ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HOC BÁCH KHOA
KHOA ĐIỆN TỬ – VIỄN THÔNG

BÁO CÁO CUỐI KÌ PBL3: CHUYÊN ĐỀ

Đề tài: **Hệ thống quản lí nhà xe thông minh**

Nhóm sinh viên thực hiện: Nguyễn Bá Thành **21DTCLC4**
Võ Đức Hiếu **21DTCLC4**
Hoàng Thị Hương Giang **21DTCLC4**

Người hướng dẫn **: TS Huỳnh Việt Thắng**
Th.S Vũ Vân Thanh

Đà Nẵng, ngày 5 tháng 12 năm 2024

TÓM TẮT

Tên đề tài: Hệ thống quản lý nhà xe thông minh

Nhóm Sinh viên thực hiện:

Hoàng Thị Hương Giang – 106210183 – 21DTCLC4

Võ Đức Hiếu – 106210184 – 21DTCLC4

Nguyễn Bá Thành – 106210200 – 21DTCLC4

Hệ thống quản lý nhà xe thông minh là một giải pháp được thiết kế với mục đích cung cấp khả năng giám sát và quản lý hiệu quả cho các nhà xe tại các khu vực như khu dân cư, trường học, trung tâm thương mại hoặc đô thị. Hệ thống này không chỉ hỗ trợ quan sát vị trí trống trong nhà xe mà còn tích hợp nhiều tính năng tiên tiến khác như cảnh báo cháy và điều khiển đèn tự động khi ánh sáng yếu hoặc trời tối. Đây là một giải pháp toàn diện, giúp đảm bảo sự an toàn, tiện lợi và tiết kiệm thời gian cho người sử dụng cũng như người quản lý. Trung tâm của hệ thống là vi điều khiển ESP32, một vi điều khiển mạnh mẽ và linh hoạt, được sử dụng để thu thập, xử lý dữ liệu từ các cảm biến khác nhau và hiển thị kết quả một cách trực quan. Các cảm biến được tích hợp trong hệ thống bao gồm cảm biến vật cản hồng ngoại, cảm biến ánh sáng và cảm biến cháy. Các tín hiệu từ các cảm biến này được gửi liên tục đến vi điều khiển ESP32 để xử lý và phân tích. Sau khi xử lý, vi điều khiển sẽ hiển thị kết quả thông qua hai phương thức. Giao diện Web người dùng có thể truy cập hệ thống qua trình duyệt để xem thông tin chi tiết về trạng thái nhà xe, bao gồm vị trí còn trống, tình trạng chiếu sáng, và cảnh báo cháy, bất kể ở đâu, chỉ cần có kết nối Internet. Màn hình Oled được tích hợp trực tiếp trong hệ thống để hiển thị thông tin cơ bản và các cảnh báo tại chỗ, phục vụ những người không sử dụng thiết bị di động hoặc máy tính. Hệ thống quản lý nhà xe thông minh không chỉ đơn thuần là một công cụ quản lý mà còn mang lại giá trị lớn trong việc tăng cường tính an toàn, tiết kiệm năng lượng và tối ưu hóa việc sử dụng không gian đỗ xe.

LỜI NÓI ĐẦU

Hiện nay, với sự gia tăng nhanh chóng về số lượng phương tiện, việc tìm kiếm chỗ đậu xe trở thành một thách thức lớn, gây lãng phí thời gian, nhiên liệu, và góp phần gia tăng ô nhiễm môi trường, cũng như giảm hiệu quả giao thông. Bên cạnh đó, các hệ thống điện trong nhà xe nếu không được bảo trì và lắp đặt đúng cách có nguy cơ cao dẫn đến chập điện và hỏa hoạn, gây ra những hậu quả nghiêm trọng.

Nhu cầu của người dùng ngày càng đòi hỏi các bãi đậu xe phải đáp ứng các tiêu chí tiện ích, hiện đại và an toàn. Để giải quyết những vấn đề này, việc tối ưu hóa không gian bãi xe và cung cấp thông tin vị trí trống một cách nhanh chóng là rất cần thiết, giúp tận dụng tối đa diện tích, giảm thiểu lãng phí, và tiết kiệm thời gian tìm kiếm chỗ đậu. Hơn nữa, hệ thống cảnh báo cháy nổ sớm đóng vai trò quan trọng trong việc phát hiện và xử lý kịp thời các tình huống khẩn cấp, hạn chế rủi ro và thiệt hại.

Ngoài ra, các phần mềm quản lý thông minh còn giúp chủ sở hữu bãi xe dễ dàng quản lý và theo dõi hoạt động, nâng cao hiệu quả vận hành. Một nhà xe thông minh không chỉ đáp ứng nhu cầu thực tế mà còn mang lại hình ảnh hiện đại, chuyên nghiệp, thu hút sự tin tưởng và hài lòng từ khách hàng.

Chính vì những lý do trên, nhóm chúng em quyết định lựa chọn đề tài: “Hệ thống quản lý nhà xe thông minh” nhằm nghiên cứu và phát triển giải pháp toàn diện, đáp ứng nhu cầu ngày càng cao của người dùng và đóng góp vào xu hướng đô thị thông minh.

Tuy nhiên trong học tập, cũng như trong quá trình làm đồ án không thể tránh khỏi những thiếu sót, chúng em rất mong được sự góp ý quý báu của thầy cũng như các bạn để em có cái nhìn trực quan và kết quả được hoàn thiện hơn cũng như có thêm nhiều kinh nghiệm cho công việc sau này.

Trong quá trình thực hiện đề tài, nhóm chúng em xin gửi lời cảm ơn chân thành đến TS. Huỳnh Việt Thắng và ThS. Vũ Vân Thanh vì sự hướng dẫn tận tình và những đóng góp quý báu. Sự hỗ trợ và chỉ dẫn của thầy đã giúp nhóm hoàn thiện đề tài một cách tốt nhất.

CAM ĐOAN

Chúng tôi xin khẳng định rằng tất cả nội dung trong báo cáo này là kết quả của quá trình nghiên cứu, tìm hiểu và thực hiện của nhóm, dưới sự hướng dẫn tận tình của giảng viên phụ trách.

Mọi tài liệu, số liệu và thông tin tham khảo được sử dụng trong báo cáo đều được trích dẫn nguồn gốc rõ ràng và tuân thủ đúng quy định. Nhóm cam kết không sao chép hoặc sử dụng trái phép bất kỳ nội dung nào từ tài liệu, báo cáo hay đồ án khác mà không có sự cho phép hoặc không được trích dẫn.

Chúng tôi hoàn toàn chịu trách nhiệm về mọi hành vi gian lận hoặc vi phạm liêm chính học thuật trong quá trình thực hiện đồ án. Nếu có bất kỳ vi phạm nào xảy ra, nhóm sẽ chịu hoàn toàn trách nhiệm trước nhà trường và các cơ quan liên quan.

Chúng tôi xin cam đoan rằng những nội dung trong đồ án này được thực hiện dưới sự hướng dẫn của TS. Huỳnh Việt Thắng và TH.S. Vũ Vân Thanh, cùng với việc nghiên cứu từ Internet, sách báo và các tài liệu trong và ngoài nước có liên quan. Chúng tôi không sao chép hay sử dụng bài làm của bất kỳ ai khác. Mọi tham khảo trong đồ án đều được trích dẫn rõ ràng, bao gồm tên tác giả, tên công trình, thời gian và địa điểm công bố.

Chúng tôi xin chịu hoàn toàn trách nhiệm về lời cam đoan này trước thầy và nhà trường.

Nhóm sinh viên thực hiện

Nguyễn Bá Thành Hoàng Thị Hương Giang Võ Đức Hiếu

MỤC LỤC

TÓM TẮT **1**

LỜI NÓI ĐẦU **2**

CAM ĐOAN **3**

MỤC LỤC **4**

DANH SÁCH CÁC BẢNG, HÌNH VẼ **6**

DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT **6**

MỞ ĐẦU **7**

Chương 1: CƠ SỞ LÝ THUYẾT **8**

1.1. Giới thiệu về Internet of Things **8**

1.2. Giới thiệu về MCU ESP32 ..... **8**

1.2.1 Sơ lược về vi điều khiển ESP32 ..... **8**

1.2.2 So sánh giữa vi điều khiển ESP8266 và ESP32 ..... **10**

1.3. Cảm biến cháy (Phát hiện lửa, khói) - Flame Sensor KY-026 ..... Error! Bookmark not defined.

1.4. Nút nhấn **11**

1.6. Màn hình Oled ..... Error! Bookmark not defined.

1.7. Module phát hiện vật cản hồng ngoại ..... Error! Bookmark not defined.

1.8. Cảm biến ánh sáng ..... Error! Bookmark not defined.

1.9. Phần mềm sử dụng **16**

1.9.1 VS Code **16**

1.9.2. Arduino IDE **17**

Chương 2: THIẾT KẾ HỆ THỐNG **18**

2.1. Sơ đồ khối của hệ thống **18**

2.2. Quy trình hoạt động của hệ thống **18**

2.3.1 Khối nguồn (5V-12V) **19**

2.3.2 Cảm biến ánh sáng LM393 ..... **21**

2.3.3. Cảm biến cháy(Flame Sensor KY-026) **21**
2.3.4. Cảm biến vật cản hồng ngoại **21**
2.4. Khối điều khiển và động cơ AC, động cơ DC **22**
2.4.1 Sơ đồ vi điều khiển, động cơ DC, động cơ AC **22**
2.4.2. Vi điều khiển ESP32 ..... **22**
2.4.3 Động cơ DC(Máy bơm nước) **22**
2.4.4. Động cơ AC(Đèn AC) **23**
2.5. Mạng truyền thông **23**
2.6. Trung tâm xử lý **23**
2.7. Giao diện người dùng **23**
2.7.1 Ứng dụng web **23**
2.7.2. Màn hình Oled **23**
Chương 3 KẾT QUẢ VÀ ĐÁNH GIÁ **24**
3.1. Kết quả **24**
3.1.1 Phần cứng **24**
3.1.2 Lưu trữ dữ liệu **25**
3.1.3. Giao diện Web **25**
3.2. Đánh giá **26**
3.2.1 Kết quả hiển thị trên Web **26**
3.2.2. Đánh giá tổng quan hệ thống **27**
KẾT LUẬN **28**
TÀI LIỆU THAM KHẢO **30**
PHỤ LỤC 30

DANH SÁCH CÁC BẢNG, HÌNH VẼ

BẢNG 1.1 So sánh vi điều khiển ESP8266 và ESP32

HÌNH 1.1 Vi điều khiển ESP32

HÌNH 1.2 Cảm biến cháy – Flame Sensor KY-026

HÌNH 1.3 Nút nhấn tròn

HÌNH 1.4 Máy bơm nước

HÌNH 1.5 Màn hình Oled

HÌNH 1.6 Module phát hiện vật cản hồng ngoại

HÌNH 1.7 Module cảm biến ánh sáng

HÌNH 2.1 Sơ đồ khối của hệ thống

HÌNH 2.2 Sơ đồ nối dây hệ thống

HÌNH 2.3 Sơ đồ mạch nguồn 5V-12V

HINH 2.4 Mạch Relay DC (Cảm biến cháy + Động cơ bơm nước DC)

HÌNH 2.5 Sơ đồ vi điều khiển, động cơ AC, động cơ DC

HÌNH 3.1 Sơ đồ phần cứng hệ thống

HÌNH 3.2 Dữ liệu lưu trữ trong MQTTX

HÌNH 3.3 Giao diện đăng nhập vào hệ thống

HÌNH 3.4 Giao diện điều khiển

DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

KÝ HIỆU:

CHỮ VIẾT TẮT:

IoT: Internet of Things

ESP: Espressif Systems

Wifi: Wireless Fidelity

MỞ ĐẦU

Đề tài: Hệ thống quản lý nhà xe thông minh được thực hiện nhằm giải quyết các vấn đề đang tồn tại trong việc quản lý bãi đậu xe hiện nay, đặc biệt là tình trạng ùn tắc, thiếu chỗ đậu xe, và nguy cơ cháy nổ tại các nhà xe. Với sự phát triển nhanh chóng của số lượng phương tiện, việc tìm kiếm một chỗ đậu xe phù hợp trở thành thách thức lớn, gây lãng phí thời gian, nhiên liệu và làm gia tăng ô nhiễm môi trường. Đề tài hướng đến việc tối ưu hóa quản lý bãi xe thông qua hệ thống hiển thị vị trí trống, giúp tận dụng tối đa diện tích bãi đậu xe và giảm thiểu tình trạng lãng phí. Người dùng có thể dễ dàng tìm thấy chỗ đậu xe nhanh chóng, tiết kiệm thời gian tìm kiếm và giảm sự bất tiện. Đồng thời, hệ thống tích hợp các tính năng cảnh báo cháy nổ sớm, giúp phát hiện và xử lý kịp thời các tình huống khẩn cấp, hạn chế thiệt hại về người và tài sản.

Hệ thống không chỉ hỗ trợ người sử dụng mà còn cung cấp các công cụ mạnh mẽ giúp chủ sở hữu bãi xe dễ dàng quản lý và theo dõi hoạt động thông qua các phần mềm hiện đại, tăng hiệu quả vận hành và nâng cao chất lượng dịch vụ. Nhà xe thông minh không chỉ giải quyết các vấn đề hiện tại mà còn mang lại hình ảnh chuyên nghiệp, hiện đại, tạo ấn tượng tốt với khách hàng và góp phần xây dựng đô thị thông minh trong tương lai. Dự án tập trung triển khai tại các bãi đậu xe ô tô, sử dụng vi điều khiển ESP32 cùng các cảm biến vật cản hồng ngoại, cảm biến lửa, cảm biến ánh sáng nhằm đảm bảo hiệu quả và độ chính xác cao trong hoạt động của hệ thống.

