|  |  |
| --- | --- |
|  | BỘ GIÁO DỤC VÀ ĐÀO TẠO  **TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HỒ CHÍ MINH** |
| **BÁO CÁO MÔN HỌC**  **MẠNG MÁY TÍNH TRONG CÔNG NGHIỆP**  **Hệ thống điều khiển và giám sát nhà thông minh sử dụng giao tiếp Modbus RTU**  Ngành: Robot và trí tuệ nhân tạo  Lớp: 22DRTA1  **GIẢNG VIÊN :**  **TS. Phạm Quốc Thiện**   |  |  |  | | --- | --- | --- | | **Sinh viên thực hiện:** | **MSSV:** | **Lớp:** | | Nguyễn Văn Đạt | 2286300010 | 22DRTA1 | | Huỳnh Long | 2286300028 | 22DRTA1 | | Nguyễn Chấn Huy | 2286300020 | 22DRTA1 |   *TP. Hồ Chí Minh, ngày 13 tháng 8 năm 2025* | |

|  |  |
| --- | --- |
|  | BỘ GIÁO DỤC VÀ ĐÀO TẠO  **TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HỒ CHÍ MINH** |
| **BÁO CÁO MÔN HỌC**  **MẠNG MÁY TÍNH TRONG CÔNG NGHIỆP**  **Hệ thống điều khiển và giám sát nhà thông minh sử dụng giao tiếp Modbus RTU**  Ngành: Robot và trí tuệ nhân tạo  Lớp: 22DRTA1  **GIẢNG VIÊN :**  **TS. Phạm Quốc Thiện**   |  |  |  | | --- | --- | --- | | **Sinh viên thực hiện:** | **MSSV:** | **Lớp:** | | Nguyễn Văn Đạt | 2286300010 | 22DRTA1 | | Huỳnh Long | 2286300028 | 22DRTA1 | | Nguyễn Chấn Huy | 2286300020 | 22DRTA1 |   *TP. Hồ Chí Minh, ngày 13 tháng 8 năm 2025* | |

**MỤC LỤC**

[**CHƯƠNG 1: TỔNG QUAN** 1](#_Toc205949671)

[1.1. Giới thiệu đề tài 1](#_Toc205949672)

[1.2. Lý do chọn đề tài 3](#_Toc205949673)

[1.3. Mục tiêu nghiên cứu 4](#_Toc205949674)

[1.4. Phạm vi và giới hạn đề tài 5](#_Toc205949675)

[**CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ GIAO TIẾP MODBUS RTU** 6](#_Toc205949676)

[**CHƯƠNG 3: THIẾT KẾ HỆ THỐNG** 7](#_Toc205949677)

[3.1. Kiến trúc tổng thể hệ thống 7](#_Toc205949678)

[3.2. Vai trò của ESP32 Client và Server 7](#_Toc205949679)

[3.3. Kết nối cảm biến và cơ cấu chấp hành 9](#_Toc205949680)

[3.4. Điều khiển thiết bị chấp hành 10](#_Toc205949681)

[3.5. Tổng quan giao tiếp Modbus RTU 11](#_Toc205949682)

[3.6. Trao đổi dữ liệu với các cơ cấu chấp hành của hai ESP32 Slave bằng Modbus Poll 13](#_Toc205949683)

[**CHƯƠNG 4: GIAO DIỆN GIÁM SÁT VÀ ĐIỀU KHIỂN** 15](#_Toc205949684)

[4.1. Yêu cầu chức năng giao diện 15](#_Toc205949685)

[4.2. Thiết kế giao diện bằng Qt Designer ( chạy trên VSCode ) 16](#_Toc205949686)

[4.3. Hiển thị dữ liệu cảm biến và gửi lệnh điều khiển từ giao diện 17](#_Toc205949687)

[4.4. Giám sát và hiển thị trên nền tảng Blynk 18](#_Toc205949688)

[**CHƯƠNG 5: KẾT QUẢ VÀ ĐÁNH GIÁ** 19](#_Toc205949689)

[5.1. Kết quả thu được 19](#_Toc205949690)

[5.2. Độ trễ, độ ổn định và hiệu quả truyền nhận dữ liệu 21](#_Toc205949691)

[5.3. Nhận xét và đánh giá hệ thống 21](#_Toc205949692)

[**CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN** 23](#_Toc205949693)

[6.1. Kết luận 23](#_Toc205949694)

[6.2. Hướng phát triển: 23](#_Toc205949695)

[**TÀI LIỆU THAM KHẢO** 25](#_Toc205949696)

# **CHƯƠNG 1: TỔNG QUAN**

## Giới thiệu đề tài

Đề tài*“Hệ thống điều khiển và giám sát nhà thông minh sử dụng giao tiếp Modbus RTU”* hướng đến việc xây dựng một mô hình điều khiển đơn giản, hiệu quả và tiết kiệm chi phí. Trong mô hình này, máy tính đóng vai trò master, và hai vi điều khiển ESP32 đảm nhiệm vai trò slave, giao tiếp với nhau thông qua giao thức Modbus RTU qua đường truyền UART (RS-485). Một ESP32 có nhiệm vụ thu thập dữ liệu từ các cảm biến nhiệt độ, độ ẩm và chuyển động, trong khi ESP32 còn lại điều khiển cơ cấu đóng/mở cửa tự động hay bật quạt theo lệnh điều khiển từ master, ngoài ra sử dụng esp32 camera để giám sát ngôi nhà. Tuy được điều khiển và giám sát thông qua một máy chủ trung tâm nhưng người dùng vẫn có thể điều khiển trực tiếp thông qua Blynk IoT.

Nghiên cứu [1] trình bày việc phát triển một nguyên mẫu hệ thống nhà thông minh sử dụng vi điều khiển ESP32 kết hợp với giao tiếp IoT. Mục tiêu là giám sát và điều khiển từ xa các thiết bị điện, đồng thời theo dõi nhiệt độ môi trường và ước tính chi phí hóa đơn điện hàng tháng. Hệ thống gồm các phần cứng chính: ESP32 WiFi module, cảm biến PZEM 004T (đo dòng, điện áp, công suất), cảm biến DHT11 (nhiệt độ và độ ẩm), LCD 16x2, bộ relay 8 kênh, buzzer cảnh báo, và cấp nguồn 220VAC/5V. Về phần mềm, hệ thống được lập trình bằng Arduino IDE và tích hợp với ứng dụng Blynk trên smartphone để thực hiện giám sát và điều khiển qua internet. Phương pháp thực hiện gồm ba giai đoạn: *(1)* phân tích và xác định nhu cầu thiết bị, *(2)* thiết kế phần cứng và phần mềm, *(3)* thử nghiệm hoạt động hệ thống. Kết quả thử nghiệm cho thấy nguyên mẫu hoạt động chính xác, cho phép điều khiển trạng thái thiết bị (ON/OFF), hiển thị thông tin dòng, công suất, nhiệt độ môi trường và ước tính điện năng tiêu thụ thực tế. Hệ thống đạt tính ổn định cao và dễ triển khai trong thực tế, góp phần nâng cao khả năng quản lý năng lượng tại hộ gia đình. Nghiên cứu [2] trình bày hệ thống nhà thông minh chi phí thấp sử dụng giao thức Modbus/RTU, nhằm giải quyết giới hạn trong mô hình điều khiển thiết bị gia dụng truyền thống. Hệ thống áp dụng cấu trúc master-slave, với AT89C51 làm thiết bị master và C8051F cùng PC đóng vai trò thiết bị slave, sử dụng giao tiếp nối tiếp tiêu chuẩn RS-232 qua mạch MAX232. Dữ liệu thiết bị được thu thập, xử lý và lưu trữ vĩnh viễn bằng chip FLASH K9F1GU0b, hỗ trợ bởi các module AD/DA 12-bit để chuyển đổi tín hiệu từ các cảm biến. Giao thức Modbus/RTU đảm bảo truyền thông tin chính xác với tốc độ 9600 bit/s, khung dữ liệu gồm địa chỉ, dữ liệu, kiểm tra lỗi và CRC. Hệ thống sử dụng lập trình ngắt và kỹ thuật đóng gói khung dữ liệu để đảm bảo thiết bị tớ có thể phản hồi chính xác theo địa chỉ được định danh. Giải pháp cũng khắc phục xung đột đường truyền do hạn chế cổng nối tiếp trên vi điều khiển thông qua thiết kế giao diện phần cứng riêng biệt. Kết quả thử nghiệm thực tế với nhiều thiết bị (điều hòa, cảm biến nhiệt độ...) cho thấy hệ thống hoạt động ổn định, không gián đoạn khi có thiết bị lỗi, dễ mở rộng lên tới 247 thiết bị, và có thể điều khiển theo thời gian thực. Thiết kế tối giản, dễ nhân bản, hứa hẹn triển khai thực tế hiệu quả với chi phí thấp. Nghiên cứu [3] đề xuất một phương pháp thiết kế hệ giám sát–điều khiển hai lớp kết hợp Modbus/RTU và Modbus/TCP để tối ưu hóa quản lý hệ thống thiết bị siêu đa nút tiết kiệm điện. Về phương pháp, lớp 1 dùng Modbus/RTU trên RS485 cho các nhóm nút nhỏ để đọc/ghi trạng thái và giá trị analog bằng các mã chức năng 01/02/03/05, khởi tạo cổng nối tiếp và đóng gói khung kèm CRC; lớp 2 dùng Modbus/TCP qua Ethernet với lập trình socket, khung MBAP và mô hình client–server để truyền lệnh tốc độ cao, đồng thời có bộ chuyển đổi (router) TCP↔RTU loại/ghép MBAP và CRC khi chuyển gói giữa hai lớp. Về kết quả, kiến trúc hai lớp cho phép điều khiển thuận tiện các nhóm ít nút và đồng thời giám sát–điều khiển toàn mạng siêu đa nút với thông lượng cao và độ tin cậy tốt, qua đó nâng cao hiệu quả vận hành và mở rộng tính ứng dụng trong nhà máy, tòa nhà và nhà thông minh.

