03/2025	Hoàn thiện sơ đồ phân cấp cho hệ thống	Xây dựng lưu đồ thuật toán	
7	31/03/2025	Hoàn thiện lưu đồ thuật toán	Viết báo cáo thuyết minh chương hai	
8	07/04/2025	Duyệt lần 2: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
9	14/04/2025	Hoàn thành báo cáo thuyết minh chương hai	Tính chọn các thiết bị có trong hệ thống	
10	21/04/2025	Hoàn thành tính chọn các thiết bị trong hệ thống	Tính chọn phần chiếu sáng	
11	28/04/2025	Hoàn thành tính chọn phần chiếu sáng	Viết báo cáo thuyết minh chương ba	
12	05/05/2025	Duyệt lần 3: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
13	12/05/2025	Hoàn thành báo cáo thuyết minh chương ba	Lập trình thiết lập kết nối các thiết bị	

14	19/05/2025	Hoàn thành lập trình kết nối các thiết bị	Xây dựng chương trình điều khiển các chế độ Auto, Manual và mô hình AI	
15	26/05/2025	Mô phỏng và kiểm tra lỗi chương trình	Viết báo cáo thuyết minh chương bốn	
16	02/06/2025	Duyệt lần 4: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
17	09/06/2025	Hoàn thành viết báo cáo thuyết minh chương bốn	Kiểm nghiệm lỗi và hoàn thiện giao diện điều khiển	
18	16/06/2025	Đánh giá và kiểm nghiệm lại các chức năng của hệ thống	Viết báo cáo thuyết minh chương 5 và nộp thuyết minh	

LỜI NÓI ĐẦU VÀ CẢM ƠN

Để hoàn thành đồ án tốt nghiệp này, bên cạnh sự nỗ lực của bản thân, chúng em đã nhận được rất nhiều sự quan tâm, chỉ dẫn và hỗ trợ quý báu. Với lòng biết ơn sâu sắc, chúng em xin trân trọng gửi lời cảm ơn đến:

Trước hết, chúng em xin bày tỏ lòng tri ân chân thành tới TS. Nguyễn Thị Kim Trúc. Cô không chỉ là người định hướng khoa học, truyền đạt những kiến thức chuyên môn sâu sắc mà còn là người tận tình chỉ bảo, động viên chúng em trong suốt quá trình thực hiện đề tài. Sự hướng dẫn của cô là nền tảng quan trọng nhất giúp chúng em hoàn thành tốt đồ án này.

Chúng em xin gửi lời cảm ơn đến tập thể Công ty CP Tổ Ong đã tạo môi trường học tập, nghiên cứu năng động tại công trình thi công dự án và hỗ trợ chúng em về cơ sở vật chất, được cung cấp các tài liệu liên quan đến đề tài cũng như hỗ trợ tư vấn làm dự án được phát triển hoàn thiện. Đặc biệt, chúng em xin cảm ơn anh Duy và anh Giang đã dành thời gian đưa ra những góp ý chuyên môn xác đáng và kịp thời, giúp chúng em giải quyết những khó khăn gặp phải.

Cuối cùng, chúng em xin trân trọng cảm ơn Ban Giám hiệu Trường Đại học Bách Khoa - ĐH Đà Nẵng cùng toàn thể Quý thầy cô Khoa Điện. Thầy cô đã trang bị cho chúng em nền tảng tri thức vững chắc và tạo mọi điều kiện thuận lợi trong suốt quá trình học tập và thực hiện đồ án tốt nghiệp.

Trân trọng cảm ơn!

LỜI CAM ĐOAN LIÊM CHÍNH HỌC THUẬT

"Chúng em xin cam đoan đề tài đồ án tốt nghiệp: "*Nghiên cứu và thiết kế hệ thống tự động hóa hồ bơi với giám sát chất lượng nước, chiếu sáng và an toàn thông minh*" là một dự án nghiên cứu độc lập của nhóm gồm hai thành viên dưới sự hướng dẫn của của giáo viên hướng dẫn TS. Nguyễn Thị Kim Trúc. Ngoài ra không có bất kì sự sao chép nào trong việc thực hiện. Sản phẩm, nội dung báo cáo là quá trình mà chúng em nỗ lực nghiên cứu và thực hiện dưới sự hỗ trợ từ TS. Nguyễn Thị Kim Trúc. Các số liệu, kết quả trình bày trong báo cáo là hoàntoàn trung thực, chúng em xin chịu toàn bộ trách nhiệm, kỷ luật của bộ môn và nhà trường nếu như có vấn đề về liêm chính học thuật xảy ra".

Sinh viên thực hiện 1 Sinh viên thực hiện 2

Hồ Tuấn Phong Trần Gia Linh

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

DANH MỤC TỪ VIẾT TẮT

VIẾT TẮT	NỘI DUNG
PLC	Programmable Logic Controller: Bộ điều khiển lập trình
HMI	Human Machine Interface: Thiết bị giao tiếp giữa con người với hệ điều hành
AI	Artificial Intelligence – Trí tuệ nhân tạo
RTU	Remote Terminal Unit – Thiết bị đầu cuối từ xa
TCP/IP	Transmission Control Protocol/Internet Protocol – Giao thức truyền dữ liệu
YOLO	You Only Look Once – Mô hình học sâu nhận diện vật thể thời gian thực
SCADA	Supervisory Control And Data Acquisition_ Hệ thống giám sát và điều khiển tập trung, cho phép con người quản lý và vận hành các hệ thống kỹ thuật từ xa

MỞ ĐẦU

1. Lý do chọn đề tài

Trong những năm gần đây, việc nâng cao chất lượng quản lý và vận hành các khu vực công cộng, đặc biệt là các khu vực như hồ bơi, đang trở thành một yêu cầu cấp thiết. Hồ bơi không chỉ đáp ứng nhu cầu thể thao và giải trí mà còn liên quan trực tiếp đến sức khỏe và sự an toàn của người sử dụng. Tuy nhiên, tại nhiều cơ sở, việc quản lý và giám sát các hoạt động trong hồ bơi chủ yếu vẫn dựa vào phương pháp thủ công hoặc bán tự động, dẫn đến nhiều hạn chế trong việc kiểm soát chất lượng nước, an toàn người bơi, và hiệu quả vận hành của hệ thống. Các vấn đề thường gặp là khó khăn trong việc giám sát chất lượng nước theo thời gian thực, không phát hiện kịp thời các tình huống nguy hiểm, đặc biệt là tình huống đuối nước, và thiếu khả năng điều khiển từ xa hoặc tích hợp giám sát tập trung.

Với sự phát triển mạnh mẽ của các công nghệ hiện đại như tự động hóa công nghiệp, Internet of Things (IoT) và trí tuệ nhân tạo (AI) , việc số hóa và thông minh hóa các hệ thống vận hành hồ bơi là một giải pháp khả thi và rất có tiềm năng. Các hệ thống tự động giúp giảm tải cho nhân viên, nâng cao hiệu quả công việc, tăng tính an toàn cho người sử dụng, và dễ dàng tích hợp vào các hệ thống giám sát tập trung như SCADA hoặc Cloud. Đặc biệt, việc ứng dụng AI vào giám sát hành vi người bơi giúp phát hiện các tình huống nguy hiểm như đuối nước một cách nhanh chóng và chính xác, nâng cao mức độ an toàn cho người bơi.

Do đó, đồ án này nghiên cứu và thiết kế hệ thống tự động hóa hồ bơi, với mục tiêu xây dựng một hệ thống giám sát và điều khiển thông minh, có thể vận hành tự động hoặc từ xa, và sử dụng AI để phát hiện các tình huống nguy hiểm. Hệ thống này không chỉ giúp cải thiện chất lượng dịch vụ mà còn đảm bảo an toàn cho người sử dụng, đáp ứng nhu cầu ngày càng cao về sự an toàn và tiện lợi trong các hoạt động công cộng.

2. Mục tiêu nghiên cứu

Đề tài hướng đến việc thiết kế và mô phỏng một hệ thống hồ bơi thông minh tích hợp các công nghệ điều khiển tự động, IoT và trí tuệ nhân tạo, nhằm nâng cao hiệu quả vận hành, đảm bảo chất lượng nước và an toàn cho người sử dụng. Cụ thể, mục tiêu nghiên cứu bao gồm:

- **Tự động hóa quá trình xử lý nước hồ bơi**:

+ Đo và giám sát liên tục các thông số pH, clo dư, mực nước bằng cảm biến.
+ Điều khiển tự động các bơm lọc, van điện từ, bơm hóa chất nhằm đảm bảo chất lượng nước đạt tiêu chuẩn.

- **Thiết kế và điều khiển hệ thống chiếu sáng hồ bơi**:

+ Cho phép vận hành theo các chế độ: thủ công, tự động theo thời gian lập trình sẵn, và điều khiển từ xa qua Web.
+ Tối ưu số lượng và vị trí đèn nhằm đảm bảo độ rọi tiêu chuẩn cho từng khu vực.

- **Giám sát an toàn người bơi bằng trí tuệ nhân tạo**:

+ Ứng dụng mô hình học sâu (YOLO11) để phát hiện các tình huống nguy hiểm như đuối nước hoặc mất ý thức không được phát hiện do các lý do khách quan.
+ Triển khai mô hình AI trên Raspberry Pi để xử lý thời gian thực từ camera giám sát.

- **Xây dựng hệ thống giám sát và điều khiển từ xa**:

+ Thiết kế giao diện Webserver hiển thị trạng thái thiết bị, cho phép điều khiển, cấu hình hệ thống và nhận cảnh báo từ xa.
+ Tích hợp truyền thông Modbus RTU và TCP/IP để kết nối các thiết bị điều khiển, cảm biến và hệ thống Web.

- **Mô phỏng, kiểm thử và đánh giá hiệu quả hệ thống**:

+ Triển khai mô hình điều khiển và giám sát trên phần mềm TIA Portal và môi trường Python.
+ Đánh giá các tiêu chí: độ chính xác mô hình AI, thời gian phản hồi, tính ổn định và khả năng vận hành đồng thời các chức năng.

3. Phạm vi và đối tượng nghiên cứu

Phạm vi nghiên cứu của đề tài bao gồm thiết kế, mô phỏng và kiểm thử một hệ thống hồ bơi thông minh trong quy mô mô hình. Hệ thống chưa triển khai trên hồ bơi thật nhưng có đầy đủ tính năng của một hệ thống thực tế, bao gồm các khâu xử lý nước, chiếu sáng, cảnh báo AI, và giám sát Web.

Cụ thể, phạm vi đề tài giới hạn ở:

- Phần cứng: Cảm biến đo pH, clo dư, mực nước; van điện từ; bơm lọc, bơm chìm; PLC Siemens S7-1200; Raspberry Pi 5; camera IP.
- Phần mềm: Lập trình điều khiển PLC bằng TIA Portal V17; huấn luyện mô hình AI YOLO11 nhận diện đuối nước; thiết kế HMI và giao diện Web giám sát.
- Truyền thông: Tích hợp kết nối Modbus RTU và TCP/IP giữa thiết bị điều khiển, cảm biến và hệ thống giám sát.

Đối tượng nghiên cứu chính: Hệ thống tự động hóa xử lý nước và chiếu sáng hồ bơi, ứng dụng mô hình học sâu để giám sát hành vi người bơi thông qua camera, giao tiếp dữ liệu giữa thiết bị điều khiển, cảm biến và nền tảng Webserver.

4. Phương pháp nghiên cứu

Trong quá trình thực hiện đề tài, nhóm đã kết hợp nhiều phương pháp nghiên cứu để từng bước xây dựng và hoàn thiện hệ thống. Cụ thể như sau:

Nhóm đã tiến hành khảo sát tại Khu nghỉ dưỡng Marriot để tìm hiểu quy trình vận hành hiện tại, đặc biệt là quá trình điều khiển và sử dụng các thiết bị điện. Qua quá trình quan sát và thu thập thông tin, nhóm đã xác định rõ nhu cầu tự động hóa cũng như những bất cập còn tồn tại trong khâu vận hành.

Phương pháp thiết kế và mô phỏng: Nhóm sử dụng phần mềm AutoCAD để thiết kế các bản vẽ kỹ thuật, bao gồm sơ đồ mạch động lực, mạch điều khiển và tủ điện. Tiếp theo, phần mềm TIA Portal được sử dụng để mô phỏng hoạt động của hệ thống.

Phương pháp huấn luyện mô hình nhận diện: Nhóm tiến hành thu thập hình ảnh thực tế, gán nhãn dữ liệu bằng phần mềm Roboflow, và huấn luyện mô hình nhận phát hiện đuối nước bằng thuật toán YOLOv11 thực thi trên nền tảng trên Google Colab với sự hỗ trợ của GPU để tăng tốc độ xử lý.

Phương pháp tích hợp hệ thống: Các thành phần như camera, mô hình AI, máy tính nhúng Raspberry Pi, bộ điều khiển PLC được kết nối và lập trình hoạt động đồng bộ. Việc tích hợp đảm bảo toàn hệ thống hoạt động đồng bộ: đo lường – xử lý – cảnh báo – điều khiển – giám sát từ xa.

Phương pháp kiểm tra và đánh giá: Nhóm kiểm tra hệ thống qua nhiều lần chạy thử. Kiểm tra tính ổn định hệ thống, khả năng điều khiển từ xa và độ thân thiện với người dùng. Ngoài ra, mô hình AI được đánh giá qua các chỉ số như Precision, Recall, F1-score, mAP.

5. Cấu trúc đồ án

Cấu trúc đồ án bao gồm

- *Chương 1: Tổng quan đề tài*

Mục tiêu của đồ án là thiết kế và xây dựng hệ thống tự động hóa hồ bơi thông minh, bao gồm giám sát chất lượng nước, chiếu sáng, và an toàn bằng AI. Hệ thống sử dụng các cảm biến để đo pH, clo và mực nước, cùng với PLC Siemens S7-1200 và Raspberry Pi cho giám sát và điều khiển từ xa. Công nghệ AI được áp dụng để phát hiện tình huống đuối nước.

- *Chương 2: Thiết kế hệ thống hồ bơi thông minh*

Chương này sẽ trình bày hệ thống được chia thành các cấp: cảm biến và thiết bị chấp hành (bơm, van, cảm biến) ; điều khiển trung tâm (PLC, HMI) ; và giám sát từ xa qua Raspberry Pi và Webserver. Các thiết bị và công nghệ như PLC Siemens, camera IP và Raspberry Pi được lựa chọn để đảm bảo hệ thống hoạt động hiệu quả và dễ dàng giám sát.

- *Chương 3: Chương trình điều khiển*

Chương này mô tả các thuật toán điều khiển cho hệ thống xử lý nước, chiếu sáng và phát hiện tình huống đuối nước qua camera AI. Các lưu đồ thuật toán chi tiết cho từng hệ thống được đưa ra, từ xử lý nước cho đến điều khiển bơm và camera giám sát.

- *Chương 4: Triển khai và mô phỏng hệ thống*

Trong chương này, các mô phỏng hệ thống được triển khai trên phần mềm TIA Portal và Raspberry Pi. Các bước thiết lập kết nối và mô phỏng các hệ thống xử lý nước, chiếu sáng, và camera giám sát được trình bày rõ ràng.

- *Chương 5: Kiểm nghiệm và đánh giá*

Chương cuối cùng sẽ tổng kết các kết quả đạt được từ hệ thống hồ bơi thông minh. Các mô phỏng và kết quả kiểm thử thực tế sẽ được trình bày để đánh giá hiệu quả của hệ thống từ đó nhìn ra những điểm còn hạn chế và đề xuất hướng phát triển trong tương lai.

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

1.1 Đặt vấn đề

Trong bối cảnh hiện nay, việc nâng cao chất lượng quản lý và vận hành các khu vực công cộng, đặc biệt là hồ bơi, trở thành một yêu cầu cấp thiết. Hồ bơi không chỉ phục vụ nhu cầu thể thao và giải trí mà còn liên quan trực tiếp đến sức khỏe và an toàn của người sử dụng. Qua quá trình khảo sát tại khu resort Marriot, hệ thống vận hành hồ bơi vẫn chủ yếu dựa vào phương pháp thủ công hoặc bán tự động, dẫn đến các hạn chế như:

- Khó khăn trong việc kiểm soát chất lượng nước theo thời gian thực.
- Không phát hiện kịp thời các tình huống nguy hiểm, đặc biệt là đuối nước.
- Thiếu khả năng vận hành từ xa và tích hợp giám sát tập trung.

Trong khi đó, sự phát triển của các công nghệ hiện đại như tự động hóa công nghiệp, truyền thông vạn vật kết nối (IoT) và trí tuệ nhân tạo (AI) mở ra cơ hội lớn để số hóa và thông minh hóa các hệ thống vận hành hiện tại [1]. Việc tích hợp thiết bị điều khiển khả trình (PLC) để điều khiển tự động và ứng dụng AI thị giác máy tính để giám sát an toàn có thể mang lại những lợi ích sau:

- Giảm tải cho nhân sự vận hành.
- Tăng tính an toàn và độ tin cậy cho người sử dụng.
- Dễ dàng tích hợp với hệ thống giám sát tập trung của khu vực (SCADA/Cloud)

Đáng chú ý, các giải pháp giám sát sự cố đuối nước bằng AI như AngelEye và các mô hình thị giác máy tính sâu (Deep learning) đã được chứng minh có hiệu quả về thời gian phản hồi chỉ từ 3–5 giây, so với 15 giây của phương pháp truyền thống [2], [3]. Nghiên cứu thực tiễn tại Thái Lan cũng chỉ ra rằng hệ thống computer vision có thể giúp giảm đáng kể tỷ lệ tai nạn dưới nước [4].

Từ thực tế đó, việc nghiên cứu và xây dựng một hệ thống hồ bơi thông minh, có khả năng giám sát từ xa và cảnh báo tình huống nguy hiểm bằng AI là cần thiết và mang tính ứng dụng cao.

1.2 Quy trình công nghệ

1.2.1 Quy trình xử lý nước hồ bơi

a. Nguyên nhân gây ô nhiễm và yêu cầu về chất lượng nước sau xử lý

Sau khi qua sử dụng nước bể bơi sẽ bị nhiễm bẩn. Những nguyên nhân gây ô nhiễm chính là sự bài tiết từ cơ thể người bơi, bụi bậm và sự phát triển các loại vi sinh vật, thực vật sống trong nước.

Các chất gây ô nhiễm cho nước bể bơi phân loại thành ba nhóm chính sau:

- Nhóm 1: Các váng dầu, váng mỡ nổi trên bề mặt .
- Nhóm 2: Các chất rắn lơ lửng .

- Nhóm 3: Các chất cặn lớn chìm dưới đáy bể .

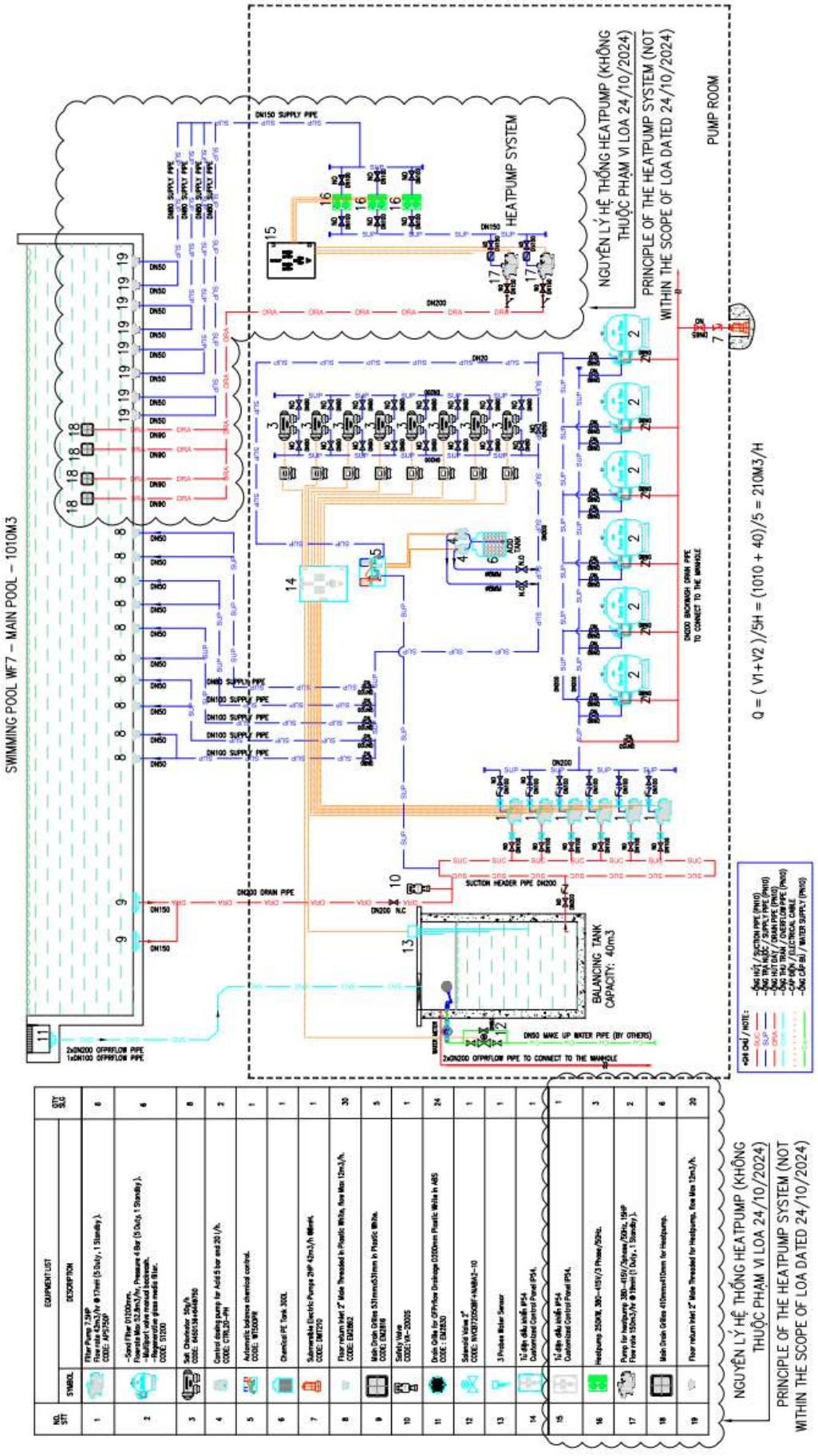
Vì vậy việc cần thiết là phải trả lại cho nước bể bơi sự trong sạch để đảm bảo sức khoẻ cho người sử dụng và môi trường cảnh quan của bể bơi.

Theo tiêu chuẩn Việt Nam số TCVN 4260-2012 [5] và Thông tư 41/2018/TT-BYT [6] về các tiêu chuẩn vệ sinh và an toàn cho các Bể bơi, nước Bể bơi sau xử lý phải đạt được các tiêu chuẩn cần thiết như sau:

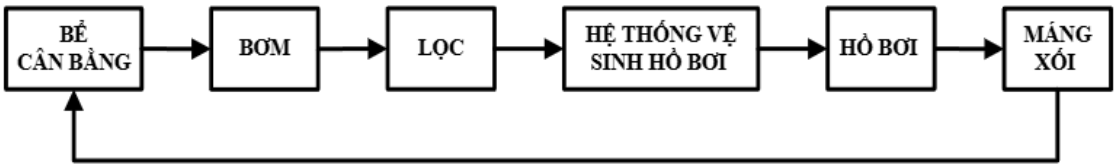
Bảng 1.1 Tiêu chuẩn nước hồ bơi sau xử lý

Tên chỉ tiêu	**Yêu cầu**
1. Nước Bể bơi không được kích thích da và các màng nhầy	
2. Độ cứng (tính theo CaCO3)	≤ 500mg/l
3. Độ pH	7.2 - 7.4
4. Clorua hợp chất	< 0.5 mg/l
5. Clo tự do	1.5 – 2 mg/l

b. Công nghệ lọc nước tuần hoàn



Hình 1.1 Sơ đồ nguyên lý



Hình 1.2 Sơ đồ khối quá trình lọc tuần hoàn

Nguyên lý của hệ thống lọc: Nguồn nước cấp được cấp vào hợp bể cân bằng thông qua van phao, bơm hút nước sẽ hút nước từ 2 đường ống hút đáy (30%) và hút bể cân bằng (70%) đi qua hệ thống lọc nước > xử lý hóa chất điện phân > bơm định lượng châm axit > cấp vào bể bơi.

Dòng chảy tuần hoàn qua rãnh thu nước xả tràn mang theo các váng dầu, váng mỡ (cặn nhóm 1) thu về bể cân bằng, bể này có tác dụng thu gom nước từ rãnh xả tràn quanh bể về rồi mới được hút vào bơm lọc. Ngoài ra, bể cân bằng còn có thể tiếp nhận lượng nước sạch bổ sung hàng ngày để điều chỉnh mực nước bể bơi.

Dòng chảy tuần hoàn thu qua đáy bể sẽ đẩy toàn bộ cặn lơ lửng (cặn nhóm 2) về hố thu đáy bể rồi hút bằng bơm qua hệ thống lọc. Các loại cặn lớn, nặng, hoặc được keo tụ bằng hóa chất keo tụ chìm xuống lắng đọng trên đáy bể (cặn nhóm 3) sẽ được hút đi hàng ngày bằng hệ thống hút vệ sinh.

Như vậy, các loại cặn đều được xử lý triệt để qua hệ thống lọc, nước trong bể bơi sẽ không bị thất thoát ra ngoài nên các hao hụt trong quá trình sử dụng sẽ chỉ là lượng nước bay hơi và rửa ngược cho bình lọc.

1.2.2 Nguyên lý hệ thống chiếu sáng

Hệ thống đèn hồ bơi WF7 được phân thành 3 loại chiếu sáng chính:

- Đèn sợi quang: Gồm 592 điểm sáng theo hiệu ứng twinkle, được bố trí ngẫu nhiên với kích thước khác nhau.
- Đèn vách hồ bơi gồm:
  - + 26 đèn PL4A (Main Pool)
  - + 8 đèn PL3 (Main Pool)
  - + 8 đèn PL2 (Lounge Pool)
  - + 10 đèn PL3 (Lounge Pool)
- Đèn bậc thang gồm 16 đèn PL3 (Main Pool) , 8 đèn PL3 (Lounge Pool)

Hệ thống đèn hồ bơi WF7 sẽ được phân chia và kết nối về 3 tủ điện đèn hồ bơi, đặt tại khu vực gần hồ bơi để đảm bảo khoảng cách giữa các đèn tới tủ điện được ngắn nhất, nhằm hạn chế sụt áp cho hệ thống đèn hồ bơi.

Nguyên lý hoạt động hệ thống đèn:

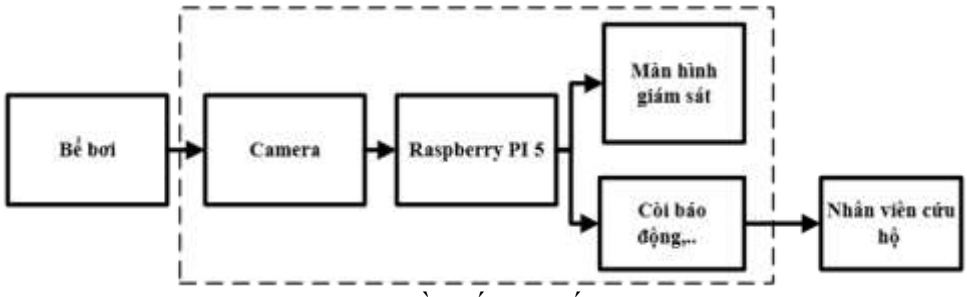
- Chế độ vận hành tự động: Hệ thống đèn hoạt động theo tín hiệu timer tại tủ điện hồ bơi WF7.

+ CCT25: Đèn bậc thang PL3 (Main Pool & Lounge Pool) hoạt động theo tín hiệu Timer 1 (từ 18 giờ đến 22 giờ)
+ CCT25A: Toàn bộ 592 điểm đèn sợi quang hoạt động theo tín hiệu Timer 2 (từ 18 giờ đến 4 giờ sáng hôm sau)
+ CCT25B: Đèn vách PL4A hoạt động theo tín hiệu Timer 3 (từ 18 giờ đến 20 giờ)
+ CCT25C: Đèn vách PL3 (Main Pool & Lounge Pool) , đèn vách PL2 (Lounge Pool) hoạt động theo tín hiệu Timer 4 (từ 18 giờ đến 20 giờ)

- Chế độ vận hành bằng tay: Được thực hiện bởi nhân viên vận hành thông qua công tắc chuyển mạch 3 vị trí tại tủ điện hồ bơi WF7.
- Chế độ vận hành từ xa: Hệ thống có thể được điều khiển từ xa thông qua giao diện điều khiển trung tâm, cho phép bật/tắt hoặc điều chỉnh các nhóm đèn bằng thiết bị di động hoặc máy tính kết nối mạng.

1.2.3 Camera và AI

Hệ thống camera AI phát hiện đuối nước hoạt động dựa trên nguyên lý tích hợp giữa công nghệ xử lý hình ảnh thời gian thực và các thuật toán học sâu nhằm giám sát liên tục các khu vực nước. Các camera IP được lắp đặt cố định xung quanh bể bơi hoặc vùng nước mở sẽ thu thập dữ liệu video và truyền về một đơn vị xử lý trung tâm, thường là thiết bị tính toán nhúng hoặc máy chủ chuyên dụng. Tại đây, các mô hình trí tuệ nhân tạo, đặc biệt là mạng nơ-ron tích chập (CNN) , được triển khai để phân tích từng khung hình, nhận diện người trong nước và đánh giá hành vi nguy cơ như mất ổn định, chuyển động yếu hoặc chìm kéo dài [7], [8]. Khi hệ thống phát hiện tín hiệu bất thường vượt ngưỡng xác định, một quy trình phân tích nâng cao sẽ đánh giá thêm để xác thực mức độ nguy hiểm [8]. Nếu tình huống được xác định là có khả năng đuối nước, hệ thống ngay lập tức phát ra tín hiệu cảnh báo đến bộ điều khiển trung tâm. Từ đó, hệ thống sẽ tự động kích hoạt các cơ chế phản hồi như chuông báo động, đèn cảnh báo và gửi thông báo đến đội ngũ cứu hộ. Đồng thời, toàn bộ dữ liệu video và nhật ký cảnh báo được lưu trữ để phục vụ mục đích phân tích sau sự cố, giúp cải tiến liên tục thuật toán và tối ưu hiệu suất hệ thống trong tương lai.



Hình 1.3 Sơ đồ khối hệ thống camera AI

Hệ thống được mô tả ở Hình 1.3 bao gồm các thành phần chính sau:

- Camera: Thu thập dữ liệu, có nhiệm vụ quay hoặc chụp hình ảnh từ bể bơi.
- Raspberry Pi 5: Được sử dụng để xử lý dữ liệu từ camera. Dựa trên bối cảnh, Raspberry Pi 5 chạy mô hình YOLO11, một mô hình học sâu tiên tiến cho phát hiện đối tượng theo thời gian thực. Mô hình này có thể được huấn luyện để nhận diện các đối tượng Raspberry Pi 5 có hai đầu ra: một kết nối với màn hình để hiển thị kết quả, và một kết nối với báo động để kích hoạt khi cần thiết.
- Màn hình: Hiển thị, cho phép theo dõi trực tiếp hình ảnh và kết quả dự đoán từ mô hình được chạy trong Raspberry Pi 5.
- Còi báo động: Cảnh báo đến nhân viên cứu hộ để kịp thời xử lý khi có tình huống nguy hiểm.

