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

LÊ TRƯỜNG DUY

HỆ THỐNG GIÁM SÁT VÀ TỐI ƯU HÓA SỬ DỤNG NƯỚC THÔNG MINH TRONG GIA ĐÌNH ỨNG DỤNG IOT VÀ AI

ĐỒ ÁN TỔNG HỢP
KỸ THUẬT ĐIỆN TỬ - VIỄN THÔNG

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2025

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

LÊ TRƯỜNG DUY - 42101294

HỆ THỐNG GIÁM SÁT VÀ TỐI ƯU HÓA SỬ DỤNG NƯỚC THÔNG MINH TRONG GIA ĐÌNH ỨNG DỤNG IOT VÀ AI

ĐỒ ÁN TỔNG HỢP
KỸ THUẬT ĐIỆN TỬ - VIỄN THÔNG

Người hướng dẫn
TS.TRẦN CÔNG THỊNH
GS.KIM DOKYONG

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2025

LỜI CẢM ƠN

Đồ án tổng hợp là đồ án cuối cùng và cũng là một cột mốc quan trọng đánh dấu chặng đường 4 năm học tập và rèn luyện của em tại trường Đại học. Đây không chỉ là cơ hội để em tổng hợp, áp dụng kiến thức chuyên môn đã tích lũy mà còn là thử thách lớn để em khẳng định sự trưởng thành trong tư duy và kỹ năng làm việc.

Để hoàn thành được đồ án này, em đã nhận được sự hỗ trợ quý báu từ quý Thầy Cô khoa Điện – Điện tử. Trước tiên, em xin gửi lời cảm ơn chân thành đến các Thầy Cô trong khoa, đặc biệt là các Thầy Cô giảng viên bộ môn Kỹ thuật điện tử – viễn thông. Nhờ có sự hướng dẫn tận tình và kiến thức chuyên môn vững vàng của quý Thầy Cô, em đã có thể hoàn thành tốt đồ án "Hệ thống giám sát và tối ưu hóa sử dụng nước thông minh trong gia đình ứng dụng IoT và AI."

Đặc biệt, em xin gửi lời cảm ơn sâu sắc đến TS.Trần Công Thịnh và GS.Kim Dokyong, người đã luôn tận tình chỉ bảo, góp ý và đồng hành cùng em trong suốt quá trình thực hiện đề tài. Sự hỗ trợ của hai Thầy đã giúp em vượt qua nhiều khó khăn và hoàn thành tốt nội dung nghiên cứu.

Mặc dù đã nỗ lực hết sức, nhưng do vẫn còn hạn chế về kinh nghiệm thực tiễn và thời gian thực hiện, em không tránh khỏi những thiếu sót trong quá trình nghiên cứu và triển khai đề tài. Em rất mong nhận được những ý kiến đóng góp từ quý Thầy Cô để em có thể rút kinh nghiệm và tiếp tục hoàn thiện bản thân trong chặng đường phía trước.

Một lần nữa, em xin chân thành cảm ơn quý Thầy Cô đã luôn tận tâm, đồng hành và tạo điều kiện cho em trong suốt quá trình học tập và thực hiện đồ án này.

TP. Hồ Chí Minh, ngày 15 tháng 7 năm 2025

Tác giả

Lê Trường Duy

Công trình được hoàn thành tại Trường Đại học Tôn Đức Thắng

Cán bộ hướng dẫn khoa học:TS. Trần Công Thịnh

GS. Kim DoKyong

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

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

CHỦ TỊCH HỘI ĐỒNG	**TRƯỞNG KHOA**
……..	……

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

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

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

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

TP. Hồ Chí Minh, ngày 15 tháng 7 năm 2025

Tác giả

(ký tên và ghi rõ họ tên)

Lê Trường Duy

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

HỆ THỐNG GIÁM SÁT VÀ TỐI ƯU HÓA SỬ DỤNG NƯỚC THÔNG MINH TRONG GIA ĐÌNH ỨNG DỤNG IOT VÀ AI

TÓM TẮT

Hiện nay, nhu cầu sử dụng nước và điện trong sinh hoạt gia đình ngày càng tăng, trong khi việc giám sát và tối ưu hóa vẫn còn nhiều hạn chế. Vì vậy, hệ thống này được phát triển nhằm nâng cao hiệu quả quản lý, tiết kiệm tài nguyên và đảm bảo vận hành thông minh cho người dùng.

Quá trình hoạt động của hệ thống bao gồm việc sử dụng ESP32 để thu thập dữ liệu từ các cảm biến như lưu lượng nước, dòng điện, hồng ngoại… Sau đó, các dữ liệu này được gửi lên máy chủ thông qua giao thức MQTT và lưu trữ trên nền tảng InfluxDB, hiển thị bằng Grafana. Người dùng có thể truy cập vào giao diện web trên nền Home Assistant để theo dõi lượng nước tiêu thụ, điện năng sử dụng, cảnh báo rò rỉ cũng như lịch bảo trì thiết bị lọc.

Ngoài ra, hệ thống còn tích hợp trí tuệ nhân tạo (AI) để phân tích xu hướng sử dụng nước theo ngày, tuần, tháng, dự đoán thời điểm cần thay lõi lọc và tự động bật tắt van điện từ để tránh thất thoát tài nguyên khi không sử dụng. Hệ thống cũng được triển khai bằng Docker trên máy chủ Ubuntu nhằm đảm bảo tính linh hoạt và dễ dàng bảo trì.

Giải pháp này không chỉ đơn giản hóa việc giám sát tiêu thụ nước và điện, mà còn giúp tối ưu hóa hoạt động của các thiết bị, nâng cao hiệu quả vận hành và đảm bảo an toàn cho người sử dụng.

SMART WATER MONITORING AND OPTIMIZATION SYSTEM IN HOUSEHOLDS BASED ON IOT AND AI

ABSTRACT

Currently, the demand for water and electricity in household activities is increasing, while monitoring and optimization are still limited. Therefore, this system was developed to improve management efficiency, save resources and ensure smart operation for users.

The system's operation includes using ESP32 to collect data from sensors such as water flow, current, infrared, etc. Then, this data is sent to the server via MQTT protocol and stored on the InfluxDB platform, displayed using Grafana. Users can access the web interface on the Home Assistant platform to monitor water consumption, electricity usage, leak warnings as well as filter maintenance schedules.

In addition, the system also integrates artificial intelligence (AI) to analyze water usage trends by day, week, month, predict when to change the filter core and automatically turn on and off the solenoid valve to avoid resource loss when not in use. The system is also deployed using Docker on Ubuntu servers to ensure flexibility and ease of maintenance. This solution not only simplifies water and electricity consumption monitoring, but also helps optimize equipment performance, improve operational efficiency and ensure user safety.

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

DANH MỤC CÁC CHỮ VIẾT TẮT

IoT	Internet of Things
AI	Artificial Intelligence
MQTT	Message Queuing Telemetry Transport
TLS	Transport Layer Security
SSL	Secure Sockets Layer
TCP/IP	Transmission Control Protocol / Internet Protocol
HTTP	HyperText Transfer Protocol
OLED	Organic Light Emitting Diode
LCD TFT	Liquid Crystal Display - Thin Film Transistor
NLP	Natural Language Processing
LSTM	Long Short-Term Memory
ARIMA	AutoRegressive Integrated Moving Average
QoS	Quality of Service
JSON	JavaScript Object Notation
GUI	Graphical User Interface
TSDB	Time Series Database
API	Application Programming Interface
CLI	Command Line Interface
IDE	Integrated Development Environment
NTP	Network Time Protocol

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Mục tiêu đề tài

Đề tài "Phát triển hệ thống giám sát và tối ưu hóa tiêu thụ nước thông minh trong hộ gia đình ứng dụng IoT và AI" tập trung vào việc tạo ra một giải pháp toàn diện, tự động hóa cao và dễ thích ứng, nhằm quản lý hiệu quả việc sử dụng nước và điện trong đời sống hàng ngày. Hệ thống cho phép giám sát liên tục các thông số quan trọng như lưu lượng nước, mức điện năng tiêu thụ và trạng thái hoạt động của các thiết bị (ví dụ: van điện tử, máy bơm…), đảm bảo độ chính xác cao. Nhờ vậy, việc ghi chép thủ công truyền thống được loại bỏ hoàn toàn, thay vào đó là dữ liệu được hiển thị một cách trực quan và cập nhật theo thời gian thực cho người sử dụng. Không chỉ dừng lại ở việc thu thập dữ liệu, hệ thống còn tích hợp các thuật toán học máy như hồi quy tuyến tính, ARIMA hay LSTM để phân tích hành vi sử dụng theo nhiều mốc thời gian khác nhau. Nhờ đó, hệ thống có thể phát hiện kịp thời các bất thường như rò rỉ nước, tiêu thụ điện tăng đột ngột hoặc thiết bị vận hành sai công suất, góp phần nâng cao hiệu quả quản lý và cảnh báo sớm các sự cố. Kết quả phân tích được trình bày qua các biểu đồ và báo cáo dễ hiểu, giúp người dùng dễ dàng giám sát và điều chỉnh thói quen sử dụng tài nguyên nước điện một cách hợp lý. Dựa trên các mức ngưỡng được thiết lập sẵn hoặc do AI tính toán, hệ thống sẽ tự động can thiệp vào hoạt động của thiết bị, ví dụ như bật/tắt van điện tử, nhằm đảm bảo hiệu suất tối ưu và tiết kiệm nguồn lực. Cách tiếp cận này không chỉ giảm thiểu chi phí điện nước mà còn giúp tăng cường độ bền cho các thiết bị cơ khí. Giải pháp được xây dựng theo kiến trúc microservices kết hợp công nghệ container hóa Docker, mang lại khả năng mở rộng linh hoạt. Nhờ đó, việc tích hợp thêm cảm biến mới hoặc thay đổi phần cứng có thể thực hiện dễ dàng mà không ảnh hưởng đến toàn bộ hệ thống. Thiết kế này đặc biệt phù hợp cho cả quy mô hộ gia đình nhỏ lẫn các công trình lớn như tòa nhà thông minh. Giao diện quản lý được phát triển trên nền tảng Home Assistant, cho phép truy cập qua trình duyệt web hoặc ứng dụng di

động, mang đến cho người dùng các công cụ theo dõi trực quan, nhận thông báo kịp thời và tùy chỉnh theo sở thích cá nhân một cách đơn giản. Cuối cùng, đề tài còn thực hiện đánh giá mức độ tiết kiệm nước và điện trước và sau khi áp dụng hệ thống, đồng thời so sánh hiệu suất giữa các ngưỡng điều chỉnh thủ công và tự động từ AI. Các kết quả thực nghiệm này sẽ làm nền tảng để đề xuất các cải tiến cho giải pháp và mở rộng triển khai thực tế, hướng tới việc phổ biến rộng rãi trong các khu dân cư hiện đại.

1.2 Yêu cầu của đề tài

Đề tài "Hệ thống giám sát và tối ưu hóa sử dụng nước thông minh trong hộ gia đình dựa trên IoT và trí tuệ nhân tạo" yêu cầu sự kết hợp chặt chẽ giữa phần cứng, phần mềm, hệ thống truyền thông và các giải pháp thuật toán thông minh. Điều này nhằm đảm bảo tính ổn định, độ chính xác và bảo mật trong quá trình hoạt động.

Về phần cứng, hệ thống sử dụng vi điều khiển ESP32 nhờ khả năng kết nối Wi-Fi ổn định và hiệu năng cao. Thiết bị này chịu trách nhiệm thu thập và truyền dữ liệu từ các cảm biến đo lưu lượng nước, cảm biến dòng điện, cũng như cảm biến hồng ngoại giúp nhận diện sự hiện diện của người dùng. Các thiết bị chấp hành như van điện từ và máy bơm được lựa chọn dựa trên công suất thực tế, đảm bảo hoạt động liên tục mà không bị quá tải hoặc hao mòn nhanh chóng. Để thuận tiện cho người dùng trong việc theo dõi, hệ thống còn được trang bị màn hình hiển thị nhỏ gọn như OLED hoặc LCD TFT, cho phép hiển thị các thông số quan trọng ngay tại vị trí lắp đặt mà không cần truy cập vào nền tảng web.

Ở tầng phần mềm và hệ thống backend, MQTT Broker (sử dụng HiveMQ) được triển khai trong môi trường Docker, nhằm đảm bảo khả năng xử lý lượng lớn dữ liệu từ cảm biến với độ trễ dưới 500 ms. Giao diện người dùng được thiết kế trên nền tảng Home Assistant, mang lại khả năng tùy chỉnh linh hoạt và phân quyền rõ ràng giữa người dùng thường và quản trị viên. Hệ thống cũng cho phép tích hợp trực tiếp các nút điều khiển thông qua Node-RED, đóng vai trò như cầu nối xử lý logic giữa dữ liệu MQTT và các hành động phản hồi: tiếp nhận dữ liệu, áp dụng quy

tắc thủ công hoặc ngưỡng dựa trên AI, và phát lệnh điều khiển đến thiết bị. Dữ liệu được lưu trữ dưới dạng chuỗi thời gian trong InfluxDB, hỗ trợ nén và truy vấn hiệu quả, đồng thời kết nối trực tiếp với Grafana để trình bày dữ liệu dưới dạng dashboard trực quan, phục vụ cho việc theo dõi và phân tích lâu dài.

Về mặt bảo mật và độ tin cậy, hệ thống áp dụng mã hóa TLS cho toàn bộ quá trình truyền tải qua giao thức MQTT nhằm ngăn chặn các cuộc tấn công từ chối dịch vụ và hành vi nghe lén. Các API giao tiếp với mô-đun AI cũng hoạt động trên nền tảng HTTPS và yêu cầu xác thực token để bảo vệ thông tin. Hơn nữa, ESP32 được cấu hình để lưu trữ tạm thời dữ liệu trong bộ nhớ nội trong ít nhất một giờ nếu mất kết nối Internet, và tự động đồng bộ khi mạng được phục hồi. Việc sử dụng Docker Compose trên nền Ubuntu Server cho phép triển khai linh hoạt và dễ dàng mở rộng hệ thống, từ việc thêm thiết bị ESP32 đến triển khai thêm các dịch vụ AI mà không gây gián đoạn. Khả năng khởi động lại, nâng cấp hoặc quay lại phiên bản trước (rollback) cũng được tối ưu hóa, đảm bảo vận hành ổn định trong môi trường sản xuất.

Cuối cùng, đề tài yêu cầu thực hiện quy trình thử nghiệm kỹ lưỡng, bao gồm hiệu chỉnh cảm biến, mô phỏng các tình huống bất thường như rò rỉ hoặc tăng đột biến lưu lượng, và đánh giá hiệu suất của MQTT dưới điều kiện tải cao. Ngoài ra, việc so sánh giữa điều khiển truyền thống và điều khiển theo ngưỡng động do AI đề xuất sẽ cung cấp số liệu định lượng về mức tiết kiệm tài nguyên. Những kết quả này sẽ là cơ sở để tinh chỉnh thuật toán, tối ưu hóa hệ thống và hoàn thiện báo cáo nghiên cứu trước khi triển khai thử nghiệm thực tế trên diện rộng.

1.3 Ý tưởng thực hiện

Hệ thống được xây dựng theo mô hình ba lớp phân tầng rõ rệt, giúp tổ chức, mở rộng và bảo trì các thành phần một cách độc lập. Lớp đầu tiên, gọi là lớp thiết bị ngoại vi (Edge Layer), bao gồm các mô-đun ESP32 kết nối trực tiếp với các cảm biến chuyên dụng như lưu lượng kế, cảm biến dòng điện và cảm biến phát hiện.

Nhiệm vụ chính của lớp này là nhận tín hiệu thô từ môi trường, tiến hành lọc và xử lý sơ bộ dữ liệu, sau đó truyền thông tin dưới dạng JSON qua giao thức MQTT.

Lớp thứ hai, được gọi là lớp nền tảng xử lý (Platform Layer), đóng vai trò như trung tâm điều phối và xử lý dữ liệu. Lớp này tích hợp các thành phần quan trọng như MQTT Broker để phân phối thông điệp, Node-RED để xử lý logic điều khiển, InfluxDB để lưu trữ dữ liệu theo thời gian, và Grafana để hiển thị dữ liệu một cách trực quan theo thời gian thực.

Lớp thứ ba là lớp ứng dụng và dịch vụ thông minh (Application Layer), nơi diễn ra sự tương tác trực tiếp giữa người dùng và hệ thống. Tại đây, Home Assistant đóng vai trò là giao diện điều khiển trung tâm, hỗ trợ người dùng giám sát trạng thái thiết bị, thiết lập các ngưỡng cảnh báo và thực hiện điều khiển thủ công một cách thuận tiện. Dịch vụ AI tích hợp còn có chức năng phân tích dữ liệu lịch sử, học hỏi thói quen sử dụng và tự động đề xuất điều chỉnh ngưỡng điều khiển phù hợp với nhu cầu của từng hộ gia đình.

Quá trình xử lý dữ liệu được thiết kế theo chu trình khép kín và đồng bộ. Sau khi ESP32 thu thập dữ liệu từ cảm biến, tín hiệu sẽ được xử lý và truyền đi thông qua MQTT. Khi bản tin đến broker, Node-RED sẽ đảm nhiệm hai nhiệm vụ chính: ghi dữ liệu vào InfluxDB để phục vụ cho việc theo dõi và phân tích dài hạn, và so sánh với các ngưỡng đã được thiết lập (có thể là tĩnh hoặc động). Nếu phát hiện dữ liệu vượt ngưỡng, Node-RED sẽ lập tức gửi lệnh điều khiển trở lại ESP32 để thực hiện các thao tác như đóng/mở van nước hoặc điều chỉnh máy bơm. Dữ liệu và trạng thái thiết bị sẽ được cập nhật ngay lập tức lên Grafana và Home Assistant, cho phép người dùng theo dõi tình hình theo thời gian thực.

Vào cuối mỗi chu kỳ (ví dụ hàng ngày), hệ thống AI sẽ truy xuất dữ liệu đã thu thập, thực hiện phân tích mô hình tiêu thụ theo thời gian và tính toán lại ngưỡng tối ưu thông qua thuật toán học máy. Ngưỡng mới sẽ được tự động cập nhật vào Node-RED mà không cần sự can thiệp thủ công, giúp hệ thống tự động điều chỉnh theo nhu cầu sử dụng nước và điện của từng gia đình.

1.4 Phương pháp thực hiện

Quá trình triển khai hệ thống bắt đầu bằng việc thực hiện khảo sát thực tế. Sau đó, tiến hành lắp ráp phần cứng, kết nối các thiết bị và kiểm tra tín hiệu để đảm bảo hoạt động ổn định. Vi điều khiển ESP32 được lập trình trên nền tảng Arduino IDE, với các chức năng chính như đọc dữ liệu từ cảm biến, lọc nhiễu, truyền và nhận dữ liệu qua giao thức MQTT, cũng như thực hiện các lệnh điều khiển.

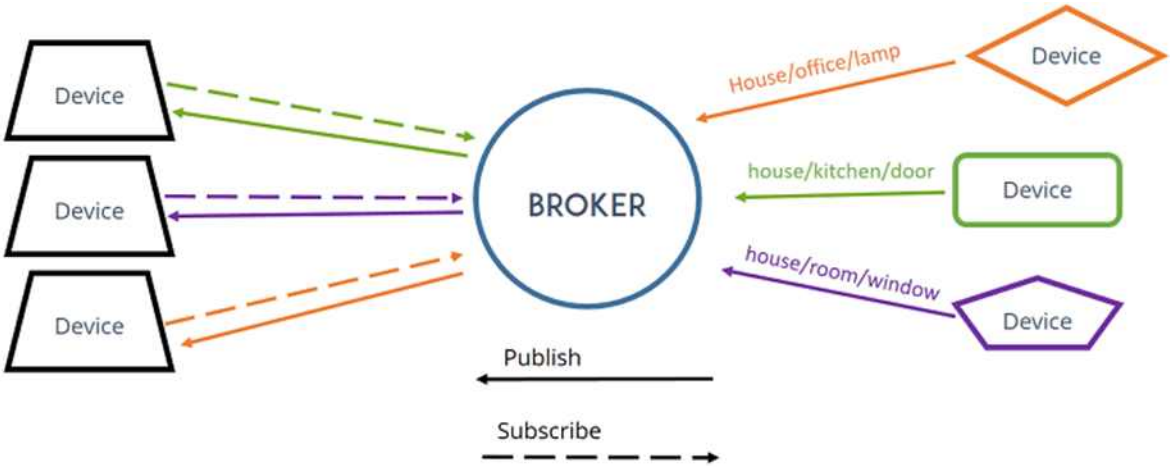
Trên máy chủ, sử dụng Docker để triển khai các dịch vụ như InfluxDB, Grafana, Node-RED, Home Assistant và AI-service thông qua tệp cấu hình docker-compose.yml. Các container được thiết lập đầy đủ về mạng và volume để đảm bảo dữ liệu không bị mất khi khởi động lại. Trong Node-RED, xây dựng luồng xử lý để nhận dữ liệu từ MQTT, lưu vào InfluxDB, kiểm tra ngưỡng và điều khiển thiết bị.

AI-service được phát triển bằng Python, đảm nhiệm việc truy xuất dữ liệu, phân tích và trả về ngưỡng dự đoán qua API /predictThreshold. Home Assistant được sử dụng làm giao diện chính, cho phép theo dõi dữ liệu, điều khiển thiết bị và thiết lập cảnh báo.

Cuối cùng, hệ thống được kiểm thử bằng cách mô phỏng các sự cố như rò rỉ nước, tăng đột biến lưu lượng hoặc mất kết nối, nhằm hiệu chỉnh các tham số và đánh giá khả năng tiết kiệm so với phương pháp vận hành thủ công. Qua đó, hoàn thiện hệ thống để chuẩn bị cho việc triển khai thực tế.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1 Giao thức MQTT



Hình 2.1 Giao thức MQTT

MQTT (Message Queuing Telemetry Transport) là một giao thức nhắn tin nhẹ, được chuẩn hóa bởi OASIS, được thiết kế đặc biệt cho các ứng dụng Internet Vật (IoT). Giao thức này sử dụng mô hình xuất bản/đăng ký (publish/subscribe) để truyền dữ liệu, phù hợp hoặc trong môi trường băng thông thấp. Hiện nay, MQTT đã được sử dụng trong nhiều lĩnh vực như công nghiệp, viễn thông và dầu khí, nơi cần truyền tải thông tin hiệu quả với độ trễ thấp.

Giao thức này mang lại nhiều lợi ích, giúp tiết kiệm đáng kể băng thông và tài nguyên hệ thống, đồng thời tạo điều kiện cho các thiết bị tương tác linh hoạt với nhau. Với những ưu điểm đó, MQTT đã trở thành sự lựa chọn hàng đầu cho các ứng dụng IoT cần khả năng truyền tải dữ liệu ổn định và tối ưu hóa.

2.1.1 Tính năng nổi bật

Cơ chế truyền thông theo mô hình xuất bản/đăng ký (Publish/Subscribe) là một phương thức truyền tin một chiều và tách biệt hoàn toàn khỏi các ứng dụng. Điều này có nghĩa là phần trong hệ thống duy trì

linh hoạt cao hơn. Việc trao đổi thông điệp diễn ra gần như tức thời, không phụ thuộc vào nội dung dữ liệu, giúp đảm bảo tốc độ và hiệu quả truyền thông. Mô hình này dựa trên giao thức TCP/IP, kết hợp với cơ chế đóng gói dữ liệu gọn nhẹ, nhằm giảm tải đường truyền và tối ưu hiệu suất toàn hệ thống.

Về chất lượng dịch vụ (QoS: Quality of Service) cho việc truyền dữ liệu, có ba cấp độ tin cậy được hỗ trợ. Ở mức QoS 0, bên gửi chỉ truyền dữ liệu một lần duy nhất, không yêu cầu xác nhận từ phía nhận, và dựa vào độ tin cậy của giao thức TCP/IP. Trong khi đó, mức QoS 1 đảm bảo dữ liệu được gửi ít nhất một lần và bắt buộc phải có xác nhận từ bên nhận, dù có thể dẫn đến việc nhận trùng lặp thông điệp. Cuối cùng, mức QoS 2 đảm bảo độ tin cậy cao nhất: khi dữ liệu được gửi, phía nhận sẽ chỉ nhận đúng một lần duy nhất, yêu cầu một cơ chế bắt tay bốn bước phức tạp để xác nhận.

Bảng 2.1Ưu điểm của giao thức MQTT

STT	Ưu điểm
1	Truyền thông tin hiệu quả hơn
2	Tăng khả năng mở rộng
3	Giảm đáng kể tiêu thụ băng thông mạng.
4	Rất phù hợp cho điều khiển và do thám
5	Tối đa hóa băng thông có sẵn
6	Chi phí thấp.
7	Rất an toàn, bảo mật
8	Được sử dụng trong các ngành công nghiệp dầu khí, các công ty lớn như Amazon, Facebook, ….
9	Tiết kiệm thời gian phát triển
10	Giao thức publish/subscribe thu thập nhiều dữ liệu hơn và tốn ít băng thông hơn so với giao thức cũ

2.1.2 Mô hình Publish/Subscriber trong giao thức MQTT

MQTT Broker có thể được triển khai theo nhiều cách khác nhau. Người dùng có thể tự cài đặt và vận hành Broker riêng của mình thông qua các mã nguồn mở hoặc các phiên bản thương mại có sẵn. Một lựa chọn phổ biến khác là sử dụng các dịch vụ Broker trên nền tảng điện toán đám mây.

Các MQTT Client là những ứng dụng hoặc thiết bị thực hiện kết nối tới Broker để trao đổi dữ liệu. Các Client được phân thành hai vai trò chính: Publisher (người gửi) và Subscriber (người nhận). Một Client có thể đảm nhiệm một trong hai vai trò hoặc cả hai đồng thời. Publisher có nhiệm vụ gửi các bản tin tới Broker, trong khi Subscriber sẽ nhận các bản tin mới ngay khi chúng được gửi lên Broker. Để dễ hình dung, mô hình Pub/Sub giống như hoạt động của một nhà môi giới bất động sản: người bán (Publisher) cung cấp thông tin về lô đất và giá cả cho môi giới (Broker). Môi giới sau đó chuyển tiếp thông tin này đến người mua (Subscriber) – người đã đăng ký nhận thông báo về đất tại khu vực cụ thể. Nhờ đó, người mua và người bán có thể giao dịch gián tiếp mà không cần liên hệ trực tiếp.

Mô hình này sở hữu ba đặc tính quan trọng: tách biệt về không gian (Space Decoupling) – các thành phần không cần biết vị trí của nhau; tách biệt về thời gian (Time Decoupling) – người gửi và người nhận không cần hoạt động đồng thời; và tách biệt về đồng bộ hóa (Synchronization Decoupling) – giúp các bên hoạt động độc lập, không phụ thuộc vào tiến trình xử lý của nhau.

MQTT cũng sử dụng cơ chế lọc thông điệp dựa trên tiêu đề (subject-based), cùng với một lớp chất lượng dịch vụ (Quality of Service – QoS) giúp dễ dàng xác định xem thông điệp đã được truyền thành công hay chưa.

2.1.3 Cách thức hoạt động

Một phiên MQTT bao gồm bốn giai đoạn chính: khởi tạo kết nối, xác thực, trao đổi dữ liệu và đóng kết nối.