Cấu trúc của đồ án bao gồm các nội dung: lý do chọn đề tài, mục tiêu nghiên cứu, phương pháp thực hiện, thiết kế hệ thống, thực nghiệm và đánh giá kết quả. Bên cạnh đó, đồ án cũng xem xét các rủi ro tiềm tàng trong quá trình triển khai và đưa ra định hướng phát triển để nâng cao khả năng ứng dụng thực tế trong tương lai. Với những lợi ích mà hệ thống mang lại, đề tài không chỉ có ý nghĩa trong việc cải thiện chất lượng dịch vụ bãi đậu xe mà còn góp phần bảo vệ môi trường và nâng cao hiệu quả quản lý đô thị.

Chương 1: CƠ SỞ LÝ THUYẾT

1.1. Giới thiệu về Internet of Things

Ngày nay, nhu cầu phát triển các ứng dụng liên quan đến Internet ngày càng cao. Và IoT (Internet of things) là một công nghệ quan trọng bởi chúng ta có thể tạo ra nhiều ứng dụng đa dạng phục vụ đa số mọi lĩnh vực trong đời sống từ nó.

Về cơ bản, IoT là một hệ thống mạng lưới mà trong đó tất cả các thiết bị, đối tượng được kết nối Internet thông qua thiết bị mạng (network devices) hoặc các bộ định tuyến (routers). IoT cho phép các đối tượng được điều khiển từ xa dựa trên hệ thống mạng hiện tại. Công nghệ tiên tiến này giúp giảm công sức vận hành của con người bằng cách tự động hóa việc điều khiển các thiết bị.

Các thành phần chính trong một hệ thống IoT:

- Thiết bị: Mỗi thiết bị sẽ bao gồm một hoặc nhiều cảm biến để phát hiện các thông số của ứng dụng và gửi chúng đến Platform.
- IoT – Platform: Nền tảng này là một phần mềm được lưu trữ trực tuyến còn được gọi là điện toán đám mây, các thiết bị được kết nối với nhau thông qua nó. Nền tảng này thu thập dữ liệu từ thiết bị, toàn bộ dữ liệu được phân tích, xử lý, phát hiện nếu có lỗi phát sinh trong quá trình hệ thống vận hành.
- Kết nối Internet: Để giao tiếp được trong IoT, kết nối Internet của các thiết bị là một điều bắt buộc. Wifi là một trong những phương thức kết nối Internet phổ biến.
- Ứng dụng: Giao diện để người dùng điều khiển.

1.2. Giới thiệu về MCU ESP32

1.2.1 Sơ lược về vi điều khiển ESP32

ESP32 là một nền tảng mạnh mẽ và tiết kiệm chi phí để phát triển các ứng dụng IoT. ESP32, do Công ty Espressif Systems (Thượng Hải, Trung Quốc) phát triển, cung cấp sự kết hợp mạnh mẽ giữa các tính năng và khả năng cho các ứng dụng IoT. ESP32 có các tính năng sau: (1) bộ xử lý lõi kép, (2) kết nối Wi-Fi và Bluetooth tích hợp,(3) số lượng lớn các chân đầu vào/đầu ra (GPIO) mục đích chung và (4) mức tiêu thụ điện năng thấp. ESP32 có giao diện Wi-Fi và Bluetooth tích hợp giúp đơn giản hóa kết nối và giao tiếp với các thiết bị khác. Nó hỗ trợ nhiều giao thức Wi-Fi khác nhau, chẳng hạn như 802.11 b/g/n và cung cấp các tùy chọn kết nối Bluetooth Classic và Bluetooth Low Energy (BLE). ESP32 cung cấp nhiều chân GPIO giúp kết nối và điều khiển các thiết bị và cảm biến bên ngoài. Các chân này hỗ trợ nhiều giao diện khác nhau, bao gồm SPI, I2C, UART và PWM. ESP32 được thiết kế để tiết kiệm điện năng, do đó cho phép

phát triển các ứng dụng IoT tiết kiệm năng lượng. ESP32 có thể được kết nối với màn hình, màn hình cảm ứng hoặc đèn báo LED để cung cấp cho người vận hành hoặc nhân viên một giao diện thân thiện với người dùng. ESP32 có thể được lập trình bằng nhiều khuôn khổ và ngôn ngữ phát triển khác nhau. Ngôn ngữ lập trình được sử dụng phổ biến nhất là C++ và có thể được lập trình bằng Arduino IDE hoặc PlatformIO. Một số mô-đun đi kèm cảm biến tích hợp, giúp đơn giản hóa việc tích hợp các khả năng cảm biến này vào các dự án IoT. Ngoài ra, mô-đun bao gồm các tính năng sau:

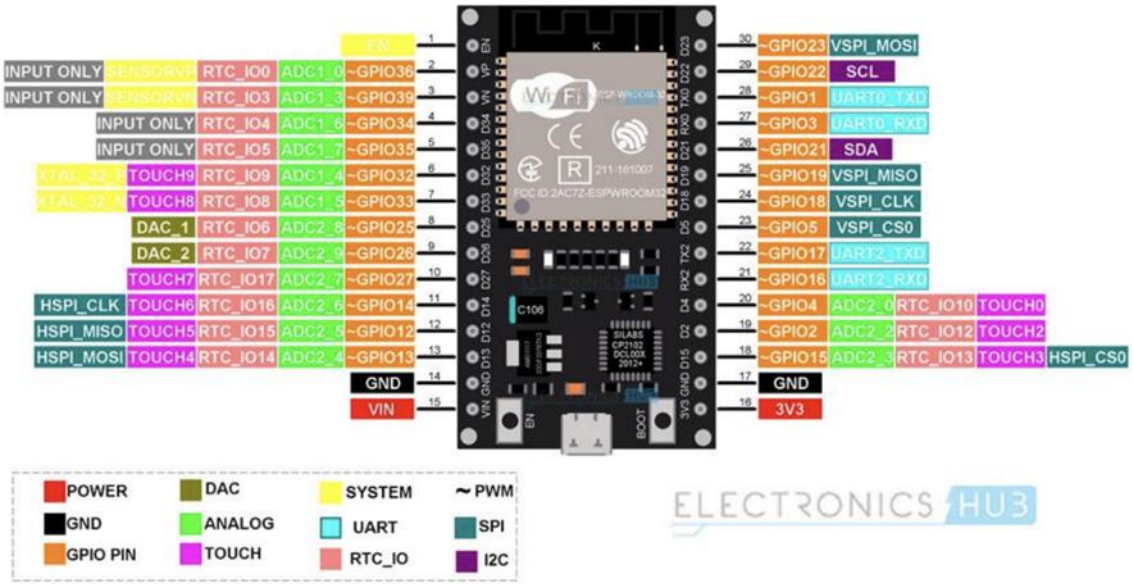
• Bộ xử lý: ESP32 (lõi kép 24 MHz);

• Bộ nhớ flash: 4 MB;

• Đầu nối thẻ nhớ microSD tích hợp;

• PSRAM (bộ nhớ truy cập ngẫu nhiên giả tĩnh): 8 MB;

• Bộ chuyển đổi Wi-Fi, Bluetooth, USB sang nối tiếp tích hợp (CP2104 hoặc CH91)



Hình 1.1. Vi điều khiển ESP32

1.2.2 So sánh giữa vi điều khiển ESP8266 và ESP32

Hiện nay, trong các hệ thống IoT, hai loại vi điều khiển phổ biến nhất và được ưa chuộng rộng rãi là ESP32 và ESP8266. Cả hai đều được sản xuất bởi công ty Espressif và đều nổi bật với khả năng kết nối mạng Wi-Fi, tuy nhiên mỗi loại lại có những đặc điểm riêng phù hợp với các nhu cầu sử dụng khác nhau. ESP8266 là loại vi điều khiển có từ sớm, nổi bật với giá thành thấp, dễ sử dụng, và khả năng kết nối Wi-Fi ổn định. Vì lý do này, ESP8266 thường được lựa chọn cho các ứng dụng IoT đơn giản, không đòi hỏi quá nhiều khả năng xử lý phức tạp. ESP8266 có thể thực hiện tốt các tác vụ cơ bản như thu thập dữ liệu từ cảm biến hoặc điều khiển các thiết bị thông minh trong nhà. Trong khi đó, ESP32 là phiên bản nâng cấp với nhiều tính năng vượt trội hơn. Được trang bị bộ xử lý lõi kép mạnh mẽ với tốc độ cao, ESP32 có khả năng xử lý nhiều tác vụ đồng thời, cùng với đó là hỗ trợ cả kết nối Wi-Fi và Bluetooth, bao gồm cả Bluetooth Low Energy (BLE). Nhờ vào khả năng kết nối đa dạng này, ESP32 phù hợp cho các ứng dụng yêu cầu tốc độ xử lý cao, độ phức tạp lớn, và có thể hoạt động hiệu quả trong các môi trường cần tiết kiệm năng lượng. Esp32 có nhiều chân. Vì vậy, nên lựa chọn Esp32 làm vi điều khiển chính của hệ thống.

Bảng 1.1. So sánh vi điều khiển ESP8266 và ESP32

	ESP8266	ESP32
Giá thành	Rẻ	Đắt
Hiệu năng	Thấp	Cao
Wifi/Bluetooth	Có Wifi, không có Bluetooth	Có Wifi và Bluetooth
Tính năng	Ít	Nhiều

1.3. Cảm biến cháy (Phát hiện lửa, khói) - Flame Sensor KY-026

Hình 1.2. Cảm biến cháy – Flame Sensor KY-026

Cảm biến phát hiện lửa- Flame sensor KY-026 dùng để phát hiện lửa, thường dùng trong các hệ thống báo cháy. Sử dụng cảm biến hồng ngoại YG1006 với tốc độ đáp ứng nhanh và độ nhạy cao. Tích hợp IC LM393 để chuyển đổi ADC, tạo 2 ngõ ra cả số và tương tự, rất linh động trong việc sử dụng. Biến trở để tùy chỉnh độ nhạy cảm biến.

Nguyên lí hoạt động:

- Phát ánh sáng hồng ngoại: Diode phát quang hồng ngoại liên tục phát ra ánh sáng hồng ngoại vào môi trường xung quanh.
- Phát hiện ngọn lửa: Nếu có ngọn lửa xuất hiện trong tầm hoạt động của cảm biến,
- So sánh cường độ ánh sáng: Phototransistor sẽ chuyển đổi ánh sáng nhận được thành tín hiệu điện.
- Đưa ra tín hiệu đầu ra: Nếu cường độ ánh sáng nhận được vượt quá ngưỡng cài đặt, cho biết đã phát hiện thấy ngọn lửa.
- Điều chỉnh độ nhạy: người dùng tùy chỉnh độ nhạy của cảm biến cho phù hợp với từng ứng dụng.

1.4. Relay 5V

Hình 1.3. Relay 5V

Thông số kỹ thuật:

- Kích thước: 19.1 x 15.5 x 15.3mm.
- Điện áp cuộn dây: 5V.
- Điện áp tiếp điểm: 125,250VAC.
- Dòng điện tiếp điểm: 10A.
- Nhiệt độ hoạt động: -45 đến 75 độ C.
- Công suất cuộn dây: mW.
- Thời gian tác động: 10ms.
- Thời gian nhả hãm: 5ms.

1.5. Máy bơm

Thông số kỹ thuật:

- Loại động cơ DC: 385.
- Điện áp sử dụng: 5V - 12VDC
- Hoạt động ở dòng điện: 0.5-0.7A

Hình 1.4 Máy bơm nước

1.6. Màn hình Oled

Hình 1.5. Màn hình OLED

Màn hình Oled 1.3 inch hiển thị đẹp, sang trọng, rõ nét vào ban ngày và khả năng tiết kiệm năng lượng tối đa với mức chi phí phù hợp, màn hình sử dụng giao tiếp I2C cho chất lượng đường truyền ổn định và rất dễ giao tiếp chỉ với 2 chân GPIO.