1.3 Giải pháp đề xuất

Để đạt được các mục tiêu nêu trên, đề tài đề xuất triển khai giải pháp theo mô hình kết hợp giữa **PLC Siemens S7-1200**, **Raspberry Pi**, **camera IP** và **mô hình AI**, cụ thể:

❖ ***Hệ thống điều khiển tự động hóa trung tâm***

- Sử dụng PLC Siemens S7-1200 (CPU) để điều khiển toàn bộ hệ thống lọc nước, bơm hóa chất và chiếu sáng.
- Thiết lập chương trình điều khiển trong phần mềm TIA Portal với các khối chức năng FB/FC rõ ràng.
- Kết nối cảm biến pH, Clo, mực nước bằng giao thức Modbus RTU hoặc analog 4-20mA.

❖ ***Hệ thống giám sát bằng AI***

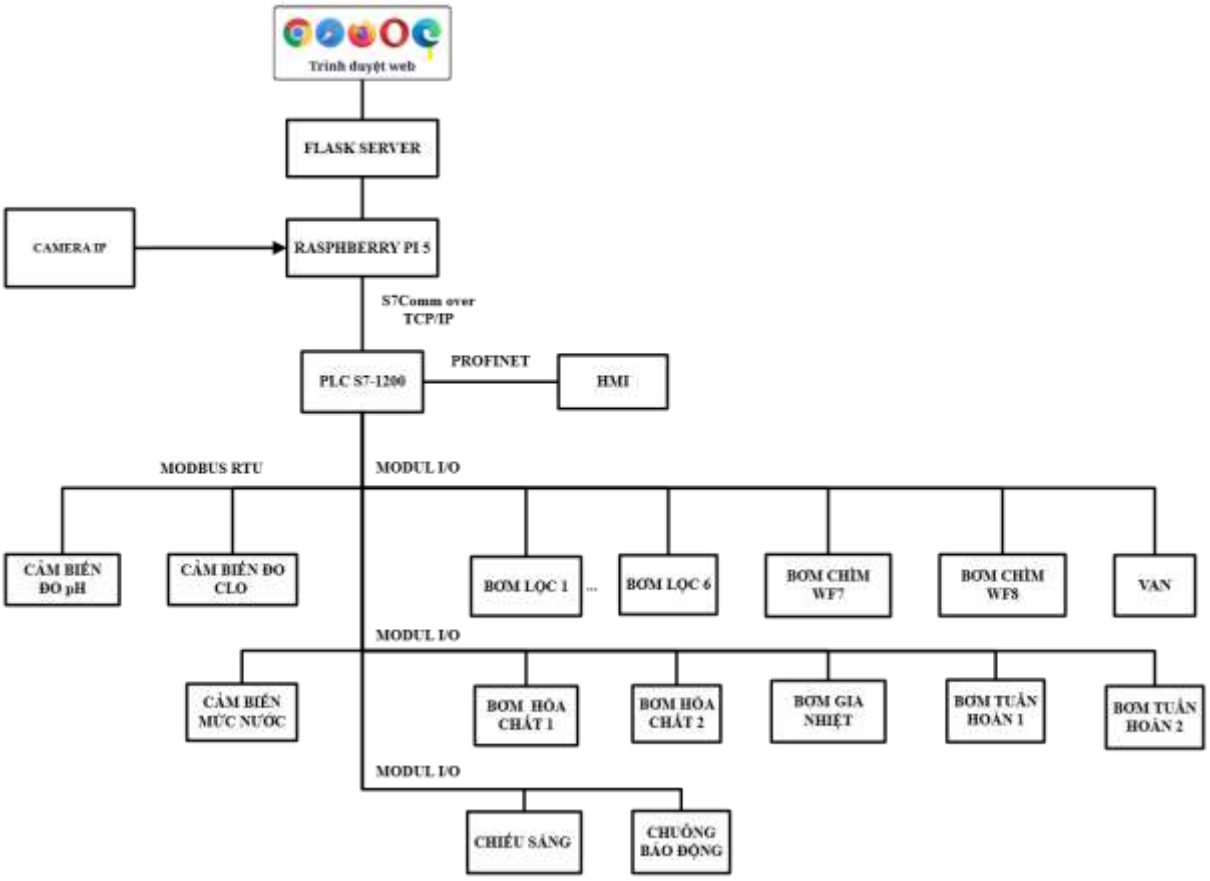
- Lắp đặt camera IP tại các vị trí chiến lược quanh hồ bơi.
- Sử dụng Raspberry Pi 5 để thu nhận luồng video và chạy mô hình AI dựa trên OpenCV + YOLO.
- AI phân tích hành vi người bơi, phát hiện các bất thường (đuối nước, bất động, bất tỉnh) .

❖ ***Hệ thống giao tiếp và giám sát từ xa***

- Tích hợp Web Server và sử dụng Raspberry Pi để làm Server sử dụng HMI của S7-1200 để hiển thị thông tin hệ thống tại chỗ.
- Giao tiếp PLC – AI – Web qua các giao thức TCP Socket.
- Giao diện cho phép:
  + Quan sát thông số thực tế (pH, Clo, mực nước) .
  + Nhận cảnh báo thời gian thực.
  + Thao tác điều khiển thiết bị từ xa.

CHƯƠNG 2. THIẾT KẾ HỆ THỐNG HỒ BƠI THÔNG MINH

2.1 Sơ đồ phân cấp hệ thống



Hình 2.1 Sơ đồ phân cấp

2.2 Mô tả hệ thống

2.2.1 Cấp trường

Là nơi thu thập dữ liệu cảm biến và kết nối các thiết bị chấp hành

a. Cảm biến

- *Cảm biến đo pH:* Giám sát độ kiềm/axit của nước, được kết nối với PLC bằng giao thức Modbus RTU thông qua modul mở rộng.
- *Cảm biến đo Clo:* Theo dõi nồng độ Clo còn dư trong nước, được kết nối với PLC bằng giao thức Modbus RTU thông qua modul mở rộng.
- *Cảm biến mức nước:* Xác định mực nước trong bể, dùng để điều khiển máy bơm và van điện từ,

b. Thiết bị chấp hành

- *Bơm lọc (BƠM LỌC 1.... BƠM LỌC 6) :* Các bơm này có nhiệm vụ lọc và tuần hoàn nước trong hồ bơi theo chu trình tự động, đảm bảo chất lượng nước luôn trong tình

trạng sạch sẽ. Mỗi bơm sẽ hoạt động theo lịch trình đã được lập trình sẵn hoặc theo các chỉ số chất lượng nước.

- *Bơm chìm (WF7, WF8) :* Các bơm chìm này được sử dụng để bơm nước ra khỏi hồ bơi hoặc khu vực phòng kỹ thuật khi xảy ra các sự cố như vỡ đường ống hoặc ngập lụt, đảm bảo an toàn cho toàn bộ hệ thống và các khu vực liên quan.
- *Van điều khiển:* Van này điều chỉnh dòng nước vào bể cân bằng, đảm bảo mực nước trong hồ luôn ở mức chuẩn. Chức năng này được tự động hóa và điều khiển thông qua các cảm biến và PLC.
- *Bơm hóa chất:* Bơm này có chức năng tự động thêm hóa chất vào nước hồ bơi để điều chỉnh pH và các nồng độ hóa chất cần thiết nhằm duy trì chất lượng nước đạt tiêu chuẩn vệ sinh và an toàn cho người sử dụng.
- *Chiếu sáng:* Hệ thống chiếu sáng được điều khiển bởi PLC, cho phép bật/tắt đèn theo thời gian cài đặt sẵn. Hệ thống có thể hoạt động ở chế độ tự động, thủ công hoặc điều khiển từ xa thông qua giao diện Web.
- *Chuông báo động*: Báo động khi có sự cố như rò rỉ, mực nước bất thường và cảnh báo cho nhân viên cứu hộ khi có tình huống đuối nước ở bể bơi

2.2.2 Cấp điều khiển

PLC S7-1200: Bộ điều khiển lập trình Siemens, đóng vai trò trung tâm điều khiển. Nó:

- Nhận dữ liệu từ cảm biến.
- Gửi tín hiệu điều khiển đến các thiết bị chấp hành như bơm, van, chiếu sáng...
- Kết nối với HMI và Raspberry Pi.

HMI (Human-Machine Interface) : Thiết bị giao tiếp giữa người vận hành và hệ thống PLC, hiển thị trạng thái và cho phép điều khiển các thiết bị tại chỗ.

2.2.3 Cấp điều khiển, giám sát

Raspberry Pi 5: Máy tính nhúng dùng làm gateway IoT.

- *Raspberry Pi kết nối với PLC:* để lấy dữ liệu theo thời gian thực thông qua các giao thức Ethernet hoặc Profinet. Điều này giúp kết nối và trao đổi dữ liệu một cách liền mạch giữa các thiết bị trong hệ thống. Truyền dữ liệu lên server Flask.
- *Truyền dữ liệu lên server Flask:* Raspberry Pi gửi dữ liệu thu được lên server Flask, nơi này sẽ xử lý các yêu cầu và phản hồi từ giao diện Web.
- *Nhận lệnh từ giao diện Web:* Hệ thống nhận lệnh điều khiển từ giao diện Web, cho phép điều khiển từ xa các thiết bị trong hồ bơi.

Flask Server: Đóng vai trò là giao diện Web trung tâm để điều khiển và giám sát hệ thống từ xa.

- *Giao diện giám sát từ xa*: Server cung cấp một bảng điều khiển dựa trên Web, cho phép người dùng truy cập và theo dõi tình trạng hoạt động của các thiết bị trong hồ bơi như chất lượng nước, hệ thống chiếu sáng và an toàn.
- *Lưu trữ dữ liệu và phân tích cảnh báo:* Server còn tích hợp chức năng lưu trữ dữ liệu hoạt động của hồ bơi và phân tích các cảnh báo được kích hoạt bởi hệ thống AI.

Trình duyệt Web: Người dùng có thể truy cập giao diện giám sát từ Web qua trình duyệt trên PC hoặc thiết bị di động để giám sát và điều khiển hệ thống từ xa.

2.2.4 Camera IP

Như đã trình bày ở mục 1.2.3, Camera IP có nhiệm vụ thu thập dữ liệu, giám sát video khu vực bể bơi , gửi luồng video về Raspberry Pi thông qua giao thức TCP/IP phục vụ tính năng phát hiện đuối nước bằng AI

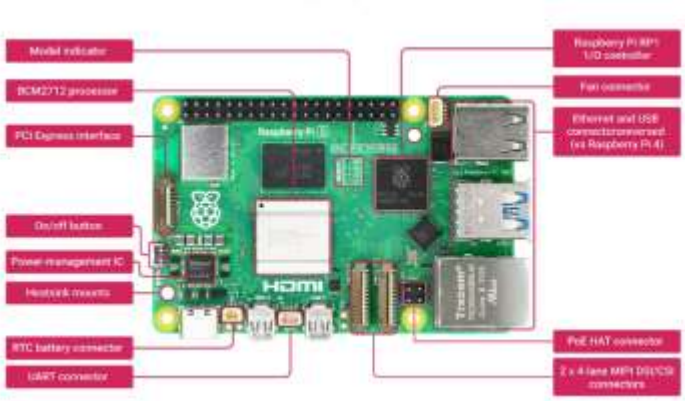
2.3 Danh mục các thiết bị

2.3.1 Raspberry PI 5

Raspberry Pi 5 giúp hệ thống hồ bơi hoạt động linh hoạt và có khả năng mở rộng. Nó đóng vai trò như một cầu nối giữa các cảm biến và PLC với giao diện Web, cho phép điều khiển từ xa và giám sát toàn bộ hệ thống hồ bơi thông qua môi trường internet. Việc sử dụng Raspberry Pi 5 giúp tiết kiệm chi phí phần cứng, dễ dàng triển khai và bảo trì, đồng thời hỗ trợ giao tiếp dữ liệu nhanh chóng và ổn định.

Raspberry Pi 5 có các thông số kỹ thuật như sau:

- Bộ xử lý:
    + Broadcom BCM2712
    + 64-bit ARM Cortex-A76 (ARMv8-A ISA)
    + Lõi tứ
    + Tiến trình SoC 16nm
    + Xung nhịp tối đa 2.4GHz
    + 64KB I & D cache
    + 512KB pre-core L2 cache và 2MB shared L3 cache
- Có thêm lớp kim loại giúp tản nhiệt hiệu quả hơn
- Kết nối Hình ảnh/Camera: 2 cổng x 4-lane MIPI 22-way CSI/DSI
- Lưu trữ: Khe cắm thẻ nhớ MicroSD, hỗ trợ chế độ truyền dữ liệu cao SDR104
- Yêu cầu nguồn điện: 5V/5.0A qua cổng USB type C, với PD (Power Delivery)
- Kích thước: 88mm x 56mm x 17mm



Hình 2.2 Raspberry PI 5

2.3.2 PLC

PLC đóng vai trò như một bộ não giúp tự động hóa các quá trình quan trọng trong hồ bơi như lọc nước, điều khiển hóa chất, và chiếu sáng. Điều này giảm thiểu sự phụ thuộc vào công việc thủ công, đảm bảo hệ thống vận hành ổn định và chính xác. Ở đồ án để thuận tiện cho việc mô phỏng và kết nối các thiết bị chúng em sử dụng PLC Siemens S7-1200 CPU 1214C, DC/DC/DC

Thông số kỹ thuật:

- Mã sản phẩm: S7-1200 6ES7214-1AG40-0XB0.
- Nguồn cấp: Điện áp AC 85-264V, tần số 47-63 Hz, dòng tiêu thụ tối đa 180 mA.
- Bộ nhớ: 50 KB làm việc, 1 MB tải.
- Đầu vào/ra: 14 DI, 10 DO, 2 AI (độ phân giải 10-bit).
- Giao tiếp: PROFINET, TCP/IP, hỗ trợ Web Server.
- Kích thước: 90 x 100 x 75 mm, trọng lượng ~420 g.
- Giao thức kết nối: PROFINET, TCP/IP, Web Server.
- Output: Digital Output (DO), Analog Output (AO) (tùy model mở rộng) .

Hình 2.3 PLC S7-1200

2.3.3 Modul mở rộng

a. Module truyền thông CM 1241 RS485

Module CM 1241 được thiết kế để bổ sung kết nối truyền thông RS-485 cho PLC S7-1200, hỗ trợ các chuẩn Modbus RTU Master/Slave, ASCII và USS, giúp kết nối với các cảm biến hoặc thiết bị tương thích RS-485 một cách dễ dàng. Thiết bị này phù hợp với firmware ≥ V4.0 và có khả năng cách ly tín hiệu lên tới 500 V giữa ngõ field và logic. Module cấp nguồn trực tiếp từ CPU PLC, hoạt động trong dải nhiệt độ –20…+60 °C, có kích thước nhỏ gọn (71 × 100 × 75 mm) , và rất thuận tiện khi cấu hình thông qua TIA Portal sử dụng GSD hoặc thư viện tương thích.

Hình 2.4 Modul truyền thông CM 1241

b. Các module DI/DO mở rộng

SM 1221/SM 1222/SM 1223 để thu thập tín hiệu kỹ thuật số từ cảm biến mức và điều khiển tải ra như bơm, van, relay, còi báo.

Hình 2.5 Các module DI/DO mở rộng

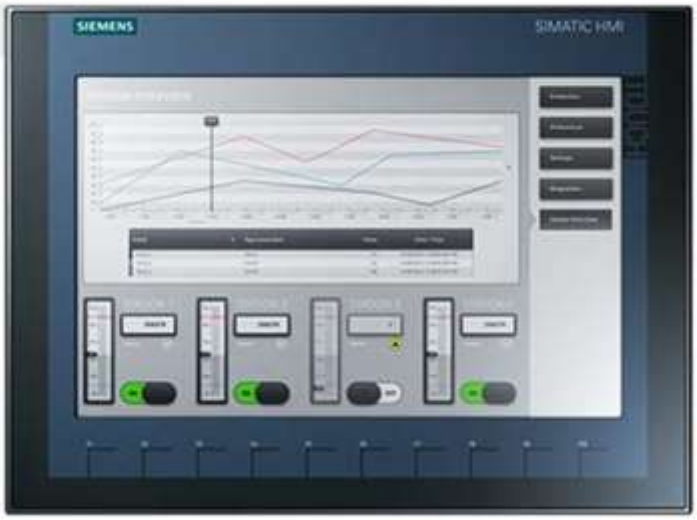
2.3.4 Màn hình HMI

HMI là giao diện giữa người vận hành và hệ thống PLC, giúp hiển thị thông tin về trạng thái của các thiết bị trong hồ bơi như bơm, hệ thống lọc, van điều khiển, và các

thông số chất lượng nước như pH và Clo, từ đó người sử dụng có thể kịp thời can thiệp và điều chỉnh hệ thống. HMI giúp tăng tính hiệu quả và giảm rủi ro do thao tác không chính xác hoặc lỗi kỹ thuật trong quá trình vận hành.

Với mục tiêu dự án là thiết kế và mô phỏng, kiểm nghiệm hệ thống tự động hóa hồ bơi, nhóm đã quyết định lựa chọn màn hình HMI với các thông số như sau:

- Dòng sản phẩm: KTP1200 Basic color DP
- Thiết kế màn hình: Thiết kế màn hình hiển thị màn ảnh rộng TFT, đèn nền LED
- Kích thước màn hình: 12 in (261.1 x 163.2 mm)
- Số lượng màu sắc: 65 536
- Độ phân giải: 1280 x 800 Pixel



Hình 2.6 Màn hình HMI KTP1200 Basic color DP

2.3.5 Cảm biến

a. Rơ le mức nước Omron 61F-GP-N

Cảm biến mực nước giúp duy trì mực nước ổn định, tránh tình trạng ngập hoặc thiếu nước, từ đó đảm bảo rằng các thiết bị bơm và lọc hoạt động đúng theo yêu cầu. Việc này giúp hệ thống hoạt động ổn định hơn và giảm thiểu các sự cố có thể xảy ra trong quá trình vận hành.

Bảng 2.1 Bảng thông số kỹ thuật Omron 61F-GP-N

Thông số	Giá trị
Model	Omron 61F-GP-N
Nguồn cấp	220 VAC, 50/60 Hz
Công suất tiêu thụ	~3.5 VA
Loại tiếp điểm	1C (SPDT – NO/NC)
Dòng chịu tải	5 A _ 220 VAC (tải trở) , 3 A _ 220 VAC (tải cảm)

Thời gian đáp ứng	ON: 80 ms / OFF: 160 ms
Số điện cực	3 (E1, E2, COM)
Điện áp điện cực	8 VAC
Dòng điện điện cực	Dưới 1 mA
Chiều dài dây cực	Tối đa 100 m
Nhiệt độ hoạt động	–10 đến +55°C
Trọng lượng	170 g
Lắp đặt	Dạng socket PTF08A

Hình 2.7 Rơ le mức nước Omron

b. Cảm biến đo pH (S-pH-01)

Để giám sát chất lượng nước sau quá trình lọc, cũng như đảm bảo nước trong hồ bơi duy trì ở mức pH an toàn, ở cuối đường ống trả sẽ được lắp cảm biến đo pH, dữ liệu từ cảm biến sẽ được đưa về PLC. Sau đó PLC sẽ so sánh với giá trị an toàn từ đó đưa ra cảnh báo cho người vận hành kịp thời khắc phục nguyên nhân làm cho độ pH không đạt chuẩn.

Bảng 2.2 Bảng thông số kỹ thuật cảm biến đo pH (S-pH-01)

Thông số	**Giá trị**
Model	S-pH-01
Thông số đo	Độ pH
Ngõ ra tín hiệu	RS485 (Modbus RTU), Analog 0–2V
Tốc độ baud	9600 bps
Địa chỉ mặc định	01
Điện áp hoạt động	3.3 ~ 5.5V DC
Dải đo	0 – 14 pH
Độ phân giải	0.01 pH

Độ chính xác	±0.1 pH
Nhiệt độ hoạt động	0 – 60°C
Thời gian phản hồi	< 1 giây
Chống nước	IP68
Chiều dài cáp	1 m
Giao thức truyền thông	Modbus RTU (8N1)

Hình 2.8 Cảm biến đo pH (S-pH-01)

c. Cảm biến đo Clo dư (BH-485-CL)

Tương tự, với nồng độ Clo dư ta cũng sẽ lắp một cảm biến đo Clo dư trong nước và được truyền thông với PLC qua giao thức Modbus RTU.

Bảng 2.3 Bảng thông số kỹ thuật cảm biến đo Clo dư (BH-485-CL)

Thông số	**Giá trị**
Model	BH-485-CL
Thông số đo	Clo dư (Residual Chlorine)
Ngõ ra tín hiệu	RS485 (Modbus RTU)
Tốc độ baud	9600 bps
Địa chỉ mặc định	01
Điện áp hoạt động	12~24V DC
Dải đo	0 – 10 mg/L
Độ phân giải	0.01 mg/L
Độ chính xác	±5% giá trị đo
Nhiệt độ hoạt động	0 – 60°C
Áp suất làm việc	≤ 0.3 MPa
Chất liệu	PVC
Chống nước	IP68
Tuổi thọ	Trên 12 tháng

Hình 2.9 Cảm biến đo Clo dư (BH-485-CL)

2.3.6 Van điện từ

Kết hợp với cảm biến mực nước giúp điều chỉnh dòng nước cấp vào bể cân bằng, giúp duy trì mực nước ổn định.

Bảng 2.4 Bảng thông số kỹ thuật van điện từ

Thông số	Giá trị
Model	NABA Series – 10 Nm
Điện áp hoạt động	AC/DC 24V ±10% hoặc AC 230V ±10%
Tần số	50/60 Hz
Mô-men xoắn danh định	10 Nm
Thời gian chạy (90°)	30 / 60 / 90 giây (tùy chọn)
Tín hiệu điều khiển	ON/OFF hoặc Modulating (0–10V / 2–10V)
Tín hiệu phản hồi	0–10V hoặc 2–10V (dành cho loại modulating)
Công suất tiêu thụ	2.5 W (standby) , tối đa 6.5 W
Cấp bảo vệ	IP54
Nhiệt độ môi trường	-10°C đến +50°C
Góc quay	90° ±3°
Giao diện cơ khí	Trục vuông 10x10 mm hoặc tròn Ø10 mm
Chiều dài cáp	1 mét
Trọng lượng	0.9 kg
Phụ kiện tùy chọn	Công tắc giới hạn, bộ báo vị trí, hoặc tay quay khẩn cấp

2.3.7 Bơm lọc

Bơm lọc có nhiệm vụ lọc nước hồ bơi, đảm bảo rằng nước trong hồ luôn sạch sẽ và trong suốt. Bơm này hoạt động liên tục để lọc các chất rắn và vi khuẩn trong nước.

a. Tính toán lựa chọn thiết bị

Thể tích bể bơi : 1010 m^3

Thể tích bể cân bằng: 40 m^3

$$Q_B = \frac{V}{T}\left(m^3/h\right) = \frac{1010+40}{5} = 210\left(m^3/h\right) = 0,056\left(m^3/s\right) \quad (2.1)$$

Trong đó:

- Q_B : Lưu lượng lọc (m^3/h)
- T: Thời gian lọc nước đối với bể bơi (h)
- V: Thể tích nước cần xử lý (m^3)

Suy ra ta chọn bình lọc như sau: Chọn 6 (5 hoạt động + 1 dự phòng) bình lọc cát hồ bơi emaux S1200 có có lưu lượng 52.8 m^3/h, kích thước: 1410x1200mm.

Dựa vào số lượng bình lọc ta chọn 6 bơm lọc (5 hoạt động + 1 dự phòng)

$$P = \frac{Q \times H \times M_{H_2O}}{102 \times H'} = \frac{(0,056 \div 5) \times 17 \times 1000}{102 \times 0,8} = 2,33(kW) \quad (2.2)$$

Trong đó:

- P : Công suất máy (kW).
- Q: Lưu lượng nước được bơm (m^3/s).
- H: Cột nước bơm (m).

$$H_b = h_{hh} + h_h + h_d + h_{cd} + h_{td} = 17 \quad (2.3)$$

  + h_{hh} : chiều cao hình học mực nước từ đầu hút tới đầu xả bơm (10m)
  + h_h : tổn thất theo chiều dài ống hút (0,2 m)
  + h_d : tổn thất theo chiều dài ống đẩy (1,2 m)
  + h_{cd} : tổn thất cục bộ theo đường ống hút và đẩy

$$h_{cd} = 30\% \times (h_h + h_d) = 0,4(m) \quad (2.4)$$

  + h_{td} : tổn thất tại cột lọc (5m)

- M_{H_2O} : tỷ trọng nước (1000kg/ m^3) .
- H': Hiệu suất máy bơm (0, 8 – 0, 9)

Bảng 2.5 Tính toán thông số tổn thất

Diễn giải	Ống hút	Ống đẩy	Vật liệu ống	Áp lực
Đường kính trong (mm)	207, 80	147, 60	uPVC	PN10
Tốc độ	0, 79	1, 56		
1000i	3, 07	15, 73		
Chiều dài	67	77		
Tổn thất	0, 2	1, 2		

Hệ số an toàn bơm: 0, 43

$$P_{mb} = \frac{2,33}{0,43} = 5,42(kW) \quad (2.5)$$

b. Thông số kỹ thuật

Dưới đây là bảng thông số kỹ thuật tổng quát của dòng máy bơm Hydrau-Power theo tài liệu hãng.

Bảng 2.6 Bảng thông số kỹ thuật bơm lọc

Thông số	Giá trị
Model	Hydrau-Power Series
Loại bơm	Centrifugal Pump – End Suction
Công suất hoạt động	5.5kW
Điện áp	3 phase – 400V / 50Hz
Cấp bảo vệ	IP55
Lưu lượng tối đa	90 – 168 m³/h tùy model
Cột áp tối đa	12 – 22 m tùy model
Vật liệu thân bơm	Composite / Thermoplastic reinforced
Cánh bơm	Noryl hoặc PPO
Trục bơm	Inox 316
Đường ống hút/xả	DN80 (3 inch) / DN65 (2.5 inch)

Code 380V/50Hz	Model	Connection Size ANSI / DIN	Horsepower (hp)	Strainer Vol (L)	RPM	Power (kW)	Noise (dB)	Head (m) / Flow Rate (m³/h) 6	8	10	12	14	16
9023902	APS750P	4"/DN100	7.5	13	2850	5.5	72	130	120	110	105	90	80
9023904	APS1500P	6"/DN150	15	30	1450	11	68	--	270	260	250	240	210

Hình 2.10 Bơm lọc

2.3.8 Máy điện phân muối

Hệ thống điện phân muối là một giải pháp thân thiện môi trường để tạo ra clo tự nhiên cho hồ bơi từ muối hòa tan trong nước. Thiết bị hoạt động dựa trên nguyên lý điện phân dung dịch NaCl, tạo ra clo hoạt tính khử khuẩn.

a. Tính toán lựa chọn thiết bị

Hệ thống khử trùng bể bơi gồm 8 máy điện phân muối có chức năng điện phân tạo ion Cl - diệt khuẩn trong buồng điện phân với tốc độ

$$50(g/h)\times 5(h)\times 8 = 2000(g) \quad (2.6)$$

cho 1050 m^3 nước tương ứng với 1,9 gram/m^3 phù hợp với tiêu chuẩn xử lý 1-2mg/l.

b. Thông số kỹ thuật

Dưới đây là bảng thông số kỹ thuật tổng hợp của thiết bị ĐPM theo tài liệu nhà sản xuất.

Bảng 2.7 Bảng thông số kỹ thuật máy điện phân muối

Thông số	Giá trị
Tên thiết bị	Bộ điện phân muối (Salt Chlorinator)
Nguyên lý hoạt động	Điện phân dung dịch muối NaCl để tạo clo
Điện áp hoạt động	AC 220V / 50Hz
Công suất tiêu thụ	Khoảng 200 – 600 W (tuỳ công suất hệ thống)
Nồng độ muối yêu cầu	3, 000 – 6, 000 ppm (tối ưu khoảng 5, 000 ppm)
Lưu lượng nước yêu cầu	≥ 5 m^3/h
Áp suất vận hành	≤ 2.5 bar
Vật liệu buồng điện phân	Titanium phủ Ruthenium / Iridium
Hiển thị	Màn hình LED hoặc LCD
Bảo vệ	Bảo vệ quá nhiệt, quá áp, không có nước, nồng độ muối thấp
Tín hiệu đầu vào	Có thể tích hợp relay điều khiển / cảm biến lưu lượng
Ứng dụng	Khử trùng nước hồ bơi, spa

Hình 2.11 Máy điện phân muối

2.3.9 Bơm định lượng

Bơm định lượng là một thành phần quan trọng trong hệ thống xử lý nước hồ bơi, đảm bảo việc cung cấp chính xác hóa chất như clo hoặc axit để duy trì chất lượng nước đạt chuẩn.

a. Tính toán lựa chọn thiết bị

Thể tích hồ bơi: 1050 m^3.

- Lượng axit cần châm: Dựa trên nhu cầu điều chỉnh pH từ 8.0 xuống 7.4.
- Nồng độ axit HCl: Sử dụng dung dịch HCl 30% (có thể từ 15%-35%) để đạt hiệu quả và an toàn.
- Theo kinh nghiệm chung, để giảm 0.1 đơn vị pH cho mỗi mét khối nước, cần khoảng 0.01 - 0.02 L dung dịch HCl 30%.

Tổng lượng axit cần châm:

Để giảm từ pH 8.0 xuống 7.4 (giảm 0.6 đơn vị pH) , cần châm khoảng:

$$0.01l \times 1050\ m^3 \times 5 = 52.5l\ HCl\ 30\%. \quad (2.7)$$

b. Thông số kỹ thuật

Dựa vào thông số tính toán được, dưới đây là bảng thông số kỹ thuật của bơm định lượng được sử dụng trong đề tài. Bơm có khả năng tự điều chỉnh lưu lượng bằng cách thông qua tín hiệu analog từ cảm biến đo pH đi kèm.