Kết nối được khởi tạo khi máy khách mở một kết nối TCP/IP đến broker. Theo mặc định, cổng 1883 được dùng cho giao tiếp không mã hóa, còn cổng 8883 dành cho

các kết nối bảo mật thông qua SSL hoặc TLS. Trong quá trình này, máy khách có thể gửi thông tin xác thực hoặc ID phiên để tiếp tục một phiên làm việc trước đó.
Xác thực không bắt buộc nhưng được nhiều broker hỗ trợ nhằm tăng cường bảo mật. Việc sử dụng tên người dùng và mật khẩu, kết hợp với mã hóa SSL/TLS, giúp ngăn chặn truy cập trái phép và đảm bảo dữ liệu không bị can thiệp trong quá trình truyền tải.
Giao tiếp: Tại giai đoạn này, máy khách có thể thực hiện các thao tác như publish (gửi dữ liệu), subscribe (đăng ký nhận dữ liệu), unsubscribe (hủy đăng ký), và gửi ping để giữ kết nối luôn hoạt động. Dữ liệu gửi qua MQTT có thể lên tới 256 MB, với định dạng linh hoạt theo nhu cầu sử dụng. Khi thực hiện publish, dữ liệu nhị phân sẽ được gửi đến một chủ đề cụ thể. Việc subscribe giúp máy khách nhận dữ liệu từ các chủ đề đã đăng ký trước đó. Chủ đề trong MQTT có cấu trúc dạng cây, phân cấp bằng dấu gạch chéo (/). MQTT cho phép sử dụng ký tự đại diện như dấu cộng (+) cho một cấp, và dấu thăng (#) cho nhiều cấp chủ đề, giúp đơn giản hóa việc quản lý đăng ký. Lệnh ping (PINGREQ/PINGRESP) được dùng để kiểm tra và duy trì trạng thái hoạt động của kết nối TCP, tránh bị timeout.
Khi máy khách muốn ngắt kết nối, nó sẽ gửi lệnh DISCONNECT đến broker để kết thúc phiên một cách hợp lệ, giúp quản lý tài nguyên hiệu quả và thuận tiện cho việc tái kết nối sau này. Nếu xảy ra mất kết nối đột ngột, broker có thể tự động gửi thông điệp di chúc (will message) đến các subscriber, nhằm thông báo tình trạng từ publisher đã ngắt kết nối..

2.1.4 Ứng dụng MQTT trong lĩnh vực IoT

Ứng dụng của MQTT trong IoT rất phổ biến nhờ vào khả năng truyền dữ liệu nhẹ, hiệu quả và đáng tin cậy trong các môi trường hạn chế về băng thông hoặc nguồn điện. Giao thức MQTT được sử dụng rộng rãi trong các hệ thống nhà thông minh, giám sát môi trường, theo dõi sức khỏe, điều khiển công nghiệp và các thiết bị IoT như cảm biến, thiết bị đo đạc và bộ điều khiển. Nhờ vào mô hình pub/sub và khả năng duy trì kết nối ổn định giữa các thiết bị với server trung tâm (Broker), MQTT

giúp các thiết bị có thể gửi và nhận dữ liệu theo thời gian thực. Ngoài ra, việc hỗ trợ nhiều mức chất lượng dịch vụ (QoS) và khả năng mã hóa bảo mật bằng SSL/TLS giúp MQTT trở thành lựa chọn lý tưởng cho các ứng dụng IoT cần độ tin cậy và bảo mật cao.

2.2 Giao thức HTTP

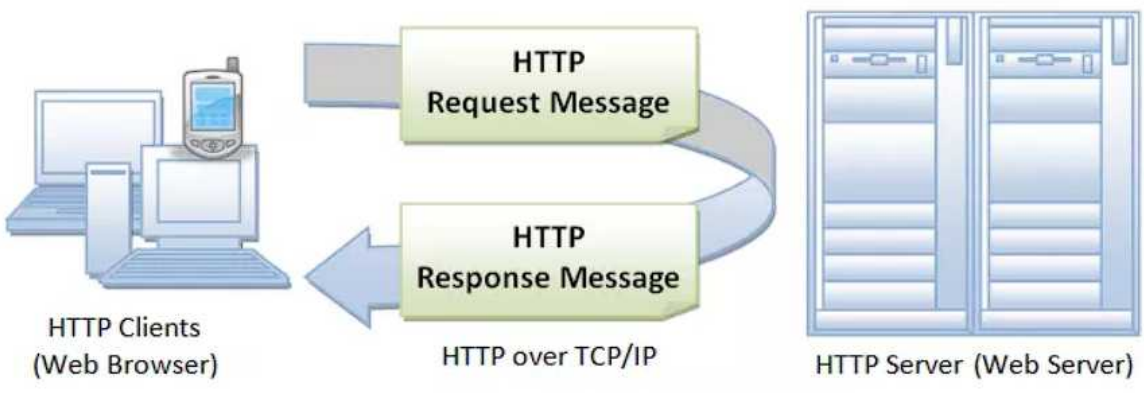
Hình 2.2 Giao thức HTTP

Ứng dụng của MQTT trong IoT rất phổ biến nhờ vào khả năng truyền dữ liệu nhẹ, hiệu quả và đáng tin cậy trong các môi trường hạn chế về băng thông hoặc nguồn điện. Giao thức MQTT được sử dụng rộng rãi trong các hệ thống nhà thông minh, giám sát môi trường, theo dõi sức khỏe, điều khiển công nghiệp và các thiết bị IoT như cảm biến, thiết bị đo đạc và bộ điều khiển. Nhờ vào mô hình pub/sub và khả năng duy trì kết nối ổn định giữa các thiết bị với server trung tâm (Broker), MQTT giúp các thiết bị có thể gửi và nhận dữ liệu theo thời gian thực. Ngoài ra, việc hỗ trợ nhiều mức chất lượng dịch vụ (QoS) và khả năng mã hóa bảo mật bằng SSL/TLS giúp MQTT trở thành lựa chọn lý tưởng cho các ứng dụng IoT cần độ tin cậy và bảo mật cao.

2.2.1 Đặc trưng của TTP

HTTP là giao thức đ để truyền tải dữ liệu trên i
giản, nhưng HTTP vẫn ào các đặc trưng cơ b
Một trong những ưu đ tính đơn giản. Giao thức n
sao cho dễ hiểu và dễ hát triển, kiểm thử ứ dụng web trở nên
thuận lợi hơn. Ngay c tạp trong phiên bản HTTP2, việc đóng
gói các thông điệp HT vẫn giữ cho giao thức này d
HTTP sở hữu khả n rộng linh hoạt thông qua việc sử dụng các HT TP
header. Nhờ đó, có th ổ sung các tính năng mới chỉ bằng cách t
giữa client và server ua một header cụ thể. Điều n
nghi nhanh chóng với đa dạng của web hiện đại.
Dù là giao thức statele – không lưu lại trạng thái giữa các yê – nhưng HTTP
vẫn có thể duy trì trạ ệc bằng cách sử dụng cookie. Cookie
HTTP cho phép tạo s trữ thông tin giữa các yêu c
đó cải thiện khả năng dõi ngữ cả nh và mang lại trải nghi ng nhất
quán hơn.
Tổng thể, HTTP là mảng và không thể thiếu trong truyền tải
liệu trên Internet, với n, mở rộng tốt và hỗ trợ duy tr
nhờ cơ chế cookie.

2.2.2 Nguyên lý kết



Hình 2.3 Nguyên lý kết nối HTTP

HTTP là một giao thức tầng ứng dụng không tự đảm bảo truyền tải tin cậy, vì vậy nó dựa vào TCP – một giao thức tầng truyền tải – để thiết lập kết nối và đảm bảo dữ liệu được truyền đầy đủ, đúng thứ tự. Tuy nhiên, việc thiết lập kết nối TCP trước mỗi lần trao đổi dữ liệu có thể gây ra độ trễ và giảm hiệu suất.

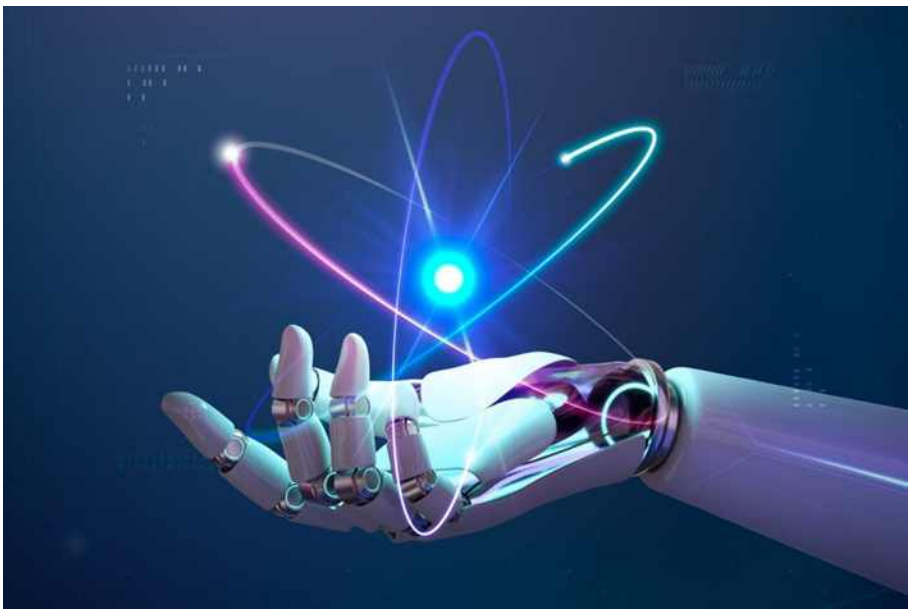
Trong HTTP/1.0, mỗi yêu cầu–phản hồi đều mở một kết nối TCP riêng biệt, gây tốn thời gian và tài nguyên, đặc biệt khi cần gửi nhiều yêu cầu liên tục. HTTP/1.1 đã cải tiến bằng cách sử dụng kết nối giữ sống (persistent connection) và hỗ trợ pipelining – cho phép gửi nhiều yêu cầu liên tiếp mà không chờ phản hồi, tuy nhiên cơ chế này vẫn gặp khó khăn về đồng bộ và quản lý luồng dữ liệu.

HTTP/2 giải quyết hạn chế đó bằng cách ghép (multiplex) nhiều luồng dữ liệu trong một kết nối TCP duy nhất, giúp truyền đồng thời nhiều yêu cầu–phản hồi mà không bị chồng chéo hoặc gián đoạn, từ đó tăng hiệu suất đáng kể.

Tuy nhiên, do TCP vẫn có những giới hạn trong bối cảnh mạng hiện đại (như độ trễ khi mất gói tin), Google đã phát triển giao thức QUIC, xây dựng trên nền UDP. QUIC tích hợp khả năng truyền tin tin cậy, mã hóa và quản lý kết nối hiệu quả hơn, giúp giảm độ trễ và cải thiện trải nghiệm người dùng.

Tóm lại, sự phát triển của HTTP qua các phiên bản, cùng với việc thử nghiệm các giao thức mới như QUIC, phản ánh nỗ lực không ngừng trong việc tối ưu hóa hiệu suất truyền thông trên Internet và nâng cao chất lượng truy cập cho người dùng.

2.3 Trí tuệ nhân tạo



Hình 2.4 Trí tuệ nhân tạo AI

Trí tuệ nhân tạo (AI)ột lĩnh vực liên ngành quan trọng trong
tính, nhằm phát triển hống có khả năng thực hiện các nhiệm vụ thường c ần
đến trí tuệ con người, luận, học tập và ra quy
chỉ là một công nghệ , mà còn là trụ cột của cuộc cách mạng
lần thứ tư, đóng vai ttrọng trong việc định hình lại cách con
sống, làm việc và tươrvới môi trường xung quanh.
AI bao gồm một hệ siờng lớn với nhiều chuyên ngành nh
liệu (data science), má học (machine learning), học sâu (deep learning), xử lý ngôn
ngữ tự nhiên (NLP), ty (computer vi sion) và robot học (robotics). Trong
đó, khoa học dữ liệu đ thu thập và phân tích lượng lớn dữ liệu
đó cung cấp đầu vào ọc máy. Máy học và học sâu l
giúp hệ thống AI học liệu một cách tự động, phát hiệnuy luật, đưa ra dự
đoán hoặc phân loại cảnh dữ liệu phức tạp mà con người khó nắ
AI không chỉ giới hạlý thông tin hay tự động hóa các công đoạn kỹ
thuật, mà còn tiến xa hô phỏng các năng lực tư duy c ời –
từ việc hiểu được văn ĩ tự nhiên, đến việc ra quyết định trong cá

huống phức tạp và thay đổi liên tục. Nhờ vậy, AI được ứng dụng rộng rãi trong nhiều lĩnh vực như: y tế (hỗ trợ chẩn đoán, phân tích hình ảnh y khoa), tài chính (phân tích tín dụng, dự báo thị trường), công nghiệp (giám sát quy trình sản xuất, phát hiện sự cố sớm), giáo dục (thiết kế chương trình học cá nhân hóa, trợ giảng ảo), giao thông (phát triển xe tự lái, tối ưu hóa điều phối), và cả trong nghệ thuật, giải trí, truyền thông số.

Trong tương lai gần, AI được xem là một trong những động lực chính thúc đẩy quá trình chuyển đổi số toàn diện ở cả cấp độ doanh nghiệp và quốc gia. Các hệ thống thông minh sẽ không chỉ thay thế con người trong những công việc lặp lại, mà còn hỗ trợ tối ưu hóa quá trình ra quyết định – nhanh hơn, chính xác hơn và phù hợp hơn với môi trường biến động hiện nay. Khi kết hợp với các công nghệ như Internet vạn vật (IoT), điện toán đám mây và blockchain, AI sẽ mở ra nhiều hướng phát triển mới, góp phần xây dựng một hệ sinh thái số hiệu quả, thông minh và có khả năng thích ứng cao trong tương lai.

2.3.1 Các công nghệ AI nổi bật

Máy học là kỹ thuật cho phép hệ thống tự động xây dựng các mô hình phân tích từ dữ liệu, mà không cần lập trình rõ ràng từng bước. Đây là một nhánh của trí tuệ nhân tạo (AI), hoạt động dựa trên nguyên lý rằng máy tính có thể tự học từ dữ liệu, nhận diện mẫu và đưa ra quyết định mà không cần lập trình rõ ràng cho từng tình huống. Nếu AI là lĩnh vực bao quát nhằm mô phỏng trí tuệ con người, thì máy học chính là nền tảng cốt lõi giúp hiện thực hóa khả năng học hỏi và thích nghi của máy móc.

Học sâu là một nhánh nâng cao của máy học, cho phép máy tính thực hiện các tác vụ phức tạp giống như con người, như nhận dạng tiếng nói, phân tích hình ảnh hoặc dự đoán xu hướng. Khác với phương pháp truyền thống xử lý dữ liệu theo công thức định sẵn, học sâu xây dựng các tầng xử lý thông tin (layers) mà trong đó máy tự học bằng cách khám phá mẫu dữ liệu lặp đi lặp lại. Hệ thống càng có nhiều tầng, khả năng học và suy luận càng hiệu quả.

Xử lý ngôn ngữ tự nhiên (NLP) là một nhánh của trí tuệ nhân tạo cho phép máy tính hiểu, diễn giải và tương tác với ngôn ngữ con người một cách tự nhiên. Nhờ vào NLP, máy có thể phân tích văn bản, nhận diện giọng nói, cảm xúc và trích xuất thông tin quan trọng từ hội thoại hoặc tài liệu. Điều này giúp thu hẹp khoảng cách trong giao tiếp giữa con người và máy móc, làm cho sự tương tác trở nên mượt mà và hiệu quả hơn.

Thị giác máy tính là một lĩnh vực của trí tuệ nhân tạo giúp máy tính có khả năng "nhìn thấy" và hiểu được hình ảnh, video từ thế giới thực thông qua camera hoặc cảm biến. Bằng cách sử dụng các thuật toán học sâu, hệ thống có thể nhận dạng, phân loại và phân tích đối tượng hoặc hành động một cách chính xác. Công nghệ này được ứng dụng rộng rãi trong nhiều lĩnh vực như nhận diện khuôn mặt, xe tự lái, giám sát an ninh, y học chẩn đoán hình ảnh và thể thao. Trong nhiều tình huống, độ chính xác và tốc độ xử lý của thị giác máy tính đã vượt qua con người.

Chúng giúp tự động hóa, nâng cao độ chính xác và tối ưu hóa hiệu suất hệ thống. AI tiếp tục là nền tảng then chốt trong quá trình chuyển đổi số và đổi mới công nghệ toàn cầu.

2.3.2 Ứng dụng trí tuệ nhân tạo AI

Trí tuệ nhân tạo (AI) đang dần trở thành công nghệ cốt lõi trong nhiều lĩnh vực, đóng vai trò quan trọng trong việc nâng cao năng suất, cải thiện quy trình và thúc đẩy đổi mới. Trong ngành y tế, AI hỗ trợ bác sĩ phân tích hình ảnh chẩn đoán, đọc và xử lý dữ liệu hồ sơ bệnh án, dự đoán nguy cơ bệnh lý và đưa ra phương pháp điều trị phù hợp với từng cá nhân. Bên cạnh đó, các hệ thống AI thông minh còn đóng vai trò quan trọng trong việc theo dõi sức khỏe từ xa, hỗ trợ phát hiện sớm các dấu hiệu bất thường và giảm tải cho các cơ sở y tế truyền thống. Trong lĩnh vực giáo dục, AI được ứng dụng để xây dựng lộ trình học tập cá nhân hóa, phù hợp với năng lực và nhu cầu của từng học sinh. Đồng thời, các trợ lý học tập ảo giúp người học dễ dàng tiếp cận kiến thức, nhận được hỗ trợ tức thời và duy trì quá trình học tập hiệu quả hơn dù ở bất kỳ đâu, bất kỳ lúc nào. Ngành tài chính cũng đang khai

thác AI để phát hiện gian lận trong giao dịch, phân tích hành vi tiêu dùng, đánh giá tín dụng và tự động hóa các hoạt động đầu tư nhờ vào khả năng xử lý dữ liệu theo thời gian thực. Trong lĩnh vực công nghiệp, AI đóng vai trò quan trọng trong việc giám sát tình trạng hoạt động của thiết bị, dự đoán sự cố để bảo trì kịp thời, tự động hóa quy trình sản xuất và kiểm tra chất lượng sản phẩm thông qua thị giác máy tính. Nhờ đó, hiệu suất vận hành và độ chính xác trong sản xuất được nâng cao, đồng thời giảm thiểu sai sót và chi phí.

Còn trong nông nghiệp, AI được ứng dụng để theo dõi tình hình mùa vụ, phân tích dữ liệu môi trường thu thập từ các cảm biến, nhận diện sớm các loại sâu bệnh và điều khiển hệ thống tưới tiêu một cách thông minh. Điều này không chỉ giúp tối ưu hóa việc sử dụng tài nguyên mà còn góp phần nâng cao năng suất và chất lượng nông sản. Lĩnh vực giao thông – vận tải đang được cải thiện rõ rệt nhờ ứng dụng trí tuệ nhân tạo. Các công nghệ như xe tự hành, hệ thống điều phối giao thông thông minh, phân tích dữ liệu giao thông thời gian thực giúp tối ưu hóa tuyến đường di chuyển và giảm thiểu ùn tắc, tai nạn. Ngoài ra, AI còn hỗ trợ trong việc giám sát an toàn bằng camera và cảm biến, nâng cao hiệu quả quản lý hạ tầng giao thông.

Trong thương mại điện tử và marketing số, AI mang lại trải nghiệm mua sắm cá nhân hóa cho người dùng bằng cách phân tích hành vi tiêu dùng và đề xuất sản phẩm phù hợp. Các chatbot thông minh hỗ trợ chăm sóc khách hàng 24/7, trong khi các thuật toán học máy giúp doanh nghiệp tối ưu hóa các chiến dịch quảng cáo, nâng cao hiệu quả tiếp cận và giữ chân khách hàng. Bên cạnh đó, lĩnh vực an ninh – quốc phòng cũng đang ứng dụng AI trong việc giám sát khu vực, phát hiện nguy cơ tiềm ẩn, AI cũng đóng vai trò quan trọng trong việc xử lý hình ảnh từ vệ tinh, giúp phân tích dữ liệu địa lý, giám sát thiên tai, theo dõi môi trường và hỗ trợ ra quyết định kịp thời trong các tình huống khẩn cấp như cháy rừng, lũ lụt hay sạt lở đất.

Trong lĩnh vực năng lượng, trí tuệ nhân tạo góp phần vào việc vận hành các hệ thống điện thông minh (smart grid), dự đoán chính xác nhu cầu tiêu thụ, tối ưu hóa

phân phối và sử dụng guồn năng lượng, đặc biệt l
như điện mặt trời và đ
Với tốc độ phát triển hanh chóng cùng khả năng ứng dụng sâ
ngành, AI đang trở th thúc đẩy quá trình chuy
triển bền vững trên toà

2.4 Mô hình Prophet

Hình 2.5 Mô hình Prophet

Trong bối cảnh công át triển vượt bậc, khả năng dự đoán xu l
lai từ dữ liệu lịch sử ở thành một trong những lợi thế cốt l
liệu, với ứng dụng rộng khắp từ dự báo thời tiết, hiệu suất cổ iếu, xu hướng thời
trang trong thương mại sớm bệnh lý trong y tế. Trong số các
kỹ thuật hỗ trợ cho kả năng này, dự báo chuỗi thời gian đón
nhưng cũng là một qu tạp kể cả đối với những người nhiều kinh
nghiệm. Nhằm đơn giản hóa quy trình này, Meta (tr
Facebook) đã phát tri – một thư viện mã bằng
Python, do Sean J. Taylor và Ben Letham xây dựng. Nó mang
ràng, cụ thể về xu hướt mùa vụ của dữ liệu góp phần cách mạng
lĩnh vực phân tích dự đoán. Prophet được thiết kế để dễ dùng
chuyên, kết hợp mô ền thống và học máy hiện đại. So với

Prophet dễ cấu hình hơn, xử lý tốt mùa vụ, dữ liệu thiếu và ngoại lệ, mang lại dự báo hiệu quả mà không cần kiến thức thống kê sâu. Prophet sử dụng mô hình cộng để phân rã chuỗi thời gian thành ba thành phần: xu hướng (trend) biểu thị biến động dài hạn của dữ liệu, mùa vụ (seasonality) phản ánh các biến động theo chu kỳ được tích hợp nhằm tăng độ chính xác. Chuỗi thời gian, về bản chất, là tập hợp dữ liệu sắp xếp theo thời gian, mỗi điểm dữ liệu đại diện cho một thời khắc cụ thể và thường chịu ảnh hưởng của xu hướng, mùa vụ và các sự kiện (nhiễu). Trước đây, các mô hình như ARIMA là lựa chọn phổ biến nhưng yêu cầu kiến thức thống kê sâu rộng. Từ năm 2017, với sự ra đời của Facebook Prophet, việc dự báo chuỗi thời gian đã trở nên đơn giản, dễ tiếp cận và hiệu quả hơn bao giờ hết, tạo ra tiềm năng tiếp cận nhiều lĩnh vực mới.

2.4.1 Cách thức hoạt động của Prophet

Mô hình này hoạt động theo 3 phân đoạn chính và được tổ chức vô cùng logic, tính tự động hóa cao, giúp người sử dụng dễ xách định được dự đoán chính xác mà không đòi hỏi hay yêu cầu cao về thống kê.Giai đoạn tiền xử lý dữ liệu đây là bước nền tảng, nơi dữ liệu thô được chuẩn bị sẵn sàng cho quá trình mô hình hóa. Một trong những ưu điểm vượt trội của Prophet là khả năng tự động xử lý các vấn đề phổ biến của dữ liệu chuỗi thời gian:

Xử lý giá trị thiếu (Missing Values): Trong thực tế, dữ liệu sẽ không hoàn hảo. Prophet có khả năng tự động nội suy (interpolate) các giá trị bị thiếu bằng cách sử dụng các phương pháp phù hợp, đảm bảo chuỗi thời gian liên tục và không bị gián đoạn. Nhờ đó giúp tránh loại bỏ nhầm dữ liệu quan trọng hoặc mất thời gian ghi thủ công.

Prophet có khả năng xử lý tốt các giá trị bất thường và ngoại lai trong chuỗi thời gian, nhờ thiết kế linh hoạt giúp giảm ảnh hưởng của những điểm dữ liệu sai lệch đến kết quả dự báo. Thay vì loại bỏ chúng một cách cứng nhắc, Prophet cố gắng giảm thiểu ảnh hưởng của chúng đến việc ước lượng xu hướng và tính thời vụ, giúp mô hình mạnh mẽ hơn.

Nhờ vào khả năng tự động vốn có, người sử dụng không cần phải chuẩn bị và làm sạch dữ liệu đầu vào thủ công, vốn thường tốn nhiều thời gian và đòi hỏi kỹ năng chuyên môn. Nhờ đó cho phép tập trung sâu hơn về việc hiểu các mẫu hình ảnh và đặc điểm của chuỗi thời gian nên sẽ đưa ra các quyết định kinh doanh hoặc phân tích hiệu quả hơn.

Giai đoạn ước lượng mô hình thực hiện sau khi dữ liệu đã được làm sạch và chuẩn bị, Prophet tiến hành xây dựng mô hình dự đoán. Tại đây nó sẽ tập trung phân tách và ước lượng từng phần của chuỗi thời gian ở từng thời điểm quan sát. Xu hướng (Trend), Prophet xác định chiều dài của dữ liệu dù ổn định, tăng hoặc giảm. Nó không những là một đường thẳng mà còn có thể tự động tìm ra những điểm mà xu hướng thay đổi (changepoints). Đây là những thời điểm mà tốc độ tăng trưởng hoặc suy giảm của chuỗi thời gian có thể đột ngột thay đổi (vd: khủng hoãng kinh tế hay ra mắt một sản phẩm nào đó). Nhờ việc cho phép xu hướng thay đổi tai các điểm đó, Prophet có thể mô hình hóa được diễn biến của dữ liệu. Tính đến Tính Thời vụ (Seasonality), đây là những biến động lặp đi lặp lại theo một chu kỳ nhất định. Mô hình này vô cùng nhạy trong việc mô hình hóa các loại dữ liệu có tính thời vụ khác nhau.

Bảng 2.2 Đặc điểm của mô hìnhProphet

Thời vụ	Đặc điểm
Hàng ngày (Daily)	Nếu dữ liệu có độ phân giải theo giờ hoặc phút
Hàng tuần (Weekly)	Biến động theo các ngày trong tuần
Hàng tháng (Monthly)	Biến động theo các tháng trong năm
Hàng năm (Yearly)	Biến động theo các mùa hoặc sự kiện lặp lại hàng năm
Tùy chỉnh (Custom)	Cho phép người sử dụng định nghĩa các chu kỳ thời vụ riêng nếu dữ liệu có mẫu hình lặp lại theo một khoảng thời gian đặc biệt nào đó

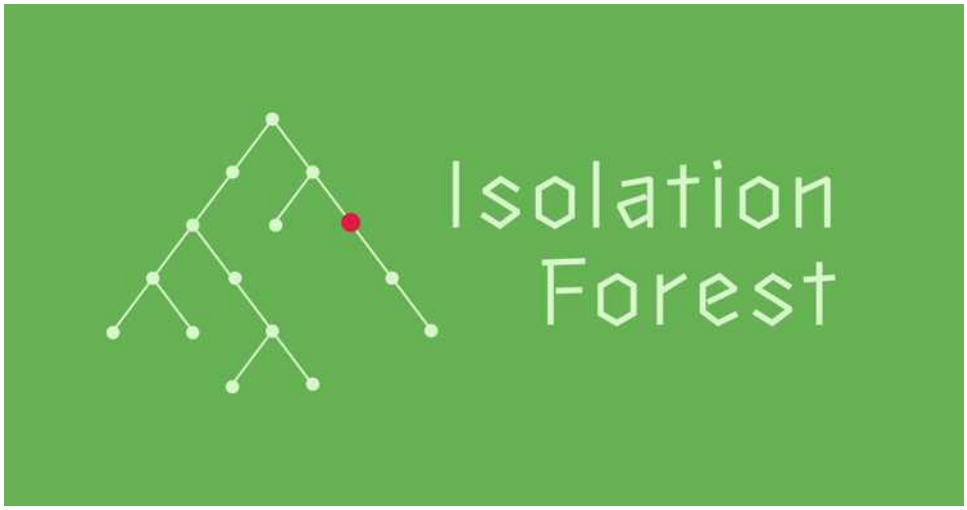
Giai đoạn dự báo thực hiện sau khi các thành phần xu hướng và tính thời vụ đã được ước lượng một cách chính xác, Prophet sẽ sử dụng chúng để tạo ra dự báo cho giai đoạn tương lai mong muốn.