Phương pháp được triển khai trong báo cáo này là sử dụng một máy tính điều khiển trung tâm làm master Modbus RTU, trong khi hai vi điều khiển ESP32 được cấu hình làm thiết bị slave giao tiếp qua chuẩn RS485. Mỗi slave đảm nhận chức năng riêng biệt: slave thứ nhất thu thập dữ liệu từ các cảm biến nhiệt độ, độ ẩm và trạng thái cửa; slave thứ hai điều khiển các cơ cấu chấp hành như mô-đun cửa tự động và quạt làm mát. Việc truyền thông giữa master và các slave được thực hiện bằng giao tiếp nối tiếp qua RS485, sử dụng khung dữ liệu Modbus RTU gồm địa chỉ thiết bị, mã chức năng, dữ liệu và mã kiểm tra CRC để đảm bảo độ tin cậy. ESP32 được lập trình để xử lý yêu cầu đọc/ghi các thanh ghi dữ liệu theo mã chức năng Modbus RTU như 03 (đọc thanh ghi giữ) và 05 (ghi giá trị cuộn dây).

Giao diện điều khiển được phát triển trên máy tính cho phép người dùng gửi lệnh và nhận dữ liệu trực tiếp qua kết nối nối tiếp. Dữ liệu cảm biến từ slave 1 được truyền về máy tính và đồng bộ lên nền tảng Blynk Cloud để phục vụ giám sát thời gian thực. Các tín hiệu điều khiển từ giao diện (máy tính hoặc ứng dụng Blynk) được xử lý và truyền đến slave 2 thông qua khung lệnh Modbus RTU để kích hoạt thiết bị đầu ra như quạt hoặc cửa tự động, dựa trên trạng thái môi trường hoặc thao tác của người dùng.

Kết quả mong đợi là hệ thống truyền thông Modbus RTU hoạt động ổn định, độ trễ thấp, giao tiếp dữ liệu chính xác giữa máy tính và các vi điều khiển ESP32. Việc điều khiển thiết bị theo điều kiện môi trường thực tế chứng minh tính hiệu quả, khả năng mở rộng và tiềm năng ứng dụng thực tế của mô hình này trong các hệ thống nhà thông minh hoặc công nghiệp quy mô nhỏ.

## Lý do chọn đề tài

Theo dữ liệu thống kê, số lượng người dùng nhà thông minh toàn cầu dự kiến sẽ tăng trưởng liên tục từ năm 2023 đến năm 2028, với mức tăng ấn tượng lên tới 424,5 triệu người dùng mới, tương ứng với mức tăng trưởng 117,69% trong vòng 5 năm. Đến năm 2028, tổng số người dùng nhà thông minh toàn cầu được ước tính sẽ đạt 785,16 triệu, đánh dấu năm thứ 9 liên tiếp tăng trưởng và thiết lập một mốc kỷ lục mới về mức độ phổ biến của công nghệ nhà thông minh trên thị trường tiêu dùng toàn cầu [1]. Nhà thông minh là một giải pháp công nghệ hiện đại mang lại khả năng tự động hóa và điều khiển từ xa các thiết bị điện trong gia đình, góp phần nâng cao mức độ tiện nghi, tiết kiệm thời gian và tối ưu hóa trải nghiệm sinh hoạt. Thông qua các thiết bị điều khiển như điện thoại thông minh, người dùng có thể quản lý các thiết bị như đèn chiếu sáng, quạt, máy lạnh, cửa ra vào, rèm cửa, máy nước nóng, và hệ thống an ninh một cách linh hoạt và trực quan. Hệ thống có thể tự động bật/tắt đèn khi phát hiện chuyển động, hẹn giờ mở cửa sổ hoặc vận hành máy lọc không khí vào buổi sáng, đồng thời cho phép theo dõi và điều khiển camera, cảm biến cửa, và chuông báo động từ bất kỳ đâu thông qua kết nối Internet. Ngoài ra, các thiết bị như điều hòa, quạt và máy nước nóng có thể được điều khiển từ xa để chuẩn bị sẵn môi trường sống trước khi người dùng trở về nhà. Cửa cuốn hoặc cổng nhà cũng có thể được tự động vận hành bằng smartphone thay vì điều khiển vật lý truyền thống. Hệ thống còn hỗ trợ theo dõi mức tiêu thụ điện năng theo thời gian thực, giúp người dùng điều chỉnh thói quen sử dụng thiết bị để tiết kiệm năng lượng hiệu quả hơn, đồng thời có thể ngăn ngừa các trường hợp cháy, nổ trong môi trường dân dụng, giúp giữ an toàn cho người dùng hay xã hội. Những đặc điểm này không chỉ mang lại sự thuận tiện và thoải mái trong đời sống thường ngày, mà còn góp phần nâng cao an toàn, tối ưu chi phí vận hành và đáp ứng xu hướng sống hiện đại, thông minh và bền vững.

ESP32 là một vi điều khiển hiệu năng cao, hỗ trợ nhiều phương thức truyền thông, cho phép truyền dữ liệu cảm biến hoặc tín hiệu điều khiển giữa thiết bị đầu cuối và máy tính giám sát một cách linh hoạt và ổn định. Trong môi trường công nghiệp, ESP32 được cấu hình để giao tiếp bằng Modbus RTU thông qua đường truyền RS485, đảm bảo khả năng truyền dữ liệu tin cậy và chống nhiễu tốt. Việc giao tiếp được thực hiện bằng khung truyền Modbus RTU gồm địa chỉ thiết bị, mã chức năng, dữ liệu và mã kiểm tra CRC, hỗ trợ các thao tác đọc–ghi trạng thái thiết bị và dữ liệu cảm biến.

Bên cạnh RS485, ESP32 vẫn có khả năng kết nối UART hoặc USB để truyền dữ liệu qua cổng COM, giúp thiết lập giao tiếp với máy tính đơn giản và dễ cấu hình. Ngoài ra, ứng dụng Blynk có thể được tích hợp để hiển thị dữ liệu cảm biến và gửi lệnh điều khiển, tuy nhiên trong mô hình này, các lệnh điều khiển và dữ liệu phản hồi đều được truyền bằng giao thức Modbus RTU, đảm bảo tính chuẩn hóa và độ tin cậy cao trong môi trường thực tế. Giải pháp này phù hợp với các hệ thống nhà thông minh hoặc nhà máy quy mô nhỏ cần giám sát nhiều thiết bị nhưng vẫn tối ưu về chi phí, hiệu năng và khả năng mở rộng.

## Mục tiêu nghiên cứu

Mục tiêu của đề tài là xây dựng và triển khai một hệ thống giám sát nhà thông minh sử dụng vi điều khiển ESP32 và giao thức truyền thông Modbus RTU qua RS485, trong đó các thiết bị ESP32 hoạt động ở vai trò Slave, còn máy tính hoặc laptop đóng vai trò Master. Hệ thống cho phép truyền và nhận dữ liệu cảm biến cũng như điều khiển thiết bị đầu ra như đèn, quạt, hoặc relay thông qua giao tiếp tuần tự sử dụng Modbus RTU, đảm bảo độ tin cậy cao và khả năng truyền trên khoảng cách xa.

Việc triển khai giao thức Modbus RTU giúp tiêu chuẩn hóa quá trình truyền thông giữa thiết bị giám sát và thiết bị điều khiển, đồng thời phù hợp với các ứng dụng công nghiệp quy mô nhỏ hoặc môi trường nhà thông minh có yêu cầu ổn định và chống nhiễu. Hệ thống sử dụng các cảm biến như nhiệt độ và độ ẩm để thu thập dữ liệu thời gian thực, sau đó truyền về thiết bị Master để xử lý và đưa ra quyết định điều khiển thích hợp thông qua khung truyền Modbus RTU. Mô hình này đảm bảo khả năng mở rộng linh hoạt, dễ tích hợp và hiệu quả về chi phí.

## Phạm vi và giới hạn đề tài

Phạm vi nghiên cứu của đề tài tập trung vào việc xây dựng mô hình giám sát nhà thông minh sử dụng hai vi điều khiển ESP32 đảm nhận vai trò Slave và một máy tính cá nhân (laptop) đóng vai trò Master, giao tiếp với nhau thông qua giao thức Modbus RTU trên đường truyền vật lý RS485. Hệ thống được thiết kế để thu thập dữ liệu từ các cảm biến môi trường như nhiệt độ và độ ẩm, đồng thời điều khiển các thiết bị chấp hành như đèn, quạt và relay theo tín hiệu từ thiết bị Master. Quá trình triển khai bao gồm cấu hình giao tiếp Modbus RTU giữa máy tính và các ESP32 qua kết nối RS485, tích hợp và lập trình cảm biến để truyền dữ liệu thời gian thực thông qua khung truyền RTU chuẩn (gồm địa chỉ thiết bị, mã chức năng và mã CRC), cũng như thiết lập cơ chế điều khiển đầu ra dựa trên dữ liệu nhận được tại Master. Ngoài ra, đề tài còn thử nghiệm khả năng giao tiếp tuần tự giữa một Master và nhiều Slave trong cùng hệ thống RS485, và dữ liệu từ các cảm biến sẽ được gửi vào Blynk cloud để người dùng có thể điều khiển từ xa. Tuy nhiên, chưa tích hợp các cơ chế mã hóa để đảm bảo bảo mật, và việc đánh giá hiệu quả chỉ tập trung vào độ chính xác và ổn định của truyền thông, chưa đi sâu vào phân tích độ trễ hay hiệu năng trong điều kiện mở rộng quy mô thiết bị.