Bảng 2.8 Bảng thông số kỹ thuật bơm định lượng

Thông số	Giá trị
Model	DP-Mini / DP-01 / DP-02 / ...
Loại bơm	Bơm định lượng điện từ
Lưu lượng	0.5 đến 20 L/h (tuỳ model)
Áp suất làm việc	0.5 đến 10 bar
Điện áp hoạt động	AC 220V / 50Hz
Công suất	30 – 45 W
Vật liệu đầu bơm	PVDF, PVC hoặc SS316
Đầu vào hóa chất	Ống PVC/FKM
Điều chỉnh lưu lượng	Tự động qua tín hiệu điều khiển (4–20 mA)

2.3.10 Bơm chìm

a. Tính toán lựa chọn thiết bị

Thể tích hố bơm chìm:

$$V = 2 \times 0.75 \times 0,8 = 1,2m^3 \quad (2.8)$$

Chọn thời gian để hút toàn bộ nước trong hố bơm chìm là 3 phút

Lưu lượng bơm cần thiết Q:

$$Q = \frac{1,2}{180} = 0,006(m^3/s) \quad (2.9)$$

Áp dụng công thức (1) với cột áp bơm 8 m ta tính được công suất máy bơm

$$P = \frac{Q \times H \times M_{H_2O}}{102 \times H'} = \frac{0,006 \times 8 \times 1000}{102 \times 0,8} = 0,59(kW) \quad (2.10)$$

Hệ số an toàn bơm: 0,43

$$P_{mb} = \frac{0,59}{0,43} = 1,36(kW) \quad (2.11)$$

b. Thông số kỹ thuật

Dùng cho hệ thống bơm sự cố, thông số kỹ thuật được trình bày ở bảng sau

Bảng 2.9 Bảng thông số kỹ thuật bơm chìm

Thông số	Giá trị
Model	Pentax DMT 210
Loại bơm	Bơm thải chìm thân ngang
Lưu lượng	0 - 60 m³/h
Điện áp hoạt động	3 pha / 50Hz
Công suất	1.5kW / 2HP
Cột áp	5.2 – 19.9 m
Cấp độ chống nước	IP68

TYPE		P2		P1 (kW)		Q (m³/h - l/min)											
						0	6	12	18	24	30	36	42	48	54	60	66
						0	100	200	300	400	500	600	700	800	900	1000	1100
1~	3~	(HP)	(kW)	1~	3~	H (m)											
DM 160	DMT 160	1,5	1,1	2,8	2,6	17,0	15,3	13,8	12,3	11,0	9,8	8,4	7,1	5,5	3,8	-	-
DM 210	DMT 210	2	1,5	3,3	3,1	19,9	18,4	16,7	15,2	13,8	12,4	11,3	10,1	8,6	7,0	5,2	-
-	DMT 310	3	2,2	-	4,1	23,9	22,2	20,6	19,1	17,8	16,3	15,0	13,8	12,3	10,9	9,1	6,4

Hình 2.12 Bơm chìm

2.3.11 Máy gia nhiệt

Máy gia nhiệt Heatpump là thiết bị sử dụng nguyên lý bơm nhiệt để làm nóng nước hồ bơi một cách tiết kiệm năng lượng. Thay vì sử dụng điện trở để sinh nhiệt trực tiếp,

Heatpump tận dụng nhiệt từ môi trường không khí để gia nhiệt cho nước, giúp tăng hiệu quả và giảm chi phí vận hành. Dưới đây là bảng thông số kỹ thuật tổng hợp từ tài liệu nhà sản xuất.

a. Tính toán lựa chọn thiết bị

Thông tin đầu vào:

- Thể tích bể: 1010 m³
- Nhiệt độ nước ban đầu: 26 °C
- Nhiệt độ mong muốn: 29 °C
- Thời gian gia nhiệt: 72 giờ
- Số lượng máy gia nhiệt: 2

Nhiệt lượng cần thiết để tăng nhiệt độ nước

$$Q = m \times c \times \Delta T = 1{,}01 \times 10^6 \times 4{,}186 \times 3 = 12{,}68 \times 10^6\ kJ \qquad (2.12)$$

Trong đó:

- m: khối lượng nước (kg) = 1010 × 1000 = 1.01 × 10⁶ kg
- c: nhiệt dung riêng của nước = 4.186 kJ/kg°C
- ΔT: độ tăng nhiệt độ = 3 °C

Công suất gia nhiệt

$$P = \frac{Q}{t} = \frac{12{,}68 \times 10^6}{72 \times 3600} = 48{,}9(kW) \qquad (2.13)$$

Công suất mỗi máy

$$P_{gn} = \frac{48{,}9}{2} = 24{,}5(kW) \qquad (2.14)$$

→ Chọn máy gia nhiệt có công suất 29, 5kW (dự phòng tổn thất nhiệt)

b. Thông số kỹ thuật

Bảng 2.10 Bảng thông số kỹ thuật máy gia nhiệt

Thông số	Giá trị
Loại thiết bị	Máy gia nhiệt bơm nhiệt (Heatpump) cho hồ bơi
Nguồn điện	AC 220V/1 pha hoặc 380V/3 pha, 50Hz
Công suất sưởi	5 – 35 kW (tuỳ model)
COP (hệ số hiệu suất)	≥ 5.0
Nhiệt độ hoạt động ngoài trời	-10°C đến +43°C
Dải nhiệt độ nước	20°C – 40°C
Môi chất lạnh	R410A / R32
Lưu lượng nước yêu cầu	13 m³/h
Áp suất tối đa	≤ 3.0 bar

Mức ồn	≤ 48 – 58 dB (tùy model)
Loại trao đổi nhiệt	Titanium trong vỏ PVC hoặc Inox
Chức năng	Sưởi hoặc giữ nhiệt, tự động xả băng
Điều khiển	Bảng điều khiển kỹ thuật số / cảm ứng

Hình 2.13 Máy gia nhiệt

2.3.12 Bơm tuần hoàn cho máy gia nhiệt

a. Tính toán lựa chọn thiết bị

Dựa vào lưu lượng nước cần thiết của máy gia nhiệt ta tính được công suất cần cho máy bơm

$$P = \frac{Q \times H \times M_{H_2O}}{102 \times H'} = \frac{(13 \times 2 \div 3600) \times 19 \times 1000}{102 \times 0,8} = 1,68(kW) \quad (2.15)$$

Hệ số an toàn là 0, 43

$$P_{mb} = \frac{1,68}{0,43} = 3,9(kW) \quad (2.16)$$

→ Chọn bơm có công suất là 4 kW và cột áp là 19m

b. Thông số kỹ thuật

Bảng 2.11 Bảng thông số kỹ thuật bơm cho máy gia nhiệt

Thông số	Giá trị
Model	Hydrau-Power Series
Loại bơm	Centrifugal Pump – End Suction
Công suất hoạt động	4kW
Điện áp	3 phase – 400V / 50Hz
Cấp bảo vệ	IP55
Lưu lượng tối đa	220 m³/h
Cột áp tối đa	12 – 22m
Vật liệu thân bơm	Composite / Thermoplastic reinforced
Cánh bơm	Noryl hoặc PPO

Trục bơm	Inox 316

Code 380V/50Hz	Model	Connection Size ANSI / DIN	Horsepower (hp)	Strainer Vol (L)	RPM	Power (kW)	Noise (dB)	Head (m) 6	8	10	12	14	16
								Flow Rate (m³/h)					
9023901	APS550P	4"/DN100	5.5	13	2850	4	72	95	90	82	75	65	50
9023902	APS750P	4"/DN100	7.5	13	2850	5.5	72	130	120	110	105	90	80
9023903	APS1000P	6"/DN150	10	30	1450	7.5	68	--	210	185	180	160	135
9023904	APS1500P	6"/DN150	15	30	1450	11	68	--	270	260	250	240	210

2.3.13 Chiếu sáng

Để tính toán chiếu sáng ta dựa vào côngg thức sau

$$\Phi_{total} = S \times E \quad (2.17)$$

Trong đó:

- Φ_{total}: tổng quang thông yêu cầu (lm)
- S: Diện tích hồ bơi (m^2)
- E: Độ rọi yêu cầu (lux, tức lm/m^2)

Số lượng đèn được tính bằng công thức

$$N = \frac{\Phi_{total}}{\Phi_d} \quad (2.18)$$

Trong đó:

- Φ_{total}: tổng quang thông yêu cầu
- Φ_d: quang thông mỗi đèn

Bảng 2.12 Tính chọn chiếu sáng

BẢNG TÍNH CHỌN SỐ LƯỢNG ĐÈN CHIẾU SÁNG						
TIÊU CHUẨN		**TÍNH CHỌN**				
LOẠI HỒ BƠI	**ĐỘ RỌI TIÊU CHUẨN Lumen/m²**	**CÔNG SUẤT ĐÈN (W)**	**QUANG THÔNG (lm)**	**SỐ LƯỢNG ĐÈN (Cái)**		**DIỆN TÍCH ĐÈN CHIẾU SÁNG (m^2)**
				Tính toán	*Lựa chọn*	
MAIN POOL	100	36	2900	25.9	26 cái	750
LOUNGE POOL	100	3	250	8.0	8 cái	20
POOL BAR	150	2	200	9.75	10 cái	13
STEP MAIN POOL	150	2	200	15.75	16 cái	21
STEP LOUNGE POOL	150	2	200	7.5	8 cái	10
POOL DECK	100	2	200	7.5	8 cái	15

Trong chương 2 này, đã trình bày sự phân cấp rõ ràng giữa các cấp độ: cấp trường (thiết bị chấp hành và cảm biến), cấp điều khiển (PLC và HMI) và cấp giám sát

(Raspberry Pi và Webserver). Cùng với đó là các thiết bị lựa chọn, nhằm giải quyết các vấn đề thực tiễn trong vận hành hồ bơi như kiểm soát chất lượng nước, chiếu sáng, và cảnh báo an toàn.

Bảng 2.13 Bảng thống kê danh mục các thiết bị chính

STT	Tên Thiết Bị	Số lượng	Ghi chú
1	Máy bơm lọc	6	Bơm tuần hoàn nước hồ bơi
2	Máy điện phân muối	6	Gắn sau mỗi bình lọc
3	Bơm định lượng hóa chất	2	Bơm châm pH và Clo
4	Cảm biến mức nước	2	Bể cân bằng + hố bơm chìm
5	Cảm biến pH	1	Gắn vào đường ống nước sau xử lý
6	Cảm biến Clo dư	1	Gắn vào đường ống nước sau xử lý
7	Bơm chìm (bơm sự cố)	2	Dưới bể cân bằng
8	Van điện từ	1	Điều khiển nước cấp vào bể cân bằng
9	Máy gia nhiệt (Heatpump)	2	Trong vùng HEATPUMP SYSTEM
10	Bơm tuần hoàn cho máy gia nhiệt	2	Hút và đẩy nước qua heat exchanger
11	Camera IP	4	Theo mô tả đề tài, gắn quanh hồ (AI giám sát)
12	Raspberry Pi 5	1	Gateway AI, server điều khiển Web, SCADA
13	PLC Siemens S7-1200 CPU 1214C, DC/DC/DC 6ES7214-1AG40-0XB0.	1	Trung tâm điều khiển toàn hệ thống
14	HMI KTP1200 Basic PN	1	Giao tiếp giữa người vận hành và PLC
15	Modul SM 1223 8 DI/8 DO	2	Modul mở rộng đầu ra, đầu vào
16	Module CM 1241 RS485	1	Modul truyền thông

CHƯƠNG 3. LƯU ĐỒ THUẬT TOÁN ĐIỀU KHIỂN

3.1 Hệ thống xử lý nước

3.1.1 Hệ thống bơm lọc:

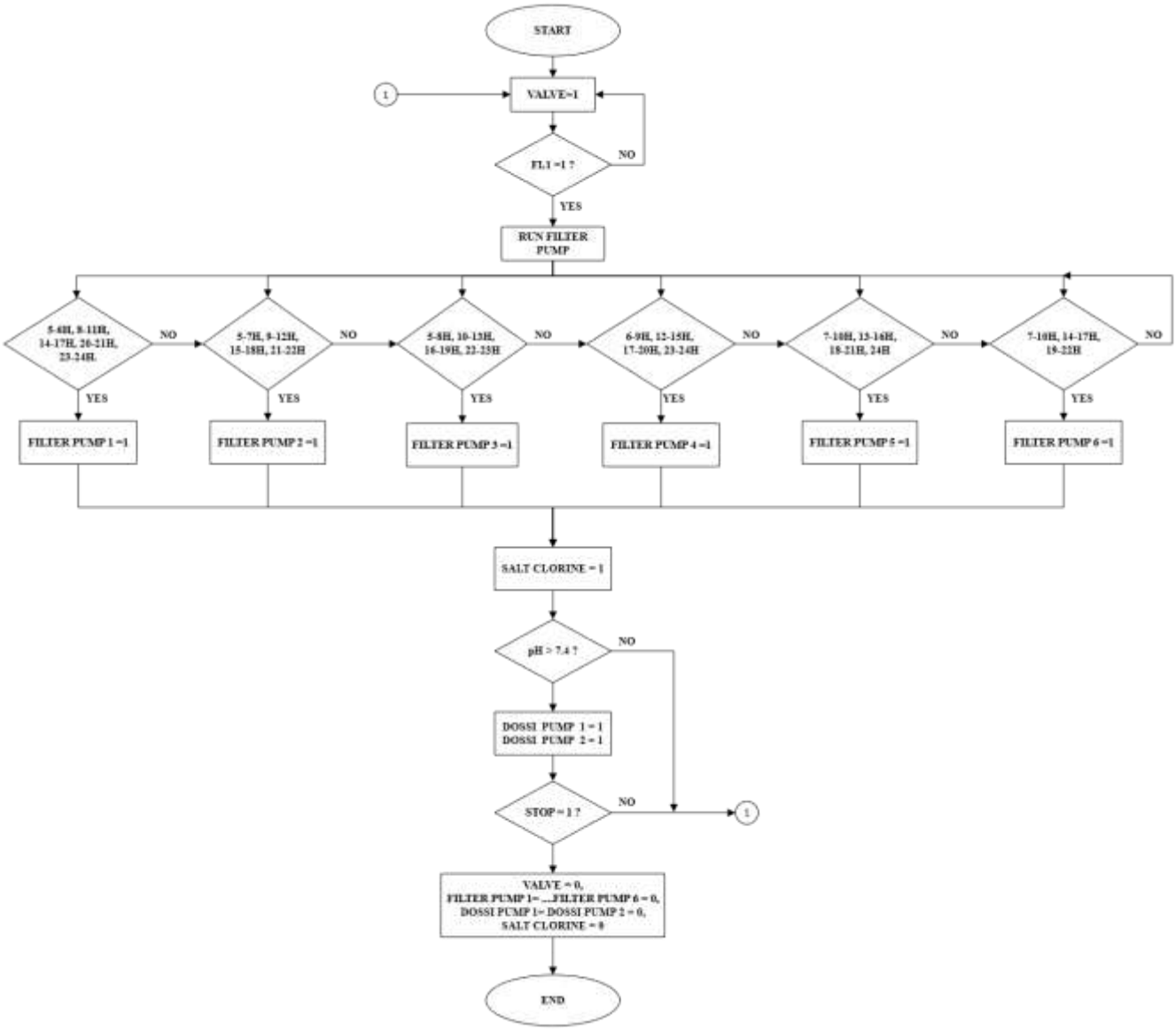
- Chế độ vận hành bằng tay: Thao tác tại tủ điện thông qua công tắc chuyển mạch A – 0 – M.
- Chế độ vận hành tự động: 6 bơm lọc hoạt động luân phiên theo tín hiệu timer, được cài đặt thời gian theo bảng và sơ đồ triết lý điều khiển bơm.
- Cho phép vận hành từ xa: Hệ thống có thể điều khiển từ xa qua SCADA/Web, cho phép theo dõi tình trạng hoạt động theo thời gian thực.

3.1.2 Hệ thống khử trùng:

- Chế độ vận hành bằng tay: Thao tác tại tủ điện thông qua công tắc chuyển mạch A – 0 – M.
- Chế độ vận hành tự động: Hoạt động đồng thời với tín hiệu từ hệ thống bơm lọc.

3.1.3 Hệ thống châm hóa chất:

Chế độ vận hành tự động: Hoạt động theo tín hiệu từ đầu dò điện cực (đo nồng độ pH, clo, v.v.) .



Hình 3.1 Lưu đồ thuật toán của hệ thống xử lý nước

3.1.4 Hệ thống heatpump

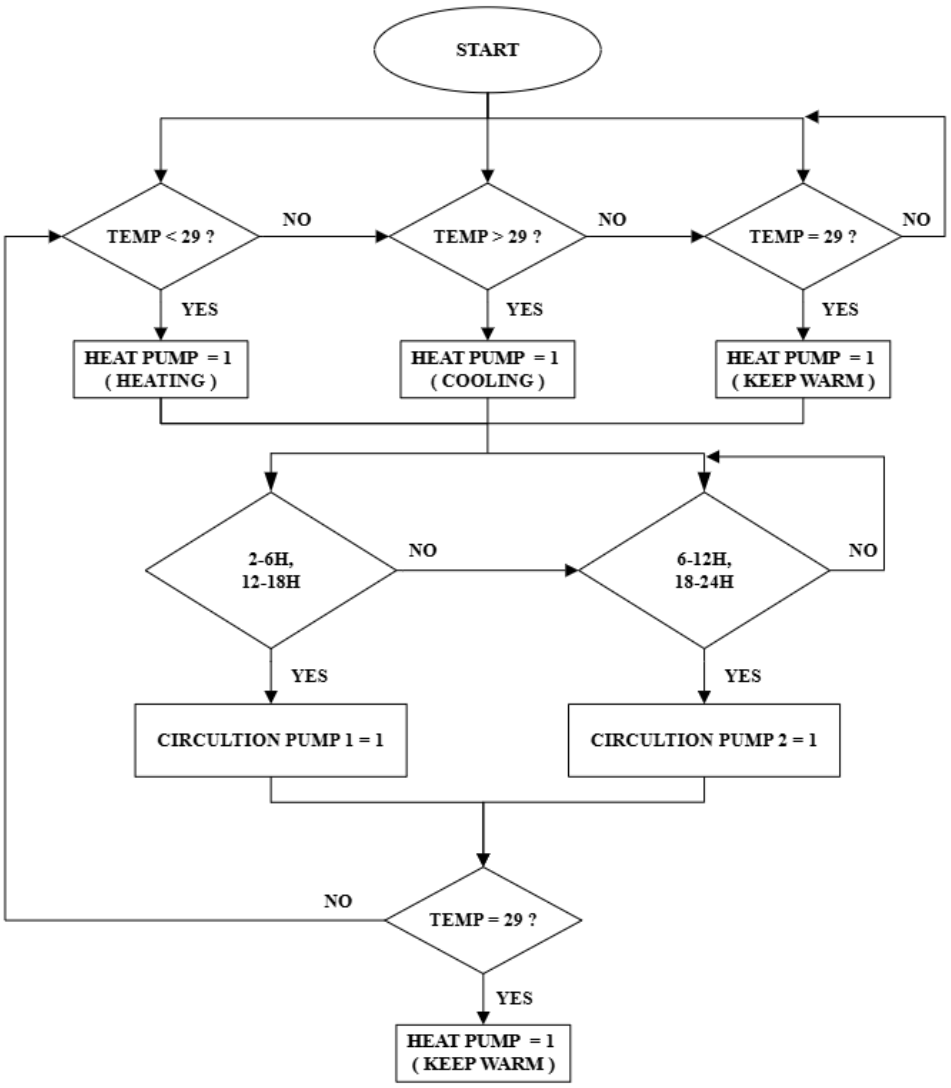
❖ Hoạt động trên bảng điều khiển gắn trên heatpump:

- Bảng điều khiển (Controller) gắn trên Heatpump là nơi cài đặt và hiển thị các thông số vận hành chính.
- Người vận hành sẽ cài đặt:
  + Nhiệt độ nước mong muốn (ví dụ: 29°C theo yêu cầu vận hành) .
  + Chế độ vận hành: Làm nóng (Heating), Duy trì nhiệt độ (Keep Warm), Làm lạnh (Cooling).
- Nguyên lý hoạt động:

+ Khi nhiệt độ nước trong hồ bơi thấp hơn mức cài đặt, cảm biến nhiệt sẽ gửi tín hiệu về bộ điều khiển.
+ Bộ điều khiển kích hoạt máy nén và quạt dàn nóng để bắt đầu quá trình trao đổi nhiệt.
+ Heatpump lấy nhiệt từ không khí môi trường, qua môi chất lạnh (gas lạnh), nén và trao đổi nhiệt vào nước qua bình trao đổi nhiệt (Heat Exchanger).
+ Khi nhiệt độ nước đạt tới mức yêu cầu, hệ thống sẽ tự động dừng hoặc chuyển sang chế độ duy trì nhiệt độ.

❖ Tủ điện cấp nguồn cho heatpump và bơm tuần hoàn:

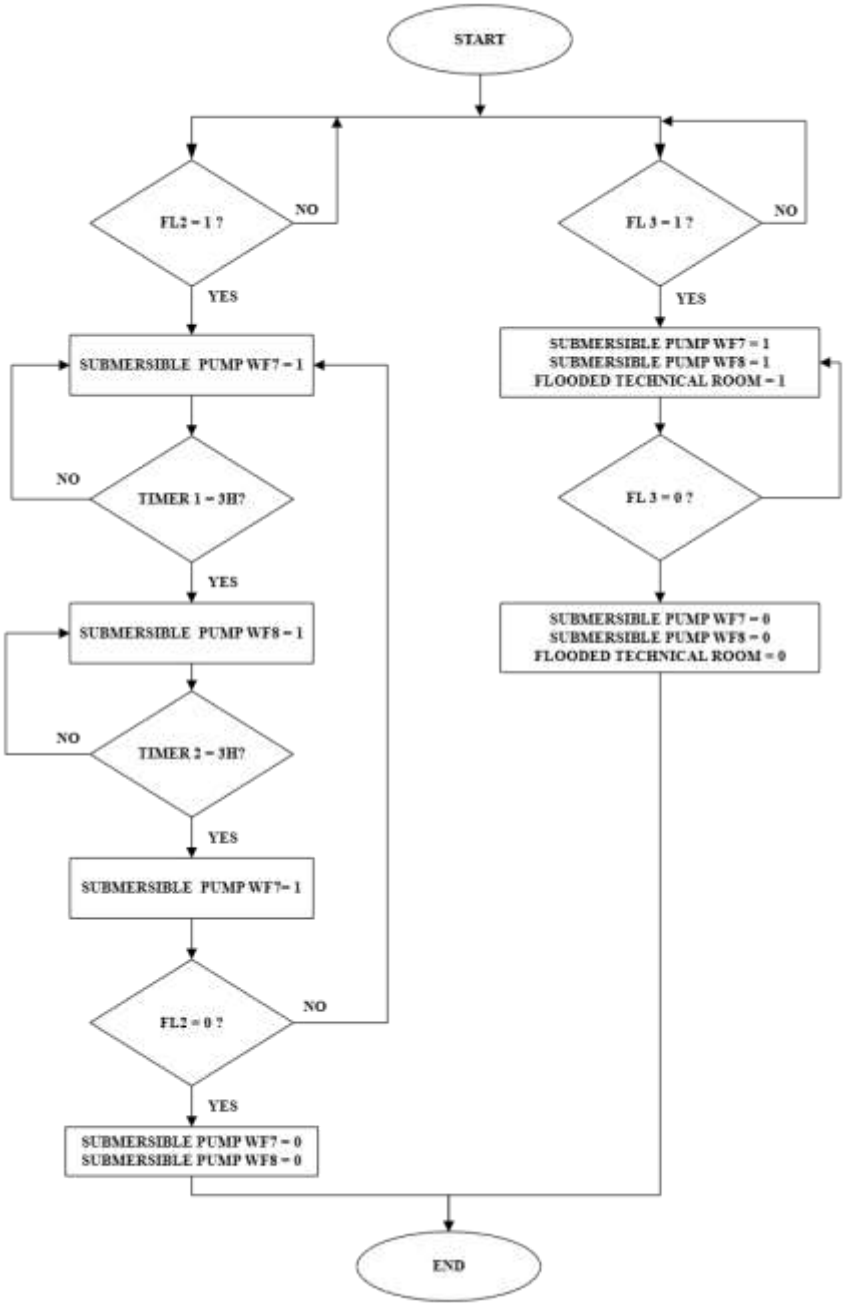
- Hệ thống Heatpump được cấp nguồn thông qua MCCB và RCCB trong tủ điện.
- Hệ thống bơm tuần hoàn:
  + Chế độ vận hành bằng tay: Thao tác tại tủ điện thông qua công tắc chuyển mạch A–O–M.
  + Chế độ vận hành tự động: 2 bơm tuần hoàn hoạt động theo tín hiệu gọi bơm từ Heatpump và luân phiên thông qua cài đặt tại timer PLC.
  + Cho phép vận hành từ xa: Người dùng có thể điều khiển, giám sát hoạt động Heatpump và bơm tuần hoàn từ SCADA hoặc Web.



Hình 3.2 Lưu đồ thuật toán của hệ thống Heatpump

3.2 Hệ thống bơm sự cố:

- Chế độ vận hành bằng tay: Thao tác tại tủ điện thông qua công tắc chuyển mạch A–O–M.
- Chế độ vận hành tự động: Hoạt động theo tín hiệu từ cảm biến mực nước gắn tại hồ bơm chìm.
  + Ở mức nước thấp: 2 bơm chìm hoạt động luân phiên.
  + Ở mức nước cao: Cả 2 bơm hoạt động đồng thời.
- Cho phép vận hành từ xa: Giám sát mực nước và trạng thái hoạt động của bơm sự cố từ xa, đồng thời nhận cảnh báo sự cố ngay lập tức.

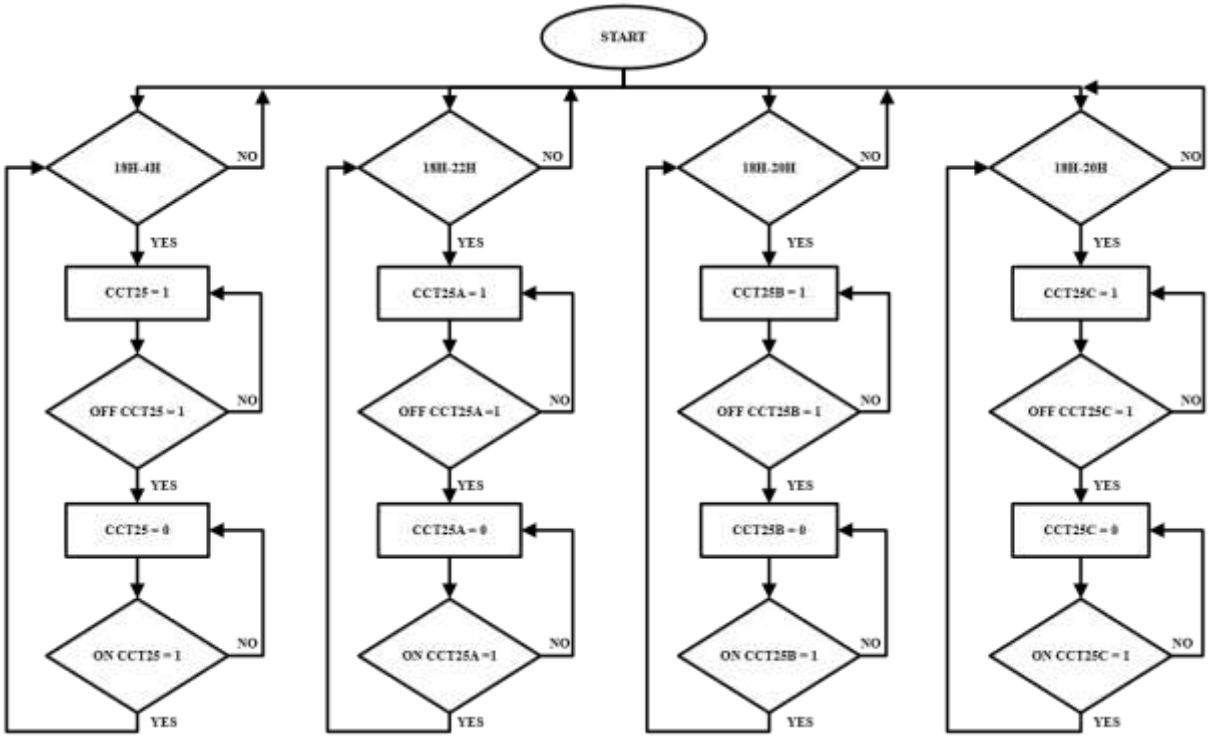


Hình 3.3 Lưu đồ thuật toán của hệ thống bơm sự cố

3.3 Hệ thống chiếu sáng

Như đã trình bày ở phần nguyên lý, hệ thống chiếu sáng sẽ được vận hành bằng 3 chế độ

- Chế độ vận hành bằng tay: Thao tác tại tủ điện thông qua công tắc chuyển mạch A–O–M.
- Chế độ vận hành tự động: Hệ thống đèn hồ bơi được chia thành 4 line đèn: CCT25, CCT25A, CCT25B, CCT25C, hoạt động theo tín hiệu thời gian định sẵn.
- Cho phép vận hành từ xa: Người dùng có thể bật/tắt hệ thống đèn từ Web

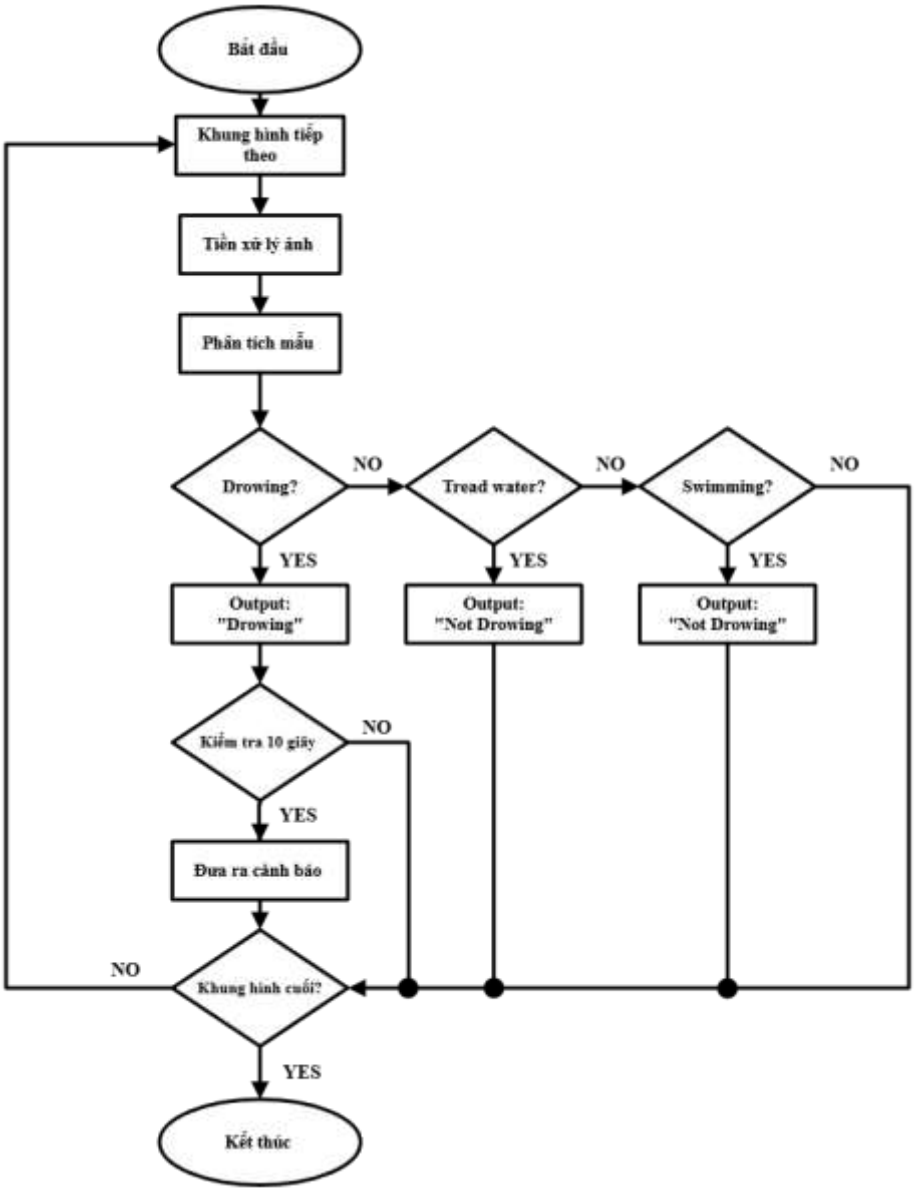


Hình 3.4 Lưu đồ thuật toán của hệ thống chiếu sáng

3.4 Hệ thống camera giám sát AI

Quy trình bắt đầu bằng việc chuyển qua các khung hình liên tiếp trong video, tiếp theo là quá trình xử lý ảnh nhằm phân tích các đặc điểm và mẫu trong hình ảnh. Căn cứ vào kết quả phân tích, hệ thống sẽ đưa ra kết luận về tình trạng của người trong hình ảnh. Nếu phát hiện dấu hiệu ngạt thở (Drowning), hệ thống sẽ đưa ra kết luận "Drowning". Nếu không, hệ thống tiếp tục kiểm tra xem người đó có đang duy trì trạng thái "tread water" (đạp nước) hay không; nếu có, kết luận sẽ là "Not Drowning". Trong trường hợp người đó không có dấu hiệu đạp nước, hệ thống sẽ tiếp tục xác định xem người đó có đang bơi hay không; nếu đúng, kết quả sẽ là "Not Drowning".

