HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

**KHOA CÔNG NGHỆ THÔNG TIN I**

**⎯⎯⎯⎯⎯⎯⎯⎯⎯**



**BÁO CÁO GIỮA KÌ**

**MÔN HỌC: IOT VÀ ỨNG DỤNG**

**Nhóm lớp học: Nhóm 05**

**Nhóm bài tập lớn: Nhóm 13**

**Đề tài: Xe giám sát thông minh**

**Danh sách sinh viên:**

**Họ và tên Mã sinh viên**

**Nguyễn Mạnh Cường B22DCCN100**

**Trần Đức Phương B22DCCN640**

**Đỗ Thành Đạt B21DCCN184**

**Bùi Văn Hiến B22DCCN292**

**Giảng viên giảng dạy: Kim Ngọc Bách**

**HÀ NỘI – 2025**

# MỤC LỤC

[**MỤC LỤC 2**](#_heading=h.upwf2njdqi65)

[**Lời cảm ơn 3**](#_heading=h.hu7lka78nrob)

[**Tóm tắt 5**](#_heading=h.jc0e2yubah7)

[**1. Giới thiệu chung 6**](#_heading=h.790g6zec2b44)

[1.1 Lý do chọn đề tài 6](#_heading=h.fwgc5bgh0t2o)

[1.2 Tổng quan về dự án 6](#_heading=h.d8gk45fmtg5l)

[1.3 Mục đích dự án 6](#_heading=h.l5m1q1ri5s9u)

[1.4 Đối tượng và phạm vi nghiên cứu 7](#_heading=h.67yeikc5lmno)

[1.4.1 Đối tượng nghiên cứu 7](#_heading=h.r5slmq5me1s5)

[1.4.2 Phạm vi nghiên cứu 8](#_heading=h.et39jf37mqrn)

[1.5 Phương pháp nghiên cứu 9](#_heading=h.9qi2ckkjn9s)

[1.5.1 Phương pháp nghiên cứu lý thuyết 9](#_heading=h.dfxeviuqou8u)

[1.5.2 Phương pháp thực nghiệm 9](#_heading=h.xdhn7n5dr3yz)

[1.5.3 Phương pháp kiểm thử 9](#_heading=h.pyzjs3mxzbop)

[1.6 Công nghệ và thiết bị cần thiết cho dự án 9](#_heading=h.phpkmk3m96yd)

[**2. Nền tảng lý thuyết 11**](#_heading=h.fd124wxso1s)

[2.1 Module Esp32 11](#_heading=h.s05c9zxr8kbe)

[2.1.1 Giới thiệu 11](#_heading=h.m24vj0o1qs2x)

[2.1.2 Thông số kỹ thuật của ESP32 11](#_heading=h.ej78b4b8c2sv)

[2.2 Module ESP32-CAM 12](#_heading=h.s616sds9sfpq)

[2.2.1 Giới thiệu chung 12](#_heading=h.ugkehhntxy7r)

[2.2.2 Thông số kĩ thuật 13](#_heading=h.foesnjn5jv7a)

[2.3 Cảm biến khoảng cách siêu âm HC-SR04 13](#_heading=h.dckq19ivf8bp)

[2.3.1 Bản chất vật lý của phép đo 14](#_heading=h.8qmkwomybzdg)

[2.3.2 Công thức tính toán và Sai số 15](#_heading=h.tfrjrpukaszk)

[2.4 Mạch cầu H L298N (Driver Động cơ) 15](#_heading=h.ek3wb74zu43)

[2.4.1 Cấu trúc mạch cầu H (H-Bridge) 16](#_heading=h.6n8fwwn00ptm)

[2.4.2 Thông số kĩ thuật 17](#_heading=h.7lyjrbnee8v)

[2.5 Giao thức truyền thông: HTTP vs WebSocket 17](#_heading=h.xrbkfjrjnziz)

[2.5.1 Nhược điểm của HTTP trong điều khiển Robot 17](#_heading=h.y74q4tzgg8jb)

[2.5.2 Ưu điểm của WebSocket 17](#_heading=h.neaiz010idt7)

[2.6 Nền tảng Server Node.js 18](#_heading=h.2kj3evbyzflu)

[2.6.1 Cơ chế Event Loop và Non-blocking I/O 18](#_heading=h.8k5iifnfszso)

[2.7 Trí tuệ nhân tạo và Thị giác máy tính (Computer Vision) 18](#_heading=h.4n2e7vvs7r4n)

[2.7.1 Mạng nơ-ron tích chập (CNN) và Pose Estimation 18](#_heading=h.a8vawzg1qfxr)

[2.7.2 Nguyên lý phát hiện hành vi 18](#_heading=h.270dwhd9wqo)

[2.7.3 Phát hiện đường tường (Wall Line Detection) 19](#_heading=h.e3rm5oock1xu)

[2.7.4 Phát hiện chuyển động (Motion Detection) 19](#_heading=h.88noh23wi35q)

[2.7.5 Tích hợp với Person Detector & Cơ chế “vẫy tay miễn nhiễm” 20](#_heading=h.37ghig62p1yw)

[**3. Phân tích yêu cầu chức năng 22**](#_heading=h.t2aah9e2aaqq)

[3.1 Chức năng đăng nhập vào hệ thống 22](#_heading=h.gagiqqeli2q2)

[3.2 Chức năng điều khiển xe 22](#_heading=h.vu9zljpjj93k)

[3.3 Chức năng xe tự động vận hành 22](#_heading=h.558hkmxnbuza)

[3.4 Chức năng giám sát hình ảnh 23](#_heading=h.5t3ka8wzai8d)

[3.5 Chức năng chuyển đổi trạng thái 23](#_heading=h.enqdr7bzuucs)

[3.6 Chức năng AI Giám sát An ninh 23](#_heading=h.tyu9nt9mw4kz)

[3.7 Chức năng xem lịch sử cảnh báo 24](#_heading=h.10q4usflv1e3)

[**4. Thiết kế hệ thống 25**](#_heading=h.sgaiajreyvl5)

[4.1 Đặc tả mô hình miền 25](#_heading=h.mvtf6zfnm535)

[4.2 Đặc tả các thành phần chức năng và hoạt động 28](#_heading=h.tslkek9sjfs2)

[4.2.1 Thành phần chức năng 29](#_heading=h.kphipjl0zbbq)

[4.2.2 Thành phần hoạt động 29](#_heading=h.8zopjh3hfu8z)

[4.3 Đặc tả luồng hoạt động 30](#_heading=h.pt2rok81ln5y)

[4.3.1 Thiết bị cảm biến và vi xử lí 30](#_heading=h.62wawqc2qk3w)

[4.3.2 Website quản lý 34](#_heading=h.z2dk3yrf4ri8)

[4.3.3 Hệ thống AI Giám sát An ninh 39](#_heading=h.934pa6tp2e2l)

[4.4 Tích hợp thiết bị 43](#_heading=h.oicm9tl8h2yu)

[4.4.1. Khối Động Cơ & Driver L298N (Nối với ESP32 Chính) 43](#_heading=h.azq4xlfanea9)

[4.4.2. Khối Camera & Servo (Giao tiếp giữa 2 ESP32) 45](#_heading=h.jlk10n2y3e08)

[4.5 Phát triển ứng dụng 45](#_heading=h.6m27wll3cfdw)

[**5. KẾT LUẬN 51**](#_heading=h.9u4mlk5tnykf)

[5.1. Kết quả đạt được 51](#_heading=h.26k44u1o8gbv)

[5.1.1. Về mặt lý thuyết 51](#_heading=h.1czvk46y1zd7)

[5.1.2. Về mặt thực tiễn 51](#_heading=h.z210ehssaxrh)

[5.2. Hạn chế 52](#_heading=h.hciqpsz3i5uu)

[5.3. Hướng phát triển tương lai 52](#_heading=h.f9iv2mj924oa)

[5.3.1. Về hệ thống 52](#_heading=h.dd9gjtaucz70)

[5.3.2. Về bảo mật 53](#_heading=h.ijz9vsx947nc)

[**Bảng phân công nhiệm vụ 54**](#_heading=h.utmei3jfdxe9)

# Lời cảm ơn

Qua chặng đường thực hiện đề tài "Xe Giám Sát Thông Minh Dựa Trên Công Nghệ IoT và Trí Tuệ Nhân Tạo", nhóm chúng em đã có cơ hội vận dụng kiến thức lý thuyết vào thực hành, đồng thời trải nghiệm những thử thách thực tế khi xây dựng một hệ thống IoT hoàn chỉnh.​

Thành công của dự án này gắn liền với sự đóng góp và hỗ trợ từ nhiều phía. Nhóm xin bày tỏ lòng biết ơn chân thành đến giảng viên Kim Ngọc Bách - người đã không chỉ truyền đạt kiến thức mà còn kiên nhẫn định hướng, góp ý xuyên suốt quá trình nghiên cứu. Những kinh nghiệm thực tế mà thầy/cô chia sẻ về lập trình nhúng ESP32, xử lý video streaming và ứng dụng AI đã giúp nhóm vượt qua nhiều khó khăn kỹ thuật.

Chúng em cũng xin cảm ơn Học viện Công nghệ Bưu chính Viễn thông đã tạo môi trường học tập thuận lợi và trang bị nền tảng kiến thức vững chắc về Internet of Things, Computer Vision và kiến trúc hệ thống phân tán - những yếu tố then chốt để hoàn thành đề tài này.​

Với sự nỗ lực tốt nhất, báo cáo vẫn không thể tránh khỏi những hạn chế nhất định. Nhóm rất mong nhận được ý kiến đóng góp từ quý thầy cô để có thể tiếp tục hoàn thiện kiến thức và kỹ năng.