Thông tin kỹ thuật:

- Điện áp sử dụng: 3.3 - 5.5 VDC.
- Công suất tiêu thụ: 0.04 W.
- Góc hiển thị: lớn hơn 160 độ.
- Số điểm hiển thị: 128x64 dot.
- Độ rộng màn hình: 1.3 inch.
- Màu hiển thị: Trắng / Xanh Dương.
- Giao tiếp: I2C

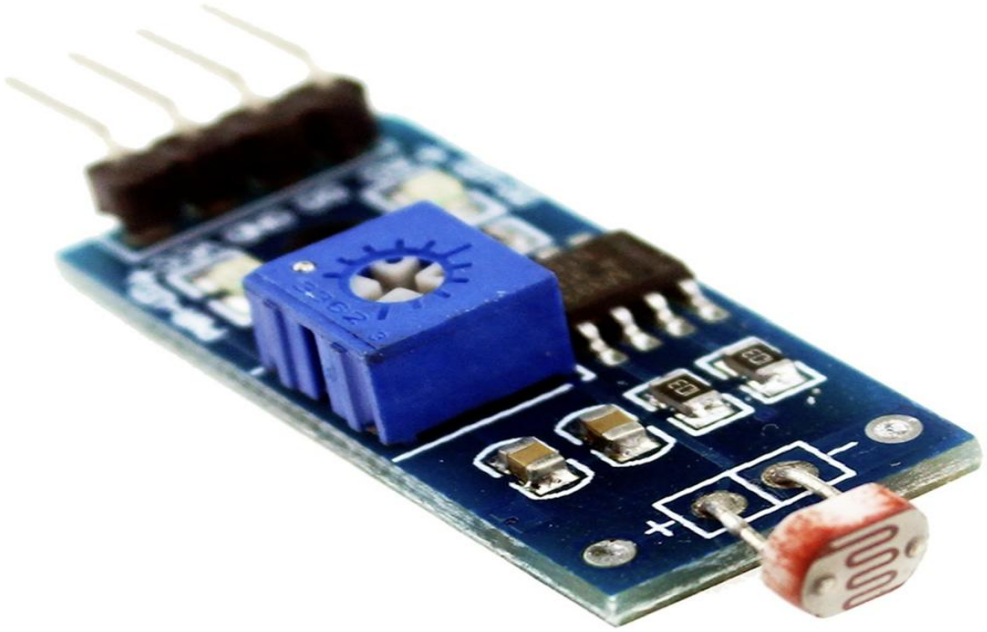
1.7. Module phát hiện vật cản hồng ngoại

Module cảm biến một cặp truyền và nhận tia hồng ngoại. Tia hồng ngoại phát ra từ led phát với một tần số nhất định, khi phát hiện hướng truyền có vật cản (mặt phản xạ), phản xạ vào đèn thu hồng ngoại, sau khi so sánh, đèn báo hiệu sẽ sáng lên, đồng thời đầu cho tín hiệu số đầu ra. Khoảng cách làm việc hiệu quả 2 ~ 5cm, điện áp làm việc là 3.3 V đến 5V. Độ nhạy sáng của cảm biến được điều chỉnh bằng chiết áp, cảm biến dễ lắp ráp, dễ sử dụng. Module cảm biến có thể được sử dụng rộng rãi trong robot tránh chướng ngại vật, xe tránh chướng ngại vật và dò đường.

Hình 1.6. Module phát hiện vật cản hồng ngoại 5V

- Điện áp sử dụng: 3.3~5vDC
- Nhận biết vật cản bằng ánh sáng hồng ngoại.
- Ngõ ra: Digital TTL
- Tích hợp biến trở chỉnh khoảng cách nhận biết vật cản.
- Kích thước: 3.2 x 1.4cm

1.8. Cảm biến ánh sáng



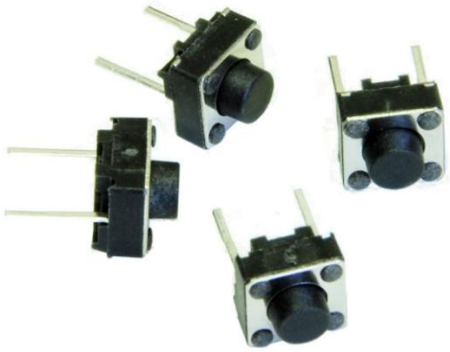
Hình 1.7. Module cảm biến ánh sáng

Module cảm biến ánh sáng là một linh kiện điện tử có điện trở thay đổi giảm theo ánh sáng chiếu vào. Quang trở làm bằng chất bán dẫn trở kháng cao, và không có tiếp giáp nào. Trong bóng tối, quang trở có điện trở đến vài MΩ. Khi có ánh sáng, điện trở giảm xuống mức một vài trăm Ω Độ chính xác: ± 5%RH và ±2o. Hoạt động của quang trở dựa trên hiệu ứng quang điện trong khối vật chất. Khi photon có năng lượng đủ lớn đập vào, sẽ làm bật electron khỏi phân tử, trở thành tự do trong khối chất và làm chất bán dẫn thành dẫn điện. Mức độ dẫn điện tuỳ thuộc số photon được hấp thụ.

Thông số kỹ thuật:

- Điện áp hoạt động: 3.3V-5V
- Kích thước PCB: 3cm * 1.6cm
- Led xanh báo nguồn và ánh sáng
- IC so sánh : LM393
- VCC: 3.3V-5V

1.9 Nút nhấn



Hình 1.3. Nút nhấn tròn

Nút nhấn tự giữ nhấn xuống đóng mạch kín, thả tay ra vẫn ở trạng thái đóng mạch. Nhấn lần nữa nút trở về trạng thái hở mạch.

1.10. Phần mềm sử dụng

1.10.1 VS Code

Các đặc điểm nổi bật của VS Code:

- Hỗ trợ đa ngôn ngữ lập trình: VS Code hỗ trợ nhiều ngôn ngữ lập trình như, C++, Java, PHP, HTML, CSS, Go, Rust, và nhiều ngôn ngữ khác.
- Tính năng IntelliSense: IntelliSense cung cấp tính năng gợi ý mã thông minh, tự động hoàn thành và kiểm tra lỗi cú pháp, giúp lập trình nhanh hơn và chính xác hơn.
- Khả năng mở rộng mạnh mẽ: VS Code có một kho tiện ích mở rộng phong phú trên Visual Studio Code Marketplace, cho phép người dùng tùy chỉnh công cụ theo nhu cầu.
- Giao diện đơn giản, thân thiện: Giao diện của Vs Code hiện đại, tối ưu cho người dùng, cho phép tùy chỉnh màu sắc, chủ đề (themes) và phím tắt theo sở thích cá nhân.
- Trình gỡ lỗi (Debugger): VS Code hỗ trợ gỡ lỗi trực tiếp trong trình biên tập, giúp theo dõi và sửa lỗi ứng dụng một cách hiệu quả.
- Tính năng tích hợp Terminal: VS Code có một terminal tích hợp ngay bên trong, giúp lập trình viên dễ dàng thực thi lệnh mà không cần mở thêm cửa sổ khác.

1.10.2. Arduino IDE

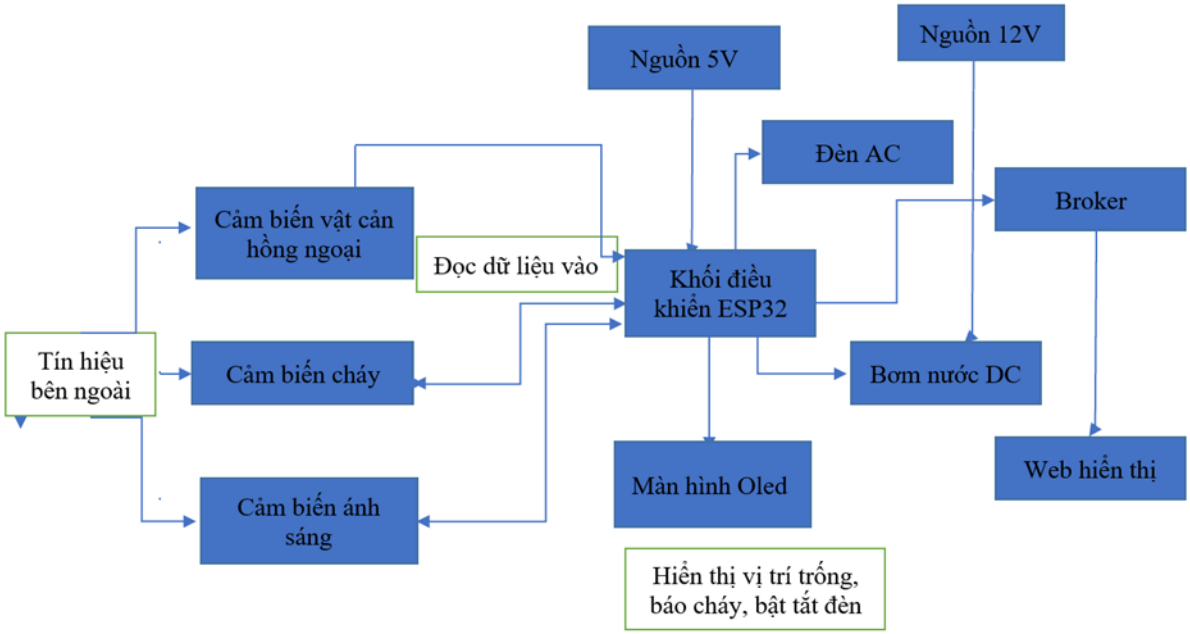
Phần mềm Arduino IDE là một môi trường phát triển tích hợp mã nguồn mở, được thiết kế đặc biệt để lập trình và quản lý các vi điều khiển trong các bo mạch Arduino.

Đặc điểm:

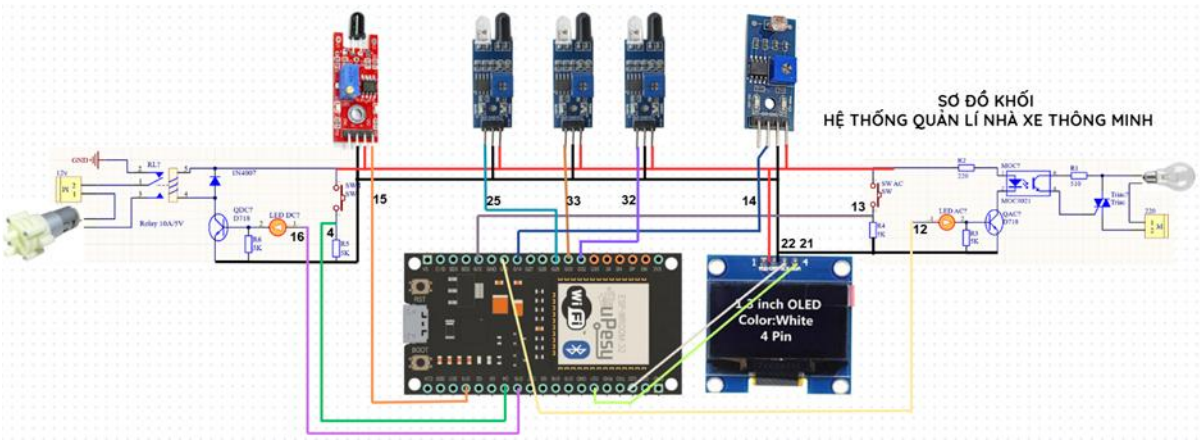
- Giao diện đơn giản, thân thiện: Arduino IDE được thiết kế với giao diện trực quan, dễ sử dụng, phù hợp với cả người mới bắt đầu và các nhà phát triển chuyên nghiệp.
- Ngôn ngữ lập trình: Sử dụng ngôn ngữ lập trình dựa trên C/C++, cùng với một thư viện phong phú hỗ trợ việc viết mã nhanh chóng và hiệu quả.
- Trình biên dịch tích hợp: Arduino IDE tích hợp trình biên dịch (compiler) để chuyển đổi mã nguồn thành mã máy, có thể chạy trên bo mạch Arduino.
- Công cụ nạp chương trình: Phần mềm cho phép tải mã nguồn đã biên dịch trực tiếp lên bo mạch thông qua kết nối USB.
- Thư viện phong phú: Arduino IDE hỗ trợ một hệ thống thư viện đa dạng, giúp người dùng dễ dàng tích hợp các chức năng như điều khiển động cơ, giao tiếp mạng, cảm biến, và nhiều ứng dụng khác.
- Hỗ trợ nhiều bo mạch: Arduino IDE tương thích với hầu hết các bo mạch Arduino như Arduino Uno, Arduino Nano, Arduino Mega, và nhiều phiên bản khác. Ngoài ra, nó còn hỗ trợ các bo mạch không phải của Arduino, nhờ tính năng tùy chỉnh bảng (board).
- Cộng đồng hỗ trợ mạnh mẽ: Là một công cụ mã nguồn mở, Arduino IDE được hỗ trợ bởi một cộng đồng đông đảo trên toàn thế giới. Người dùng có thể dễ dàng tìm thấy tài liệu, hướng dẫn, và giải pháp cho các vấn đề liên quan.
- Khả năng mở rộng: Arduino IDE cho phép người dùng cài đặt các thư viện và tiện ích bổ sung để mở rộng chức năng, giúp phát triển các ứng dụng phức tạp hơn.

Chương 2: THIẾT KẾ HỆ THỐNG

2.1. Sơ đồ khối của hệ thống



Hình 2.1 Sơ đồ khối hệ thống



Hình 2.2 Sơ đồ nối dây hệ thống

2.2. Quy trình hoạt động của hệ thống

- Thu thập dữ liệu: Thu thập dữ liệu từ môi trường thông qua các cảm biến ánh sáng, cháy, siêu âm. Dữ liệu thu được sẽ được chuyển đến khối vi điều khiển trung tâm để xử lý.