# **CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ GIAO TIẾP MODBUS RTU**

RS485 là một chuẩn giao tiếp nối tiếp được sử dụng rộng rãi trong các hệ thống giám sát và điều khiển nhờ vào khả năng truyền dữ liệu ổn định trên khoảng cách dài và khả năng chống nhiễu điện từ cao. RS485 hoạt động theo nguyên lý truyền tín hiệu vi sai qua hai dây dẫn A và B, giúp tăng độ tin cậy trong giao tiếp. Chuẩn này cho phép thiết lập mạng đa điểm, kết nối lên đến 32 thiết bị trên cùng một bus truyền dẫn, phù hợp với các môi trường công nghiệp hoặc giám sát tập trung.

Trong ứng dụng với giao thức Modbus RTU, RS485 đóng vai trò là lớp truyền dẫn vật lý. Khung truyền Modbus RTU được cấu tạo gồm: 1 byte địa chỉ thiết bị, 1 byte mã chức năng, 2 byte địa chỉ bắt đầu, n byte dữ liệu, và 2 byte mã kiểm tra CRC, đảm bảo tính toàn vẹn và độ chính xác khi truyền nhận. Quá trình giao tiếp gồm khởi tạo cổng serial, cấu hình các tham số như tốc độ truyền, bit dữ liệu, chế độ ngắt và cơ chế kiểm tra lỗi. Sau đó, dữ liệu được gửi và nhận tuần tự qua buffer, với CRC đóng vai trò phát hiện lỗi truyền tải.

Việc sử dụng RS485 trong hệ thống Modbus RTU giúp triển khai mô hình giám sát tập trung với chi phí thấp, kiến trúc đơn giản nhưng hiệu quả truyền cao, đáp ứng yêu cầu điều khiển thông minh trong môi trường có nhiều thiết bị nút.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address Code | Function Code | Start Address | Data | CRC |
| 1 Byte | 1 Byte | 2 Byte | N Byte | 2 Byte |

Bảng 2.1: Cấu trúc khung truyền modbus RTU (RS485)

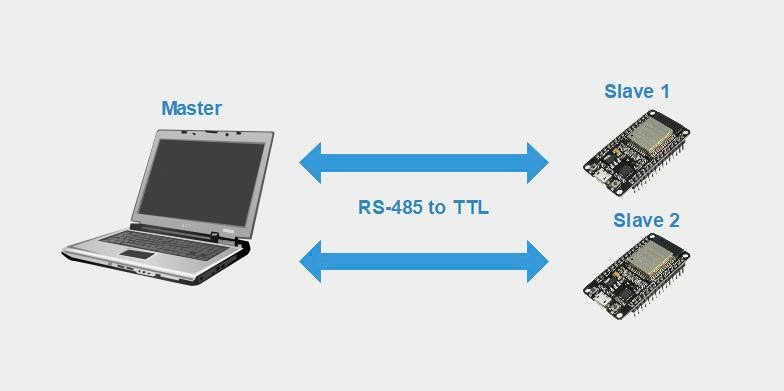
# **CHƯƠNG 3: THIẾT KẾ HỆ THỐNG**

## 3.1. Kiến trúc tổng thể hệ thống

Hệ thống giám sát nhà thông minh được thiết kế dựa trên kiến trúc phân tán, trong đó sử dụng hai vi điều khiển ESP32 đảm nhiệm vai trò Modbus RTU Server (Slave) và một máy tính (laptop) đóng vai trò Modbus RTU Client (Master). Tất cả các thiết bị hoạt động trong cùng một mạng Wi-Fi nội bộ (LAN).

Các thiết bị ESP32 sẽ thực hiện việc thu thập dữ liệu môi trường từ các cảm biến như: nhiệt độ, độ ẩm, khí gas, hồng ngoại… Sau đó, dữ liệu sẽ được gửi về Master thông qua giao thức Modbus RTU. Laptop thực hiện việc phân tích dữ liệu, hiển thị (có GUI) và gửi lệnh điều khiển ngược lại đến các thiết bị chấp hành như LED, loa, servo cửa, và

camera ESP32-CAM.