Trân trọng cảm ơn!

# Tóm tắt

**Xe Giám Sát Thông Minh Dựa Trên Công Nghệ IoT và Trí Tuệ Nhân Tạo**

Xe giám sát thông minh là một giải pháp IoT kết hợp AI phục vụ điều khiển từ xa và giám sát môi trường thời gian thực. Hệ thống sử dụng vi điều khiển ESP32 làm trung tâm điều khiển, module ESP32-CAM để truyền hình ảnh video trực tiếp, và cảm biến siêu âm HC-SR04 để phát hiện vật cản trong khoảng cách 30cm. Người dùng có thể điều khiển xe robot 4 bánh từ xa thông qua trình duyệt web trên điện thoại hoặc máy tính mà không bị giới hạn bởi khoảng cách sóng RF truyền thống.​

Hệ thống hoạt động theo kiến trúc Client-Server với giao thức WebSocket, đảm bảo độ trễ thấp dưới 200ms cho trải nghiệm điều khiển mượt mà. Camera ESP32-CAM livestream video liên tục cho phép người dùng quan sát môi trường, trong khi cảm biến siêu âm quét vật cản song song và tự động can thiệp quyền điều khiển khi phát hiện nguy hiểm ở chế độ tự động. Hệ thống hỗ trợ hai chế độ hoạt động: điều khiển thủ công từ giao diện web và vận hành tự động với cơ chế tránh vật cản thông minh.​

Điểm nổi bật của dự án là tích hợp công nghệ Trí tuệ nhân tạo trên Server để phân tích luồng video thời gian thực, có khả năng tự động nhận diện hai hành vi nguy hiểm: người vượt rào (Climbing Detection) và người bị ngã (Fall Detection) thông qua mạng nơ-ron tích chập CNN và Pose Estimation. Khi phát hiện sự cố, hệ thống tự động lưu bằng chứng hình ảnh kèm theo thời gian vào cơ sở dữ liệu để người dùng tra cứu lịch sử cảnh báo.​

Hệ thống giúp theo dõi và điều khiển xe robot theo thời gian thực, hỗ trợ cảnh báo sớm khi phát hiện hành vi bất thường, nâng cao hiệu quả quản lý và giám sát an ninh. Dự án chứng minh giá trị ứng dụng thực tiễn cao trong lĩnh vực giám sát an ninh, giám sát công trình và quản lý nhà thông minh, góp phần tối ưu hóa nhân lực và tiết kiệm chi phí vận hành.

# 

# 1. Giới thiệu chung

## 1.1 Lý do chọn đề tài

Trong bối cảnh công nghệ IoT (Internet of Things) phát triển mạnh mẽ, nhu cầu về các thiết bị tự hành có khả năng giám sát từ xa ngày càng tăng cao. Các robot truyền thống thường bị giới hạn bởi khoảng cách điều khiển (sóng RF) hoặc thiếu tính năng quan sát trực quan. Việc kết hợp khả năng truyền hình ảnh thời gian thực (Livestream) với cơ chế tự động bảo vệ (tránh vật cản) là một bước tiến quan trọng. Đề tài này giải quyết bài toán điều khiển robot linh hoạt thông qua mạng Wi-Fi , tận dụng các thiết bị di động thông minh làm bộ điều khiển, đồng thời đảm bảo an toàn cho thiết bị nhờ cảm biến đo khoảng cách.

## 1.2 Tổng quan về dự án

Dự án tập trung xây dựng một mô hình xe robot 4 bánh (chassis 4WD) sử dụng vi điều khiển trung tâm là ESP32. Hệ thống hoạt động theo mô hình Client-Server:

* Client (Robot): Thu thập hình ảnh từ Camera, đo khoảng cách từ cảm biến HC-SR04 và điều khiển động cơ với hai chế độ.
* Client (Người dùng): Sử dụng trình duyệt Web trên điện thoại hoặc máy tính để xem hình ảnh và gửi lệnh điều khiển. Điểm nổi bật của dự án là khả năng xử lý song song: vừa truyền hình ảnh mượt mà, vừa liên tục quét vật cản để can thiệp quyền điều khiển khi gặp nguy hiểm.
* Server: trung gian xử lí giao tiếp thời gian thực, phân loại và định tuyến dữ liệu giữa 2 loại client

Ngoài ra, hệ thống còn tích hợp công nghệ Trí tuệ nhân tạo (AI) trên Server để phân tích luồng video thời gian thực. Hệ thống có khả năng tự động nhận diện hai hành vi nguy hiểm: Vượt rào trái phép và Người bị ngã. Khi phát hiện sự cố, hệ thống sẽ lưu lại bằng chứng hình ảnh và thời gian vào lịch sử để người dùng tra cứu.

## 1.3 Mục đích dự án

Hệ thống xe giám sát thông minh hỗ trợ người dùng thao tác với xe từ xa, do đó người dùng có thể xử lí tình huống một cách linh hoạt và chính xác nhất như phát hiện người vượt rào, người bị ngã, xem xét các chỗ khuất khó có thể quan sát bằng mắt người,…

Hệ thống tự động thu thập dữ liệu và gửi cho người dùng, tiết kiệm thời gian của người dùng hơn và giảm thiểu sai xót trong quá trinh đo

Hệ thống hỗ trợ phân tích dữ liệu dài hạn, dữ liệu được lưu trữ có thể sử dụng để người dùng xem lại lịch sử phát hiện của xe

Hệ thống nhờ vào cảm biến cảm biến siêu âm để tránh và cảnh báo các vật cản ở phía trước, cùng với hệ thống ai cho phép phát hiện người vượt rào, người bị ngã, giúp người dùng có thể tiếp cận được những thông tin một cách nhanh nhất.

Hệ thống có thể phục vụ một cách tối ưu trong lĩnh vực giám sát như giám sát nhà ở, giám sát công trường,…

Trong việc giám sát công trường, hệ thống giúp người dùng tối ưu hóa được nhân lực và thời gian làm việc, giúp giảm thiểu chi phí về nhân lực và thời gian

## 1.4 Đối tượng và phạm vi nghiên cứu

### 1.4.1 Đối tượng nghiên cứu

Đề tài tập trung nghiên cứu sâu vào các thành phần phần cứng và công nghệ phần mềm cốt lõi để xây dựng một hệ thống IoT hoàn chỉnh theo mô hình tập trung Cụ thể bao gồm:

* Về phần cứng:
  + Vi điều khiển và Xử lý ảnh: Tập trung nghiên cứu module ESP32 và ESP32-CAM, tận dụng sức mạnh của chip ESP32 để thực hiện song song hai tác vụ nặng: thu thập/nén hình ảnh từ Camera OV2640 và điều khiển các chân GPIO.
  + Cảm biến đo lường: Nghiên cứu nguyên lý hoạt động và đặc tính kỹ thuật của cảm biến siêu âm HC-SR04 để ứng dụng vào bài toán đo khoảng cách và phát hiện vật cản trong môi trường thực tế.
  + Cơ cấu chấp hành: Nghiên cứu modul L298N để điều khiển tốc độ và chiều quay của động cơ DC, từ đó kiểm soát hướng di chuyển của Robot (tiến, lùi, rẽ trái, rẽ phải).
* Về phần mềm và Giao thức (Software & Protocols):
  + Kiến trúc Client-Server: Nghiên cứu cách xây dựng một Server trung gian sử dụng nền tảng Node.js, đóng vai trò điều phối
  + Giao thức truyền tin: Tập trung phân tích và ứng dụng giao thức WebSocket nhằm giải quyết bài toán độ trễ trong việc truyền tải video stream và tín hiệu điều khiển thời gian thực (Real-time).
  + Giao diện người dùng: Nghiên cứu xây dựng Web Dashboard tương tác cao (sử dụng JavaScript), cho phép hiển thị luồng video và gửi lệnh điều khiển mượt mà trên trình duyệt.

### 1.4.2 Phạm vi nghiên cứu

Để đảm bảo tính khả thi và tập trung vào các mục tiêu cốt lõi của đề tài trong thời gian cho phép, phạm vi nghiên cứu được xác định như sau:

* Phạm vi về chức năng hệ thống:
  + Đề tài tập trung vào việc thiết kế và chế tạo một mô hình xe Robot 4 bánh (4WD Chassis) thực tế, không dừng lại ở mức mô phỏng.
  + Hệ thống thực hiện trọn vẹn quy trình khép kín: Thu thập dữ liệu (Ảnh/Khoảng cách) -> Truyền tải qua Internet -> Xử lý tại Server -> Hiển thị cho người dùng và ngược lại (Gửi lệnh điều khiển).
  + Tính năng tự động hóa chỉ tập trung vào bài toán tránh vật cản cơ bản: Tự động dừng và ngắt động cơ khi vật cản nằm trong vùng nguy hiểm (< 30cm) khi ở chế độ điều khiển cơ bản, còn ở chế độ tự động sẽ tự động tìm hướng đi phù hợp nhất.
* Phạm vi về môi trường hoạt động:
  + Robot được thiết kế để hoạt động trong môi trường trong nhà, sân vườn hoặc bề mặt phẳng, ít gồ ghề.
  + Điều kiện ánh sáng phải đủ tốt để Camera OV2640 có thể ghi nhận hình ảnh rõ nét (do hạn chế về khả năng quay đêm của esp32-cam).
  + Hệ thống hoạt động dựa trên nền tảng mạng Wi-Fi 2.4GHz. Phạm vi điều khiển phụ thuộc vào vùng phủ sóng của Router Wi-Fi hoặc khả năng kết nối 4G/5G của thiết bị Server trung gian.
* Phạm vi về công nghệ truyền tải:
  + Dữ liệu hình ảnh được truyền dưới dạng các khung hình nén JPEG liên tiếp (MJPEG Stream) thay vì các chuẩn nén video phức tạp (như H.264/H.265) để phù hợp với năng lực xử lý giới hạn của vi điều khiển ESP32.
  + Tốc độ phản hồi của hệ thống được tối ưu hóa để đạt mức trễ dưới 200ms trong điều kiện mạng ổn định, đảm bảo trải nghiệm điều khiển thời gian thực.