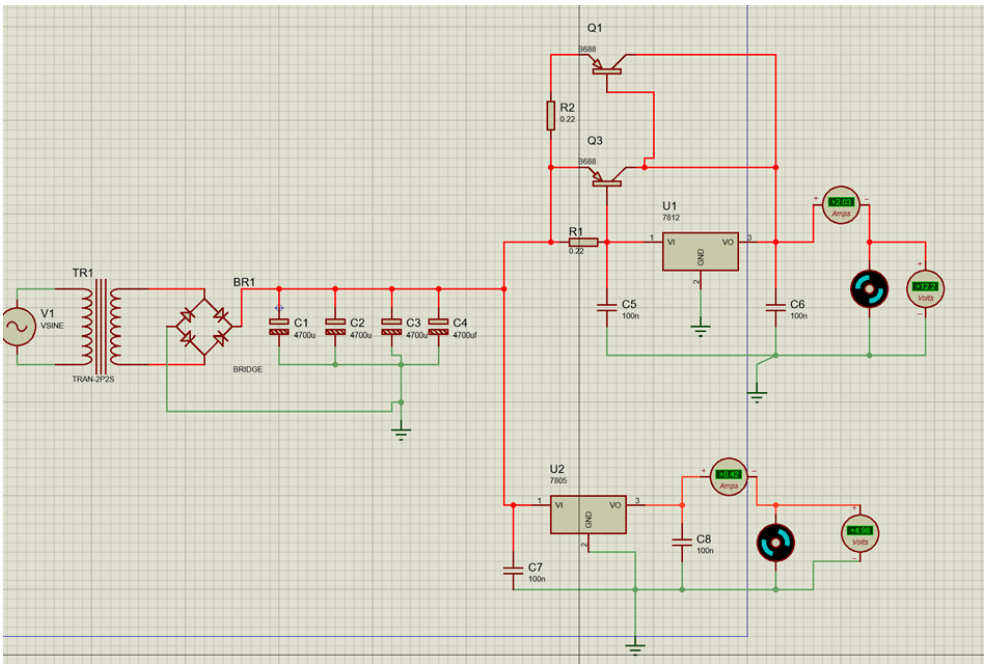
- Truyền dữ liệu: Dữ liệu từ bộ vi điều khiển được gửi qua Wi-Fi đến trung tâm xử lý.
- Xử lý dữ liệu: Tiếp nhận lệnh từ các cảm biến, sau đó điều khiển các thiết bị thực thi như động cơ, đèn để thực hiện các tác vụ theo yêu cầu.
- Hiển thị và cảnh báo: Hiển thị trực quan các thông tin về tình trạng nhà xe như số lượng chỗ trống, tình trạng cháy giúp người dùng dễ dàng theo dõi và quản lý.

2.3. Thành phần chi tiết của hệ thống

2.3.1 Khối nguồn (5V-12V)

Khối nguồn: Cung cấp nguồn điện ổn định cho toàn bộ hệ thống. Cung cấp nguồn cho ESP32, động cơ bơm nước.

Sơ đồ nguồn 5V-12V

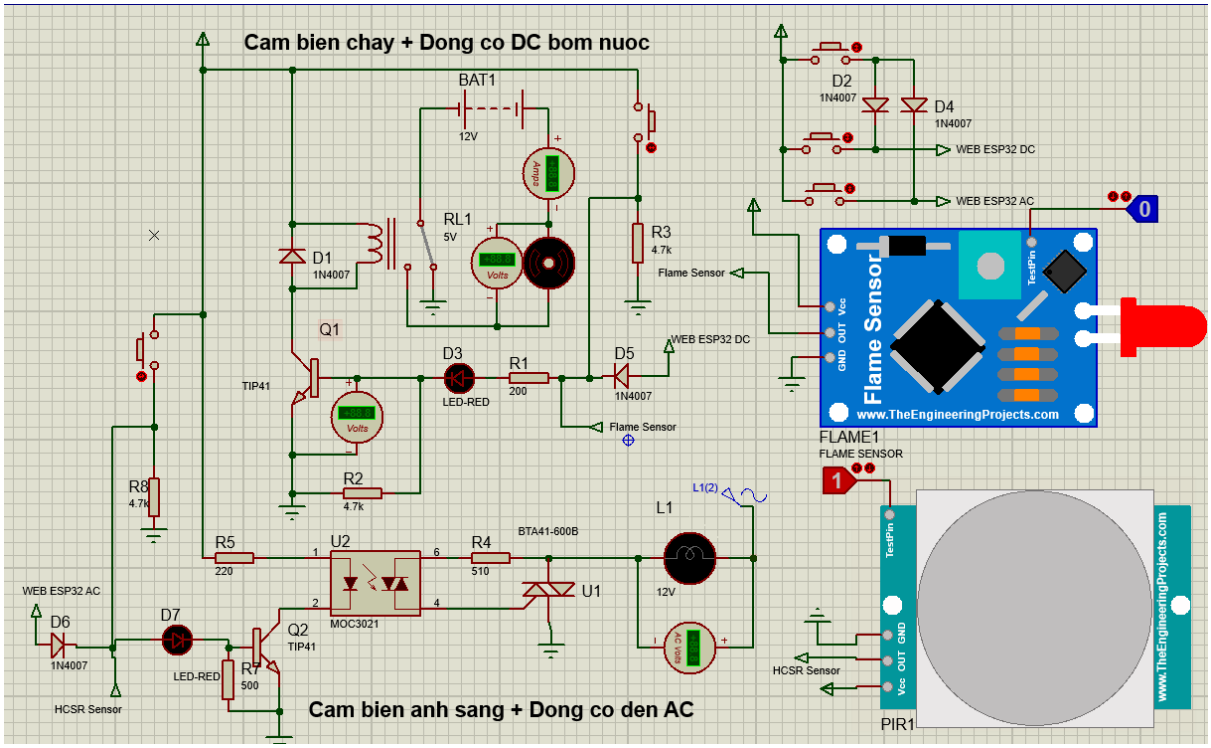


Hình 2.3. Sơ đồ mạch nguồn 5V-12V

Nguyên lý hoạt động: Máy biến áp TR1 giảm áp từ nguồn xoay chiều 220V xuống mức thấp hơn (khoảng 25V AC). Sau đó, cầu chỉnh lưu BR1 chuyển đổi điện áp xoay chiều thành điện áp một chiều dạng sóng gợn. Các tụ lọc C1, C2, C3, C4 giúp làm ổn định điện áp. IC 7812 và 7805 đảm bảo đầu ra ổn định lần lượt ở 12V và 5V, các tụ gốm C5, C6, C7, C8 để ổn định điện áp, giảm nhiễu. Hai transistor công suất Q1, Q3 cùng điện trở cân bằng dòng R1, R2 giúp khuếch đại dòng điện, đảm bảo đáp ứng tải lớn và giảm nhiệt cho IC.

2.3.2 Khối động cơ DC/AC

Mạch Động cơ DC(Cảm biến cháy + Động cơ bơm nước)



Hình 2.4. Mạch Relay DC (Cảm biến cháy + Động cơ bơm nước DC)

- Khi có sự xuất hiện của ngọn lửa, cảm biến lửa (Flame Sensor) sẽ kích hoạt transistor Q1.
- Điều này sẽ đóng relay RL1, cung cấp nguồn điện 12V cho động cơ DC bơm nước.
- Mạch này sẽ điều khiển hoạt động của bơm nước DC để dập tắt ngọn lửa.

Mạch Triac AC (Cảm biến ánh sáng + Động cơ đèn AC):

- Khi cảm biến ánh sáng (HCSR Sensor) phát hiện sự thay đổi cường độ ánh sáng,
- Tín hiệu sẽ được xử lý và kích hoạt Triac U1.
- Triac U1 sẽ điều khiển hoạt động của đèn AC, bật/tắt đèn dựa trên cường độ ánh sáng.

2.3.3 Cảm biến ánh sáng LM393

Cảm biến ánh sáng phát hiện mức độ sáng hoặc tối của môi trường xung quanh. Sau đó sẽ gửi tín hiệu đến vi điều khiển. Vi điều khiển sẽ xử lý là gửi thông báo đến cho bóng đèn AC. Nếu trời tối đèn sẽ được bật sáng, còn nếu trời sáng thì đèn sẽ được tắt.

2.3.4. Cảm biến cháy(Flame Sensor KY-026)

Cảm biến cháy có khả năng nhận diện ánh sáng phát ra từ ngọn lửa để phát hiện lửa từ xa. Khi phát hiện dấu hiệu cháy, cảm biến sẽ gửi tín hiệu cảnh báo đến hệ thống điều khiển, kích hoạt còi báo động nhằm thông báo nguy cơ cháy. Đồng thời sẽ kết nối với máy bơm nước để bơm nước dập tắt lửa.

Phạm vi đo: khoảng 80cm với góc quét 60°

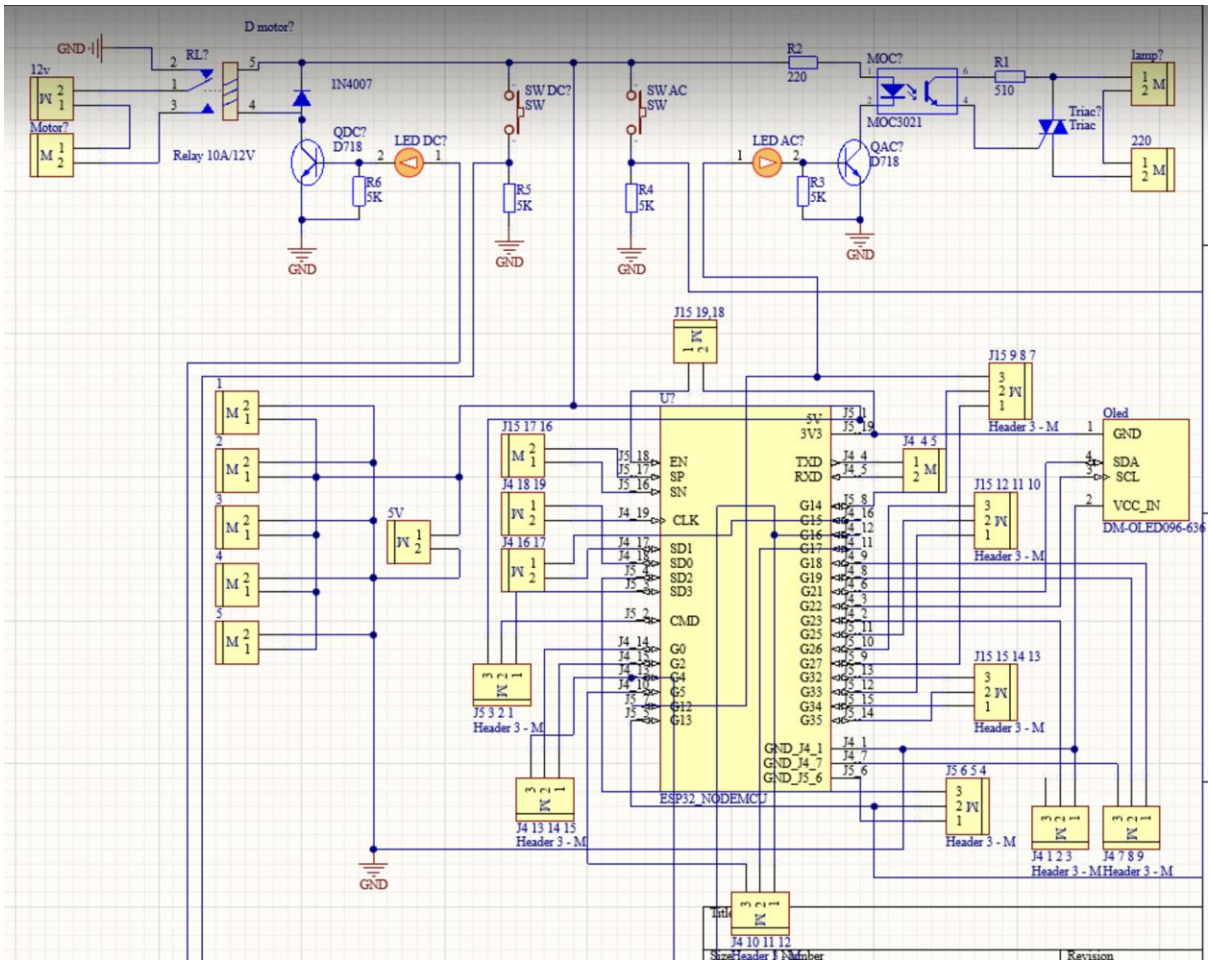
2.3.5. Cảm biến vật cản hồng ngoại

Cảm biến vật cản hồng ngoại phát hiện và đo khoảng cách đến vật cản bằng cách sử dụng tia hồng ngoại. Khi tia hồng ngoại phát ra từ cảm biến gặp vật cản, một phần tia sẽ phản xạ lại và được cảm biến thu nhận. Tín hiệu này được xử lý để xác định có vật cản trong phạm vi hoạt động và gửi tín hiệu đến khối điều khiển. Khi phát hiện có vật tức là có xe, tín hiệu sẽ gửi đến vi điều khiển và hiển thị có xe lên màn hình Oled và Web.

Phạm vi hoạt động: 2 ~ 5cm

2.4. Khối điều khiển và động cơ AC, động cơ DC

2.4.1 Sơ đồ vi điều khiển, động cơ DC, động cơ AC



Hình 2.5. Sơ đồ vi điều khiển, động cơ AC, động cơ DC

2.4.2. Vi điều khiển ESP32

Vi điều khiển ESP32 sẽ nhận tín hiệu từ các cảm biến, xử lý dữ liệu và sau đó gửi thông tin đến Broker sau đó hiển thị thông tin lên Web và màn hình LCD, đồng thời điều khiển bơm nước và đèn AC hoạt động.

2.4.3 Động cơ DC(Máy bơm nước)

Máy bơm nước sẽ hoạt động dựa trên tín hiệu điều khiển từ vi điều khiển ESP32. Khi ESP32 nhận được tín hiệu báo cháy, nó sẽ kích hoạt động cơ bơm nước để phun nước. Ngược lại, nếu ESP32 không nhận được tín hiệu báo cháy, động cơ bơm nước sẽ không hoạt động.

2.4.4. Động cơ AC(Đèn AC)

Đèn sẽ hoạt động dựa trên tín hiệu điều khiển từ vi điều khiển ESP32. Khi ESP32 nhận được tín hiệu báo trời tối, nó sẽ kích hoạt đèn để bật sáng. Ngược lại, nếu ESP32 không nhận được tín hiệu báo trời tối, đèn sẽ tắt.