Tạo ra tương lai (Future Forecasts) thông qua mẫu hình mà nó học được từ các dữ liệu tổng quá khứ, Prophet sẽ ngoại suy các thành phần này vào tương lai. Nó cộng tổng các thành phần đã ước lượng (xu hướng, tính thời vụ, ngày lễ) để tạo ra dự đoán cuối cùng cho từng thời điểm trong giai đoạn dự báo.

Khoảng tin cậy (Confidence Intervals) Prophet không chỉ đưa ra một điểm dự báo duy nhất mà còn cung cấp khoảng tin cậy (uncertainty intervals). Vì vậy nó cho biết được khoảng mà giá trị thực có thể nằm bên trong đó, giúp cho người sủ dụng suy luận được mức độ tin cậy của dự báo. Các khoảng này có thể được điều chỉnh để phản ánh mức độ biến động trong dữ liệu lịch sử và mức độ tin cậy mong muốn của dự đoán.

2.5 Mô hình Isolation Forest

Isolation Forest là một thuật toán học máy không giám sát (unsupervised learning) được dùng để phát hiện những dữ liệu hay điểm dữ liệu không bình thường (anomaly detection). IF xuất hiện lần đầu tiên năm 2008 bởi Fei Tony Liu và các cộng sự. Khác với nhiều phương pháp truyền thống cố gắng mô hình hóa hành vi "bình thường", Isolation Forest tiếp cận bài toán bằng cách tìm cách cô lập các điểm dữ liệu lạ – vốn thường xuất hiện với tần suất thấp và có giá trị khác biệt so với phần lớn dữ liệu còn lại.



Hình 2.6 Mô hình Isolation Forest

Thuật toán dùng cấu tr – tương tự cây quyết ưng khác biệt ở chỗ không dùng nhi chí đánh giá như độ tinh khiết. Thay vào đó, mỗi cây trong rừng cợc xây dựng bằng cách chọn ngẫu nhi (feature), sau đó chọn trị nằm trong khoảng giữa giá trị nhỏ nhá và lớn nhất của đặc tr đó để thực hiện phép chia tách. Quá tr tục được lặp lại đệ quy chến khi không thể chia nhỏ thêm được.

Do các điểm bất thườ biệt so với dữ liệu chung, chúng th bị cô lập chỉ sau vài bại, các điểm bình thường th bước phân tách hơn đ ập hoàn toàn. Dựa vào độ dài đu gốc đến lá trong nhiề cây, thuật toán sẽ tính toán một điểm bất thường (anomaly score). Những điểm cờng đi ngắn thường được đánh giá l

Isolation Forest có nhiều ưu điểm nổi bật: hiệu suất tính toán cao, thời gian chạy tuyến tính theo số lượ lụng thấp, và đặc biệt l dữ liệu hoặc giả định hân phối xác suất. Thuật toán này rất ph các hành vi bất thườn gian lận tài chính, lỗi, hoặc sự cố hệ thống, và cũng hoạt trên cả tập dữ liệu lớn lẫn nhỏ.

2.5.1 Ứng dụng của Isolation Forest

IF là một trong những oán phát hiện bất thường (anomaly detect và được dùng rộng rã c biệt trong các tình inh

những mẫu dữ liệu hiếm gặp hoặc có hành vi không tuân theo quy luật chung. Ví dụ như ứng dụng trong phát hiện gian lận giao dịch thẻ tín dụng.

Trong tài chính, đa số giao dịch là bình thường, trong khi giao dịch gian lận rất ít và thường có dấu hiệu bất thường như: địa điểm lạ, số tiền cao đột ngột, hoặc thời điểm mua hàng khác với thói quen người dùng. Điều này khiến các giao dịch gian lận trở thành "dị thường" trong tổng thể dữ liệu – rất phù hợp với nguyên lý hoạt động của Isolation Forest.

Không giống như nhiều thuật toán học máy yêu cầu dữ liệu có nhãn để học các mẫu gian lận, Isolation Forest hoạt động mà không cần thông tin nhãn trước đó. Thay vào đó, nó sử dụng cơ chế cô lập dữ liệu bằng cách xây dựng các cây phân tách ngẫu nhiên (Isolation Trees) nhằm phát hiện ra các điểm có hành vi khác biệt. Nếu một giao dịch bị cô lập sau rất ít lần chia tách (tức có đường đi ngắn từ gốc đến lá), thuật toán sẽ gán cho nó điểm bất thường cao, thể hiện khả năng đó là một giao dịch gian lận.

Vì không cần dữ liệu được gán nhãn trước và có thể làm việc tốt với khối lượng dữ liệu lớn, Isolation Forest rất phù hợp cho các hệ thống tự động phát hiện gian lận mà không cần con người phải huấn luyện mô hình.

Bảng 2.3Chức năng của mô hình Isolation Forest

Lĩnh vực	Chức năng
Phát hiện xâm nhập hệ thống mạng (intrusion detection)	Phát hiện các hoạt động truy cập mạng bất thường.
Phân tích dữ liệu cảm biến IoT	Tìm các giá trị bất thường trong dữ liệu thu thập từ thiết bị thông minh.
Giám sát thiết bị công nghiệp	Phát hiện sớm các dấu hiệu lỗi hoặc hỏng hóc máy móc qua dữ liệu vận hành.
Phân tích hành vi người dùng	Nhận diện hành vi bất thường trên các hệ thống trực tuyến như thương mại điện tử, mạng xã hội.
Phát hiện lỗi dữ liệu (data	Xác định những dòng dữ liệu sai lệch, nhiễu, hay

quality issues)	lỗi nhập liệu trong kho dữ liệu lớn.

2.6 Hệ điều hành Ub

Hình 2.7 Hệ điều hành Ubuntu

Ubuntu là một hệ điềuành mã nguồn mở được phát triển dựa tr
hướng đến việc cung ɑ cho người dùng một môi trường sử dụng
thân thiện. Hệ điều hà Ltd., một công ty công nghệ có trụ sở tại
Anh, duy trì và phát triển. Ubuntu được phát hành miễn phí vớ
gồm phiên bản dành cần, máy chủ và cả thiết bị di động.

Bảng 2.4Tính năng nổi bậc của Ubuntu

Tính năng	Chi tiết
Thừa hưởng tính năng nổi bật của Linux	Phát triển trên nền Linux, Ubuntu khai tlế mạnh của hệ điều hành này, bao gồm tính chỉnh hiệu năng, tốc độ vận hành ấn tượn mật kiên cố chống lại các mã độc hay phần
Hỗ trợ người dùng trong việc cài đặt	Người sử dụng có thể dùng đĩa chạy trực t (Live CD hoặc USB). Thông qua trình cài đặt Ubiquity, người sử dụng có thể đưa ra quyết định cài đặt HDH lên r không. Bên cạnh đó, người dùng hệ điề cứng có thể phân vùng lại ổ cứng để cài sountu và có thể dễ dàng gỡ cài đặt nếu không còn nh .

Giao diện	Ubuntu được triển khai giao diện đồ họa nity, đây là thiết kế để tối ưu hóa không gian hiển thị cũng như nâng cao trải nghiệm người dùng một cách hiệu quả và t .
Ứng dụng	Ubuntu được phân phối cùng nhiều phần n như Firefox, LibreOffice và trình tải torrmission. Ngoài ra, Ubuntu còn cung cấp một kho freegiúp người dùng dễ dàng tìm kiếm, cài đặt và quản lý phần mềm theo nhu cầu.

2.7 Phần mềm Dock

Hình 2.8 Phần mềm Docker

Docker là nền tảng m đóng gói và triển khai ứng dụng trong các container – môi trường cách ly giúp đảm bảo tính nhất quáy trên nhiều hệ thống khác nhau. Mỗi container chứa toàn bộ ứng dụng cùn cấu hình và môi trườn ứng dụng luôn chạy ổn định d ai ở đâu – từ máy cá nhâ đến máy chủ hay cloud.

Khác với việc cài đặt m thủ công, Docker cho phép tạo các image – bản sao hoàn chỉnh của ứng c – từ đó sinh ra các container độc lp, gọn nhẹ và nhất quán. Điều này giúp k phổ biến "chạy được trên máy tôi, nhưn chạy ở nơi khác".

Trong đồ án này, Docker được dùng để triển khai các thành phần như MQTT Broker, Node-RED, InfluxDB, Grafana, Home Assistant,… Mỗi dịch vụ chạy trong một container riêng biệt, giúp việc quản lý, cập nhật và mở rộng dễ dàng hơn, đồng thời giảm rủi ro xung đột giữa các phần mềm.

Nhờ cơ chế chia sẻ nhân hệ điều hành, container sử dụng ít tài nguyên hơn so với máy ảo truyền thống nhưng vẫn đảm bảo cách ly tốt. Điều này mang lại hiệu quả cao trong triển khai các hệ thống giám sát và điều khiển thông minh như trong đề tài.

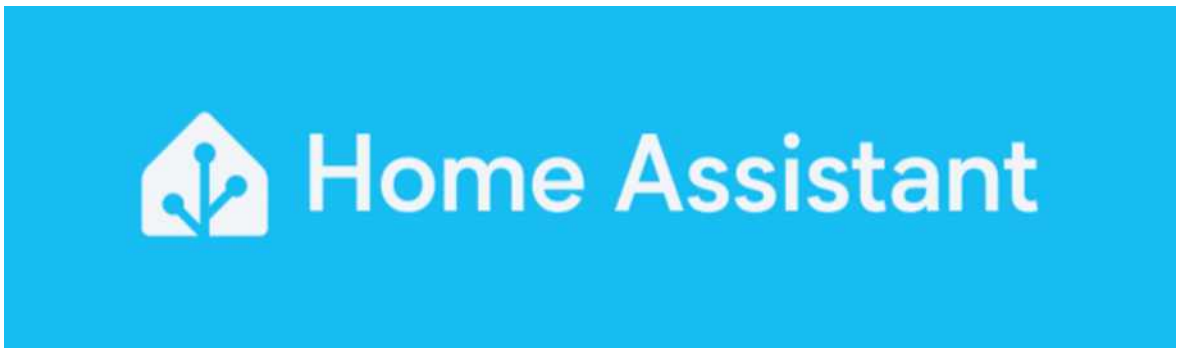
2.7.1 Các khái niệm liên quan

Bảng 2.5 Các khái niệm liên quan Docker

Khái niệm	Nội dung
Docker Engine	Là công cụ đóng gói ứng dụng và thành phần chính của Docker
Docker Hub	Được ví như "GitHub dành cho Docker images", nơi chứa hàng ngàn image công khai do cộng đồng chia sẻ. Người dùng chỉ cần pull image về và cấu hình theo nhu cầu là có thể sử dụng ngay, giúp triển khai nhanh chóng và tiện lợi.
Container	Là một phiên bản đang chạy (instance) của một image. Người dùng có thể tạo, khởi động, dừng, di chuyển hoặc xóa container thông qua Docker CLI hoặc Docker API
Docker Client	Là một công cụ quan trọng giúp người sử dụng giao tiếp với Docker host
Docker Daemon	Xử lý yêu cầu từ Client qua REST API và quản lý các đối tượng Docker; có thể giao tiếp với daemon khác để điều phối service.
Dockerfile	Một tập tin bao gồm các chỉ dẫn để build một image
Docker Volumes	Phần dữ liệu được tạo ra khi container được khởi tạo
Docker	Tập hợp các Docker Images cùng tên nhưng khác tags. VD:

Repository	golang:1.11-alpine
Docker Networking	Cho phép kết nối các container lại với nhau. Kiày có thể trên 1 host hoặc nhiều host
Docker Compose	Công cụ giúp chạy ứng dụng gồm nhiều cor một cách dễ dàng. Ta cấu hình mọi thứ trong file docker-compose.yml để tái sử dụng à quản lý thuận tiện. Docker Compose sẵn cùng Docker.
Docker Swarm	Phối hợp triển khai container
Docker Services	Container được triển khai trong môi trường pro Mỗi service chỉ chi từ một image, nhưng định nghĩa cách cha port nào, chạy bao nhiêu bản sao container để đ khả nă mở rộng.

2.8 Phần mềm Homssistant



Hình 2.9 Phần mềm Home Assistant

Home Assistant, còn gọi là HA hoặc HASS, là một nền tảng m
quản lý nhà thông minát triển bằng ngôn ngữ lập trình Python.
này có thể hoạt độngu hệ điều hành khác nhau và cho phép ı dùng
điều khiển thiết bị thóg minh thông qua giao diện web hoặc ứng dụng trên điện thoại. Home Assistantn có hai phiên bản chính, trong đó “Hom

là phiên bản cốt lõi, có thể cài đặt như một phần mềm thông thường trên các hệ điều hành phổ biến.

"Home Assistant OS" kết hợp giữa "Home Assistant Core" và các công cụ khác. Phiên bản này có thể cài đặt lên một chiếc máy tính như Raspberry Pi, máy ảo. Khi được cài đặt lên một thiết bị, cả hai phiên bản của Home Assistant sẽ biến thiết bị đó thành một hub trung tâm, cho phép kết nối và điều khiển các thiết bị nhà thông minh. Vai trò này tương tự như Gateway trong hệ sinh thái Xiaomi, Aqara, hay hub tổng của Samsung SmartThings. Là phần mềm mã nguồn mở được hỗ trợ bởi cộng đồng kỹ sư và lập trình viên toàn cầu, Home Assistant tương thích với hầu hết các thiết bị nhà thông minh, mở ra khả năng kết nối linh hoạt và đa dạng. Dù sử dụng phiên bản nào, người dùng cũng cần cài đặt Home Assistant trước. Sau đó, hệ thống sẽ tự động quét các thiết bị có sẵn và cho phép người dùng cấu hình để chúng hoạt động theo nhu cầu cá nhân.

Như vậy, nếu xét về sự tiện lợi và triển khai nhanh chóng, các nền tảng nhà thông minh phổ biến có phần vượt trội. Tuy nhiên, Home Assistant lại nổi bật ở khả năng điều khiển trong mạng nội bộ và mức độ tùy biến cao

2.8.1 Ứng dụng của Home Assistant

Home Assistant hoạt động như một hub tổng điều khiển nhà thông minh, cho phép người dùng thiết lập các ngữ cảnh để tự động hóa các tác vụ — từ đơn giản đến phức tạp. Nó đóng vai trò là cầu nối giữa các thiết bị sử dụng nhiều công nghệ IoT khác nhau, mang đến sự linh hoạt và kiểm soát toàn diện cho hệ thống nhà thông minh.

Nền tảng nhà thông minh mã nguồn mở này cho phép lưu trữ dữ liệu tại chỗ (On-Premises), giúp tăng cường tính riêng tư và an toàn mà không phụ thuộc vào nền tảng đám mây. Home Assistant có thể kết nối các thiết bị nội bộ hoặc tích hợp với các dịch vụ đám mây từ các nhà cung cấp nền tảng nhà thông minh, dù là mã nguồn mở hay đóng.

Bên cạnh đó, Home Ant hỗ trợ nhiều thành phầ n tích hợ dạng add -on hoặc plugin), cho phép ết nối với các hệ sinh thái IoT phổ biến như Google, Apple, Amazon, cũng như cáết bị như Philips, Xiaomi…

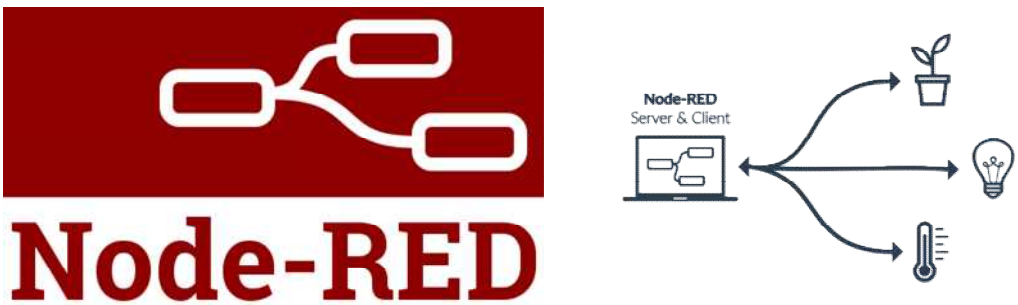
Thay vì phải cài đặt ứ từng hãng, Home Assistant giúp t g quản lý tất cả thiết bị ờng minh trong một nền tảng duy nhất. Điều này không chỉ giúp dễ dàng điều khiđiều kiện để thiết lập các ngữ cảnh linh cho phép các thiết bị àng với nhau.

Việc điều khiển thôngua một "máy c hủ" nội bộ giúp tăng tính bảo mật cho ngôi nhà và dữ liệu cá nhá thời, trong trường hợp mất kết nối Inte – điều thường ảnh hưởng đếphụ thuộc máy chủ đặt ở n – Home Assistant vẫn hoạt độn trong mạng cục bộ.

Người dùng có thể điá à thông minh bằng giọng nói thông qua ảo như Google Assistặc Amazon Alexa. Tuy nhiên, do tính n tùy biến cao, Home Ant không hẳn phù hợp với tất cả mọi ng và sử dụng đòi hỏi ng i dùng có kiến thức nhất định về hệ thrình, đặc biệt là khi cài đặt Hore trên các hệ điều hành như Windows Mac, hoặc khi cần tùyĭ Python.

Dù vậy, cộng đồng HAssistant rất đông đảo và nhiệt tình, nên i dùng vẫn có thể dễ dàng học hỏi n đề trong quá trình sử dụng.

2.9 Phần mềm Node-Red



Hình 2.10 Phần mềm Node-Red

Node-RED là một cônguồn mở, trực quan, giúp xây dựng lu (workflow) và ứng dụ. Nó cung cấp giao diện đồ họa tr nh duyệt web,

cho phép người dùng dễ dàng kéo – thả và kết nối các nút (node) để xử lý dữ liệu, điều khiển thiết bị hoặc tích hợp với các dịch vụ khác nhau.

Node-RED là công cụ mã nguồn mở được phát triển trên nền tảng Node.js, sử dụng giao diện đồ họa trực quan qua trình duyệt. Người dùng có thể kéo – thả các nút từ thư viện để xây dựng luồng xử lý, thực hiện các tác vụ như xử lý dữ liệu, kết nối với dịch vụ web, cơ sở dữ liệu, thiết bị IoT…

Với cộng đồng rộng lớn và nhiều gói mở rộng, Node-RED hỗ trợ kết nối các nền tảng phổ biến như MQTT, HTTP, MySQL, MongoDB, Raspberry Pi, Arduino... Công cụ này rất linh hoạt, phù hợp cho phát triển ứng dụng IoT, tự động hóa và quản lý dữ liệu.

Bảng 2.6Ưu điểm của Node-Red

Ưu điểm	Nội dung
Giao diện trực quan	Cung cấp một giao diện đồ họa dễ sử dụng dựa trên trình duyệt web. Người dùng có thể dễ dàng kéo và thả các nút để xây dựng và kết nối các luồng xử lý dữ liệu một cách trực quan, không cần viết mã phức tạp.
Các nút và luồng đa dạng	Cung cấp một giao diện đồ họa dễ sử dụng dựa trên trình duyệt web. Người dùng có thể dễ dàng kéo và thả các nút để xây dựng và kết nối các luồng xử lý dữ liệu một cách trực quan, không cần viết mã phức tạp.
Tích hợp IoT	Được phát triển đặc biệt cho việc xây dựng ứng dụng IoT. Ta có thể tương tác với các thiết bị như Raspberry Pi, Arduino, các cảm biến và các nền tảng IoT khác. Node-RED hỗ trợ các nút và giao thức như MQTT, CoAP, Modbus để kết nối và thu thập dữ liệu từ các thiết bị IoT một cách hiệu quả.
Xử lý dữ liệu linh hoạt	Ta có thể dùng các nút có sẵn trong Node-RED để thực hiện các tác vụ như chuyển đổi định dạng, phân tích văn bản, tính toán, lưu trữ và đồng bộ dữ liệu.

Giao tiếp thời gian thực	Hỗ trợ giao tiếp thời gian thực và khả năng phát hiện sự kiện. Ta có thể tạo các luồng làm việc để theo dõi, giám sát và phản ứng
Tích hợp dịch vụ web	Cho phép Ta tương tác với các dịch vụ web phổ biến như Twitter, Facebook, Dropbox và nhiều dịch vụ khác. Ta có thể gửi yêu cầu API, nhận dữ liệu từ các dịch vụ này và thực hiện các hành động liên quan.

2.9.1 Ứng dụng của Node-red

Node-RED là một công cụ mạnh mẽ để xây dựng các ứng dụng IoT. Ta có thể sử dụng Node-RED để kết nối, thu thập dữ liệu và điều khiển các thiết bị IoT như cảm biến, bộ điều khiển và thiết bị mạng. Ta cũng có thể tạo các luồng làm việc để xử lý dữ liệu, giám sát và phản ứng đối với sự kiện từ các thiết bị IoT.

Node-RED cho phép Ta tự động hóa các quy trình và công việc. Ta có thể xây dựng các luồng làm việc để thực hiện các nhiệm vụ như quản lý lịch trình, gửi thông báo, xử lý dữ liệu, tương tác với hệ thống và dịch vụ khác. Node-RED giúp giảm thiểu công sức và thời gian làm việc cần thiết cho các quy trình tự động.

Node-RED hỗ trợ xử lý dữ liệu và tích hợp hệ thống thông qua các nút giúp chuyển đổi định dạng, tính toán, lưu trữ, cũng như kết nối với cơ sở dữ liệu, tệp, giao thức web, API và các dịch vụ đám mây.

Node-RED cho phép Ta xây dựng các bảng điều khiển (dashboard) và giao diện người dùng để giám sát và quản lý các quy trình, dữ liệu và thiết bị. Ta có thể tạo các trang web tương tác, biểu đồ, bảng và các yếu tố trực quan khác để hiển thị thông tin và điều khiển hoạt động của hệ thống.

Node-RED hỗ trợ phân tích dữ liệu và AI thông qua tích hợp với các thư viện như TensorFlow và Node.js Machine Learning, cho phép xây dựng mô hình, phân loại dữ liệu và nhận dạng hình ảnh một cách trực quan.

Node-RED là nền tảng trực quan mạnh mẽ giúp xây dựng các luồng điều khiển IoT một cách nhanh chóng và dễ dàng, thuận tiện cho xử lý dữ liệu thời gian thực.

2.10 Cơ sở dữ liệu IrDB

Hình 2.11 Phần mềm InfluxDB

InfluxDB là một hệ thr sở dữ liệu được phát triển bởi InfluxDat
InfluxDB là mã nguồr sử dụng miễn phí. Phi
Enterprise cung cấp cátrì và kiểm soát truy cập đặc biệt cho kha
hàng doanh nghiệp, vảy chủ trong mạng doanh nghiệp. Ngo
ra, phiên bản InfluxDB 2.0 mới hoạt động như một dịch vụ
chỉnh với giao diện ngi dùng dựa trên web để thu thập và trực q
Hệ thống quản lý cơ sược viết bằng ngôn ngữ lập tr
Google , còn được g Phiên bản đầu tiên của
InfluxQL, một ngôn ndo InfluxData phát triển, cho các truy vấn cơ sở
dữ liệu bên ngoài.
InfluxDB 2.0 được vin ngữ mới có tên là Flux , đư
phát hành trên GitHub dưới dạng một dự án mã nguồn mở. D cập nhật
trên GitHub bởi các nầm việc với dữ liệu chuỗi thời gian. Flux
ngôn ngữ độc lập dàn chuỗi thời gian (TSDB). Nó có thể đ
sử dụng với InfluxDBphiên bản 1.7 trở lên, độc lập hoặ c với cở dữ liệu của bên
thứ ba.
Flux được tối ưu hóa cnh ETL (trích xuất, chuyển đổi, tải) trong
dữ liệu và không tươnới ngôn ngữ truy vấn InfluxQL đ
Tuy nhiên, InfluxData đang lên kế hoạch chuyển đổi cho các khách hàng hiện tại để
dịch mã InfluxQL sang

Cú pháp Flux dựa trên JavaScript, dễ học và linh hoạt. Một điểm mạnh của Flux là khả năng tích hợp nhiều nguồn dữ liệu khác nhau thông qua API của bên thứ ba. Do đó, Flux tương thích với các công cụ phân tích như Jupyter . Giao diện trao đổi dữ liệu Apache Arrow cho phép giao tiếp với các hệ thống khác và tích hợp trong môi trường dữ liệu lớn.

InfluxDB lý tưởng cho cơ sở dữ liệu chuỗi thời gian (TSDB), nơi lưu trữ chuỗi thời gian . Các cơ sở dữ liệu này được sử dụng, cùng với những mục đích khác, để lưu trữ và phân tích dữ liệu cảm biến hoặc giao thức có dấu thời gian trong một khoảng thời gian nhất định. Ví dụ: các thiết bị Internet vạn vật (IoT) hoặc thiết bị đo lường khoa học cung cấp hàng triệu tập dữ liệu đầu vào trong một luồng dữ liệu liên tục.

Dữ liệu này phải được xử lý nhanh chóng khi đến cơ sở dữ liệu. Vì lý do này, InfluxDB tích hợp sẵn dịch vụ thời gian sử dụng Giao thức Thời gian Mạng (NTP) để đảm bảo đồng bộ hóa thời gian giữa tất cả các hệ thống.

2.10.1 Ưu điểm của InfluxDB

So với các cơ sở dữ liệu quan hệ thông thường, các TSDB như InfluxDB mang lại lợi thế rõ ràng về tốc độ khi lưu trữ và xử lý dữ liệu đo lường có dấu thời gian. Một hệ quản trị cơ sở dữ liệu truyền thống sẽ chậm lại khi tổ chức các chỉ mục phức tạp, vốn không được sử dụng trong lĩnh vực ứng dụng này. InfluxDB duy trì tốc độ ghi cao ổn định nhờ sử dụng cấu trúc chỉ mục đơn giản và hiệu quả. Không giống như phiên bản 1.x, InfluxDB Cloud 2.0 mới của InfluxData là một giải pháp đám mây có thể chạy trên Amazon Web Services (AWS), Google Cloud Platform (GCP) hoặc Microsoft Azure. Với điện toán không máy chủ , bạn không cần cơ sở hạ tầng máy chủ riêng. Trong phiên bản đám mây, bạn không còn phải đặt trước các máy chủ riêng lẻ nữa. Thay vì cấu hình thủ công, hệ thống tự động điều chỉnh theo tải – điều này rất quan trọng trong các ứng dụng IoT công nghiệp và học máy, nơi dữ liệu có thể tăng đột biến.

Trong khi phiên bản đầu tiên yêu cầu ngăn xếp TICK (Telegraf, InfluxDB, Chronograf và Kapacitor), InfluxDB 2.0 đã có tất cả mọi thứ bạn cần. Cả phiên bản

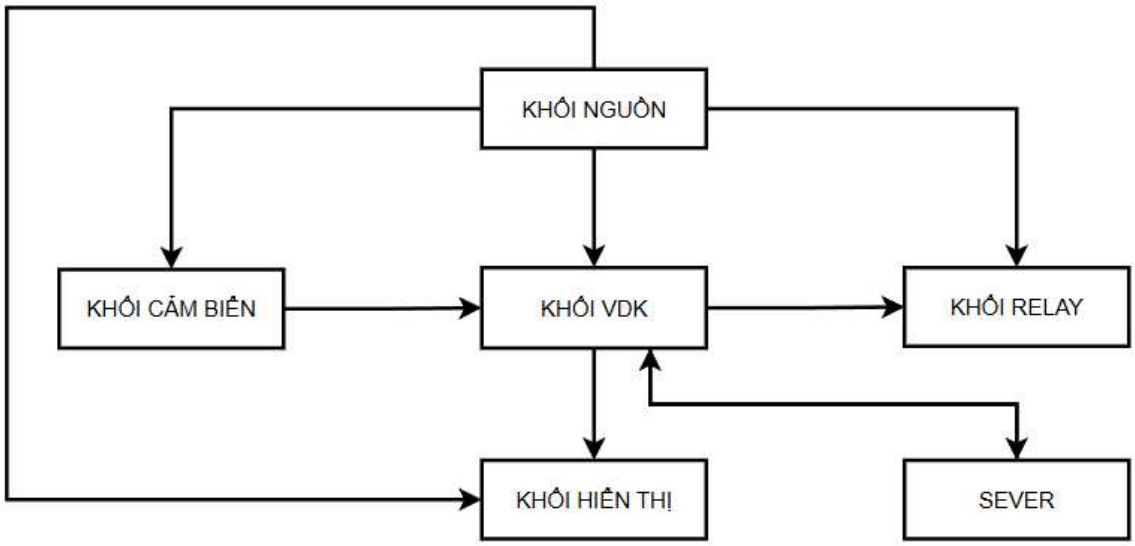
cục bộ và đám mây đều chứa toàn bộ hệ thống quản lý cơ sở dữ liệu trong một tệp chương trình duy nhất hiện có sẵn cho Linux 64-bit, Linux cho bộ xử lý ARM, macOS và dưới dạng container docker . Telegraf, v.v. vẫn có thể được sử dụng để thu thập dữ liệu cho InfluxDB 2.0.