## 1.5 Phương pháp nghiên cứu

Để thực hiện đề tài, nhóm áp dụng ba phương pháp nghiên cứu chính:

### 1.5.1 Phương pháp nghiên cứu lý thuyết

* Nghiên cứu tài liệu: Tìm hiểu cấu trúc phần cứng, sơ đồ chân và thư viện lập trình của module ESP32-CAM, cảm biến siêu âm HC-SR04, module L298N, module ESP32
* Nghiên cứu công nghệ: Phân tích mô hình kiến trúc Client-Server và giao thức WebSocket để giải quyết bài toán truyền tải hình ảnh và điều khiển thời gian thực (Real-time).

### 1.5.2 Phương pháp thực nghiệm

* Thiết kế phần cứng: Tính toán công suất, thiết kế sơ đồ nguyên lý và lắp ráp hoàn chỉnh mô hình Robot với mạch cầu H L298N và nguồn pin Li-ion.
* Phát triển phần mềm: Xây dựng hệ thống điều khiển đồng bộ gồm: Server trung gian (Node.js) xử lý định tuyến, firmware trên Robot để đọc cảm biến/chụp ảnh, và web Dashboard cho người dùng.

### 1.5.3 Phương pháp kiểm thử

* Tiến hành chạy thử nghiệm để kiểm tra độ trễ (latency) của thao tác điều khiển và độ ổn định của luồng video.
* Kiểm chứng độ chính xác của thuật toán an toàn: đảm bảo Robot tự động ngắt động cơ tức thì khi cảm biến phát hiện vật cản trong phạm vi 30cm nếu ở chế độ điều khi và tự động tránh vật cản khi ở chế độ tự động.

## 1.6 Công nghệ và thiết bị cần thiết cho dự án

Phần cứng:

* Vi điều khiển trung tâm (ESP32): Đóng vai trò bộ não xử lý chính, trực tiếp điều khiển các thiết bị ngoại vi và truyền hình ảnh.
* ESP32- Cam: chịu trách nhiệm thu thập dữ liệu hình ảnh
* Cảm biến siêu âm (HC-SR04): Được ESP32 kích hoạt và đọc tín hiệu để đo khoảng cách vật cản.
* Module điều khiển động cơ (L298N): Nhận tín hiệu điều khiển từ ESP32 để vận hành động cơ DC .
* Khung xe robot (Chassis): Bộ khung 4 bánh (4WD) kèm 4 động cơ giảm tốc DC.
* Nguồn điện: Pin Li-ion 18650 để cấp nguồn ổn định cho hệ thống.

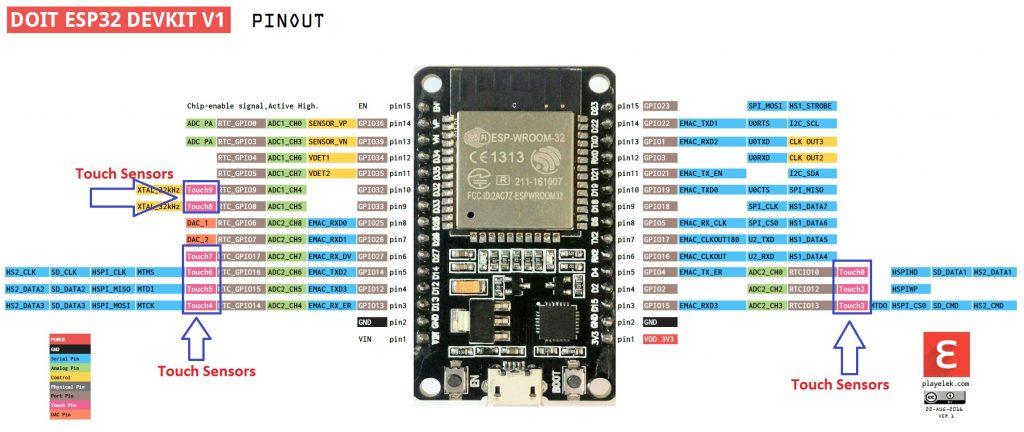
Phần mềm:

* Server trung gian: Node.js .
* Giao thức truyền thông: WebSocket .
* Công cụ lập trình Firmware: Arduino IDE .
* Giao diện người dùng: HTML5, CSS3, JavaScript .

# 2. Nền tảng lý thuyết

## 2.1 Module Esp32

### 2.1.1 Giới thiệu



ESP32 là một hệ thống vi điều khiển trên chip (SoC) giá rẻ của Espressif Systems, nhà phát triển của ESP8266 SoC. Nó là sự kế thừa của SoC ESP8266 và có cả hai biến thể lõi đơn và lõi kép của bộ vi xử lý 32-bit Xtensa LX6 của Tensilica với Wi-Fi và Bluetooth tích hợp.

Điểm tốt về ESP32, giống như ESP8266 là các thành phần RF tích hợp của nó như bộ khuếch đại công suất, bộ khuếch đại nhận tiếng ồn thấp, công tắc ăng-ten, bộ lọc và Balun RF. Điều này làm cho việc thiết kế phần cứng xung quanh ESP32 rất dễ dàng vì bạn cần rất ít thành phần bên ngoài.

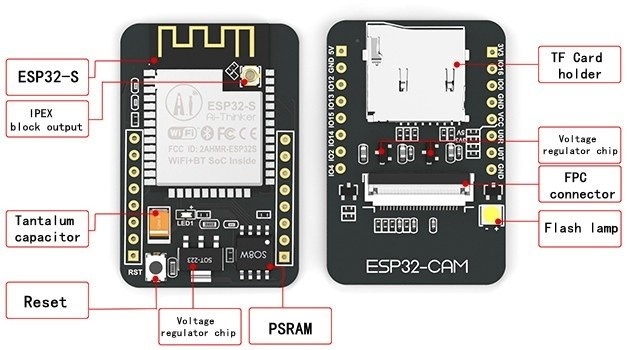
Một điều quan trọng khác cần biết về ESP32 là nó được sản xuất bằng công nghệ 40 nm công suất cực thấp của TSMC. Vì vậy, việc thiết kế các ứng dụng hoạt động bằng pin như thiết bị đeo, thiết bị âm thanh, đồng hồ thông minh, ..., sử dụng ESP32 sẽ rất dễ dàng.

### 2.1.2 Thông số kỹ thuật của ESP32

ESP32 có nhiều tính năng hơn ESP8266 và rất khó để đưa tất cả các thông số kỹ thuật vào bài này. Dưới đây là một số thông số quan trọng:

* Bộ vi xử lý LX6 32-bit lõi đơn hoặc lõi kép với xung nhịp lên đến 240 MHz.
* 520 KB SRAM, 448 KB ROM và 16 KB SRAM RTC.
* Hỗ trợ kết nối Wi-Fi 802.11 b / g / n với tốc độ lên đến 150 Mbps.
* Hỗ trợ cho cả thông số kỹ thuật Bluetooth v4.2 và BLE cổ điển.
* 34 GPIO có thể lập trình.
* 18 kênh SAR ADC 12 bit và 2 kênh DAC 8 bit
* Kết nối nối tiếp bao gồm 4 x SPI, 2 x I2C, 2 x I2S, 3 x UART.
* Ethernet MAC cho giao tiếp mạng LAN vật lý (yêu cầu PHY bên ngoài).
* 1 bộ điều khiển host cho SD / SDIO / MMC và 1 bộ điều khiển slave cho SDIO / SPI.
* Động cơ PWM và 16 kênh LED PWM.
* Khởi động an toàn và mã hóa Flash.
* Tăng tốc phần cứng mật mã cho AES, Hash (SHA-2), RSA, ECC và RNG.

## 2.2 Module ESP32-CAM



### **2.2.1 Giới thiệu chung**

ESP32-CAM có một camera kích thước nhỏ, rất cạnh tranh trong ngành, giống như mô-đun chính, mô-đun này có thể được xử lý công việc độc lập, module có kích thước nhỏ gon chỉ 40 x 27 x12 mm, dòng nghỉ chỉ 6mA.