2.5. Mạng truyền thông

Kết nối Wi-Fi:

- Truyền tải dữ liệu từ bộ vi điều khiển đến máy chủ hoặc nền tảng đám mây.
- Giao thức truyền thông: MQTT

2.6. Trung tâm xử lý

Máy chủ hoặc nền tảng đám mây:

- Lưu trữ dữ liệu thu thập được từ hệ thống cảm biến.
- Phân tích dữ liệu để phát hiện các vấn đề như có xe, không có xe, trời tối, trời sáng, có cháy
- Phần mềm xử lý dữ liệu: MQTTX

2.7. Giao diện người dùng

2.7.1 Ứng dụng web

Vi điều khiển nhận được tín hiệu từ các cảm biến sẽ gửi thông tin đến và được hiển thị trên Web

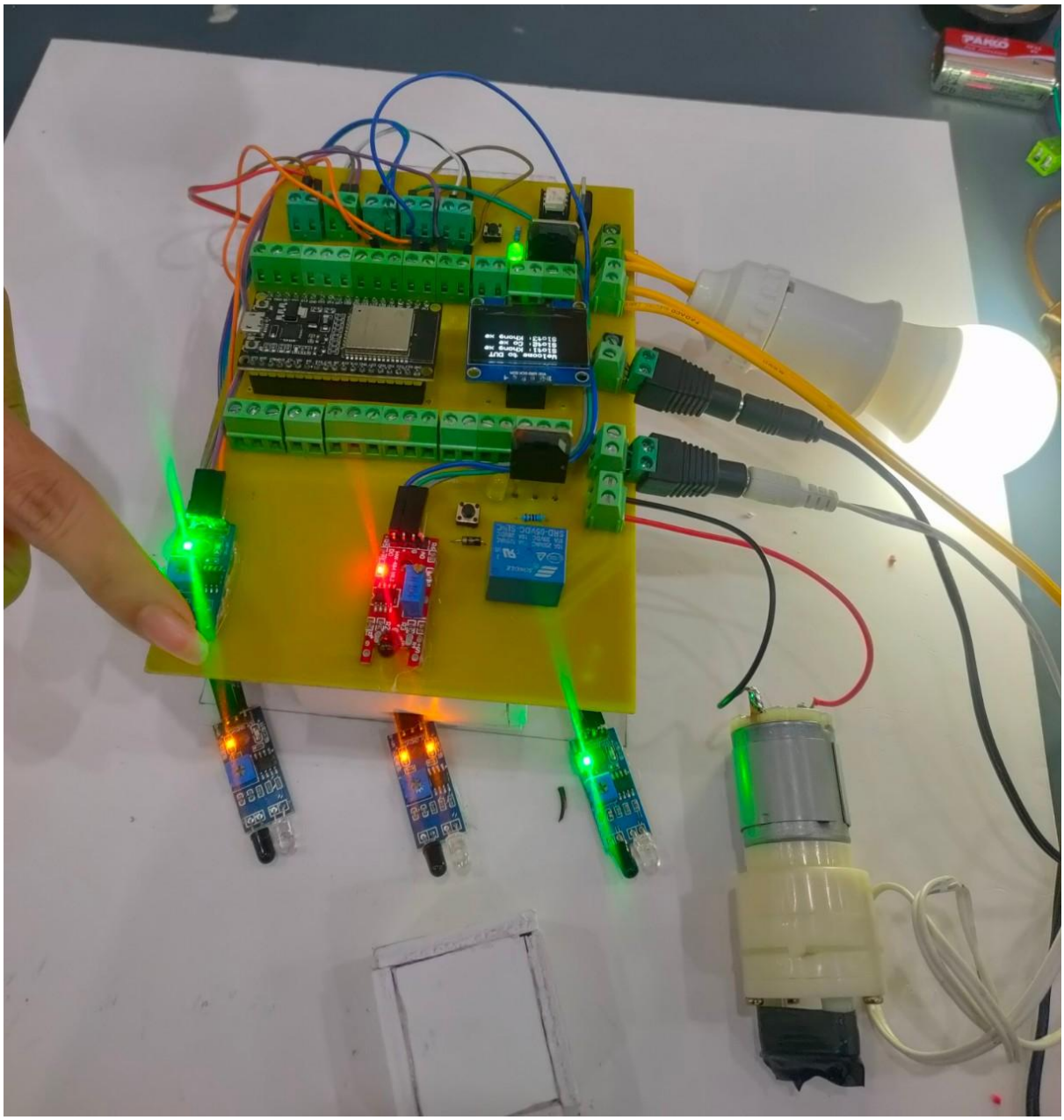
2.7.2. Màn hình Oled

Vi điều khiển nhận được tín hiệu từ các cảm biến sẽ gửi thông tin đến và được hiển thị trên màn hình OLED

Chương 3 KẾT QUẢ VÀ ĐÁNH GIÁ

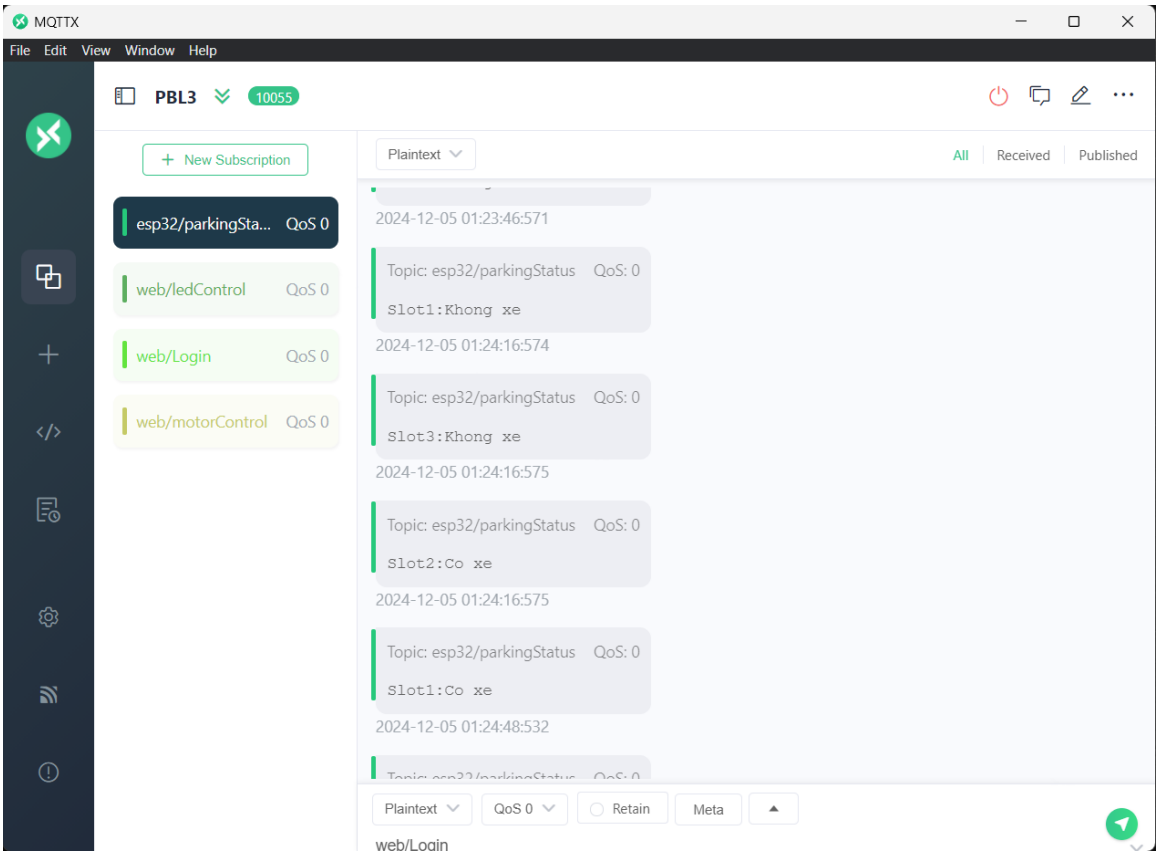
3.1. Kết quả

3.1.1 Phần cứng



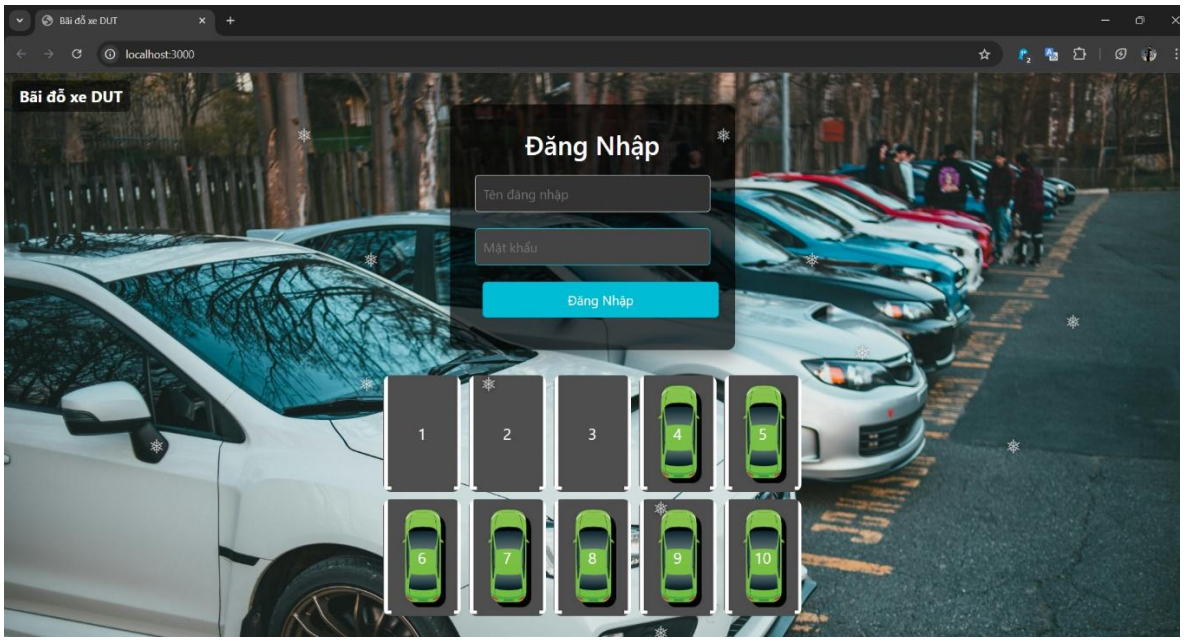
Hình 3.1. Sơ đồ phần cứng hệ thống

3.1.2 Lưu trữ dữ liệu

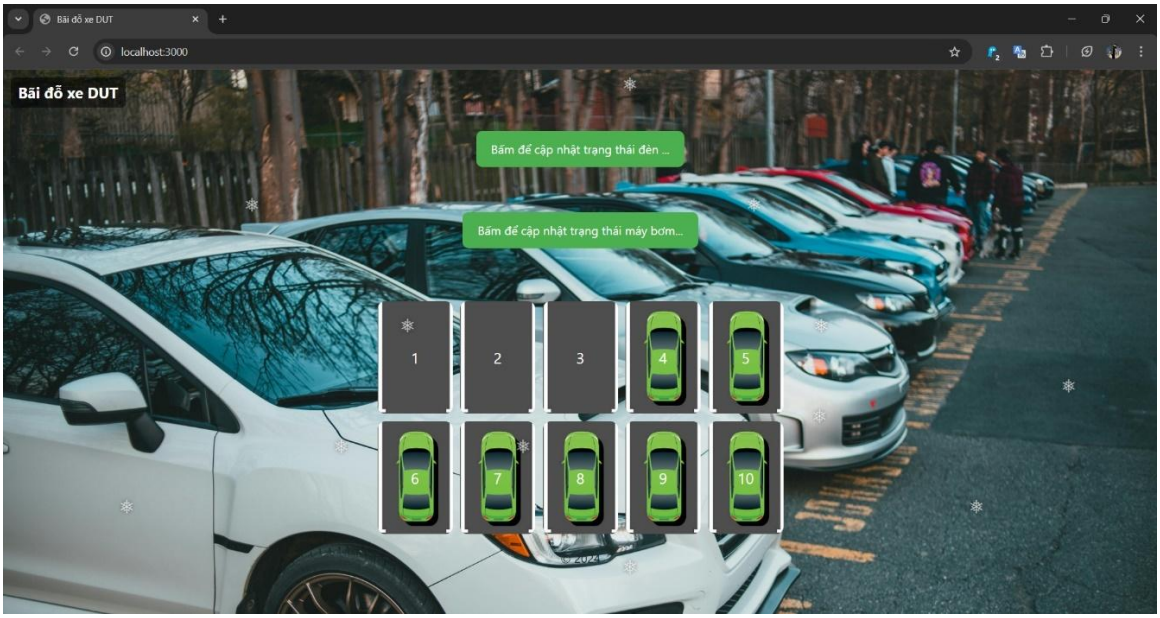


Hình 3.2. Dữ liệu lưu trữ trong MQTTX

3.1.3. Giao diện Web



Hình 3.3. Giao diện đăng nhập vào hệ thống



Hình 3.4. Giao diện điều khiển hệ thống

3.2. Đánh giá

3.2.1 Kết quả hiển thị trên Web

Ưu điểm:

- Người dùng dễ dàng quan sát vị trí trống của nhà xe, tình trạng nhà xe
- Hiển thị dữ liệu ngay lập tức, theo dõi trạng thái nhà xe liên tục
- Tiết kiệm chi phí phát triển: Sử dụng công nghệ web giúp giảm chi phí phát triển ứng dụng riêng biệt cho từng nền tảng, vì một giao diện web có thể hoạt động trên nhiều loại thiết bị.
- Tính thân thiện với người dùng: Giao diện web có thể được thiết kế trực quan, dễ sử dụng, giúp người dùng thao tác nhanh chóng và hiệu quả.
- Quản lý và giám sát từ xa: Chủ nhà xe hoặc người quản lý có thể theo dõi, điều hành và kiểm soát hệ thống từ xa thông qua trình duyệt, mà không cần có mặt tại chỗ.
- Khả năng mở rộng: Hệ thống hiển thị web dễ dàng mở rộng chức năng hoặc thêm tính năng mới mà không cần thay đổi kiến trúc cơ bản.

