HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1

BÀI TẬP LỚN

Tên đề tài: Bãi đỗ xe thông minh

Giảng viên: Kim Ngọc Bách

Môn học: IOT và ứng dụng

Tên lớp: D22CNPM02

Nhóm bài tập lớn : 8

Danh sách thành viên nhóm:

1. *Nguyễn Ngọc Thịnh* *MSV: B22DCCN832*
2. *Nguyễn Viết Quyền* *MSV: B22DCCN676*
3. *Nguyễn Quyết Tiến* *MSV: B22DCCN724*
4. *Nguyễn Tân Thành* *MSV: B22DCCN796*

Hà Nội, 2025

MỤC LỤC

DANH MỤC BẢNG BIỂU **1**
DANH MỤC HÌNH VẼ **2**
I. GIỚI THIỆU CHUNG **3**
1.1 Lý do chọn đề tài 3
1.2 Tổng quan dự án 3
1.3 Mục đích của dự án 3
1.4 Đối tượng và phạm vi nghiên cứu 4
1.5 Phương pháp nghiên cứu 4
1.6 Công nghệ và thiết bị cần thiết cho dự án 4
II. NỀN TẢNG LÝ THUYẾT **6**
2.1 ESP32 Cam 6
2.2 ESP32 (ESP-WROOM-32) 7
2.3 Module Camera OV2640 8
2.4 Cảm biến siêu âm SRF05 10
2.5 Động cơ Servo SG90 11
2.5 Giao thức truy nhập mạng 13
2.6 Giao thức truyền thông IoT 13
a) WebSocket 13
b) MQTT 14
2.7 Giao thức ngoại vi phần cứng 15
a) UART 15
b) PWM - Điều khiển servo barie SG90 16
c) Trigger/Echo 16
2.8 Giao thức bảo mật và quản lý hệ thống 17
a) HTTPS/SSL-TLS - Mã hóa đường truyền 17
b) RESTful API - Giao thức tương tác Backend 18
c) OTA Update 20
III. PHÂN TÍCH YÊU CẦU CHỨC NĂNG **21**
3.1 Chức năng đăng nhập 21
3.2 Chức năng xem dashboard 21
3.3 Admin đặt ngưỡng cảnh bảo 21
3.4 Thiết lập lịch đóng/mở hệ thống 21
3.5 Thống kê báo cáo lượng xe, thời gian đỗ trung bình 22

3.6 Thêm mới, cấu hình cảm biến hoặc camera.....22
3.7 Cập nhật phần mềm thiết bị từ xa..... 22
3.8 Tạo tài khoản, phân quyền người dùng.....23
IV. THIẾT KẾ HỆ THỐNG..... 24
4.1 Đặc tả mô hình hình miền.....24
4.2 Đặc tả các thành phần chức năng và hoạt động..... 25
4.3 Đặc tả luồng hoạt động..... 27
4.4 Tích hợp thiết bị.....30
4.5 Phát triển ứng dụng.....32
a) Đăng nhập..... 32
b) Dashboard..... 32
c) Hiển thị trạng thái từng chỗ đỗ xe(trống/có xe)..... 33
d) Đặt ngưỡng cảnh báo..... 34
e) Thiết lập lịch đóng/mở hệ thống..... 35
f) Thống kê báo cáo lượng xe, thời gian đỗ trung bình.....35
g) Thêm mới, cấu hình cảm biến hoặc camera.....36
h) Cập nhật phần mềm thiết bị từ xa..... 37
i) Tạo tài khoản, phân quyền người dùng.....37
V. KẾT LUẬN.....39
5.1 Kết quả đạt được..... 39
5.2 Hạn chế..... 39
5.3 Hướng phát triển trong tương lai..... 40
TÀI LIỆU THAM KHẢO.....41

DANH MỤC BẢNG BIỂU

Bảng 2.1: Đặc điểm kỹ thuật của wifi..... 14

Bảng 2.2 Đặc tính kỹ thuật của giao thức MQTT..... 15

Bảng 4.1 Thư viện sử dụng của vi xử lý..... 28

Bảng 4.2 Các hàm chức năng của vi xử lý..... 29

Bảng 4.3 các thư viện sử dụng phía server..... 30

Bảng 4.4 Các endpoint trong dự án..... 30

Bảng 4.5 Bảng mô tả cảm biến được sử dụng..... 33

DANH MỤC HÌNH VẼ

Hình 2.1 Hình ảnh vi xử lý ESP32 - Cam..... 7
Hình 2.2 Vi xử lý ESP32 (ESP-WROOM-32)..... 8
Hình 2.3 Hình ảnh thực tế camera OV2640..... 10
Hình 2.4 Cảm biến siêu âm SRF05..... 11
Hình 2.5 Động cơ servo (SG90)..... 12
Hình 4.1 Mô hình miền Bãi đỗ xe thông minh (Smart Parking Domain Model).....25
Hình 4.2: Đặc tả các thành phần chức năng..... 26
Hình 4.3 Sơ đồ mạch thiết kế cho vi xử lý ESP32 CAM..... 31
Hình 4.4 Sơ đồ mạch thiết kế cho vi xử lý ESP32 DEV-KIT..... 32
Hình 4.5 Giao diện đăng nhập.....33
Hình 4.6 : Giao diện dashboard.....34
Hình 4.7: Giao diện sơ đồ bãi đỗ xe.....35
Hình 4.8 : Giao diện ngưỡng cảnh báo và quy tắc..... 36
Hình 4.9 : Giao diện lịch hoạt động..... 36
Hình 4.10: Giao diện thống kê & báo cáo.....37
Hình 4.11: Giao diện quản lý thiết bị..... 38
Hình 4.12: Giao diện cập nhật firmware..... 38
Hình 4.13: Giao diện quản lý người dùng.....39

I. GIỚI THIỆU CHUNG

1.1 Lý do chọn đề tài

Vấn đề bãi đỗ xe tại các đô thị lớn đang đối mặt với nhiều thực trạng nan giải:

- Thiếu hụt trầm trọng: Nhu cầu đỗ xe, đặc biệt là ô tô, vượt quá khả năng đáp ứng của cơ sở hạ tầng hiện tại.
- Lãng phí thời gian: Người lái xe phải tiêu tốn quá nhiều thời gian chỉ để tìm kiếm một chỗ trống trong bãi.
- Vi phạm và ùn tắc: Tình trạng đỗ xe trái phép như chặn lối đi bộ (sidewalk-blocking) hay đỗ xe song song (double-parked) diễn ra phổ biến, làm giảm khả năng lưu thông của đô thị.
- Khó khăn trong quản lý: Các bãi đỗ xe thường xuyên trong tình trạng quá tải (full lots) và việc thực thi quy định còn hạn chế (limited enforcement), khiến người quản lý gặp nhiều trở ngại và quá tải trong việc kiểm soát.

⇒ Do đó, việc phát triển một hệ thống bãi đỗ xe thông minh là cần thiết để giải quyết bài toán quy hoạch, tiết kiệm thời gian cho tài xế và hỗ trợ người quản lý kiểm soát hiệu quả hơn.

1.2 Tổng quan dự án

Dự án xây dựng một "Hệ thống Bãi đỗ xe Thông minh" (Smart Parking System) tập trung vào ba yếu tố cốt lõi: Tự động hóa (Automation), Giám sát (Monitoring) và Tối ưu hóa (Optimization).

- Mô hình hoạt động: Hệ thống được triển khai theo mô hình Client-Server kết hợp với IoT. Quy trình vận hành khép kín từ lúc xe vào bãi, nhận diện biển số, kiểm tra điều kiện, tự động mở barie cho đến khi xe rời bãi.
- Kiến trúc: Hệ thống sử dụng Gateway IoT để thu thập và xử lý sơ bộ dữ liệu từ thiết bị trước khi gửi lên Server Backend và lưu trữ tập trung tại CSDL để phục vụ quản lý và truy xuất lịch sử.
- Kết quả hướng tới: Hoàn thiện kết nối phần cứng ổn định, tích hợp camera nhận diện biển số dựa trên công nghệ AI (YOLO + OpenCV) và cung cấp giao diện quản lý trực quan.

1.3 Mục đích của dự án

Dự án hướng tới các mục tiêu cụ thể sau:

- Giám sát tự động: Hệ thống có khả năng nhận diện, ghi nhớ biển số xe và cấp phép cho xe ra vào một cách tự động mỗi khi có lượt truy cập.
- Tự động hóa điều khiển: Thay thế việc đóng mở thủ công bằng hệ thống barie tự động sử dụng động cơ servo kết hợp với tín hiệu từ cảm biến siêu âm.

- Tối ưu hóa không gian: Phát hiện chính xác vị trí đỗ còn trống hoặc đã có xe thông qua mạng lưới cảm biến đặt tại từng ô đỗ, giúp tận dụng tối đa diện tích bãi xe.
- Hỗ trợ người dùng: Hiển thị thông tin chỗ trống thời gian thực cho tài xế thông qua thiết bị hiển thị hoặc ứng dụng, giúp việc tìm chỗ nhanh chóng hơn.
- Quản lý và thống kê: Cung cấp công cụ cho người quản lý để thống kê số lượng xe, thời gian đỗ, giám sát tình trạng hoạt động của cảm biến và lưu vết dữ liệu dài hạn.

1.4 Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu: Các thiết bị và hệ thống quản lý bãi đỗ xe thông minh, bao gồm các cảm biến siêu âm (SRF05), module camera OV2640 để thu thập hình ảnh, vi điều khiển (ESP32 & ESP32 CAM), và các thiết bị điều khiển chấp hành như động cơ servo (SG90), thanh chắn (barie), và màn hình hiển thị thông tin.

Phạm vi nghiên cứu: Đề tài tập trung vào việc nghiên cứu, thiết kế và triển khai một hệ thống quản lý bãi đỗ xe tự động hoạt động trong môi trường đô thị. Phạm vi của hệ thống sẽ bao gồm việc lắp đặt cảm biến tại các vị trí đỗ xe để giám sát trạng thái, sử dụng camera OV2640 để nhận diện biển số, thu thập và truyền tải luồng dữ liệu (video stream/data) qua mạng Wi-Fi sử dụng giao thức MQTT để đảm bảo tốc độ thời gian thực, xử lý và hiển thị thông tin chỗ trống thông qua giao diện Website/App. Hệ thống cũng được trang bị giải pháp đóng mở thanh chắn tự động dựa trên kết quả xử lý ảnh và tín hiệu từ cảm biến.

1.5 Phương pháp nghiên cứu

Phương pháp nghiên cứu lý thuyết: Tìm hiểu và nghiên cứu các công nghệ cốt lõi như vi điều khiển ESP32, cảm biến siêu âm SRF05, đặc tính kỹ thuật của camera OV2640, công nghệ xử lý ảnh (YOLO/OpenCV) và đặc biệt là giao thức MQTT để tối ưu hóa việc truyền tải hình ảnh và tín hiệu điều khiển độ trễ thấp.

Phương pháp thực nghiệm: Tiến hành thiết kế mô hình, lắp đặt và cấu hình hệ thống thực tế để đánh giá chất lượng hình ảnh từ camera OV2640, độ ổn định và tốc độ phản hồi của kết nối WebSocket khi truyền tải dữ liệu liên tục, độ chính xác của cảm biến tại từng ô đỗ, và độ tin cậy của cơ chế điều khiển barie.

Phương pháp phân tích: Sử dụng các công cụ phân tích để xử lý dữ liệu ra vào và log hệ thống. Đánh giá và so sánh các chỉ số thực tế (như độ trễ đường truyền MQTT, tỷ lệ nhận diện đúng, thời gian phản hồi của servo) với các yêu cầu kỹ thuật đề ra để xác định hiệu quả hoạt động. Đồng thời, tiến hành các điều chỉnh cần thiết về phần cứng và tối ưu hóa gói tin MQTT để hoàn thiện hệ thống.

1.6 Công nghệ và thiết bị cần thiết cho dự án

Phần cứng:

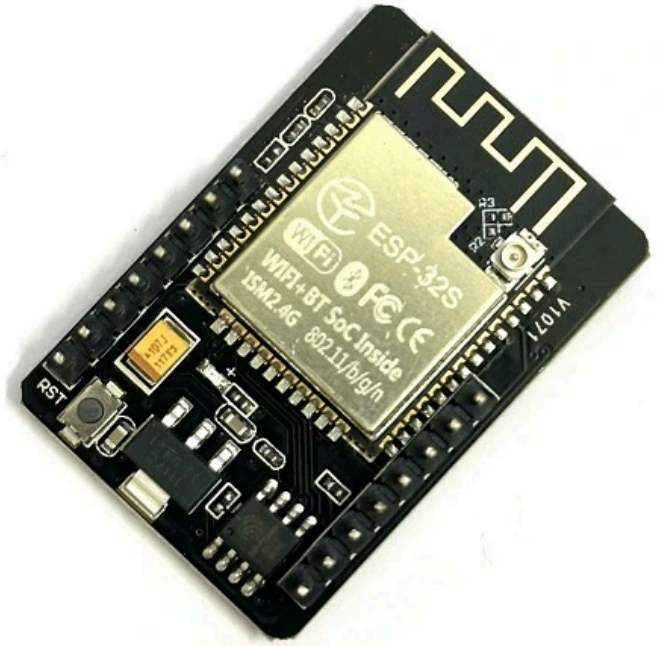
- Kit phát triển ESP32 (Wifi + Bluetooth)
- Kit phát triển ESP32 Cam (Wifi + Bluetooth)
- Module Camera OV2640 (nhận diện biển số)
- Cảm biến siêu âm SRF05
- Động cơ Servo SG90
- Board test
- Dây cắm

Phần mềm:

- Arduino IDE: Lập trình và nạp code cho ESP32
- Visual Studio Code: Thiết kế Web App và lập trình AI
- ReactJS: Thiết kế giao diện Frontend
- NodeJS: Xử lý Backend
- SQLite: Cơ sở dữ liệu lưu trữ
- Python, OpenCV, YOLOv5/v8: Xử lý ảnh và nhận diện biển số
- Giao thức: WebSocket, MQTT

II. NỀN TẢNG LÝ THUYẾT

2.1 ESP32 Cam



Hình 2.1 Hình ảnh vi xử lý ESP32 - Cam

Hệ thống dựa trên vi xử lý ESP32 lõi kép (Dual-core) 32-bit hoạt động ở xung nhịp lên đến 240MHz. Điểm đột phá về mặt lý thuyết của module này là việc tích hợp thêm bộ nhớ PSRAM (Pseudo Static RAM) 4MB bên ngoài. Vì việc xử lý hình ảnh đòi hỏi bộ nhớ đệm (buffer) rất lớn mà SRAM nội bộ của chip không đủ đáp ứng, PSRAM đóng vai trò "kho chứa" trung gian khổng lồ, cho phép lưu trữ tạm thời các khung hình độ phân giải cao trước khi xử lý hoặc truyền đi.