Bảng 2.7Ưu điểm của InfluxDB

Tính năng	Nội dung
Gói miễn phí (có giới hạn)	Không cần tải xuống, không cần cài đặt và không cần cơ sở hạ tầng máy chủ nội bộ, giới thiệu nhanh nhất về công nghệ InfluxDB 2.0, gói miễn phí được thiết kế để bắt đầu sử dụng InfluxDB và cho các dự án sở thích nhỏ.
Hỗ trợ Flux	Flux là ngôn ngữ lập trình và truy vấn độc lập dành cho cơ sở dữ liệu chuỗi thời gian, giúp tăng năng suất bằng cách cho phép tái sử dụng mã dễ dàng. Flux được phát triển và tối ưu hóa để làm việc với dữ liệu trong InfluxDB 2.0, nhưng cũng có thể được sử dụng với các nguồn dữ liệu khác.
API hợp nhất	API InfluxDB v2 hợp nhất cung cấp quyền truy cập vào tất cả các thành phần của InfluxDB, chẳng hạn như thu thập dữ liệu, truy vấn, lưu trữ và trực quan hóa. Điều này cho phép chuyển đổi liền mạch giữa phiên bản nguồn mở đã cài đặt và phiên bản InfluxDB Cloud 2.0.
Hình ảnh hóa và bảng thông tin	Dựa trên dự án Chronograf cải tiến từ phiên bản đầu tiên của InfluxDB, giao diện người dùng mới cung cấp kết quả nhanh hơn đáng kể khi hình ảnh hóa và truy vấn dữ liệu theo thời gian thực.

CHƯƠNG 3. THIẾT KẾ PHẦN CỨN

3.1 Sơ đồ khối hệ thố

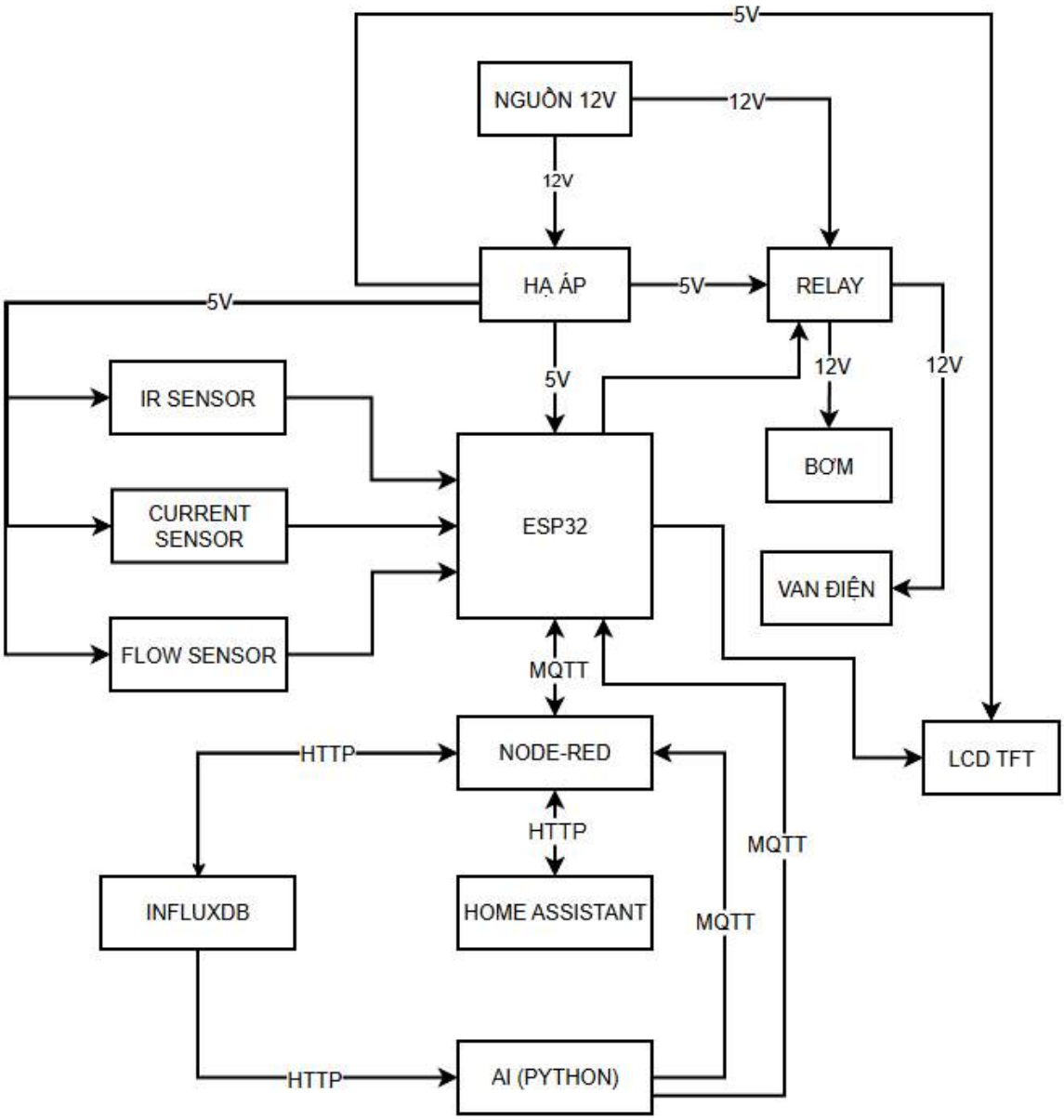


Hình 3.1 Sơ đồ khối tổng quát

Sơ đồ khối hình 3.1 thể hiện kiến trúc tổng quan của hệ thốg giám sát và điều khiển sử dụng nước tị hệ thống này, khối nguồn giữ vai tr
cấp điện áp ổn định: nguồn 5V DC được phân phối cho vi điều các cảm biến, module relay và màn hình hiển thị, trong khi nguồn 12VDC đư
cho các thiết bị tải nh từ và bơm nước thông qua chân th
relay, giúp đảm bảo sựkhả năng chịu tải của hệ thống.
Dữ liệu từ các cảm bi như cảm biến lưu lượng nước, cảm biến
biến hồng ngoại được vi điều khiển trung tâm (ESP32). Tại đây
vi điều khiển xử lý cá hiệu đầu vào, thực hiện các thuật toán p và tính
toán cần thiết, sau đó ết quả ra màn hình hiển thị để người d
dõi các thông số như lg thái bơm, mức tiêu th
Đồng thời, dữ liệu cũg được truyền lên server trung tâm (cà
nhằm lưu trữ, giám sáa và phục vụ cho việc phân tích chuy
không chỉ đóng vai tr còn cho phép người d

điều khiển từ xa đến l thống. Các lệnh này sẽ được vi điều khi
lý, sau đó thực thi việc tắt relay hoặc cập nhật giao diện hiển thị tương ứng.
Toàn bộ hệ thống hoạtheo nguyên tắc tự động, có thể giám sát
theo thời gian thực, đồi dễ dàng mở rộng và tích hợp v
dụng thông minh trongtương l ai như nhà thông minh hoặc trang

3.2 Sơ đồ khối chi tiế



Hình 3.2 Sơ đồ khối chi tiết

Hình 3.2 là thống giám sát và tối ưu hóa sử dụng nước thông minh trong gia đình được thiết kế với cấu trúc chặt chẽ, bao gồm các khối chức năng chính phối hợp với nhau để thực hiện các nhiệm vụ giám sát, điều khiển và phân tích dữ liệu. Toàn bộ hệ thống sử dụng nguồn điện một chiều 12VDC làm đầu vào. Nguồn này được đưa vào module hạ áp để chuyển đổi xuống mức 5VDC, phục vụ cho vi điều khiển ESP32, các cảm biến, màn hình LCD TFT và module relay. Đồng thời, nguồn 12V cũng được đưa trực tiếp đến module relay để điều khiển các thiết bị tải như bơm nước và van điện từ thông qua chân thường mở của relay.

Các cảm biến được tích hợp bao gồm cảm biến lưu lượng nước (flow sensor), cảm biến dòng điện (current sensor) và cảm biến hồng ngoại (IR sensor). Dữ liệu từ các cảm biến này được truyền về ESP32 để xử lý. Sau khi xử lý tín hiệu, vi điều khiển sẽ gửi lệnh điều khiển đến relay để đóng/ngắt bơm và van nước, đồng thời hiển thị thông số lên màn hình LCD TFT. Việc hiển thị thông tin giúp người dùng nắm bắt trực tiếp tình trạng hệ thống như lưu lượng nước, dòng điện tiêu thụ và trạng thái hoạt động của các thiết bị.

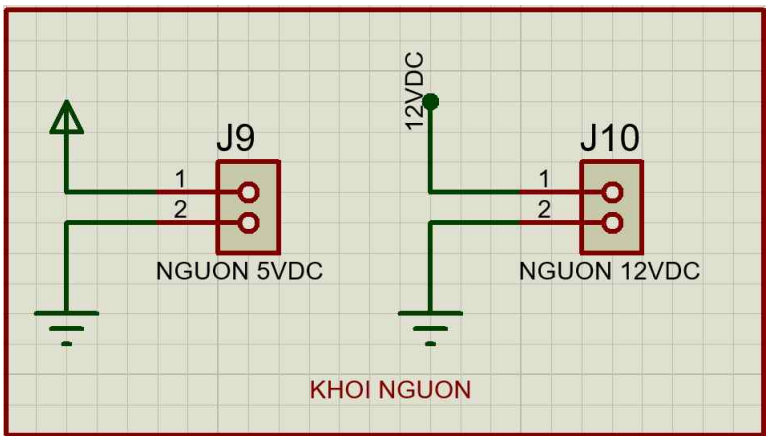
Bên cạnh đó, ESP32 cũng truyền dữ liệu lên nền tảng IoT thông qua giao thức MQTT. Node-RED đóng vai trò trung gian xử lý luồng dữ liệu, liên kết với nền tảng Home Assistant để xây dựng giao diện giám sát và điều khiển từ xa. Dữ liệu thu thập từ hệ thống được lưu trữ trên InfluxDB dưới dạng cơ sở dữ liệu thời gian thực, hỗ trợ truy xuất, thống kê và phân tích hiệu quả. Ngoài ra, hệ thống còn tích hợp trí tuệ nhân tạo (AI) được xây dựng bằng Python nhằm phân tích xu hướng sử dụng nước, dự đoán thời điểm cần thay lõi lọc hoặc phát hiện bất thường như rò rỉ nước. AI có thể truy cập dữ liệu từ InfluxDB hoặc nhận trực tiếp qua Node-RED để đưa ra quyết định hỗ trợ tối ưu vận hành hệ thống.

Tổng thể hệ thống hoạt động theo nguyên tắc tự động và kết nối chặt chẽ giữa phần cứng – phần mềm – dữ liệu, cho phép người dùng giám sát và điều khiển thiết bị từ xa thông qua Internet, đồng thời nâng cao hiệu quả sử dụng nước và điện năng trong gia đình.

Bảng 3.1Chức năng của từng phần trong sơ đồ k

Thành phần	Chức năng
Nguồn 12V + Hạ áp	Cấp điện áp 5V và 12V cl
ESP32	Thu thập – xử lý – điều niển – truyền dữ liệu
Cảm biến (IR, dò điện,lưu lượng)	Giám sát lưu lượng nướ hiện sử dụng
Relay + Bơm + Van	Thực hiện điều khiển vật đóng/ngắt thiết bị nước
Node-RED	Xử lý luồng logic, kết nốiệ thống
Home Assistant	Giao diện giám sát và điề
InfluxDB	Lưu trữ dữ liệu thời gian t
AI (Python)	Dự đoán, phân tích xu ướng, cảnh báo thông minh
LCD TFT	Hiển thị thông số hệ thốn

3.3 Khối nguồn



Hình 3.3 Khối nguồn

Hình 3.3 thể hiện kh , nguồn điện chính được cung adapter chuyển đổi điệ (AC) sang điện một chiều (DC) với đ iện áp đầu ra 12VDC ổn định. NC này sau đó được chia làm hai nhánh riêng biệt cho các mụcác nhau trong mạch.

Đầu tiên, nguồn 12VIqua một module giảm áp LM2596S LEL – đây là một module chuyểi DC -DC có khả năng hạ áp từ 12V xuống mức điện áp thấp hơn theo nhu cầuờng hợp này, LM2596S đư
giảm áp xuống 5VDCể cung cấp nguồn ổn định cho vi điều khiển, các cảm biến, cũng như phần input (khiển) của module relay. Việc sử dụng modu le LM2596S giúp đảm bạy cảm với điện áp như vi đi
biến được hoạt động điện áp phù hợp, tránh hiện tượng quá áp
thiết bị.

Thứ hai, dòng điện 12 giảm áp vẫn được sử dụng trực tiếp để
nguồn cho các thiết bị ở điện áp cao hơn, điển hình là các thi
van điện từ. Cụ thể, pân Out (NO hoặc NC) của module relay được nối với nguồn 12VDC này, nện trực tiếp để đóng/mở van điện từ khi r
được kích hoạt bởi tín hiệu từ vi điều khiển.

Việc thiết kế như vậch biệt rõ ràng giữa nguồn điều khiển (5VDC) và nguồn tải bảo an toàn và ổn định cho hệ thống. H thời, sử dụng LM2596 còn giúp tăng hiệu suất sử dụng năng l nhiệt so với các giải pp tuyến tính như IC 7805.

Hình 3.4 Module hạ áp LM2596S

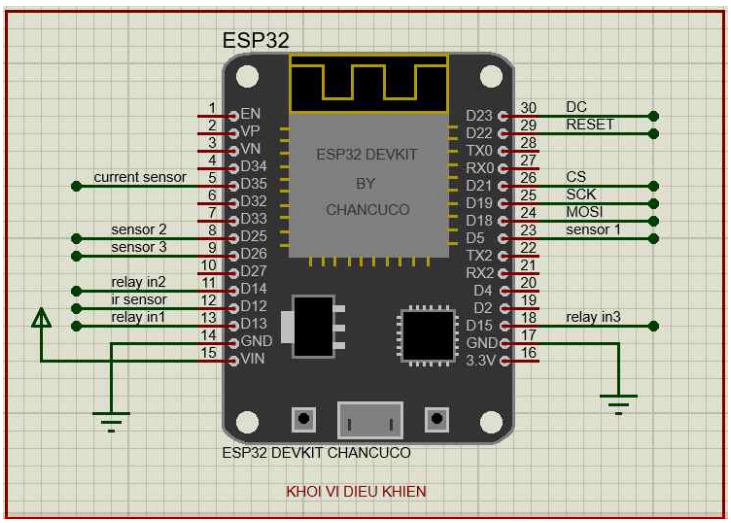
LM2596S là một bộ đáp bước giảm (buck converter) đ
chuyển đổi điện áp đầo cao hơn thành điện áp đầu ra thấn một cách hiệu quả. Đây là một tronều chỉnh điện áp phổ biến nhất trong các ứng dụng điện tử, nhờ vào; với dòng điện cao và hi

Bảng 3.2Thông số kĩ thuật của LM2596S

Đặc điểm	Thông số
Input	Từ4.5V - 40V.
Output	Điều chỉnh từ 1.23V đến 37V.
Dòng output max	3A.
Hiệu suất	75%-90% tùy vào điều kiện tải và điện áp.
Tần số	~150KHz
Bảo vệ quá tải	Tích hợp bảo vệ quá nhiệt và ngắn mạch.
Kích thước	Dạg SIP-7 hoặc TO-220.

3.4 Khối vi điều khiể

Khối vi điều khiển sử ıg ESP32 như hình 3.5, một vi điều kh tích hợp WiFi và Bluetooth với hiệu suả vai trò trung tâm điều phối v
động của hệ thống. ES32 thực hiện thu thập dữ liệu từ các cảnbiến như: cảm biến đo lưu lượng nước, cện, cảm biến siêu âm ho
ngoại. Dữ liệu này sauể đưa ra các quyết định điều khiển ph
như: kích hoạt bơm ran điện từ, bật cảnh báo r hoặc hiển thị thông tin lên màn hình OLED/LCD.

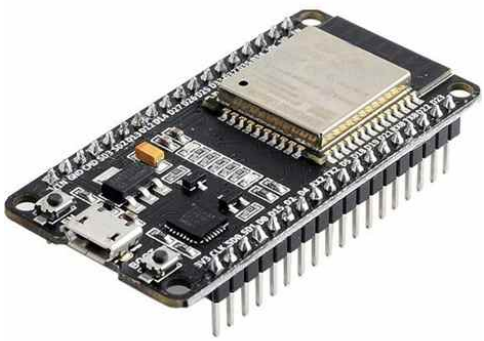


Hình 3.5 Khối vi điều khiển

Ngoài ra, ESP32 còn c giao tiếp mạng, truyền tải dữ liệu l
thống máy chủ thông ıa giao thức MQTT chạy trên n ền tảng WiFi. Điều này cho phép người dùng có tđiều khiển từ xa qua nền tảng Home As
hoặc ứng dụng web tídợp.
ESP32 cũng là đầu mìu kiện thông qua các luồng điều khiển được thiết kế trên Node-RED, từ đó tạo nên hệ thống vận hành tự đ. Với khả năng xử lý mạnh ẽ, tiêu thụ năng lượng thấp và tích hợp k ESP32 là nền tảng phù ứng dụng IoT như h
tối ưu hóa sử dụng nưa đình.

ESP32-30P-Micro-CP2102

ESP32 là một vi điềuhợp (SoC) có chi phí thấp v iện năng thấp, được tích hợp sẵi Wi-Fi và Bluetooth hai chế độ (Classic xử lý trung tâm của SP32 là Tensilica Xtensa® LX6, với ha lõi đơn hoặc lõi kép, cho phéphọn hiệu năng phù hợp theo nhu cầu ứng Trên cùng một con chn tích hợp nhiều thành phần phần cứng n tắc ăng-ten, RF balun, bộ khuếch đại công suất (PA), bộ khuếch yếu (LNA), các bộ lọc và quản lý năng lượng. Nhờ đó, nó giúp giả lượng linh kiện bên ngối ưu hóa thiết kế phần cứng tổng thể.
Được phát triển bởi Essif Systems (Trung Quốc) và sản xuất tr nghệ 40nm của TSM ESP32 là phiên bản nâng cấp từ điều khiển ESP8266, mang lại khh hoạt và hiệu năng xử lý mạnh mẽ h biệt phù hợp cho các úoT hiện đại.



Hình 3.6 ESP32 Dev Kit

Bảng 3.3Thông số kỹ thuật ESP32

Đặc điểm	Thông số
Vin	3.3V/5V
Băng tần	2.4GHz, 802.11
Tần số Clock	b/n/g/e/i
BlueTooth	160/240Hz
Hỗ trợ	BLE, SPI, UART, I2C, I2S, ADC, DAC
Dãy nhiệt độ hoạt động	-40 độ C ~125 độ C
Giao thức mạng	IPv4, TCP/UDP, HTTP, FTP

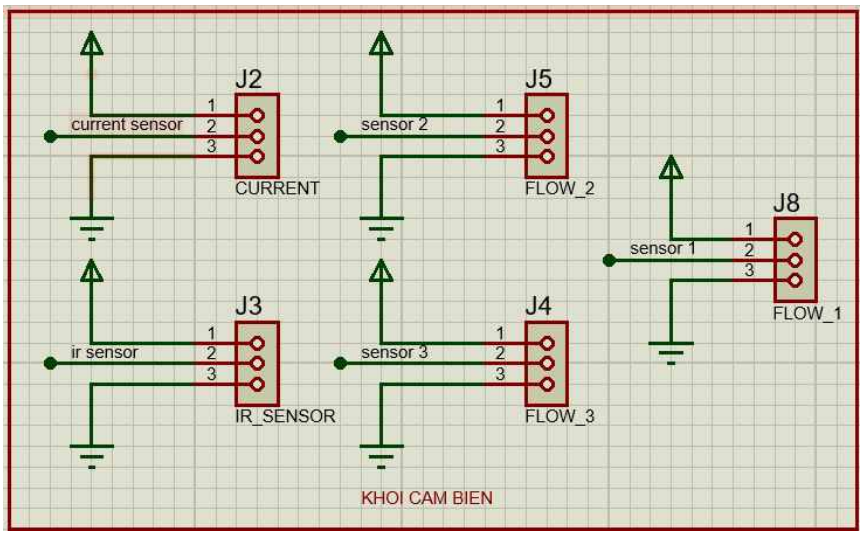
Bảng 3.4Chức chân các chân ESP32

Tên chân	Chức năng
VIN	Có hai chân nguồn chính trên bo mạch ESP32: chân VIN và chân 3V3. Chân VIN có thể được sử dụng để cấp nguồn trực tiếp cho ESP32 và các thiết bị ngoại vi nếu có nguồn 5V đã được điều chỉnh ổn định. Trong khi đó, chân 3V3 là đầu ra của mạch ổn áp tích hợp trên bo mạch, cung cấp điện áp 3.3V, có thể lấy dòng từ chân này với công suất tối đa khoảng 600 mA để cấp cho các module hoặc cảm biến khác.
GND	Chân nối đất
GPIO	Bo mạch phát triển ESP32 có 25 chân GPIO, và mỗi chân có thể được gán các chức năng khác nhau thông qua việc lập trình các thanh ghi thích hợp. Có nhiều loại GPIO khác nhau, bao gồm: chỉ hỗ trợ tín hiệu số, hỗ trợ tín hiệu analog, hỗ trợ cảm ứng điện dung, v.v. Các chân GPIO hỗ trợ analog và cảm ứng điện dung đều có thể được cấu hình để hoạt động như chân GPIO kỹ thuật số. Ngoài ra, tất cả các chân GPIO đều có thể được cấu hình để hoạt động như các ngắt (interrupts).
ADC	ESP32 tích hợp hai bộ ADC SAR độ phân giải 12 bit và hỗ trợ đo

	tín hiệu trên 15 kênh (các chân có hỗ trợ analog). Một số chân trong số này có thể được sử dụng để xây dựng bộ khuếch đại có thể lập trình (programmable gain amplifier) nhằm đo các tín hiệu analog nhỏ. Ngoài ra, ESP32 còn được thiết kế để có thể đo điện áp ngay cả khi đang ở chế độ ngủ (sleep mode).
DAC	Các kênh DAC có thể đóng vai trò như một biến trở số, cho phép điều chỉnh tín hiệu analog để điều khiển thiết bị một cách linh hoạt.
Cảm ứng	Bo mạch có 9 chân GPIO hỗ trợ cảm ứng điện dung. Khi một tải điện dung (chẳng hạn như ngón tay người) tiến gần đến chân GPIO, ESP32 sẽ phát hiện sự thay đổi điện dung đó.
SPI	ESP32 có ba giao diện SPI (SPI, HSPI và VSPI) hoạt động được ở cả chế độ slave và master. Các giao diện SPI này cũng hỗ trợ các tính năng SPI phổ biến sau: - 4 chế độ thời gian truyền dữ liệu theo định dạng SPI - Tốc độ truyền dữ liệu lên đến 80 MHz và các mức xung nhịp được chia từ 80 MHz - Bộ đệm FIFO hỗ trợ tối đa 64 byte. Trong số ba giao diện SPI của ESP32, chỉ có VSPI và HSPI là có thể sử dụng được cho giao tiếp SPI thông thường; giao diện SPI thứ ba đã được sử dụng nội bộ để kết nối với chip bộ nhớ flash tích hợp. Trong các thư viện chuẩn, các chân thuộc giao diện VSPI thường được sử dụng phổ biến.
I2C	ESP32 có một bus I2C duy nhất cho phép kết nối tối đa 112 cảm biến và thiết bị ngoại vi. Theo mặc định, chân SDA và SCL được gán lần lượt cho GPIO21 và GPIO22. Tuy nhiên, có thể thay đổi hai chân này linh hoạt bằng cách sử dụng lệnh wire.begin(SDA, SCL) để mô phỏng giao thức I2C trên bất kỳ chân GPIO nào.
UART	Bo mạch phát triển ESP32 có hai giao diện UART là UART0 và UART2, hỗ trợ giao tiếp bất đồng bộ (RS232 và RS485) cũng như

	giao tiếp ngoại IrDA với tốc độ lên đến 5 UART trên ESP32 hỗ trợ điều khiển tín hiệu phần cứng CTS đồng thời cũng hỗ trợ điều khiển luồng bằng phần mềm thông qua các tín hiệu XON và XOFF.
PWM	Bo mạch 15 kênh PWM (gần như tất cả các chân điều khiển bởi một bộ điều khiển PWM. Các tín hiệu đầu ra PWM này có thể được sử dụng để điều khiển động cơ kỹ LED. Bộ điều khiển PWM bao gồm các bộ định thời (PWM timer) và các bộ tạo xung PWM (PWM operator). Mỗi thời cung cấp tín hiệu định thời theo kiểu đồng bộ hoặc độc và mỗi bộ tạo xung PWM sẽ tạo ra dạng sóng cho một kênh PW
EN	Được sử dụng để kích hoạt ESP32. Khi chân này được HIGH, con chip sẽ được kích hoạt và hoạt động kéo xuống LOW, con chip sẽ chuyển sang chế tiêu thụ năng lượng thấ.

3.5 Khối cảm biến



Hình 3.7 Khối cảm biến

Khối cảm biến hình 3.7 là thành phần then chốt trong hệ thống vụ thu thập dữ liệu thực tế từ môi trường và thiết bị để gửi về vi điều k

Trong hệ thống này, khối cảm biến bao gồm ba cảm biến lưu lượng nước YF-S201, một cảm biến dòng điện ACS712, và một cảm biến hồng ngoại OMRON E18-D80NK, được bố trí tại các vị trí chiến lược nhằm giám sát toàn diện quá trình sử dụng nước và điện năng.

Cụ thể, ba cảm biến YF-S201 được lắp đặt trên các đường ống nước đầu vào và đầu ra của hệ thống, dùng để đo lưu lượng nước sử dụng tại các khu vực khác nhau trong hộ gia đình. Mỗi cảm biến hoạt động theo nguyên lý tạo xung khi có dòng nước chảy qua, cho phép ESP32 tính toán được tổng lượng nước tiêu thụ trong ngày, tuần hoặc tháng. Dữ liệu từ các cảm biến này không chỉ giúp người dùng theo dõi lượng nước sử dụng mà còn là đầu vào quan trọng cho hệ thống phân tích xu hướng sử dụng nước và đưa ra cảnh báo khi vượt ngưỡng cài đặt.

Tiếp theo, cảm biến dòng điện ACS712 được tích hợp để theo dõi mức tiêu thụ điện năng của các thiết bị như máy bơm nước hoặc máy lọc. Cảm biến này đo cường độ dòng điện chạy qua tải, giúp xác định thiết bị đang hoạt động ở trạng thái bình thường, quá tải hay tiêu tốn điện không cần thiết. Dữ liệu thu được sẽ được lưu trữ và hiển thị thông qua nền tảng Grafana, đồng thời dùng để xây dựng các cảnh báo và biểu đồ giám sát trên giao diện người dùng.

Cuối cùng, cảm biến hồng ngoại OMRON E18-D80NK đóng vai trò phát hiện sự hiện diện của người dùng tại vị trí sử dụng nước như bồn rửa hoặc khu vực rửa tay. Khi có người đứng gần, cảm biến phát hiện vật cản và gửi tín hiệu về ESP32 để tự động kích hoạt bơm nước và mở van điện từ. Điều này giúp hệ thống vận hành một cách thông minh, tiết kiệm nước và đảm bảo tiện nghi mà không cần người dùng thao tác thủ công.