Nếu không có dấu hiệu nào rõ rệt, hệ thống sẽ kiểm tra thêm trong vòng 10 giây để xác nhận tình trạng. Nếu tình trạng vẫn không rõ ràng, một cảnh báo sẽ được đưa ra, nhằm thông báo về khả năng ngạt thở. Cuối cùng, hệ thống sẽ kiểm tra xem có phải đã đến khung hình cuối cùng của đoạn video hay không, và kết thúc quy trình.



Hình 3.5 Lưu đồ thuật toán của hệ thống camera giám sát AI

CHƯƠNG 4. TRIỂN KHAI VÀ MÔ PHỎNG HỆ THỐNG

4.1 Phân kênh đầu ra, đầu vào

Bảng 4.1 Bảng phân kênh đầu vào, đầu ra của hệ thống

KÊNH ĐẦU VÀO			
STT	**Kí hiệu**	**Địa chỉ**	**Mô tả**
1	NO_FL1	I0.0	Tiếp điểm thường mở cảm biến mực nước FL1
2	NC_FL1	I0.1	Tiếp điểm thường đóng cảm biến mực nước FL1
3	FL2	I0.2	Tiếp điểm thường mở cảm biến mực nước FL2
4	FL3	I0.3	Tiếp điểm thường mở cảm biến mực nước FL3
5	OCR1	I0.4	Relay nhiệt bơm lọc 1
6	OCR2	I0.5	Relay nhiệt bơm lọc 2
7	OCR3	I0.6	Relay nhiệt bơm lọc 3
8	OCR4	I0.7	Relay nhiệt bơm lọc 4
9	OCR5	I1.0	Relay nhiệt bơm lọc 5
10	OCR6	I1.1	Relay nhiệt bơm lọc 6
11	OCR7	I1.2	Relay nhiệt bơm chìm 1
12	OCR8	I1.3	Relay nhiệt bơm chìm 2
13	OCR_CP1	I1.4	Relay nhiệt bơm tuần hoàn 1
14	OCR_CP2	I1.5	Relay nhiệt bơm tuần hoàn 2
15	FB_ON_SP7	I2.0	Phản hồi động cơ bơm chìm 1
16	HEATPUMP_RELAY_1	I2.1	Tín hiệu báo chạy từ máy gia nhiệt 1
17	HEATPUMP_RELAY_2	I2.2	Tín hiêu báo chạy từ máy gia nhiệt 2
18	TRIP_DOSSI_PUMP1	I2.3	Tín hiệu báo lỗi bơm định lượng 1

19	TRIP_DOSSI_PUMP2	I2.4	Tín hiệu báo lỗi bơm định lượng 2
20	Clo		Cảm biến đo Clo dư
21	Temp		Cảm biến đo nhiệt độ
22	pH		Cảm biến đo pH
KÊNH ĐẦU RA			
STT	**Kí hiệu**	**Địa chỉ**	**Mô tả**
1	FILTER_PUMP_1	Q0.0	Bơm lọc 1
2	FILTER_PUMP_2	Q0.1	Bơm lọc 2
3	FILTER_PUMP_3	Q0.2	Bơm lọc 3
4	FILTER_PUMP_4	Q0.3	Bơm lọc 4
5	FILTER_PUMP_5	Q0.4	Bơm lọc 5
6	FILTER_PUMP_6	Q0.5	Bơm lọc 6
7	SUMBERSIBLE_PUMP_WP7	Q0.6	Bơm chìm 1
8	SUMBERSIBLE_PUMP_WP8	Q0.7	Bơm chìm 2
9	CCT25	Q1.0	Cụm đèn CCT25
10	CCT25A	Q1.1	Cụm đèn CCT25A
11	CCT25B	Q2.0	Cụm đèn CCT25B
12	CCT25C	Q2.1	Cụm đèn CCT25C
13	VALVE	Q2.2	Van điện từ
14	SALT CLORINATOR	Q2.3	Máy điện phân muối
15	CIRCULATION_PUMP_1	Q2.4	Bơm tuần hoàn 1
16	CIRCULATION_PUMP_2	Q2.5	Bơm tuần hoàn 2
17	FLOODED_TECHNICAL_ROOM	Q2.6	Cảnh báo ngập phòng kỹ thuật
18	PAUSE_DOSSI_PUMP1	Q2.7	Dừng bơm định lượng 1
19	PAUSE_DOSSI_PUMP2	Q3.0	Dừng bơm định lượng 2

20	DROWING_1	Q3.1	Cảnh báo đuối nước khu vực 1
21	DROWING_2	Q3.2	Cảnh báo đuối nước khu vực 2
22	DROWING_3	Q3.3	Cảnh báo đuối nước khu vực 3
23	DROWING_4	Q3.4	Cảnh báo đuối nước khu vực 4

4.2 Thiết lập kết nối giữa Raspberry Pi với PLC

Để thiết lập kết nối giao tiếp giữa Raspberry Pi với PLC, ta sẽ sử dụng thư viện "snap7" để kết nối thông qua giao thức TCP/IP

```
# -------------------- PLC CONFIG --------------------
PLC_IP = '192.168.1.24'
RACK = 0
SLOT = 1
DB_NUMBER = 1

tagArr = {}
# -------------------- PLC DATA READER ------------------
```

Hình 4.1 Thiết lập địa chỉ IP kết nối với PLC

Sau khi thiết lập kết nối, ta sẽ tạo các tag tương ứng với các tag bên PLC để có thể đọc/ghi dữ liệu.

```python
# ------------------- PLC DATA READER -------------------
3 usages
def read_plc_data():
    client = snap7.client.Client()
    try:
        client.connect(PLC_IP, RACK, SLOT)
        data = client.db_read(DB_NUMBER, 0, 35)
        return {
            'pH': get_real(data, 6),
            'Clo': get_real(data, 10),
            'TEMP': get_real(data, 14),
            'WATER_LEVEL_BLANCED_TANK': get_int(data, 20),
            'WATER_LEVEL_SUMPIT_TANK' : get_int(data, 22),

            'AUTO': get_bool(data, 0, 0),
            'MAN': get_bool(data, 0, 1),
            'START': get_bool(data, 0, 2),
            'STOP': get_bool(data, 0, 3),
            'RUN_AUTO': get_bool(data, 0, 4),
            'RUN_MAN': get_bool(data, 0, 5),

            'FILTER_PUMP_1': get_bool(data, 0, 6),
            'FILTER_PUMP_2': get_bool(data, 0, 7),
            'FILTER_PUMP_3': get_bool(data, 1, 0),
            'FILTER_PUMP_4': get_bool(data, 1, 1),
            'FILTER_PUMP_5': get_bool(data, 1, 2),
            'FILTER_PUMP_6': get_bool(data, 1, 3),
```

Hình 4.2 Kết nối các tag trên Raspberry tương ứng với tag bên PLC

Chi tiết đoạn code kết nối Raspberry với PLC được trình bày ở PHỤ LỤC 1.

4.3 Mô phỏng hệ thống xử lý nước

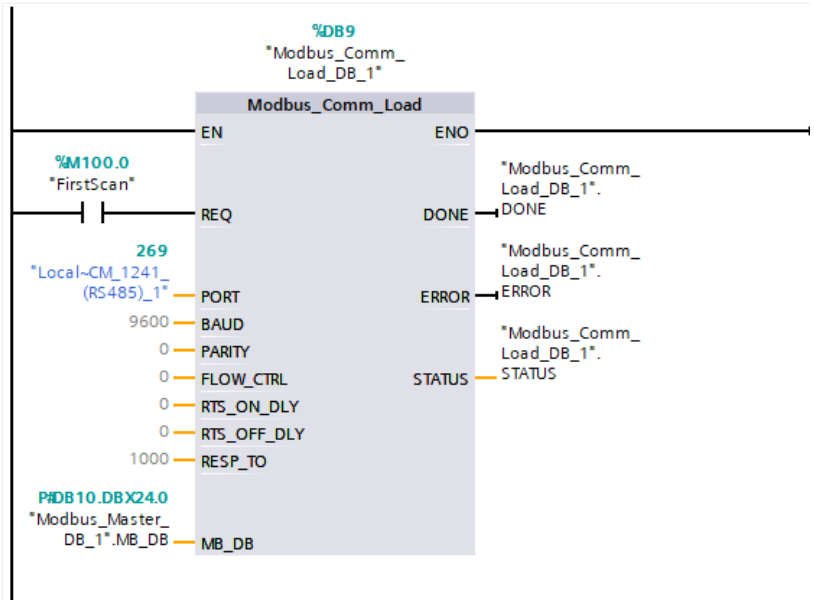
4.3.1 Triển khai trên TIA PORTAL V17

Để triển khai điều khiển và giám sát hệ thống xử lý nước ta cần thực hiện trên Tia Portal như sau:

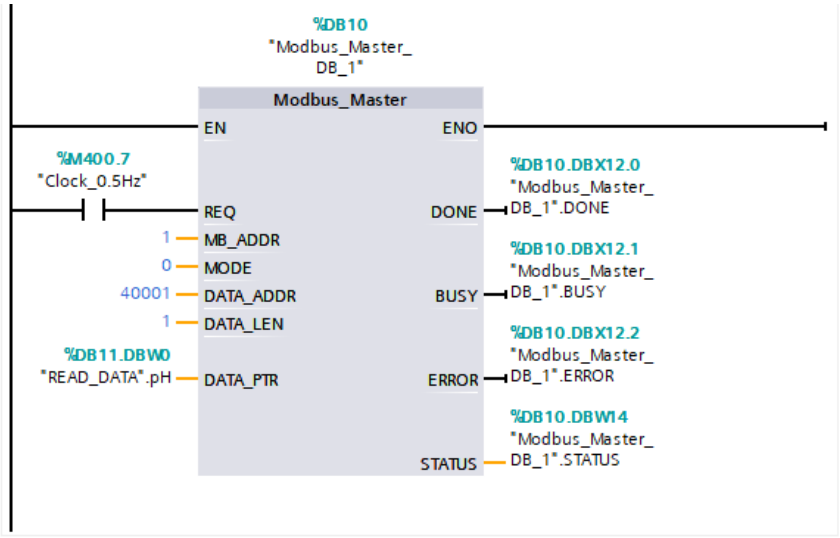
- Thiết kế khối hàm FB/FC để điều khiển hệ thống lọc nước, bơm hóa chất, van điện từ dựa trên tín hiệu từ cảm biến.
- Lập trình các chế độ vận hành: Manual – Auto, người dùng có thể chọn chế độ từ HMI và giao diện Web.
- Ứng dụng Modbus RTU để nhận dữ liệu từ cảm biến pH, Clo và báo động khi có tín hiệu vượt ngưỡng an toàn.

Đối với PLC Siemens dòng S7-1200, việc tích hợp giao thức Modbus RTU thông qua module truyền thông mở rộng như CM 1241 (RS485) cho phép PLC hoạt động như một Modbus Master, chủ động gửi yêu cầu đọc/ghi dữ liệu đến các thiết bị Modbus Slave như cảm biến pH, cảm biến nhiệt độ, hay bộ đo nồng độ clo. S7-1200 sử dụng các

khối chức năng tiêu chuẩn như Modbus_Comm_Load để cấu hình thông số truyền thông, và Modbus_Master để thực hiện giao tiếp dữ liệu.



Hình 4.3 Khối Modbus_Comm_Load



Hình 4.4 Khối Modbus_Master

Bảng 4.2 Các tham số khối Modbus_Comm_Load

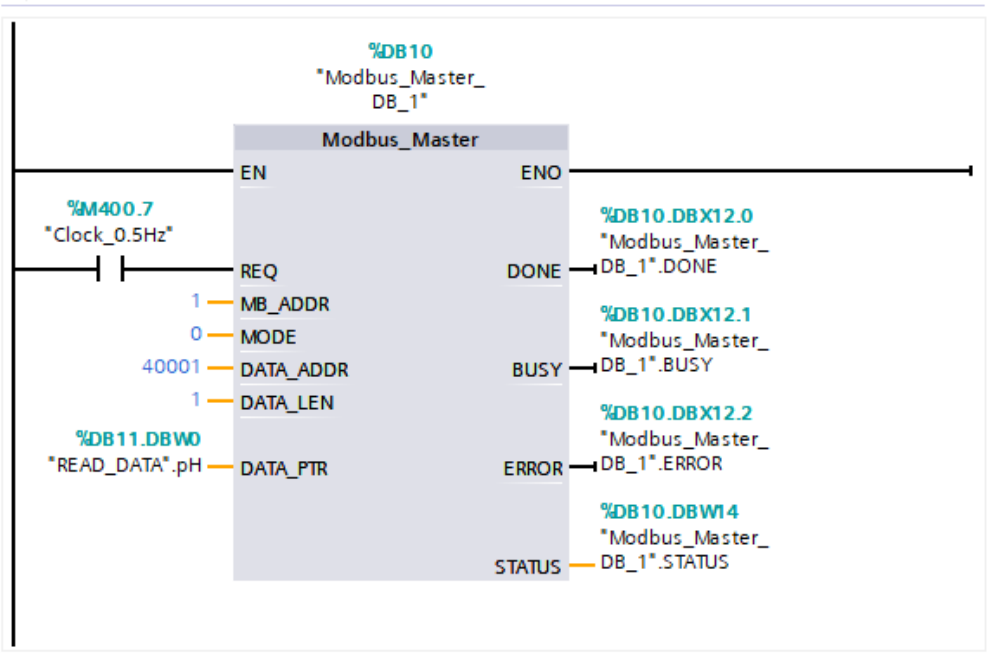
Thành phần	Ý nghĩa
PORT	Chỉ định cổng truyền thông RS485 cho module CM 1241. (Ví dụ: 269 tương ứng với Local_CM_1241) .
BAUD	Tốc độ truyền dữ liệu (Baudrate) – 9600 bps là tốc độ phổ biến cho Modbus RTU.
PARITY	Kiểm tra bit Parity, giúp kiểm tra lỗi trong quá trình truyền (0 nếu không sử dụng).

FLOW_CTRL	Kiểm soát luồng (Flow control), dùng để điều khiển truyền dữ liệu đúng thứ tự, đặt 0 nếu không dùng.
RTS_ON_DLY	Thời gian trễ khi bật tín hiệu RTS, để tạo ra sự đồng bộ với thiết bị ngoại vi.
RTS_OFF_DLY	Thời gian trễ khi tắt tín hiệu RTS.
DONE, ERROR, STATUS	Biến phản hồi trạng thái thực hiện của khối.

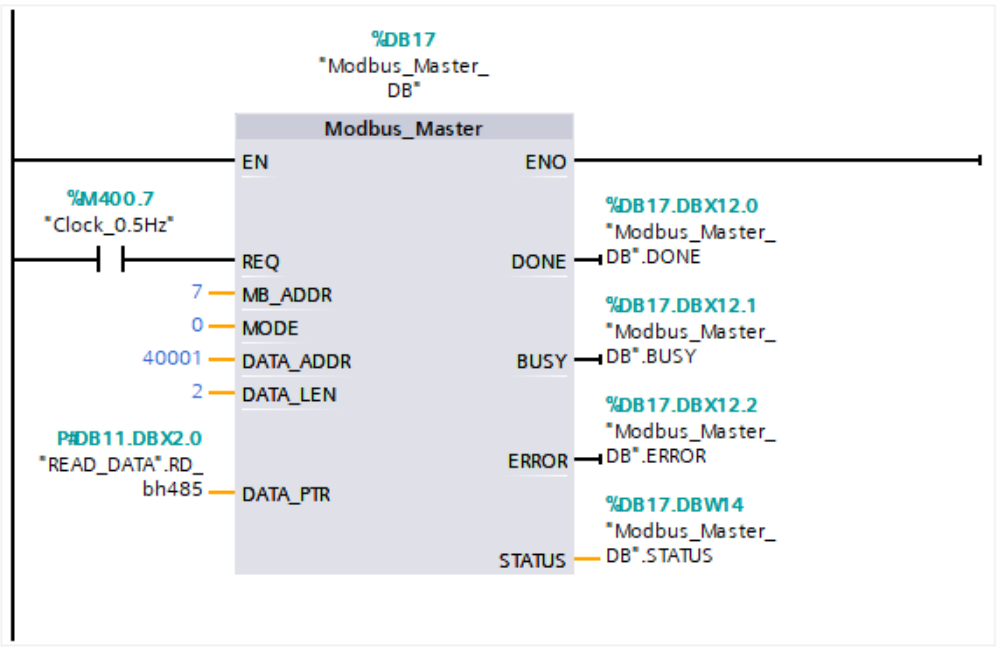
Bảng 4.3 Các tham số khối Modbus_Comm_Load

Thành phần	**Ý nghĩa**
MB_ADDR	Địa chỉ Modbus của cảm biến.
MODE	Chế độ đọc/ghi, 0 có nghĩa là đọc các thanh ghi giữ (Holding Register).
DATA_ADDR	Địa chỉ thanh ghi trong Modbus mà bạn muốn đọc (thực tế là địa chỉ bắt đầu).
DATA_LEN	Số lượng thanh ghi cần đọc (1 thanh ghi = 2 byte).
DATA_PTR	Con trỏ đến vị trí trong bộ nhớ PLC nơi dữ liệu sẽ được lưu trữ (DBW).
DONE	Trạng thái chỉ ra rằng khối chức năng đã hoàn thành việc đọc hoặc ghi dữ liệu.
BUSY	Trạng thái chỉ ra rằng khối chức năng đang hoạt động, chưa hoàn thành.
ERROR	Trạng thái lỗi nếu có sự cố trong quá trình truyền dữ liệu.
STATUS	Biến trạng thái cho biết tình trạng hoạt động của khối chức năng Modbus.

Dựa vào tài liệu [10], ta cấu hình giao tiếp Modbus RTU cho cảm biến trong hình 4.5 và hình 4.6

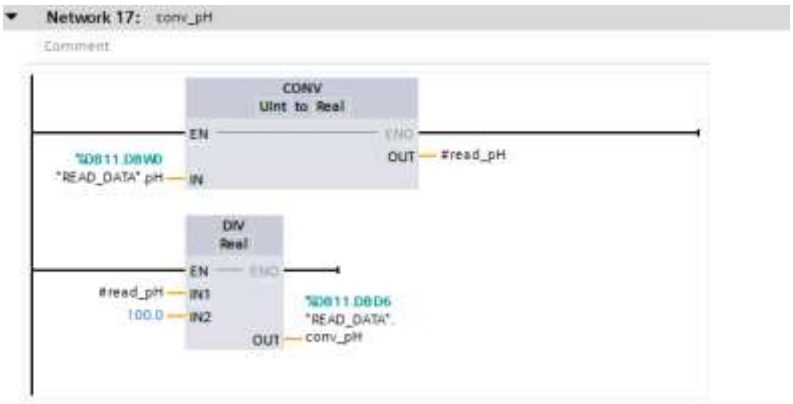


Hình 4.5 Khối Modbus_Master giao tiếp với cảm biến pH

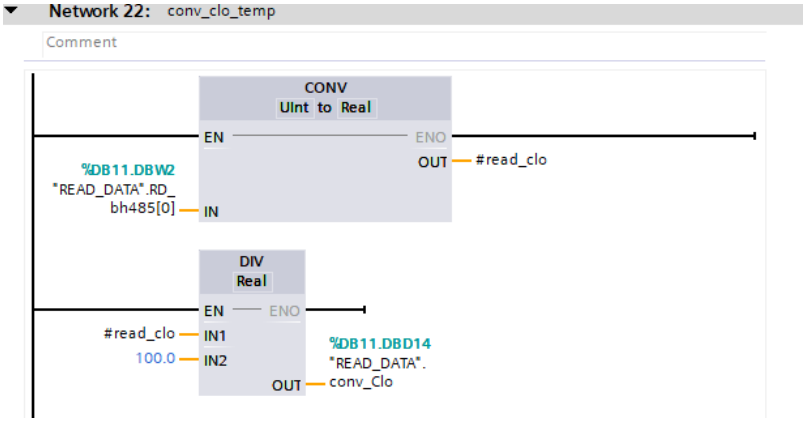


Hình 4.6 Khối Modbus_Master giao tiếp với cảm biến đo Clo dư và nhiệt độ

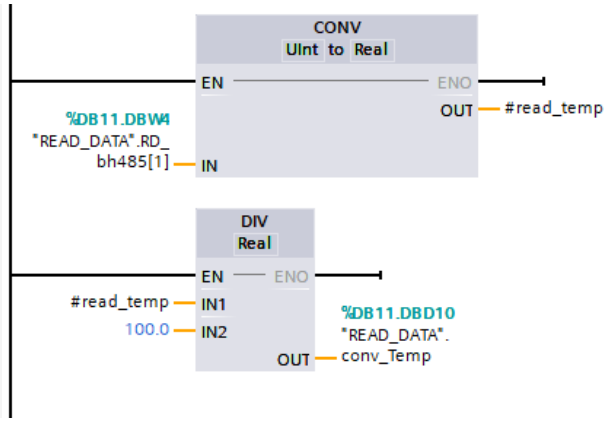
Sau khi thiết lập kết nối đọc được dữ liệu từ cảm biến, ta thực hiện chuyển đổi kiểu dữ liệu để hiển thị lên màn hình HMI và giao diện Web.



Hình 4.7 Chuyển đổi giá trị đo pH



Hình 4.8 Chuyển đổi giá trị đo Clo



Hình 4.9 Chuyển đổi giá trị đo nhiệt độ

Dựa vào lưu đồ thuật toán ta lập trình PLC điều khiển hệ thống xử lý nước theo chế độ bằng tay, chế độ tự động và điều khiển từ xa. Chương trình PLC được trình bày ở PHỤ LỤC 2.

4.3.2 Triển khai trên Raspberry Pi

Tương tự như trên Tia Portal, ta cũng thiết lập các chế độ bằng tay và chế độ tự động cùng với đó là lập trình gửi cảnh báo cho người quản lý thông qua email khi có sự cố xảy ra để kịp thời xử lý và lưu trữ lịch sử hoạt động của hệ thống.

Để có thể gửi cảnh báo đến người quản lý ta cần lập trình các bước như sau:

- Đầu tiên, thiết lập địa chỉ email nhận cảnh báo.

```
122
123     # ---------- EMAIL CONFIG -------------
124     EMAIL_ADDRESS = "kenkanekikun00@gmail.com"  # Thay bằng email của bạn
125     EMAIL_PASSWORD = "fetymkzmgtmjgfaa"  # Thay bằng app password Gmail của bạn
126     EMAIL_RECEIVERS = ["htphong11022002@gmail.com"]  # Thay bằng email người nhận
127
```

Hình 4.10 Thiết lập địa chỉ email nhận cảnh báo

- Thứ hai, lập trình đọc dữ liệu từ PLC kiểm tra trạng thái của hệ thống.

```
1 usage
def check_trip_and_alert(plc_data):
    now = time.time()
    ALERT_INTERVAL = 60  # Thời gian trễ giữa các lần gửi email

    # Kiểm tra trip bơm
    for pump, trip in plc_data.items():
        if trip and pump.startswith('TRIP'):
            last_time = last_alert_times.get(pump, 0)
            if now - last_time > ALERT_INTERVAL:  # Gửi email nếu chưa gửi trong ALERT_INTERVAL
                # Gửi email cảnh báo trip bơm
                subject = f"Cảnh báo TRIP bơm: {pump}"
                body = f"Phát hiện trip {pump} lúc {datetime.now()}"
                send_alert_email(subject, body)
                last_alert_times[pump] = now  # Cập nhật thời gian gửi email
```

Hình 4.11 Kiểm tra trạng thái hệ thống

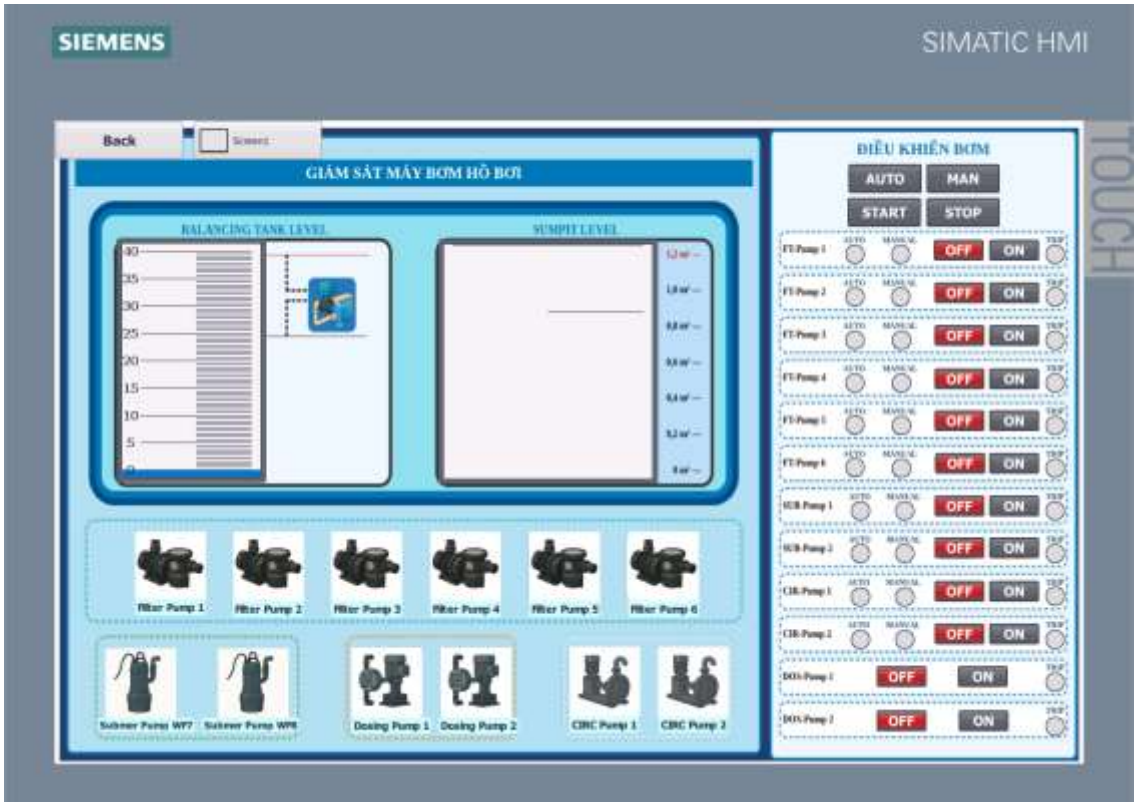
- Thứ ba, khi xuất hiện lỗi từ thiết bị, hệ thống sẽ tự động gửi email cảnh báo lỗi từ thiết bị đó. Sau đó, lỗi sẽ được tự động lưu vào trong database nơi lưu trữ lịch sử hoạt động của hệ thống.

```
226
227     # ------------------- MySQL CONFIG -------------------
228     db = mysql.connector.connect(
229         host="localhost",
230         user="root",
231         password="11022002Tt",
232         database="plc_data"
233     )
234     cursor = db.cursor()
235
        1 usage
236     def write_to_sql(plc_data):
237         query = """
238         INSERT INTO plc_water_quality_data (timestamp, temperature, ph, chlorine)
239         VALUES (%s, %s, %s, %s)
240         """
241         values = (
242             datetime.now(),
243             plc_data['TEMP'],
244             plc_data['pH'],
245             plc_data['Clo']
246         )
247         cursor.execute(query, values)
248         db.commit()
249
250     lastPumpStates = {}
251     lastMode = None
```

Hình 4.12 Ghi lịch sử hoạt động vào database

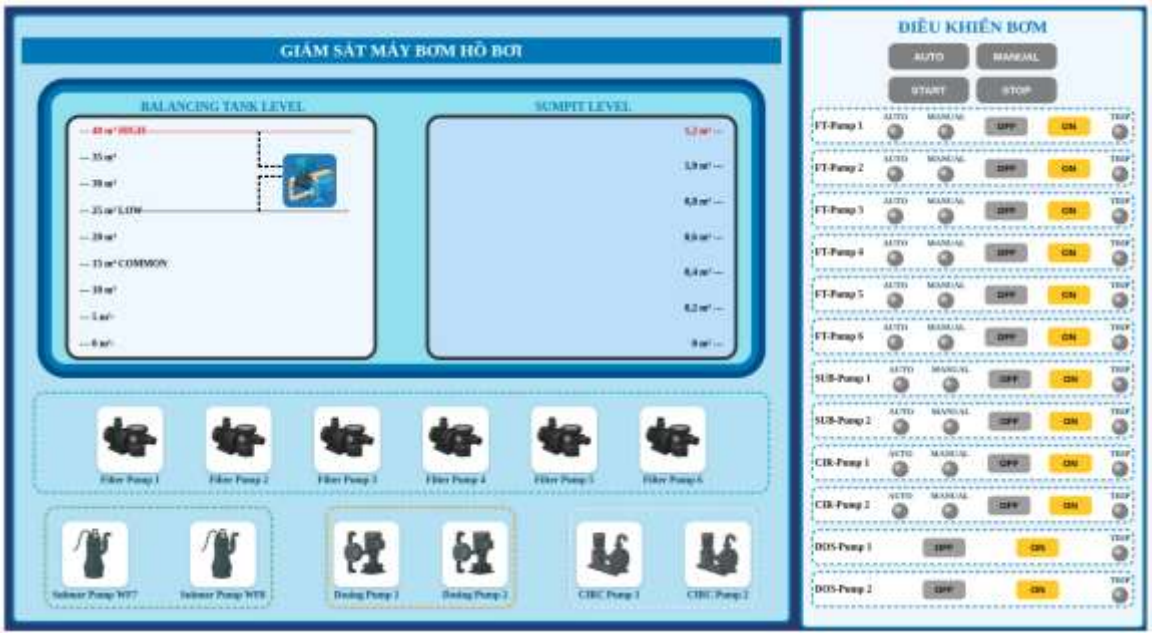
Ngoài ra, chi tiết về thời gian bật tắt bơm, giá trị Clo,... cũng cần được lưu trữ để thuận tiện cho công tác quản lý, chi tiết nằm ở trong PHỤ LỤC 1.

Hình 4.13 Giao diện HMI hệ thống giám sát chất lượng nước



Hình 4.14 Giao diện HMI hệ thống giám sát máy bơm

Hình 4.15 Giao diện Web hệ thống giám sát chất lượng nước

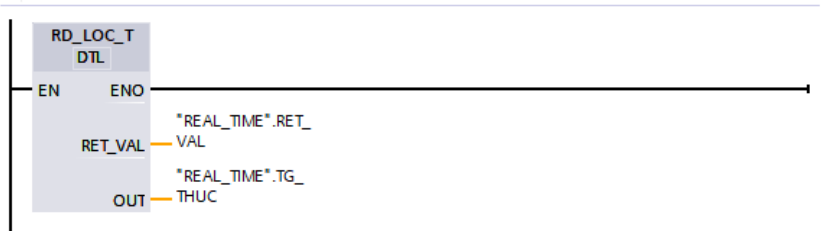


Hình 4.16 Giao diện Web hệ thống giám sát máy bơm

4.4 Mô phỏng hệ thống chiếu sáng

4.4.1 Triển khai trên TIA PORTAL V17

Dựa vào lưu đồ thuật toán, hệ thống chiếu sáng được vận hành dựa trên thời gian thực. Do đó, ta sẽ dùng khối *Read Local Time* để đọc thời gian hiện tại từ đồng hồ hệ thống của PLC, qua đó lập trình điều khiển hệ thống chiếu sáng. Cụ thể đoạn code lập trình được trình bày ở PHỤ LỤC 2.