Nhờ kiến trúc này, ESP32-CAM có thể vận hành độc lập (Standalone) như một IP Camera thực thụ: vừa thu thập hình ảnh từ OV2640, vừa nén ảnh, vừa duy trì kết nối Wi-Fi/Bluetooth để stream video mượt mà qua WebSocket, đồng thời điều khiển các thiết bị ngoại vi (như Servo barie) mà không bị "treo" hay quá tải.

Thông số kỹ thuật:

- Vi điều khiển: ESP32-S (Xtensa® 32-bit LX6 dual-core)
- Tần số xung nhịp: 160MHz - 240MHz.
- Bộ nhớ: 520KB SRAM nội bộ + 4MB PSRAM (Yếu tố quyết định để stream HD).
- Kết nối không dây: Wi-Fi 802.11 b/g/n + Bluetooth 4.2/BLE.
- Lưu trữ: Hỗ trợ khe cắm thẻ nhớ TF/MicroSD (lên đến 4GB) để lưu log hoặc hình ảnh.
- Giao diện Camera: FPC 24 chân, hỗ trợ trực tiếp các cảm biến OV2640 và

OV7670.

- Đèn Flash: Tích hợp đèn LED siêu sáng (thường dùng GPIO 4) để trợ sáng.

Chuẩn giao tiếp:

- Giao tiếp Camera (DVP & SCCB): Module dành riêng một lượng lớn chân GPIO để tạo bus song song (Parallel Bus) nhận dữ liệu hình ảnh từ camera với tốc độ cao, đồng thời dùng giao thức I2C để điều khiển cấu hình camera.
- Giao tiếp Mạng (Wi-Fi Stack): Hỗ trợ đầy đủ các giao thức mạng TCP/IP, HTTP Server (để tạo giao diện Web), và đặc biệt là WebSocket (để truyền luồng video thời gian thực độ trễ thấp).
- Giao tiếp Ngoại vi (UART/PWM):
    + UART: Dùng để nạp code và debug qua cổng Serial.
    + PWM: Dùng để điều khiển động cơ Servo đóng mở barie.

2.2 ESP32 (ESP-WROOM-32)

Hình 2.2 Vi xử lý ESP32 (ESP-WROOM-32)

ESP32 (ESP-WROOM-32) là một dòng vi điều khiển (microcontroller) System-on-Chip (SoC) giá rẻ, tiết kiệm năng lượng được phát triển bởi Espressif Systems.

ESP32 được xây dựng dựa trên kiến trúc vi xử lý Tensilica Xtensa® LX6 32-bit lõi kép (Dual-core).

- Lõi 0 (Protocol CPU): Chuyên trách xử lý các tác vụ kết nối mạng (Wi-Fi, Bluetooth, MQTT stack).

- Lõi 1 (Application CPU): Chuyên trách xử lý logic chương trình người dùng (đọc cảm biến SRF05). Kiến trúc này cho phép hệ thống vận hành đa nhiệm (Multitasking) mượt mà thông qua hệ điều hành thời gian thực (FreeRTOS), đảm bảo việc duy trì kết nối Wi-Fi không làm gián đoạn việc điều khiển barie tức thời.

Thông số kỹ thuật:

- Điện áp hoạt động: 2.2V - 3.6V (Logic level 3.3V). Lưu ý: Cần dùng bộ chuyển mức logic nếu giao tiếp với thiết bị 5V.
- Tần số xung nhịp: Có thể điều chỉnh từ 80 MHz đến 240 MHz.
- Bộ nhớ: 520 KB SRAM nội bộ (đủ lớn để lưu trữ các bộ đệm dữ liệu MQTT).
- Kết nối không dây: Wi-Fi 2.4 GHz (tốc độ lên đến 150 Mbps) và Bluetooth 4.2/BLE.
- Số chân GPIO: 34 chân lập trình được, hỗ trợ đa chức năng (PWM, ADC, DAC, I2C, SPI).

Chuẩn giao tiếp:

- Giao tiếp Cảm biến (Digital I/O): Sử dụng các chân GPIO để gửi xung Trigger và đọc xung Echo từ cảm biến siêu âm SRF05 nhằm tính toán khoảng cách.
- Giao tiếp Điều khiển (PWM - Pulse Width Modulation): Sử dụng bộ định thời (Timer) phần cứng để xuất xung PWM điều khiển góc quay của động cơ Servo SG90 một cách chính xác và mượt mà.
- Giao tiếp Mạng (TCP/IP): Sử dụng chồng giao thức TCP/IP tích hợp để kết nối với MQTT Broker hoặc WebSocket Server, giúp truyền tải dữ liệu trạng thái bãi đỗ xe theo thời gian thực.

2.3 Module Camera OV2640

Hình 2.3 Hình ảnh thực tế camera OV2640

OV2640 là cảm biến hình ảnh màu CMOS có độ phân giải 2 Megapixel (UXGA), kích thước quang học 1/4 inch, được sản xuất bởi hãng OmniVision Technologies.

Đây là module camera phổ biến nhất cho nền tảng ESP32 (như ESP32-CAM). Nó được tích hợp bộ xử lý ảnh và nén ảnh (JPEG) ngay trên chip, giúp giảm tải đáng kể cho vi điều khiển, biến nó thành lựa chọn số một cho các dự án giám sát và truyền hình ảnh qua mạng (WebSocket/HTTP)

Khác với công nghệ BSI của OV3660, OV2640 sử dụng kiến trúc OmniPixel2™.

- Lý thuyết: Đây là kiến trúc cải tiến của công nghệ chiếu sáng mặt trước (Front-side Illumination), tập trung vào việc tối ưu hóa cấu trúc điểm ảnh (pixel architecture) để tăng hệ số lấp đầy (fill factor) và độ nhạy sáng.
- Hiệu quả: Mặc dù là công nghệ cũ hơn BSI, nhưng nhờ kích thước điểm ảnh lớn hơn (2.2µm so với 1.4µm của OV3660), OV2640 vẫn cung cấp chất lượng hình ảnh tốt, độ nhạy cao trong điều kiện ánh sáng yếu và tỷ lệ tín hiệu trên nhiễu (S/N ratio) ổn định, đáp ứng tốt cho các ứng dụng nhận diện biển số trong bãi đỗ xe thông thường.

Thông số kỹ thuật:

- Độ phân giải tối đa: 1600 x 1200 pixels (UXGA - 2MP).
- Kích thước điểm ảnh (Pixel Size): 2.2 µm x 2.2 µm (Điểm ảnh lớn giúp thu

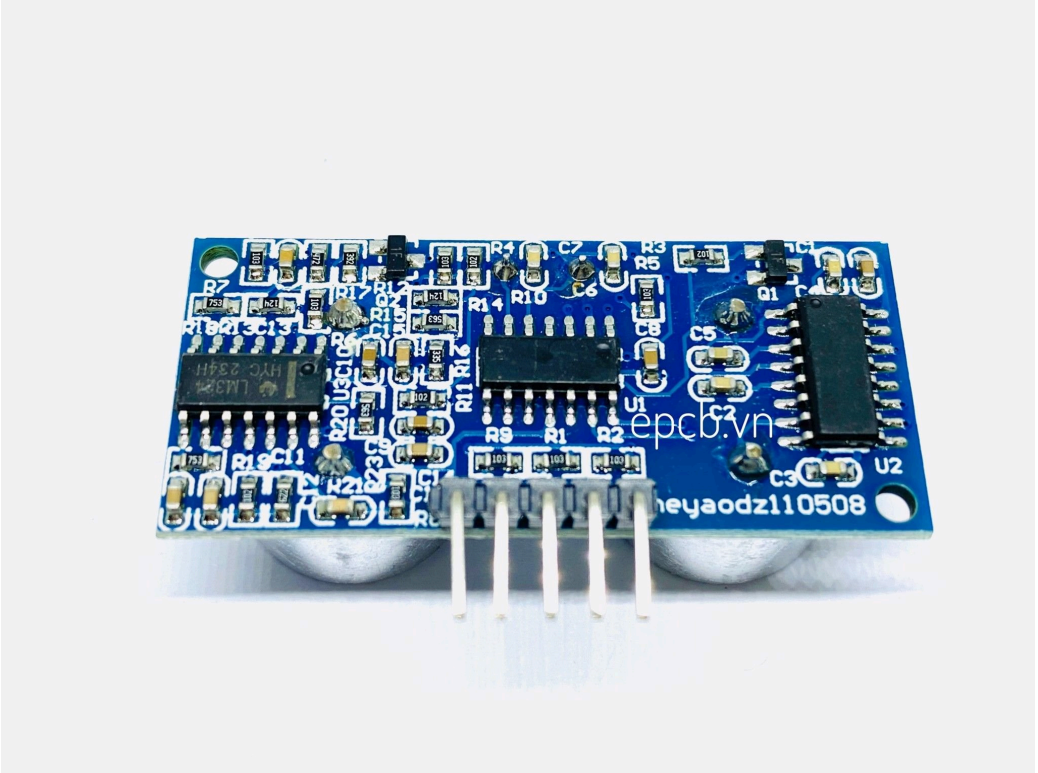
sáng tốt).

- Vùng hoạt động của ảnh (Image Area): 3590 µm x 2684 µm.
- Kích thước ống kính (Lens Size): 1/4 inch.
- Định dạng đầu ra (Output Formats): Đây là điểm mạnh nhất của OV2640:
    + Raw RGB, RGB565/555.
    + YUV(422/420), YCbCr422.
    + JPEG nén (8-bit compressed data): Hỗ trợ xuất trực tiếp ảnh JPEG từ phần cứng.
- Tốc độ khung hình:
    + UXGA/SXGA: 15 fps.
    + SVGA: 30 fps.
    + CIF: 60 fps.

Chuẩn giao tiếp:

- Giao tiếp điều khiển (SCCB): Tương thích hoàn toàn với giao thức I2C. ESP32 dùng nó để cấu hình các thanh ghi (registers) của camera như độ phân giải, độ nén JPEG, lật ảnh, v.v.
- Giao tiếp dữ liệu (DVP - Digital Video Parallel): Sử dụng các chân GPIO (Y2-Y9, VSYNC, HREF, PCLK) để truyền dữ liệu ảnh song song về ESP32.

2.4 Cảm biến siêu âm SRF05



Hình 2.4 Cảm biến siêu âm SRF05

Cảm biến siêu âm SRF05 được sử dụng để đo khoảng cách đến một vật thể mà

không cần tiếp xúc. Nguyên lý hoạt động của nó dựa trên việc phát và thu sóng siêu âm (sóng âm có tần số cao hơn ngưỡng tai người nghe được).

- Nguyên lý hoạt động:
  + Cảm biến phát ra một xung sóng siêu âm ngắn từ chân TRIG (Trigger).
  + Sóng này di chuyển trong không khí, khi gặp vật cản sẽ phản xạ lại.
  + Cảm biến nhận lại sóng phản xạ này tại chân ECHO.
  + Vi điều khiển sẽ đo khoảng thời gian (t) từ lúc phát xung đi cho đến khi nhận được xung về.
- Tính toán khoảng cách: Khoảng cách (d) được tính bằng công thức:

$$d = \frac{v * t}{2}$$

  Trong đó:
  + v là vận tốc của sóng âm trong không khí (xấp xỉ 343 m/s).
  + t là thời gian đo được.
  + Chúng ta chia cho 2 vì thời gian t là tổng thời gian sóng đi và về.

2.5 Động cơ Servo SG90

Hình 2.5 Động cơ servo (SG90)

Servo motor (động cơ servo) là một loại động cơ điện đặc biệt cho phép điều

khiển chính xác vị trí góc quay của trục động cơ. Servo SG90 là một loại servo mini rất phổ biến, giá rẻ và thường được sử dụng trong các dự án nhỏ.

- Cấu tạo: Bao gồm một động cơ DC, một hộp số giảm tốc, một biến trở (potentiometer) để theo dõi vị trí và một mạch điều khiển.
- Nguyên lý điều khiển: Vị trí của trục servo được điều khiển bằng cách gửi một chuỗi xung tín hiệu điều khiển (PWM - Pulse Width Modulation) đến dây tín hiệu của nó. Độ rộng của xung sẽ quyết định góc quay của servo.
    + Ví dụ, với servo SG90, xung rộng 1ms thường tương ứng với góc 0 độ, 1.5ms tương ứng với góc 90 độ (vị trí trung tâm), và 2ms tương ứng với góc 180 độ.

Ứng dụng: Thường được dùng trong các cơ cấu cần sự chuyển động chính xác như cánh tay robot, cơ cấu lái của xe mô hình, hoặc điều khiển các van, khớp nối.