Cảm biến lưu lượng nước YF-S201

Cảm biến lưu lượng nước YF-S201 là loại cảm biến phổ biến, thường được sử dụng trong các thiết bị như máy bơm mini, máy nước nóng, hoặc hệ thống bơm nước cho hồ cá. Cảm biến này hoạt động dựa trên nguyên lý quay của cánh quạt và cảm biến Hall tích hợp bên trong. Khi dòng nước chảy qua, nó làm quay cánh quạt, từ đó cảm

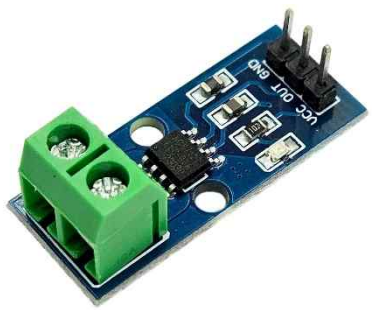
biến Hall sẽ phát hiệnà tạo ra các xung vuông dạng tín hiệu N
tốc độ dòng chảy.

Hình 3.8 Cảm biến lưu lượng nước YF-S201

Bảng 3.5Thông số kĩ thuật cảm biến lưu lượng nu

Đặc điểm	Thông số
Nguồn	5 - 24V
Dòng tiêu thụ	< 10mA
Áp lực chịu được	1.75Mpa
Lưu lượng đo	1 - 30 (L/min)
Nhiệt độ hoạt động	< 120 độ C
Độ ẩm	35% - 90% RH
Kích thước	61 36 x 34mm

Cảm biến dòng điện .



Hình 3.9 Cảm biến dòng điện ACS712

Cảm biến dòng điện ect Current Sensor) dựa tr all
để đo dòng điện AC/Dó kích thước nhỏ gọn, dễ kết nối, giá trị t
điện áp Analog tuyến ıh theo cường độ dòng điện cần đo nên r

trình với Vi điều khiển, thích hợp với các ứng dụng cần đo dòng AC/DC với độ chính xác cao.

Đo dòng điện DC: Khi đo DC phải mắc tải nối tiếp Ip+ và Ip- đúng chiều, khi dòng điện đi từ Ip+ đến Ip- Vout sẽ ra mức điện áp tương ứng 2.5~5VDC tương ứng dòng 0~Max, nếu mắc ngược Vout sẽ ra điện thế 2.5~0VDC tương ứng với 0~(-Max).Khi cấp nguồn 5VDC cho module khi chưa có dòng Ip (chưa có tải mắc nối tiếp) thì Vout = 2.5VDC, khi dòng Ip(dòng của tải) bằng Max thì Vout=5DC, Vout sẽ tuyến tính với dòng Ip trong khoản 2.5~5VDC tương ứng với dòng 0~Max, để kiểm tra có thể dùng đồng hồ VOM thang đo.

Đo dòng điện AC: Khi đo dòng điện AC, do dòng điện AC không có chiều nên không cần quan tâm chiều, khi cấp nguồn 5DC cho module khi chưa có dòng Ip (chưa có tải mắc nối tiếp với domino) thì Vout = 2.5VDC, khi có dòng xoay chiều Ip (dòng AC) do dòng xoay chiều độ lớn thay đổi liên tục theo hàm Sin, nên điện thế Vout sẽ có độ lớn tuyến tính với dòng điện AC từ 0~5VDC tương ứng với (-Max)~Max (dòng xoay chiều), để kiểm tra dùng đồng hồ VOM thang đo AC đo Vout.

Bảng 3.6Thông số kỹ thuật cảm biến dòng điện ACS712

Đặc điểm	Thông số
Vin	5VDC
Độ nhạy output	63~190mV/A
Độ nhiễu tín hiệu analog	Thấp
Độ trễ output để đáp ứng với input	5μs
Điện trở dây dẫn trong	1.2mΩ

Cảm biến vật cản hồng ngoại OMDHON E18-D80NK

Cảm biến vật cản hồng ngoại OMDHON E18-D80NK Adjustable IR Infrared Proximity Sensor có chất lượng tốt, độ bền và độ ổn định cao, đặc biệt cảm biến có khoảng cách điều chỉnh chính xác từ 3~80cm với thấu kính hồng ngoại chất lượng tốt, so sánh ngược lại là các loại cảm biến giá rẻ trên thị trường với thấu kính chất

lượng kém và khả năng đúng như thông số (không thể điều ch
gần được).

Hình 3.10 Cảm biến hồng ngoại OMDHON E18-D80NK

Cảm biến vật cản hng ngoại OMDHON E18 -D80NK Adjustable IR Infrared Proximity Sensor sử dụng ngoại để xác định vật cản phía tr m biến, cảm biến phát raa hồng ngoại với dải tần số chuyên biệtg chống nhiễu tốt kể cả ở điều n ánh sáng ngoài trời.

Cảm biến vật cản hng ngoại OMDHON E18 -D80NK Adjustable IR Infrared Proximity Sensor có thể chỉnh khoảng cách phát hiện vật cản m biến trở trên cảm biến cảm biến có ngõ ra tín hiệu là cấu trúc c NPN - Open Collector nên sẽ cần phải có trở kéo (khoảng 1~1 dương VCC để tạo thàHigh).

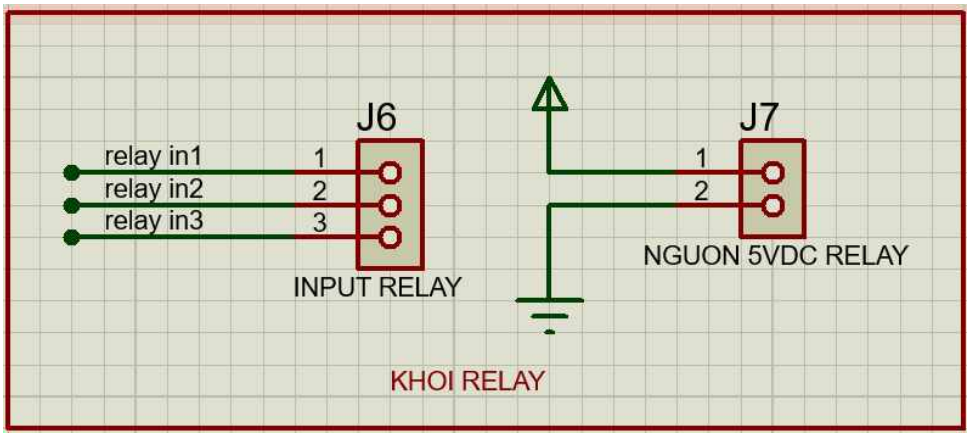
Cảm biến hoạt động điện áp 5VDC, dòng tiêu thụ nhỏ khoản khoảng cách phát hiện có thể ừ 3 đến 80 cm. Tần số đáp ứng của cảm biến vào khoảng 500Hz, phù hợhát hiện chuyển động tốc độ vừa. Thiết bị được thiết kế dạng lrụ với ren ngoài, dễ lắp đặt vào các h hộp kỹ thuật.

Bảng 3.7 Thông số kỹ thuật cảm biến hồng ngo

Đặc điểm	Thông số
Vin	5VDC
Khoảng điều chỉnh phện vật cản	3~80cm
Góc khuếch tán (góc c	3~5 độ
Dòng kích output	< 300mA

Kích thước	18x70mm

3.6 Khối Relay



Hình 3.11 Khối Relay

Hình 3.11 là khối rela đóng vai trò là phần tử chấp hành trun
điều khiển ESP32 điề khiển các thiết bị điện hoạt động ở mức điện áp và dòng
điện cao hơn, mà không gây ảnh hưởng đến mạch logic điều kh
3.3V hoặc 5V. Trong g giám sát và tối ưu hóa sử dụng n
này, khối relay được hiển trực tiếp van điện từ v c,
thông qua tín hiệu điề 2.
Cụ thể, đầu vào (IN) lay được kết nối với các chân GPIO của
khiển ESP32 để nhận hiệu điều khiển mức thấp (LOW) hoặc mức cao (HIGH),
tùy theo loại relay kíai nhận tín hiệu điều khiển, relay sẽ chuyển mạch
trạng thái từ thường đg (NC) sang thường mở (NO), cho phép
từ nguồn chính chạy qa đến tải (van điện từ hoặc bơm nước).
Nguồn cấp cho mạch khiển của module relay là 5VDC, đư
áp thông qua module LM2596S. Trong khi đó, nguồn 12VDC ấp cho các thiết bị
tải được lấy trực tiếp pter 12V, thông qua tiếp điểm chuyển mạch của relay.
Cách bố trí này đảm h ly giữa mạch điều khiển logic v tải
công suất, giúp tăng đn cho hệ thống.
Việc sử dụng relay mg lại khả năng đóng/ngắt thiết bị điện t cách tự động và
chính xác theo logic xử lý của hệ thống. Ví dụ, khi cảm biến l

có người sử dụng, ES gửi tín hiệu kích hoạt relay để mở van điện từ và khởi động bơm nước. Ngư t hiện sự hiện diện,ay sẽ ngắt nguồn đến các thiết bị nhằm ết kiệm nước và điện năng.

Module 4 Relay Opto 12VDC

Mạch 4 Relay được thiển bật/tắt các thiết bị điện AC hoặc DC thông qua relay cơ. Mạch hỗ trợ tùy chọn mức kích hoạt là n c cao (High) hoặc mức thấp (Low) thôngqua jumper cấu hình, giúp dễ dàng tương t vi điều khiển và hệ thắc nhau.

Một điểm nổi bật của ạch là được tích hợp Opto cách ly (Optoc cường độ an toàn và k g chống nhiễu, đặc biệt hữu ích trmôi trường công nghiệp hoặc khi điều n thiết bị điện áp cao. Đây là ưu đi nhiều loại mạch relay ic trên thị trường không có tính năng cách Mạch phù hợp cho cáđiều khiển thi ết bị điện tự động như: hệ thống nhà thông minh, điều khiển giám sát thiết bị từ xa..

Hình 3.12 Module 4 Relay 12V

Bảng 3.8 Thông số kỹ thuật Module Relay

Đặc điểm	Thông số
Điện áp sử dụng	12V
Dòng tiêu thụ	khoảng 200mA /1Rel
Tín hiệu kích mức hig	5V
Tiếp điểm đóng ngắt Relay trên mạch	Max 250VAC-10A hoặc 30VDC-10A

Kích thước	72 L) * 55 (W) * 19 (H) mm

Van điện từ thường m(NO)

Van điện từ VAKS pl 12V - loại thường mở) là thiết bị c
khiển dòng nước trong tưới tiêu tự động hoặc ứng dụng dân dụng
hoạt động ở điện áp 1 DC, an toàn khi sử dụng với nước. Ở trạ
điện, van luôn mở cho nước chảy qua. Khi cấp điện, van sẽ đón
nước. Thiết bị được c đồng thau bền bỉ, cho tuổi thọ cao. Van
hướng lắp cố định: nó có màng lọc. Sản phẩm ph
mô hình tưới rau, tưới, phun sương, và có thể kết hợp với bộ
hành tự động. Van c phiên bản N/O (Normally Open - thường mở), N/C (Normally Closed - thường đóng) và N/F (No Pressure - không áp lực) đáp ứng đa dạng nhu cầu sử dụng Ngoài ra, van có kh ả năng hoạt động ổn định trong thời gian dài mà không sinh nhiệt quá mức, giúp nâng cao độ bền của h
đóng mở nhanh và độ o kiểm soát dòng chảy chính xác, hạn chế
tối đa hiện tượng rò rỉ ng ống.

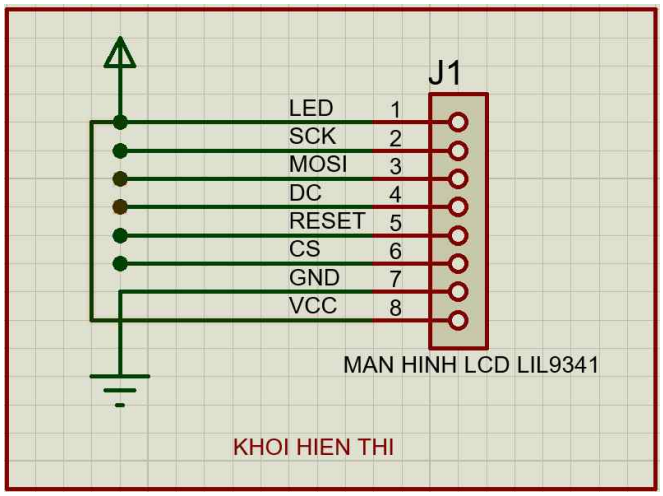
Hình 3.13 Van điện từ VAK (NO)

Bảng 3.9 Thông số kỹ thuật van điện từ

Đặc điểm	Thông số
Loại	NO (thường mở)
Kích thước	¼ inch
Điện áp làm việc	12VDC

Dòng điện kích	500mA
Áp lực	0.02-0.8Mpa

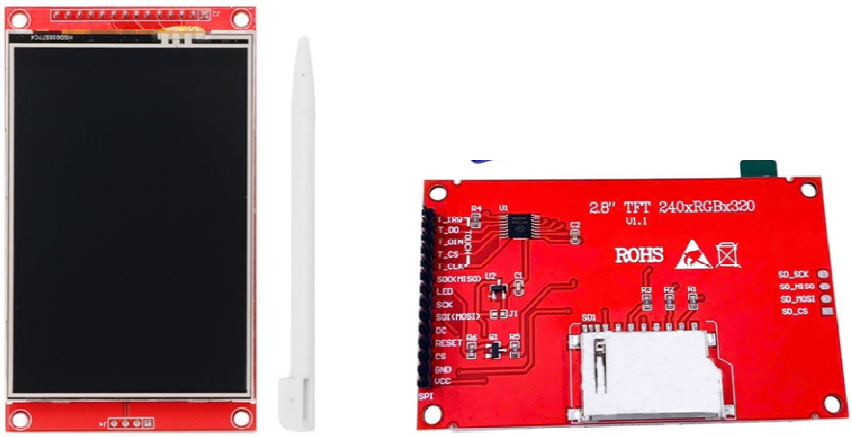
3.7 Khối hiển thị



Hình 3.14 Khối hiển thị

Khối hiển thị như hình sử dụng màn hình màu LIL9431 ới vi điều khiển ESP32 qua chuẩPI, có nhiệm vụ cung cấp thông tin trực quan về tình trạng hoạt động của hệ thốn Màn hình hiển thị các thông số chính nh sử dụng, trạng thái máy bơm (bật/tắt), t ình trạng van điện từ, và thống AI (ví dụ như cđộ phân giải cao và h sắc, LIL9431 cho phé trình bày thông tin rõ ràng, dễ hiểu, đồ tượng trực quan để ng dõi và thao tác tại

Màn hình LCD TFT Touch Screen 2.8 inch ILI9341 SPI Interface



Hình 3.15 Màn hình LCD TFT LIL9341

Màn hình cảm ứng TF inch (điện trở, ILI9341, giao tiếp SPI) l lựa chọn phổ biến trong các ứng dụng cần hiển thị và điều khiển cảm ứn màn hình dễ dàng kết khiển như Arduino, ESP32… giúp l trình đơn giản và hiệuhép xây dựng giao diện điều khiển trực quan ngay trên màn hình cảm ứng, thích hợp cho các dự án Io lệ thống nhúng và bảng điều khiển thôngninh.

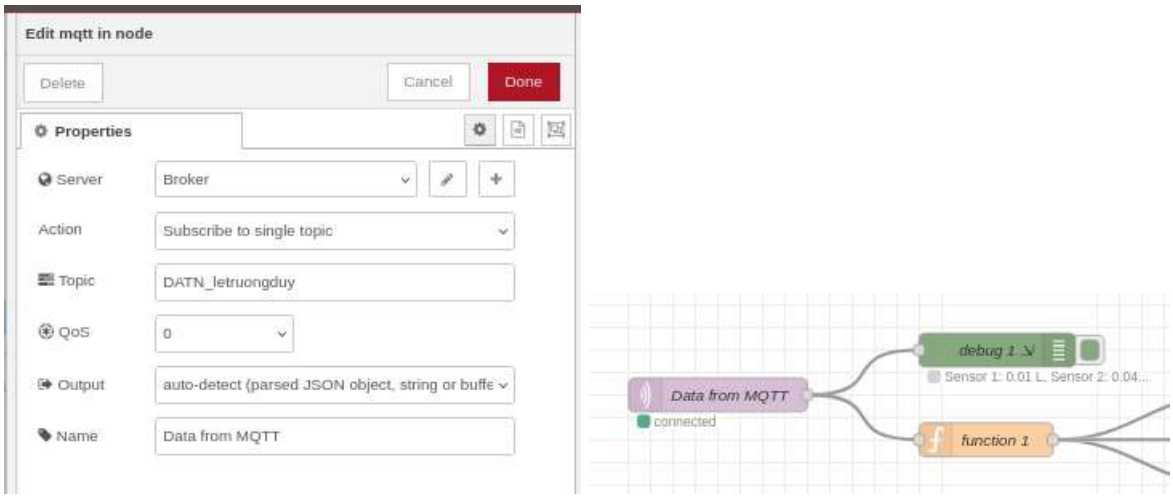
Bảng 3.10 Thông số kỹ thuật màn hình LCD

Đặc điểm	Thông số
Điện áp sử dụng	3.3~5VDC
Điện áp giao tiếp	TTL 3.3~5VDC
IC Driver hiển thị	ILI9341 giao tiếp SPI
Cỡ màn hình	2.8 inch
Độ phân giải	240 x 320 pixels
IC Driver cảm ứng	XPT2046 giao tiếp SPI
Tích hợp khe thẻ nhớ	Giao tiếp SPI
Kích thước hiển thị	46(W) X 65(H)mm
Kích thước toàn màn h	85x50 mm

CHƯƠNG 4. THIẾT LẬP PHẦN MỀM VÀ ĐIỀU KHIỂN

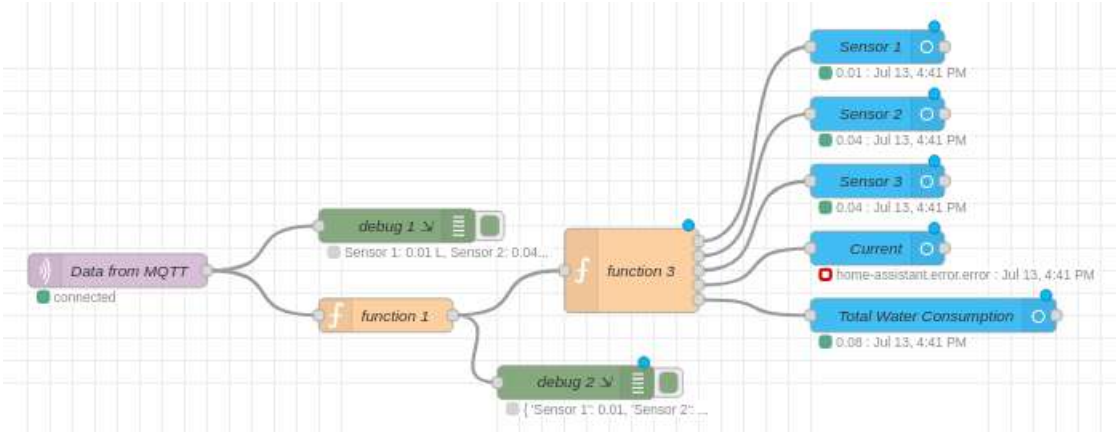
4.1 Thiết kế luồng điều kiện điều khiển với Node-RED

Node-RED là một nền tảng lập trình trực quan mã nguồn mở, cho phép người dùng xây dựng luồng xử lý dữ liệu (flows) một cách linh hoạt thông qua giao diện kéo thả. Trong hệ thống giám sát và tối ưu hóa sử dụng nước thông minh, Node-RED đóng vai trò trung tâm trong việc tiếp nhận dữ liệu từ phần cứng (ESP32 và cảm biến), xử lý logic điều kiện, phát hiện bất thường, điều khiển thiết bị ngoại vi, lưu trữ dữ liệu, kết nối với nền tảng AI và tương tác với Home Assistant để hiển thị thông tin và thực hiện các hành động tự động hóa. Các thành phần trong Node-RED được xây dựng thành sơ đồ luồng logic có cấu trúc rõ ràng và mạch lạc.



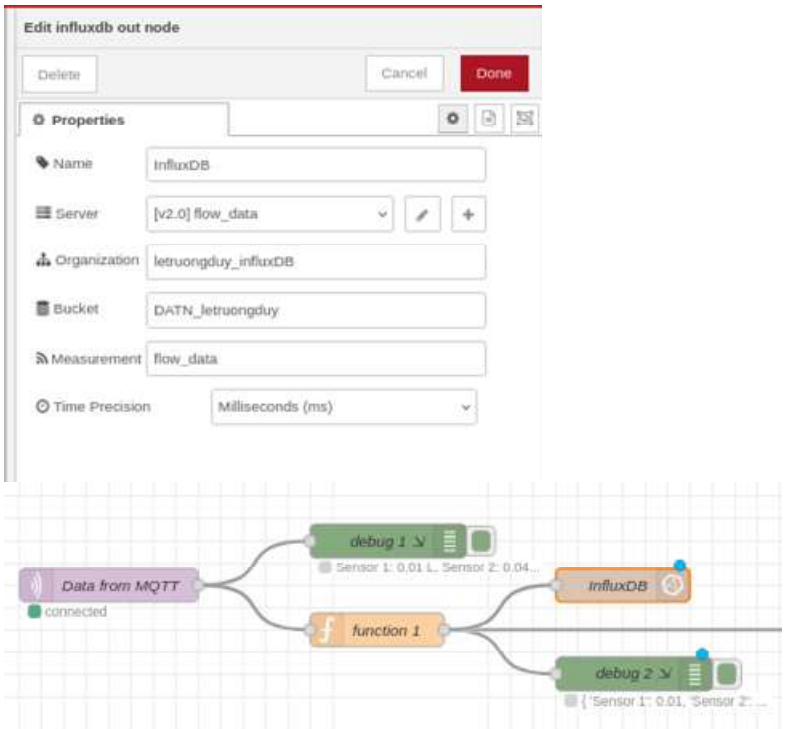
Hình 4.1 Thiết lập Node Mqtt in

Đầu tiên, dữ liệu từ các cảm biến được gắn với vi điều khiển ESP32 sẽ được truyền đến Node-RED thông qua giao thức MQTT như hình 4.1. Node tiếp nhận dữ liệu này là một mqtt in có nhãn “Data from MQTT”. Dữ liệu thu được là một chuỗi JSON bao gồm thông tin từ cảm biến đo lưu lượng nước (Sensor 1, 2, 3), cảm biến dòng điện (Current), cảm biến phát hiện người dùng (IR hoặc siêu âm), được xử lý sơ bộ trong function 1 để tách giá trị và gán vào các biến riêng lẻ phục vụ cho các bước xử lý tiếp theo. Sau đó, dữ liệu được chuyển tới node function 3 để thực hiện tính toán nâng cao và phân nhánh xử lý.



Hình 4.2 Luồng xử lý tín hiệu vào và hiển thị

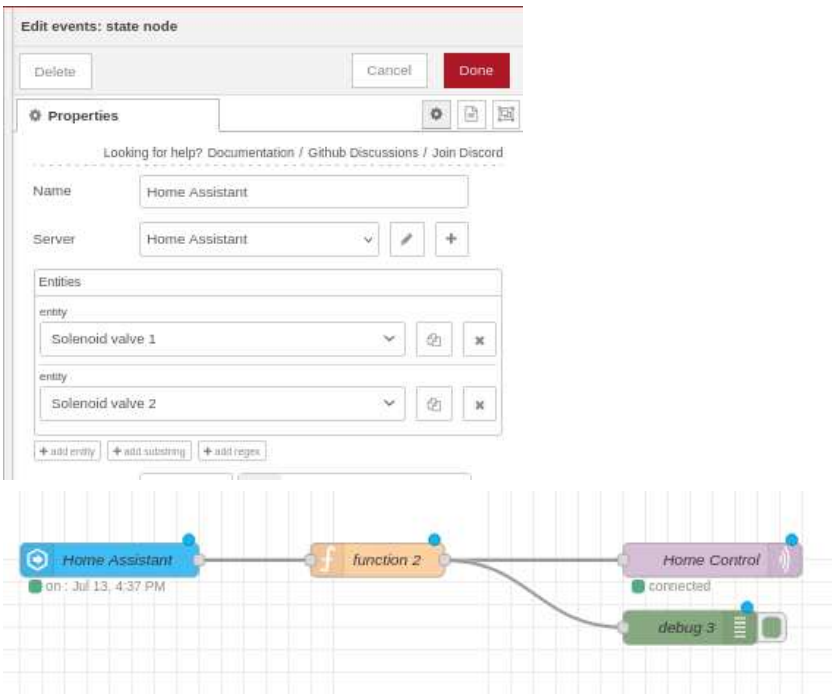
Sau khi được xử lý, ừng cảm biến được đưa đến các node t
Sensor 1, Sensor 2, Sensor 3 và Current thể hiện trong hình 4 , tại đây giá trị đo được được hiển thị kè p để phục vụ giám sát t gian thực. Đồng thời, các node debug ợc kết nối để lập trình viên có thể theo dõi dữ liệu và kiểm tra hg trong giai đoạn phát triển v nh thử



Hình 4.3 Thiết lập Node InfluxDB in

nghiệm.

Hình 4.2 thể hiện các dữ liệu quan trọng như lưu lượng nước và điện năng tiêu thụ cũng được lưu trữ thông qua node InfluxDB để phục vụ cho mục đích thống kê và hiển thị biểu đồ phân tích trên giao diện trực quan như Grafana, Home Assistant và



Hình 4.4 Thiết lập Node Even State

đặc biệt là sẽ được truy vấn để phục vụ cho phân tích AI chuyên sâu.

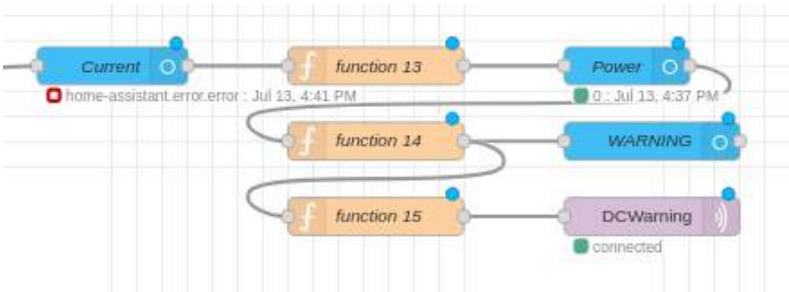
Trong hệ thống giám sát và điều khiển sử dụng nước, Home Assistant đóng vai trò là nền tảng giao diện trung tâm giúp người dùng theo dõi trạng thái hệ thống, thực hiện điều khiển và nhận cảnh báo theo thời gian thực. Để kết nối Node-RED với Home Assistant, hệ thống sử dụng hai node chính là event state và mqtt out mang nhãn Home Control.

Cụ thể, trong hình 4.4 node event state được sử dụng như điểm giao tiếp đầu tiên để Node-RED thiết lập kết nối với Home Assistant. Trong node này, địa chỉ IP của máy chủ Home Assistant cùng với các thông số xác thực như token truy cập được khai báo, nhằm đảm bảo việc kết nối giữa hai nền tảng diễn ra an toàn và ổn định. Ngoài ra, node này cũng đóng vai trò như một listener, có thể lắng nghe sự thay đổi

trạng thái từ phía Hom Assistant và kích hoạt luồng xử lý tươn -
RED nếu cần.