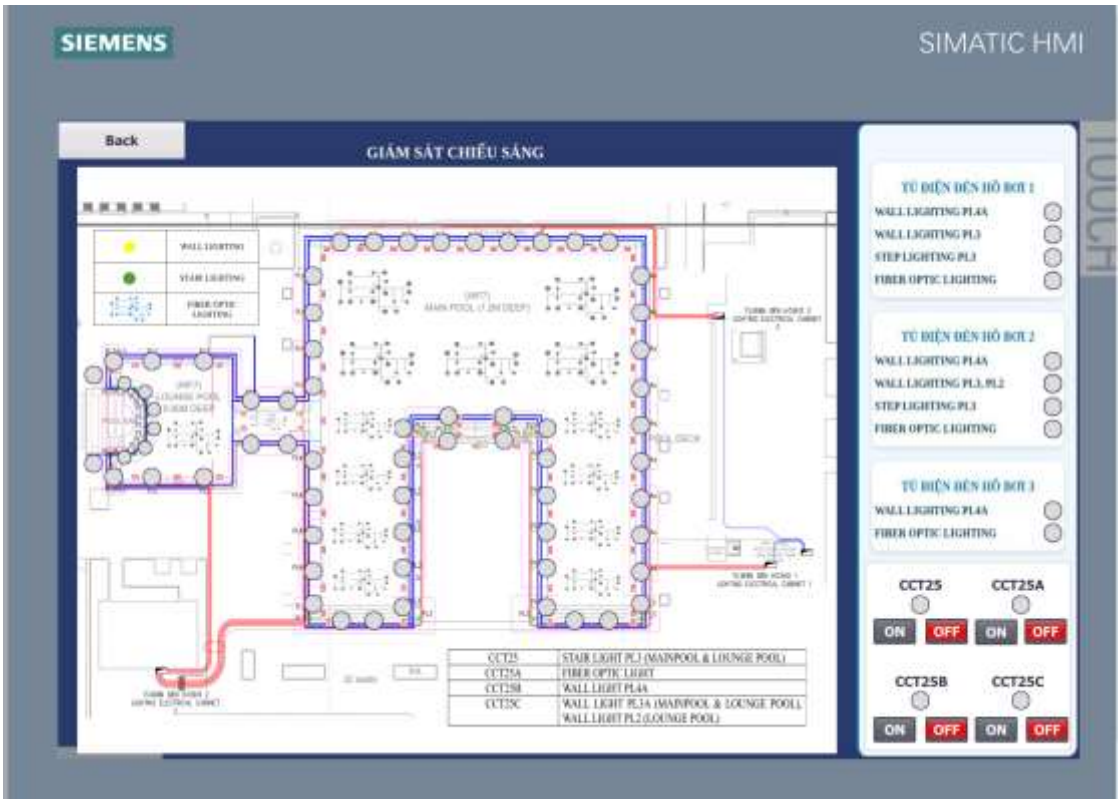
Hình 4.17 Khối lệnh Read Local Time

Bảng 4.4 Bảng thông số khối hàm Read Local Time

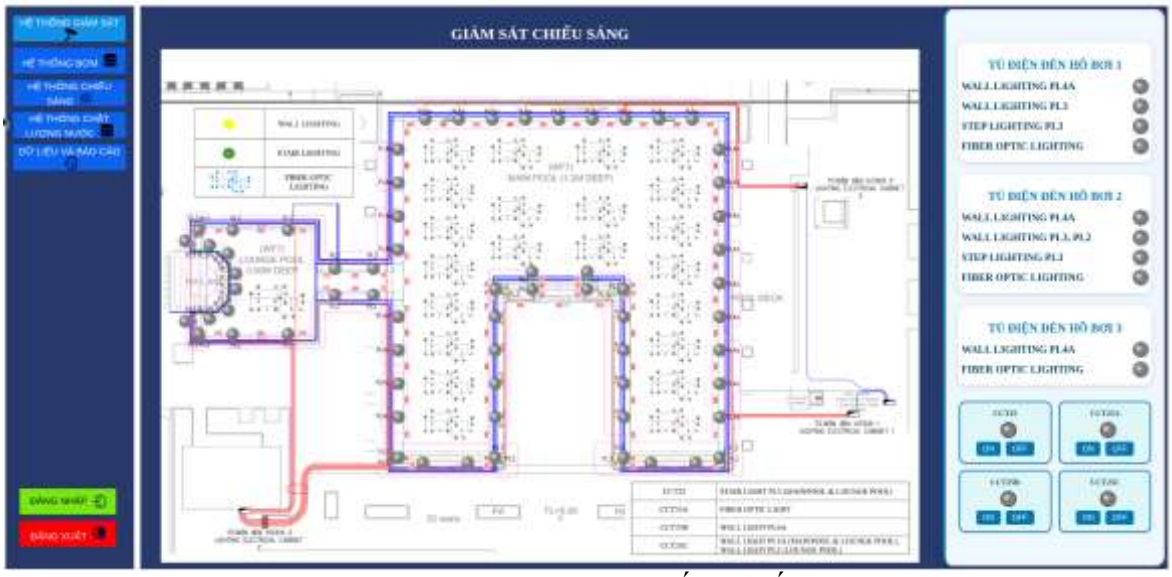
Thành phần	**Ý nghĩa / Chức năng**
EN (Enable)	Tín hiệu điều kiện kích hoạt khối (true thì khối mới thực thi).
ENO (Enable Output)	Trạng thái đầu ra cho biết khối có thực thi hay không.
RET_VAL	Giá trị trả về, thường là mã lỗi (0 nếu thành công).
OUT	Biến đầu ra logic báo hiệu khối đã thực hiện xong.
DTL (Date and Time Large)	Cấu trúc kiểu dữ liệu để lưu thời gian, kiểu DTL (bao gồm năm, tháng, ngày, giờ, phút, giây, ms).
"REAL_TIME".RET_VAL	Biến lưu mã lỗi trả về.
"REAL_TIME".TG_THUC	Biến lưu thời gian thực lấy được từ PLC (kiểu DTL).

4.4.2 Triển khai trên Raspberry Pi

Để xây dựng hệ thống chiếu sáng tự động trong hồ bơi thông minh và lưu trữ lịch sử hoạt động, ta thực hiện các bước tương tự như đã làm với hệ thống cảnh báo. Cụ thể, xây dựng một giao diện điều khiển chiếu sáng, tích hợp việc điều khiển đèn từ xa qua Web, đồng thời lưu trữ lịch sử hoạt động chiếu sáng vào cơ sở dữ liệu để có thể theo dõi.



Hình 4.18 Giao diện màn hình HMI hệ thống chiếu sáng

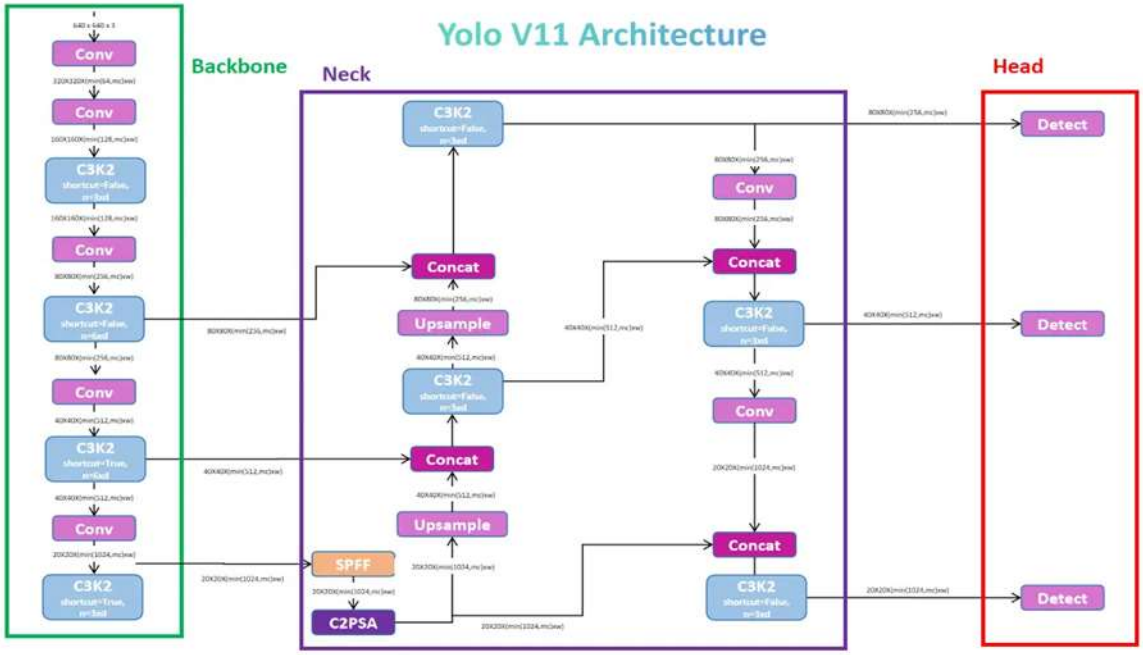


Hình 4.19 Giao diện Web hệ thống chiếu sáng

4.5 Mô phỏng hệ thống camera giám sát AI

4.5.1 Huấn luyện mô hình

Nhằm phát triển một hệ thống giám sát và cảnh báo an toàn tại bể bơi, thuật toán YOLO11 được ứng dụng như một công cụ để thực hiện nhận dạng, định vị đối tượng và phát hiện nguy cơ tiềm ẩn. Hệ thống được xây dựng dựa trên việc thu thập và sử dụng các tập dữ liệu hình ảnh đa dạng về hoạt động dưới nước nhằm đảm bảo độ tin cậy, tính chính xác và hiệu quả trong nhận diện.



Hình 4.20 Cấu trúc mô hình YOLO11

Dữ liệu huấn luyện được thu thập từ các bể bơi thực tế tại Đà Nẵng (bể bơi công cộng và tư nhân); các thư viện uy tín như Roboflow, ... Trong đó, tập dữ liệu gồm hơn 18 000 hình ảnh, được gắn nhãn thủ công với 4 lớp: "Swimming" (bơi lội), "Drowning" (đuối nước), "Tread_water" (đứng nước) và "Normal" (bình thường). Dữ liệu được chuẩn hóa (640x640 pixel) và tăng cường (data augmentation) bằng các kỹ thuật như xoay, thay đổi độ sáng để cải thiện khả năng tổng quát hóa của mô hình. Tập dữ liệu được chia như sau:

- Tập huấn luyện (Train Set): Gồm 15331 hình ảnh, dùng để huấn luyện mô hình.
- Tập kiểm định (Validation Set): Gồm 2577 hình ảnh, dùng để đánh giá hiệu suất mô hình trong quá trình huấn luyện.
- Tập kiểm tra (Test Set): Gồm 545 hình ảnh, dùng để đánh giá chính xác hiệu suất của mô hình sau huấn luyện

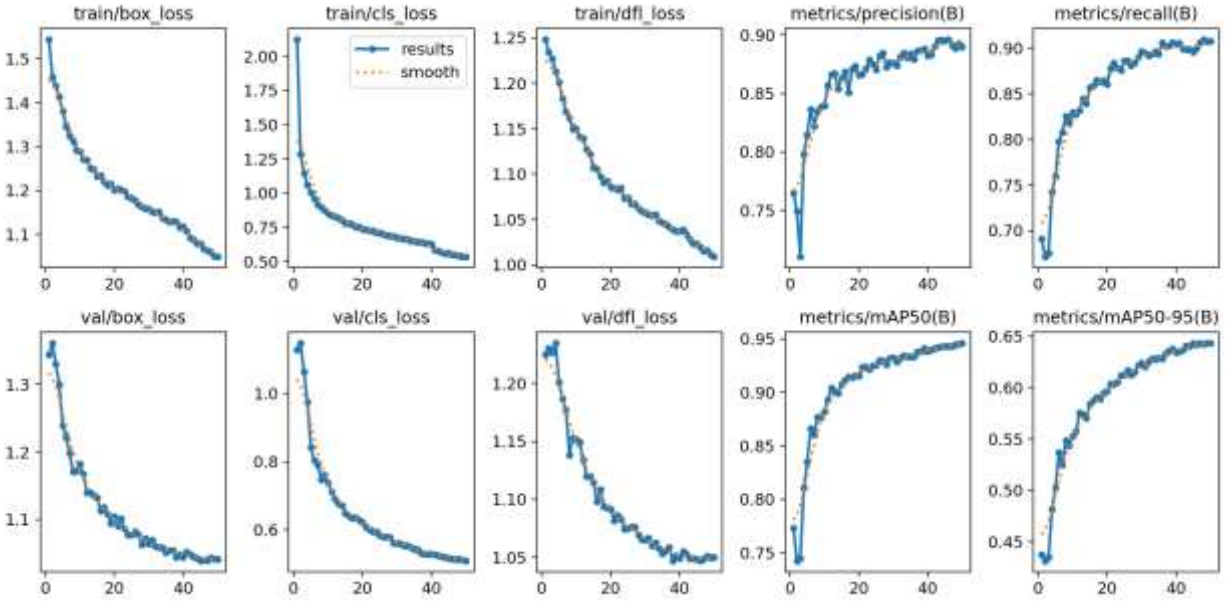
Việc đánh giá tính hiệu quả của mô hình được đề xuất sử dụng các số liệu đánh giá thường được sử dụng trong phân loại và phát hiện đối tượng, bao gồm *Precision, Recall* và *mAP* [9]. Trong đó Precision thể hiện mức độ chính xác trong các phát hiện của mô hình, Recall thể hiện cho khả năng mô hình không bỏ sót dữ liệu cần được phát hiện còn mAP50 là độ chính xác trung bình của mô hình khi ngưỡng phát hiện được thiết lập ở mốc 0, 5. Công thức cho ba tham số này như sau

$$Precision(Pr) = \frac{TP}{TP + FP} \tag{4.1}$$

$$Recall(Rc) = \frac{TP}{TP + FN} \tag{4.2}$$

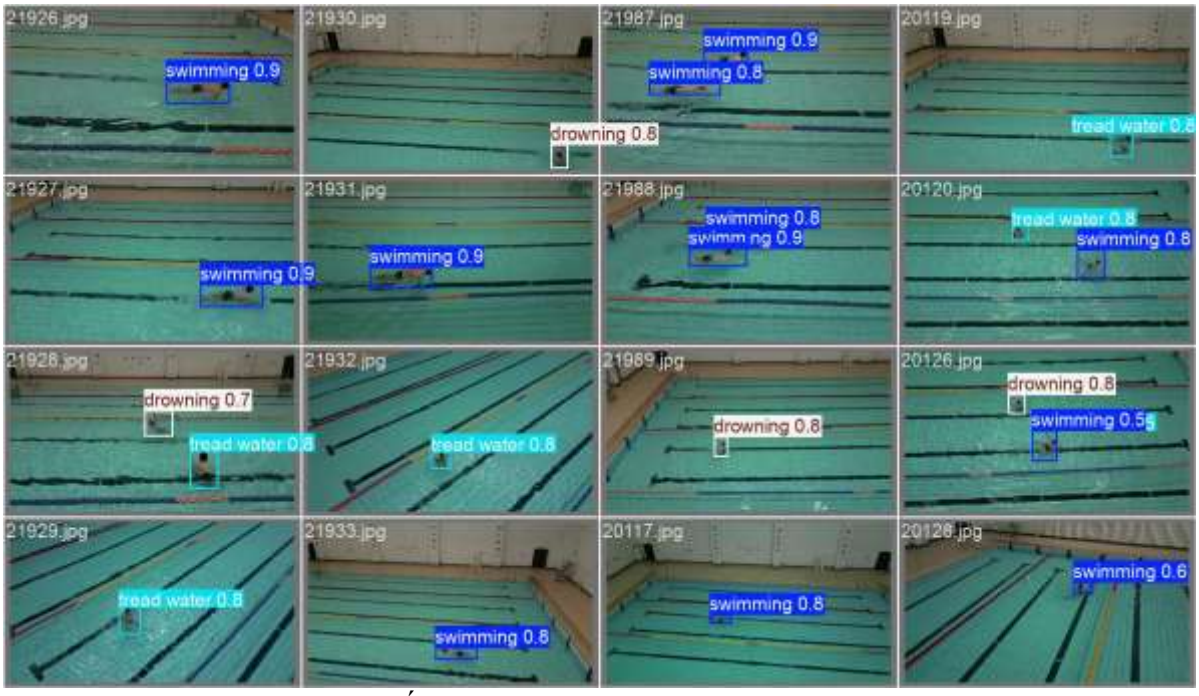
$$mAP50 = \frac{1}{N}\sum_{i=1}^{N} AP_{50,i} \tag{4.3}$$

Sau khi mô hình được huấn luyện và đánh giá trên tập train và valid với 50 vòng lặp trên tập dữ liệu. Kết quả cuối cùng thu được như hình 4.21



Hình 4.21 Đồ thị kết quả huấn luyện mô hình

Để kiểm tra độ chính xác của mô hình, nhóm đã tiến hành phân loại trên tập test.



Hình 4.22 Kết quả dự đoán của mô hình với tập validation

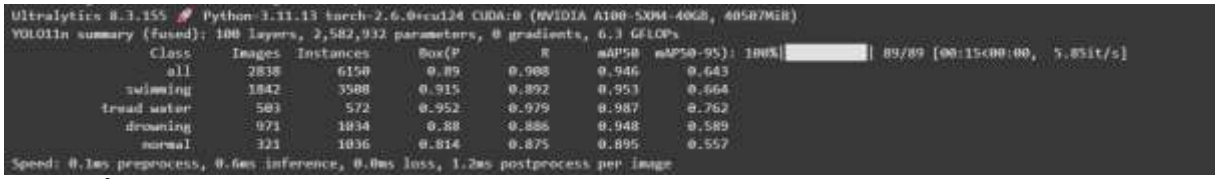
Dựa vào các đồ thị kết quả ở hình 4.21:

- **Train Loss**: Các đồ thị thể hiện rằng **train box loss, cls loss, và dfl loss** đều giảm liên tục qua các epochs, cho thấy mô hình đang học một cách hiệu quả,

không gặp phải hiện tượng overfitting. Điều này chứng tỏ mô hình không chỉ học tốt từ dữ liệu huấn luyện mà còn có khả năng tổng quát tốt.

- **Validation Loss**: Loss trên **tập validation** cũng giảm tương tự như trên tập huấn luyện, điều này chứng tỏ mô hình không chỉ học tốt trên tập huấn luyện mà còn generalize tốt với dữ liệu mới (validation set). Điều này cho thấy mô hình có khả năng dự đoán chính xác trên dữ liệu chưa thấy.
- **Precision và Recall**: Cả **precision** và **recall** đều đạt mức ổn định cao, lần lượt khoảng 0.9, trong suốt quá trình huấn luyện. **Precision** cao cho thấy mô hình phân loại chính xác đối tượng, còn **Recall** cao cho thấy mô hình phát hiện được gần như tất cả các đối tượng có trong ảnh.
- **mAP (mean Average Precision)**: Các chỉ số **mAP50** và **mAP50-95** đều đạt mức rất cao, lần lượt khoảng 0.9 và 0.64. Điều này cho thấy mô hình có khả năng phân loại chính xác ngay cả với các đối tượng có sự chồng lấn lớn (IoU ≥ 0.5), mặc dù hiệu suất với các ngưỡng IoU nhỏ hơn vẫn có thể được cải thiện thêm.

Kết hợp với các thông số tổng quát sau:



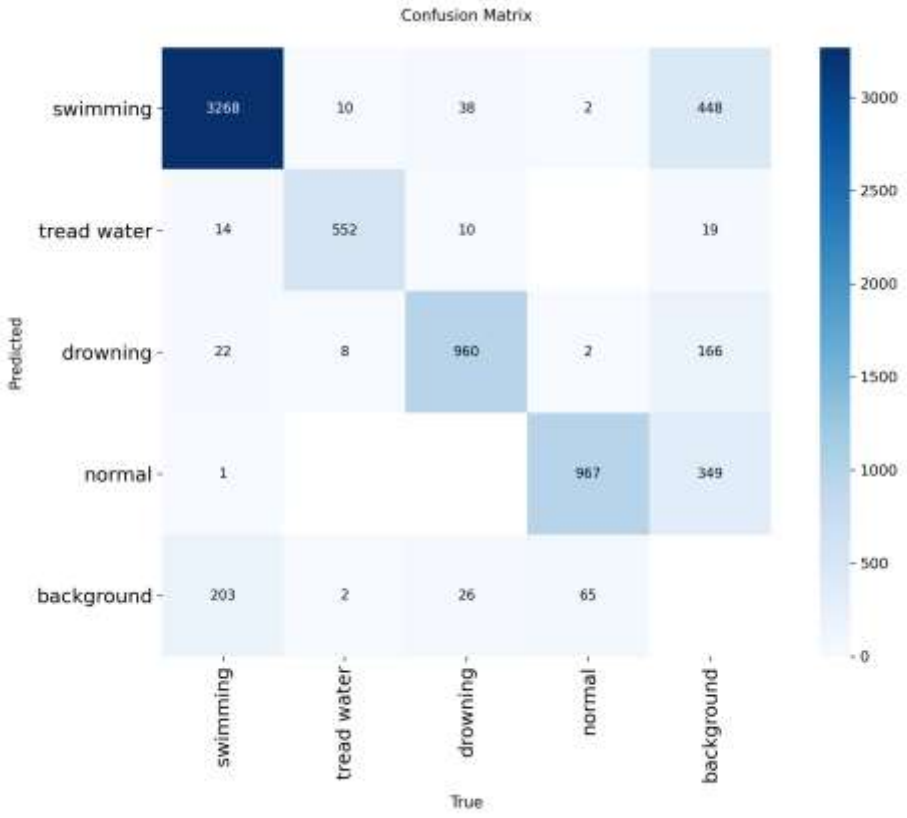
Ta thấy:

- **Precision**: Với **precision = 0.92**, mô hình có khả năng phân loại chính xác rất cao, tức là tỷ lệ các dự đoán đúng trên tổng số dự đoán là rất lớn.
- **Recall**: **Recall = 0.88** cho thấy mô hình phát hiện được hầu hết các đối tượng trong ảnh, mặc dù mức độ phát hiện có phần thấp hơn một chút so với precision, điều này có thể xuất phát từ việc một số đối tượng có thể bị bỏ sót, đặc biệt trong các tình huống phức tạp hoặc với đối tượng nhỏ.
- **mAP**: **mAP50 = 0.946** và **mAP50-95 = 0.643**. mAP50 cho thấy mô hình phân loại rất chính xác trên các đối tượng có sự chồng lấn lớn, trong khi mAP50-95 cho thấy hiệu suất hơi thấp với các đối tượng có sự chồng lấn nhỏ hơn, nhưng vẫn nằm trong phạm vi chấp nhận được.
- **Speed**: Mô hình hoạt động rất nhanh, với **0.1s tiền xử lý**, **0.6ms suy luận**, và **1.2ms hậu kỳ**. Điều này chứng tỏ mô hình đã được tối ưu hóa tốt và có thể hoạt động trong thời gian thực trên các thiết bị với cấu hình thấp, chẳng hạn như Raspberry Pi.

Kết luận tổng quan sau huấn luyện:

Mô hình YOLO11 của nhóm đã đạt được kết quả rất tốt sau 50 epochs huấn luyện. Các chỉ số **precision**, **recall**, và **mAP** đều rất cao, cho thấy mô hình có khả năng phân loại và phát hiện đối tượng rất chính xác. Mô hình cũng không gặp vấn đề **overfitting** và có thể tổng quát tốt trên dữ liệu mới. **Speed** của mô hình cũng rất nhanh, phù hợp cho các ứng dụng thời gian thực.

Tóm lại, với các chỉ số như **mAP = 0.946**, **precision = 0.92**, và **recall = 0.88**, mô hình của nhóm tự đánh giá có thể được áp dụng thành công vào các ứng dụng thực tế, và vẫn có thể được cải thiện thêm để đạt hiệu suất tối ưu hơn.



Hình 4.23 Ma trận nhầm lẫn trên tệp test của mô hình

Kết quả thu được ma trận nhầm lẫn ở hình 4.23 với các kết quả như sau:

- **Lớp "Swimming"**: Mô hình có độ chính xác cao với 3268 ảnh phân loại đúng. Tuy nhiên, có sự nhầm lẫn với lớp "background" (448 ảnh), điều này cho thấy mô hình cần cải thiện khả năng phân biệt giữa các hành vi và các yếu tố nền.
- **Lớp "Tread Water"**: Lớp này có kết quả khá tốt với 552 ảnh phân loại đúng. Tuy nhiên, vẫn có một số nhầm lẫn nhỏ với các lớp khác như "swimming" (14 ảnh) và "drowning" (10 ảnh). Mặc dù vậy, kết quả này là hợp lý và có thể chấp nhận được trong giai đoạn huấn luyện đầu tiên.

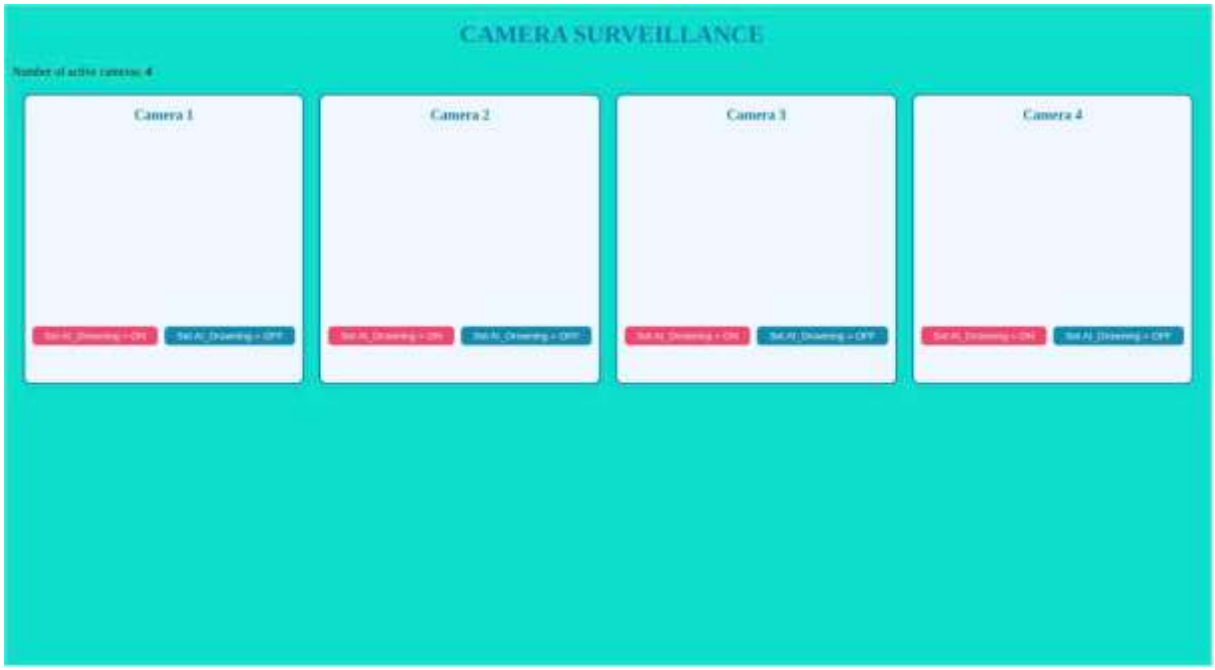
- **Lớp "Drowning"**: Độ chính xác khá cao với 960 ảnh phân loại đúng. Tuy nhiên, mô hình vẫn gặp phải một số nhầm lẫn với "swimming" (22 ảnh) và "background" (166 ảnh). Điều này cho thấy mô hình cần cải thiện để phân biệt tốt hơn giữa các tình huống nguy hiểm và các hành vi khác.
- **Lớp "Normal"**: Lớp này đạt độ chính xác cao với 967 ảnh phân loại đúng, nhưng lại bị nhầm lẫn với "background" (349 ảnh). Tôi nhận thấy rằng mô hình vẫn có khó khăn trong việc phân biệt giữa các hành vi bình thường và các yếu tố nền, điều này cần được cải thiện.