2.5 Giao thức truy nhập mạng

ESP32 trong hệ thống sử dụng Wi-fi để kết nối Internet. Đây là giao thức truy nhập mạng chủ đạo, cho phép thiết bị truyền dữ liệu từ cảm biến và nhận lệnh điều khiển theo thời gian thực

Đặc điểm kỹ thuật:

Thuộc tính	Giá trị
Tần số hoạt động	2,4GHz(phù hợp với môi trường tầng hầm và trong nhà)
Chuẩn hỗ trợ	802.11 b/g/n
Phạm vi phủ sóng	30-50m trong nhà
Băng thông	Tối đa ~ 72 Mbps
Phương thức xác thực	WPA2, WPA3

Bảng 2.1: Đặc điểm kỹ thuật của wifi

Lý do hệ thống chọn Wi-fi:

- ESP32, ESP32 Cam tích hợp Wi-Fi → không cần module ngoài → giảm chi phí
- Phù hợp bãi đỗ xe nhỏ và trung bình
- Cho phép truyền dữ liệu theo thời gian thực(ngưỡng hệ thống yêu cầu < 3s)
- Dễ triển khai và mở rộng

Vai trò trong hệ thống:

- Kết nối ESP32, ESP32 Cam → server/ Backend
- Truyền dữ liệu cảm biến trạng thái chỗ đỗ liên tục
- Kết nối camera LPR tại cổng → gửi ảnh/biển số xe về server
- Nhận lệnh điều khiển barie từ giao diện người dùng

2.6 Giao thức truyền thông IoT

a) WebSocket

WebSocket tạo kết nối liên tục giữa server ↔ ứng dụng web, cho phép truyền dữ liệu thời gian thực mà không cần gửi request liên tục

Cơ chế hoạt động:

- Kết nối bắt đầu bằng HTTP handshake
- Sau khi nâng cấp, kết nối chuyển sang chuẩn WebSocket
- Dữ liệu được gửi theo gói frame dạng text hoặc nhị phân
- Hai chiều cùng lúc(full-duplex) giữa server ↔ client

- Không cần tạo request mới → giảm độ trễ đáng kể

Ưu điểm kỹ thuật:

- Độ trễ cực thấp(10-100ms)
- Tối ưu khi cập nhật giao diện theo thời gian thực
- Phù hợp cho các dashboard giám sát liên tục
- Tối ưu giao diện web: hiển thị chỗ đỗ theo thời gian thực
- Có thể kết hợp với SSL/TLS thành WSS để tăng bảo mật

Ứng dụng trong hệ thống:

- Server gửi dữ liệu cập nhật trạng thái các ô đỗ cho web UI theo thời gian thực
- Hiển thị bản đồ bãi xe realtime cho người quản lý
- Nhận lệnh mở barie ngay lập tức từ giao diện(open/close)
- Gửi thông báo sự kiện: xe vào/ra, bãi đầy, lỗi cảm biến

b) MQTT

MQTT ((Message Queuing Telemetry Transport) là giao thức truyền thông nhẹ, tối ưu cho các thiết bị IoT có tài nguyên hạn chế như ESP32. Trong hệ thống bãi đỗ xe thông minh, MQTT được sử dụng làm giao thức trung tâm để truyền dữ liệu cảm biến, điều khiển barie và trao đổi thông tin với server

Đặc tính kỹ thuật:

Thuốc tính	Mô tả
Mô hình	Publish/Subscribe
Broker	Hỗ trợ Mosquitto, EMQX, HiveMQ
Dung lượng gói	Rất nhỏ (~20 byte) → Tiết kiệm băng thông
QoS	0(nhanh), 1(an toàn), 2(chính xác tuyệt đối)

Bảng 2.2 Đặc tính kỹ thuật của giao thức MQTT

Lý do sử dụng:

- Phù hợp khi có nhiều cảm biến hoạt động đồng thời
    + Mỗi ô đỗ là một node IoT. Khi số lượng tăng lên MQTT giúp hệ thống vẫn ổn định nhờ mô hình Publish/Subscribe
- Tối ưu năng lượng và tài nguyên cho ESP32
    + Gói tin nhỏ → tốc độ nhanh
    + Chiếm ít RAM/Flash
    + Hoạt động tốt trong mạng Wi-Fi yếu của tầng hầm
- Không mất dữ liệu nhờ QoS
    + QoS 1: đảm bảo trạng thái chỗ đỗ không bị mất ngay cả khi mạng chập

chờn
    + QoS 2: dùng cho các lệnh quan trọng như mở barie
- Dễ mở rộng hoặc nâng cấp hệ thống

Kiến trúc sử dụng MQTT trong hệ thống

Hệ thống gồm 3 thành phần MQTT chính:

- ESP32(Publisher/Subscriber):
    + Publish: trạng thái cảm biến, dữ liệu siêu âm
    + Subscribe: lệnh mở/đóng barie, cấu hình
- MQTT Broker
    + Nhận và phân phối dữ liệu(Mosquitto/EMQX)
- Backend Server
    + Subscribe toàn bộ topic cảm biến → xử lý và lưu vào DB
    + Publish lệnh điều khiển thiết bị khi bị người quản lý thao tác

Vai trò của MQTT trong hệ thống bãi đỗ xe thông minh:

- Truyền dữ liệu cảm biến siêu âm theo thời gian thực
- Gửi cảnh báo khi xe vào/ra hoặc cảm biến lỗi
- Đồng bộ hóa trạng thái barie
- Kích hoạt điều khiển từ xa(mở/đóng barie)
- Kết nối camera LPR và quản lý sự kiện biển số

2.7 Giao thức ngoại vi phần cứng

a) UART

UART là giao thức truyền thông không đồng bộ, nghĩa là cả hai thiết bị giao tiếp không cần sử dụng chung một xung clock. Thay vào đó, chúng phải thống nhất trước một bộ thông số truyền thông, bao gồm:

- Baud rate (tốc độ truyền bit/giây)
- Số bit dữ liệu (Data bits)
- Bit kiểm lỗi (Parity bit)
- Bit kết thúc (Stop bit)

Dữ liệu được truyền dưới dạng chuỗi bit liên tục qua 2 chân chính:

- TX(Transmit): gửi dữ liệu
- RX(Receive): nhận dữ liệu

Cơ chế truyền của UART dự trên khung dữ liệu(UART frame) có cấu trúc:

Start bit | Data bits | Parity bit (Tùy chọn) | Stop bit

Đặc tính kỹ thuật:

Trong dự án, ESP32 sử dụng bộ chuyển đổi USB-UART(CH340 hoặc CP2102) để giao tiếp với máy tính, các tham số truyền thông phổ biến gồm:

- Tốc độ baud: 115200 bps(chuẩn cho ESP32)
- Độ dài dữ liệu: 8 bit

- Parity: None
- Stop bit: 1 bit
- Kiểu dữ liệu: ASCII hoặc nhị phân
- Khoảng cách truyền tối ưu: < 1 m (do UART không hỗ trợ truyền xa)

Vai trò:

- Nạp firmware qua USB(sử dụng CH340 hoặc CP2102)
- Gửi dữ liệu debug từ ESP32 về máy tính:
  + Trạng thái kết nối Wi-fi
  + Giá trị cảm biến siêu âm
  + Tín hiệu điều khiển servo barie
  + Lỗi kết nối WebSocket

b) PWM - Điều khiển servo barie SG90

Nguyên lý PWM:

- Chu kỳ 20ms(50Hz)
- Độ rộng xung quanh(pulse width) quyết định góc quay:
  + 1 ms → 0°
  + 1.5 ms → 90°
  + 2 ms → 180°

Ứng dụng:

- Điều khiển đóng/mở barie chính xác
- Tạo chuyển động mượt, tránh gây nguy hiểm cho các phương tiện

Các yêu cầu kỹ thuật khi dùng PWM:

- Cần cấp nguồn ổn định(5V, >= 500 mA)
- Chống nhiễu để tránh servo rung
- Có thể kết hợp feedback từ cảm biến chống kẹt

c) Trigger/Echo

Cơ chế hoạt động:

1. ESP32 phát xung Trigger 10µs
2. Cảm biến phát sóng siêu âm 40 kHz
3. Khi gặp vật, sóng phản xạ về chân Echo
4. Thời gian Echo HIGH tỷ lệ với khoảng cách

Công thức

$$d = \frac{t \text{ x } 343}{2}$$

Trong đó:

+ t: thời gian Echo ở mức HIGH (s)

+ 343 m/s: vận tốc âm thanh

Lý do lựa chọn:

- Rẻ, dễ triển khai, đáp ứng chính xác trạng thái chỗ đỗ
- Khoảng cách đo phù hợp (2-400cm)
- Độ nhạy cao với các vật lớn(ô tô)

2.8 Giao thức bảo mật và quản lý hệ thống

a) HTTPS/SSL-TLS - Mã hóa đường truyền

HTTPS(HyperText Transfer Protocol Secure) là phiên bản an toàn của giao thức HTTP, hoạt động dựa trên cơ chế mã hóa SSL/TLS.Đây là giao thức bảo mật được sử dụng rộng rãi nhằm bảo vệ tính an toàn của dữ liệu khi truyền qua môi trường Internet. Trong hệ thống bãi đỗ xe thông minh, HTTPS đóng vai trò then chốt trong việc bảo vệ dữ liệu nhạy cảm như biển số xe, lệnh điều khiển barie, thông tin người dùng và lịch sử ra/vào.

Cơ chế hoạt động:

HTTPS sử dụng SSL/TLS để thiết lập một phiên giao tiếp an toàn giữa client và server. Quá trình này diễn ra thông qua các bước:

1. Trao đổi phiên bản TLS và danh sách thuật toán mã hóa được hỗ trợ
2. Server gửi chứng số(SSL Certificate)
    - Chứng chỉ này được cấp bởi CA
    - Chứa Public Key của server
3. Client xác thực chứng chỉ
    - Kiểm tra xem chứng chỉ có hợp lệ, có bị giả mạo không
4. Client sinh ra khóa phiên(session key)
    - Được mã hóa bằng Public Key của server
    - Chỉ server (sở hữu Private Key) mới giải mã được
5. Cả hai bên thiết lập kênh mã hóa
    - Từ đây trở đi, mọi dữ liệu được mã hóa bằng Session Key

Các lớp bảo mật HTTPS cung cấp:

- Mã hóa dữ liệu:
    + Tất cả dữ liệu(biển số, trạng thái cảm biến, lệnh mở barie) đều được mã hóa trước khi truyền
    + Kẻ tấn công dù có "chặn" được gói tin cũng không đọc được nội dung
      → Bảo vệ dữ liệu biển số xe và thông tin người dùng
- Toàn vẹn dữ liệu:
    + Dữ liệu được gắn thêm mã kiểm tra(Message Authentication Code – MAC)
      → Nếu bị chỉnh sửa giữa đường, server sẽ phát hiện

- Xác thực:
    + Client chỉ giao tiếp với server có chứng chỉ hợp lệ
      → Tránh giả mạo server hoặc tấn công Middle Man
- Chống nghe lén:
    + Kỹ thuật viên mạng hoặc hacker không thể đọc được:
        + Biển số xe camera gửi lên
        + Token đăng nhập(JWT)
        + Dữ liệu lịch sử ra/vào
        + Lệnh điều khiển barie

Ứng dụng trong hệ thống:

- Giao tiếp Web ↔ Backend

  Trang web quản lý bãi xe yêu cầu:
    + Đăng nhập(JWT token)
    + Xem trạng thái từng ô đỗ
    + Mở barie thủ công
    + Xem lịch sử biển số xe

  → Tất cả API đều sử dụng HTTPS để:
    + Bảo vệ thông tin người dùng
    + Tránh bị đánh cắp token
    + Ngăn sửa đổi lệnh điều khiển
    + Bảo vệ dữ liệu lịch sử
- ESP32 ↔ Server

  Một số tác vụ:
    + ESP32 gửi dữ liệu trạng thái bằng HTTP(S) POST
    + Server gửi cấu hình hoặc firmware update

  HTTPS đảm bảo:
    + Không ai đọc được dữ liệu tử cảm biến
    + Không ai chèn lệnh độc hại vào thiết bị
    + Tránh giả mạo server → tránh update firmware độc hại
- Camera LPR ↔ Server

  Camera nhận diện biển số gửi:
    + Ảnh phương tiện
    + Chuỗi kí tự biển số
    + Thời gian ghi nhận

  Dữ liệu này rất nhạy cảm, và được bảo vệ bằng HTTPS để:
    + Tránh rò rỉ biển số xe
    + Ngăn thay đổi kết quả nhận diện
    + Ngăn gửi ảnh giả mạo để bypass hệ thống

b) RESTful API - Giao thức tương tác Backend

RESTful API (Representational State Transfer) là phong cách thiết kế API dựa trên giao thức HTTP/HTTPS, cho phép frontend (web/app) và backend (server) giao tiếp với nhau theo cách thống nhất và dễ mở rộng. Trong hệ thống bãi đỗ xe thông minh, REST API là "cầu nối" giữa:

- Giao diện web quản lý
- Hệ thống xử lý backend
- Cơ sở dữ liệu (trạng thái ô đỗ, lịch sử xe, người dùng…)

Vai trò của REST API trong hệ thống:

- Truy vấn trạng thái bãi xe
    + Frontend gọi API để lấy danh sách các chỗ đỗ (trống/đã có xe).
    + Ví dụ: hiển thị bản đồ bãi xe theo thời gian thực.
- Ghi nhận xe vào/ra
    + Khi camera LPR hoặc cảm biến kích hoạt sự kiện, backend ghi nhận:
        - biển số
        - thời gian vào/ra
        - vị trí đỗ
- Điều khiển barie thủ công
    + Nhân viên bấm nút mở barie trên giao diện → frontend gửi request lên REST API → backend kiểm tra quyền (JWT, role) → gửi lệnh tới thiết bị.
- Xem lịch sử sự kiện
    + Quản lý có thể xem lại log: xe vào/ra lúc nào, biển số gì, ô nào, nhân viên nào thao tác.