Bên cạnh đó, node mt với nhãn Home Control được sử dụng (bản
tin điều khiển từ Honsistant đến Node -RED thông qua giao thức MQTT. Cụ
thể, trên giao diện Hongười dùng được cung cấp hai nút nhấn (bu
tương ứng với hai trại: bật van điện từ và tắt van điện từ. Kl
nhấn vào một trong hai nút này, Home Assistant sẽ gửi mc
MQTT mà node Home Control đã đăng ký. Node-RED sẽ nhậin này và thực
hiện hành động điều lứng thông qua các node xử lý logic v
nối với van điện từ. Đây là cơ chế đi ều khiển thủ công cho phép người dùng can
thiệp trực tiếp vào hoạ độc lập với phần điều khiển tự động từ
AI, đặc biệt hữu ích trcác tình huống cần bảo trì, kiểm tra hệ tl
cấp nước theo ý muốn

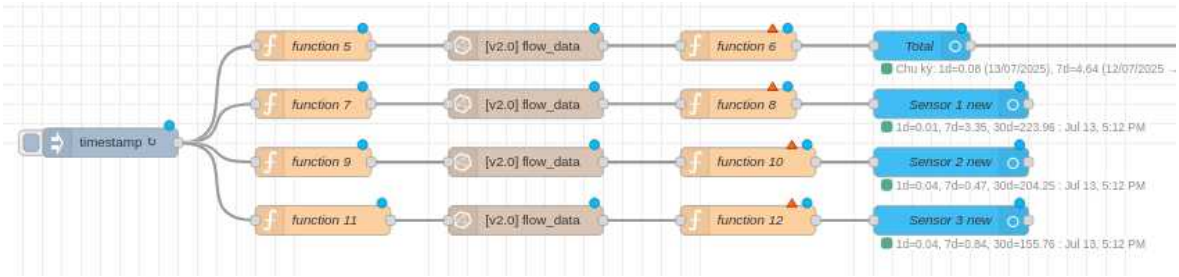
Việc tích hợp giữa Ne -RED và Home Assistant thông qua node event state và
mqtt out – Home Control giúp hệ thống trở nên linh hoạt hơn
bảo tính giám sát và ktừ phía người dùng. Cơ ch
điều kiện thuận lợi chviệc mở rộng các chức năng tự động hác trong tương
lai, như thiết lập kịchứng thông minh, lập lịch bật/tắt van hoặc đưa ra
cảnh báo sớm dựa trênước hàng ngày.



Hình 4.5 Luồng cảnh báo máy bơm

Về phần tính toán, hình 4.5 hệ thống sử dụng một số node funtion chuyên biệt để
xử lý số liệu. Node fnction 13 thực hiện phép tính công suất tiêu thụ điện bằng
công thức P = I × V, ó I là dòng điện đo được từ cảm biến v
mặc định cung cấp cho thiết bị, kết quả được đưa ra node Pc

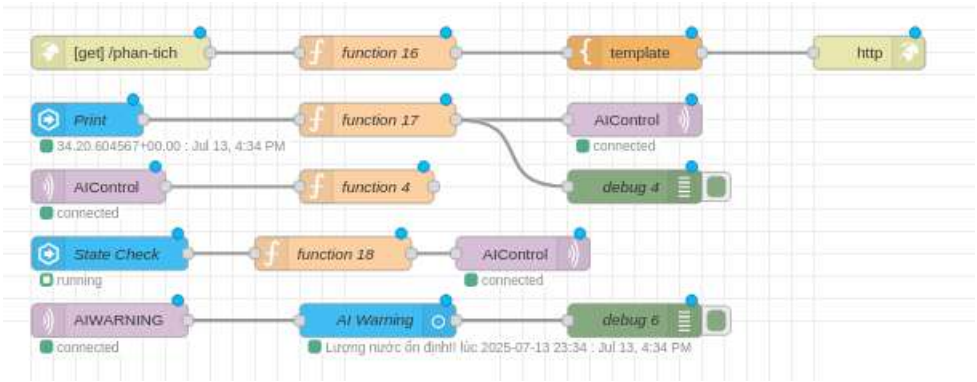
node function 14 thựclượng từ Sensor 1 để tính tổng l
sử dụng và hiển thị qu Water Consumption. Tất cả dữ liệu đ c đồng
bộ hóa và liên tục cập ệ thống có thể đánh giá và
dụng theo thời gian.
Để đảm bảo tính an tơn và phòng ngừa sự cố, hệ thống cũng đ
hiện bất thường thônode function 14, function 15, DCWarning và
WARNING. Khi phát hiện lưu lượng nước tăng đột ngột, tiê
thường hoặc chênh lệc đầu vào và đầu ra vư
lập tức kích hoạt cảnh báo gửi đến giao diện người dùng và đ
ngắt thiết bị (bơm, vaMQTT về lại ESP 32. Việđảm bảo ngăn
ngừa các tình huống rị hoặc tiêu thụ điện không kiểm soát.



Hình 4.6 Luồng tính lượng nước tiêu thụ ngày/tuần/

Một cụm luồng đặc bi khác như hình 4.6 đóng vai trò quan tviệc giám
sát dài hạn là cụm các5, function 7, function 9, v Cụm
này thực hiện nhiệm v toán tổng lượng nước tiêu thụ theo từng
tuần, và tháng. Cụ thể lấy dữ li ệu thời gian thực từ v2.0_flow_data
để tính tổng lượng nu function 7 xử lý dữ liệu của từng tuần,
function 9 phân tích theo tháng, và function 11 có thể được sử
các chỉ số chu kỳ hoặnăng mở rộng. Các giá trị sau xử ý được đưa tới các
node function 6, function 8, function 10, function 12 để tổng
quả.
Đặc biệt, luồng xử lý ıy còn được kết nối với node Total và các
mục đích hiển thị tổnlượng nước tiêu thụ theo từng khoảng th gian cụ thể. Kết
quả cuối cùng được trưới dạng biểu diễn chu kỳ có gắn nh

gian như ngày, tuần, tỗ trợ rất tốt cho việc trực quan hóa số li
phát hiện xu hướng, vục vụ mô hình AI trong vi liệu lịch
sử. Nhờ cơ chế ghi nh chu kỳ hóa, hệ thống có thể dễ d
khi người dùng yêu c thụ trong bất kỳ khoảng thời gian n
Bên cạnh đó, các chuy còn giúp xác định khi nào có sự tăng
nước tiêu thụ và hỗ trnh bảo trì hệ thống lọc nước định kỳ.



Hình 4.7 Luồng liên kết AI

Một trong những luồnxử lý quan trọng nhất trong hệ thống thiện trong hình 4.7
là sự tích hợp giữa N -RED và nền tảng trí tuệ nhân tạo (Alỗ trợ phân
tích, dự đoán và điều Quá trình này được thực hiện thông qua s
phối hợp của ba nhómode chính: Lệnh Print, AIControl, AIWARNING, và State
Check, kết hợp với hệ thống xử lý AI viết bằng Python và nền
ngôn ngữ như Gemini.

Cụ thể, node Lệnh Prit trong Node -RED có chức năng khởi tạphân tích,
bằng cách gửi bản tin MQTT tới một topic được đăng ký sẵn l
này đóng vai trò như yêu cầu hệ thống AI (được xây dựng bằ
thực hiện việc tổng hợ và xuất ra các biểu đồ phân tích chuyên
tiêu thụ nước theo thời hiện rò rỉ, nhận diện hành vi tiêu th
hoặc phân tích theo mùa vụ. Sau khi AI thực hiện phân tích v
kết quả sẽ được chuyểhatbot ngôn ngữ Gemini để xử lý trực quan hóa
nâng cao. Gemini sẽ đ tự nhiên hơn bằng ngôn ngữ con ng
(như mô tả xu hướng, ờng, gợi ý bảo trì…), giútiếp
nhận và hiểu thông tiri dùng. Các nội dung đã được xử lý n

được gửi ngược lại qua MQTT và hiển thị trong giao diện Home Assistant thông qua node AIWARNING. Node này có nhiệm vụ nhận kết quả phản hồi từ AI và hiển thị trực quan các cảnh báo, nhận định hoặc khuyến nghị ngay tại giao diện người dùng, góp phần nâng cao khả năng giám sát và tương tác.

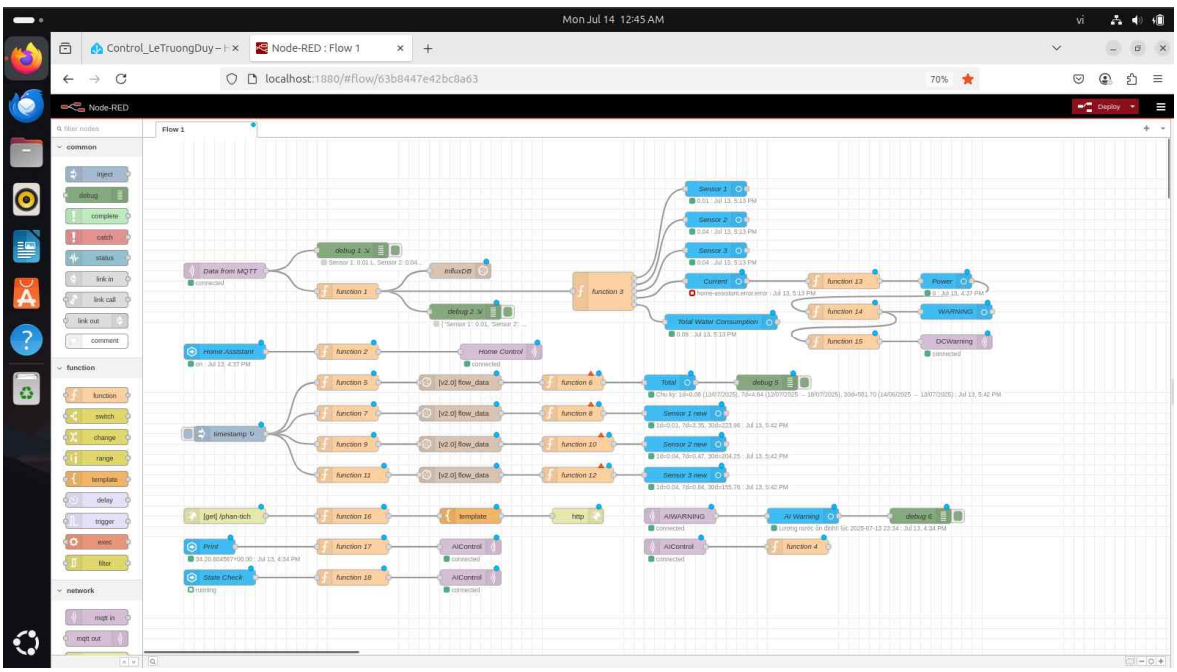
Bên cạnh chức năng phân tích và phản hồi, hệ thống còn tích hợp cơ chế phản ứng thông minh với bất thường thông qua node State Check. Khi hệ thống AI phát hiện một sự cố bất thường như rò rỉ nước, chênh lệch dòng chảy lớn, hoặc tiêu thụ điện năng tăng đột biến, nó sẽ tự động gửi email cảnh báo tới người dùng. Cảnh báo này bao gồm cả thông tin hiện tại của hệ thống và biểu đồ từ quá trình phân tích. Trên giao diện Home Assistant, người dùng có thể phản hồi bằng cách nhấn vào nút xác nhận được kết nối với node State Check. Nếu người dùng xác nhận rằng họ đã tiếp nhận cảnh báo và đang kiểm tra sự cố, hệ thống AI sẽ hiểu rằng việc điều khiển không cần thiết và sẽ hủy lệnh đóng van điện từ. Tuy nhiên, nếu sau 5 phút kể từ khi gửi cảnh báo mà không nhận được phản hồi xác nhận từ phía người dùng, hệ thống AI sẽ chủ động gửi lệnh điều khiển qua topic AIControl, ra lệnh cho Node-RED tự động ngắt van điện từ nhằm đảm bảo an toàn, giảm thiểu nguy cơ thất thoát nước hoặc hỏng thiết bị.

Cơ chế phối hợp này thể hiện một hệ thống có tính thích ứng cao, nơi AI không chỉ đóng vai trò phân tích mà còn ra quyết định theo ngữ cảnh, đồng thời vẫn để lại cơ hội cho người dùng can thiệp thủ công nếu cần thiết. Việc sử dụng kết hợp giữa phân tích bằng mô hình học máy, xử lý ngôn ngữ tự nhiên bằng Gemini và phản hồi MQTT real-time qua các node chuyên biệt trong Node-RED đã xây dựng nên một hệ thống điều khiển và giám sát không chỉ thông minh mà còn linh hoạt và an toàn trong quá trình vận hành thực tế.

Bảng 4.1 Chức năng trừng Node trong luồng

Chức năng	Mô tả	Node chính
Thu thập dữ liệu	Nhận dữ liệu từ cảm biến qua MQTT	mqtt in, function 1
Xử lý logic	Phân tích dữ liệu cảm	function 3, 5, 7, 9, 11, 13, 14

	biết tính toán	
Theo dõi chu kỳ	Tính lượng nước tiêu thụ theo chu kì	function 5 → 6, 7 → 8, 9 → 10
Cảnh báo	Kiểm tra bất thường gửi cảnh báo	function 15, DCWarning
Lưu trữ	Lưu dữ liệu vào cơ sở dữ liệu	InfluxDB
Tích hợp AI	Gửi dữ liệu đến mô hình AI, nhận phản hồi	function 17, 18, AIControl, AIWARNING, Print, State Check
Giao tiếp Home Assistant	Hiển thị và điều khiển từ dashboard	Home Control, Sensor 1 2 3, Total, Current, Total Water Consumption, Power, WARNING, Sensor 1 2 3 new



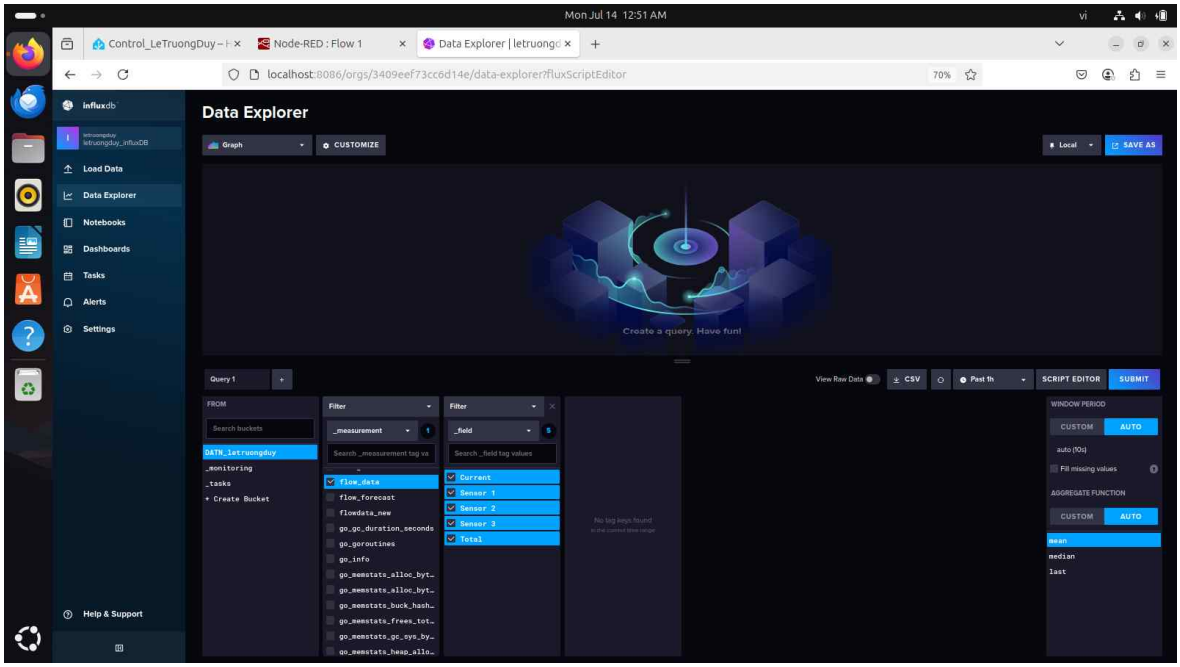
Hình 4.8 Luồng Node-Red hoàn chỉnh

4.2 Thiết lập cơ sở dữ liệu InfluxDB

Trong hệ thống giám sát và tối ưu hóa sử dụng nước, InfluxDB được sử dụng làm nền tảng cơ sở dữ liệu thời gian thực (time-series) nhằm lưu trữ dữ liệu từ các cảm biến như lưu lượng nước, dòng điện, tổng lượng nước tiêu thụ, trạng thái van điện từ, và các chỉ số phân tích khác. Việc sử dụng InfluxDB giúp hệ thống theo dõi chính xác sự biến đổi của các giá trị theo thời gian, phục vụ hiệu quả cho việc giám sát, phân tích và trực quan hóa.

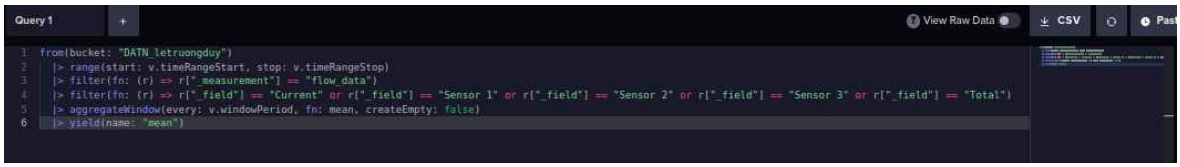
Phiên bản được sử dụng trong hệ thống là InfluxDB v2.x, với giao diện trực quan Data Explorer cho phép người dùng truy vấn, lọc và hiển thị dữ liệu một cách linh hoạt. Trong hệ thống, tất cả dữ liệu được lưu vào bucket có tên DATN_letruongduy. Các bản ghi dữ liệu được tổ chức dưới dạng measurement như flow_data, flow_forecast, và các field bao gồm Sensor 1, Sensor 2, Sensor 3, Current, Total, tương ứng với dữ liệu lưu lượng nước tại các điểm khác nhau, dòng điện tiêu thụ và tổng lưu lượng nước đã dùng.

Node-RED sử dụng node influxdb out để ghi dữ liệu vào InfluxDB. Trong node này, người lập trình cấu hình các thông số như URL máy chủ (localhost:8086), tổ chức (organization), token xác thực, và tên bucket (DATN_letruongduy). Dữ liệu được xử lý qua các node như function 3, function 13, function 14 sẽ được định dạng thành JSON có cấu trúc chuẩn và gửi vào InfluxDB kèm theo dấu thời gian (timestamp).



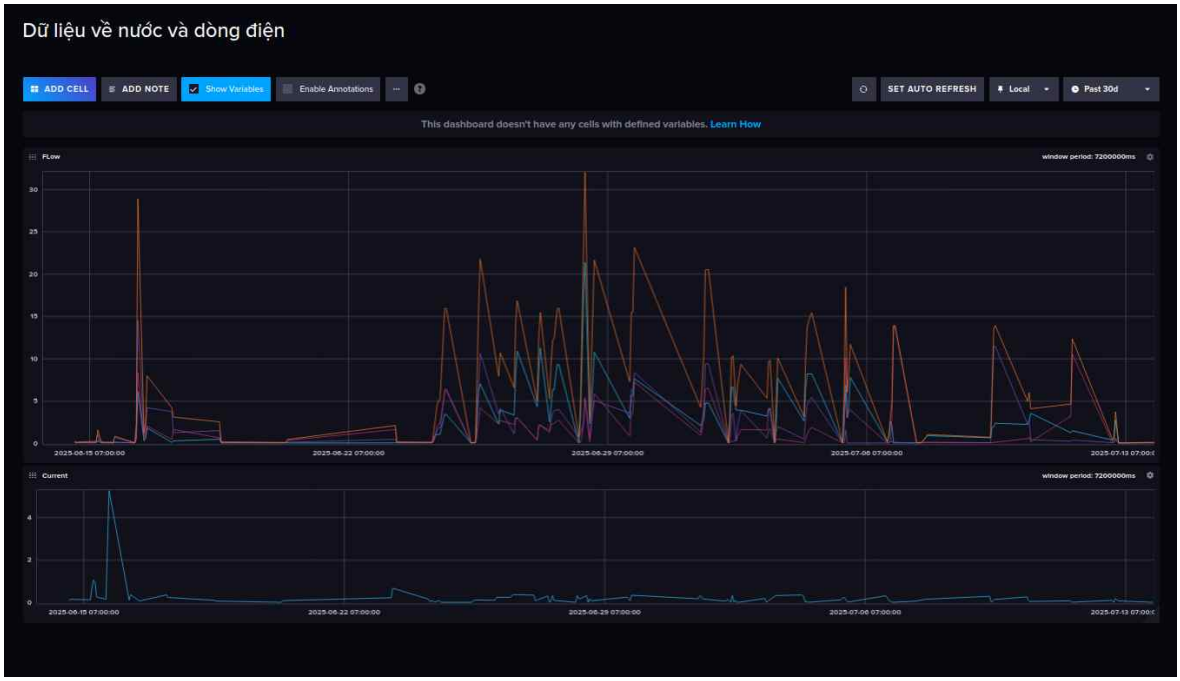
Hình 4.9 Thiết lập InfluxDB Explorer

Để phục vụ việc trực can hóa và thống kê, người dùng có thể thụ
liệu trực tiếp thông qua công cụ Data Explorer hình 4.9. Trong hệ thống, một truy
vấn cơ bản được sử dụiệu cảm biến như sau:



Hình 4.10 Truy vấn dữ liệu Influx bằng Flux

Hình 4.10 cho thấy âu truy vấn trên thực hiện việc lấy t cả dữ liệu thuộc
measurement flow_data trong khoảng thời gian do ngu
(v.timeRangeStart đến Sau đó, hệ thống lọc các field cần thiết
bao gồm Current, Sens Sensor 2, Sensor 3, và Total. Đây là nh hông số cốt
lõi để đánh giá hiệu soạt động của hệ thống nước. Truy vấn t
hàm aggregateWindow() với tham số mean để tính giá trị tı
khoảng thời gian, giúm mượt dữ liệu và loại bỏ nhiễu. Dữ liệ
xuất ra với nhãn mean thể hiển thị bằng biểu đồ đường, bảng số
ra file CSV.



Hình 4.11 Dữ liệu được lưu trữ trong Influx dạng bi

Việc truy vấn này khười dùng dễ dàng theo dõi tình hình tiêu thụ nước theo thời gian th như hình 4.11, mà còn là nền tảng để hai thác dữ liệu lịch sử, phục v các tác vụ phân tích xu hướng, dự báo h phát hiện rò rỉ. Đồng ừ InfluxDB cũng được sử dụng để sinh biểu đồ tổng hợp thông qua nde Lệnh Print trong Node -RED và gửiảng AI để phân tích chuyên sâu bằng mô hình ngôn ngữ tự nhiên Gemini.

4.3 Giao diện hiển th Assistant

Giao diện hiển thị củ Assistant được xây dựng nhằm cung cấ người dùng một cái nhìn trự thao tác đối với toàn b sát và điều khiển nướ qua dashboard được thiết kế t người dùng có thể the các cảm biến, trạng thái hoạt động của t bị, thực hiện các hànlển thủ công và nhận các cảnh báo từ AI cách rõ ràng và thuận

Khác với nhiều nền tảợ kéo -thả giao diện, Home Assistt không hỗ trợ xây dựng giao diện bằng tao tác kéo thả đơn thuần. Để tạo ra các g trực quan, đẹp mắt và có độ tùyìng cần viết mã cấu h

ngôn ngữ YAML. YAngôn ngữ đánh dấu có cấu trúc đơn gi
mẽ, được sử dụng trone Assistant để định nghĩa cách bố trí giao diện, hiển thị
các thực thể (entity), tchức các thẻ hiển thị (card), thiết lập điều kiện hiển thị và
liên kết dữ liệu từ các biến. Với YAML, lập trình viên có th
layout, phối màu, biểuthông tin, bảng dữ liệu, biểu thị trạng thá
và liên kết trực tiếp tới trang web nhúng (iframe) hoặc biểu đồ động.
Việc xây dựng dashbcd bằng YAML giúp hệ thống đạt được t
dễ mở rộng, và hoàn y biến theo yêu cầu của từng dự án cụ th
biệt phù hợp với các h giám sát như đề tài này. Đây cũ
của
Home Assistant khi so sánh với các nền tảng giao diện giám sát
Thống kê tiêu thụ nướ như hình 4.12, hệ thống hi thị bảng tổng hợp
lượng nước tiêu thụ th ày, 7 ngày và 30 ngày, được chia theo từng
cảm biến (Sensor 1, Sensor 2, Sensor 3) và Total. Dữ liệu này đu -
RED và gửi đến Homt qua MQTT, sau đó hiển thị trong các entities card
hoặc grid card. Đây làu vực ngườ i dùng theo dõi nhiều nhất để
đánh giá hành vi sử dụn và dài hạn.

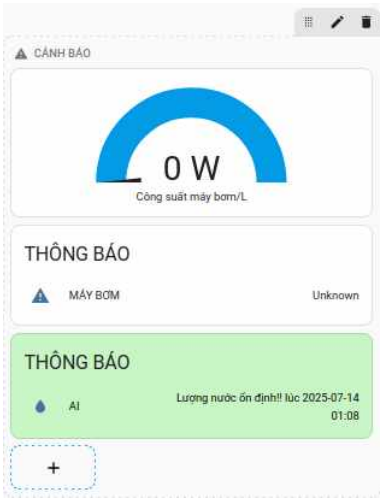
Hình 4.12 Lượng nước tiêu thụ theo ngày/tuần/tha

Điều khiển thủ công tlt bị hình 4.13, bên phải bảng thống kê điều khiển
Solenoid valve 1 và Solenoid valve 2 tương ứng với hai van đ
ON/OFF, Home Assistant sẽ gửi bản tin điều khiển qua MQTT -RED xử lý

và kích hoạt relay. Cc thiết kế bằng button -card hoặc toggle và có thể thay đổi giao diện, biểvà màu sắc để phù hợp với tình tr

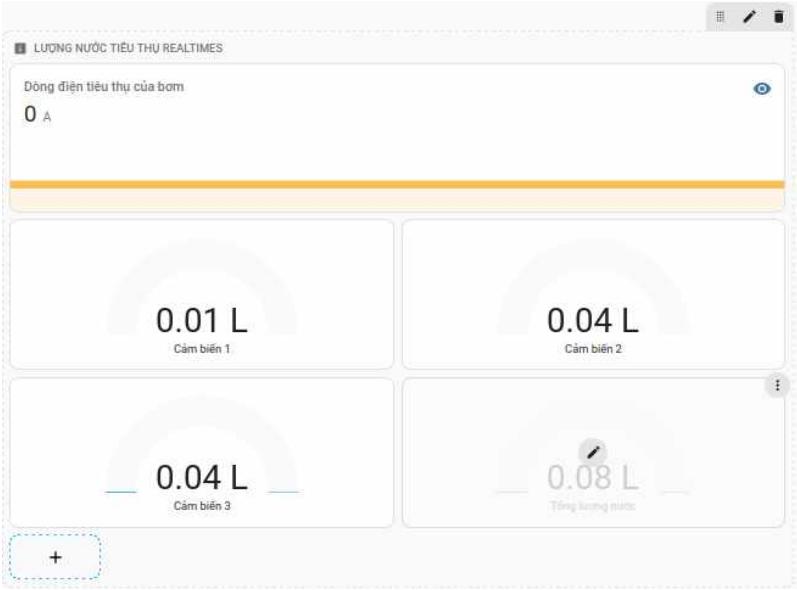
Hình 4.13 Nút điều khiển thủ công

Giám sát hoạt động b hình 4.14. Sử dụng gauge card để hin thị công suất của máy bơm theo đơn vị iện theo đơn vị A (ampe). Thông số n được tính toán trong Ne -RED và truyền về Home Assistant, ùng biết khi nào bơm hoạt độêu thụ điện hiện t ại. Cảnh báo hệ thống bao gồm các khu vực hiểnhị thông báo máy bơm, thông báo từ AI v từ người dùng. Các thy có thể là chuỗi văn bản từ Node -RED gửi về qua MQTT, hoặc giá trị tnsor xử lý cảnh báo có/không. M c và biểu tượng được thiết lập gbiệt giữa trạng thái bình thường v



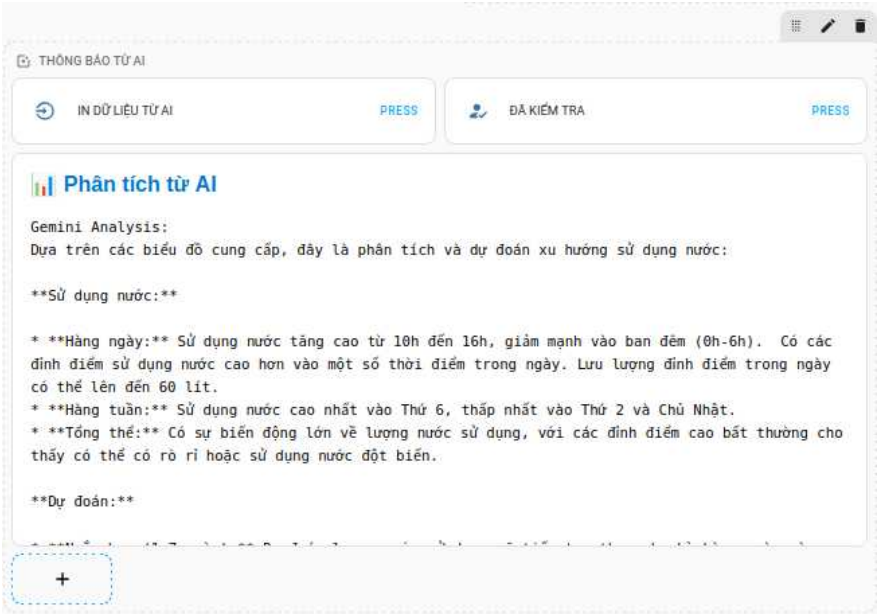
Hình 4.14 Giám sát hoạt động bơm và thông báo

Biểu đồ và thông tin chi tiết theo thời gian thực hình 4.15. Tại đây, dữ liệu của từng cảm biến (Cảm biến 1,Cảm biến 2, v.v.) được cập nhật trực tiếp thực. Ngoài ra, các biểu diễn dòng điện hoặc lưu lượng cũng c thêm vào bằng historysor card, hoặc tích hợp với Grafana.