Nhược điểm:

- Phụ thuộc vào kết nối mạng: Nếu kết nối Internet không ổn định hoặc bị gián đoạn, hệ thống hiển thị web có thể không hoạt động hiệu quả hoặc bị chậm.
- Tốc độ tải dữ liệu: Thời gian tải trang web có thể ảnh hưởng đến trải nghiệm người dùng, đặc biệt khi hệ thống cần xử lý lượng lớn thông tin hoặc hình ảnh.

- Khả năng tương thích: Các trình duyệt hoặc thiết bị không tương thích với công nghệ web được sử dụng có thể làm giảm tính ổn định và hiệu quả của hệ thống.
- Chi phí bảo trì: Hệ thống hiển thị web yêu cầu bảo trì thường xuyên, bao gồm cập nhật phần mềm, vá lỗi và đảm bảo hiệu suất ổn định.
- Phụ thuộc vào nguồn điện: Hệ thống hiển thị thường yêu cầu nguồn điện liên tục. Nếu mất điện, hiển thị web sẽ không hoạt động, gây gián đoạn cho người dùng.
- Hiệu suất trên thiết bị cũ: Các thiết bị phần cứng cũ hoặc có cấu hình thấp có thể không đủ mạnh để xử lý các ứng dụng web hiện đại, dẫn đến hiệu suất kém.

3.2.2. Đánh giá tổng quan hệ thống

Ưu điểm:

- Hệ thống đã hoàn thiện được việc thu nhập, gửi dữ liệu, hiển thị. Hệ thống vận hành ổn định
- Hệ thống sử dụng công nghệ IoT để thu thập dữ liệu từ nhiều cảm biến khác nhau(cảm biến ánh sáng, cảm biến cháy, cảm biến vật cản hồng ngoại)
- Hệ thống có Web hiển thị giúp người dùng dễ dàng quan sát và quản lý hiệu quả.
- Hệ thống có bảo mật trang Web giúp bảo vệ dữ liệu.

Nhược điểm:

- Hiệu suất trong thực tế: Báo cáo chưa thử nghiệm hệ thống trong các điều kiện như môi trường khắc nghiệt hoặc khi số lượng cảm biến tăng lên đáng kể.

KẾT LUẬN

Dự án "Hệ thống quản lý nhà xe thông minh" đã được thiết kế và triển khai thành công, đạt được toàn bộ các mục tiêu đặt ra từ giai đoạn ý tưởng đến thực tế triển khai. Hệ thống này được phát triển với mục tiêu cải thiện hiệu quả quản lý và giám sát tình trạng hoạt động của nhà xe, đáp ứng nhu cầu hiện đại hóa trong bối cảnh đô thị hóa và ứng dụng công nghệ cao.

Một trong những tính năng nổi bật của hệ thống là khả năng thu thập dữ liệu từ nhiều loại cảm biến môi trường được bố trí trong khu vực nhà xe. Những cảm biến này có thể đo đạc và cung cấp các thông tin quan trọng như số lượng vị trí đỗ xe còn trống, trạng thái cháy hoặc sự cố xảy ra, và tình trạng hoạt động của hệ thống chiếu sáng. Tất cả dữ liệu này được hệ thống xử lý và lưu trữ để sử dụng lâu dài hoặc phân tích, nhằm đưa ra các cải tiến phù hợp.

Đặc biệt, hệ thống cung cấp giao diện trực quan thông qua nền tảng web, giúp người dùng dễ dàng theo dõi tình trạng nhà xe ở bất cứ đâu và bất cứ lúc nào. Tính năng này không chỉ hỗ trợ người quản lý trong việc giám sát và vận hành mà còn mang lại tiện ích lớn cho người sử dụng nhà xe.

Với việc áp dụng công nghệ IoT, dự án này đã minh chứng cho khả năng ứng dụng hiệu quả của các giải pháp thông minh trong việc tối ưu hóa quản lý không gian công cộng. Việc này góp phần nâng cao chất lượng dịch vụ và sự hài lòng của người sử dụng.

Về mặt kết quả, hệ thống đã đạt được các thành tựu đáng chú ý như:

- Thu thập và lưu trữ dữ liệu: Dữ liệu thu thập được từ các cảm biến đều được gửi đến vi điều khiển. Vi điều
- Hiển thị dữ liệu trực quan: Giao diện web hiển thị dữ liệu giúp người dùng dễ dàng quan sát tình trạng nhà xe
- Cảnh báo kịp thời: Hệ thống đưa ra thông báo như có cháy giúp người dùng phát hiện kịp thời và xử lý.
- Khả năng tích hợp: Sử dụng các nền tảng IoT như ESP32, ESP8266 kết hợp với cơ sở dữ liệu MySQL và giao thức truyền thông HTTP, hệ thống dễ dàng mở rộng để tích hợp thêm nhiều cảm biến hoặc ứng dụng vào các khu vực khác nhau.

Bên cạnh đó, hệ thống còn có những hạn chế như:

- Hiệu suất: Hệ thống chưa được kiểm thử trong điều kiện có khối lượng dữ liệu lớn hoặc môi trường khắc nghiệt để đánh giá khả năng vận hành ổn định trong thời gian dài.
- Tối ưu giao diện: Giao diện web cần bổ sung thêm các tính năng nâng cao như lọc dữ liệu, thông báo push, hoặc tùy chỉnh hiển thị để nâng cao trải nghiệm người dùng.

Hệ thống quản lý nhà xe thông minh không chỉ mang lại giá trị nghiên cứu mà còn mở ra những cơ hội ứng dụng rộng rãi trong nhiều lĩnh vực thực tiễn. Đây là một công cụ quản lý hiệu quả, thích hợp cho nhiều khu vực như khu dân cư, trường học, khu công nghiệp, trung tâm thương mại và các khu đô thị đông đúc. Nhờ vào sự hỗ trợ của công nghệ tiên tiến, hệ thống này đáp ứng tốt nhu cầu quản lý nhà xe ngày càng gia tăng trong bối cảnh đô thị hóa và phát triển kinh tế.

Hệ thống không chỉ dừng lại ở việc giám sát và quản lý vị trí đỗ xe; nó còn tối ưu hóa không gian đỗ xe, giúp giảm thiểu thời gian tìm kiếm chỗ trống cho người dùng. Điều này không chỉ nâng cao trải nghiệm mà còn gia tăng sự hài lòng cho người sử dụng. Đặc biệt, hệ thống tích hợp các chức năng như cảnh báo cháy nổ, điều khiển ánh sáng tự động, và cung cấp thông tin thời gian thực thông qua các giao diện thân thiện trên nền tảng web hoặc ứng dụng di động. Nhờ đó, không gian đỗ xe trở nên an toàn và hiện đại hơn, đồng thời giảm bớt khối lượng công việc cho người quản lý.

Hơn nữa, việc triển khai hệ thống quản lý nhà xe thông minh còn mang ý nghĩa xã hội lớn lao. Nó không chỉ nâng cao nhận thức của cộng đồng về việc đỗ xe đúng nơi quy định, tránh lãng phí tài nguyên và không gian công cộng, mà còn giúp xây dựng ý thức văn minh và lịch sự trong việc sử dụng cơ sở hạ tầng chung. Điều này đặc biệt quan trọng trong bối cảnh các đô thị lớn thường xuyên đối mặt với tình trạng ùn tắc giao thông và thiếu hụt không gian đỗ xe.

Trong tương lai, nhóm nghiên cứu có thể tập trung vào việc :

- Phát triển ứng dụng cho phép tìm kiếm, đặt chỗ và thanh toán dễ dàng.
- Hỗ trợ thanh toán không tiếp xúc qua ví điện tử và QR code.
- Dự đoán nhu cầu đỗ xe bằng trí tuệ nhân tạo
- Mở rộng các dịch vụ như bảo trì xe, sạc xe điện, hoặc dịch vụ rửa xe để tăng giá trị cho khách hàng.

TÀI LIỆU THAM KHẢO

[1] https://iotdesignpro.com/projects/iot-based-smart-parking-using-esp8266

[2] https://www.electronicssimplified.in/category/esp32/

[3] https://www.circuitschools.com/control-servo-motor-using-rotary-encoder-with-arduino-and-lcd/

[4] https://www.electroniclinic.com/esp32-projects/#google_vignette

[5] https://dientunguyenhien.vn/

[6] https://www.youtube.com/watch?v=DIVAkjSbEoQ&t=2s

PHỤ LỤC

Timeline làm việc nhóm

Thời gian	Nội dung công việc
18/8	Tìm hiểu, lựa chọn đề tài
20/8	Thống nhất đề tài: Hệ thống quản lý nhà xe thông minh
29/8	Báo cáo tiến độ lần 1
16/9	-Kết nối Wifi với ESP32 -Chạy cảm biến cân nặng hiển thị trên LCD -Nguyên cứu thay thế cảm biến cân nặng Loadcell thành cảm biến lực BF350 để tiện cho việc mô phỏng và thu nhỏ hệ thống. -Thống nhất làm mô hình 3 xe. -Nguyên cứu mạch động cơ AC
28/9	-Hiếu: Giao tiếp được MQTTX đến Web -Thành: Mô phỏng động cơ DC và các cảm biến -Giang: Tìm hiểu cách làm giao diện Web bằng HTML
17/10	Báo cáo tiến độ lần 2
24/10	Nhóm thay đổi định hướng hệ thống, bỏ cảm biến cân, Thiết kế lại mạch
7/11	-Nhóm thiết kế xong phần cứng
20/11	-Kết nối các cảm biến cháy, ánh sáng vào mô hình
27/11	-Hoàn thành các yêu cầu như báo cáo, sản phẩm