Các phương thức API tiêu biểu:

- RESTful API sử dụng các phương thức HTTP để biểu diễn hành động:
    + GET → Lấy dữ liệu
    + POST → Tạo mới/gửi lệnh
    + PUT/PATCH → Cập nhật dữ liệu
    + DELETE → Xóa dữ liệu

Đặc tính của RESTful API trong hệ thống:

- Dựa trên HTTP/HTTPS:
    + Dễ triển khai, tương thích mọi nền tảng.
    + Kết hợp với HTTPS để bảo mật dữ liệu (biển số, token…).
- Chuẩn hóa giao tiếp giữa frontend và backend
    + Dữ liệu thường ở dạng JSON → frontend dễ xử lý.

    + Một API có thể phục vụ nhiều client (web, mobile, dashboard quản lý).

- Dễ mở rộng và tích hợp
    + Có thể dễ dàng thêm API mới: báo cáo thống kê, biểu đồ lưu lượng xe, API cho ứng dụng di động…
    + Dễ tích hợp với hệ thống khác (thanh toán, ERP, CRM…) nếu mở rộng.

c) OTA Update

OTA (Over-The-Air Update) là cơ chế cho phép cập nhật firmware của thiết bị IoT thông qua mạng Internet mà không cần kết nối cáp USB hoặc tháo thiết bị khỏi vị trí lắp đặt. Đây là một thành phần quan trọng trong các hệ thống IoT thực tế, đặc biệt đối với thiết bị được đặt ở môi trường ngoài trời như cảm biến bãi đỗ xe.

Trong hệ thống bãi đỗ xe thông minh, OTA giúp cập nhật firmware của ESP32 để sửa lỗi, tối ưu hiệu năng hoặc bổ sung tính năng mới mà không cần can thiệp vật lý.

Nguyên lý hoạt động:

1. ESP32 kiểm tra phiên bản firmware mới trên server
2. Nếu có bản mới, thiết bị tải về file firmware (nhị phân .bin) thông qua HTTP/HTTPS hoặc liên kết được gửi qua WebSocket/MQTT
3. ESP32 ghi firmware mới vào vùng nhớ Flash
4. Thiết bị tự động khởi động lại và chạy phiên bản mới

Các lợi ích trong hệ thống bãi đỗ xe:

- Cập nhật hàng loạt thiết bị
- Vá lỗi nhanh chóng
- Nâng cấp tính năng mà không tháo thiết bị
- Giảm chi phí bảo trì

Yêu cầu để OTA hoạt động an toàn:

- Firmware phải được **ký số** để tránh bị giả mạo
- Kênh tải firmware phải qua HTTPS
- Thiết bị cần kiểm tra checksum để đảm bảo integrity
- Cần có cơ chế rollback nếu cập nhật thất bại

III. PHÂN TÍCH YÊU CẦU CHỨC NĂNG

3.1 Chức năng đăng nhập

- Giao diện sẽ có hai ô điền tài khoản và mật khẩu và 1 nút Đăng nhập để truy cập web.
- Dữ liệu về tài khoản và mật khẩu sẽ có trong cơ sở dữ liệu, với 3 thuộc tính: id (PK), username, password. Trong đó password được mã hoá
- Nếu đăng nhập thành công, web sẽ chuyển hướng tới giao diện chính tuỳ thuộc vào vai trò của người dùng
- Đăng nhập thất bại, hệ thống sẽ đẩy ra một thông báo sai tên tài khoản hoặc mật khẩu và quay lại giao diện đăng nhập.

3.2 Chức năng xem dashboard

- Thống kê tổng quan (Tổng chỗ đỗ, Đang có xe, Còn trống, Cảnh báo) : Hiển thị 4 thẻ (cards) số liệu quan trọng nhất ở đầu trang
- Trực quan hóa Sơ đồ bãi xe:
    + Hiển thị lưới các ô đỗ xe mô phỏng vị trí thực tế.
    + Trạng thái Có xe: Hiển thị màu Xanh lá (theo mô tả của bạn) hoặc Xanh dương (theo chú thích trong ảnh).
    + Trạng thái Trống: Hiển thị màu Xanh dương (theo mô tả của bạn) hoặc Xanh lá
- Số liệu phải tự động cập nhật real-time thông qua WebSocket/API polling.

3.3 Admin đặt ngưỡng cảnh bảo

- Thiết lập ngưỡng kích hoạt: Nhập giá trị % công suất (bãi đầy/sắp đầy) và % pin/thời gian mất kết nối để hệ thống tự động phát cảnh báo.
- Quản lý trạng thái quy tắc: Cung cấp nút gạt để Bật/Tắt nhanh từng loại cảnh báo và cấu hình số lần tự động thử kết nối lại (retry) trước khi báo lỗi.
- Cấu hình kênh thông báo: Tùy chọn Bật/Tắt các kênh gửi tin (Email, SMS, Push Notification) và cập nhật danh sách người nhận.
- Kiểm soát tần suất (Chống Spam): Cài đặt khoảng thời gian tối thiểu giữa các lần gửi thông báo để tránh gửi liên tục khi thiết bị chập chờn.

3.4 Thiết lập lịch đóng/mở hệ thống

- Quản lý danh sách lịch trình: Hiển thị chi tiết tên lịch, hành động cụ thể (đóng/mở barrier, bật/tắt đèn), mốc thời gian và các ngày áp dụng trong tuần (T2-CN).
- Thao tác nhanh: Cung cấp nút gạt (Switch) để kích hoạt hoặc tạm dừng từng lịch trình ngay trên danh sách mà không cần xóa.

- Thêm mới và Chỉnh sửa: Form cấu hình cho phép chọn giờ thực hiện, lặp lại theo thứ trong tuần và gán hành động điều khiển cho từng nhóm thiết bị.
- Thiết lập giờ mặc định: Cấu hình khung giờ mở/đóng cửa cố định cho ngày thường và cuối tuần, hoặc tùy chọn chế độ hoạt động 24/7.
- Thống kê tổng quan: Hiển thị dashboard tóm tắt tổng số lịch trình, số lịch đang chạy và khung giờ hoạt động hiện tại của bãi xe.

3.5 Thống kê báo cáo lượng xe, thời gian đỗ trung bình

- Thống kê chỉ số tổng quan: Hiển thị 4 chỉ số hiệu suất chính (KPIs) gồm: Tổng lượt xe, Thời gian đỗ trung bình, Công suất trung bình và Giờ cao điểm (kèm theo chỉ số % tăng/giảm so với kỳ trước).
- Biểu đồ lưu lượng: So sánh số lượng xe vào và xe ra theo ngày/giờ dưới dạng biểu đồ cột đôi.
- Biểu đồ phân bố: Thể hiện tỷ trọng lấp đầy của từng khu vực (Khu A, B, C, D) dưới dạng biểu đồ tròn (Donut chart).
- Bộ lọc dữ liệu đa chiều: Cho phép lọc báo cáo theo khoảng thời gian (Tuần này, Tháng này, Tùy chỉnh) và theo đối tượng (Tất cả, Từng khu vực).
- Mẫu báo cáo tự động: Cung cấp các file báo cáo đã được tổng hợp sẵn theo chu kỳ (Báo cáo ngày, tuần, tháng) để tải xuống nhanh.
- Xuất dữ liệu: Chức năng xuất dữ liệu chi tiết ra file Excel để lưu trữ hoặc file PDF để in ấn báo cáo trực tiếp.

3.6 Thêm mới, cấu hình cảm biến hoặc camera

- Quản lý danh sách chi tiết: Hiển thị bảng dữ liệu thời gian thực gồm: Mã/Tên thiết bị, Loại (Cảm biến/Camera/Barrier), Khu vực lắp đặt, Trạng thái kết nối (Online/Offline), Mức pin hiện tại và Phiên bản Firmware.
- Thêm mới thiết bị: Cung cấp Form đăng ký thiết bị vào mạng lưới, yêu cầu nhập các trường định danh như: Loại thiết bị, Tên gợi nhớ, Vị trí và Địa chỉ vật lý (IP/MAC).
- Điều khiển & Bảo trì: Hỗ trợ menu thao tác nhanh cho từng thiết bị bao gồm: Chỉnh sửa thông tin, Xóa thiết bị và Gửi lệnh khởi động lại (Remote Reboot) từ xa.
- Tìm kiếm & Bộ lọc: Công cụ tìm kiếm theo tên/mã và bộ lọc dữ liệu đa chiều theo Loại thiết bị, Khu vực hoặc Trạng thái hoạt động.

3.7 Cập nhật phần mềm thiết bị từ xa

- Thống kê tiến độ cập nhật: Dashboard hiển thị 4 chỉ số trạng thái thực tế của toàn hệ thống: Số thiết bị đã cập nhật mới nhất, Số thiết bị lỗi thời (cần cập nhật), Số đang chạy cập nhật và Số đang chờ trong hàng đợi.

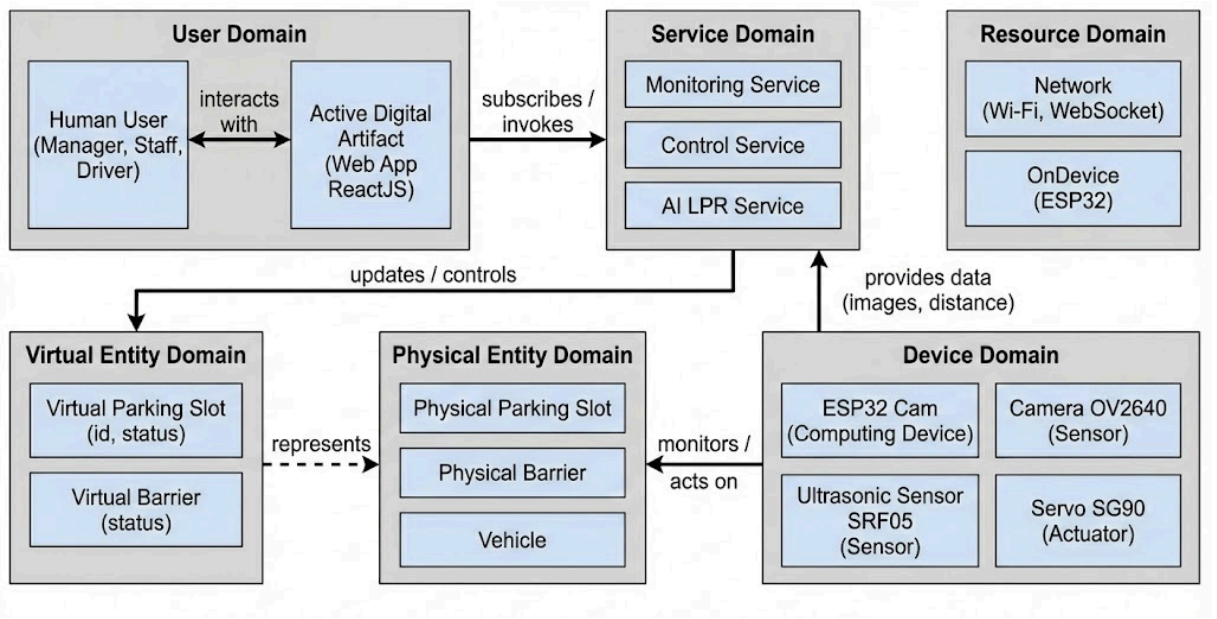
- Quản lý kho Firmware: Hiển thị danh sách các phiên bản phần mềm khả dụng kèm thông tin chi tiết (số hiệu phiên bản, ngày phát hành, dung lượng, ghi chú thay đổi - changelog) và tính năng tải lên bản firmware mới.
- Cấu hình triển khai: Cho phép chọn đối tượng cập nhật theo nhóm (Khu vực A, B, C...) hoặc từng thiết bị cụ thể để áp dụng bản vá hàng loạt.
- Theo dõi & Kiểm soát: Giám sát hàng đợi cập nhật theo thời gian thực (Real-time) với thanh tiến độ % cho từng thiết bị; cung cấp các lệnh điều khiển như "Bắt đầu", "Tạm dừng" hoặc "Làm mới".

3.8 Tạo tài khoản, phân quyền người dùng

- Quản lý danh sách người dùng: Hiển thị bảng thông tin chi tiết (Họ tên, Email, SĐT), trạng thái hoạt động (Online/Offline/Locked) và lịch sử đăng nhập gần nhất; hỗ trợ tìm kiếm và lọc theo vai trò.
- Phân quyền vai trò (RBAC): Thiết lập cơ chế phân quyền chặt chẽ cho 4 nhóm đối tượng:
    + Quản trị viên: Toàn quyền hệ thống.
    + Quản lý: Được phép cấu hình thiết bị, xem báo cáo và điều khiển.
    + Nhân viên: Giới hạn quyền giám sát và thao tác barrier thủ công.
    + Khách: Quyền chỉ đọc (View-only) để xem trạng thái bãi xe.
- Thao tác quản trị: Cung cấp đầy đủ các chức năng vòng đời tài khoản: Thêm mới, Chỉnh sửa thông tin, Đặt lại mật khẩu (Reset Password), Khóa/Mở khóa tạm thời và Xóa tài khoản.