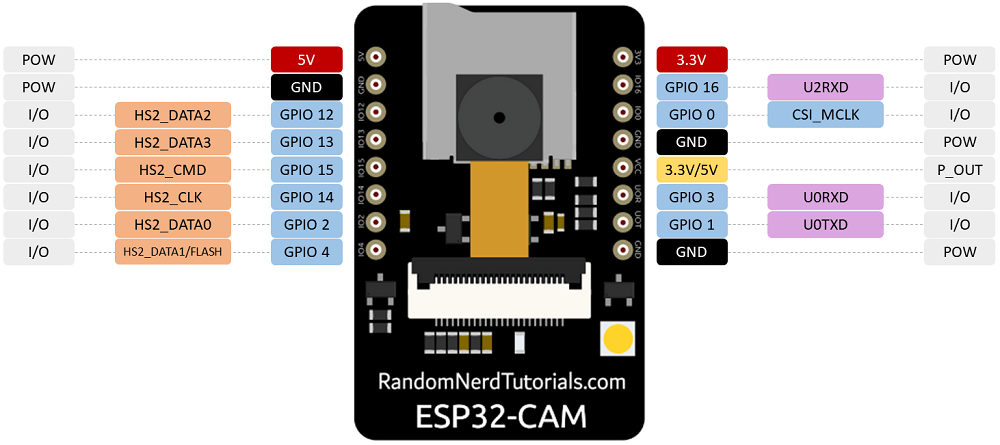
ESP-32CAM có thể được sử dụng rộng rãi trong các ứng dụng IoT khác nhau, thích hợp cho thiết bị thông minh gia đình, điều khiển không dây công nghiệp, giám sát không dây kiểm soát, nhận dạng không dây QR, tín hiệu hệ thống định vị không dây…Nó là một giải pháp lý tưởng cho các ứng dụng IoT

Mạch thu phát Wifi BLE ESP32 này là mạch chính hãng AI – Thinker có chất lượng độ ổn định và độ bền rất cao, sử dụng camera OV2640 chất lượng cao hình ảnh sắc nét, không nhiễu sọc, không xảy ra tình trạng treo khi hoạt động do sử dụng ic cấp nguồn chất lượng cao.

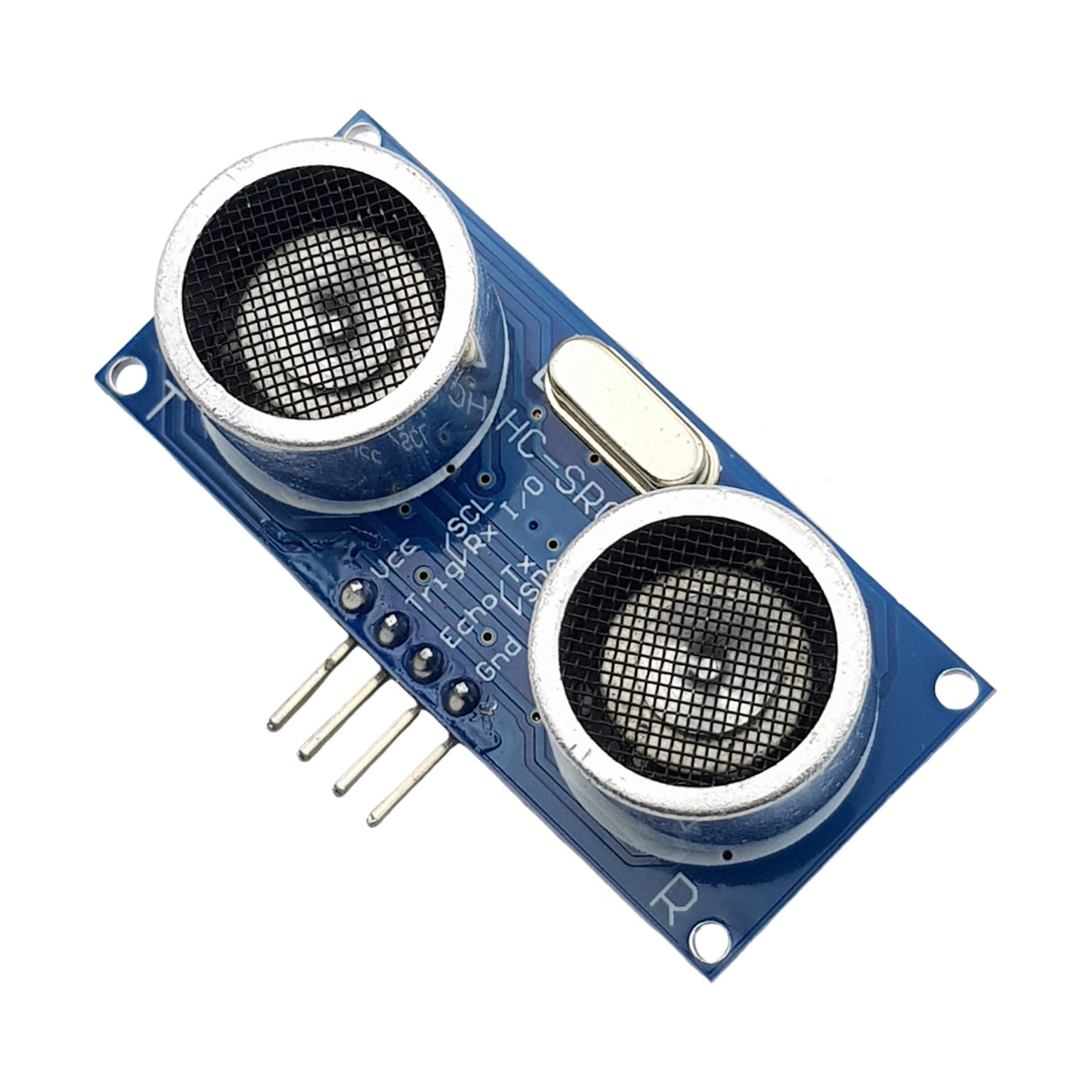
Mạch thu phát Wifi BLE ESP32-CAM Ai-Thinker này có thể sử dụng Arduino IDE để biên dịch và viết code, được hỗ trợ mạnh mẽ từ cộng đồng.

### **2.2.2 Thông số kĩ thuật**

* IC chính: ESP32-S (AI-Thinker)
* Mô-đun Wi-Fi BT SoC 802.11 b/g/n/e/i
* CPU 32-bit công suất thấp, cũng có thể phục vụ bộ xử lý ứng dụng
* Tốc độ đồng hồ lên đến 160MHz, sức mạnh tính toán lên đến 600 DMIPS
* Tích hợp 520 KB SRAM, 4MPSRAM bên ngoài
* Dải tần số: 1421 ~ 2484 Mhz
* Bluetooth: 4.2 BR/EDR BLE
* Hỗ trợ UART / SPI / I2C / PWM / ADC / DAC
* Hỗ trợ máy ảnh OV2640 và OV7670, đèn flash tích hợp
* Hỗ trợ tải lên WiFI hình ảnh
* Hỗ trợ thẻ TF
* Hỗ trợ nhiều chế độ ngủ
* Nhúng Lwip và FreeRTOS
* Hỗ trợ chế độ hoạt động STA / AP / STA + AP
* Hỗ trợ cấu hình thông minh / công nghệ AirKiss
* Hỗ trợ nâng cấp cục bộ và từ xa cho cổng nối tiếp (FOTA)



## 2.3 Cảm biến khoảng cách siêu âm HC-SR04



### **2.3.1 Bản chất vật lý của phép đo**

Cảm biến HC-SR04 hoạt động như một hệ thống Sonar thu nhỏ. Nó bao gồm một bộ phát sóng siêu âm (Transmitter) và một bộ thu (Receiver) hoạt động ở tần số 40 kHz - ngưỡng tần số mà tai người không nghe thấy được nhưng lan truyền tốt trong không khí.

* **Tín hiệu Trigger:** Để bắt đầu đo, vi điều khiển phải gửi một xung điện áp mức cao (5V) kéo dài tối thiểu 10\us vào chân Trig.
* **Chuỗi xung (Burst):** Ngay sau khi nhận Trigger, cảm biến tự động phát ra một chuỗi 8 xung vuông ở tần số 40 kHz. Việc phát chuỗi 8 xung (thay vì 1 xung) giúp tăng năng lượng sóng âm và giúp bộ lọc tại đầu thu dễ dàng phân biệt tín hiệu phản hồi thật với các tạp âm môi trường.
* **Tín hiệu Echo:** Khi chuỗi xung gặp vật cản và phản xạ lại đầu thu, mạch so sánh bên trong cảm biến sẽ kéo chân Echo lên mức cao. Độ rộng của xung Echo (T\_{pulse}) tỷ lệ thuận với khoảng cách.