Code ESP32

```
#include <Arduino.h>
#include <WiFi.h> // Sử dụng WiFiClient cho kết nối không bảo mật
#include <PubSubClient.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SH110X.h>


// Thông tin Wi-Fi và MQTT broker
#define ssid "Hieu_TkZ"
#define password "123123123"

// Thông tin MQTT broker
const char *mqtt_server = "broker.emqx.io";
#define mqtt_port 1883
#define mqtt_username "hieutk1302"
#define mqtt_password "hieutk1302"


WiFiClient espClient; // Sử dụng WiFiClient cho kết nối không bảo mật
PubSubClient client(espClient);


bool isLoginProcessed = false; // Biến trạng thái đã xử lý login


// Định nghĩa kích thước màn hình OLED
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
// Khởi tạo đối tượng OLED
Adafruit_SH1106G display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);


// Định nghĩa chân
#define DC 16        // động cơ
#define BUTTON_DC 4  // nút động cơ
#define AC 12        // đèn
#define BUTTON_AC 13 // nút đèn


// cảm biến
#define Light 14 // cảm biến ánh sáng
#define Flame 26 // cảm biến lửa


// Biến lưu trạng thái cảm biến
#define cambienxe1 32 // Chân cảm biến xe 1
#define cambienxe2 33 // Chân cảm biến xe 2
#define cambienxe3 25 // Chân cảm biến xe 3
```

```
int xe1 = 0;
int xe2 = 0;
int xe3 = 0;



bool ACState = false;          // Trạng thái đèn/motor
bool lastButtonACState = HIGH; // Trạng thái nút trước đó
bool overrideACSensor = false; // Biến để xác định có bỏ qua cảm biến
hay không
int lastLightState = HIGH;     // Lưu trạng thái cảm biến trước đó


bool DCState = false;          // Trạng thái đèn/motor
bool lastButtonDCState = HIGH; // Trạng thái nút trước đó
bool overrideDCSensor = false; // Biến để xác định có bỏ qua cảm biến
hay không
int lastFlameState = HIGH;     // Lưu trạng thái cảm biến trước đó

// Khai báo hàm
void connectWiFi();
void reconnect();
void callback(char *topic, byte *payload, unsigned int length);
void oled();
void parking();
void DongcoAC();
void DongcoDC();

void setup()
{
  Serial.begin(115200);
  connectWiFi(); // Kết nối Wi-Fi
  client.setServer(mqtt_server, mqtt_port);
  Serial.println("setServer completed !");
  client.setCallback(callback); // Thiết lập callback để nhận tin từ
MQTT




  // Khởi tạo các chân cảm biến và động cơ
  pinMode(cambienxe1, INPUT);
  pinMode(cambienxe2, INPUT);
  pinMode(cambienxe3, INPUT);


  pinMode(DC, OUTPUT);
  pinMode(BUTTON_DC, INPUT_PULLUP);
  digitalWrite(DC, digitalRead(Flame)); // motor bật hay tắt lúc đầu là
do cảm biến


  pinMode(AC, OUTPUT);
```

```
  pinMode(BUTTON_AC, INPUT_PULLUP);
  digitalWrite(AC, digitalRead(Light)); // đèn bật hay tắt lúc đầu là
do cảm biến


  pinMode(Flame, INPUT_PULLUP);
  pinMode(Light, INPUT_PULLUP);


  // Khởi động màn hình OLED
  Wire.begin(21, 22);
  display.begin(0x3C, true);          // Khởi tạo với địa chỉ I2C 0x3C
  display.clearDisplay();             // Xóa bộ đệm hiển thị
  display.setTextSize(0.7);           // Đặt kích thước chữ
  display.setTextColor(SH110X_WHITE); // Đặt màu chữ


  // Hiển thị thông điệp chào mừng trên OLED
  display.setCursor(0, 0);
  display.println("Hello, PBL3");
  display.setCursor(0, 10);
  display.println("ESP32 & OLED");
  display.display(); // Cập nhật hiển thị
  delay(2000);       // Đợi 2 giây trước khi tiếp tục
}


void loop() {
  if (!client.connected()) {
    Serial.println("Connecting to broker...");
    reconnect();
  }
  client.loop(); // Giữ kết nối với broker


  // Gửi trạng thái bãi đỗ xe liên tục mỗi 2 giây
  static unsigned long lastTime = 0;
  unsigned long currentTime = millis();

  if (currentTime - lastTime >= 2000) {  // Mỗi 2 giây gửi một lần
    parking();
    client.publish("esp32/motorState", DCState ? "On" : "Off");
    client.publish("esp32/ledState", ACState ? "Off" : "On");
    lastTime = currentTime;  // Cập nhật thời gian gửi tiếp theo
  }


  DongcoDC();
  DongcoAC();
  oled();
}



void callback(char *topic, byte *payload, unsigned int length) {
    String msg = "";
```

```
    for (int i = 0; i < length; i++) {
        msg += (char)payload[i]; // Chuyển payload thành chuỗi
    }

    Serial.print("Tin nhắn nhận từ topic ");
    Serial.print(topic);
    Serial.print(": ");
    Serial.println(msg);

    // Xử lý từng topic
    if (strcmp(topic, "web/ledControl") == 0) {
        if (msg == "Turn on") {
            ACState = true;
            digitalWrite(AC, HIGH); // Bật đèn
            overrideACSensor = true;
        } else if (msg == "Turn off") {
            ACState = false;
            digitalWrite(AC, LOW); // Tắt đèn
            overrideACSensor = true;
        }
        client.publish("esp32/ledState", ACState ? "On" : "Off");

    } else if (strcmp(topic, "web/motorControl") == 0) {
        if (msg == "Turn on") {
            DCState = true;
            digitalWrite(DC, HIGH); // Bật máy bơm
            overrideDCSensor = true;
        } else if (msg == "Turn off") {
            DCState = false;
            digitalWrite(DC, LOW); // Tắt máy bơm
            overrideDCSensor = true;
        }
        client.publish("esp32/motorState", DCState ? "On" : "Off");

      }else if (strcmp(topic, "web/Logout") == 0) {
      if (msg == "True") {
        isLoginProcessed = false;
        Serial.println("Logout successful, isLoginProcessed reset.");
    }


    } else if (strcmp(topic, "web/Login") == 0) {
        if (!isLoginProcessed) {
            if (msg == "admin|admin") {
                client.publish("web/Login", "true");  // Trả kết quả
đúng
            } else {
                client.publish("web/Login", "false"); // Trả kết quả
sai
            }
            isLoginProcessed = true; // Đánh dấu đã xử lý
        }

    }
}
```

```
// Hàm kết nối với mqtt broker
void reconnect()
{
  while (!client.connected())
  {
    Serial.print("Attempting MQTT connection...");
    String clientID = "ESPClient-";
    clientID += String(random(0xffff), HEX);


    // Kết nối với broker MQTT với username và password
    if (client.connect(clientID.c_str(), mqtt_username, mqtt_password))
    {
      Serial.println("connected MQTT");

      client.subscribe("esp32/ledState");
      Serial.println("connected topic esp32/ledState");

      client.subscribe("esp32/motorState");
      Serial.println("connected topic esp32/motorState");

      client.subscribe("web/ledControl");
      Serial.println("connected topic web/ledControl");

      client.subscribe("esp32/parkingStatus");
      Serial.println("connected topic esp32/parkingStatus"); // đăng ký
topic để gửi trạng thái ô đậu xe

      client.subscribe("web/motorControl");
      Serial.println("connected topic web/motorControl");

      client.subscribe("web/Login");
      client.subscribe("web/Logout");
      Serial.println("connected topic web/Login");
    }
    else
    {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      delay(2000);
    }
  }
}



// Hàm kết nối Wi-Fi
void connectWiFi()
{
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi..");
  while (WiFi.status() != WL_CONNECTED)
  {
```

```
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConnected to WiFi !");
}
// Hàm Oled
void oled(){
// Đọc trạng thái cảm biến
xe1 = digitalRead(cambienxe1);
xe2 = digitalRead(cambienxe2);
xe3 = digitalRead(cambienxe3);


// Hiển thị thông điệp chào mừng
display.clearDisplay();
display.setCursor(30, 0);
display.println("Welcome to DUT");



// Mảng chứa trạng thái và vị trí
int xe[] = {xe1, xe2, xe3};
const char* slots[] = {"Slot1", "Slot2", "Slot3"};
const char* statuses[] = {"Co xe", "Khong xe"};

for (int i = 0; i < 3; i++) {
    display.setCursor(30, 10 + (i * 10));
    display.print(slots[i]);
    display.print(": ");
    display.println(statuses[xe[i]]);
}


// Kiểm tra nếu tất cả vị trí đều có xe
if (xe1 == LOW && xe2 == LOW && xe3 == LOW) {
    display.clearDisplay();
    display.setCursor(30, 20);
    display.println("Parking Full");
    display.setCursor(30, 40);
    display.println("See you again");
}


// Cập nhật màn hình OLED
display.display();
// Thêm một khoảng thời gian để giảm tần suất đọc
delay(100);
}


void parking() {
  int giaTri1 = digitalRead(cambienxe1);
  int giaTri2 = digitalRead(cambienxe2);
  int giaTri3 = digitalRead(cambienxe3);
```

```
    client.publish("esp32/parkingStatus", giaTri1 == HIGH ?
"Slot1:Khong xe" : "Slot1:Co xe");
    client.publish("esp32/parkingStatus", giaTri2 == HIGH ?
"Slot2:Khong xe" : "Slot2:Co xe");
    client.publish("esp32/parkingStatus", giaTri3 == HIGH ?
"Slot3:Khong xe" : "Slot3:Co xe");
}


void DongcoAC()
{
  // Đọc trạng thái nút nhấn đèn
  bool currentButtonACState = digitalRead(BUTTON_AC);
  // Đọc trạng thái cảm biến ánh sáng
  int currentLightState = digitalRead(Light);




  // Khi nút nhấn được nhấn (nhấn vào = LOW)
  if (lastButtonACState == HIGH && currentButtonACState == LOW)
  {
    overrideACSensor = true; // Ưu tiên nút nhấn, bỏ qua cảm biến
    ACState = !ACState;      // Đảo trạng thái đèn/motor
    digitalWrite(AC, ACState ? LOW : HIGH);
  }
  lastButtonACState = currentButtonACState;


  // Kiểm tra nếu cảm biến thay đổi trạng thái
  if (currentLightState != lastLightState)
  {
    overrideACSensor = false;           // Khôi phục quyền điều khiển
của cảm biến
    lastLightState = currentLightState; // Cập nhật trạng thái cảm biến
  }
  // Nếu không bị nút nhấn ưu tiên, dùng tín hiệu từ cảm biến
  if (!overrideACSensor)
  {
    if (currentLightState == HIGH)
    {
      digitalWrite(AC, HIGH); // Cảm biến light tắt đèn
      ACState = false;       // Cập nhật trạng thái đèn
    }
    else
    {
      digitalWrite(AC, LOW); // Cảm biến Flame bật đèn
      ACState = true;         // Cập nhật trạng thái đèn
    }
    delay(10);
  }
}
```

```
void DongcoDC()
{
  // Đọc trạng thái nút nhấn
  bool currentButtonDCState = digitalRead(BUTTON_DC);
  // Đọc trạng thái cảm biến Flame
  int currentFlameState = digitalRead(Flame);




  // Khi nút nhấn được nhấn (nhấn vào = LOW)
  if (lastButtonDCState == HIGH && currentButtonDCState == LOW)
  {
    overrideDCSensor = true; // Ưu tiên nút nhấn, bỏ qua cảm biến
    DCState = !DCState;      // Đảo trạng thái đèn/motor
    digitalWrite(DC, DCState ? HIGH : LOW);
  }
  lastButtonDCState = currentButtonDCState;


  // Kiểm tra nếu cảm biến thay đổi trạng thái
  if (currentFlameState != lastFlameState)
  {
    overrideDCSensor = false;           // Khôi phục quyền điều khiển
của cảm biến
    lastFlameState = currentFlameState; // Cập nhật trạng thái cảm biến
  }


  // Nếu không bị nút nhấn ưu tiên, dùng tín hiệu từ cảm biến
  if (!overrideDCSensor)
  {
    if (currentFlameState == LOW)
    {
      digitalWrite(DC, LOW); // Cảm biến motor tắt
      DCState = false;       // Cập nhật trạng thái motor
    }
    else
    {
      digitalWrite(DC, HIGH); // Cảm biến motor bật
      DCState = true;         // Cập nhật trạng thái motor
    }
    delay(10);
  }
}
```