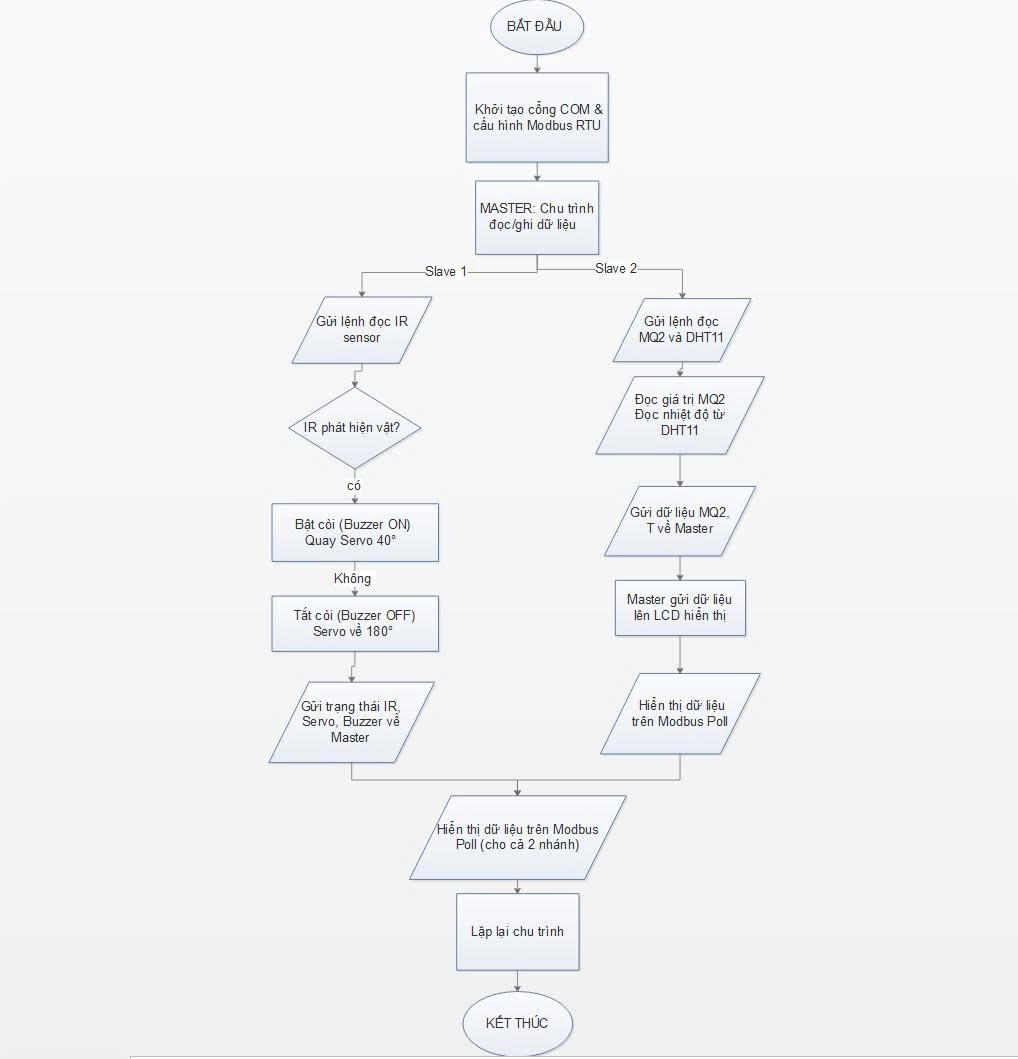


*Hình 3.1. Sơ đồ tổng quát hệ thống*

## 3.2. Vai trò của ESP32 Client và Server

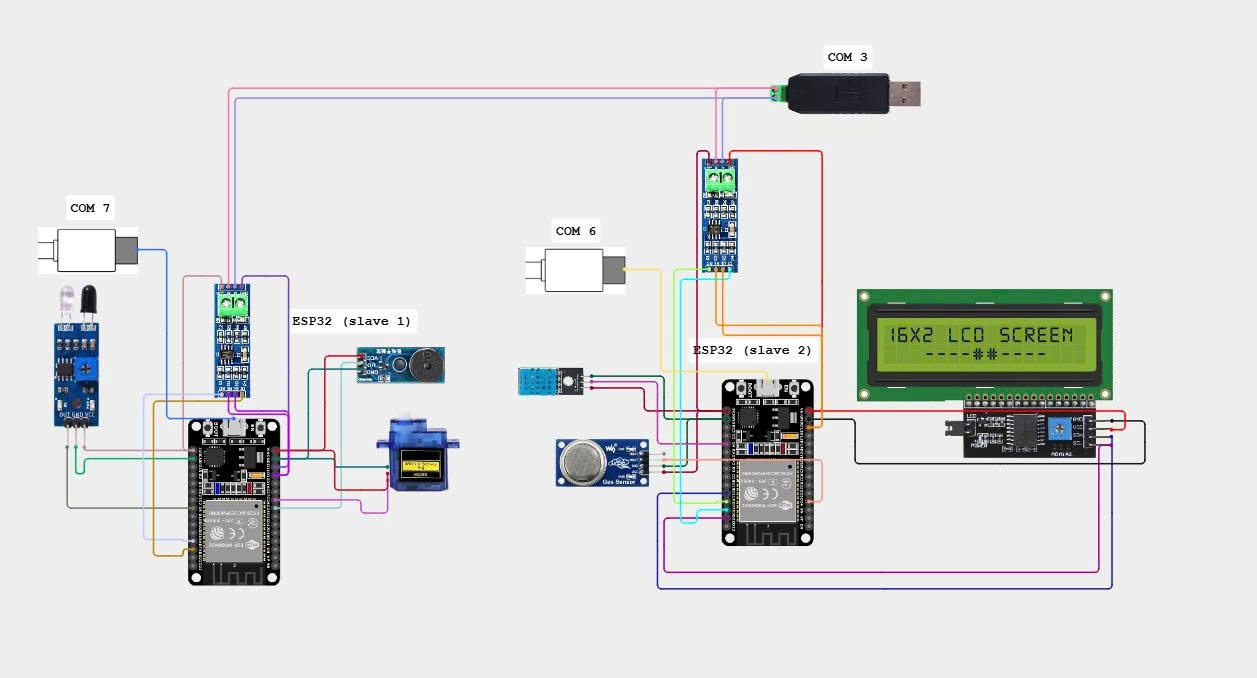
Trong đề tài này, máy tính đóng vai trò là thiết bị Client (Master), sử dụng phần mềm Modbus Poll để điều khiển và giám sát toàn bộ hệ thống thông qua giao thức Modbus RTU. Hệ thống bao gồm hai vi điều khiển ESP32 hoạt động ở chế độ Slave, mỗi thiết bị phụ trách một khu vực riêng biệt với các cảm biến và cơ cấu chấp hành cụ thể. Slave 1 được kết nối với cảm biến hồng ngoại dùng để phát hiện chuyển động, khi phát hiện có người, vi điều khiển sẽ điều khiển servo mini mô phỏng cửa tự động, đồng thời kích hoạt buzzer để phát âm thanh cảnh báo. Trong khi đó, Slave 2 đảm nhận việc thu thập dữ liệu từ cảm biến khí gas MQ2 và cảm biến nhiệt độ/độ ẩm DHT11, các thông số này sẽ được hiển thị trên màn hình LCD 16x2. Ngoài ra, Slave 2 còn kết nối với ESP32-CAM, cho phép ghi lại hình ảnh giám sát trong trường hợp phát hiện rò rỉ khí hoặc chuyển động bất thường.

Các thiết bị ESP32 luôn trong trạng thái lắng nghe kết nối từ Master trên cổng 502, là cổng tiêu chuẩn của giao thức Modbus RTU. Khi nhận được lệnh đọc, mỗi Slave sẽ phản hồi bằng dữ liệu thu thập được từ các cảm biến. Ngược lại, khi nhận lệnh ghi, thiết bị sẽ điều khiển các phần tử chấp hành tương ứng như servo, buzzer, v.v. Việc truyền nhận dữ liệu giữa các thiết bị được tổ chức theo mô hình client–server, đảm bảo hệ thống hoạt động ổn định, phản hồi nhanh và hỗ trợ khả năng mở rộng, phù hợp với các ứng dụng giám sát và điều khiển từ xa trong công nghiệp cũng như trong các mô hình nhà thông minh thực tiễn.



Hình 3.2. Lưu đồ giải thuật của hệ thống

### 3.3. Kết nối cảm biến và cơ cấu chấp hành



*Hình 3.3. Sơ đồ kết nối của các Esp32 (slave 1,2)*

|  |  |  |  |
| --- | --- | --- | --- |
| Slave | Thiết bị | Kiểu kết nối | GPIO |
| Slave 1 | IR sensor | Output | G12 |
|  | Mini servo | WPM | G13 |
|  | Buzzer | Input/Output | G14 |
|  | TTL – RS485 | RO,RE,DI,DE | RX/TX:G16-G17;RE/DE:G27 |
| Slave2 | LCD | SDA/SCL | G21/G22 |
|  | MQ2 | AOUT | G34 |
|  | DHT11 | DATA | G15 |
|  | TTL – RS485 | RO,RE,DI,DE | RX/TX:G16-G17;RE/DE:G4 |

*Bảng 3.1. Bảng địa chỉ sơ đồ kết nối chân esp32*

## 3.4. Điều khiển thiết bị chấp hành

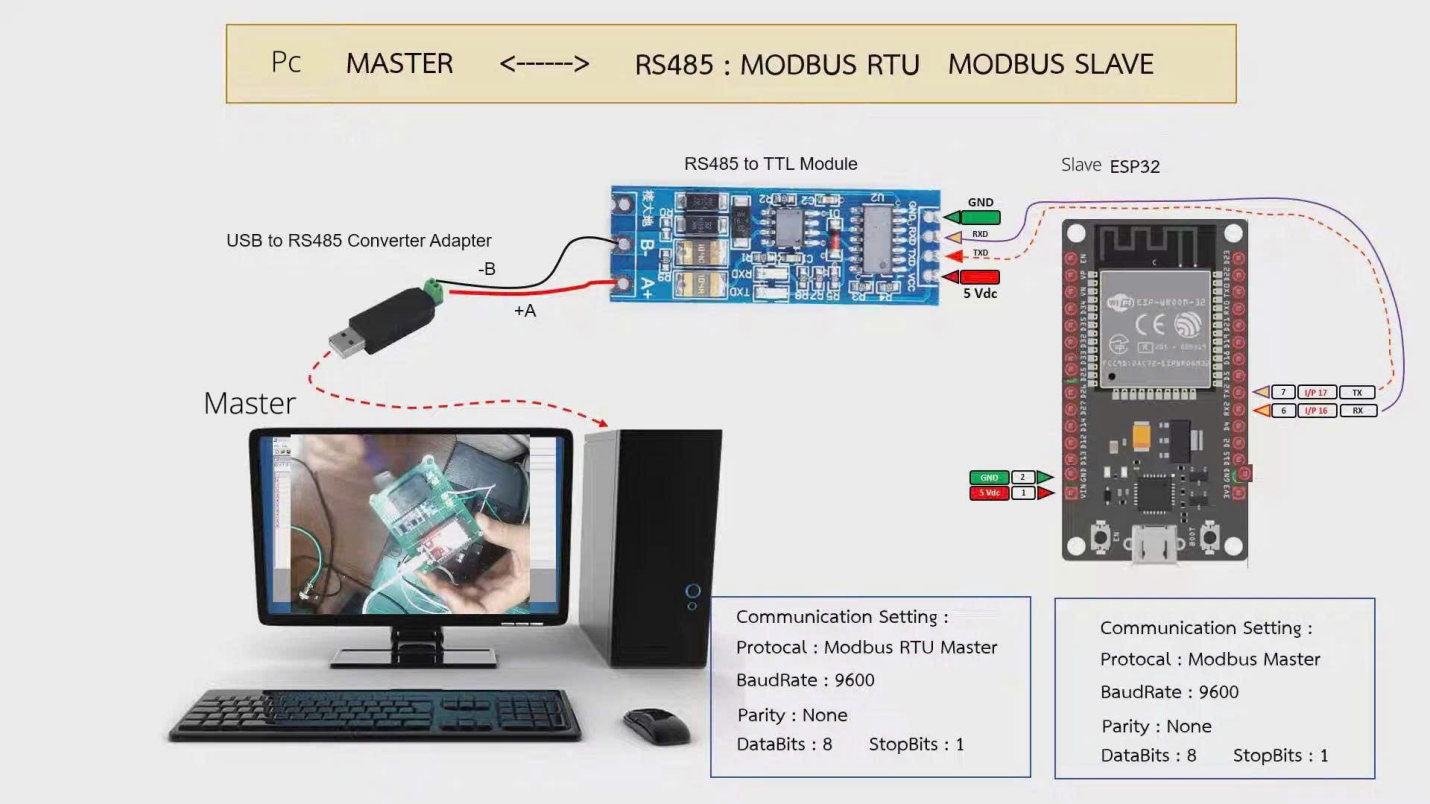
Phòng 1 sử dụng ESP32 (slave 1) được kết nối với cảm biến IR, buzzer, và servo mini. Cảm biến IR được dùng để phát hiện chuyển động hoặc có người xuất hiện, từ đó tự động điều khiển mở cửa bằng servo và kích hoạt buzzer cảnh báo trong các tình huống cần thiết, chẳng hạn như phát hiện người vào ban đêm. Servo mini trong phòng 1 đóng vai trò mô phỏng cơ cấu cửa tự động, mang lại tính hiện đại và thuận tiện cho người sử dụng.

Phòng 2 sử dụng ESP32 (slave 2) và tích hợp nhiều thiết bị hơn, bao gồm: ESP32-CAM, LCD I2C, cảm biến MQ2, DHT11, cảm biến IR, servo mini và loa cảnh báo (buzzer). Cảm biến MQ2 dùng để phát hiện rò rỉ khí gas hoặc khói trong phòng, còn DHT11 đo nhiệt độ và độ ẩm không khí, giúp người dùng dễ dàng giám sát môi trường. Các dữ liệu từ MQ2 và DHT11 sẽ được hiển thị lên màn hình LCD 16x2 nhằm cung cấp thông tin trực quan và kịp thời. Một cảm biến IR thứ hai cũng được gắn ở phòng 2 để phát hiện người và điều khiển đóng/mở cửa bằng servo tương tự như phòng 1. Khi có cảnh báo từ MQ2 hoặc bất thường nhiệt độ từ DHT11, loa cảnh báo sẽ phát âm thanh cảnh báo.

ESP32-CAM được kết nối trực tiếp với slave 2 và có nhiệm vụ giám sát hình ảnh trong phòng 2 từ xa thông qua WiFi, hoặc phục vụ nhận diện khuôn mặt để điều khiển cửa tự động trong tương lai. Nhờ đó, người dùng có thể quan sát tình trạng trong phòng qua điện thoại hoặc máy tính.