4.5.2 Triển khai mô hình trên Raspberry Pi

Sau khi huấn luyện ta tiến hành chạy mô hình trên Raspberry Pi, sau đó kết nối với Camera IP và đưa lên giao diện Web.

```
28
29  # -------------------- YOLO CONFIG --------------------
30  model = YOLO("models_cus/best_ncnn_model")
31  labels = model.names
32  input_size = (640, 640)
33
34  camera_sources = [
35      "rtsp://admin:11022002Tt@192.168.63.80:554/Streaming/Channels/101?rtsp_transport=udp",
36      "rtsp://admin:11022002Tt@192.168.63.80:554/Streaming/Channels/101?rtsp_transport=udp",
37      "rtsp://admin:11022002Tt@192.168.63.80:554/Streaming/Channels/102?rtsp_transport=udp",
38  ]
```

Hình 4.24 Triển khai model trên Raspberry Pi

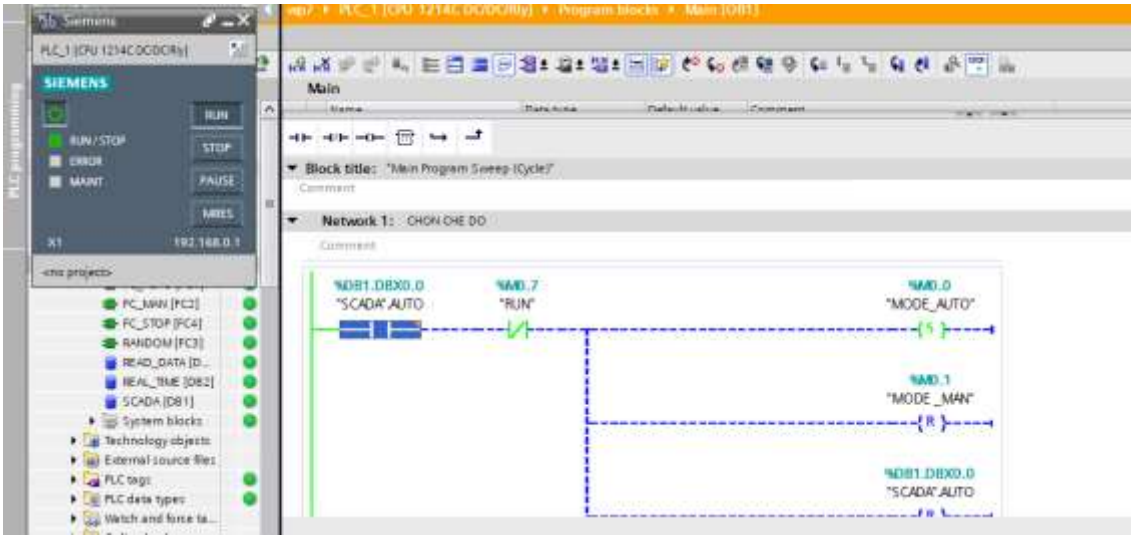


Hình 4.25 Giao diện camera giám sát

CHƯƠNG 5. KIỂM NGHIỆM ĐÁNH GIÁ HỆ THỐNG

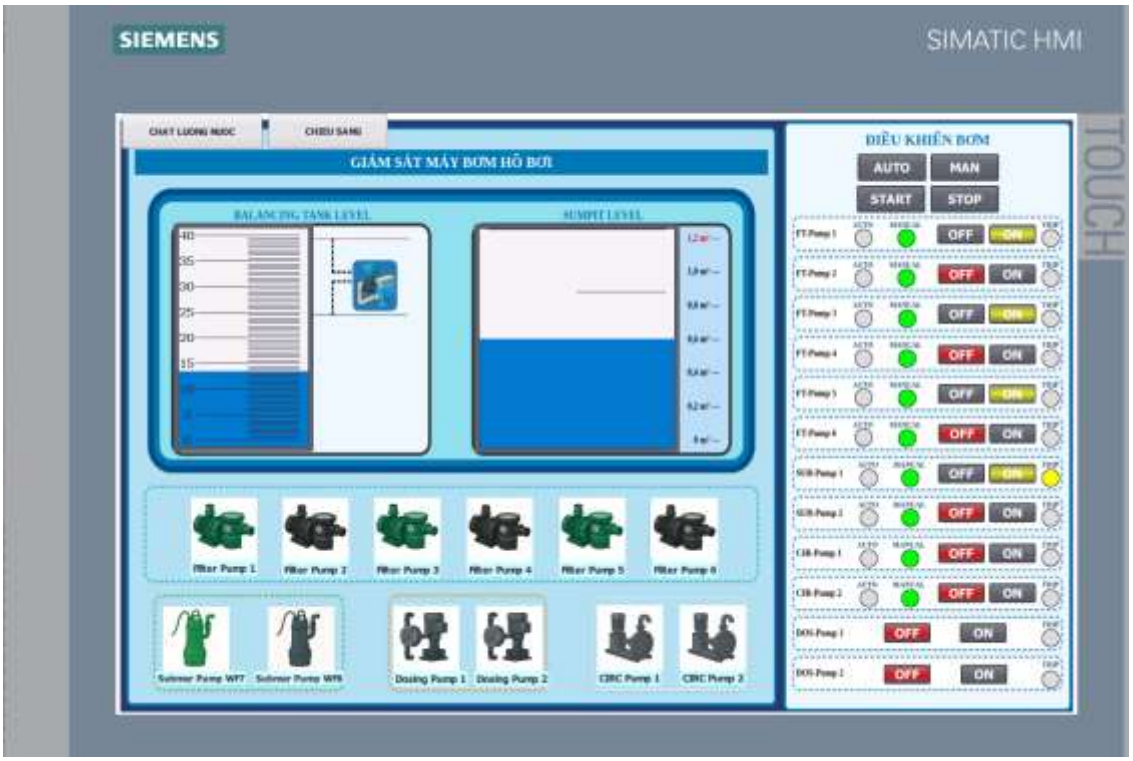
5.1 Mô tả quá trình mô phỏng

Chạy phần mềm TIA PORTAL V17



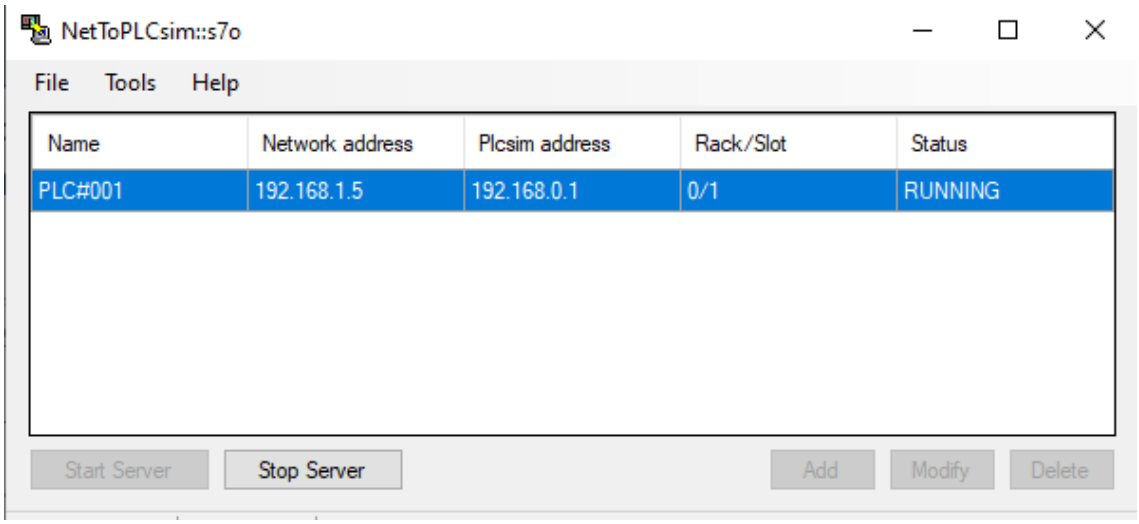
Hình 5.1 Chạy chương trình trên Tia Portal

Mô phỏng HMI trên WinCC



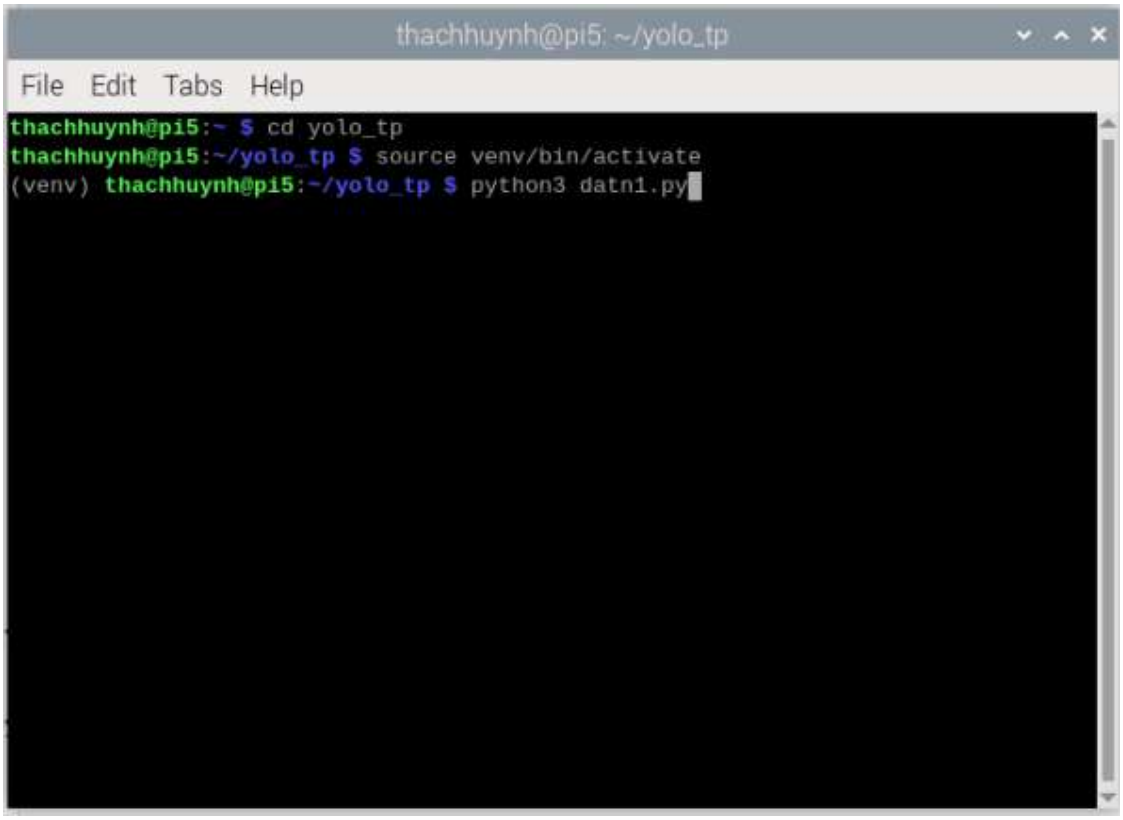
Hình 5.2 Mô phỏng HMI

Cài đặt kết nối NetToPLCsim



Hình 5.3 Kết nối NetToPLCsim

Khởi chạy trên Raspberry Pi



Hình 5.4 Khởi chạy Raspberry Pi

Chức năng từng câu lệnh:

- **'cd yolo_tp'** : chuyển đến thư mục chứa file dự án từ thư mục chính

- **'source venv/bin/activate'** : khởi chạy môi trường ảo trong thư mục vừa chuyển đến để có thể sử dụng tài nguyên cũng như các thư viện trong thư mục nhưng không ảnh hưởng đến các phần khác của thư mục
- **'python3 datn1.py'**: câu lệnh khởi chạy Server trên board Raspberry Pi

5.2 Kiểm nghiệm chức năng của hệ thống

Sau khi hoàn tất các bước thiết kế và lập trình hệ thống, nhóm tiến hành mô phỏng toàn bộ hệ thống hồ bơi thông minh bằng phần mềm TIA Portal V17 cho phần điều khiển PLC, và kết hợp môi trường Python trên Raspberry Pi để triển khai hệ thống giám sát Web và mô hình AI.

Quy trình kiểm thử gồm 3 giai đoạn chính:

- Giai đoạn 1: Kiểm nghiệm chức năng cơ bản của từng phần riêng lẻ (xử lý nước, chiếu sáng, giám sát camera).
- Giai đoạn 2: Kết nối các thành phần lại thành một hệ thống tích hợp và mô phỏng hoạt động thực tế.
- Giai đoạn 3: Đánh giá hiệu quả hoạt động, tốc độ phản hồi, độ ổn định và tính đồng bộ giữa các thành phần.

5.2.1 Hệ thống xử lý nước

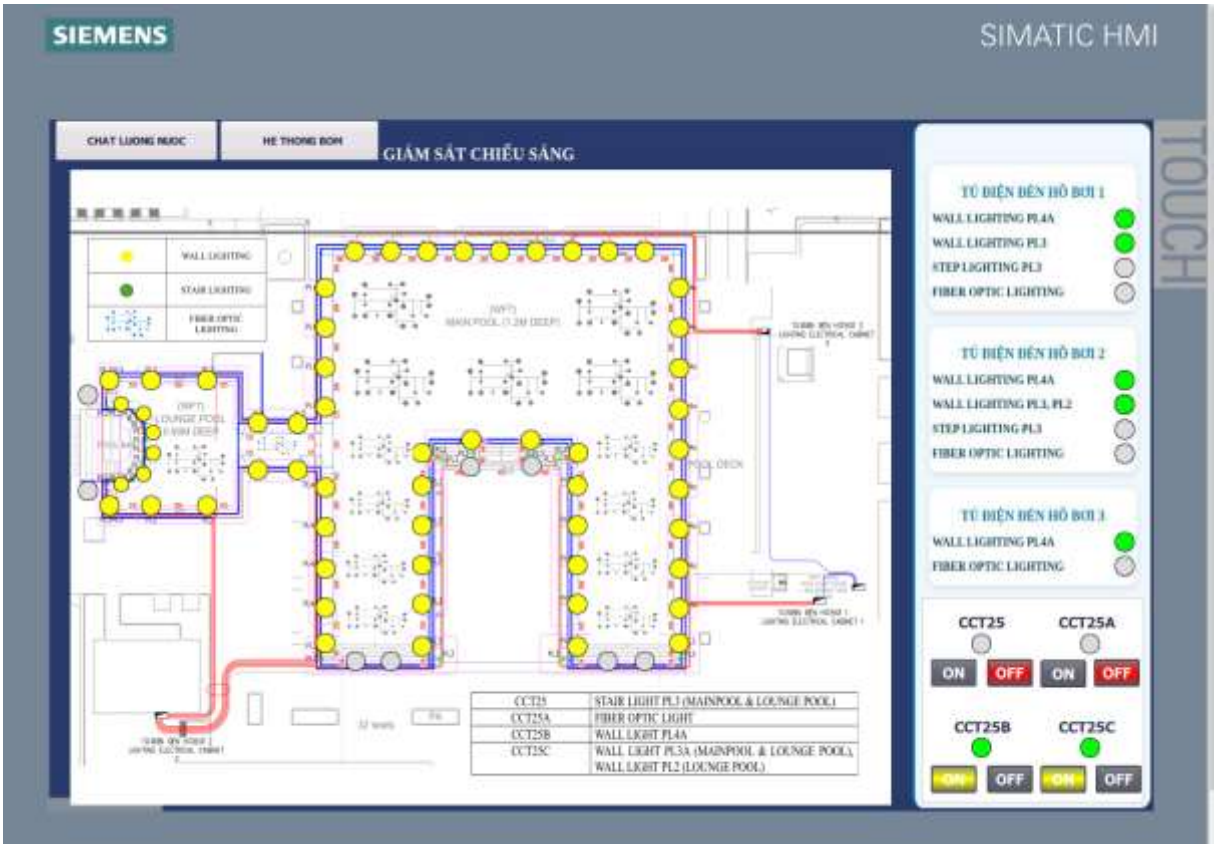
Hình 5.5 Mô phỏng hệ thống giám sát chất lượng nước

Hình 5.6 Mô phỏng hệ thống điều khiển máy bơm

Nhận xét:

- Cảm biến pH, clo, mực nước được kết nối và truyền dữ liệu thành công đến PLC thông qua giao thức Modbus RTU.
- Chương trình PLC đã điều khiển chính xác bơm lọc, bơm hóa chất và van cấp nước theo ngưỡng giới hạn đã định.
- Giao diện HMI hiển thị đầy đủ trạng thái thiết bị và giá trị đo của cảm biến theo thời gian thực.

5.2.2 Hệ thống chiếu sáng



Hình 5.7 Mô phỏng hệ thống giám sát chiếu sáng

Nhận xét: Các đèn trong hệ thống hoạt động đúng lịch trình, đồng thời người vận hành có thể bật/tắt từ xa qua Web hoặc bằng màn hình HMI

5.2.3 Giao diện giám sát Webserver

Trong hình 5.8, ta thấy được quá trình vận hành hệ thống sẽ lưu lại lịch sử đưa vào database, qua đó người dùng có thể xuất báo cáo dưới dạng file excel dễ dàng cho công tác kiểm tra và quản lý.

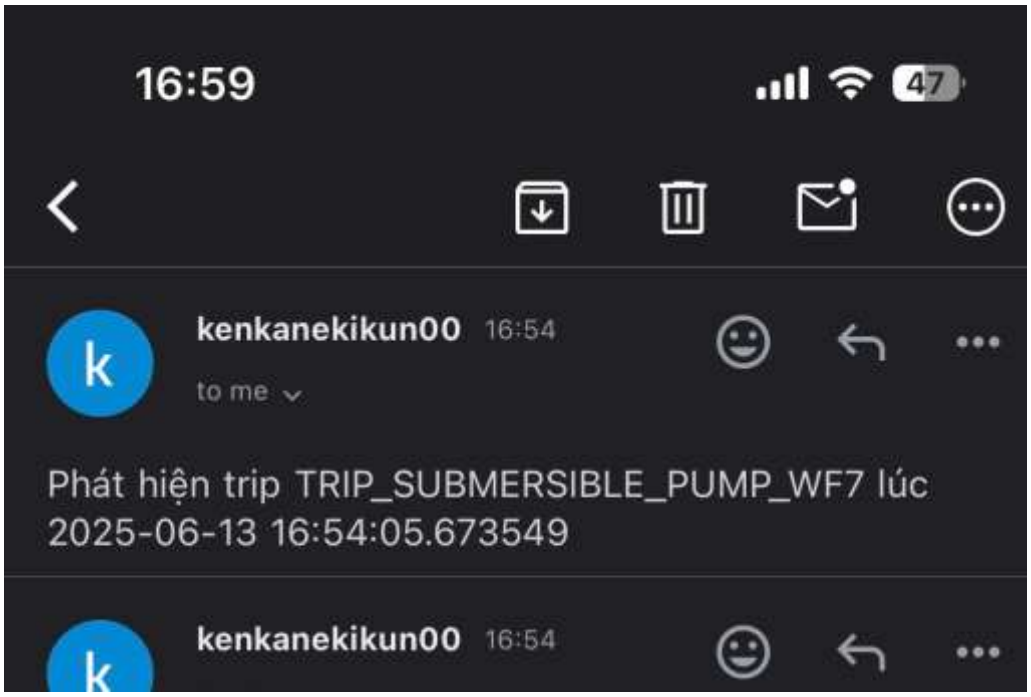
Hình 5.8 Tính năng truy cập lịch sử hoạt động

Qua hình 5.9, hệ thống đảm bảo được khả năng điều khiển bật - tắt từ xa thông qua Webserver.

Hình 5.9 Điều khiển bơm thông qua Webserver

Qua hình 5.10, hình 5.11, hình 5.12 hệ thống sẽ đưa ra cảnh báo khi xuất hiện tín hiệu cảnh báo lỗi như cảnh báo lỗi bơm, cảnh báo ngập phòng kỹ thuật, cảnh báo vượt ngưỡng Clo và pH an toàn được thu thập thông qua các cảm biến được kết nối với PLC.

Hình 5.10 Cảnh báo ngập phòng kỹ thuật

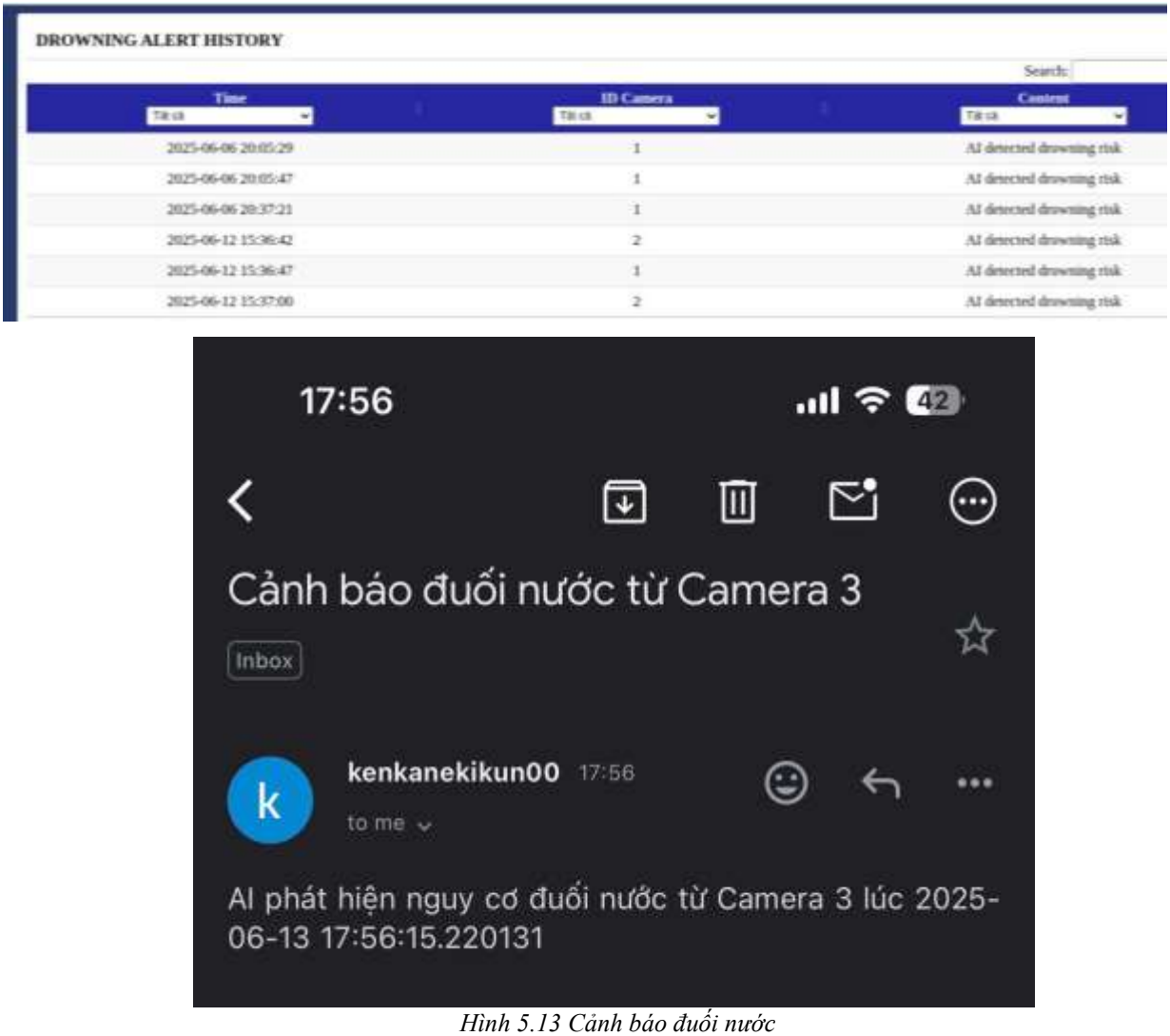


Hình 5.11 Cảnh báo lỗi bơm

Hình 5.12 Cảnh báo giá trị pH khi nằm ngoài ngưỡng an toàn

5.2.4 Hệ thống camera giám sát AI

Hình 5.13 mô tả hệ thống camera AI khi phát hiện đuối nước sẽ gửi tín hiệu đến PLC kích hoạt chuông báo động được đặt tại các góc của hồ bơi, đồng thời gửi email cảnh báo đến nhân viên cứu hộ.



Hình 5.13 Cảnh báo đuối nước

KẾT LUẬN

1. Đánh giá

Tổng kết, dự án này của nhóm đã thực hiện được yêu cầu đề ra, cụ thể:

- ✓ Điều khiển và giám sát từ xa
- ✓ Lưu trữ dữ liệu từ cảm biến
- ✓ Cảnh báo lỗi hệ thống thông qua email
- ✓ Cảnh báo lỗi khi nước sau xử lý không đạt tiêu chuẩn
- ✓ Hệ thống tự ngưng hoạt động khi có lỗi
- ✓ Lưu trữ sự cố hệ thống
- ✓ Cảnh báo nguy hiểm khi phát hiện người đuối nước

2. Hạn chế

Mô phỏng AI chưa áp dụng mô hình học sâu chuyên biệt: Cần cải tiến mô hình học sâu để tăng độ chính xác và hiệu quả trong việc phát hiện tình huống đuối nước. Hiện tại, mô hình AI sử dụng trong hệ thống chưa được tối ưu hóa cho các tình huống phức tạp hoặc những đối tượng với nhiều hành động khác nhau.

Chưa triển khai thực tế tích hợp phần cứng thật: Hệ thống đã được mô phỏng và thử nghiệm trên phần mềm, nhưng chưa được triển khai trên phần cứng thực tế. Cần thực hiện tích hợp thực tế để kiểm tra hiệu suất hệ thống và đảm bảo tính ổn định của các thiết bị và cảm biến trong môi trường thực tế.

3. Hướng phát triển

Tích hợp SCADA (WinCC, Ignition): Hệ thống cần được kết nối với các nền tảng SCADA như WinCC hoặc Ignition để quản lý và giám sát hệ thống từ xa dễ dàng hơn. Điều này sẽ giúp tăng cường khả năng theo dõi và điều khiển trong thời gian thực.

Phát triển AI theo dõi hành vi nâng cao hơn (hành động bất thường, tập trung đám đông...): Mô hình AI sẽ được phát triển để nhận diện hành vi bất thường khác ngoài đuối nước, chẳng hạn như người bơi bị ngất, hoặc các hành động của nhóm người (tập trung đám đông). Điều này sẽ giúp hệ thống phát hiện các tình huống nguy hiểm phức tạp hơn và nâng cao tính an toàn.

Ứng dụng thực tế tại các khu nghỉ dưỡng lớn: Mở rộng và triển khai hệ thống tại các khu nghỉ dưỡng lớn, khách sạn hoặc các bể bơi công cộng để nâng cao sự tiện lợi, an toàn cho người sử dụng và giúp chủ quản lý có thể giám sát hiệu quả hơn các khu vực bơi lội.

TÀI LIỆU THAM KHẢO

[1] ResearchGate, “Computer Vision Enabled Drowning Detection System, ” Dharshana Kasthurirathna et al., 2021

[2] AngelEye Safety Systems, “AI-powered drowning detection, ”

[3] ResearchGate, “An automatic drowning detection surveillance system for challenging outdoor pool environments, ” IEEE ICCV, 2003.

[4] PMC, “Exploring Recent Technological Approaches for Drowning Detection, ” 2024.

[5] Bộ Y tế, “Thông tư 41/2018/TT-BYT, ” Quy định chất lượng nước sạch sinh hoạt, QCVN 01-1: 2018/BYT, 2018.

[6] QCVN, “TCVN 4260: 2012 – Công trình thể thao, bể bơi – Tiêu chuẩn thiết kế, ” Bộ KH&CN, 2012.

[7] M. Shatnawi, F. Albreiki, A. Alkhoori và M. Alhebshi, “Deep Learning and Vision-Based Early Drowning Detection, ” Information, vol. 14, no. 1, p. 52, Jan. 2023.

[8] U. Handalage et al., “Computer Vision Enabled Drowning Detection System, ” in Proc. 3rd Intl. Conf. Advancements in Computing, Colombo, Sri Lanka, Dec. 2021

[9] He, Luhao & Zhou, Yongzhang & Liu, Lei & Ma, Jianhua. (2024) . Research and Application of YOLO11-Based Object Segmentation in Intelligent Recognition at Construction Sites. Buildings. 14.

[10] Siemens, “Master-Slave Communication via a CM PtP Using the Modbus RTU protocol”, Entry-ID 68202723, Version 2.0.1, Jan. 2018.