Hình 4.15 Ghi nhận dữ liệu lượng nước và dòng đ

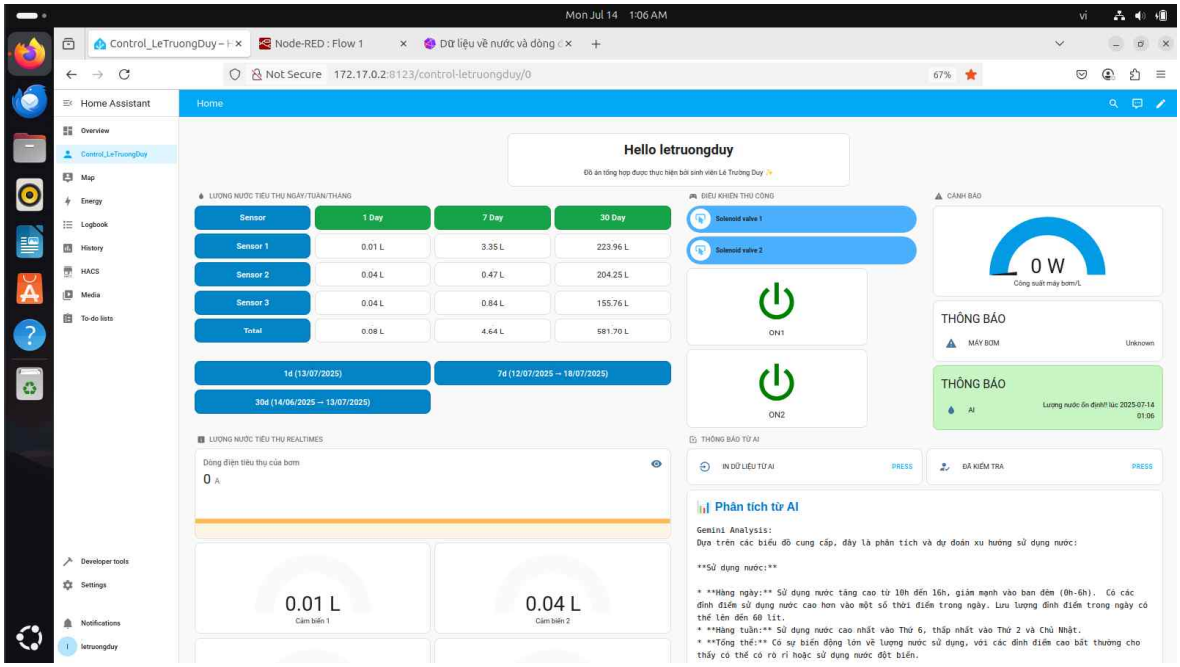
Vùng hiển thị phân tí trong giao diện Home Assistant hình 4.16 không sử dụng markdown cardnhư các nội dung tĩnh thông thường, m trang web động do Noe -RED sinh ra. Trang web này được xây node như http in, template, http response trong Node-RED. Dữ ân tích từ AI (cụ thể là mô hình ngGemini) được gửi về Node -RED qua MQTT, sau đó được xử lý và kết xuất giao diện web có định dạng H Nội dung của trang web bao gồm ông minh như xu hướng ti ngày, giờ cao điểm sử o sánh giữa các ngày trong tuần, dự báo đặc biệt là cảnh báo v thường hoặc nghi ngờ có rò bộ nội dung này được trình bày si ới dạng bảng, biểu đồ, đoạn văn mô tả, quan và màu sắc nổi b HTML và CSS.



Hình 4.16 Vùng hiển thị và tương tác với AI

Sau khi được tạo, tran này có thể được truy cập thông qua địa c
mạng cục bộ. Trong go diện Home Assistant, lập trình viên sử
nhúng trực tiếp trang dashboard. Việc này cho phép n
AI luôn được cập nhật tự động, đồng thời người dùng có thể t
thời gian thực dưới d động, mà không bị giới hạn bởi định d
YAML hay markdown đơn giản. Cách tiếp cận này giúp nân
chuyên nghiệp cho hệ g t hời phát huy tối đa năng lực của Node -RED
trong việc dựng giao o động và khả năng hiển thị dữ liệu AI m
lạc, dễ hiểu.

Nút xác nhận từ ngườ c ác nút “In dữ liệu từ AI” và
để xác nhận rằng ngư báo và không cần động.
Các nút này có thể đu t_button hoặc script, v
chúng được Node-RED đọc qua node event state.



Hình 4.17 Giao diện Home Assitant hoàn chỉnl

Hình 4.17 biểu diễn m giao diện điều khiển và hiển thị Home A

4.4 Tích hợp trí tuệ ạo (AI) vào hệ thống

Hệ thống sử dụng trí ệ nhân tạo để phân tích dữ liệu tiêu thụ n
phát hiện các bất thườ rỉ nước hoặc hành vi tiêu thụ đột biến),
đưa ra dự đoán và cảnến người dùng. Các mô hình AI
khai sử dụng thư viện n Forest, kết hợp với nền tảng ngôn ngữ t
nhiên Gemini để tạo ra phản hồi dễ hiểu và trực quan. Hệ th
toàn tự động, kết nối QTT để gửi và nhận tín hiệu điều khiển
phần phần mềm và ph

Bản 4.2 Bảng so sánh Prophet và Isolation Forest

Thuộc tính	Prophet	Isolation Forest
Loại học máy	Học có giám sát (Supervised)	Học không giám sát (Unsupervised)
Ứng dụng chính	Dự báo chuỗi thời gian	Phát hiện bất thường
Cơ chế	Mô hình cộng (trend	Cây phân chia ngẫu nl

	+ seasonality)	
Huấn luyện	Có (fit)	Có (fit nhưng không c
Dữ liệu đầu vào	ds (time), y (giá trị)	Dữ liệu số
Đầu ra	Dự oán yhat, CI	Nhãn 1/-1 (bình thườmg)

4.4.1 Dữ liệu đầu và(

Dữ liệu đầu vào đóng rọng trong việc cung cấp cơ s học hỏi và đưa ra dự đoán chính ɔng hệ thống này, dữ liệu được truy vấn liệu thời gian thực InuxDB, nơi lưu trữ các thông số đo đượ trong vòng 30 ngày gống lấy dữ liệu các tr "Sensor 2", và "Sensor 3" với tần suất mỗi 5 phút một mẫu, to thành chuỗi thời gian đều đặn để pic vụ dự báo.

```
# ======= LẤY DỮ LIỆU 30 NGÀY =======
def get_data():
    query = f'''
    from(bucket: "{bucket}")
      |> range(start: -30d)
      |> filter(fn: (r) => r._measurement == "flow_data")
      |> filter(fn: (r) => r._field == "Total" or r._field == "Sensor 2" or r._field == "Sensor 3")
      |> aggregateWindow(every: 5m, fn: mean, createEmpty: false)
      |> pivot(rowKey:["_time"], columnKey: ["_field"], valueColumn: "_value")
      |> yield(name: "mean")
    '''
```

Hình 4.18 Code truy vấn dữ liệu InfluxDB

Hàm get_data() là bư cline xử lý dữ liệu, ược thiết kế để truy vấn và chuẩn hóa dữ an từ InfluxDB - một cơ sở di ưu hóa cho dữ liệu chuỗi thời gian như hình 4.19. Trước hết, nó định nghĩa ruy vấn Flux để lấy dữ liệu từ bucket tày gần nhất. C âu truy vấn sẽ lọc các bản ghi chỉ liên quan đến chính: "Total", "Sensor 2" v Sau đó, sử dụng aggr eWindow để lấy trung bình mỗi 5 phút, giúp giảm nhiễu và chuẩn hóa thời gian

```
df = query_api.query_data_frame(query)
df.rename(columns={'_time': 'Time'}, inplace=True)
df['Time'] = pd.to_datetime(df['Time'], utc=True).dt.tz_localize(None)
df.sort_values('Time', inplace=True)
df.reset_index(drop=True, inplace=True)

df.set_index('Time', inplace=True)
start_time = df.index.min().floor('5min')
end_time = df.index.max().ceil('5min')
full_range = pd.date_range(start=start_time, end=end_time, freq='5min')
df = df.reindex(full_range).rename_axis("Time").reset_index()

for col in ['Total', 'Sensor 2', 'Sensor 3']:
    df[col] = df[col].fillna(0)
    df[col] = df[col].apply(lambda x: max(x, 0))
return df
```

Hình 4.19 Chuẩn hóa thời gian dữ liệu

Tiếp theo, như hình 4. pivot chuyển đổi định dạng dữ liệu từ (nhiều bản ghi cùng thời gian trường) sang dạng ngang (mỗi h timestamp với đầy đủ phục vụ tốt cho học máy. Ph ần còn lại xử lý thời gian: rename, sort, reset_index, tái định dạng về định dạng dat có timezone để tương t. Cuối cùng, dữ liệu được chuẩn hóa để bảo không có giá trị án (vì lưu lượng nước không thể âm) và ếu (NaN) được điền bằng 0. Hàame đã được làm sạch, sẵn s vào mô hình học máy.

4.4.2 Triển khai mô h Prophet

Prophet là mô hình dựdo Facebook phát triển, đ dự đoán lưu lượng nư ngày. Dữ liệu cảm biếđược chuyển đổi sang định dạng pù hợp với Prophet (ds, y), sau đó mô hìn với các thành phần m và theo tuần. Prophet trả về các giá trị đoán (yhat) cùng với at_lower, yhat_up per). Ngoài việc vẽ biểu đồ dự đoán, hệ thống òn khai thác các thành phần mùa vụ như điểm trong ngày, ngàyhiều nhất trong tuần, và từ đó đ về thói quen sử dụng.

```python
def run_prophet_forecast(df, forecast_days, timestamp):
    df_prophet = df.rename(columns={"Time": "ds", "Total": "y"})
    df_prophet['ds'] = df_prophet['ds'].dt.tz_localize(None)

    model = Prophet(daily_seasonality=True, weekly_seasonality=True)
    model.fit(df_prophet)

    future = model.make_future_dataframe(periods=forecast_days * 144, freq='10min')
    forecast = model.predict(future)
    ...
    return model, forecast
```

Hình 4.20 Huấn luyện mô hình Prophet

Hàm run_prophet_forecast() trong hình 4.22 chịu trách nhiệm ấn luyện mô hình Prophet và tạo dữ liệrophet yêu cầu dữ liệu đầu v (timestamp) và y (giá trị cần dự báo). Do đó, hàm thực hiện đổTime thành ds và Total thành y. Prophet được cấu hình để tính mùa vụ hàn nhằm học được thói quướ c có tính lặp theo giờ và r Hàm make_future_dataframe() tạo ra tập dữ liệu thời gian trong ng lai, mỗi ngày có 144 điểm (vì mỗi đphút). Sau đó, phương th DataFrame với các cộdự đoán trung bình), yhat_lower và yha (khoảng tin cậy), giúpđịnh vùng giá trị hợp lý. Prophet không c ự báo mà còn phân tách chuỗi thành các thành phần theo mùa vụ và kiện phân tích sâu hơrKết quả mô hình được l các bước sau như phát và phân tích bằng AI ngôn ngữ.

4.4.3 Triển khai mô h Iso lation Forest

Isolation Forest là mô hình phát hiện bất thường được sử dụng liệu có sai khác lớn s với dự đoán từ Prophet. Sau khi tính phần dư giữa dữ liệu thực tế và dự đoán (reit toán Isolation Forest xác định các điểm ng hi ngờ là bất thường. Ngoài sử dụng logic kiểm tra khoảng tin cậy (confidence interval) và đặc biệt là phát hiện rò rỉ bằng cách xe thời gian lưu lượng nđịnh và rất nhỏ vào ban đêm – thời điểm không nên có hoạt độNhững thời điểm nghi ngờ đư cảnh báo thông qua email và MQTT nếu chưa từng được gửi trư

```
model_if = IsolationForest(contamination=0.005, random_state=42)
df_merged['anomaly_score'] = model_if.fit_predict(df_merged[['residual']])
df_merged['is_anomaly'] = df_merged['anomaly_score'] == -1
```

Hình 4.21 Hàm tính toán bất thường

Isolation Forest là thu y không giám sát chuyên dùng để phát hiện các điểm bất thường (ouip dữ liệu. Trong hệ thống n hần dư (residuals) giữa dữ liệdự báo từ Prophet được tính toán bằng c hiệu residual = Total – yhat hình 4.24. Những phần dư này đượ biểu hiện cho sai lệch hành vi tiếc.

Khi truyền phần dư vàn Forest, thuật toán sẽ tự động phát hiện các điểm "bị cô lập" dạ nhất, tức là những giá trị quá khác biệt s lại của dữ liệu. Các điểm này được đánh dấu là bất thường b -1, sau đó được chuyển thành cá trị True/False. Việc sử dụng th điều kiện khác như vượt ngoài khoảng tin cậy dự báo) v is_leak_like (nghi rò rỉ khi có xu hướng tăng đều trong giờ tiế chế phát hiện đa tầng,iúp hệ thống không chỉ nhạy bén với đột biến mà còn với các tình huống tăng ch – đặc trưng của rò rỉ.

4.4.4 Giao tiếp giữa ệ thống

Việc tích hợp AI vớiệ thống được thực hiện thông qua giao tiếp MQTT. Khi người dùng trong Hoistant nhấn nút Print, một lệnh sẽ đ AIControl, Node-RED chuyển lệnh này đến AI. Khi nhận đượ lấy dữ liệu, dự báo bằg Prophet, phát hiện bất thường bằng Iso các biểu đồ như: dữ liệu gy, dự báo tương lai, mùa v biểu đồ bất thường từnt cả biểu đồ được chuyển cho nền tảng G phân tích ngôn ngữ, ắng văn bản mô tả xu h thường và cảnh báo dễ hiểu. Kết quả này được gửi lại qua MQ để Node-RED hiển thịb nhúng trong Home Assistant.

Đây là phần thiết lập MQTT để AI có thể giao tiếp với các thiết bị IoT hoặc hệ thống giám sát trung tâm. MQTT là giao thức nhắn tin nhẹ thiết kế cho hệ thống nhúng và các mạng thông thấp. Trong đoạn mã MQTT được tạo với giao thức 3.1.1 như hình 4.26, sau đó đăng ký các hàm callback để xử lý như kết nối thành công (on_connect), nhận tin nhắn (on_message), hoặc gửi dữ liệu (on_publish).

Phần cấu hình TLS đảm bảo mã hóa trong quá trình truyền tải, phù hợp với yêu cầu bảo mật của các hệ thống IoT hiện đại. Sau khi thiết lập người dùng và mật khẩu, client kết nối tới máy chủ HiveMQ và bắt đầu vòng lặp xử lý không đồng bộ (loop_start). Đây là cơ sở để AI có thể tự động gửi lệnh điều khiển, gửi cảnh báo, và nhận phản hồi từ người dùng hoặc hệ thống khác một cách thời gian thực.

```
# Khởi tạo MQTT client và kết nối
client_mqtt = mqtt.Client(protocol=mqtt.MQTTv311)
client_mqtt.on_connect = on_connect
client_mqtt.on_message = on_message # This is the unified on_message
client_mqtt.on_publish = on_publish
client_mqtt.tls_set(cert_reqs=ssl.CERT_NONE)
client_mqtt.tls_insecure_set(True)
client_mqtt.username_pw_set(mqttUser, mqttPassword)
client_mqtt.connect(mqttServer, mqttPort, 60)
client_mqtt.loop_start()
```

Hình 4.22 Hàm khởi tạo MQTT

4.4.5 Phản hồi và rào tự động

Sau khi phát hiện bất thường, hệ thống AI gửi ngay một cảnh báo đến người dùng đăng nội dung cảnh báo là AIWARNING. Giao diện Home Assistant sẽ hiển thị thông báo và yêu cầu người dùng xác nhận bằng cách kiểm tra. Nếu trong vòng 5 phút người dùng không phản hồi, một tiến trình tự động gửi lệnh OFF1 hoặc OFF2 đến MQTT để tắt van điện tương ứng (Sensor 2 hoặc Sensor 3). Điều này giúp hệ thống chủ động phản ứng với các bất thường, giảm thiểu tổn thất khi người dùng không kịp

```
def handle_anomaly_or_leak(sensor_name, is_leak):
    ...
    def auto_shutdown():
        if not user_acknowledged[sensor_name]:
            client_mqtt.publish(mqtt_topic, "OFF1" if sensor_name == "Sensor 2" else "OFF2")
    pending_shutdown[sensor_name] = threading.Timer(300, auto_shutdown)
    pending_shutdown[sensor_name].start()
```

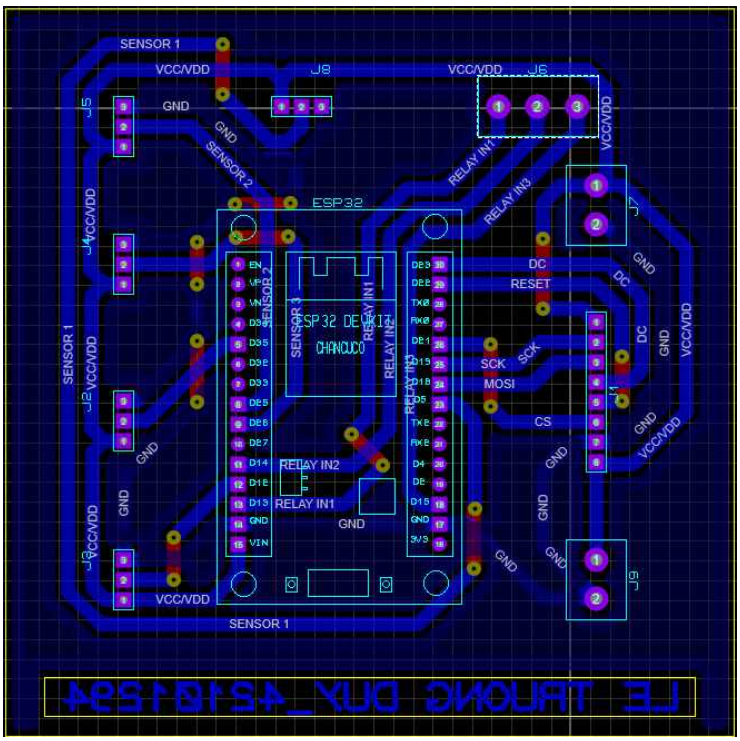
Hình 4.23 Hàm cảnh báo và tắt van tự động

Khi phát hiện bất thường hoặc nghi ngờ rò rỉ, hệ thống AI sẽ
qua email và MQTT như trong hình 4.28. Tuy nhiên, để đảm trong
trường hợp không có can thiệp của con người, hệ thống triển kh
tắt thiết bị. Trong đoạn mã trên, một Timer được khởi tạo với chờ 300
giây (5 phút). Nếu sau người dùng không gửi phản hồi "đ
qua MQTT, hàm auto_shutdown() sẽ tự động được gọi để gửi l bị tương
ứng (OFF1 hoặc OFF2).

Cơ chế này hoạt động hệ thống giám sát chủ động, giảm thiểu nguy cơ thất
thoát nước hoặc hỏng thiết bị khi không có người giám sát trực ti
email được thực hiện SMTP sử dụng smtplib và cấu trúc MIMEText
tải nội dung một cách chuẩn hóa. Tính năng phản hồi này là một lõi để hệ
thống trở nên thực tiễn khả năng vận hành liên tục.

CHƯƠNG 5. THI CÔNG VÀ THỬ NGH

5.1 Thiết kế PCB



Hình 5.1 PCB

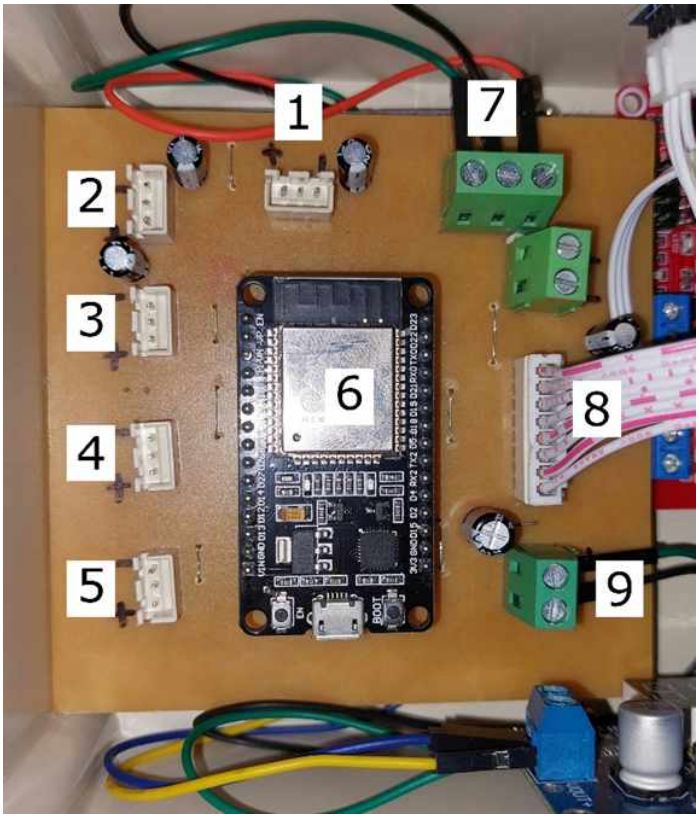
Hình 5.1 là một bảng mạch PCB,được thiết kế một cách khoa học với đầy đủ các cổng kết nối và bố trícể đảm bảo sự ổn định v ng quá trình vận hành cũaPCB, có hai đầu nối Domino loại 2 chân, trong đó: Domino J9 đóng vai trò là ngõ vào cấp điện áp 5VD ho toàn bộ mạch, được lấy từ máp LM2596S; Domino J7 là ngõ ra dùng để cấp nguồn rio module Relay, giúp tách biệt nguồn c nguồn tải nhằm tránh ıu và đảm bảo hoạt động ổn định cho relay Ngoài ra, Domino J5 là đầu nối loại 3 chân, trong đó một chân được dùng làm ngõ tín hiệu (Signal) từ vi khiển ESP32, truyền trực tiếp đến chân điều khiển Input của module Relay để tực hiện việc đón g/ngắt thiết bị theo lập t. Các cảm biến khác trc thống được thiết kế sử dụng dây EJS, giúp chuẩn hóa kết nối và thuận tiện ł tháo lắp, thay thế hoặc triển khai mở rc

thống. Cụ thể, các cổng kết nối cảm biến bao gồm: J4, J5 và J8: là các cổng kết nối dành cho ba cảm biến lưu lượng nước, cho phép giám sát lưu lượng tại các nhánh khác nhau trong hệ thống. J2: là cổng kết nối với cảm biến đo dòng điện, phục vụ cho việc theo dõi mức tiêu thụ điện năng. J3: được sử dụng để kết nối với cảm biến hồng ngoại, nhằm phát hiện vật thể phục vụ mục đích điều khiển bơm nước hoặc mở van.

Ở trung tâm của bảng mạch là vi điều khiển ESP32, được xem như bộ não của hệ thống, đảm nhận vai trò tiếp nhận tín hiệu từ các cảm biến, xử lý dữ liệu, điều khiển các thiết bị đầu ra như relay, đồng thời truyền thông tin lên các nền tảng hiển thị hoặc điều khiển từ xa.

5.2 Mạch cứng hoàn chỉnh

Hình 5.2 mô tả mạch cứng đã được thi công hoàn chỉnh, được thiết kế và bố trí một cách tối ưu trên một bảng mạch in có kích thước 92.5mm x 92.5mm. Hệ thống sử dụng nguồn điện một chiều 12VDC làm đầu vào, sau đó được hạ áp xuống 5VDC thông qua module chuyển đổi điện áp LM2596S.



Hình 5.2 Mạch cứng hoàn chỉnh

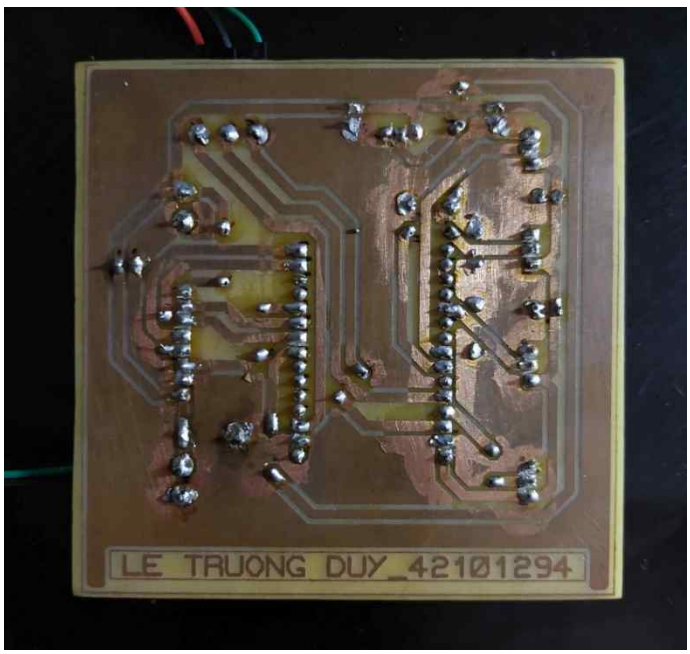
Nguồn 5VDC sau khi c hạ áp sẽ được phân phối đến các th
gồm: vi điều khiển ES32 – đóng vai trò trung tâm điều khiển a
cảm biến lưu lượng n – dùng để đo lưu lượng nước tiêu tl
nhau, một cảm biến đđiện – dùng để giám sát lượng đa
hệ thống, một cảm biế hồng ngoại – phát hiện vật thể để điều hiển quá trình bơm
nước tự động, một mole Relay – thực hiện chức năng đóng/ắt thiết bị điện, và
cuối cùng là màn hình TFT – đảm nhận vai trò hiển thị c
lượng nước, điện năng và thông tin bảo trì lõi lọ c.
Toàn bộ các linh kiện được kết nối và sắp xếp hợp lý tr
mạch nhằm đảm bảo h ổn định, dễ dàng thi công, và thuận ti
hay mở rộng hệ thống tương lai.