IV. THIẾT KẾ HỆ THỐNG

4.1 Đặc tả mô hình hình miền



Hình 4.1 Mô hình miền Bãi đỗ xe thông minh (Smart Parking Domain Model)

a. Miền User: Là vùng của các đối tượng tương tác trực tiếp với hệ thống
   - Human User: Bao gồm Người quản lý (Manager) giám sát hệ thống, Nhân viên (Staff) vận hành và Tài xế (Driver) xem thông tin bãi đỗ
   - Active Digital Artifact: Là giao diện kỹ thuật số giúp người dùng tương tác, cụ thể ở đây là Web App (Dashboard quản lý và giao diện xem chỗ trống)

b. Miền Virtual Entity: Là các đại diện kỹ thuật số trong phần mềm cho các đối tượng thực tế
   - Parking Lot: Không gian quản lý chung trên phần mềm
   - Parking Slot: Đối tượng lưu trữ trạng thái (Trống/Có xe) trên giao diện
   - Barrier: Đối tượng đại diện cho trạng thái đóng/mở của cổng

c. Miền Physical Entity: Là các đối tượng thực tế mà hệ thống giám sát và tác động.
   - Parking Slot: Vị trí đỗ xe thực tế
   - Physical Barrier: Thanh chắn barie tại cổng ra vào
   - Vehicle: Phương tiện ra vào bãi

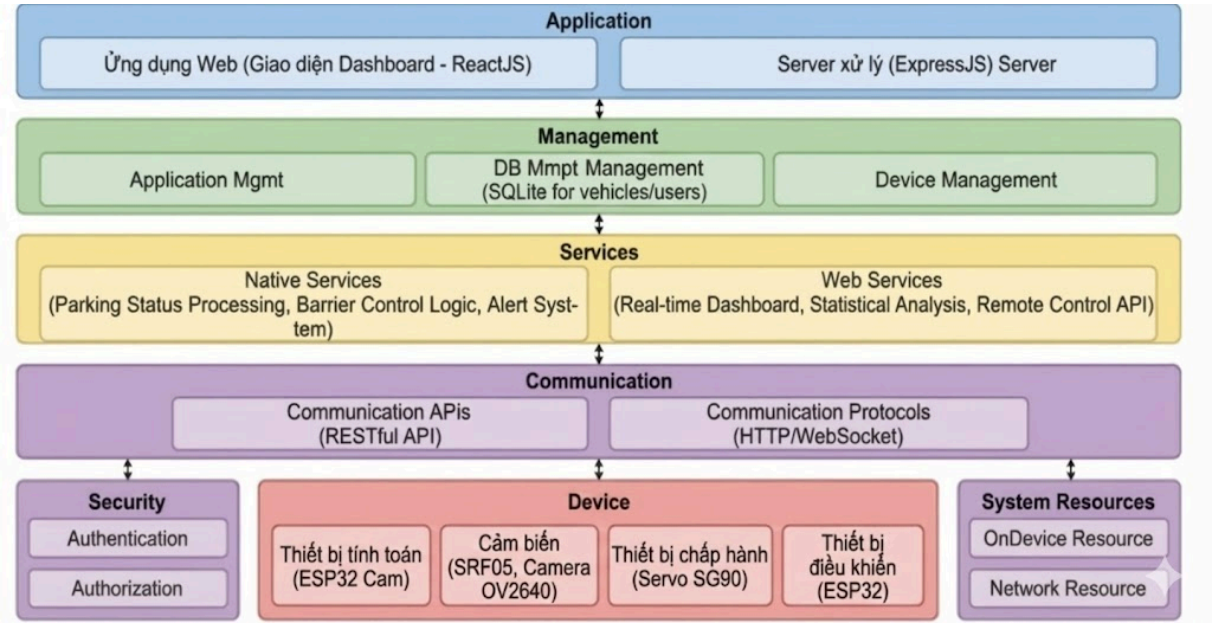
d. Miền Service: Cung cấp các chức năng logic để hệ thống vận hành
   - Monitoring Service: Dịch vụ giám sát trạng thái ô đỗ từ cảm biến
   - Control Service: Dịch vụ điều khiển đóng mở Barie
   - AI Service (LPR): Dịch vụ nhận diện biển số xe từ hình ảnh

e. Miền Resource:
   - Network Resource: Hạ tầng mạng Wi-Fi, Internet (HTTP, WebSocket) kết nối các thành phần

- OnDevice Resource: Tài nguyên tính toán trên ESP32 Cam để xử lý ảnh và tín hiệu cảm biến

f. Miền Device: Chứa các phần cứng IoT.
   - Computing Device: Vi điều khiển ESP32 CAM, ESP32
   - Sensor: Cảm biến siêu âm SRF05 và Module Camera OV2640
   - Actuator: Động cơ Servo SG90 (điều khiển Barie)

4.2 Đặc tả các thành phần chức năng và hoạt động



Hình 4.2: Đặc tả các thành phần chức năng

a) Thành phần chức năng

- Thiết bị IoT: Bao gồm các cảm biến (SRF05, Camera OV2640), thiết bị chấp hành (Servo SG90) và thiết bị tính toán (ESP32 Cam) được ánh xạ tới khối Device và được quản lý bởi khối Management (Device Management)
- Tài nguyên: Các tài nguyên xử lý trên thiết bị và tài nguyên mạng được ánh xạ tới khối Device (OnDevice Resource) và khối Communication (Communication APIs và Protocols như RESTful, WebSocket)
- Dịch vụ điều khiển: Các logic xử lý nghiệp vụ cốt lõi như Parking Status Processing (Xử lý trạng thái đỗ), Barrier Control Logic (Logic điều khiển Barie) và Alert System (Hệ thống cảnh báo) được ánh xạ tới khối Services (Native Services)
- Dịch vụ Web: Các dịch vụ phục vụ giao diện người dùng như Real-time Dashboard, Statistical Analysis và Remote Control API được ánh xạ tới khối Services (Web Services)
- Cơ sở dữ liệu: Hệ quản trị cơ sở dữ liệu SQLite lưu trữ thông tin xe vào/ra và người dùng được ánh xạ tới khối Management (Database Management) và khối Security (Authorization)

- Ứng dụng Web: Giao diện Dashboard (ReactJS) và Server xử lý (ExpressJS) được ánh xạ tới khối Application, khối Management (Application Management) và khối Security (Authentication) để xác thực người dùng

b) Thành phần hoạt động

Application:

- Web App: ReactJS Web App
- Application Server: ExpressJS App Server
- Database Server: SQLite

Management:

- Application Management: ExpressJS App Management
- Database Management: SQLite Management
- Device Management: Web App

Service:

- Native Services:
    + Parking Status Processing: Dịch vụ nhận dữ liệu thô từ cảm biến siêu âm, lọc nhiễu và xác định logic trạng thái của từng vị trí đỗ (trống/có xe)
    + Barrier Control Logic: Dịch vụ tự động xử lý logic đóng/mở thanh chắn (servo) dựa trên cảm biến hoặc lệnh từ người quản trị
    + Alert System: Logic giám sát để gửi cảnh báo khi bãi xe đã đầy (full slot) hoặc khi thiết bị cảm biến mất kết nối.
- Web Services:
    + Real-time Dashboard (Web Socket): hiển thị trạng thái bãi xe trực quan lên Web App theo thời gian thực
    + Statistical Analysis: truy xuất dữ liệu từ SQLite để báo cáo số lượng xe ra vào, thời gian đỗ trung bình và lịch sử hoạt động
    + Remote Control API: API cho phép nhân viên vận hành điều khiển đóng/mở barie thủ công thông qua giao diện Web

Communication:

- Communication APIs: RESTful API
- Communication Protocols: HTTP, WebSocket

Security:

- Authentication: Web App, Database
- Authorization: Web App, Database

Device:

- Computing Device: ESP32 CAM, ESP32
- Sensor: cảm biến siêu âm SRF05
- Sensor: module Camera OV2640
- Actuator: Servo SG90

4.3 Đặc tả luồng hoạt động

a). Thiết bị cảm biến và xử lý

Thư viện sử dụng:

Thư viện	Chức năng
WiFi.h	Thư viện chuẩn của ESP32 giúp thiết bị kết nối với mạng Wi-Fi để truyền dữ liệu lên Internet
esp_camera.h	Thư viện chuyên dụng để cấu hình và điều khiển module Camera OV2640
PubSubClient.h	Thư viện MQTT Client phổ biến nhất cho Arduino/ESP32. Giúp ESP32 kết nối tới MQTT Broker (Server), Publish dữ liệu cảm biến và Subscribe lệnh điều khiển
ArduinoJson.h	Thư viện xử lý dữ liệu định dạng JSON
ESP32Servo.h	Thư viện điều khiển động cơ Servo để điều khiển góc quay SG90 đóng/mở Barie

Bảng 4.1 Thư viện sử dụng của vi xử lý

Các hàm chức năng:

Hàm	Chức năng
setup()	Hàm khởi tạo chạy một lần đầu tiên: Thiết lập chế độ các chân (GPIO), khởi động Camera, kết nối WiFi và gọi hàm kết nối MQTT
connectMQTT()	Kết nối tới MQTT Broker (Mosquitto/Aedes trên Server). Nếu mất kết nối, hàm này sẽ thử kết nối lại (reconnect)
connectWifi()	Thực hiện kết nối ESP32 với mạng Wi-Fi thông qua SSID và Password được cấu hình trước
initCamera()	Cấu hình các thông số cho Camera OV2640 (độ phân giải, định dạng ảnh JPEG, chân kết nối) và khởi động module camera
measureDistance()	Phát xung siêu âm từ chân Trig và đo thời gian phản hồi tại chân Echo, sau đó tính toán ra khoảng cách (cm) để xác định có xe hay không
controlBarrier(int state)	Điều khiển động cơ Servo SG90
sendSensorData()	Đóng gói dữ liệu trạng thái bãi đỗ thành chuỗi JSON và Publish (xuất bản) lên Server thông qua topic

	parking/sensor/update
captureAndSend()	Chụp một bức ảnh từ Camera OV2640, chuyển đổi dữ liệu ảnh sang Base64 và Publish lên topic parking/camera/trigger để nhận diện biển số

Bảng 4.2 Các hàm chức năng của vi xử lý

Luồng hoạt động chi tiết:

- Khởi tạo hệ thống: Đầu tiên, đặt chế độ hoạt động (INPUT/OUTPUT) cho các chân kết nối với cảm biến và thiết bị chấp hành: chân Trig (phát xung) và Servo Pin làm đầu ra, chân Echo (nhận xung) làm đầu vào. Tiếp theo, tiến hành khởi tạo cấu hình cho Camera OV2640 và thực hiện kết nối vào mạng Wifi thông qua SSID và Password đã được nạp sẵn. Sau khi có kết nối mạng, thiết bị sẽ thiết lập kết nối WebSocket tới Server.
- Đo lường và giám sát: Trong vòng lặp chính (loop), hệ thống sẽ kích hoạt cảm biến siêu âm SRF05 sau mỗi khoảng thời gian định kỳ. Cảm biến phát ra sóng siêu âm và đo thời gian phản hồi để tính toán khoảng cách từ cảm biến đến vật thể gần nhất.
- Xử lý phát hiện xe (Tại cổng vào/ra):
  + Nếu khoảng cách đo được nhỏ hơn ngưỡng thiết lập, hệ thống xác định có xe đang dừng trước cổng.
  + Vi xử lý kích hoạt Camera OV2640 chụp một bức ảnh hiện tại.
  + Ảnh và dữ liệu cảm biến được đóng gói và gửi ngay lập tức lên Server qua giao thức WebSocket (sự kiện hardware/lpr) để hệ thống AI nhận diện biển số.
- Xử lý điều khiển Barie:
  + Hệ thống liên tục lắng nghe các sự kiện từ Server thông qua hàm callback của Socket.io.
  + Khi nhận được lệnh control barrier với trạng thái "OPEN" (do AI nhận diện biển số hợp lệ), vi xử lý sẽ điều khiển động cơ Servo quay một góc 90 độ để mở thanh chắn.
  + Sau khi xe đã đi qua, vi xử lý điều khiển Servo quay về vị trí 0 độ để đóng thanh chắn, hoàn tất một chu trình ra/vào.
- Cập nhật trạng thái ô đỗ (Tại vị trí đỗ):
  + Đối với các node cảm biến lắp tại ô đỗ, nếu khoảng cách đo được thay đổi (chuyển từ Trống sang Có xe hoặc ngược lại), ESP32 sẽ gửi ngay gói tin JSON chứa slotId và status lên Server để cập nhật màu sắc trên giao diện Web theo thời gian thực.

b) Website quản lý

Thư viện sử dụng

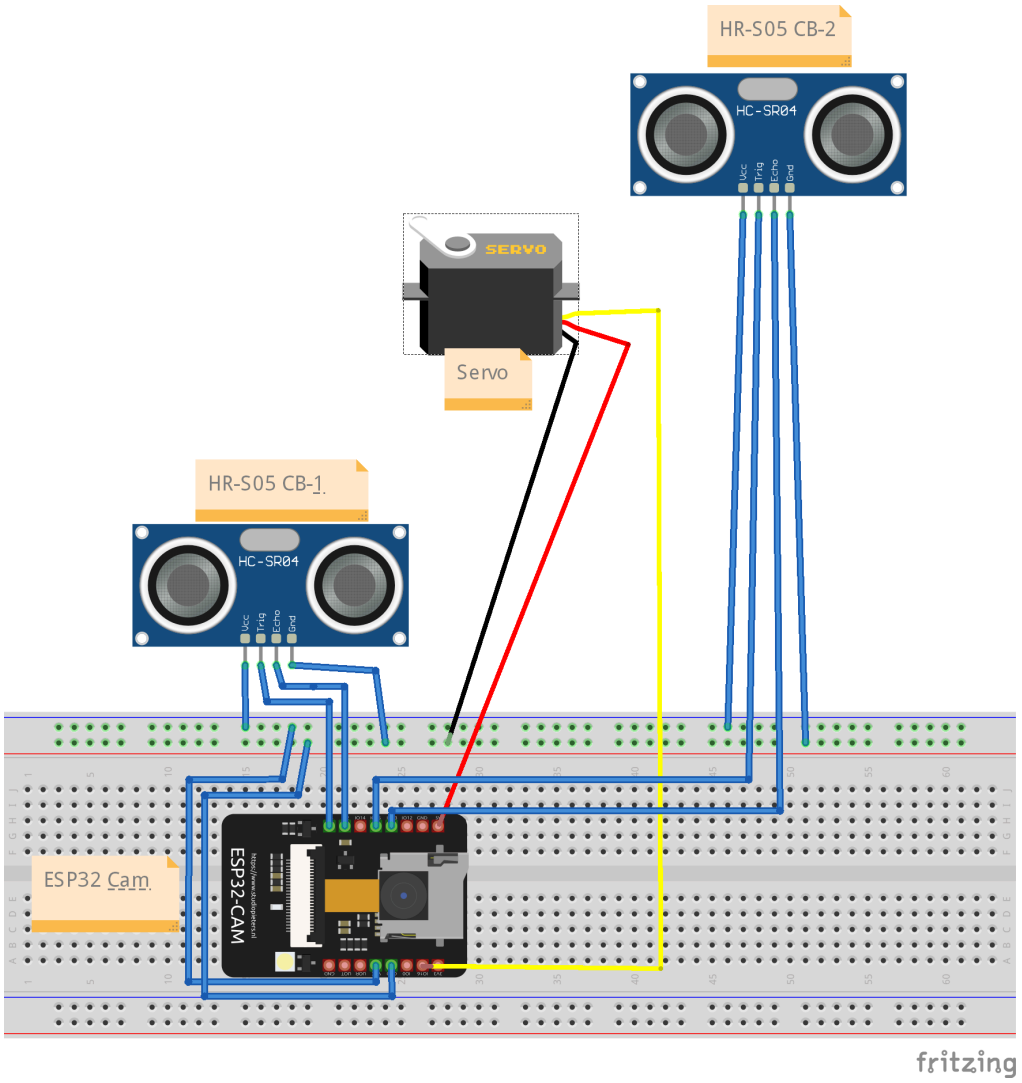
Thư viện	Chức năng
Express	Framework cho Node.js, dùng để xây dựng các ứng dụng web và API RESTful với hệ thống định tuyến và middleware linh hoạt
sqlite3	Driver để NodeJS làm việc được với file cơ sở dữ liệu SQLite
socket.io	Thư viện Websocket cho NodeJS cho phép thiết lập thời gian thực giữa client và server 1 cách liên tục