Toàn bộ hệ thống được cấp nguồn bằng sạc dự phòng 5V, đảm bảo tính di động và ổn định cho mô hình. Với sự phân chia thiết bị hợp lý giữa hai phòng và khả năng giao tiếp giữa các ESP32 với PC qua Modbus RTU, mô hình nhà thông minh không chỉ đảm bảo sự tiện nghi, an toàn, mà còn thể hiện rõ tính ứng dụng thực tế của công nghệ IoT (Internet of Things) trong việc điều khiển, giám sát và bảo vệ không gian sống hiện đại.

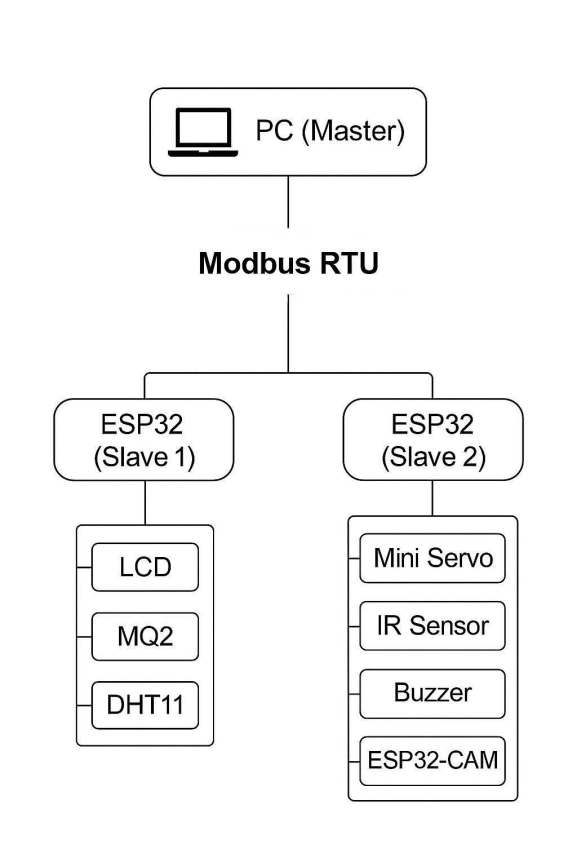
## 3.5. Tổng quan giao tiếp Modbus RTU



*Hình 3.4. Sơ đồ giao tiếp của esp32 với máy tính*

Trong đề tài này đã lựa chọn giao thức Modbus RTU làm nền tảng truyền thông chính giữa thiết bị trung tâm là laptop (Client – Master) và hai vi điều khiển ESP32 (Server – Slave 1 và Slave 2). Giao thức này dựa trên chuẩn truyền thông nối tiếp (serial), thường sử dụng cổng RS-485 hoặc UART, giúp giảm thiểu nhiễu và phù hợp với khoảng cách truyền dài, đồng thời mang lại khả năng điều khiển thiết bị chính xác và ổn định. Trong hệ thống, laptop sẽ giao tiếp với các ESP32 thông qua chuẩn UART hoặc RS-485, sử dụng cổng COM (serial port) được thiết lập trên phần mềm Modbus Master. Mỗi thiết bị ESP32 sẽ có một địa chỉ Slave ID duy nhất, ví dụ như Slave 1 là ID 1, Slave 2 là ID 2, nhằm phân biệt rõ ràng khi truyền nhận dữ liệu. Nhờ cơ chế địa chỉ hóa này, Master có thể gửi yêu cầu đến từng thiết bị cụ thể mà không gây xung đột đường truyền.

Trên mỗi ESP32, trong đề tài này sử dụng thư viện ModbusRTU (ví dụ từ SimpleModbus hoặc ModbusMaster) để cấu hình hoạt động như một Modbus Server (Slave). ESP32 sẽ liên tục lắng nghe tín hiệu truyền đến từ Master thông qua chân UART đã cấu hình, sau đó xử lý các gói tin chứa lệnh đọc hoặc ghi dữ liệu theo chuẩn Modbus RTU. Các lệnh sử dụng trong hệ thống bao gồm Function Code 0x04 hoặc 0x02 để đọc dữ liệu từ các cảm biến như nhiệt độ, độ ẩm, khí gas hoặc trạng thái phát hiện chuyển động, trong đó ESP32 sẽ phản hồi các giá trị đã lưu tại các thanh ghi tương ứng. Ngoài ra, Function Code 0x05 hoặc 0x0F được dùng để ghi giá trị điều khiển đến các thiết bị chấp hành như buzzer, servo hoặc LED, khi đó việc ghi giá trị ‘1’ hoặc ‘0’ sẽ tương ứng với bật hoặc tắt thiết bị tùy theo thiết kế hệ thống.

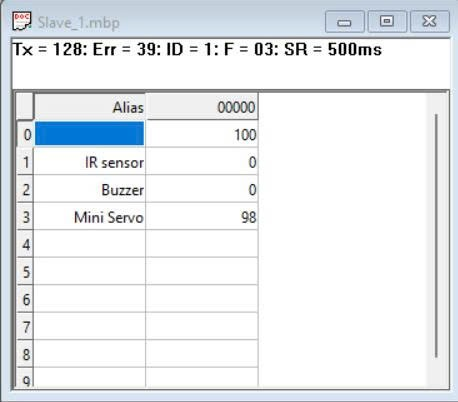
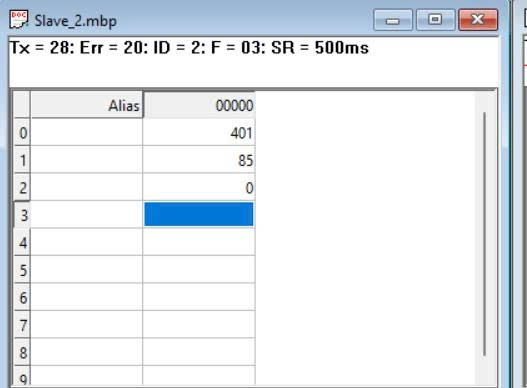


*Hình 3.5. Sơ đồ khối của của PC(master) với esp32(slave)*

Máy tính (Client) sử dụng phần mềm Modbus Poll – một công cụ giả lập Master phổ biến để gửi các yêu cầu đọc hoặc ghi dữ liệu đến các thiết bị Slave. Trong phần mềm, người dùng thiết lập cổng COM như COM3, cấu hình tốc độ truyền dữ liệu (baud rate) chẳng hạn 9600, khai báo địa chỉ Slave ID và lựa chọn loại thanh ghi cần truy cập như Input Register, Coil hoặc Holding Register. Modbus Poll sẽ định kỳ gửi các gói tin theo chuẩn Modbus RTU đến ESP32, sau đó hiển thị dữ liệu phản hồi lên giao diện dưới dạng bảng số liệu giúp người dùng theo dõi trạng thái cảm biến và điều khiển thiết bị một cách trực quan.

Các ESP32 hoạt động độc lập, không có giao tiếp trực tiếp với nhau mà chỉ nhận lệnh từ laptop (Master), đảm bảo đúng mô hình Client – Server (Master – Slave) của giao thức Modbus RTU. Thiết kế này giúp tăng tính ổn định, dễ bảo trì và thuận tiện trong việc mở rộng hệ thống bằng cách bổ sung thêm các thiết bị Slave mới với địa chỉ ID riêng biệt. Tóm lại, việc sử dụng Modbus RTU thông qua giao tiếp có dây như UART hoặc RS-485 kết hợp cùng phần mềm trực quan như Modbus Poll đã tạo nên một nền tảng điều khiển hiệu quả, bền vững và phù hợp cho các hệ thống nhúng, mô hình IoT hoặc các ứng dụng nhà thông minh đòi hỏi tính ổn định và khả năng chống nhiễu tốt trong môi trường thực tế.

## 3.6. Trao đổi dữ liệu với các cơ cấu chấp hành của hai ESP32 Slave bằng Modbus Poll

Hình 3.6. Địa chỉ thanh ghi cho 2 ESP32 slave

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Thiết bị** | **ID Modbus** | **Địa chỉ thanh ghi** | **Chức năng** | **Kiểu dữ liệu** | **Giá trị mô tả** |
| **ESP32 Slave 1** *(Điều khiển)* | 1 | 0 | Mã thiết bị ("Name") | String/int | Mã định danh thiết bị |
|  |  | 1 | Còi (Buzzer) | int | 0: Tắt, 1: Bật |
|  |  | 2 | Servo | int | Góc quay servo (0–180) |
|  |  | 3 | Cảm biến hồng ngoại (IR Sensor) | int | 0: Không có vật, 1: Có vật cản |
| **ESP32 Slave 2** *(Giám sát)* | 2 | 0 | Mã thiết bị ("Name") | String/int | Mã định danh thiết bị |
|  |  | 1 | Nhiệt độ (Temp) | float/int | °C |
|  |  | 2 | Độ ẩm (Humidity) | float/int | % |
|  |  | 3 | Nồng độ khí gas (MQ2/Smoke) | int | Giá trị ADC hoặc % tương đối |

*Bảng 3.2. Bảng địa chỉ thanh ghi cho hai ESP32 Slave*

Trong hệ thống này, ESP32 Slave 1 phụ trách điều khiển các cơ cấu chấp hành như còi, servo và xử lý tín hiệu từ cảm biến hồng ngoại. Trong khi đó, ESP32 Slave 2 đảm nhiệm việc thu thập dữ liệu từ các cảm biến môi trường như cảm biến nhiệt độ, độ ẩm (DHT11) và cảm biến khí gas (MQ2).