### **2.3.2 Công thức tính toán và Sai số**

Khoảng cách d được tính theo công thức vật lý:

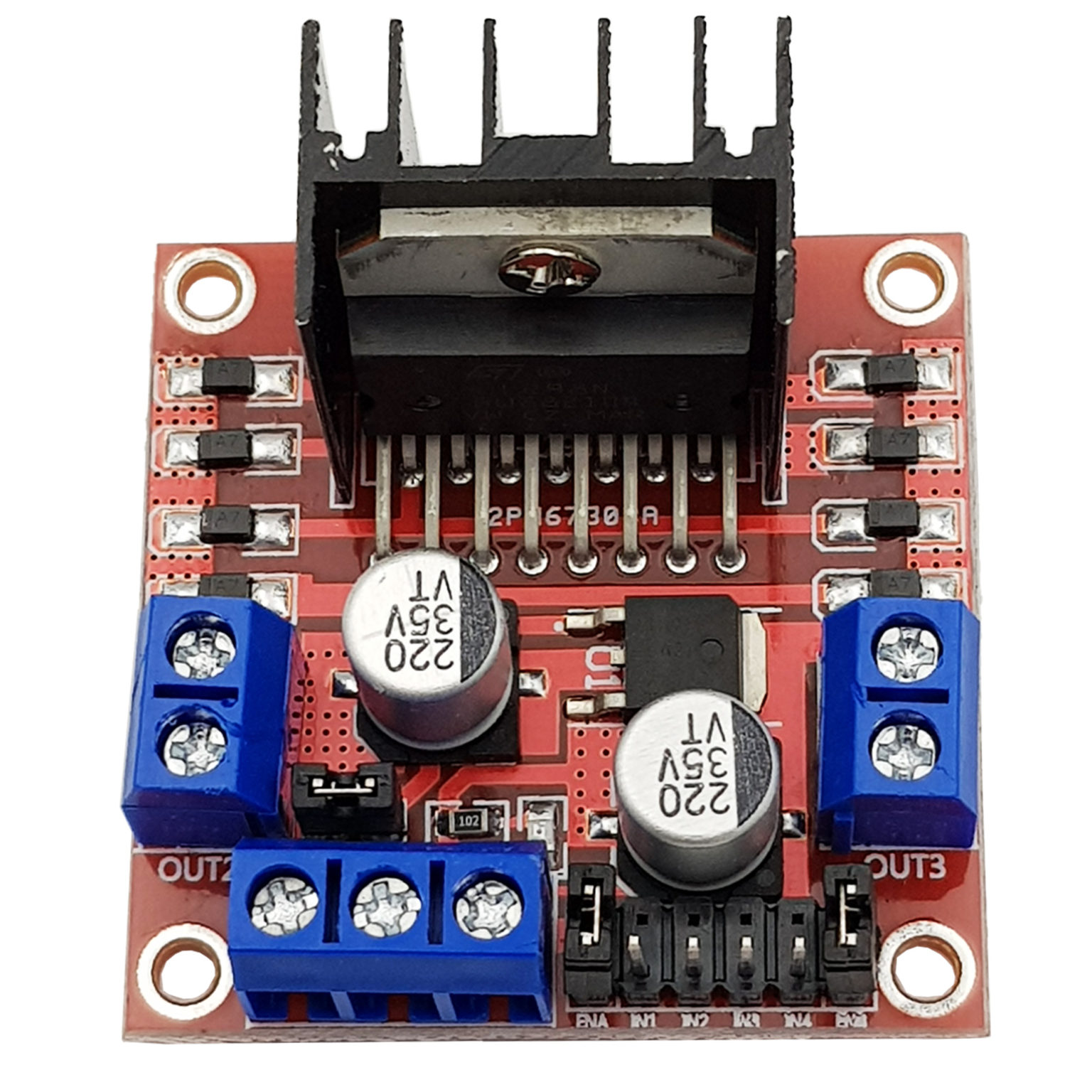
d=v \* t/2

Trong đó:

* v là vận tốc âm thanh trong không khí. Vận tốc này phụ thuộc vào nhiệt độ môi trường theo công thức thực nghiệm: 
* t là thời gian đo được từ chân Echo (us).
* Phép chia cho 2 là bắt buộc vì sóng âm phải đi hành trình khứ hồi (Từ cảm biến -> Vật cản -> Cảm biến).

**Giới hạn kỹ thuật:** Cảm biến HC-SR04 có góc đo khoảng 15 đến 30 độ. Điều này tạo ra các "điểm mù" ở hai bên sườn xe, đòi hỏi thuật toán điều khiển phải xử lý cẩn thận (ví dụ: quay đầu quét xung quanh) để tránh va chạm.

## 2.4 Mạch cầu H L298N (Driver Động cơ)



### **2.4.1 Cấu trúc mạch cầu H (H-Bridge)**

Mạch cầu H là một cấu trúc điện tử gồm 4 công tắc bán dẫn (thường là Transistor BJT hoặc MOSFET) xếp thành hình chữ H, với động cơ nằm ở giữa. L298N sử dụng Transistor BJT công suất.

* **Trạng thái quay thuận:** Hai transistor chéo nhau (Q1 và Q4) đóng, dòng điện chạy từ cực dương qua động cơ xuống đất -> Xe tiến.
* **Trạng thái quay nghịch:** Hai transistor chéo còn lại (Q2 và Q3) đóng, dòng điện đảo chiều -> Xe lùi.
* **Trạng thái hãm (Brake):** Hai transistor cùng phía (Q1, Q2 hoặc Q3, Q4) cùng đóng hoặc cùng ngắt, làm ngắn mạch hai đầu động cơ, tạo ra lực hãm điện từ giúp xe dừng nhanh hơn đà quán tính.

### **2.4.2 Thông số kĩ thuật**

* IC chính: L298 – Dual Full Bridge Driver
* Điện áp đầu vào: 5~30VDC
* Công suất tối đa: 25W 1 cầu (lưu ý công suất = dòng điện x điện áp nên áp cấp vào càng cao, dòng càng nhỏ, công suất có định 25W).
* Dòng tối đa cho mỗi cầu H là: 2A
* Mức điện áp logic: Low -0.3V~1.5V, High: 2.3V~Vss
* Kích thước: 43x43x27mm

Mặc dù phổ biến, L298N sử dụng công nghệ BJT cũ nên có nhược điểm là sụt áp (Voltage Drop) khá lớn, khoảng 1.4V - 2V trên mỗi cầu. Ví dụ: Nếu cấp nguồn 7.4V, điện áp thực tế đến động cơ chỉ còn khoảng 5.4V. Điều này cần được tính toán khi lập trình PWM để đảm bảo động cơ đủ mô-men xoắn khởi động.

## 2.5 Giao thức truyền thông: HTTP vs WebSocket

Để hệ thống hoạt động thời gian thực (Real-time), việc lựa chọn giao thức truyền thông là yếu tố sống còn.

### **2.5.1 Nhược điểm của HTTP trong điều khiển Robot**

Giao thức HTTP (Hypertext Transfer Protocol) hoạt động theo cơ chế "Yêu cầu - Phản hồi" (Request - Response).

* **Độ trễ cao:** Mỗi khi người dùng nhấn nút "Tiến", trình duyệt phải mở một kết nối TCP mới, gửi header HTTP, chờ Server xử lý, nhận phản hồi và đóng kết nối. Quá trình bắt tay (Handshake) lặp đi lặp lại này tạo ra độ trễ từ 100ms - 500ms, khiến việc điều khiển xe bị giật cục, thiếu chính xác.
* **Tốn băng thông:** Header của HTTP chứa nhiều thông tin meta-data dư thừa cho mỗi lần gửi lệnh nhỏ.

### **2.5.2 Ưu điểm của WebSocket**

WebSocket giải quyết triệt để vấn đề trên bằng cách thiết lập một đường ống giao tiếp hai chiều (Full-duplex) bền vững (Persistent connection) qua TCP.

* **Handshake một lần:** Quá trình bắt tay chỉ diễn ra một lần duy nhất khi khởi tạo kết nối. Sau đó, kênh truyền luôn mở.
* **Binary Stream:** WebSocket hỗ trợ gửi dữ liệu nhị phân (Binary Frames). Điều này cực kỳ quan trọng để truyền luồng ảnh JPEG từ Robot lên Server một cách nguyên bản mà không cần mã hóa Base64 (gây tăng 33% kích thước dữ liệu), giúp tối ưu hóa băng thông đường truyền.

## 2.6 Nền tảng Server Node.js

### **2.6.1 Cơ chế Event Loop và Non-blocking I/O**

Node.js hoạt động đơn luồng (Single-threaded) dựa trên cơ chế vòng lặp sự kiện (Event Loop). Thay vì tạo ra một luồng (Thread) mới cho mỗi kết nối (như PHP hay Java truyền thống) - điều sẽ ngốn rất nhiều RAM khi có hàng nghìn kết nối, Node.js sử dụng cơ chế Non-blocking I/O.

* Khi có dữ liệu hình ảnh từ Robot gửi lên, Node.js không "chờ" xử lý xong mới làm việc khác, mà đăng ký một sự kiện (Callback).
* Khi cần chuyển tiếp dữ liệu cho người dùng, Node.js thực hiện ngay lập tức.  
  Cơ chế này giúp Server Node.js cực kỳ nhẹ và nhanh, phù hợp hoàn hảo cho các ứng dụng I/O-bound (thiên về vào/ra dữ liệu) như Streaming video và điều khiển Robot.

## 2.7 Trí tuệ nhân tạo và Thị giác máy tính (Computer Vision)

### **2.7.1 Mạng nơ-ron tích chập (CNN) và Pose Estimation**

Hệ thống tích hợp AI để phát hiện hành vi con người. Thay vì sử dụng các phương pháp xử lý ảnh cổ điển (như trừ nền, phát hiện chuyển động), dự án sử dụng mô hình học sâu (Deep Learning) chuyên dụng cho bài toán **Pose Estimation** (Ước lượng tư thế).

### **2.7.2 Nguyên lý phát hiện hành vi**

Mô hình AI MediaPipe sẽ trích xuất tọa độ của 17 điểm khớp xương (Keypoints) trên cơ thể người từ khung hình RGB.