Bản 5.1 Bảng chú thích các linh kiện trong mạ

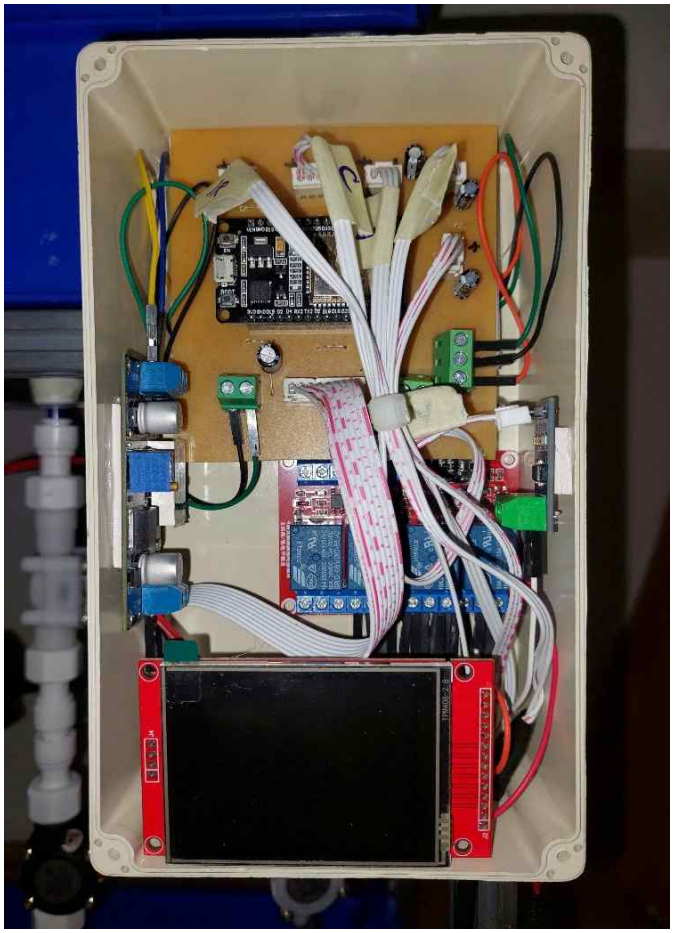
STT	Chú thích
1	Cảm biến lưu l YF -S401 (Sensor 1)

2	Cảm biến lưu lượng nước YF-S201 (Sensor 2)
3	Cảm biến lưu lượng nước YF-S201 (Sensor 3)
4	Cảm biến dòng điện ACS712
5	Cảm biến hồng ngoại OMRON E18-D80NK
6	ESP32 30pin
7	Tín hiệu đầu vào cho modul relay
8	Màn hình LCD LIL9431
9	Nguồn vào 5V

Hơn nữa, bảng mạch được gia công tỉ mỉ với lớp đồng trên PCB có đường dẫn rõ ràng và không xuất hiện lỗi hàn như hình 5.3. Các điểm hàn được thực hiện chắc chắn, hạn chế tối đa nguy cơ lỏng chân linh kiện trong quá trình hoạt động. Mạch cũng được thiết kế với tiêu chí dễ dàng kiểm tra và sửa chữa, khi các module và linh kiện chính đều dễ tiếp cận.



Hình 5.3 Mặt sau của mạch cứng



Hình 5.4 Hộp đựng mạch hoàn chỉnh

Tổng thể, hình 5.4 không chỉ minh họa sự hoàn thiện của mạch mà còn thể hiện tính thẩm mỹ và sự chuyên nghiệp trong quá trình thi công, từ khâu thiết kế trên phần mềm cho đến sản xuất thực tế. Việc kiểm tra và vận hành thử nghiệm sẽ là bước tiếp theo nhằm đảm bảo sản phẩm đáp ứng đầy đủ các yêu cầu và mục tiêu ban đầu. Board mạch cứng hoàn chỉnh được đặt vừa vặn trong hộp đựng bằng mica HÌNH có kích thước 200x120x75mm, hộp này sẽ chứa bảng mạch cứng và các module được dùng trong hệ thống.

5.3 Mô hình hoàn chỉnh

Hình 5.5 thể hiện toàn bộ mô hình hệ thống đã được lắp ráp hoàn chỉnh, bao gồm đầy đủ các thành phần phần cứng cần thiết nhằm mô phỏng một hệ thống giám sát và điều khiển nước thực tế. Mô hình được thiết kế tích hợp cả hộp điều khiển trung

tâm, nơi chứa bảng mạch điều khiển cùng các linh kiện điện tử chính như vi điều khiển, các cảm biến, module relay, màn hình hiển thị và bộ nguồn.

Bên cạnh đó, hệ thống còn bao gồm hai thùng chứa nước có cùng kích thước tiêu chuẩn 34 x 26.5 x 13 cm, với dung tích mỗi thùng xấp xỉ 11.7 lít, đảm bảo đủ thể tích để mô phỏng quá trình lưu trữ và cấp phát nước. Các thùng nước này được liên kết thông qua hệ thống đường ống loại ống nối nhanh RO, giúp việc lắp đặt nhanh chóng, chắc chắn và dễ dàng thay thế khi cần thiết.

Ngoài ra, hệ thống cũng được tích hợp van điện từ và bơm nước mini, cho phép điều khiển dòng chảy một cách tự động thông qua tín hiệu từ trung tâm điều khiển. Tất cả các thành phần nêu trên đều được cố định chắc chắn trên một khung nhôm định hình 2020 có kích thước 55 x 34 x 26.5 cm, giúp mô hình trở nên gọn gàng, chắc chắn, đồng thời nâng cao tính thẩm mỹ, tạo cảm giác chuyên nghiệp và phù hợp để trưng bày hoặc thuyết minh trong các buổi báo cáo, triển lãm, hay hội thảo chuyên đề.

Bên trong hộp điều khiển còn được bố trí hợp lý các jack kết nối nguồn và thiết bị ngoại vi nhằm tối ưu hóa thao tác vận hành và bảo trì. Hệ thống dây dẫn được sắp xếp gọn gàng bằng ống gen nhiệt và các kẹp giữ dây, đảm bảo an toàn điện và tăng độ bền cho mô hình. Mô hình không chỉ đóng vai trò là một bản thu nhỏ của hệ thống thực tế, mà còn là công cụ hiệu quả để đánh giá, trình diễn và thử nghiệm các thuật toán điều khiển, góp phần phục vụ cho quá trình nghiên cứu và học tập.

Hình 5.5 Mô hình hệ thống hoàn chỉnh

5.4 Kết quả thử nghiệm và đánh giá

5.4.1 Thử nghiệm

Sau khi hoàn thiện phần mềm, hệ thống đã được tiến hành
thử nghiệm thực tế trong môi trường giả lập tương đương với hộ
thử nghiệm cho thấy hệ thống hoạt động ổn định, đáp ứng đầy đủ các chức năng đã đề ra.

Cụ thể, hệ thống ghi nhận chính xác lưu lượng nước từ ba cảm
vị trí khác nhau, đồng thời hiển thị thông số lên màn hình LCD TFT theo thời gian
thực. Dữ liệu lưu lượng thu thập từ cảm biến dòng
InfluxDB và trực quan hóa bằng Grafana, giúp người dùng dễ dàng theo dõi lịch sử
sử dụng.

Khi phát hiện có vật thể, cảm biến hồng ngoại, hệ thống tự động kích hoạt bơm
nước, sau đó ngắt khi không có người sử dụng, đảm bảo tiết kiệm
lượng. Ngoài ra, chức năng điều khiển từ xa thông qua giao diện Home Assistant

hoạt động ổn định, cho phép người dùng giám sát và bật/tắt thiết bị ngay trên điện thoại hoặc máy tính kết nối mạng.

Trong thử nghiệm chức năng AI, hệ thống có thể ghi nhận mức tiêu thụ nước theo ngày và tuần, từ đó đưa ra cảnh báo khi có dấu hiệu bất thường, phân tích và dự đoán được xu hướng dùng nước của người dùng. Hệ thống cũng có khả năng phát hiện rò rỉ nước khi có dòng chảy liên tục.

Toàn bộ mô hình được cố định chắc chắn trên khung nhôm định hình, giúp các thành phần hoạt động ổn định, không rung lắc hoặc bị nhiễu tín hiệu. Thử nghiệm trong điều kiện hoạt động liên tục 24 giờ cho thấy nhiệt độ vi điều khiển và nguồn cấp vẫn ở mức an toàn, không xảy ra treo mạch hoặc reset ESP32.

Từ các kết quả trên, có thể khẳng định rằng hệ thống đáp ứng tốt các tiêu chí về giám sát, điều khiển thông minh, tiết kiệm tài nguyên và sẵn sàng triển khai trong thực tế.

5.4.2 Đánh giá thực tế

Bảng 5.2 Số liệu đánh giá thực tế

Lần đo	Sensor1 (L)	Sai số	Sensor2 (L)	Sai số	Sensor3 (L)	Sai số	Current (A)	Sai số
1	1.08	8%	1.0	0%	1.07	7%	0.410	0%
2	1.04	4%	1.1	10%	0.9	10%	0.414	0.4%
3	1.05	5%	0.98	2%	0.9	10%	0.408	0.8%
4	1.08	8%	0.95	5%	0.95	5%	0.407	0.7%
5	1.02	2%	1.02	2%	0.9	10%	0.405	0.5%
Tổng	5.4%		3.8%		8.4%		0.48%	

Bảng 5.2 cho ta thấy được sai số linh kiện sau 5 lần đo. Đối với các cảm biến lưu lượng nước YF-S201 để tính toán lượng nước sử dụng là Sensor 1 2 3 khi cho 1 lít nước đi qua lần lượt mỗi Sensor 5 lần ta có thể thấy tổng sai số của Sensor 1 là 5.4%, Sensor 2 là 3.8% và Sensor 3 là 8.4%. Nếu tính thành tổng lượng nước tiêu thụ và lấy tổng 3 sai số này để tính ra tổng sai số của tổng lượng nước sử dụng sẽ là

5.86% và theo lý thuyết sai số của YF-S201 là 5-10% thì con số 5.4%, 3.8%, 8.4% hay 5.86% của tổng lượng nước tính cả nhiễu cảm biến và dòng điện không ổn định thì đây là một con số hoàn toàn chấp nhận được.

Về phần cảm biến dòng điện lấy dòng điện chuẩn là 410mA được đo chính xác bằng VOM ta thấy sai số qua 5 lần đo đều dưới 1% , xét đến sai số lý thuyết của cảm biến dòng điện ACS712 là 1.5-5% thì sai số của hệ thống gần như là chính xác, đạt độ tin cậy cao.

Tổng thể, hệ thống sử dụng các cảm biến YF-S201 và ACS712 mặc dù vẫn có sai số tuy nhiên độ chính xác vẫn nằm trong giới hạn cho phép, hoạt động ổn định và đáng tin cậy cho mục đích đo lường lưu lượng nước và dòng điện trong các ứng dụng thực tế.

CHƯƠNG 6. KẾT LUẬN

6.1 Kết quả đạt được

Qua quá trình nghiên cứu, thiết kế và thực hiện, đề tài "Hệ thống giám sát và tối ưu hóa sử dụng nước thông minh trong gia đình ứng dụng IoT và AI" đã đạt được nhiều kết quả khả quan. Hệ thống phần cứng đã được hoàn thiện trên một bảng mạch PCB tích hợp đầy đủ các thành phần như vi điều khiển ESP32, cảm biến lưu lượng nước, cảm biến dòng điện, cảm biến hồng ngoại, module relay, van điện từ, bơm nước và màn hình LCD TFT. Nguồn cấp cho toàn hệ thống được ổn định thông qua module hạ áp LM2596S từ nguồn đầu vào 12VDC. Phần mềm điều khiển được xây dựng trên nền tảng Home Assistant kết hợp với Node-RED để xử lý logic, InfluxDB và Grafana để lưu trữ và trực quan hóa dữ liệu. Hệ thống giao tiếp với máy chủ qua giao thức MQTT không dây, cho phép thu thập, xử lý dữ liệu và điều khiển thiết bị đầu ra một cách tự động và hiệu quả. Ngoài ra, giao diện người dùng được thiết kế trực quan, hiển thị các thông tin quan trọng như lượng nước sử dụng, điện năng tiêu thụ, cảnh báo rò rỉ và lịch bảo trì lõi lọc. Hệ thống cũng bước đầu ứng dụng trí tuệ nhân tạo để phân tích dữ liệu tiêu thụ, từ đó đề xuất thời điểm thay lõi lọc và phát hiện các bất thường trong vận hành. Mô hình hoàn chỉnh được thi công gọn gàng trên khung nhôm định hình, đảm bảo tính thẩm mỹ và có khả năng triển khai thực tế trong môi trường hộ gia đình.

6.2 Định hướng phát triển trong tương lai

Trong tương lai, hệ thống có thể được mở rộng và nâng cấp để ứng dụng trong quy mô lớn hơn như trang trại thông minh. Hệ thống có tiềm năng trở thành một giải pháp quản lý nước toàn diện cho nông nghiệp, nơi việc sử dụng nước cần được điều chỉnh linh hoạt theo điều kiện thời tiết, mùa vụ và loại cây trồng. Việc tích hợp thêm các cảm biến môi trường như độ ẩm đất, nhiệt độ, ánh sáng sẽ giúp hệ thống đưa ra quyết định chính xác hơn trong quá trình tưới tiêu tự động. Đồng thời, trí tuệ nhân tạo có thể được phát triển sâu hơn bằng các mô hình học máy nhằm dự đoán

nhu cầu sử dụng nước và tối ưu hóa lịch trình vận hành thiết bị. Ngoài ra, việc ứng dụng các công nghệ truyền thông tầm xa như LoRa hoặc NB-IoT sẽ mở rộng khả năng triển khai ở những vùng có diện tích rộng hoặc điều kiện mạng WiFi hạn chế. Cuối cùng, hệ thống có thể được tích hợp với ứng dụng điều khiển từ xa qua điện thoại, kèm theo chức năng cảnh báo sớm khi có sự cố, góp phần tăng tính tiện lợi, an toàn và hiệu quả cho người sử dụng.

TÀI LIỆU THAM KHẢO

Tiếng Việt

Nguyễn Văn Tuấn (2022).*Ứng dụng IoT trong hệ thống tưới nước tự động sử dụng ESP32 và MQTT*. Tạp chí Khoa học & Công nghệ. Trường Đại học Bách Khoa Hà Nội.

Lê Quốc Huy (2021).*Nghiên cứu và triển khai hệ thống giám sát năng lượng sử dụng cảm biến dòng điện và nền tảng Home Assistant*(Luận văn tốt nghiệp đại học). Trường Đại học Công nghệ Thông tin TP.HCM.

Trần Minh Khang (2023).*Tối ưu hóa hệ thống tưới nước bằng trí tuệ nhân tạo*. Tạp chí Tự động hóa Ngày nay, số 8.

Bộ môn Điện tử - Viễn thông (2020).*Giáo trình hệ thống nhúng với ESP32*.Trường Đại học Sư phạm Kỹ thuật TP.HCM.

Trần Hữu Đức (2022).*Hệ thống cảnh báo rò rỉ nước dựa trên cảm biến lưu lượng và ESP8266*.Tạp chí Khoa học Trường Đại học Công nghiệp TP.HCM.

Tiếng Anh

D. Kumar, S. Bhatia, and M. Rani (2020). *IoT-Based Smart Water Management Systems: A Review.* International Journal of Engineering Research & Technology (IJERT), vol. 9, no. 6, pp. 654–659.

N. A. Ismail, M. A. Rahman, and M. A. Razzaque (2019). *Smart Water Usage Monitoring and Leakage Detection Using IoT*. IEEE 6th International Conference on Smart Instrumentation. Measurement and Applications (ICSIMA), pp. 1–5.

T. Subramani et al (2021). *Design and Implementation of IoT-based Water Monitoring System for Smart Homes*. Journal of Emerging Technologies and Innovative Research (JETIR), vol. 7, no. 4, pp. 1232–1239.

A. Gupta and R. Singh (2020). *Integration of Artificial Intelligence with IoT for Smart Water Managemet*. Procedia Computer Science, vol. 167, pp. 1234–1240.

PHỤ LỤC A

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <TFT_eSPI.h>

// WiFi
const char* ssid = "wifi của trường duy";
const char* password = "letruongduy";

// MQTT HiveMQ
constchar*mqttServer="46b057df071c4ab09dbfdebed9927199.s1.eu.hivemq.cloud;
const int mqttPort = 8883;
const char* mqttUser = "letruongduy";
const char* mqttPassword = "Letruongduy972003";
const char* mqtt_topic_relays = "HomeControl";
const char* mqtt_topic_sensors = "DATN_letruongduy";

// Relay GPIO
const int relay1Pin = 14;
const int relay2Pin = 15;

WiFiClientSecure espClient;
PubSubClient client(espClient);
TFT_eSPI tft = TFT_eSPI();

// Sensor pins
```

```
const int irSensorPin = 12;
const int relayPin = 13;
const byte sensorPin1 = 5;
const byte sensorPin2 = 25;
const byte sensorPin3 = 26;

// Calibration
float calibrationFactor1 = 11.0;
float calibrationFactor2 = 5.0;
float calibrationFactor3 = 5.0;

// Flow pulse counts
volatile byte pulseCount1 = 0;
volatile byte pulseCount2 = 0;
volatile byte pulseCount3 = 0;

// Flow values
float flowRate1, flowRate2, flowRate3;
unsigned int flowMilliLitres1, flowMilliLitres2, flowMilliLitres3;
unsigned long totalMilliLitres1, totalMilliLitres2, totalMilliLitres3;
unsigned long totalAllSensors;
unsigned long oldTime;

// Debounce
volatile unsigned long lastPulseTime1 = 0;
volatile unsigned long lastPulseTime2 = 0;
volatile unsigned long lastPulseTime3 = 0;
const unsigned long debounceDelay = 50;
const unsigned long minPulseInterval = 10;
```

```
// Dòng điện
const int currentPin = 35;
const float sensitivity = 152.0;
const float ADC_resolution = 4095.0;
const float ref_voltage = 3.3;
float zeroCurrentVoltage = 0.0;
float currentMeasurement = 0.0;

// Trạng thái máy bơm từ MQTT
String bomStatus = "UNKNOWN";

// Thông báo từ AI
String thongBaoAI = "UNKNOWN";

// ==== MQTT CALLBACK ====
void callback(char* topic, byte* payload, unsigned int length) {
  String msg;
  for (int i = 0; i < length; i++) msg += (char)payload[i];
  Serial.printf("Message arrived [%s]: %s\n", topic, msg.c_str());

  if (strcmp(topic, mqtt_topic_relays) == 0 || strcmp(topic, "AIControl") == 0) {
    if (msg == "ON1") digitalWrite(relay1Pin, LOW);
    else if (msg == "OFF1") digitalWrite(relay1Pin, HIGH);
    else if (msg == "ON2") digitalWrite(relay2Pin, LOW);
    else if (msg == "OFF2") digitalWrite(relay2Pin, HIGH);
  }

  if (strcmp(topic, "DCWarning") == 0) {
```

```
    if (msg == "ON DINH" || msg == "WARNING") bomStatus = msg;
    else bomStatus = "UNKOWN";
  }

  if (strcmp(topic, "AIWARNING") == 0) {
    if (msg.indexOf("Lượng nước ổn định") != -1) {
      thongBaoAI = "LUONG NUOC ON DINH";
    } else if (msg.indexOf("Bất thường") != -1) {
      thongBaoAI = "BAT THUONG";
    } else if (msg.indexOf("Rò rỉ") != -1 || msg.indexOf("Ro ri") != -1) {
      thongBaoAI = "RO RI";
    } else {
      thongBaoAI = "THONG TIN KHAC";
    }
  }
}

// ==== MQTT RECONNECT ====
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("ESP32ClientCombined", mqttUser, mqttPassword)) {
      Serial.println("connected");
      client.subscribe(mqtt_topic_relays);
      client.subscribe("AIControl");
      client.subscribe("DCWarning");
      client.subscribe("AIWARNING");
    } else {
      Serial.print("failed, rc=");
```

```
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

// ==== PULSE INTERRUPT HANDLERS ====
void IRAM_ATTR pulseCounter1() {
  unsigned long t = millis();
  if (t - lastPulseTime1 >= debounceDelay) {
    pulseCount1++;
    lastPulseTime1 = t;
  }
}
void IRAM_ATTR pulseCounter2() {
  unsigned long t = millis();
  if (t - lastPulseTime2 >= debounceDelay) {
    pulseCount2++;
    lastPulseTime2 = t;
  }
}
void IRAM_ATTR pulseCounter3() {
  unsigned long t = millis();
  if (t - lastPulseTime3 >= debounceDelay) {
    pulseCount3++;
    lastPulseTime3 = t;
  }
}
```

```
// ==== ĐỌC DÒNG ====
void readCurrentSensor() {
  long total = 0;
  for (int i = 0; i < 100; i++) {
    total += analogRead(currentPin);
    delay(2);
  }
  float avg = total / 100.0;
  float voltage = (avg / ADC_resolution) * ref_voltage * 1000.0;
  float diff = voltage - zeroCurrentVoltage;
  currentMeasurement = diff / sensitivity / 2;
  if (abs(currentMeasurement) < 0.05) currentMeasurement = 0;
}

// ==== HIỂN THỊ MÀN HÌNH ====
void updateDisplay() {
  tft.fillScreen(TFT_BLACK);
  tft.setTextSize(2);
  tft.setTextColor(TFT_WHITE);
  int textWidth = tft.textWidth("THONG TIN HIEN THI");
  tft.setCursor((tft.width() - textWidth) / 2, 10);
  tft.print("THONG TIN HIEN THI");

  tft.setTextColor(TFT_SKYBLUE);
  tft.setCursor(10, 40);
  tft.print("NUOC TIEU THU: ");
  tft.print(totalAllSensors / 1000.0);
  tft.print(" (L)");
```

```
  tft.setTextColor(TFT_YELLOW);
  tft.setCursor(10, 70);
  tft.print("MAY BOM NUOC: ");
  tft.print(bomStatus);

  tft.setTextColor(TFT_GREEN);
  tft.setCursor(10, 100);
  tft.print("THONG BAO TU AI:");
  tft.setTextColor(TFT_RED);
  tft.setCursor(10, 130);
  tft.print(thongBaoAI.c_str());

  tft.setTextColor(TFT_SKYBLUE);
  tft.setCursor(10, 160);
  tft.print("TINH TRANG VAN DIEN TU:");

  int buttonWidth = 60, buttonHeight = 30, circleRadius = 12;
  int textOffset = tft.textWidth("VAN1: ");
  int vanY = 190;

  int van1TextX = 10;
  tft.setTextColor(TFT_MAGENTA);
  tft.setCursor(van1TextX, vanY);
  tft.print("VAN1: ");
  int van1ButtonX = van1TextX + textOffset;
  int van1ButtonY = vanY + (tft.fontHeight() - buttonHeight) / 2;
  if (digitalRead(relay1Pin) == LOW) {
    tft.fillRoundRect(van1ButtonX, van1ButtonY, buttonWidth, buttonHeight, 15,
TFT_GREEN);
```

```
  tft.fillCircle(van1ButtonX + buttonWidth - circleRadius - 3, van1ButtonY + buttonHeight / 2, circleRadius, TFT_BLACK);
 } else {
  tft.fillRoundRect(van1ButtonX, van1ButtonY, buttonWidth, buttonHeight, 15, TFT_RED);
  tft.fillCircle(van1ButtonX + circleRadius + 3, van1ButtonY + buttonHeight / 2, circleRadius, TFT_BLACK);
 }

 int van2TextX = van1ButtonX + buttonWidth + 20;
 tft.setCursor(van2TextX, vanY);
 tft.print("VAN2: ");
 int van2ButtonX = van2TextX + textOffset;
 int van2ButtonY = vanY + (tft.fontHeight() - buttonHeight) / 2;
 if (digitalRead(relay2Pin) == LOW) {
  tft.fillRoundRect(van2ButtonX, van2ButtonY, buttonWidth, buttonHeight, 15, TFT_GREEN);
  tft.fillCircle(van2ButtonX + buttonWidth - circleRadius - 3, van2ButtonY + buttonHeight / 2, circleRadius, TFT_BLACK);
 } else {
  tft.fillRoundRect(van2ButtonX, van2ButtonY, buttonWidth, buttonHeight, 15, TFT_RED);
  tft.fillCircle(van2ButtonX + circleRadius + 3, van2ButtonY + buttonHeight / 2, circleRadius, TFT_BLACK);
 }
}

// ==== SETUP ====
void setup() {
```

```
Serial.begin(9600);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  Serial.print(".");
  delay(1000);
}
Serial.println("WiFi connected");

client.setServer(mqttServer, mqttPort);
client.setCallback(callback);
espClient.setInsecure();

tft.init();
tft.setRotation(1);
tft.fillScreen(TFT_BLACK);

pinMode(sensorPin1, INPUT_PULLUP);
pinMode(sensorPin2, INPUT_PULLUP);
pinMode(sensorPin3, INPUT_PULLUP);

pinMode(irSensorPin, INPUT);
pinMode(relayPin, OUTPUT);
digitalWrite(relayPin, LOW);

pinMode(relay1Pin, OUTPUT);
pinMode(relay2Pin, OUTPUT);
digitalWrite(relay1Pin, LOW);
digitalWrite(relay2Pin, LOW);
```

```
  attachInterrupt(digitalPinToInterrupt(sensorPin1), pulseCounter1, FALLING);
  attachInterrupt(digitalPinToInterrupt(sensorPin2), pulseCounter2, FALLING);
  attachInterrupt(digitalPinToInterrupt(sensorPin3), pulseCounter3, FALLING);

  long total = 0;
  for (int i = 0; i < 1000; i++) {
    total += analogRead(currentPin);
    delay(1);
  }
  zeroCurrentVoltage = (total / 1000.0) / ADC_resolution * ref_voltage * 1000.0;
  pulseCount1 = pulseCount2 = pulseCount3 = 0;
  oldTime = millis();
}

// ==== LOOP ====
void loop() {
  int irState = digitalRead(irSensorPin);
  digitalWrite(relayPin, irState == HIGH ? LOW : HIGH);

  if (!client.connected()) reconnect();
  client.loop();

  unsigned long now = millis();
  if (now - oldTime > 1000) {
    detachInterrupt(digitalPinToInterrupt(sensorPin1));
    detachInterrupt(digitalPinToInterrupt(sensorPin2));
    detachInterrupt(digitalPinToInterrupt(sensorPin3));

    flowRate1 = ((1000.0 / (now - oldTime)) * pulseCount1) / calibrationFactor1;
```

```
  flowMilliLitres1 = (flowRate1 / 60.0) * 1000.0;
  totalMilliLitres1 += flowMilliLitres1;

  flowRate2 = ((1000.0 / (now - oldTime)) * pulseCount2) / calibrationFactor2;
  flowMilliLitres2 = (flowRate2 / 60.0) * 1000.0;
  totalMilliLitres2 += flowMilliLitres2;

  flowRate3 = ((1000.0 / (now - oldTime)) * pulseCount3) / calibrationFactor3;
  flowMilliLitres3 = (flowRate3 / 60.0) * 1000.0;
  totalMilliLitres3 += flowMilliLitres3;

  totalAllSensors = totalMilliLitres1 + totalMilliLitres2 + totalMilliLitres3;
  readCurrentSensor();

  if (pulseCount1 > 0 || pulseCount2 > 0 || pulseCount3 > 0 ||
abs(currentMeasurement) > 0.05) {
    String msg = "Sensor 1: " + String(totalMilliLitres1 / 1000.0) + " L, " +
                 "Sensor 2: " + String(totalMilliLitres2 / 1000.0) + " L, " +
                 "Sensor 3: " + String(totalMilliLitres3 / 1000.0) + " L, " +
                 "Current: " + String(currentMeasurement, 3) + " A, " +
                 "Total: " + String(totalAllSensors / 1000.0) + " L";
    client.publish(mqtt_topic_sensors, msg.c_str());
    Serial.println(msg);
  }
  updateDisplay();
  pulseCount1 = pulseCount2 = pulseCount3 = 0;
  oldTime = now;
  attachInterrupt(digitalPinToInterrupt(sensorPin1), pulseCounter1, FALLING);
  attachInterrupt(digitalPinToInterrupt(sensorPin2), pulseCounter2, FALLING);
```

```
    attachInterrupt(digitalPinToInterrupt(sensorPin3), pulseCounter3, FALLING);
  }
}
```