PHỤ LỤC 1: LẬP TRÌNH SERVER RASPBERRY PI

```
import eventlet

eventlet.monkey_patch ()

from flask import Flask, render_template, Response, request, send_file,
jsonify
from io import BytesIO
import pandas as pd
from flask_socketio import SocketIO, emit
import cv2
import numpy as np
from ultralytics import YOLO
import snap7
from snap7.util import get_real, get_bool, set_bool, get_int
import time
import mysql.connector
from datetime import datetime
import threading
import smtplib
from email.mime.multipart import MIMEMultipart  # Thêm import này
from email.mime.text import MIMEText  # Thêm import này
from email.mime.application import MIMEApplication

# -------------------- FLASK AND SOCKET.IO CONFIG --------------------
app = Flask (__name__)
socketio = SocketIO (app, cors_allowed_origins="*")

# -------------------- YOLO CONFIG --------------------
model = YOLO ("models_cus/best_ncnn_model")
labels = model.names
input_size =  (640, 640)

camera_sources = [

"rtsp://admin:11022002Tt@192.168.63.80:554/Streaming/Channels/101?rtsp_tran
sport=udp",

"rtsp://admin:11022002Tt@192.168.63.80:554/Streaming/Channels/101?rtsp_tran
sport=udp",

"rtsp://admin:11022002Tt@192.168.63.80:554/Streaming/Channels/102?rtsp_tran
sport=udp",
]


# -------------------- YOLO DETECTION + DROWNING ALERT --------------------
def yolo_detect (frame, cam_id) :
    global last_person_detect_time, person_detected, alarm_sent

    results = model (frame, verbose=False)
    detections = results[0].boxes
    found_person = False

    for i in range (len (detections) ) :
        xyxy_tensor = detections[i].xyxy.cpu ()
        xyxy = xyxy_tensor.numpy () .squeeze ()
```

```
            xmin, ymin, xmax, ymax = xyxy.astype (int)

            classidx = int (detections[i].cls.item () )
            classname = labels[classidx]
            conf = detections[i].conf.item ()

            if conf > 0.5:
                if classname.lower () == "person":
                    found_person = True

                color =  (0, 255, 0)
                cv2.rectangle (frame,  (xmin, ymin) ,  (xmax, ymax) , color, 2)
                label = f'{classname}: {int (conf * 100) }%'
                cv2.putText (frame, label,  (xmin, ymin - 10) ,
cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

        current_time = time.time ()

        if found_person:
            if not person_detected:
                last_person_detect_time = current_time
                person_detected = True
                alarm_sent = False
            elif not alarm_sent:
                remaining = ALARM_THRESHOLD -  (current_time -
last_person_detect_time)
                if remaining > 0:
                    print (f"⏳ Camera {cam_id} - Đếm ngược phát hiện người:
{remaining:.1f}s", end='\r')
                else:
                    print (f"\n🚨 Camera {cam_id} - Phát hiện người trong
{ALARM_THRESHOLD}s - Gửi cảnh báo đuối nước")
                    # Gửi tín hiệu cảnh báo đuối nước vào PLC cho từng camera
                    if cam_id == 1:
                        fn_Data_Write ( (34, 0) , True)  # Camera 1
                    elif cam_id == 2:
                        fn_Data_Write ( (34, 1) , True)  # Camera 2
                    elif cam_id == 3:
                        fn_Data_Write ( (34, 2) , True)  # Camera 3
                    elif cam_id == 4:
                        fn_Data_Write ( (34, 3) , True)  # Camera 4

                    write_drowning_event_to_sql (cam_id)
                    alarm_sent = True
        else:
            person_detected = False
            alarm_sent = False

        return frame


# -------------------- VIDEO STREAM --------------------
def generate_frames (cam_id) :
    if cam_id >= len (camera_sources) or not camera_sources[cam_id]:
        return
    cap = cv2.VideoCapture (camera_sources[cam_id], cv2.CAP_FFMPEG)

    # Force OpenCV to retain the latest frame only
    cap.set (cv2.CAP_PROP_BUFFERSIZE, 1)

    prev_time = time.time ()
```

```
    while cap.isOpened () :
        ret, frame = cap.read ()
        if not ret:
            continue
        frame = yolo_detect (frame, cam_id)  # Truyền cam_id vào đây
        curr_time = time.time ()
        fps = 1.0 /  (curr_time - prev_time)
        prev_time = curr_time
        cv2.putText (frame, f"FPS: {fps:.2f}",  (10, 30) ,
cv2.FONT_HERSHEY_SIMPLEX, 1,  (0, 255, 0) , 2)
        _, buffer = cv2.imencode ('.jpg', frame)
        frame_bytes = buffer.tobytes ()
        yield  (b'--frame\r\nContent-Type: image/jpeg\r\n\r\n' +
frame_bytes + b'\r\n')
    cap.release ()


# ---------- EMAIL CONFIG -------------
EMAIL_ADDRESS = "kenkanekikun00@gmail.com"  # Thay bằng email của bạn
EMAIL_PASSWORD = "fetymkzmgtmjgfaa"  # Thay bằng app password Gmail của bạn
EMAIL_RECEIVERS = ["htphong11022002@gmail.com"]  # Thay bằng email người
nhận


def send_alert_email (subject, body, to_emails=EMAIL_RECEIVERS) :
    msg = MIMEMultipart ()
    msg['From'] = EMAIL_ADDRESS
    msg['To'] = ", ".join (to_emails)
    msg['Subject'] = subject
    msg.attach (MIMEText (body, 'plain') )
    try:
        server = smtplib.SMTP ('smtp.gmail.com', 587)
        server.starttls ()
        server.login (EMAIL_ADDRESS, EMAIL_PASSWORD)
        server.sendmail (EMAIL_ADDRESS, to_emails, msg.as_string () )
        server.quit ()
        print ("Email alert sent successfully")
    except Exception as e:
        print ("Failed to send email:", e)


def send_email_with_attachment (subject, body, attachment, filename,
to_emails) :
    msg = MIMEMultipart ()
    msg['From'] = EMAIL_ADDRESS
    msg['To'] = ", ".join (to_emails)
    msg['Subject'] = subject
    msg.attach (MIMEText (body, 'plain') )

    part = MIMEApplication (attachment.read () , Name=filename)
    part['Content-Disposition'] = f'attachment; filename="{filename}"'
    msg.attach (part)

    try:
        server = smtplib.SMTP ('smtp.gmail.com', 587)
        server.starttls ()
        server.login (EMAIL_ADDRESS, EMAIL_PASSWORD)
        server.sendmail (EMAIL_ADDRESS, to_emails, msg.as_string () )
        server.quit ()
        print ("✅ Email báo cáo đã gửi thành công.")
    except Exception as e:
```

```
        print ("❌ Gửi email thất bại:", e)


# -------------------- ALERT CHECK --------------------
last_alert_times = {}  # Lưu thời gian gửi email để tránh gửi nhiều lần
# Giá trị ngưỡng cảnh báo
CLO_MIN = 0.2
CLO_MAX = 1.0
PH_MIN = 7.2
PH_MAX = 7.6


def check_trip_and_alert (plc_data) :
    now = time.time ()
    ALERT_INTERVAL = 60  # Thời gian trễ giữa các lần gửi email

    # Kiểm tra trip bơm
    for pump, trip in plc_data.items () :
        if trip and pump.startswith ('TRIP') :
            last_time = last_alert_times.get (pump, 0)
            if now - last_time > ALERT_INTERVAL:  # Gửi email nếu chưa gửi
trong ALERT_INTERVAL
                # Gửi email cảnh báo trip bơm
                subject = f"Cảnh báo TRIP bơm: {pump}"
                body = f"Phát hiện trip {pump} lúc {datetime.now () }"
                send_alert_email (subject, body)
                last_alert_times[pump] = now  # Cập nhật thời gian gửi
email

    # Kiểm tra tín hiệu AI_Drowning_1, AI_Drowning_2, AI_Drowning_3,
AI_Drowning_4
    for cam_id in range (1, 5) :  # Từ camera 1 đến camera 4
        key = f'AI_Drowning_{cam_id}'  # AI_Drowning_1, AI_Drowning_2,
AI_Drowning_3, AI_Drowning_4
        if plc_data.get (key) :  # Nếu tín hiệu AI_Drowning_x là True
(phát hiện đuối nước)
            last_time = last_alert_times.get (key, 0)
            if now - last_time > ALERT_INTERVAL:  # Gửi email nếu chưa gửi
trong ALERT_INTERVAL
                # Gửi email cảnh báo đuối nước cho từng camera
                subject = f"Cảnh báo đuối nước từ camera {cam_id}"
                body = f"AI phát hiện nguy cơ đuối nước từ camera {cam_id}
lúc {datetime.now () }"
                send_alert_email (subject, body)
                last_alert_times[key] = now  # Cập nhật thời gian gửi email

    # Kiểm tra Clo
    if 'Clo' in plc_data:
        clo_val = plc_data['Clo']
        if clo_val < CLO_MIN or clo_val > CLO_MAX:
            key = 'ALERT_CLO'
            last_time = last_alert_times.get (key, 0)
            if now - last_time > ALERT_INTERVAL:
                status = "thấp" if clo_val < CLO_MIN else "cao"
                subject = f"Cảnh báo: Nồng độ Clo {status}"
                body = f"Nồng độ Clo hiện tại là {clo_val:.2f} ppm, nằm
ngoài giới hạn an toàn  ({CLO_MIN} - {CLO_MAX}) lúc {datetime.now () }"
                send_alert_email (subject, body)
                last_alert_times[key] = now

    # Kiểm tra pH
```

```
    if 'pH' in plc_data:
        ph_val = plc_data['pH']
        if ph_val < PH_MIN or ph_val > PH_MAX:
            key = 'ALERT_PH'
            last_time = last_alert_times.get (key, 0)
            if now - last_time > ALERT_INTERVAL:
                status = "thấp" if ph_val < PH_MIN else "cao"
                subject = f"Cảnh báo: Giá trị pH {status}"
                body = f"Giá trị pH hiện tại là {ph_val:.2f}, nằm ngoài
giới hạn an toàn  ({PH_MIN} - {PH_MAX}) lúc {datetime.now () }"
                send_alert_email (subject, body)
                last_alert_times[key] = now


# -------------------- MySQL CONFIG --------------------
db = mysql.connector.connect (
    host="localhost",
    user="root",
    password="11022002Tt",
    database="plc_data"
)
cursor = db.cursor ()


def write_to_sql (plc_data) :
    query = """
    INSERT INTO plc_water_quality_data  (timestamp, temperature, ph,
chlorine)
    VALUES  (%s, %s, %s, %s)
    """
    values =  (
        datetime.now () ,
        plc_data['TEMP'],
        plc_data['pH'],
        plc_data['Clo']
    )
    cursor.execute (query, values)
    db.commit ()


lastPumpStates = {}
lastMode = None


def write_pump_activity_to_sql (plc_data) :
    global lastPumpStates, lastMode

    currentMode = 'AUTO' if plc_data.get ('AUTO') else 'MAN'
    mode_changed =  (lastMode is not None and currentMode != lastMode)
    lastMode = currentMode

    pumps = [
         ('FILTER_PUMP_1', 'START_FILTER_PUMP_1', 'STOP_FILTER_PUMP_1') ,
         ('FILTER_PUMP_2', 'START_FILTER_PUMP_2', 'STOP_FILTER_PUMP_2') ,
         ('FILTER_PUMP_3', 'START_FILTER_PUMP_3', 'STOP_FILTER_PUMP_3') ,
         ('FILTER_PUMP_4', 'START_FILTER_PUMP_4', 'STOP_FILTER_PUMP_4') ,
         ('FILTER_PUMP_5', 'START_FILTER_PUMP_5', 'STOP_FILTER_PUMP_5') ,
         ('FILTER_PUMP_6', 'START_FILTER_PUMP_6', 'STOP_FILTER_PUMP_6') ,
         ('SUBMERSIBLE_PUMP_WF7', 'START_SUBMERSIBLE_PUMP_WF7',
'STOP_SUBMERSIBLE_WF7') ,
         ('SUBMERSIBLE_PUMP_WF8', 'START_SUBMERSIBLE_PUMP_WF8',
```

```
'STOP_SUBMERSIBLE_WF8') ,
          ('RUN_DOSSI_PUMP_1', 'START_DOSSI_PUMP_1', 'STOP_DOSSI_PUMP_1') ,
          ('RUN_DOSSI_PUMP_2', 'START_DOSSI_PUMP_2', 'STOP_DOSSI_PUMP_2') ,
          ('CIRCULATION_PUMP_1', 'START_CIRCULATION_PUMP_1',
'STOP_CIRCULATION_PUMP_1') ,
          ('CIRCULATION_PUMP_2', 'START_CIRCULATION_PUMP_2',
'STOP_CIRCULATION_PUMP_2')
    ]

    for tag, _, _ in pumps:
        current = plc_data.get (tag, False)
        previous = lastPumpStates.get (tag)

        # 1. Nếu trạng thái bơm thay đổi
        if previous is not None and current != previous:
            query = """
            INSERT INTO plc_pump  (timestamp_start, timestamp_stop,
pump_name, mode, status)
            VALUES  (%s, %s, %s, %s, %s)
            """
            now = datetime.now ()
            values =  (
                now if current else None,
                None if current else now,
                tag,
                currentMode,
                'ON' if current else 'OFF'
            )
            cursor.execute (query, values)
            db.commit ()

        # 2. Nếu mode thay đổi thì ghi trạng thái hiện tại của bơm
        elif mode_changed:
            query = """
            INSERT INTO plc_pump  (timestamp_start, timestamp_stop,
pump_name, mode, status)
            VALUES  (%s, %s, %s, %s, %s)
            """
            now = datetime.now ()
            values =  (
                now if current else None,
                None if current else now,
                tag,
                currentMode,
                'ON' if current else 'OFF'
            )
            cursor.execute (query, values)
            db.commit ()

        lastPumpStates[tag] = current


def write_trip_status_to_sql (plc_data) :
    trip_pumps = [
        'TRIP_FILTER_PUMP_1',
        'TRIP_FILTER_PUMP_2',
        'TRIP_FILTER_PUMP_3',
        'TRIP_FILTER_PUMP_4',
        'TRIP_FILTER_PUMP_5',
        'TRIP_FILTER_PUMP_6',
        'TRIP_SUBMERSIBLE_PUMP_WF7',
```

```
        'TRIP_SUBMERSIBLE_PUMP_WF8',
        'TRIP_DOSSI_PUMP_1',
        'TRIP_DOSSI_PUMP_2',
        'TRIP_CIRCULATION_PUMP_1',
        'TRIP_CIRCULATION_PUMP_2',
        'FLOODED_TECHNICAL_ROOM'
    ]

    for pump in trip_pumps:
        trip = plc_data.get (pump)

        if trip is not None:  # chỉ lưu khi có trip tín hiệu
            if pump == 'FLOODED_TECHNICAL_ROOM':
                trip_status = 2
            else:
                trip_status = trip

            query = """
            INSERT INTO plc_pump_trip  (timestamp, trip_signal,
trip_status)
            VALUES  (%s, %s, %s)
            """
            values =  (datetime.now () , pump, trip_status)
            cursor.execute (query, values)
            db.commit ()


def write_drowning_event_to_sql (camera_id) :
    query = """
    INSERT INTO ai_drowning_events  (timestamp, camera_id, message)
    VALUES  (%s, %s, %s)
    """
    values =  (datetime.now () , camera_id, "AI detected drowning risk")
    cursor.execute (query, values)
    db.commit ()


# -------------------- PLC CONFIG --------------------
PLC_IP = '192.168.1.24'
RACK = 0
SLOT = 1
DB_NUMBER = 1

tagArr = {}


# -------------------- PLC DATA READER --------------------
def read_plc_data () :
    client = snap7.client.Client ()
    try:
        client.connect (PLC_IP, RACK, SLOT)
        data = client.db_read (DB_NUMBER, 0, 35)
        return {
            'pH': get_real (data, 6) ,
            'Clo': get_real (data, 10) ,
            'TEMP': get_real (data, 14) ,
            'WATER_LEVEL_BLANCED_TANK': get_int (data, 20) ,
            'WATER_LEVEL_SUMPIT_TANK': get_int (data, 22) ,

            'AUTO': get_bool (data, 0, 0) ,
            'MAN': get_bool (data, 0, 1) ,
```

```
'START': get_bool (data, 0, 2) ,
'STOP': get_bool (data, 0, 3) ,
'RUN_AUTO': get_bool (data, 0, 4) ,
'RUN_MAN': get_bool (data, 0, 5) ,

'FILTER_PUMP_1': get_bool (data, 0, 6) ,
'FILTER_PUMP_2': get_bool (data, 0, 7) ,
'FILTER_PUMP_3': get_bool (data, 1, 0) ,
'FILTER_PUMP_4': get_bool (data, 1, 1) ,
'FILTER_PUMP_5': get_bool (data, 1, 2) ,
'FILTER_PUMP_6': get_bool (data, 1, 3) ,
'SUBMERSIBLE_PUMP_WF7': get_bool (data, 1, 4) ,
'SUBMERSIBLE_PUMP_WF8': get_bool (data, 1, 5) ,
'RUN_DOSSI_PUMP_1': get_bool (data, 24, 0) ,
'RUN_DOSSI_PUMP_2': get_bool (data, 24, 1) ,
'CIRCULATION_PUMP_1': get_bool (data, 24, 6) ,
'CIRCULATION_PUMP_2': get_bool (data, 24, 7) ,

'TRIP_FILTER_PUMP_1': get_bool (data, 1, 6) ,
'TRIP_FILTER_PUMP_2': get_bool (data, 1, 7) ,
'TRIP_FILTER_PUMP_3': get_bool (data, 2, 0) ,
'TRIP_FILTER_PUMP_4': get_bool (data, 2, 1) ,
'TRIP_FILTER_PUMP_5': get_bool (data, 2, 2) ,
'TRIP_FILTER_PUMP_6': get_bool (data, 2, 3) ,
'TRIP_SUBMERSIBLE_PUMP_WF7': get_bool (data, 2, 4) ,
'TRIP_SUBMERSIBLE_PUMP_WF8': get_bool (data, 2, 5) ,
'TRIP_DOSSI_PUMP_1': get_bool (data, 2, 6) ,
'TRIP_DOSSI_PUMP_2': get_bool (data, 2, 7) ,
'TRIP_CIRCULATION_PUMP_1': get_bool (data, 25, 4) ,
'TRIP_CIRCULATION_PUMP_2': get_bool (data, 25, 5) ,

'VALVE': get_bool (data, 3, 0) ,
'FLOODED_TECHNICAL_ROOM': get_bool (data, 3, 1) ,
'CCT25': get_bool (data, 3, 2) ,
'CCT25A': get_bool (data, 3, 3) ,
'CCT25B': get_bool (data, 3, 4) ,
'CCT25C': get_bool (data, 3, 5) ,

'START_FILTER_PUMP_1': get_bool (data, 3, 6) ,
'START_FILTER_PUMP_2': get_bool (data, 3, 7) ,
'START_FILTER_PUMP_3': get_bool (data, 4, 0) ,
'START_FILTER_PUMP_4': get_bool (data, 4, 1) ,
'START_FILTER_PUMP_5': get_bool (data, 4, 2) ,
'START_FILTER_PUMP_6': get_bool (data, 4, 3) ,
'START_SUBMERSIBLE_PUMP_WF7': get_bool (data, 4, 4) ,
'START_SUBMERSIBLE_PUMP_WF8': get_bool (data, 4, 5) ,
'START_DOSSI_PUMP_1': get_bool (data, 24, 2) ,
'START_DOSSI_PUMP_2': get_bool (data, 24, 3) ,
'START_CIRCULATION_PUMP_1': get_bool (data, 25, 0) ,
'START_CIRCULATION_PUMP_2': get_bool (data, 25, 1) ,

'STOP_FILTER_PUMP_1': get_bool (data, 18, 5) ,
'STOP_FILTER_PUMP_2': get_bool (data, 18, 6) ,
'STOP_FILTER_PUMP_3': get_bool (data, 18, 7) ,
'STOP_FILTER_PUMP_4': get_bool (data, 19, 0) ,
'STOP_FILTER_PUMP_5': get_bool (data, 19, 1) ,
'STOP_FILTER_PUMP_6': get_bool (data, 19, 2) ,
'STOP_SUBMERSIBLE_WF7': get_bool (data, 19, 3) ,
'STOP_SUBMERSIBLE_WF8': get_bool (data, 19, 4) ,
'STOP_DOSSI_PUMP_1': get_bool (data, 24, 4) ,
'STOP_DOSSI_PUMP_2': get_bool (data, 24, 5) ,
```

```
            'STOP_CIRCULATION_PUMP_1': get_bool (data, 25, 2) ,
            'STOP_CIRCULATION_PUMP_2': get_bool (data, 25, 3) ,

            'ON_CCT25': get_bool (data, 4, 6) ,
            'ON_CCT25A': get_bool (data, 4, 7) ,
            'ON_CCT25B': get_bool (data, 5, 0) ,
            'ON_CCT25C': get_bool (data, 18, 4) ,

            'OFF_CCT25': get_bool (data, 18, 0) ,
            'OFF_CCT2A': get_bool (data, 18, 1) ,
            'OFF_CCT2B': get_bool (data, 18, 2) ,
            'OFF_CCT25C': get_bool (data, 18, 3) ,

            'SALT_CLORINATOR': get_bool (data, 5, 1) ,
            'Al_Drowning_1': get_bool (data, 34, 0) ,
            'Al_Drowning_2': get_bool (data, 34, 1) ,
            'Al_Drowning_3': get_bool (data, 34, 2) ,
            'Al_Drowning_4': get_bool (data, 34, 3) ,
        }
    except Exception as e:
        return {'error': str (e) }
    finally:
        client.disconnect ()


# -------------------- GHI DỮ LIỆU XUỐNG PLC --------------------
def fn_Data_Write (offset_bit, value) :
    try:
        client = snap7.client.Client ()
        client.connect (PLC_IP, RACK, SLOT)
        data = bytearray (1)
        set_bool (data, 0, offset_bit[1], value)
        client.db_write (DB_NUMBER, offset_bit[0], data)
        client.disconnect ()
    except Exception as e:
        print ("❌ Lỗi ghi PLC:", e)


# -------------------- SOCKET.IO HANDLERS --------------------
@socketio.on ('cmd_AUTO')
def handle_cmd_AUTO (data) :
    print (f"📥 cmd_AUTO = {data}")
    fn_Data_Write ( (0, 0) , data)

@socketio.on ('cmd_MAN')
def handle_cmd_MAN (data) :
    print (f"📥 cmd_MAN = {data}")
    fn_Data_Write ( (0, 1) , data)

@socketio.on ('cmd_START')
def handle_cmd_START (data) :
    print (f"📥 cmd_START = {data}")
    fn_Data_Write ( (0, 2) , data)

@socketio.on ('cmd_STOP')
def handle_cmd_STOP (data) :
    print (f"📥 cmd_STOP = {data}")
    fn_Data_Write ( (0, 3) , data)
```

```python
@socketio.on ('cmd_ON_CCT25')
def handle_cmd_ON_CCT25 (data) :
    print (f"📥 cmd_ON_CCT25 = {data}")
    fn_Data_Write ( (4, 6) , data)


@socketio.on ('cmd_ON_CCT25A')
def handle_cmd_ON_CCT25A (data) :
    print (f"📥 cmd_ON_CCT25A = {data}")
    fn_Data_Write ( (4, 7) , data)


@socketio.on ('cmd_ON_CCT25B')
def handle_cmd_ON_CCT25B (data) :
    print (f"📥 cmd_ON_CCT25B = {data}")
    fn_Data_Write ( (5, 0) , data)


@socketio.on ('cmd_ON_CCT25C')
def handle_cmd_ON_CCT25C (data) :
    print (f"📥 cmd_ON_CCT25C = {data}")
    fn_Data_Write ( (18, 4) , data)


@socketio.on ('cmd_OFF_CCT25')
def handle_cmd_OFF_CCT25 (data) :
    print (f"📥 cmd_OFF_CCT25 = {data}")
    fn_Data_Write ( (18, 0) , data)


@socketio.on ('cmd_OFF_CCT25A')
def handle_cmd_OFF_CCT25A (data) :
    print (f"📥 cmd_OFF_CCT25A = {data}")
    fn_Data_Write ( (18, 1) , data)


@socketio.on ('cmd_OFF_CCT25B')
def handle_cmd_OFF_CCT25B (data) :
    print (f"📥 cmd_OFF_CCT25B = {data}")
    fn_Data_Write ( (18, 2) , data)


@socketio.on ('cmd_OFF_CCT25C')
def handle_cmd_OFF_CCT25C (data) :
    print (f"📥 cmd_OFF_CCT25C = {data}")
    fn_Data_Write ( (18, 3) , data)


@socketio.on ('cmd_START_FILTER_PUMP_1')
def handle_START_FILTER_PUMP_1 (data) :
    print (f"📥 cmd_START_FILTER_PUMP_1 = {data}")
    fn_Data_Write ( (3, 6) , data)


@socketio.on ('cmd_START_FILTER_PUMP_2')
def handle_START_FILTER_PUMP_2 (data) :
    print (f"📥 cmd_START_FILTER_PUMP_2 = {data}")
    fn_Data_Write ( (3, 7) , data)
```

```python
@socketio.on ('cmd_START_FILTER_PUMP_3')
def handle_START_FILTER_PUMP_3 (data) :
    print (f"📥 cmd_START_FILTER_PUMP_3 = {data}")
    fn_Data_Write ( (4, 0) , data)

@socketio.on ('cmd_START_FILTER_PUMP_4')
def handle_START_FILTER_PUMP_4 (data) :
    print (f"📥 cmd_START_FILTER_PUMP_4 = {data}")
    fn_Data_Write ( (4, 1) , data)

@socketio.on ('cmd_START_FILTER_PUMP_5')
def handle_START_FILTER_PUMP_5 (data) :
    print (f"📥 cmd_START_FILTER_PUMP_5 = {data}")
    fn_Data_Write ( (4, 2) , data)

@socketio.on ('cmd_START_FILTER_PUMP_6')
def handle_START_FILTER_PUMP_6 (data) :
    print (f"📥 cmd_START_FILTER_PUMP_6 = {data}")
    fn_Data_Write ( (4, 3) , data)

@socketio.on ('cmd_START_SUBMERSIBLE_PUMP_WF7')
def handle_START_SUBMERSIBLE_PUMP_WF7 (data) :
    print (f"📥 cmd_START_SUBMERSIBLE_PUMP_WF7 = {data}")
    fn_Data_Write ( (4, 4) , data)

@socketio.on ('cmd_START_SUBMERSIBLE_PUMP_WF8')
def handle_START_SUBMERSIBLE_PUMP_WF8 (data) :
    print (f"📥 cmd_START_SUBMERSIBLE_PUMP_WF8 = {data}")
    fn_Data_Write ( (4, 5) , data)

@socketio.on ('cmd_START_DOSSI_PUMP_1')
def handle_START_DOSSI_PUMP_1 (data) :
    print (f"📥 cmd_START_DOSSI_PUMP_1 = {data}")
    fn_Data_Write ( (24, 2) , data)

@socketio.on ('cmd_START_DOSSI_PUMP_2')
def handle_START_DOSSI_PUMP_2 (data) :
    print (f"📥 cmd_START_DOSSI_PUMP_2 = {data}")
    fn_Data_Write ( (24, 3) , data)

@socketio.on ('cmd_STOP_DOSSI_PUMP_1')
def handle_STOP_DOSSI_PUMP_1 (data) :
    print (f"📥 cmd_STOP_DOSSI_PUMP_1 = {data}")
    fn_Data_Write ( (24, 4) , data)

@socketio.on ('cmd_STOP_DOSSI_PUMP_2')
def handle_STOP_DOSSI_PUMP_2 (data) :
    print (f"📥 cmd_STOP_DOSSI_PUMP_2 = {data}")
    fn_Data_Write ( (24, 5) , data)

@socketio.on ('cmd_START_CIRCULATION_PUMP_1')
def handle_START_CIRCULATION_PUMP_1 (data) :
    print (f"📥 cmd_START_CIRCULATION_PUMP_1 = {data}")
    fn_Data_Write ( (25, 0) , data)

@socketio.on ('cmd_START_CIRCULATION_PUMP_2')
def handle_START_CIRCULATION_PUMP_2 (data) :
    print (f"📥 cmd_START_CIRCULATION_PUMP_2 = {data}")
    fn_Data_Write ( (25, 1) , data)
```

```python
@socketio.on ('cmd_STOP_FILTER_PUMP_1')
def handle_STOP_FILTER_PUMP_1 (data) :
    print (f"📥 cmd_STOP_FILTER_PUMP_1 = {data}")
    fn_Data_Write ( (18, 5) , data)

@socketio.on ('cmd_STOP_FILTER_PUMP_2')
def handle_STOP_FILTER_PUMP_2 (data) :
    print (f"📥 cmd_STOP_FILTER_PUMP_2 = {data}")
    fn_Data_Write ( (18, 6) , data)

@socketio.on ('cmd_STOP_FILTER_PUMP_3')
def handle_STOP_FILTER_PUMP_3 (data) :
    print (f"📥 cmd_STOP_FILTER_PUMP_3 = {data}")
    fn_Data_Write ( (18, 7) , data)

@socketio.on ('cmd_STOP_FILTER_PUMP_4')
def handle_STOP_FILTER_PUMP_4 (data) :
    print (f"📥 cmd_STOP_FILTER_PUMP_4 = {data}")
    fn_Data_Write ( (19, 0) , data)

@socketio.on ('cmd_STOP_FILTER_PUMP_5')
def handle_STOP_FILTER_PUMP_5 (data) :
    print (f"📥 cmd_STOP_FILTER_PUMP_5 = {data}")
    fn_Data_Write ( (19, 1) , data)

@socketio.on ('cmd_STOP_FILTER_PUMP_6')
def handle_STOP_FILTER_PUMP_6 (data) :
    print (f"📥 cmd_STOP_FILTER_PUMP_6 = {data}")
    fn_Data_Write ( (19, 2) , data)

@socketio.on ('cmd_STOP_SUBMERSIBLE_PUMP_WF7')
def handle_STOP_SUBMERSIBLE_WF7 (data) :
    print (f"📥 cmd_STOP_SUBMERSIBLE_PUMP_WF7 = {data}")
    fn_Data_Write ( (19, 3) , data)

@socketio.on ('cmd_STOP_SUBMERSIBLE_PUMP_WF8')
def handle_STOP_SUBMERSIBLE_WF8 (data) :
    print (f"📥 cmd_STOP_SUBMERSIBLE_PUMP_WF8 = {data}")
    fn_Data_Write ( (19, 4) , data)

@socketio.on ('cmd_STOP_CIRCULATION_PUMP_1')
def handle_STOP_CIRCULATION_PUMP_1 (data) :
    print (f"📥 cmd_STOP_CIRCULATION_PUMP_1 = {data}")
    fn_Data_Write ( (25, 2) , data)

@socketio.on ('cmd_STOP_CIRCULATION_PUMP_2')
def handle_STOP_CIRCULATION_PUMP_2 (data) :
    print (f"📥 cmd_STOP_CIRCULATION_PUMP_2 = {data}")
    fn_Data_Write ( (25, 3) , data)

@socketio.on ('cmd_AI_Drowning')
def handle_cmd_AI_Drowning (data) :
    cam_id = data.get ('cam_id')
    value = bool (data.get ('value') )
    print (f'📥 cmd_AI_Drowning | Camera {cam_id} = {value}')

    # Gửi tín hiệu về PLC cho từng camera
    if cam_id == 1:
```

```
        fn_Data_Write ( (34, 0) , value)  # Camera 1: Gửi tín hiệu
AI_Drowning_1
    elif cam_id == 2:
        fn_Data_Write ( (34, 1) , value)  # Camera 2: Gửi tín hiệu
AI_Drowning_2
    elif cam_id == 3:
        fn_Data_Write ( (34, 2) , value)  # Camera 3: Gửi tín hiệu
AI_Drowning_3
    elif cam_id == 4:
        fn_Data_Write ( (34, 3) , value)  # Camera 4: Gửi tín hiệu
AI_Drowning_4

    if value:
        write_drowning_event_to_sql (cam_id)  # Ghi sự kiện vào cơ sở dữ
liệu
        # Gửi email cảnh báo đuối nước
        subject = f"Cảnh báo đuối nước từ Camera {cam_id}"
        body = f"AI phát hiện nguy cơ đuối nước từ Camera {cam_id} lúc
{datetime.now () }"
        send_alert_email (subject, body)  # Gửi email cảnh báo


@socketio.on ('flooded_alert_triggered')
def handle_flooded_alert () :
    now = time.time ()
    key = 'FLOODED_TECHNICAL_ROOM'
    ALERT_INTERVAL = 300  # mỗi 5 phút gửi lại 1 lần nếu vẫn ngập
    last_time = last_alert_times.get (key, 0)

    if now - last_time > ALERT_INTERVAL:
        subject = "Cảnh báo: Phòng điều khiển bị ngập!"
        body = f"Hệ thống phát hiện phòng điều khiển bị ngập lúc
{datetime.now () .strftime ('%Y-%m-%d %H:%M:%S') }."
        send_alert_email (subject, body)
        last_alert_times[key] = now


# -------------------- ĐỌC VÀ GỬI TAG RA GIAO DIỆN --------------------
def fn_tagRead () :
    global tagArr
    tagArr = read_plc_data ()
    for k, v in tagArr.items () :
        socketio.emit (k, v)

def fn_tag () :
    for key, value in tagArr.items () :
        socketio.emit (key, value)

def fn_read_data_scan () :
    fn_tagRead ()
    fn_tag ()

def sql_logging_loop () :
    while True:
        if tagArr and 'error' not in tagArr:
            write_to_sql (tagArr)
            write_pump_activity_to_sql (tagArr)
            write_trip_status_to_sql (tagArr)
        time.sleep (5)

# -------------------- BACKGROUND LOOP --------------------
```

```
def background_loop () :
    while True:
        plc_data = read_plc_data ()  # Đọc dữ liệu từ PLC
        if plc_data and 'error' not in plc_data:
            check_trip_and_alert (plc_data)  # Kiểm tra trip bơm và
AI_Drowning, gửi email ngay nếu có
            global tagArr
            tagArr = plc_data
            for k, v in tagArr.items () :
                socketio.emit (k, v)  # Gửi dữ liệu lên giao diện
        time.sleep (1)  # Chờ 1 giây trước khi tiếp tục


@socketio.on ('connect')
def handle_connect () :
    print ("Client connected")


@socketio.on ('Client-send-data')
def handle_manual_refresh (data) :
    fn_tagRead ()
    fn_tag ()


import pandas as pd
from io import BytesIO
from flask import request, jsonify
import xlsxwriter

import pandas as pd
from io import BytesIO
from flask import request, jsonify
import xlsxwriter


@app.route ('/export_report', methods=['POST'])
def export_report () :
    data = request.get_json ()
    start = data.get ('start')
    end = data.get ('end')
    email = data.get ('email')

    try:
        # Truy vấn dữ liệu
        query1 = "SELECT * FROM plc_water_quality_data WHERE timestamp
BETWEEN %s AND %s"
        cursor.execute (query1,  (start, end) )
        df_quality = pd.DataFrame (cursor.fetchall () , columns=[desc[0]
for desc in cursor.description])

        query2 = "SELECT * FROM plc_pump WHERE timestamp_start BETWEEN %s
AND %s OR timestamp_stop BETWEEN %s AND %s"
        cursor.execute (query2,  (start, end, start, end) )
        df_pump = pd.DataFrame (cursor.fetchall () , columns=[desc[0] for
desc in cursor.description])

        query3 = "SELECT * FROM plc_pump_trip WHERE timestamp BETWEEN %s
AND %s"
        cursor.execute (query3,  (start, end) )
        df_trip = pd.DataFrame (cursor.fetchall () , columns=[desc[0] for
desc in cursor.description])
```

```
        query4 = "SELECT * FROM ai_drowning_events WHERE timestamp BETWEEN
%s AND %s"
        cursor.execute (query4,  (start, end) )
        df_ai = pd.DataFrame (cursor.fetchall () , columns=[desc[0] for
desc in cursor.description])

        # Tạo file Excel
        output = BytesIO ()
        writer = pd.ExcelWriter (output, engine='xlsxwriter')

        # Define logo image path
        logo_image_path =
'/home/thachhuynh/yolo_tp/static/public/images/logo_truong.png'

        # Function to add logo to each sheet and format
        def add_logo_and_format (sheet_name, df) :
            sheet = writer.sheets[sheet_name]

            # Add logo image to the top-left of the sheet  (keeping the
position as A1)
            sheet.insert_image ('A1', logo_image_path, {'x_scale': 0.1,
'y_scale': 0.1})

            # Write headers at row 6, not row 2
            header_format = writer.book.add_format (
                {'bold': True, 'border': 1, 'bg_color': '#f2f2f2', 'align':
'center', 'valign': 'vcenter'})
            for col_num, value in enumerate (df.columns.values) :
                sheet.write (5, col_num, value, header_format)  # Headers
at row 6  (index 5)

            # Write data starting from row 6  (index 5) to avoid overlap
with logo
            for row_num, row in df.iterrows () :
                for col_num, value in enumerate (row) :
                    sheet.write (row_num + 6, col_num, value)  # Start
writing data at row 6

            # Apply auto column width
            for col_num, value in enumerate (df.columns.values) :
                sheet.set_column (col_num, col_num, max (df[value].apply
(lambda x: len (str (x) ) ) .max () , len (value) ) + 2)

        # Write each sheet to Excel and apply logo and formatting
        df_quality.to_excel (writer, sheet_name='WaterQuality',
index=False)
        add_logo_and_format ('WaterQuality', df_quality)

        df_pump.to_excel (writer, sheet_name='PumpActivity', index=False)
        add_logo_and_format ('PumpActivity', df_pump)

        df_trip.to_excel (writer, sheet_name='PumpTrip', index=False)
        add_logo_and_format ('PumpTrip', df_trip)

        df_ai.to_excel (writer, sheet_name='DrowningAlert', index=False)
        add_logo_and_format ('DrowningAlert', df_ai)

        writer.close ()
        output.seek (0)
```