Bảng 4.3 các thư viện sử dụng phía server

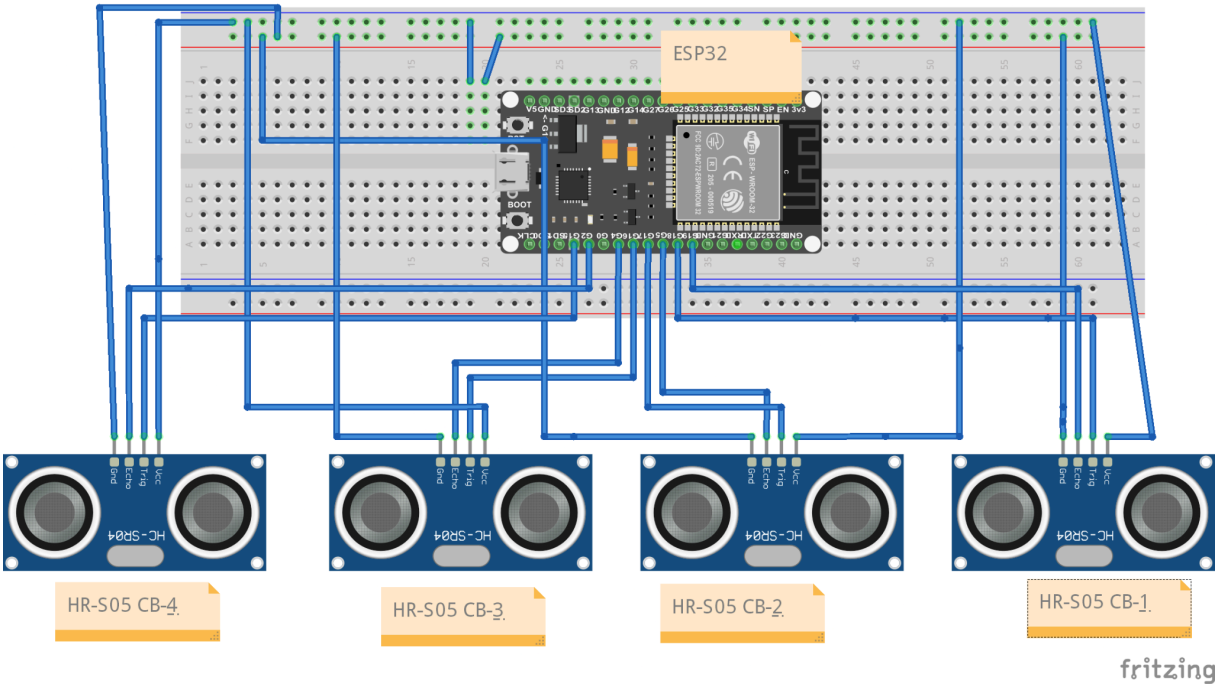
STT	Mô tả	Endpoint	Phương thức	Tham số	Trả về
1	Đăng nhập	/auth/login	POST	username, password	Token của user
2	Đăng kí	/auth/sign	POST	username, password, role	User (trừ password) + token
3	Lấy trạng thái ô đỗ	/slots	GET	Không có	Danh sách các ô đỗ
4	Lấy lịch sử ra vào	/logs	GET	Không có	Danh sách log
5	Điều khiển barie thủ công	/device/ control	POST	action (“Open” / “Close”)	message: success/fail
6	Cập nhật trạng thái cảm biến	/hardware/ sensor	POST	sensorId, status (0 hoặc 1), distance (cm)	message: update success/fail
7	Kết quả nhận diện biển số	/hardware/ lpr	POST	number, image	action: accept/deny
8	Thống kê	/stats	GET	type: month, day,..	Dữ liệu thống kê tương ứng

Bảng 4.4 Các endpoint trong dự án

4.4 Tích hợp thiết bị



Hình 4.3 Sơ đồ mạch thiết kế cho vi xử lý ESP32 CAM



Hình 4.4 Sơ đồ mạch thiết kế cho vi xử lý ESP32 DEV-KIT

- Vi điều khiển chính – ESP32 Cam vvà DevKit V0:
  ESP32 đóng vai trò là bộ xử lý trung tâm, có nhiệm vụ:
  + Lọc biển số xe qua 2 cảm biến kiểm tra để thông quan mở servo cho xe vào hoặc ra.
  + Đọc dữ liệu từ 4 cảm biến siêu âm HC-SR04 đặt tại các vị trí ô đỗ.
  + Xử lý khoảng cách để xác định ô trống / ô có xe.
  + Điều khiển servo barie mở/đóng khi có xe ra/vào.
  + Có thể gửi dữ liệu lên web/app.

  ESP32 được đặt trên breadboard, toàn bộ dây signal (trig/echo/servo) được dẫn về chân GPIO của ESP32.
- Mỗi cảm biến đo khoảng cách và phát hiện xe có trong ô hay không.
  Mỗi module có 4 chân:
  + VCC → 5V
  + GND → GND chung
  + Trig → Chân GPIO của ESP32
  + Echo → Chân GPIO của ESP32 (cần thêm điện trở giảm áp nếu Echo = 5V)
- Bảng kết nối tới esp32

Cảm biến	Chân Triger	Chân Echo	Mô Tả
CB-1	I04	I02	Cảm biến siêu âm cho cổng vào bãi
CB-2	I015	I013	Cảm biến siêu âm cho cổng ra bãi

CB-3	D18	D19	Cảm biến siêu âm chỗ đỗ xe
CB-4	D16	D4	Cảm biến siêu âm chỗ đỗ xe
CB-5	D17	D5	Cảm biến siêu âm chỗ đỗ xe
CB-6	D15	D8	Cảm biến siêu âm chỗ đỗ xe

Bảng 4.5 Bảng mô tả cảm biến được sử dụng

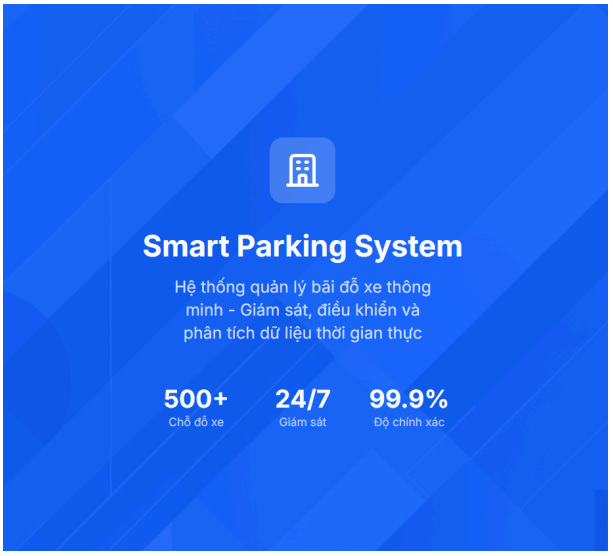
- Servo barrier

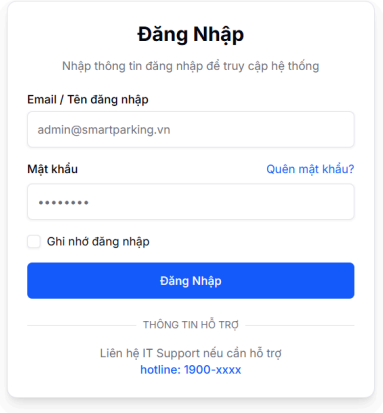
  Servo (SG90) dùng để làm thanh chắn barie.
  + Dây Đỏ → 5V
  + Dây Nâu/Đen → GND
  + Dây Vàng → Tín hiệu điều khiển từ ESP32 (PWM) -016
- Servo nhận lệnh khi:
  + Cảm biến CB-1 hoặc CB-2 phát hiện xe → mở barie
  + Sau một thời gian hoặc khi xe đã qua → đóng lại

4.5 Phát triển ứng dụng

a) Đăng nhập

- Trước khi sử dụng hệ thống cần thực hiện đăng nhập để có thể truy cập vào hệ thống quản lý, nếu sai tài khoản hoặc mật khẩu hệ thống kiểm tra và gửi thông báo sai tài khoản hoặc mật khẩu.





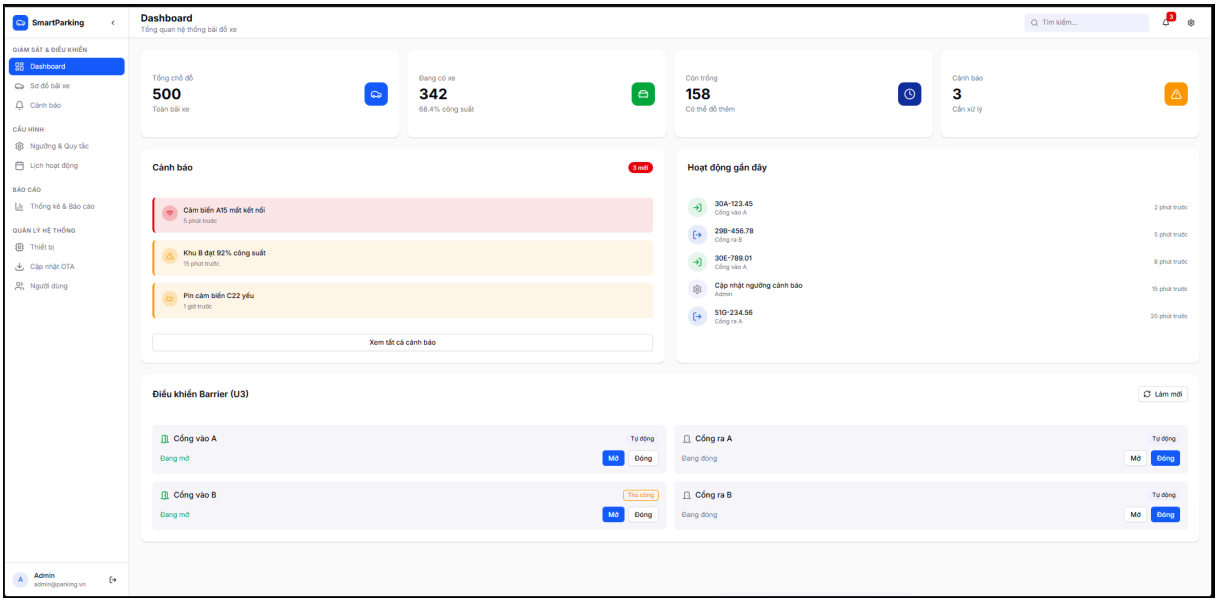
Hình 4.5 Giao diện đăng nhập

- Khi đăng nhập thành công với vai trò là quản lí, sẽ tương tác với các chức năng ở dưới.

b) Dashboard

- Thẻ thống kê nhanh: Hiển thị 4 số liệu quan trọng gồm tổng số chỗ đỗ, số chỗ đang có xe, số chỗ còn trống và số cảnh báo đang hoạt động.