Giao tiếp giữa các thiết bị tuân theo cấu hình sau: chuẩn vật lý RS485, sử dụng giao thức Modbus RTU, tốc độ truyền 9600 bps, cấu hình 8 bit dữ liệu, không chẵn lẻ, 1 bit dừng (8N1), hoạt động ở chế độ half-duplex. Cấu trúc truyền dữ liệu được thiết kế để phần mềm Modbus Poll có thể gửi yêu cầu đọc dữ liệu bằng Function code 03 (Read Holding Registers) và gửi lệnh điều khiển bằng Function code 06 (Write Single Register).

Phần mềm Modbus Poll giao tiếp với các thiết bị thông qua cổng COM được chuyển đổi từ USB sang RS485. Khi phần mềm gửi lệnh đến đúng địa chỉ ID và thanh ghi tương ứng, thiết bị ESP32 sẽ phản hồi dữ liệu một cách chính xác.

Ví dụ, khi ghi giá trị 1 vào thanh ghi địa chỉ 1 của ESP32 Slave 1, còi sẽ được bật. Ngược lại, khi yêu cầu đọc dữ liệu từ thanh ghi địa chỉ 1 của ESP32 Slave 2, phần mềm sẽ nhận về giá trị nhiệt độ hiện tại đo được từ cảm biến DHT11 kết nối với thiết bị đó.

Mỗi khi kết nối giữa phần mềm và thiết bị thành công, Modbus Poll sẽ hiển thị thông báo dạng “Rx = …” biểu thị dữ liệu đã nhận. Trong trường hợp kết nối thất bại hoặc địa chỉ thanh ghi không chính xác, dòng chữ "No connection" sẽ xuất hiện để cảnh báo người dùng.

# **CHƯƠNG 4: GIAO DIỆN GIÁM SÁT VÀ ĐIỀU KHIỂN**

## 4.1. Yêu cầu chức năng giao diện

Hệ thống được thiết kế để giám sát và điều khiển các thiết bị thông qua giao thức Modbus RTU, trong đó máy tính đóng vai trò Master và hai bo mạch ESP32 đảm nhận vai trò Slave. Giao diện người dùng cho phép cấu hình kết nối serial (COM port), bao gồm lựa chọn cổng, thiết lập baud rate, parity, số bit dữ liệu và stop bit. Trạng thái kết nối cần được hiển thị rõ ràng, giúp người vận hành dễ dàng nhận biết khi thiết bị đang trực tuyến hoặc bị ngắt kết nối.

Về chức năng giám sát, giao diện hiển thị dữ liệu cảm biến theo thời gian thực, bao gồm: nhiệt độ (°C), độ ẩm (%), nồng độ khí gas (ppm hoặc giá trị ADC), góc quay của servo và dữ liệu từ cảm biến hồng ngoại (IR sensor). Các giá trị này được đọc từ các thanh ghi Modbus RTU của từng ESP32 theo chu kỳ định sẵn, sau đó cập nhật trực tiếp lên các thành phần giao diện như QLabel, QProgressBar hoặc bảng QTableWidget để người dùng dễ quan sát.

Về chức năng điều khiển, giao diện hỗ trợ gửi lệnh bật/tắt và thay đổi trạng thái cho các thiết bị chấp hành như đèn, quạt, camera. Các thao tác này được thực hiện thông qua việc ghi dữ liệu vào các coil/register tương ứng trên ESP32. Sau khi gửi lệnh, hệ thống sẽ đọc lại trạng thái để xác nhận và phản hồi cho người dùng, ví dụ: đổi màu biểu tượng hoặc hiển thị nhãn trạng thái.

Bên cạnh đó, giao diện cần có nút lưu dữ liệu nhằm ghi lại các thông tin cảm biến và trạng thái thiết bị, cho phép người vận hành theo dõi sự thay đổi theo thời gian. Dữ liệu lưu trữ có thể phục vụ cho việc phân tích hoặc xuất báo cáo sau này.

Ngoài các chức năng cơ bản, hệ thống cũng cần cơ chế phản hồi trạng thái hoạt động của thiết bị ngay sau khi nhận lệnh để đảm bảo tính tin cậy. Đồng thời, giao diện có thể tích hợp thêm biểu đồ, thanh trạng thái kết nối hoặc bảng dữ liệu lịch sử để nâng cao trải nghiệm sử dụng.

Cuối cùng, hệ thống cần được thiết kế với khả năng mở rộng, cho phép bổ sung thêm cảm biến hoặc các Slave mới mà không ảnh hưởng đến hoạt động chung. Ngoài ra, cần có cơ chế xử lý lỗi kết nối hoặc mất liên lạc với Slave, kèm thông báo kịp thời để hỗ trợ vận hành an toàn và liên tục.

|  |  |  |
| --- | --- | --- |
| **STT** | **Chức năng** | **Mô tả chi tiết** |
| 1 | Kết nối với Server ESP32 qua Modbus RTU | Cho phép giao diện trên máy tính kết nối tới ESP32 thông qua cổng nối tiếp (USB RS-485) sử dụng giao thức Modbus RTU. |
| 2 | Hiển thị dữ liệu cảm biến | Giao diện hiển thị liên tục các giá trị cảm biến như: nhiệt độ, độ ẩm, khí gas, ánh sáng được gửi từ ESP32 Slave. |
| 3 | Cập nhật dữ liệu tự động | Giao diện tự động làm mới dữ liệu cảm biến sau mỗi khoảng thời gian định sẵn (ví dụ 1 giây) |
| 4 | Điều khiển thiết bị chấp hành | Người dùng có thể điều khiển các thiết bị như LED, quạt, camera bằng cách gửi lệnh ghi giá trị tới các thanh ghi (register) điều khiển. |
| 5 | Trạng thái kết nối | Giao diện thông báo tình trạng giao tiếp nối tiếp: kết nối thành công, mất tín hiệu, lỗi CRC hoặc không phản hồi từ Slave |
| 6 | Giao diện trực quan, dễ sử dụng | Thiết kế đơn giản, dễ thao tác với các nút, biểu đồ hoặc bảng hiển thị dữ liệu rõ ràng |
| 7 | Ghi lại log hoặc lưu dữ liệu (tuỳ chọn) | Lưu dữ liệu cảm biến vào file (CSV hoặc Excel) để phục vụ mục đích phân tích sau này |
| 8 | Khả năng mở rộng | Có thể thêm nhiều ESP32 Slave khác nhau (địa chỉ khác nhau) hoặc bổ sung thiết bị mà không cần thay đổi toàn bộ chương trình. |

*Bảng 4.1. Mô tả chi tiết các chức năng của giao diện*

## 4.2. Thiết kế giao diện bằng Qt Designer ( chạy trên VSCode )

Để xây dựng giao diện giám sát và điều khiển cho hệ thống nhà thông minh, sử dụng ngôn ngữ Python kết hợp với thư viện PyQt5. Việc thiết kế giao diện trực quan được thực hiện bằng công cụ Qt Designer, sau đó tích hợp vào mã nguồn Python và chạy trong môi trường Visual Studio Code (VSCode).

Qt Designer là công cụ hỗ trợ kéo–thả trực quan, giúp người dùng tạo giao diện bằng cách đặt các thành phần như nút bấm (QPushButton), nhãn (QLabel), ô nhập (QLineEdit), thanh hiển thị giá trị (QLCDNumber, QProgressBar),… Việc sử dụng Qt Designer giúp rút ngắn đáng kể thời gian lập trình giao diện và giảm sai sót khi bố cục thủ công bằng code.

Lệnh này sẽ sinh ra một file Python chứa toàn bộ cấu trúc giao diện mà Qt Designer đã tạo. File này có thể được import hoặc kế thừa trong chương trình chính.

Trong quá trình lập trình, đề tài sử dụng VSCode như môi trường phát triển chính do hỗ trợ tốt Python, tự động gợi ý cú pháp và dễ dàng tích hợp nhiều công cụ như pyuic5, pymodbus, pyqt5-tools, v.v. Ngoài ra, quá trình chạy thử, sửa lỗi và kiểm tra giao tiếp với ESP32 đều được thực hiện thuận tiện trên VSCode.

Nhờ việc tách biệt phần thiết kế giao diện (file .ui) và phần xử lý logic (mã Python), mà đề tài có thể phát triển và bảo trì chương trình một cách linh hoạt, dễ dàng chỉnh sửa giao diện mà không làm ảnh hưởng đến phần xử lý chính.

## 4.3. Hiển thị dữ liệu cảm biến và gửi lệnh điều khiển từ giao diện

Giao diện người dùng được xây dựng trên máy tính bằng Python, sử dụng thư viện PyQt5 kết hợp với pymodbus, cho phép kết nối đến từng địa chỉ IP của hai Slave thông qua giao thức Modbus RTU. Cụ thể, giao diện sẽ gửi lệnh đọc các thanh ghi (Function Code 0x03 – Read Holding Registers) từ từng ESP32 để nhận dữ liệu cảm biến như: nhiệt độ, độ ẩm, nồng độ khí gas và cường độ ánh sáng. Các dữ liệu này sau đó được cập nhật liên tục lên giao diện theo chu kỳ, sử dụng bộ hẹn thời gian (QTimer) của PyQt5.

Người dùng có thể trực tiếp điều khiển các thiết bị như đèn, quạt, hoặc kích hoạt camera ESP32-CAM ngay từ giao diện bằng cách nhấn các nút tương tác. Các nút này khi được nhấn sẽ gửi lệnh ghi thanh ghi hoặc ghi bit (write single coil/register) đến ESP32 tương ứng. Việc xác định ESP32 nào sẽ nhận lệnh được xử lý trong mã nguồn thông qua địa chỉ IP hoặc ID thiết bị.