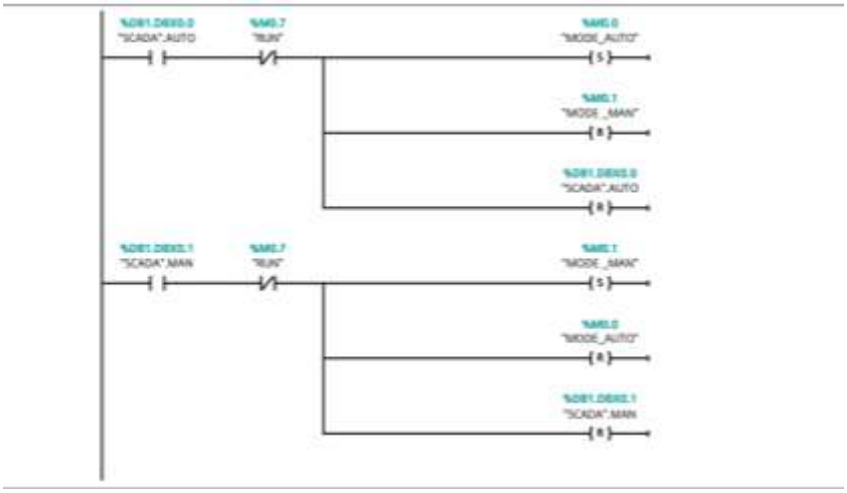
```
        # Gửi file Excel qua email
        send_email_with_attachment (
            subject="Báo cáo hồ bơi từ {} đến {}".format (start, end) ,
            body="Báo cáo hồ bơi được đính kèm theo file Excel.",
            attachment=output,
            filename="BaoCaoHoBoi.xlsx",
            to_emails=[email]
        )
        return jsonify ({"message": "Báo cáo đã được gửi email thành
công."})
    except Exception as e:
        return jsonify ({"message": f"Lỗi khi tạo/gửi báo cáo: {str (e)
}"}) , 500

@app.route ('/')
def index () :
    active_cameras = [{"id": i, "url": url} for i, url in enumerate
(camera_sources) if url]
    return render_template ("index.html", active_cameras=active_cameras,
num_cameras=len (active_cameras) )


@app.route ('/video_feed/<int:cam_id>')
def video_feed (cam_id) :
    return Response (generate_frames (cam_id) , mimetype='multipart/x-
mixed-replace; boundary=frame')


@app.route ('/api/plc')
def api_plc () :
    data = read_plc_data ()
    return jsonify (data)

@app.route ('/api/pump_history')
def get_pump_history () :
    from flask import request
    pump_name = request.args.get ('pump_name')

    query = """
        SELECT timestamp_start AS timestamp, status
        FROM plc_pump
        WHERE pump_name = %s AND timestamp_start >= NOW () - INTERVAL 6
HOUR AND timestamp_start IS NOT NULL

        UNION ALL

        SELECT timestamp_stop AS timestamp, status
        FROM plc_pump
        WHERE pump_name = %s AND timestamp_stop >= NOW () - INTERVAL 6 HOUR
AND timestamp_stop IS NOT NULL

        ORDER BY timestamp
    """
    cursor.execute (query,  (pump_name, pump_name) )
    rows = cursor.fetchall ()
    result = [{'timestamp': row[0].strftime ('%Y-%m-%d %H:%M:%S') ,
'status': row[1]} for row in rows]
    return jsonify (result)


@app.route ('/api/pump_activity_by_time')
```

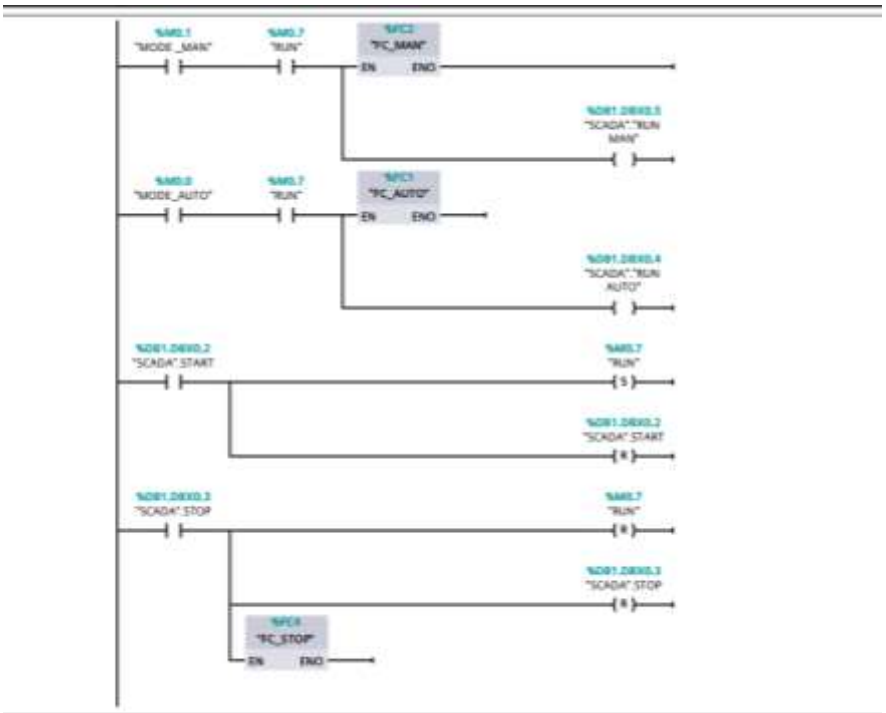
```
def pump_activity_by_time () :
    from flask import request
    start = request.args.get ('start')
    end = request.args.get ('end')

    query = """
        SELECT timestamp_start, timestamp_stop, pump_name, mode, status
        FROM plc_pump
        WHERE
             (timestamp_start BETWEEN %s AND %s)
            OR
             (timestamp_stop BETWEEN %s AND %s)
        ORDER BY COALESCE (timestamp_start, timestamp_stop)
    """
    cursor.execute (query,  (start, end, start, end) )
    rows = cursor.fetchall ()
    result = [
        {
            'start': row[0].strftime ('%Y-%m-%d %H:%M:%S') if row[0] else
'',
            'stop': row[1].strftime ('%Y-%m-%d %H:%M:%S') if row[1] else
'',
            'pump_name': row[2],
            'mode': row[3],
            'status': row[4]
        }
        for row in rows
    ]
    return jsonify (result)

@app.route ('/api/water_quality_by_time')
def water_quality_by_time () :
    start = request.args.get ('start')
    end = request.args.get ('end')

    cursor.execute ("""
        SELECT timestamp, chlorine, ph, temperature
        FROM plc_water_quality_data
        WHERE timestamp BETWEEN %s AND %s
        ORDER BY timestamp
    """,  (start, end) )

    rows = cursor.fetchall ()
    result = [
        {'timestamp': row[0].strftime ('%Y-%m-%d %H:%M:%S') , 'chlorine':
row[1], 'ph': row[2], 'temperature': row[3]}
        for row in rows
    ]

    return jsonify (result)


@app.route ('/api/ai_drowning_by_time')
def ai_drowning_by_time () :
    start = request.args.get ('start')
    end = request.args.get ('end')

    cursor.execute ("""
        SELECT timestamp, camera_id, message
        FROM ai_drowning_events
        WHERE timestamp BETWEEN %s AND %s
```

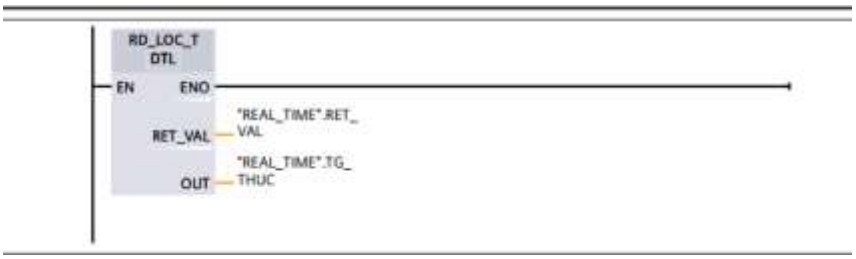
```
        ORDER BY timestamp
    """,  (start, end) )

    rows = cursor.fetchall ()
    result = [
        {'timestamp': row[0].strftime ('%Y-%m-%d %H:%M:%S') , 'camera_id':
row[1], 'message': row[2]}
        for row in rows
    ]

    return jsonify (result)


@app.route ('/api/plc_pump_trip_by_time')
def plc_pump_trip_by_time () :
    start = request.args.get ('start')
    end = request.args.get ('end')

    cursor.execute ("""
        SELECT timestamp, trip_signal, trip_status
        FROM plc_pump_trip
        WHERE timestamp BETWEEN %s AND %s
        ORDER BY timestamp
    """,  (start, end) )

    rows = cursor.fetchall ()
    result = [
        {'timestamp': row[0].strftime ('%Y-%m-%d %H:%M:%S') ,
'trip_signal': row[1], 'trip_status': row[2]}
        for row in rows
    ]

    return jsonify (result)

# -------------------- CHẠY GIAO DIỆN TRÊN HOST --------------------
if __name__ == '__main__':
    socketio.start_background_task (background_loop)
    socketio.start_background_task (sql_logging_loop)
    socketio.run (app, host='0.0.0.0', port=5000, debug=True)
```

PHỤ LỤC 2: CHƯƠNG TRÌNH ĐIỀU KHIỂN PLC

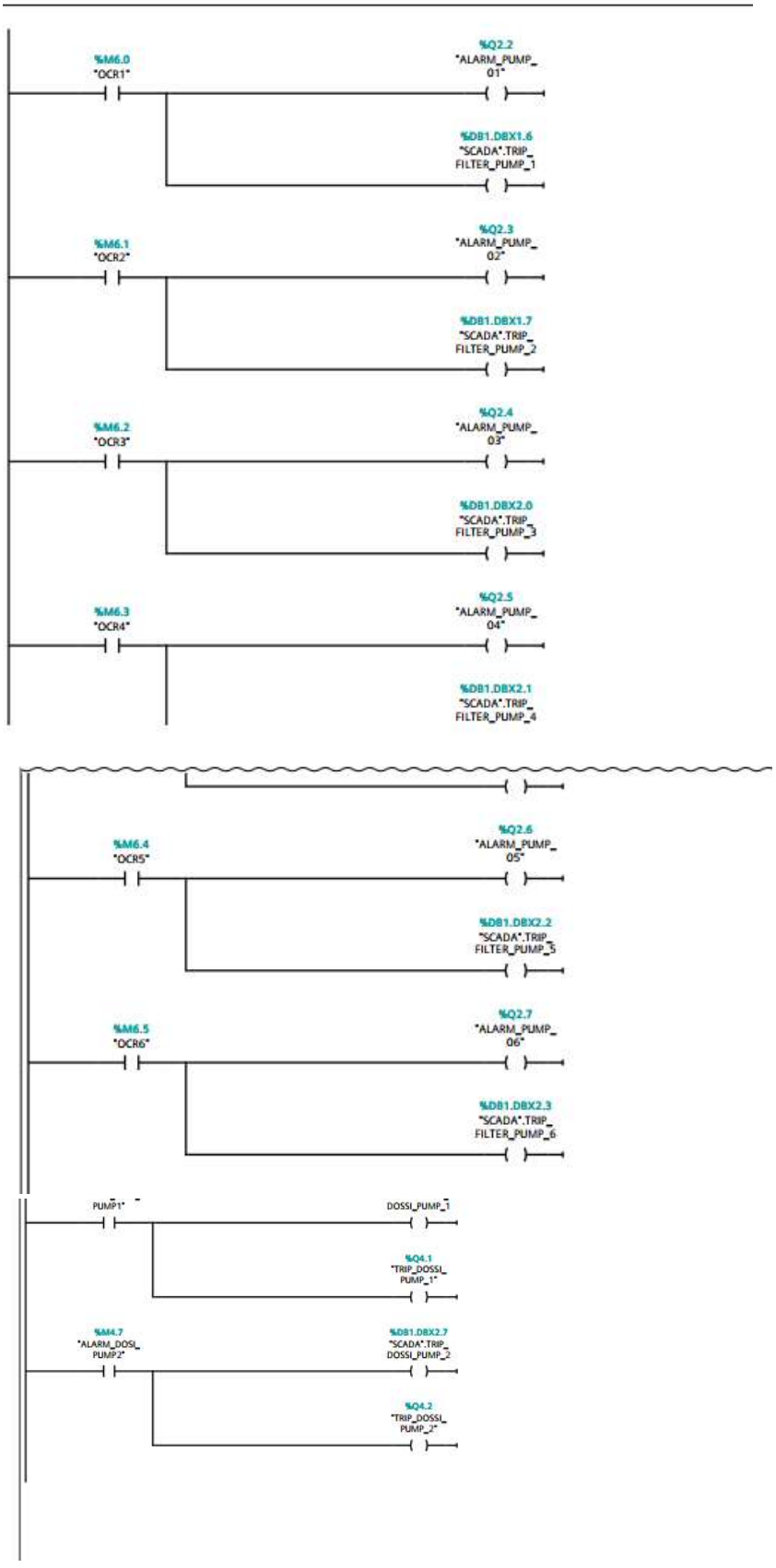
Chọn chế độ:



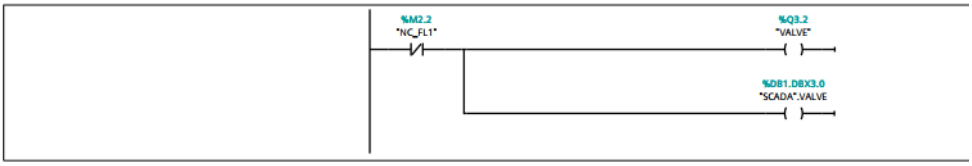




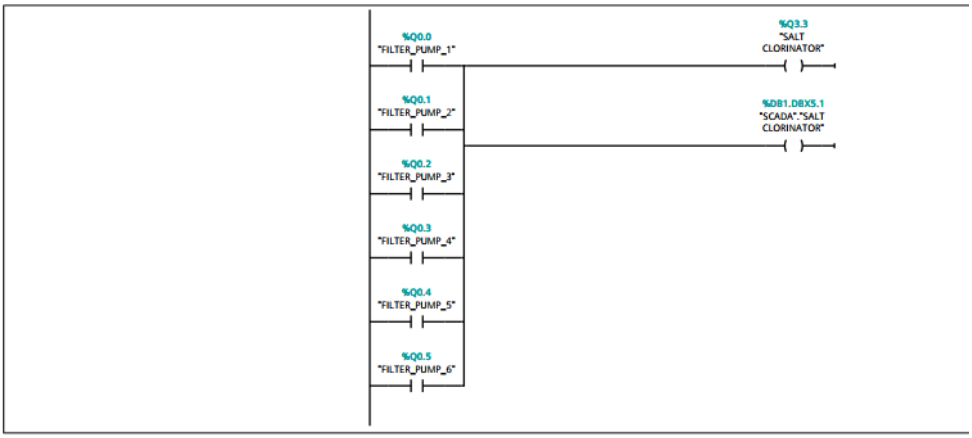
Cảnh báo lỗi:


%M6.0
"OCR1"
%Q2.2
"ALARM_PUMP_01"
%DB1.DBX1.6
"SCADA".TRIP_FILTER_PUMP_1
%M6.1
"OCR2"
%Q2.3
"ALARM_PUMP_02"
%DB1.DBX1.7
"SCADA".TRIP_FILTER_PUMP_2
%M6.2
"OCR3"
%Q2.4
"ALARM_PUMP_03"
%DB1.DBX2.0
"SCADA".TRIP_FILTER_PUMP_3
%M6.3
"OCR4"
%Q2.5
"ALARM_PUMP_04"
%DB1.DBX2.1
"SCADA".TRIP_FILTER_PUMP_4
%M6.4
"OCR5"
%Q2.6
"ALARM_PUMP_05"
%DB1.DBX2.2
"SCADA".TRIP_FILTER_PUMP_5
%M6.5
"OCR6"
%Q2.7
"ALARM_PUMP_06"
%DB1.DBX2.3
"SCADA".TRIP_FILTER_PUMP_6
PUMP1
DOSSI_PUMP_1
%Q4.1
"TRIP_DOSSI_PUMP_1"
%M4.7
"ALARM_DOSI_PUMP2"
%DB1.DBX2.7
"SCADA".TRIP_DOSSI_PUMP_2
%Q4.2
"TRIP_DOSSI_PUMP_2"

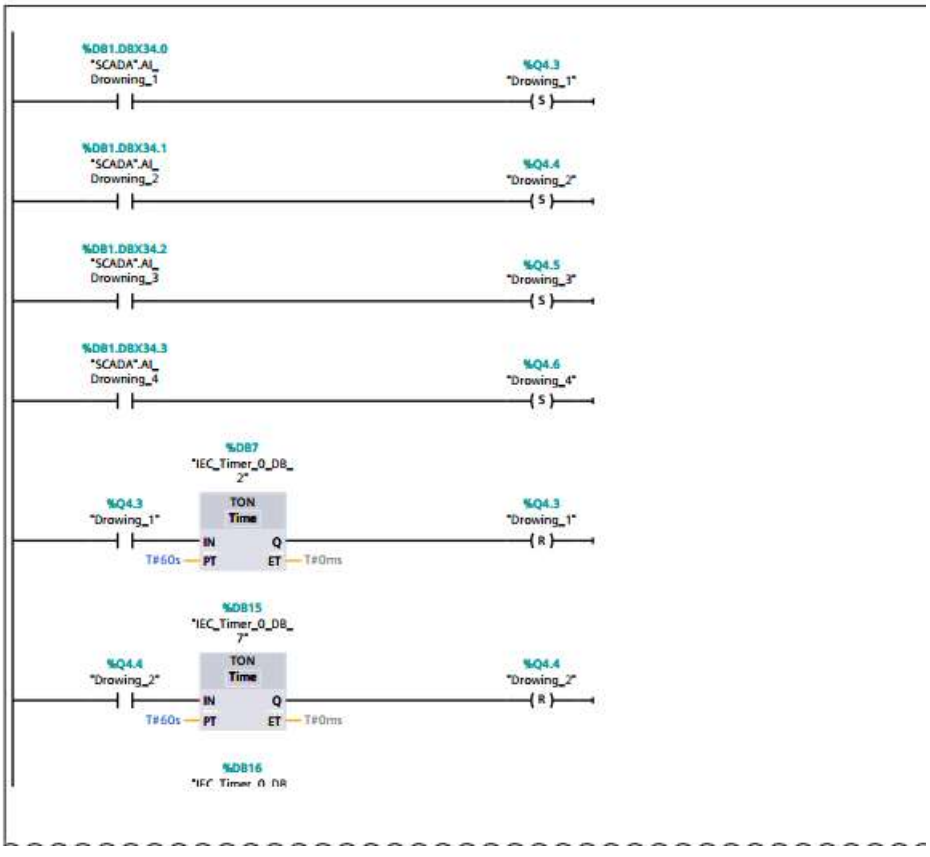
Network 5: VAN

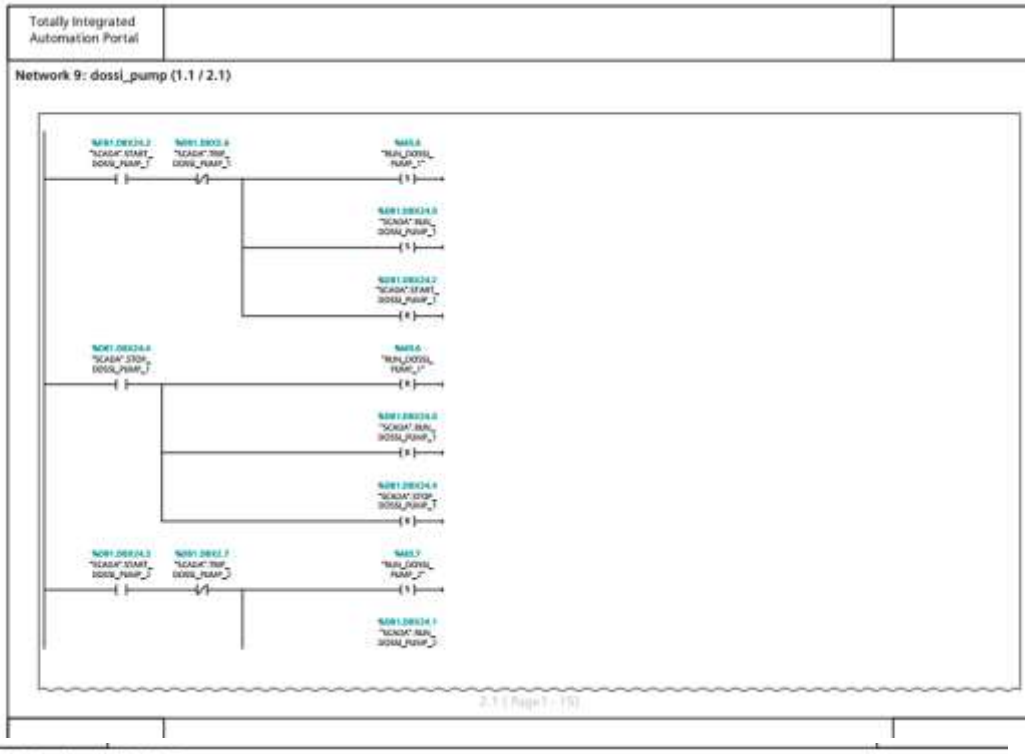


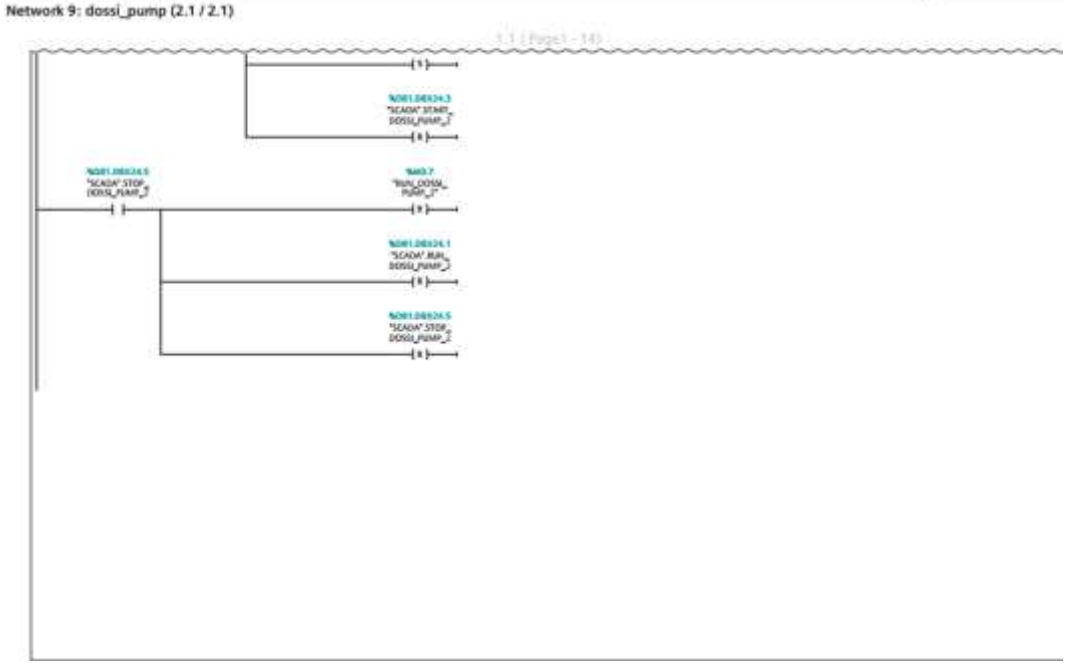
Network 6: SALT CLORINE



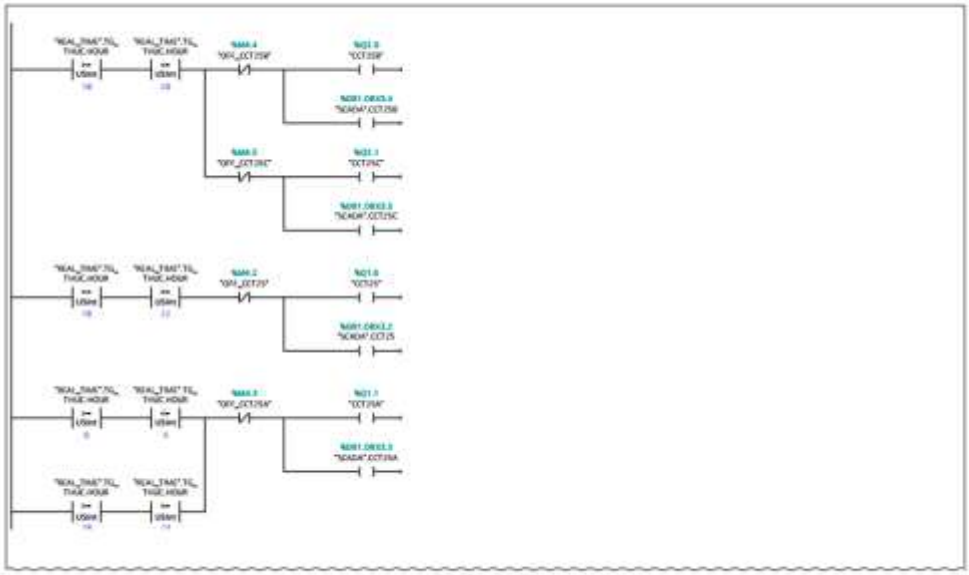
Network 7: AI (1.1 / 2.1)



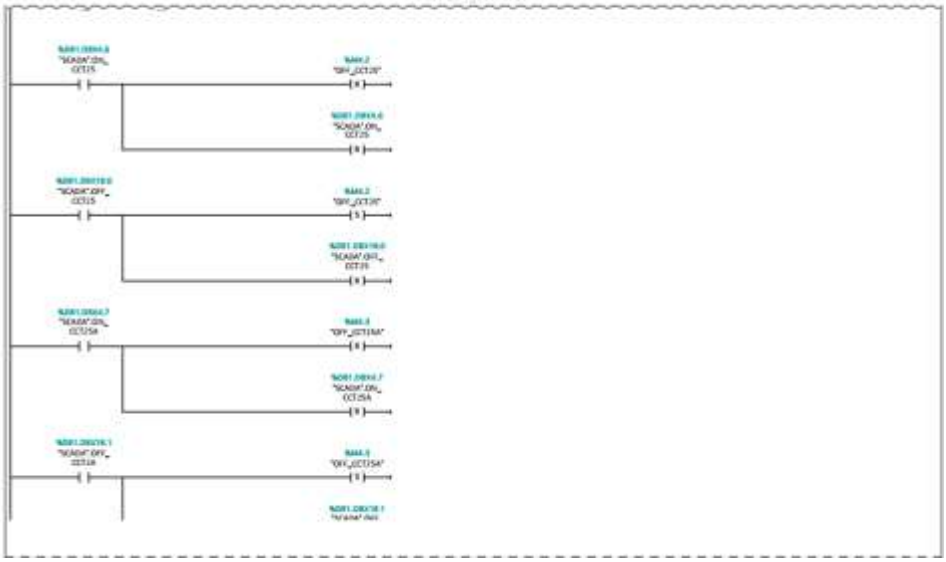

Totally Integrated
Automation Portal
Network 9: dossi_pump (1.1 / 2.1)


Network 9: dossi_pump (2.1 / 2.1)

Network 10: CHIEU_SANG (1.1 / 4.1)

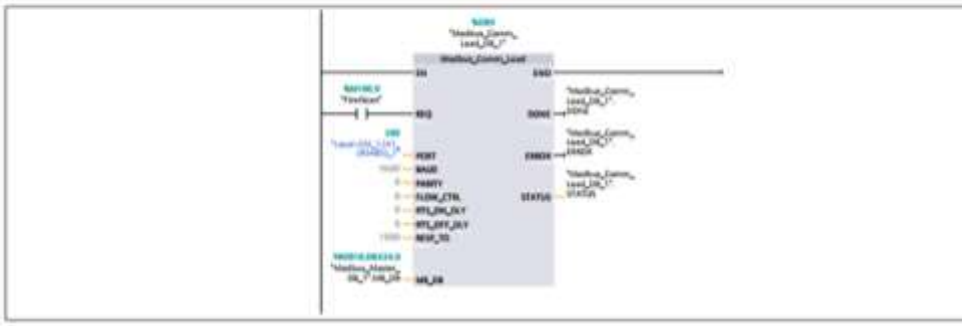


Network 10: CHIEU_SANG (2.1 / 4.1)



Network 11: HEATPUMP

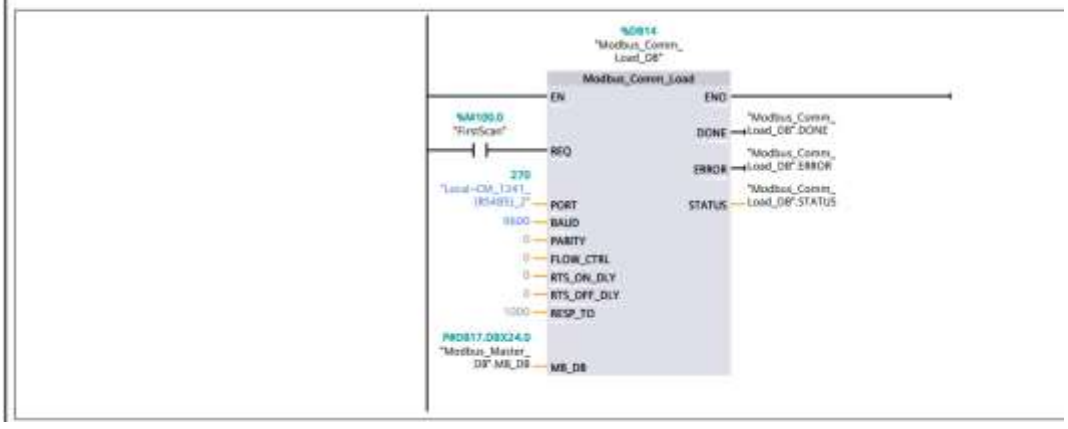
Network 13: modbus pH sensor

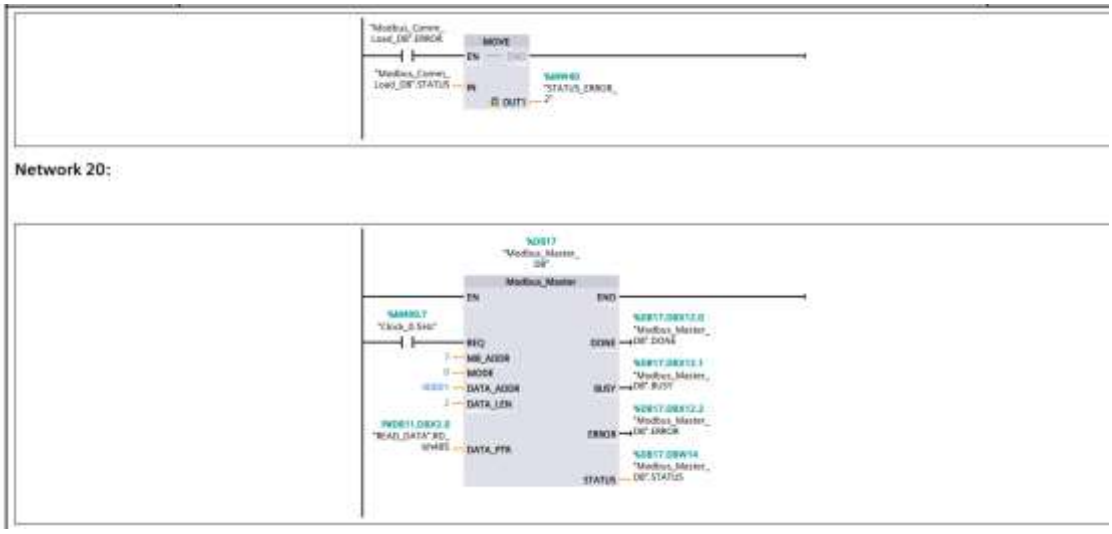


Network 14:

Network 16:

Network 18: MOBUS BH-485





Network 20:

Network 22: conv_clo_temp