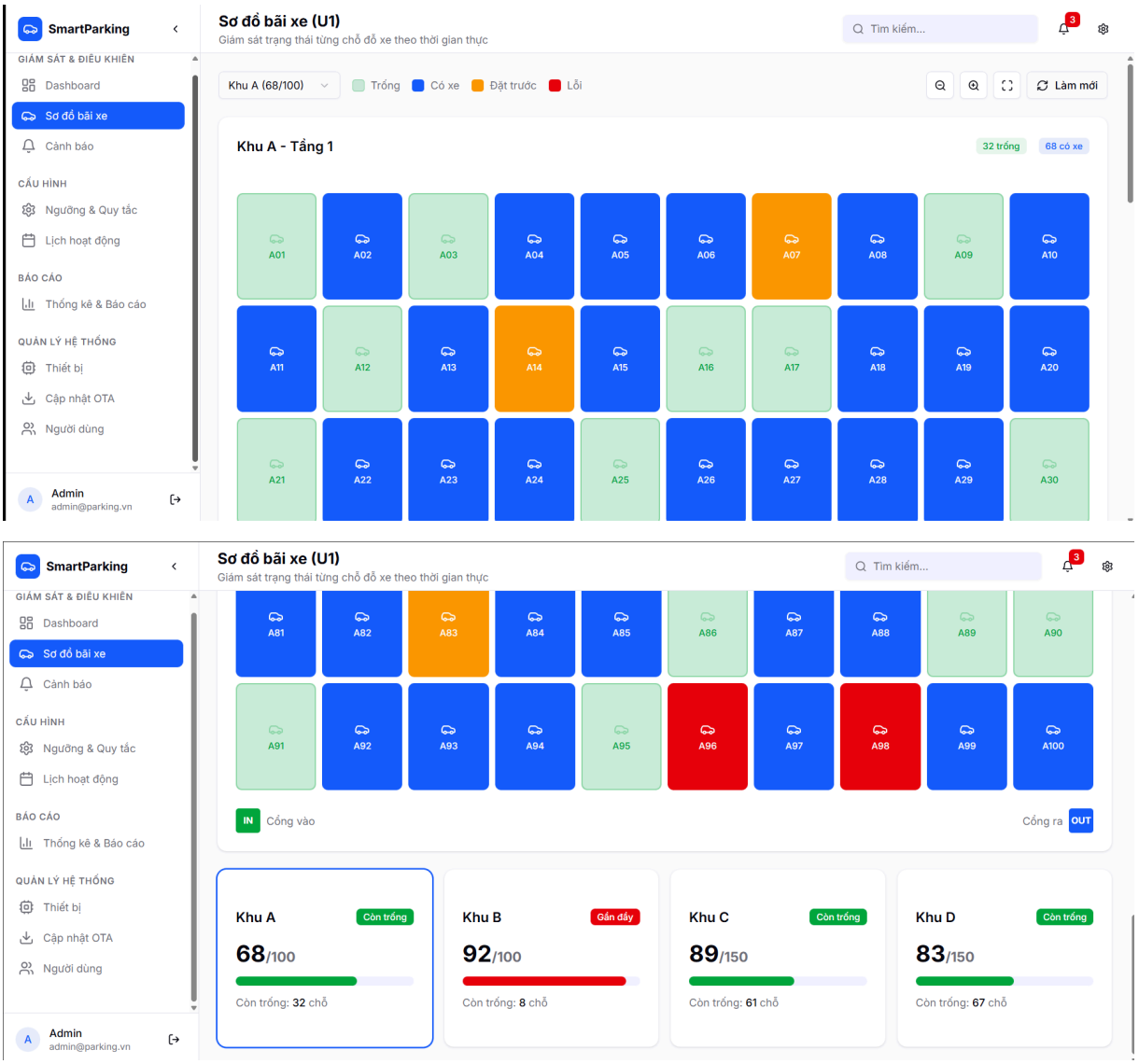
- Sơ đồ bãi xe: Hiển thị trực quan toàn bộ các ô đỗ xe theo dạng lưới. Mỗi ô có màu sắc thể hiện trạng thái: xanh lá (có xe), xanh dương (trống), đỏ (lỗi cảm biến). Thanh tiến độ phía trên cho biết tỷ lệ lấp đầy hiện tại.
- Bảng cảnh báo: Liệt kê các cảnh báo mới nhất như bãi sắp đầy, cảm biến lỗi, thiết bị mất kết nối. Mỗi cảnh báo có biểu tượng màu thể hiện mức độ nghiêm trọng.
- Điều khiển barrier: Cho phép nhân viên mở/đóng barrier cổng vào và cổng ra bằng thao tác thủ công. Hiển thị trạng thái hiện tại của từng barrier và có thể chuyển đổi giữa chế độ tự động và thủ công.
- Hoạt động gần đây: Ghi nhận các sự kiện xe vào/ra mới nhất kèm biển số và thời gian.



Hình 4.6 : Giao diện dashboard

c) Hiển thị trạng thái từng chỗ đỗ xe(trống/có xe)

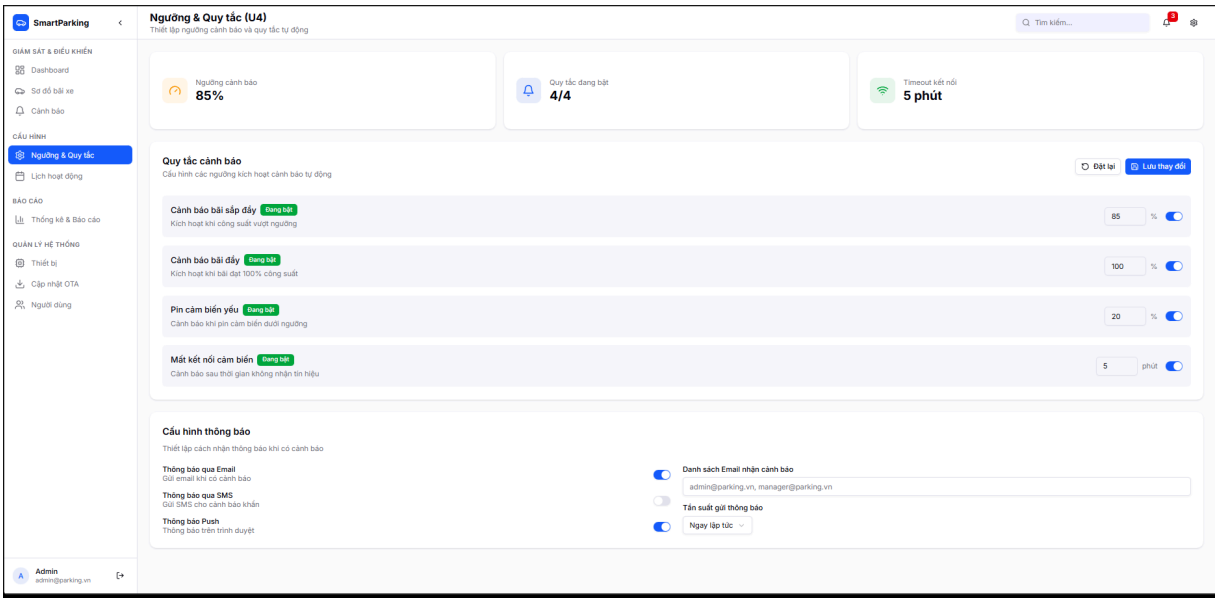
- Giao diện hiện chi tiết tất cả các bãi đỗ xe về từng vị trí và trạng thái hoạt động của vị trí đó với các trạng thái:
    + Đỏ: Lỗi (có thể từ lỗi cảm biến hay lỗi khác)
    + Xanh dương: Có xe đã đỗ
    + Xanh lá: Vị trí trống
    + Vàng: Được đặt trước
- Quản lý có thể xem chi tiết từng bãi đỗ xe và vị trí đỗ (Phần này chủ yếu dành cho user tài nhập truy cập để xem vị trí)



Hình 4.7: Giao diện sơ đồ bãi đỗ xe

d) Đặt ngưỡng cảnh báo

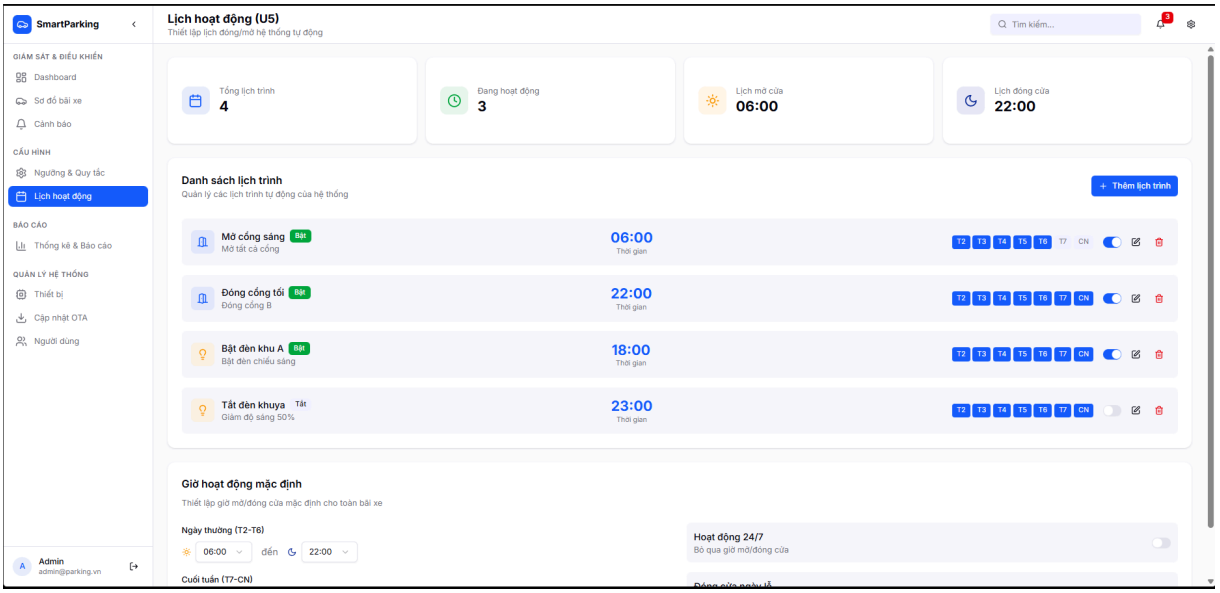
- Ngưỡng công suất: Thiết lập mức % lấp đầy để hệ thống gửi cảnh báo (ví dụ: 80% gửi cảnh báo thường, 95% gửi cảnh báo nghiêm trọng).
- Quy tắc thông báo: Bật/tắt các kênh nhận thông báo gồm email, SMS và thông báo đẩy. Có thể đặt thời gian chờ tối thiểu giữa các thông báo để tránh spam.
- Quy tắc thiết bị: Cấu hình thời gian chờ trước khi đánh dấu thiết bị offline, số lần thử kết nối lại và tùy chọn tự động khởi động lại khi gặp lỗi.
- Cấu hình thông báo: Thiết lập cách nhận thông báo khi có cảnh báo: sms, email….



Hình 4.8 : Giao diện ngưỡng cảnh báo và quy tắc

e) Thiết lập lịch đóng/mở hệ thống

- Danh sách lịch trình: Hiển thị các lịch đã cài đặt với thông tin ngày áp dụng, khung giờ và hành động thực hiện (mở/đóng barrier, bật/tắt đèn báo).
- Thêm lịch mới: Form cho phép tạo lịch trình mới bằng cách chọn các ngày trong tuần, giờ bắt đầu/kết thúc và loại hành động. Mỗi lịch có thể bật/tắt độc lập.



Hình 4.9 : Giao diện lịch hoạt động

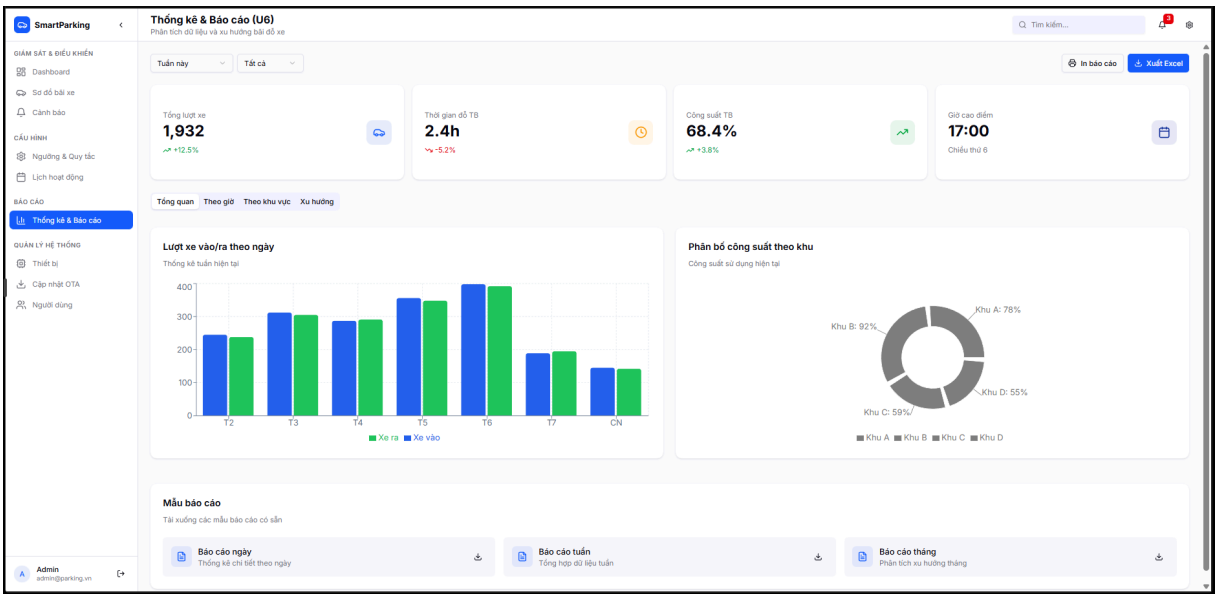
f) Thống kê báo cáo lượng xe, thời gian đỗ trung bình

- Bộ lọc thời gian: Chọn khoảng thời gian cần xem báo cáo (hôm nay, 7 ngày, 30 ngày hoặc tùy chỉnh).
- Biểu đồ lưu lượng: Thể hiện số lượng xe vào/ra theo từng khung giờ hoặc theo

ngày dưới dạng biểu đồ đường và biểu đồ cột.

- Biểu đồ công suất: Hiển thị tỷ lệ sử dụng bãi xe theo thời gian và phân bổ theo từng khu vực dưới dạng biểu đồ tròn.