Bên cạnh đó, trong quá trình triển khai sử dụng phần mềm Modbus Poll để kiểm thử kết nối và xác minh tính đúng đắn của dữ liệu được gửi và nhận từ các ESP32. Công cụ này giúp mô phỏng các lệnh Modbus RTU và theo dõi phản hồi của từng Slave trong thời gian thực, từ đó hỗ trợ việc gỡ lỗi và tinh chỉnh cấu hình thiết bị.

Các giá trị cảm biến như nhiệt độ, độ ẩm, khí gas và ánh sáng được hiển thị bằng các QLabel, giúp người dùng dễ dàng quan sát thông tin cập nhật theo thời gian thực. Các nút điều khiển QPushButton được sử dụng để gửi lệnh bật/tắt cho các thiết bị như đèn, quạt và camera. Ngoài ra, QLineEdit hoặc QStatusBar được dùng để hiển thị trạng thái kết nối của từng thiết bị ESP32, giúp người dùng biết được khi nào một kết nối bị lỗi hoặc mất tín hiệu.

Giao diện cũng có tiềm năng mở rộng để hiển thị thêm các biểu đồ trực quan như đường biểu diễn nhiệt độ theo thời gian, hoặc bảng dữ liệu dạng nhật ký để ghi lại lịch sử đo đạc. Điều này không chỉ tăng tính trực quan mà còn giúp hỗ trợ quá trình phân tích dữ liệu môi trường trong nhà.

Các lệnh đọc dữ liệu cảm biến được gửi đến ESP32 theo chu kỳ thông qua Modbus RTU để cập nhật giá trị mới, đồng thời các lệnh điều khiển thiết bị được truyền đi dưới dạng ghi vào các thanh ghi tương ứng. Việc cập nhật dữ liệu được thực hiện tự động theo thời gian thực nhờ vào QTimer, đảm bảo thông tin trên giao diện luôn đồng bộ với trạng thái thực tế của hệ thống.

## 4.4. Giám sát và hiển thị trên nền tảng Blynk

Bên cạnh giao diện giám sát và điều khiển trên máy tính, triển khai thêm giải pháp hiển thị dữ liệu và điều khiển thiết bị thông qua ứng dụng Blynk – một nền tảng IoT phổ biến hỗ trợ giao tiếp giữa thiết bị phần cứng và điện thoại thông minh. Việc tích hợp Blynk cho phép người dùng theo dõi và điều khiển hệ thống nhà thông minh mọi lúc mọi nơi, chỉ cần có kết nối Internet.

Trong mô hình của đề tài, các vi điều khiển ESP32 Slave không chỉ giao tiếp với máy tính (PC) qua Modbus RTU mà còn được cấu hình để kết nối trực tiếp với Blynk Cloud bằng cách sử dụng thư viện Blynk cho ESP32. Các cảm biến như nhiệt độ, độ ẩm, ánh sáng, khí gas được ESP32 đọc định kỳ và gửi dữ liệu lên nền tảng Blynk thông qua các virtual pins (chân ảo). Người dùng có thể quan sát giá trị cảm biến dưới dạng đồng hồ số, biểu đồ, hoặc trạng thái đèn màu trên giao diện điện thoại.

Song song với việc hiển thị, ứng dụng Blynk còn cung cấp các nút bấm ảo (button widget), cho phép người dùng điều khiển từ xa các thiết bị như đèn, quạt, hoặc bật/tắt camera ESP32-CAM. Khi người dùng nhấn nút trên app, một tín hiệu sẽ được gửi đến ESP32 thông qua server của Blynk, từ đó kích hoạt hành động tương ứng trên phần cứng.

Quá trình tích hợp Blynk diễn ra song song với giao thức Modbus RTU trên ESP32, nhưng được xử lý độc lập để đảm bảo không gây xung đột trong kết nối. Điều này giúp hệ thống có tính linh hoạt cao: trong trường hợp mất kết nối với máy tính, người dùng vẫn có thể duy trì khả năng giám sát và điều khiển thiết bị thông qua Blynk trên điện thoại.

Việc sử dụng nền tảng Blynk giúp nâng cao trải nghiệm người dùng, đồng thời mở rộng khả năng truy cập và điều khiển từ xa cho hệ thống giám sát nhà thông minh.

# **CHƯƠNG 5: KẾT QUẢ VÀ ĐÁNH GIÁ**

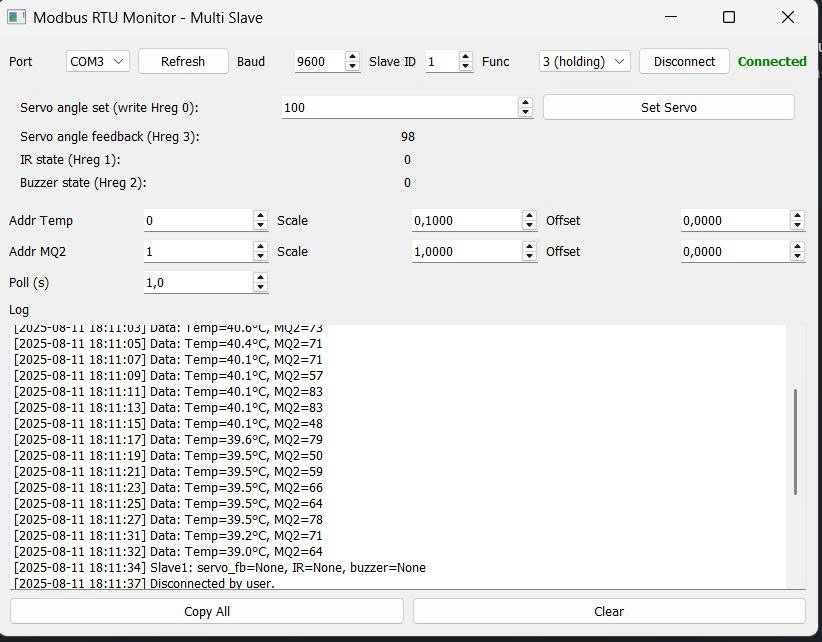
## 5.1. Kết quả thu được

Hệ thống nhà thông minh được triển khai thành công cả về phần cứng lẫn phần mềm, đáp ứng đúng mục tiêu đề ra. Về phần cứng, mô hình gồm hai phòng được điều khiển bởi hai module ESP32 đóng vai trò slave, trong khi máy tính đóng vai trò master thông qua giao thức Modbus RTU.

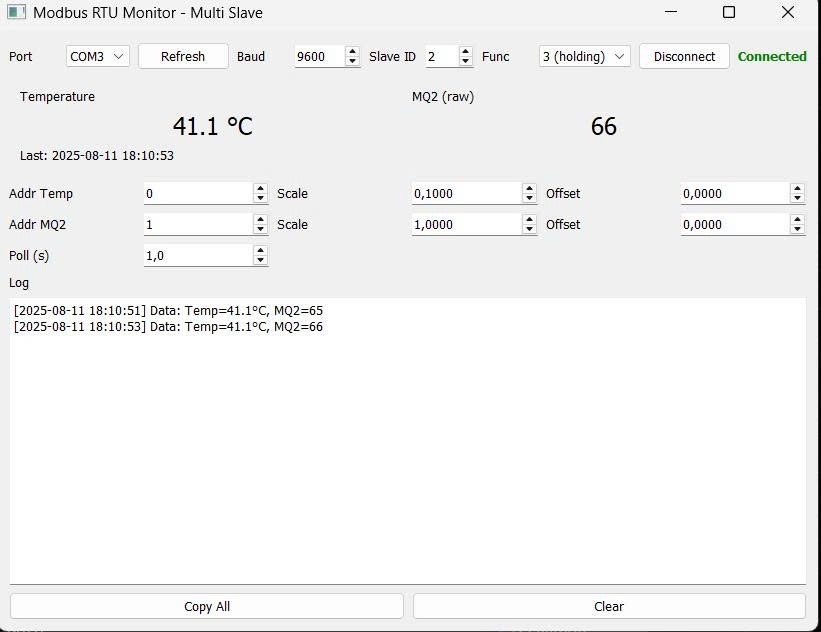
Phòng 1 sử dụng ESP32 (slave 1) kết nối với cảm biến IR, servo mini và buzzer. Cảm biến IR phát hiện chuyển động hoặc sự xuất hiện của con người để tự động điều khiển mở cửa qua servo và kích hoạt buzzer trong trường hợp cảnh báo (ví dụ: phát hiện người vào ban đêm). Servo mini đóng vai trò mô phỏng cơ cấu cửa tự động, tăng tính hiện đại và thuận tiện.

Phòng 2 sử dụng ESP32 (slave 2) với nhiều thiết bị hơn, gồm ESP32-CAM, LCD I2C, cảm biến MQ2, cảm biến DHT11, cảm biến IR, servo mini và buzzer. Cảm biến MQ2 phát hiện rò rỉ khí gas hoặc khói, DHT11 đo nhiệt độ và độ ẩm, kết quả được hiển thị trên màn hình LCD 16x2. Một cảm biến IR khác cũng điều khiển đóng/mở cửa bằng servo, trong khi buzzer phát âm thanh khi có cảnh báo khí gas hoặc môi trường bất thường.

Về phần mềm, nhóm đã sử dụng Modbus Poll để kiểm tra, giám sát và hiển thị giá trị cảm biến trong quá trình thử nghiệm giao thức Modbus RTU. Sau khi xác nhận việc truyền nhận dữ liệu ổn định, nhóm phát triển giao diện điều khiển và giám sát chính bằng QT5 Designer. Giao diện này cho phép hiển thị đầy đủ giá trị cảm biến, trạng thái cơ cấu chấp hành và cung cấp các nút điều khiển thiết bị từ máy tính.