* **Đầu vào:** Khung ảnh 
* **Đầu ra:** Tập hợp các điểm
* Logic phát hiện Ngã (Fall Detection):  
  Thuật toán phân tích sự thay đổi của Keypoint theo thời gian thực. Một người được coi là ngã khi thỏa mãn đồng thời các điều kiện:
  1. **Góc nghiêng cơ thể:** Trục nối điểm "Vai" và "Hông" tạo với phương thẳng đứng một góc lớn hơn 45 độ.
  2. **Tỷ lệ khung bao:** Chiều rộng của khung bao cơ thể (Bounding Box Width) lớn hơn chiều cao (Height).
  3. **Vận tốc hạ thấp:** Tọa độ y của điểm "Đầu" tăng nhanh đột ngột trong khoảng thời gian delta t ngắn.

### **2.7.3 Phát hiện đường tường (Wall Line Detection)**

**Đầu vào :** Khung hình RGB/Video ở thời gian thực.

**Đầu ra :** Đường biểu diễn tường dưới dạng phương trình: y = ax + b

Trong đó:

* **a**: hệ số góc (slope), độ nghiêng
* **b**: hệ số tự do (intercept)

**Quy trình**

Mô-đun sử dụng Canny Edge + Hough Transform để trích xuất các đoạn thẳng từ ảnh và chọn ra đường có thể đại diện cho tường.

1. Tăng cường ảnh

* Chuyển ảnh sang grayscale
* Làm mờ Gaussian để giảm nhiễu

1. Trích xuất cạnh

* Dùng bộ lọc Canny để tìm biên dạng trên ảnh

1. Phát hiện các đường thẳng

* Dùng HoughLinesP để tìm tập các đoạn thẳng có trong ảnh

1. Lọc đường tường  
    Chỉ giữ các đường:

* có độ nghiêng nhỏ (|slope| < 0.5) → gần nằm ngang
* nằm ở nửa dưới khung hình  
   → vì tường hoặc bậc cửa thường xuất hiện ở vùng thấp của ảnh

5. Tính slope & intercept trung bình động

Nếu tìm được nhiều đường thỏa mãn → lấy trung bình

* Làm mượt theo công thức EMA:

a = 0.9 \* a\_cũ + 0.1 \* a\_mới

b = 0.9 \* b\_cũ + 0.1 \* b\_mới

Nếu không tìm được tường → hệ thống dùng giá trị mặc định (tường nằm ~ 60% chiều cao màn hình).

### **2.7.4 Phát hiện chuyển động (Motion Detection)**

Sử dụng **Background Subtraction (MOG2)** để phát hiện vùng chuyển động.

**Quy trình**

1. Làm mờ và chuyển ảnh sang grayscale
2. Dùng cv2.createBackgroundSubtractorMOG2() để tách vùng foreground
3. Ngưỡng nhị phân + lọc nhiễu
4. Tìm contour để dựng bounding box của các vật thể

**Đầu ra**

Danh sách vật thể dạng:

[ { x, y, w, h, cx, cy, area },..]

3. Phát hiện xâm nhập (Intrusion Detection)

Sau khi có đường tường và các vật thể, hệ thống kiểm tra xem vật thể có vượt qua đường tường hay không.

**Nguyên lý**

Điểm được coi là xâm nhập nếu:

cy<(a∗cx+b)cy < (a \* cx + b)cy<(a∗cx+b)

tức tâm vật thể nằm phía trên đường tường.

Nếu đúng → đánh dấu: obj['is\_intruder'] = True

Các vật thể vi phạm sẽ thêm vào danh sách intruders.

Trạng thái

* **An toàn** → bounding box màu xanh
* **Xâm nhập** → bounding box màu đỏ + nhãn INTRUDER .

### **2.7.5 Tích hợp với Person Detector & Cơ chế “vẫy tay miễn nhiễm”**

Sau khi xác định các moving\_objects, hệ thống thực hiện kiểm tra xâm nhập theo thứ tự ưu tiên sau:

* **Kiểm tra vượt tường (dựa trên đường tường đã phát hiện hoặc ngưỡng tường ảo)**

wall\_y = a \* cx + b # nếu dùng Hough Line

hoặc

wall\_y = h \* Config.WALL\_THRESHOLD # nếu dùng ngưỡng cố định

if cy < wall\_y:

obj['is\_intruder'] = True

* **Gắn kết quả nhận diện người (MediaPipe Pose)**
* Nếu PersonDetector.is\_person\_detected == True → đánh dấu vật thể gần nhất với tâm người là “người thật”.
* Chỉ những vật thể là **người thật** mới được phép thực hiện hành động vẫy tay để xác nhận an toàn.

**Cơ chế xác nhận an toàn bằng vẫy tay**

* Khi PersonDetector.detect\_wave() trả về confirmed == True → tìm object gần nhất với vị trí cổ tay (wrist) đang vẫy → gán:

obj['confirmed\_safe'] = True

obj['safe\_until'] = time.time() + 30 # miễn nhiễm 30 giây

|  |  |  |  |
| --- | --- | --- | --- |
| Điều kiện | Màu khung | Nhãn hiển thị | Kích hoạt còi/báo động |
| Không vượt tường | Xanh lá | “SAFE” | Không |
| Vượt tường + chưa xác nhận an toàn | Đỏ | “INTRUDER!” | Có |
| Vượt tường + đã vẫy tay thành công | Vàng | “AUTHORIZED” hoặc “SAFE” | Không (trong thời gian miễn nhiễm) |
| Vượt tường + đã hết thời gian miễn nhiễm | Đỏ | “INTRUDER! (Expired)” | Có |

Xử lý âm thanh & thông báo

* Khi có intruder thật sự (vượt tường + không được xác nhận an toàn):

trigger\_buzzer() # kêu còi

send\_telegram\_alert(frame) # gửi ảnh qua bot Telegram

log\_intrusion\_event() # ghi log + lưu ảnh

Ưu điểm của cơ chế kết hợp này :

* Giảm đáng kể báo động giả do lá cây, động vật, bóng đổ.
* Cho phép người quen (nhân viên, gia chủ) dễ dàng “tắt cảnh báo” chỉ bằng một cử chỉ vẫy tay tự nhiên.
* Vẫn duy trì khả năng phát hiện trèo tường ngay cả khi không có đường tường vật lý rõ ràng (dùng ngưỡng ảo).

Kết luận phần Motion + Intrusion Detection

Hệ thống sử dụng kết hợp Background Subtractor MOG2 → Contour detection → Kiểm tra vị trí tường → MediaPipe Pose + Wave Gesture Recognition, tạo nên một lớp bảo vệ thông minh, có khả năng phân biệt “người lạ nguy hiểm” và “người quen” chỉ trong vài giây mà không cần thẻ từ hay mật khẩu.

# 3. Phân tích yêu cầu chức năng

## 3.1 Chức năng đăng nhập vào hệ thống

* 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, hoten, ngaysinh, diachi.
* Nếu đăng nhập thành công, web sẽ chuyển hướng tới giao diện chính.
* 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 điều khiển xe

Đây là chức năng tương tác chính, cho phép người dùng điều hướng Robot từ xa.

* Luồng dữ liệu: User (Web) -> Server -> ESP32 -> L298N.
* Các thao tác điều khiển:
  + Điều hướng: Cung cấp 5 lệnh cơ bản: Tiến (Forward), Lùi (Backward), Quay trái (Turn Left), Quay phải (Turn Right), Dừng (Stop).
  + Điều tốc: Cho phép thay đổi tốc độ động cơ thông qua thanh trượt (Slider) trên giao diện Web. Giá trị này được chuyển đổi thành độ rộng xung PWM (0-255) trên ESP32.
* Yêu cầu phi chức năng: Độ trễ (Latency) từ lúc nhấn nút đến khi xe di chuyển phải nhỏ hơn 200ms để đảm bảo khả năng kiểm soát.

## 3.3 Chức năng xe tự động vận hành

Đây là chức năng nâng cao, cho phép Robot tự di chuyển và tìm đường mà không cần can thiệp từ người dùng.

* Kích hoạt: Người dùng chuyển công tắc "Auto Mode" trên Web Dashboard. Server gửi lệnh xuống ESP32.
* Thuật toán di chuyển (Obstacle Avoidance Algorithm): Robot thực hiện vòng lặp logic cục bộ trên ESP32:
  1. Quét môi trường: Đọc dữ liệu từ cảm biến siêu âm HC-SR04.
  2. Ra quyết định:
     + Nếu khoảng cách an toàn (> 30cm): Robot duy trì trạng thái Tiến thẳng.
     + Nếu gặp vật cản (< 30cm):
       1. Dừng động cơ ngay lập tức.
       2. Lùi lại một khoảng ngắn (trong 0.5s) để tạo khoảng trống.
       3. Quay đầu (Trái hoặc phải ngẫu nhiên hoặc theo quy luật) trong một khoảng thời gian định trước.
       4. Tiếp tục đo khoảng cách. Nếu thoáng, quay lại bước Tiến thẳng.
* Phản hồi: Trong quá trình tự hành, Robot gửi trạng thái hoạt động về Server (ví dụ: nếu gặp người vượt rào hoặc bị ngã) để người dùng giám sát.

## 3.4 Chức năng giám sát hình ảnh

Module ESP32-CAM hoạt động độc lập với chế độ di chuyển của Robot, đảm bảo người dùng luôn nhìn thấy môi trường xung quanh dù Robot đang chạy Tự động hay Thủ công.

* Luồng dữ liệu: ESP32-CAM -> Server -> User Client.
* Yêu cầu:
  + Hình ảnh phải được truyền liên tục, độ trễ thấp để người dùng giám sát hành vi của Robot

## 3.5 Chức năng chuyển đổi trạng thái

Hệ thống cần đảm bảo việc chuyển đổi giữa hai chế độ diễn ra mượt mà và an toàn.