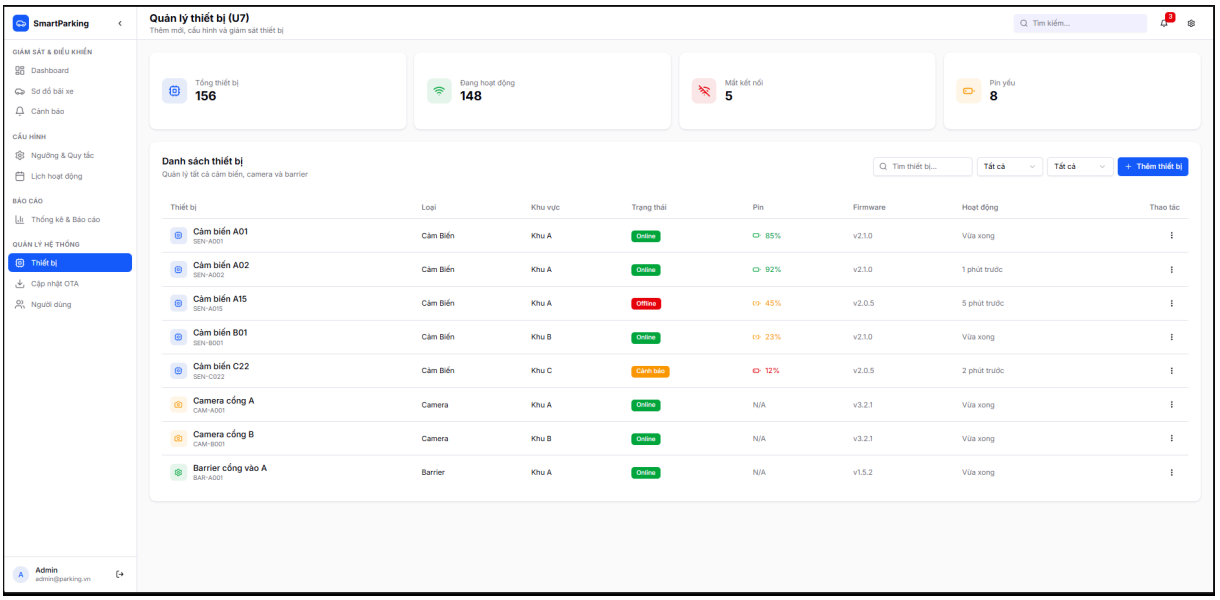
- Thống kê tổng hợp: Các chỉ số quan trọng gồm tổng lượt xe, thời gian đỗ trung bình, khung giờ cao điểm và công suất trung bình.
- Xuất báo cáo: Cho phép tải về báo cáo dạng PDF hoặc Excel.



Hình 4.10: Giao diện thống kê & báo cáo

g) Thêm mới, cấu hình cảm biến hoặc camera

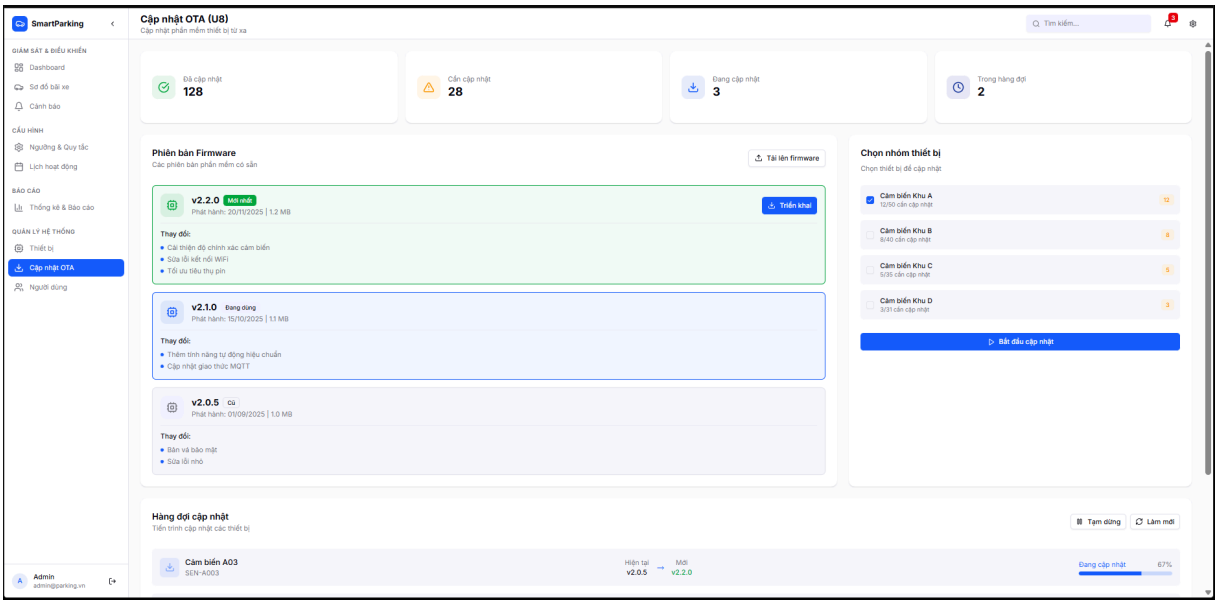
- Danh sách thiết bị: Bảng liệt kê tất cả thiết bị trong hệ thống (cảm biến, camera, barrier, đèn LED) với thông tin: mã thiết bị, tên, loại, vị trí lắp đặt, trạng thái hoạt động và phiên bản firmware.
- Thao tác quản lý: Các nút cho phép thêm thiết bị mới, chỉnh sửa cấu hình, khởi động lại từ xa hoặc xóa thiết bị khỏi hệ thống.
- Form thêm thiết bị: Nhập các thông tin cần thiết như loại thiết bị, tên, vị trí lắp đặt và địa chỉ IP/MAC.



Hình 4.11: Giao diện quản lý thiết bị

h) Cập nhật phần mềm thiết bị từ xa

- Danh sách firmware: Hiển thị các phiên bản firmware có sẵn với thông tin phiên bản, loại thiết bị áp dụng, ngày phát hành và mô tả thay đổi.
- Thực hiện cập nhật: Chọn thiết bị hoặc nhóm thiết bị cần cập nhật, có thể cập nhật ngay hoặc lên lịch cập nhật tự động.
- Theo dõi tiến độ: Hàng đợi cập nhật hiển thị tiến độ của từng thiết bị đang được cập nhật với trạng thái và thanh tiến độ.

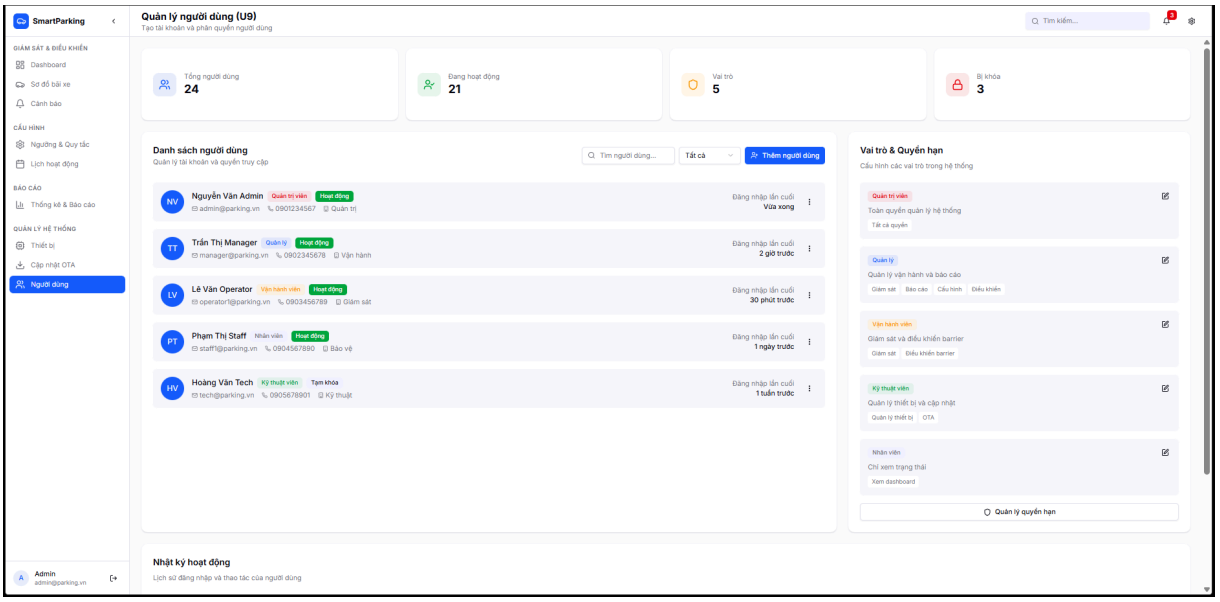


Hình 4.12: Giao diện cập nhật firmware

i) Tạo tài khoản, phân quyền người dùng

- Danh sách người dùng: Bảng hiển thị tất cả tài khoản trong hệ thống với thông tin họ tên, email, vai trò, trạng thái và thời gian đăng nhập gần nhất.

- Phân quyền vai trò: Hệ thống có 4 vai trò với quyền hạn khác nhau:
- Quản trị viên: Toàn quyền trên hệ thống
- Quản lý: Quản lý thiết bị, xem báo cáo, điều khiển barrier
- Nhân viên: Xem giám sát và điều khiển barrier thủ công
- Khách: Chỉ xem trạng thái bãi xe
- Thao tác quản lý: Thêm tài khoản mới, chỉnh sửa thông tin, đặt lại mật khẩu, khóa/mở khóa hoặc xóa tài khoản.



Hình 4.13: Giao diện quản lý người dùng

V. KẾT LUẬN

5.1 Kết quả đạt được

Về mặt lý thuyết: Qua việc tìm hiểu và nghiên cứu các thiết bị phần cứng như vi điều khiển ESP32, module camera OV2640, cảm biến siêu âm SRF05 và động cơ Servo SG90, nhóm đã hiểu rõ cơ chế hoạt động và cách thức tương tác, đồng bộ giữa các thành phần trong một hệ thống IoT. Các giao thức truyền thông tiên tiến như WebSocket và MQTT đã được nghiên cứu và ứng dụng thành công. Trong đó, MQTT giúp giải quyết bài toán truyền tải hình ảnh từ camera về máy chủ với độ trễ thấp và đảm bảo tính tin cậy cho các lệnh điều khiển đóng mở barie và cập nhật trạng thái cảm biến theo thời gian thực. Websocket đảm bảo thông tin được truyền liên tục từ server đến client

Về mặt thực tiễn: Hệ thống có thể phát hiện chính xác trạng thái có xe hoặc chỗ trống trong phạm vi từ 2cm đến 300cm nhờ cảm biến siêu âm, đồng thời thực hiện nhận diện biển số xe thông qua camera OV2640 với tốc độ xử lý nhanh, phù hợp với yêu cầu kiểm soát ra vào tại các bãi đỗ xe quy mô vừa và nhỏ. Hệ thống tự động hóa quy trình kiểm soát (từ nhận diện đến mở barie) đã giúp giảm thiểu sự can thiệp thủ công của nhân viên bảo vệ, tiết kiệm thời gian tìm chỗ cho tài xế và nâng cao hiệu quả quản lý đô thị.

5.2 Hạn chế

Hệ thống còn tồn tại những hạn chế sau:

- Hệ thống hiện tại sử dụng động cơ Servo SG90 có mô-men xoắn nhỏ, chỉ phù hợp với các mô hình barie nhẹ mô phỏng, chưa đủ khả năng vận hành các thanh chắn thực tế có trọng lượng lớn.
- Khả năng truyền tải hình ảnh và dữ liệu phụ thuộc hoàn toàn vào mạng Wi-Fi. Nếu mất kết nối hoặc mạng không ổn định, tính năng giám sát hình ảnh (camera) và điều khiển từ xa sẽ bị gián đoạn.
- Camera OV2640 có hạn chế về khả năng chụp ảnh trong điều kiện thiếu sáng

hoặc khi xe di chuyển với tốc độ cao, có thể ảnh hưởng đến độ chính xác của thuật toán nhận diện biển số.

- Cảm biến siêu âm SRF05 hoạt động dựa trên sóng âm nên có thể gặp sai số trong môi trường có nhiều tạp âm hoặc bề mặt phản xạ của xe quá phức tạp.

5.3 Hướng phát triển trong tương lai

Về hệ thống:

- Nghiên cứu và tích hợp các giải pháp thanh toán không tiền mặt (QR Code, ví điện tử) để trừ tiền trực tiếp khi xe ra khỏi bãi, hướng tới mô hình bãi đỗ xe thông minh hoàn toàn tự động.
- Nâng cấp phần cứng camera (ví dụ sử dụng các dòng Global Shutter hoặc cảm biến hỗ trợ ngược sáng tốt hơn) để đảm bảo khả năng nhận diện biển số chính xác 24/7, bất kể ngày đêm hay thời tiết.
- Hỗ trợ thêm các giao thức truyền thông tầm xa và tiết kiệm năng lượng như LoRaWAN để hệ thống có thể triển khai tại các bãi đỗ xe rộng lớn hoặc các tầng hầm nơi sóng Wi-Fi yếu.
- Phát triển tính năng dẫn đường nội bộ: Tích hợp hệ thống đèn LED điều hướng hoặc bản đồ trên ứng dụng di động để dẫn tài xế đến vị trí trống gần nhất, thay vì chỉ hiển thị số lượng chỗ trống.
- Mở rộng hệ thống để tích hợp thêm các cảm biến môi trường (như cảm biến khí thải CO, cảm biến cháy) để cảnh báo an toàn trong các hầm gửi xe kín.

Về bảo mật:

- Sử dụng các chính sách phân quyền theo vai trò (RBAC - Role-Based Access Control) chặt chẽ, đảm bảo nhân viên, quản lý và khách hàng chỉ truy cập được những thông tin và chức năng phù hợp với quyền hạn.
- Đảm bảo luồng dữ liệu hình ảnh và các lệnh điều khiển MQTT đều được mã hóa bằng giao thức MQTTS (Secure MQTT) hoặc TLS/SSL, giúp ngăn chặn việc xem trộm camera trái phép hoặc giả mạo lệnh điều khiển barie từ bên ngoài.

TÀI LIỆU THAM KHẢO

1. *Internet of things - Wikipedia*
2. *Random Nerd Tutorials | Learn ESP32, ESP8266, Arduino, and Raspberry Pi*
3. *Technical Documents | Espressif Systems*
4. *250+ ESP32 Projects, Tutorials and Guides with Arduino IDE | Random Nerd Tutorials*
5. *MQTT - The Standard for IoT Messaging*
6. *MQTT Explorer | An all-round MQTT client that provides a structured topic overview*