Hình 5.1. Giao diện giám sát điều khiển slave1 ESP32



Hình 5.1. Giao diện giám sát điều khiển slave2 ESP32

Kết quả cuối cùng cho thấy hệ thống hoạt động ổn định: cảm biến đọc dữ liệu chính xác, các thiết bị chấp hành phản hồi nhanh chóng, dữ liệu truyền qua Modbus RTU ổn định, và giao diện QT5 Designer trực quan, dễ sử dụng.

Bên cạnh khả năng hiển thị và điều khiển, hệ thống đã được bổ sung chức năng lưu trữ dữ liệu để phục vụ việc phân tích và đánh giá sau này. Các giá trị cảm biến theo thời gian được ghi lại, từ đó có thể xuất báo cáo hoặc xây dựng biểu đồ biến thiên thông số môi trường. Điều này mở ra khả năng ứng dụng hệ thống cho các bài toán giám sát dài hạn, dự đoán xu hướng hoặc cảnh báo sớm khi phát hiện bất thường.

## 5.2. Độ trễ, độ ổn định và hiệu quả truyền nhận dữ liệu

Hệ thống truyền nhận dữ liệu giữa các node ESP32 và phần mềm giám sát hoạt động tương đối ổn định, không xảy ra mất gói hay sai lệch dữ liệu đáng kể trong quá trình thử nghiệm. Việc sử dụng giao thức Modbus RTU giúp duy trì tính đồng bộ và đảm bảo dữ liệu cảm biến được cập nhật đều đặn lên giao diện hiển thị.

Tuy nhiên, qua quá trình kiểm tra thực tế, nhận thấy cảm biến khói MQ2 và cảm biến nhiệt độ DHT11 vẫn tồn tại một độ trễ nhất định trong phản hồi. Cụ thể, khi loại bỏ nguồn nhiệt hoặc nguồn khói/gas ra khỏi khu vực đo, giá trị hiển thị trên giao diện vẫn duy trì ở mức cao thêm khoảng 1 giây trước khi giảm về giá trị bình thường. Nguyên nhân có thể do đặc tính của cảm biến cần thời gian để nguội hoặc để không khí xung quanh trở lại trạng thái bình thường, kết hợp với chu kỳ lấy mẫu và gửi dữ liệu của hệ thống.

Mặc dù vậy, độ trễ này không ảnh hưởng lớn đến khả năng phát hiện và cảnh báo, vì hệ thống vẫn đảm bảo tính ổn định và hiệu quả trong việc truyền nhận dữ liệu. Kết quả đo đạc cho thấy tần suất cập nhật dữ liệu ổn định, thông tin được hiển thị trên giao diện gần như tức thời với các cảm biến còn lại như cảm biến IR và trạng thái cửa servo.

## 5.3. Nhận xét và đánh giá hệ thống

Hệ thống điều khiển và giám sát nhà thông minh sử dụng giao tiếp Modbus RTU đã được triển khai thành công với đầy đủ phần cứng và phần mềm. Các thiết bị trong hai phòng được điều khiển và giám sát tập trung thông qua giao diện lập trình trên Qt5 Designer, đồng thời giá trị cảm biến cũng được hiển thị trên Modbus Poll để kiểm tra tính chính xác của quá trình truyền nhận.

Kết quả cho thấy hệ thống hoạt động ổn định, các cảm biến và cơ cấu chấp hành phản hồi chính xác theo yêu cầu. Chức năng phát hiện chuyển động bằng cảm biến IR kết hợp điều khiển servo mô phỏng cửa tự động và còi buzzer hoạt động tốt, mang lại tính trực quan và ứng dụng cao. Các cảm biến MQ2, DHT11 và LCD I2C ở phòng 2 hiển thị thông tin rõ ràng, giúp người dùng dễ dàng theo dõi tình trạng môi trường.

Tuy nhiên, hệ thống vẫn tồn tại một số hạn chế, điển hình là độ trễ khoảng 1 giây của cảm biến khói và nhiệt độ khi môi trường thay đổi đột ngột. Khi bỏ nguồn nhiệt hoặc khí ra, giá trị đo vẫn duy trì cao trong một khoảng thời gian ngắn trước khi trở về mức bình thường. Điều này xuất phát từ đặc tính vật lý của cảm biến và thời gian xử lý dữ liệu.

Nhìn chung, đề tài đã hoàn thành tốt mục tiêu đặt ra, đảm bảo tính ổn định, độ tin cậy và khả năng mở rộng. Hệ thống có thể tích hợp thêm nhiều thiết bị, cảm biến và chức năng khác để đáp ứng nhu cầu thực tế của một mô hình nhà thông minh hiện đại.

# **CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

## 6.1. Kết luận

Đề tài “Hệ thống điều khiển và giám sát nhà thông minh sử dụng giao tiếp Modbus RTU” đã được triển khai thành công, minh chứng cho khả năng ứng dụng của các mạng truyền thông công nghiệp vào các hệ thống tự động hóa dân dụng. Trong công nghiệp, Modbus RTU là một chuẩn giao tiếp phổ biến nhờ tính đơn giản, ổn định, khả năng kết nối nhiều thiết bị trên cùng một đường truyền, và đặc biệt phù hợp cho các môi trường yêu cầu giám sát – điều khiển tin cậy.

Việc áp dụng Modbus RTU trong đề tài cho phép kết nối đồng bộ các cảm biến và thiết bị chấp hành như DHT11, MQ2, cảm biến IR, servo, buzzer, LCD, ESP32-CAM,… thông qua cùng một hạ tầng truyền thông, tương tự cách các dây chuyền sản xuất hoặc hệ thống SCADA trong công nghiệp vận hành. Dữ liệu thu thập được hiển thị và điều khiển thông qua phần mềm Modbus Poll và giao diện trực quan thiết kế bằng QT5 Designer, đảm bảo tính trực quan và dễ vận hành.

Kết quả cho thấy hệ thống hoạt động ổn định, độ trễ thấp (~1 giây đối với một số cảm biến như khói và nhiệt độ), và có khả năng mở rộng để kết nối thêm thiết bị. Điều này chứng tỏ các nguyên lý và ưu điểm của mạng truyền thông công nghiệp hoàn toàn có thể áp dụng cho các giải pháp nhà thông minh, đồng thời mở ra tiềm năng tích hợp vào các hệ thống IoT công nghiệp (IIoT) quy mô lớn, đòi hỏi tính ổn định, bảo mật và khả năng mở rộng cao.

## 6.2. Hướng phát triển:

Trong tương lai, hệ thống “Điều khiển và giám sát nhà thông minh sử dụng giao tiếp Modbus RTU” có thể được mở rộng và nâng cấp để đáp ứng tốt hơn nhu cầu thực tế cũng như phù hợp với xu hướng công nghiệp 4.0. Trước hết, việc tích hợp thêm các mạng truyền thông công nghiệp như Modbus TCP/IP, EtherNet/IP hoặc PROFINET sẽ giúp nâng cao tốc độ và ổn định truyền dữ liệu, đồng thời dễ dàng kết nối với hạ tầng mạng máy tính công nghiệp hiện đại.

Bên cạnh đó, hệ thống có thể kết nối với phần mềm SCADA hoặc các thiết bị HMI để hỗ trợ giám sát – điều khiển từ xa với giao diện trực quan, đồng thời lưu trữ dữ liệu dài hạn và phân tích xu hướng vận hành. Khi kết hợp với các nền tảng IoT và điện toán đám mây, người dùng có thể truy cập và điều khiển thiết bị ở bất cứ đâu, đồng thời khai thác phân tích dữ liệu lớn hoặc trí tuệ nhân tạo để tối ưu hiệu suất và dự đoán sự cố.

Vấn đề bảo mật cũng cần được chú trọng khi hệ thống mở rộng kết nối ra ngoài. Việc triển khai cơ chế mã hóa dữ liệu, xác thực thiết bị và giám sát an ninh mạng sẽ giúp đảm bảo an toàn cho thông tin và thiết bị. Ngoài ra, hệ thống có thể được mở rộng số lượng và chủng loại cảm biến như cảm biến áp suất, lưu lượng, độ rung hoặc camera AI để áp dụng không chỉ trong nhà thông minh mà còn trong sản xuất, giám sát an toàn và bảo trì dự đoán.

Cuối cùng, việc tối ưu phần cứng và phần mềm nhằm giảm độ trễ, nâng cao tốc độ xử lý và cải thiện khả năng chịu tải khi kết nối nhiều thiết bị sẽ giúp hệ thống hoạt động ổn định hơn. Với các định hướng này, hệ thống có thể phát triển thành một mô hình thu nhỏ của mạng truyền thông công nghiệp hiện đại, phục vụ tốt cả trong đào tạo, nghiên cứu và ứng dụng thực tế.

# **TÀI LIỆU THAM KHẢO**

[1] A. B. Lasera and I. H. Wahyudi, "Pengembangan Prototipe Sistem Pengontrolan Daya Listrik berbasis IoT ESP32 pada Smart Home System," *Elinvo (Electronics, Informatics, and Vocational Education),* vol. 5, no. 2, pp. 112-120, 2020.

[2] X. Li, "Low cost intelligent household design based on Modbus/RTU protocol," in *Journal of Physics: Conference Series*, 2019, vol. 1345, no. 4: IOP Publishing, p. 042014.

[3] G. Yue, "Design of intelligent monitoring and control system based on modbus," in *2020 5th International conference on communication, image and signal processing (CCISP)*, 2020: IEEE, pp. 149-153.