* Quy trình chuyển đổi:
  + Khi chuyển từ Manual -> Auto: Robot dừng lại 1 giây, reset các biến trạng thái, sau đó bắt đầu chạy thuật toán tự động.
  + Khi chuyển từ Auto -> Manual: Robot dừng khẩn cấp ngay lập tức, trao lại quyền kiểm soát cho các nút bấm của người dùng.
* Ưu tiên: Lệnh "Dừng" (Stop) hoặc chuyển chế độ từ người dùng luôn có độ ưu tiên cao nhất, có thể ngắt ngang bất kỳ tiến trình tự động nào của Robot.

## 3.6 Chức năng AI Giám sát An ninh

Đây là chức năng chạy nền trên Server, phân tích từng khung hình nhận được từ Robot.

* Phát hiện Vượt rào (Climbing Detection):
  1. Người dùng thiết lập một "Vùng cấm" hoặc "Vạch hàng rào" trên giao diện Camera.
  2. Server phân tích tọa độ người trong ảnh. Nếu trọng tâm hoặc chân của người nằm cao hơn ngưỡng quy định (vạch rào), hệ thống kích hoạt cảnh báo.
* Phát hiện Ngã (Fall Detection):
  1. Server tính toán tỷ lệ cơ thể (Aspect Ratio) và vận tốc di chuyển của các điểm khớp.
  2. Nếu tỷ lệ thay đổi đột ngột (từ đứng sang nằm) và tọa độ đầu hạ thấp nhanh trong thời gian ngắn (< 1s), hệ thống xác định là "Ngã".
* Cơ chế xử lý sự kiện: Khi phát hiện 1 trong 2 hành vi trên:
  1. Gửi cảnh báo tức thì tới giao diện điều khiển (Popup/Âm thanh).
  2. Chụp lại khung hình tại thời điểm đó (Snapshot).
  3. Ghi dữ liệu vào Cơ sở dữ liệu (Database).

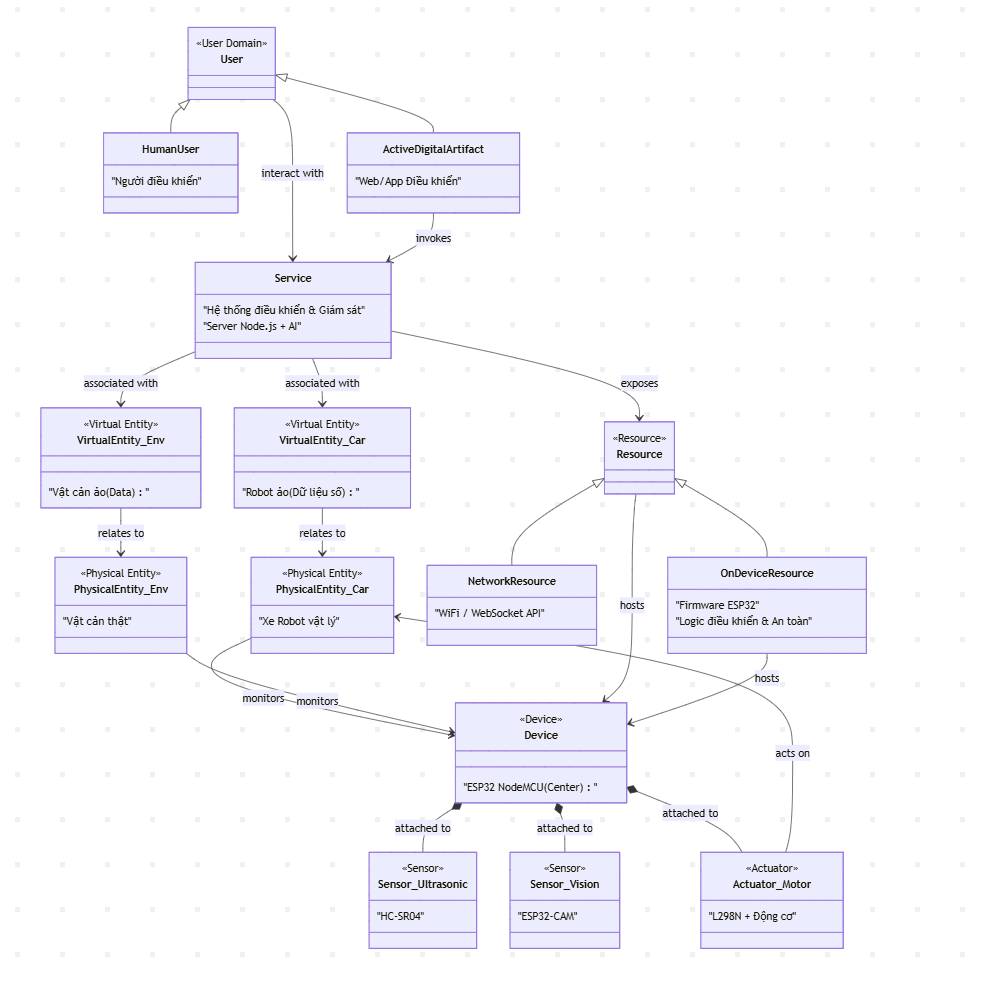
## 3.7 Chức năng xem lịch sử cảnh báo

Chức năng này cho phép người dùng xem lại các sự kiện đã xảy ra.

* Giao diện: Một trang Lịch sử (History Log) hiển thị danh sách dạng bảng hoặc thẻ.
* Thông tin hiển thị:
  + Thời gian: Ngày/Giờ xảy ra sự kiện (Timestamp).
  + Loại sự kiện: "Vượt rào" hoặc "Bị ngã".
  + Hình ảnh bằng chứng: Ảnh chụp khoảnh khắc sự kiện xảy ra (người dùng có thể bấm vào để xem to).
* Thao tác: Cho phép lọc theo ngày, lọc theo loại sự kiện, hoặc xóa lịch sử.

# 4. Thiết kế hệ thống

## 4.1 Đặc tả mô hình miền



1. Miền Người dùng (User Domain)

* User (Người dùng): Là thực thể tương tác với hệ thống để điều khiển hoặc giám sát.
* Human User: Người vận hành xe thực tế.
* Active Digital Artifact (App): Là giao diện Website (Web Dashboard) hiển thị hình ảnh từ Camera và các nút điều khiển. Đây là cầu nối giữa người và máy.

2. Miền Dịch vụ (Service Domain)

* Service: Đại diện cho Server trung gian (Node.js) và các dịch vụ chạy nền (như AI phát hiện ngã/vượt rào). Dịch vụ này nhận lệnh từ App, xử lý logic và gửi xuống thiết bị, đồng thời nhận dữ liệu từ thiết bị để hiển thị lên App.

3. Miền Thực thể (Entity Domain)

Đây là sự ánh xạ giữa thế giới thực và thế giới số:

* Physical Entity (Thực thể vật lý):
  + *Xe Robot:* Chiếc xe thật đang chạy.
  + *Môi trường:* Các vật cản, địa hình, hoặc đối tượng (người) mà Camera nhìn thấy.
* Virtual Entity (Thực thể ảo):
  + *Robot ảo:* Là các biến số trong phần mềm đại diện cho xe (Ví dụ: trạng thái status=”moving” , tốc độ =”100”).
  + *Dữ liệu môi trường:* Là các giá trị số hóa (Ví dụ: khoảng cách =”20” cm , hình ảnh JPEG nhận được).

4. Miền Tài nguyên (Resource Domain)

* Network Resource: Bao gồm kết nối Wi-Fi và giao thức WebSocket API giúp truyền tải dữ liệu.
* OnDevice Resource: Là mã nguồn (Firmware C++) chạy trên ESP32. Đây là nơi chứa các thuật toán quan trọng như: logic tự động dừng khi gặp vật cản, logic đọc cảm biến.

5. Miền Thiết bị (Device Domain) - Trung tâm hệ thống

Đây là phần cứng cốt lõi, được tổ chức theo yêu cầu ESP32 NodeMCU làm trung tâm:

* Device (Thiết bị chính): Là vi điều khiển ESP32 NodeMCU. Nó đóng vai trò là bộ não trung tâm, chịu trách nhiệm kết nối mạng và điều phối tất cả các thành phần khác.
* Các thành phần gắn kèm (Attached to):
  + Sensor (Ultrasonic): Cảm biến HC-SR04 được gắn vào ESP32 để đo khoảng cách.
  + Sensor (Vision/Camera): Module ESP32-CAM. Trong mô hình miền này, ESP32-CAM được coi là một "Cảm biến hình ảnh thông minh" (Smart Vision Sensor). Nó được ESP32 kích hoạt hoặc hoạt động song song để cung cấp dữ liệu hình ảnh cho hệ thống.
  + Actuator (Cơ cấu chấp hành): Module L298N và Động cơ. ESP32 gửi tín hiệu PWM xuống L298N để thực hiện hành động vật lý (quay bánh xe) tác động lên Thực thể vật lý (Xe Robot).

## 4.2 Đặc tả các thành phần chức năng và hoạt động

### 4.2.1 Thành phần chức năng

Phần này mô tả sự ánh xạ từ các thiết bị và tài nguyên thực tế của dự án vào các khối chức năng trong kiến trúc hệ thống tổng quát.