Code web

```
<!DOCTYPE html>
<html lang="vi">
<head> <!-- Dinh dang web-->
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Bãi đỗ xe DUT</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://unpkg.com/mqtt/dist/mqtt.min.js"></script>
    <style>
        /* Định dạng dòng chữ tiêu đề */
        .title {
            position: absolute; /* Cố định vị trí */
            top: 10px; /* Cách mép trên 10px */
            left: 10px; /* Cách mép trái 10px */
            font-size: 20px; /* Kích thước chữ */
            font-weight: bold; /* Chữ đậm */
            color: #fff; /* Màu chữ trắng */
            background-color: rgba(0, 0, 0, 0.5); /* Nền đen mờ */
            padding: 5px 10px; /* Khoảng cách bên trong */
            border-radius: 5px; /* Góc bo tròn */
        }
        /* Đặt container chiếm toàn bộ màn hình */
        body, html {
            height: 100%;
            margin: 0;
            display: flex;
            flex-direction: column; /* Sắp xếp các phần tử theo cột */
            align-items: center; /* Căn giữa theo chiều ngang */
            justify-content: center; /* Căn giữa theo chiều dọc */
        }

        /* Nội dung từng phần tử */
        .content {
            width: 300px;
            background-color: #f3f3f3;
            border: 1px solid #ddd;
            padding: 20px;
            margin: 10px 0; /* Khoảng cách giữa các phần tử */
            text-align: center;
        }
        .slot {
            transition: transform 0.3s ease, background-color 0.3s
ease;
        }
        .slot:hover {
            transform: scale(1.1);
            border: 2px solid white;
        }
        .parking-lot {
            grid-template-columns: repeat(auto-fit, minmax(120px,
1fr));
            gap: 15px;
        }
```

```
     /* Phần ô đăng nhập */
.login-container {
    background-color: rgba(0, 0, 0, 0.7);
    padding: 30px;
    border-radius: 10px;
    box-shadow: 0px 10px 20px rgba(0, 0, 0, 0.3);
    width: 350px;
    color: white;
    text-align: center;
    transition: transform 0.3s ease-in-out;
}

.login-container:hover {
    transform: scale(1.05);
}

.login-container input {
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #ccc;
    border-radius: 5px;
    background-color: #333;
    color: white;
    font-size: 16px;
    transition: all 0.3s ease;
}

.login-container input:focus {
    border-color: #00bcd4;
    background-color: #444;
    outline: none;
}

.login-container button {
    padding: 10px 20px;
    border: none;
    background-color: #00bcd4;
    color: white;
    font-size: 16px;
    border-radius: 5px;
    cursor: pointer;
    width: 100%;
    transition: background-color 0.3s ease;
}

.login-container button:hover {
    background-color: #0288d1;
}

.status {
    font-size: 16px;
    font-weight: bold;
    text-align: center;
    padding: 10px;
    border-radius: 5px;
    transition: all 0.3s ease;
```

```
      }

      .status p {
        margin: 0;
      }

      .status.success {
        background-color: #28a745;
        color: white;
      }

      .status.failure {
        background-color: #dc3545;
        color: white;
      }

      .status.loading {
        background-color: #ffc107;
        color: black;
      }

      .status.success:hover {
        background-color: #218838;
      }

      .status.failure:hover {
        background-color: #c82333;
      }

      .status.loading:hover {
        background-color: #e0a800;
      }


    </style>
</head>
<style>
    body {
        background-image: url('background.jpg'); /* Đường dẫn đến hình
nền */
        background-size: cover; /* Hình nền phủ toàn màn hình */
        background-repeat: no-repeat; /* Không lặp lại hình nền */
        background-attachment: fixed; /* Hình nền cố định khi cuộn
trang */
    }
</style>

<!-- Hieu ung tuyet roi---->
<style>
    /* customizable snowflake styling */
    .snowflake {
      color: #fff;
      font-size: 1em;
      font-family: Arial, sans-serif;
      text-shadow: 0 0 5px #000;
    }
    .snowflake {
        opacity: 0.8;
```

```
        font-size: 1.5em;
    }


    .snowflake,.snowflake .inner{animation-iteration-
count:infinite;animation-play-state:running}@keyframes snowflakes-
fall{0%{transform:translateY(0)}100%{transform:translateY(110vh)}}@keyf
rames snowflakes-
shake{0%,100%{transform:translateX(0)}50%{transform:translateX(80px)}}.
snowflake{position:fixed;top:-10%;z-index:9999;-webkit-user-
select:none;user-select:none;cursor:default;animation-name:snowflakes-
shake;animation-duration:3s;animation-timing-function:ease-in-
out}.snowflake .inner{animation-duration:10s;animation-name:snowflakes-
fall;animation-timing-function:linear}.snowflake:nth-of-
type(0){left:1%;animation-delay:0s}.snowflake:nth-of-type(0)
.inner{animation-delay:0s}.snowflake:first-of-type{left:10%;animation-
delay:1s}.snowflake:first-of-type .inner,.snowflake:nth-of-type(8)
.inner{animation-delay:1s}.snowflake:nth-of-type(2){left:20%;animation-
delay:.5s}.snowflake:nth-of-type(2) .inner,.snowflake:nth-of-type(6)
.inner{animation-delay:6s}.snowflake:nth-of-type(3){left:30%;animation-
delay:2s}.snowflake:nth-of-type(11) .inner,.snowflake:nth-of-type(3)
.inner{animation-delay:4s}.snowflake:nth-of-type(4){left:40%;animation-
delay:2s}.snowflake:nth-of-type(10) .inner,.snowflake:nth-of-type(4)
.inner{animation-delay:2s}.snowflake:nth-of-type(5){left:50%;animation-
delay:3s}.snowflake:nth-of-type(5) .inner{animation-
delay:8s}.snowflake:nth-of-type(6){left:60%;animation-
delay:2s}.snowflake:nth-of-type(7){left:70%;animation-
delay:1s}.snowflake:nth-of-type(7) .inner{animation-
delay:2.5s}.snowflake:nth-of-type(8){left:80%;animation-
delay:0s}.snowflake:nth-of-type(9){left:90%;animation-
delay:1.5s}.snowflake:nth-of-type(9) .inner{animation-
delay:3s}.snowflake:nth-of-type(10){left:25%;animation-
delay:0s}.snowflake:nth-of-type(11){left:65%;animation-delay:2.5s}
    </style>
    <div class="snowflakes" aria-hidden="true">
      <div class="snowflake">
        <div class="inner">❅</div>
      </div>
      <div class="snowflake">
        <div class="inner">❅</div>
      </div>
      <div class="snowflake">
        <div class="inner">❅</div>
      </div>
      <div class="snowflake">
        <div class="inner">❅</div>
      </div>
      <div class="snowflake">
        <div class="inner">❅</div>
      </div>
      <div class="snowflake">
        <div class="inner">❅</div>
      </div>
      <div class="snowflake">
        <div class="inner">❅</div>
      </div>
      <div class="snowflake">
```

```
        <div class="inner">❄</div>
      </div>
      <div class="snowflake">
        <div class="inner">❄</div>
      </div>
      <div class="snowflake">
        <div class="inner">❄</div>
      </div>
      <div class="snowflake">
        <div class="inner">❄</div>
      </div>
      <div class="snowflake">
        <div class="inner">❄</div>
      </div>
    </div>
<!--------------------------->
<!---Hiệu ứng nút nhấn-->
<style>
    /* Phong cách chung cho các nút */
    button {
        background-color: #4CAF50; /* Màu nền */
        color: white; /* Màu chữ */
        padding: 12px 20px; /* Khoảng cách bên trong */
        margin: 10px; /* Khoảng cách giữa các nút */
        border: none; /* Không viền */
        border-radius: 8px; /* Bo tròn các góc */
        font-size: 16px; /* Kích thước chữ */
        cursor: pointer; /* Hiển thị con trỏ chuột */
        transition: transform 0.2s, background-color 0.2s; /* Hiệu ứng
*/


    }

    /* Hiệu ứng hover */
    button:hover {
        background-color: #45a049; /* Thay đổi màu nền khi hover */
        transform: scale(1.05); /* Phóng to nhẹ khi hover */
    }

    /* Hiệu ứng khi bấm */
    button:active {
        transform: scale(0.95); /* Thu nhỏ nhẹ khi bấm */
        background-color: #3e8e41; /* Đổi màu nền khi bấm */
    }

    /* Căn chỉnh hai khu vực chứa nút */
    #buttonContainer1, #buttonContainer2 {
        text-align: center; /* Căn giữa các nút */
        margin: 20px auto; /* Khoảng cách giữa các khu vực */
    }
</style>


<body>
    <div class="title">Bãi đỗ xe DUT</div>
```

```
    <script> //ket noi  MQTT broker

         const client = mqtt.connect('ws://broker.emqx.io:8083/mqtt', {
            username: 'hieutk1302',
            password: 'hieutk1302',
        });

        client.on('connect', () => {
            console.log("Connected to EMQX from web");


        });
    </script>


    <div id="buttonContainer1"></div> <!-- Thêm phần tử chứa nút -->
    <div id="buttonContainer2"></div> <!-- Thêm phần tử chứa nút -->
    <!------------------------------------ Ô đăng nhập --------------
----------------------------------->
        <div class="login-container">
            <h2>Đăng Nhập</h2>
            <input type="text" id="username" placeholder="Tên đăng
nhập">
            <input type="password" id="password" placeholder="Mật
khẩu">
            <button onclick="login()">Đăng Nhập</button>
        </div>
        <div id="loginStatus" class="status"> </div>

    </div>


    <script>
        window.onload = function () {
            client.publish("web/Logout", "True", { qos: 0, retain:
false }); // Reset trạng thái trên ESP32
            document.getElementById("loginStatus").textContent = ""; //
Xóa thông báo trạng thái
        };
        client.subscribe("web/Login", (err) => {
            if (err) {
                console.error("Lỗi khi đăng ký topic web/Login", err);
            }
            else {console.log("Đăng ký topic web/Login thành công ! ")}
        });
        let isAdmin = false; // Biến xác định quyền truy cập

       // Lắng nghe phản hồi từ ESP32
client.on('message', (topic, message) => {
    if (topic === "web/Login") {
        const loginStatus = message.toString();

        if (loginStatus === "true") {
            isAdmin = true; // Xác nhận đăng nhập thành công
            updateLoginStatus("success");
            setTimeout(() => {
```

```
                document.querySelector(".login-
container").style.display = "none"; // Ẩn ô đăng nhập
                nutdieukhien(); // Hiển thị nút điều khiển
                nutDieuKhienMayBom();
            }, 1000); // 1s
        } else if (loginStatus === "false") {
            isAdmin = false;
            updateLoginStatus("failure"); // Đăng nhập thất bại
        }
    }
});

// Đăng nhập
function login() {
    const username = document.getElementById('username').value;
    const password = document.getElementById('password').value;

    if (!username || !password) {
        document.getElementById("loginStatus").textContent = "Vui lòng
nhập tên đăng nhập và mật khẩu!";
        return;
    }

    const loginPayload = `${username}|${password}`;
    client.publish("web/Login", loginPayload, { qos: 0, retain: false
}, (err) => {
        if (err) {
            document.getElementById("loginStatus").textContent = "Lỗi
khi gửi thông tin đăng nhập!";
        } else {
            document.getElementById("loginStatus").textContent = "Đang
xác thực...";
        }
    });
}


    </script>
    <!--
==========================================================================
==========================-->



    <script> //Hien thi trang thai xe
        client.subscribe("esp32/parkingStatus", (err) =>{});
        client.on('message', (topic, message) => {
            if (topic === 'esp32/parkingStatus') {
                console.log('Message received:', message.toString());

                // Xử lý trạng thái cho Slot 1
                if (message.toString().includes('Slot1:Co xe')) {
                    document.getElementById('slot1').classList.add('occ
upied');
                    document.getElementById('slot1').classList.remove('
empty');
```

```
                } else if (message.toString().includes('Slot1:Khong
xe')) {
                    document.getElementById('slot1').classList.remove('
occupied');
                    document.getElementById('slot1').classList.add('emp
ty');
                }

                // Xử lý trạng thái cho Slot 2
                if (message.toString().includes('Slot2:Co xe')) {
                    document.getElementById('slot2').classList.add('occ
upied');
                    document.getElementById('slot2').classList.remove('
empty');
                } else if (message.toString().includes('Slot2:Khong
xe')) {
                    document.getElementById('slot2').classList.remove('
occupied');
                    document.getElementById('slot2').classList.add('emp
ty');
                }

                // Xử lý trạng thái cho Slot 3
                if (message.toString().includes('Slot3:Co xe')) {
                    document.getElementById('slot3').classList.add('occ
upied');
                    document.getElementById('slot3').classList.remove('
empty');
                } else if (message.toString().includes('Slot3:Khong
xe')) {
                    document.getElementById('slot3').classList.remove('
occupied');
                    document.getElementById('slot3').classList.add('emp
ty');
                }
            }
        });


    </script>
<!-- ============================== Ô để xe
============================== -->
    <style>
        .parking-lot {
            display: grid;
            grid-template-columns: repeat(5, 100px); /* 5 ô mỗi hàng */
            gap: 10px;
            margin: 20px;
        }

        .slot {
            width: 100px; /* Chiều rộng cố định */
            height: 150px; /* Chiều cao lớn hơn để tạo hình chữ nhật
đứng */
            display: flex;
            align-items: center;
            justify-content: center;
            font-size: 20px;
```

```
            color: white;
            border-radius: 10px;
            cursor: pointer;
        }


        .empty {
            background-image: url('empty.png'); /* Đường dẫn tới hình
ảnh ô trống */
            background-size: cover; /* Đảm bảo ảnh phủ kín ô */
            background-position: center; /* Canh giữa hình ảnh */
        }

        .occupied {
            background-image: url('occupied.png'); /* Đường dẫn tới
hình ảnh ô có xe */
            background-size: cover;
            background-position: center;
        }
    </style>
<!-- ========================================================== -->
<div class="parking-lot">
    <!-- Tạo các ô đậu xe -->
    <div class="slot empty" id="slot1">1</div>
    <div class="slot empty" id="slot2">2</div>
    <div class="slot empty" id="slot3">3</div>

    <div class="slot occupied" id="slot4">4</div>
    <div class="slot occupied" id="slot5">5</div>
    <div class="slot occupied" id="slot6">6</div>
    <div class="slot occupied" id="slot7">7</div>
    <div class="slot occupied" id="slot8">8</div>
    <div class="slot occupied" id="slot9">9</div>
    <div class="slot occupied" id="slot10">10</div>
    <!-- Thêm các ô khác nếu cần -->
</div>


<!--================================================================-->

<!-- Nut dieu khien led -->
<script>
    function nutdieukhien() {
        // Đăng ký các topic
        client.subscribe("esp32/ledState", (err) => {
            if (err) console.error("Subscription error:", err);
        });
        client.subscribe("web/ledControl", (err) => {
            if (err) console.error("Subscription error:", err);
        });

        let LedState = "Turn off";

        // Xử lý tin nhắn từ ESP32
        client.on('message', (topic, message) => {
            if (topic === "esp32/ledState") {
```

```
                LedState = message.toString() === "On" ? "Turn on" :
"Turn off";
                document.getElementById('btn1').innerHTML = LedState
=== "Turn on" ? "Đèn đang bật" : "Đèn đang tắt";
            }
        });

        // Gửi lệnh điều khiển
        function sendCommand() {
            client.publish('web/ledControl', LedState, (err) => {
                if (!err) console.log("Đã gửi tin nhắn: " + LedState);
            });
        }

        // Tạo nút điều khiển
        const button = document.createElement('button');
        button.id = 'btn1';
        button.innerHTML = 'Đèn đang tắt';
        button.onclick = function () {
            LedState = LedState === "Turn on" ? "Turn off" : "Turn on";
            sendCommand();
        };
        document.getElementById('buttonContainer1').appendChild(button)
;
    }
</script>


<script>
    function nutDieuKhienMayBom() {
        // Đăng ký các topic
        client.subscribe("esp32/motorState", (err) => {
            if (err) console.error("Subscription error:", err);
        });
        client.subscribe("web/motorControl", (err) => {
            if (err) console.error("Subscription error:", err);
        });

        let motorState = "Turn off";

        // Xử lý tin nhắn từ ESP32
        client.on('message', (topic, message) => {
            if (topic === "esp32/motorState") {
                motorState = message.toString() === "On" ? "Turn on" :
"Turn off";
                document.getElementById('btn2').innerHTML = motorState
=== "Turn on" ? "Máy bơm đang bật" : "Máy bơm đang tắt";
            }
        });

        // Gửi lệnh điều khiển
        function sendCommand() {
            client.publish('web/motorControl', motorState, (err) => {
                if (!err) console.log("Đã gửi tin nhắn: " +
motorState);
            });
        }
```

```
        // Tạo nút điều khiển
        const button = document.createElement('button');
        button.id = 'btn2';
        button.innerHTML = 'Máy bơm đang tắt';
        button.onclick = function () {
            motorState = motorState === "Turn on" ? "Turn off" : "Turn
on";
            sendCommand();
        };
        document.getElementById('buttonContainer2').appendChild(button)
;
    }
</script>


    <script>
    function updateLoginStatus(status) {
        const loginStatus = document.getElementById("loginStatus");
        if (status === "success") {
            loginStatus.className = "status success";
            loginStatus.textContent = "Đăng nhập thành công !";
        } else if (status === "failure") {
            loginStatus.className = "status failure";
            loginStatus.textContent = "Sai mật khẩu hoặc tài khoản !";
            client.publish("web/Logout", "True", { qos: 0, retain:
false }); // Reset trạng thái trên ESP32
        }

        // Sau 1 giây, ẩn thông báo
        setTimeout(() => {
            loginStatus.className = "status"; // Reset class
            loginStatus.textContent = "";    // Xóa nội dung
        }, 1000); // 1000ms = 1s
    }
    </script>

</body>

<footer>
    <p> &copy; 2024   </p>
</footer>

</html>
```