* Thiết bị IoT (IoT Devices): Được ánh xạ tới khối Device. Cụ thể bao gồm: 2 cảm biến (Camera OV2640, Cảm biến siêu âm HC-SR04), 1 cơ cấu chấp hành (Mạch cầu H L298N & Động cơ), và 2 vi xử lý (ESP32 NodeMCU, ESP32-CAM).
* Quản lý thiết bị (Device Management): Được ánh xạ tới khối Management. Chức năng này do Server đảm nhiệm, theo dõi trạng thái kết nối (Online/Offline) của Robot.
* Tài nguyên (Resources): Được ánh xạ tới khối Device (Tài nguyên trên thiết bị - OnDevice Resource) và khối Communication (Các API giao tiếp và giao thức mạng).
* Dịch vụ điều khiển (Controller Service): Được ánh xạ tới khối Services (Native Services). Đây là các logic cốt lõi như thuật toán tránh vật cản tự động chạy trên ESP32 và thuật toán AI phát hiện ngã chạy trên Server.
* Dịch vụ Web (Web Service): Được ánh xạ tới khối Services. Bao gồm dịch vụ Streaming hình ảnh và API cung cấp lịch sử cảnh báo.
* Cơ sở dữ liệu (Database): Được ánh xạ tới khối Management (Quản lý dữ liệu) và khối Security (Lưu trữ thông tin xác thực).
* Ứng dụng Web (Web App): Được ánh xạ tới khối Application, đồng thời tương tác với khối Management (Quản lý phiên người dùng) và khối Security (Đăng nhập/Xác thực).

### 4.2.2 Thành phần hoạt động

Phần này liệt kê cụ thể các công nghệ, phần mềm và giao thức được sử dụng để hiện thực hóa các chức năng trên.

Application (Tầng Ứng dụng):

* Web App: Dashboard điều khiển được xây dựng bằng HTML5, CSS3, JavaScript (Client-side).
* Application Server: Máy chủ ứng dụng chạy nền tảng Node.js (Express framework).
* Database Server: Hệ quản trị cơ sở dữ liệu MongoDB (NoSQL) để lưu trữ phi cấu trúc (Log sự kiện, đường dẫn ảnh).

Management (Khối Quản lý):

* Application Management: Quản lý logic nghiệp vụ trên Node.js.
* Database Management: Sử dụng thư viện Mongoose (ODM) để quản lý kết nối và truy vấn MongoDB.
* Device Management: Module quản lý danh sách WebSocket Client trên Server.

Services (Khối Dịch vụ):

* Native Services:
  + *AI Service:* Module xử lý ảnh (sử dụng thư viện Pose Estimation) để phát hiện hành vi.
  + *Safety Logic:* Mã nguồn C++ trên ESP32 xử lý phanh khẩn cấp.
* Web Services: Dịch vụ phát luồng video (MJPEG Streamer) và RESTful API cho lịch sử.

Communication (Khối Giao tiếp):

* Communication APIs: Định nghĩa các sự kiện (Events).
* Communication Protocols: Sử dụng WebSocket cho điều khiển thời gian thực và HTTP cho tải tài nguyên.

Security (Khối Bảo mật):

* Authentication: Cơ chế xác thực vai trò (Role-based) khi thiết lập kết nối WebSocket (User vs Robot).
* Authorization: Kiểm tra quyền gửi lệnh trước khi chuyển tiếp xuống Robot.

Device (Tầng Thiết bị):

* Sensors: Cảm biến siêu âm HC-SR04, Camera OV2640.
* Actuator: Module điều khiển động cơ L298N.
* Computing Devices: Vi điều khiển ESP32 NodeMCU (Điều khiển trung tâm) và ESP32-CAM (Thu thập hình ảnh).

## 4.3 Đặc tả luồng hoạt động

### **4.3.1 Thiết bị cảm biến và vi xử lí**

a) Thư viện sử dụng

|  |  |  |
| --- | --- | --- |
| **Tên Thư Viện** | **Loại thư viện** | **Chức năng & Vai trò trong dự án này** |
| **Arduino.h** | **Lõi (Core)** | **Cầu nối phần cứng:** Cung cấp các lệnh cơ bản (pinMode, digitalWrite, delay...) giúp lập trình chip ESP32 bằng ngôn ngữ Arduino dễ hiểu. |
| **esp\_camera.h** | Phần cứng (Hardware) | **Quản lý Camera:** Cấu hình chân kết nối với module OV2640, thiết lập độ phân giải và chụp dữ liệu hình ảnh để gửi đi. |
| **WiFi.h** | Mạng (Network) | **Phát Wifi:** Tạo ra điểm phát sóng (SoftAP) tên là "MyWiFiCar" để điện thoại kết nối trực tiếp vào xe. |
| **ESPAsyncWebServer.h** & **AsyncTCP.h** | Mạng (Network) | **Máy chủ Web Bất đồng bộ:** Giúp ESP32 xử lý đa nhiệm (vừa livestream hình ảnh nặng, vừa nhận lệnh điều khiển xe ngay lập tức) mà không bị "treo" hay trễ (lag). |
| **iostream**  & **sstream** | Chuẩn C++ | **Xử lý chuỗi:** Dùng để cắt và phân tích chuỗi lệnh từ Web gửi về (ví dụ: cắt "MoveCar,1" thành lệnh MoveCar và giá trị 1). |

b) Các hàm chức năng

|  |  |
| --- | --- |
| **Tên Hàm** | **Chức năng & Nhiệm vụ** |
| **setupCamera()** | Khởi động Camera OV2640. Thiết lập các thông số như chân kết nối (Y2, Y3...), độ phân giải (VGA), và chất lượng ảnh. Nếu camera lỗi, nó sẽ báo lỗi ngay tại đây. |
| **sendCameraPicture()** | Chụp một bức ảnh từ camera (esp\_camera\_fb\_get) và gửi nó qua sóng Wifi đến trình duyệt web của bạn để hiển thị hình ảnh thời gian thực (Livestream). |
| **onCarInputWebSocketEvent()** | Hàm xử lý lệnh từ người dùng. Khi bạn bấm nút trên điện thoại, hàm này sẽ nhận tín hiệu. Nó phân loại xem đó là lệnh lái xe, chỉnh tốc độ hay bật đèn. |
| **sendCarCommands()** | Gửi lệnh xuống mạch dưới. Hàm này dùng lệnh để đẩy chuỗi lệnh qua chân U0T (TX) xuống chân RX của ESP32 Motor. |
| **setUpPinModes()** | Cấu hình PWM cho đèn Flash (LED) trên module ESP32-CAM, giúp bạn có thể chỉnh độ sáng đèn qua thanh trượt trên Web. |

|  |  |
| --- | --- |
| **parseCommand(String data)** | Bộ giải mã thông minh. Nhiệm vụ của nó là cắt chuỗi dữ liệu nhận được từ ESP32-CAM. |
| **controlMotor(int command)** | Trung tâm điều hướng. Dựa vào con số nhận được (0, 1, 2, 3, 4), nó quyết định xe sẽ làm gì:  - 1: Gọi hàm đi thẳng.  - 2: Gọi hàm đi lùi.  - 3,4: Xoay trái/phải.  - 0: Dừng lại. |
| **setMotors(...)** | Trình điều khiển phần cứng (Driver). Hàm này trực tiếp ra lệnh HIGH hoặc LOW cho các chân IN1, IN2, IN3, IN4 để các bánh xe quay theo chiều mong muốn. Đây là hàm cấp thấp nhất tiếp xúc với mạch L298N. |
| **runAutoMode()** | Chế độ Tự lái (Auto Pilot). Hàm này chạy liên tục trong vòng lặp loop() nếu bạn bật chế độ tự động. Nó chứa kịch bản di chuyển (Tiến -> Dừng -> Lùi -> Xoay) dựa trên thời gian. |
| **ledcSetup & ledcAttachPin** | Cấu hình bộ tạo xung PWM của ESP32. Nó quy định tần số xung (1000Hz) để điều khiển tốc độ động cơ mượt mà, không bị giật. |

### **4.3.2 Website quản lý**

a) Kiến trúc hệ thống Website

Website Dashboard được thiết kế theo mô hình Client-Server với kiến trúc 3 tầng phân tán như mô tả trong sơ đồ kiến trúc:​

Tầng Application (Ứng dụng):

* Web Dashboard: Giao diện người dùng được xây dựng bằng HTML5, CSS3 và JavaScript thuần, chạy trên trình duyệt web
* Application Server: Máy chủ ứng dụng Node.js với framework Express.js xử lý logic nghiệp vụ
* Database Server: Hệ quản trị cơ sở dữ liệu MongoDB lưu trữ lịch sử cảnh báo và thông tin người dùng

Tầng Communication (Giao tiếp):

* WebSocket Protocol: Thiết lập 2 kênh truyền thông song công riêng biệt - một kênh /ws/user cho người dùng, một kênh /ws/robot cho xe robot​
* HTTP/JSON API: Các endpoint RESTful để truy vấn lịch sử cảnh báo và quản lý dữ liệu

Tầng Device (Thiết bị):

* ESP32 NodeMCU và ESP32-CAM kết nối đến Server qua WiFi, nhận lệnh điều khiển và gửi dữ liệu cảm biến theo thời gian thực

b) Công nghệ sử dụng

|  |  |  |
| --- | --- | --- |
| **Thành phần** | **Công nghệ** | **Vai trò** |
| Frontend | HTML5, CSS3, JavaScript (Vanilla) | Xây dựng giao diện người dùng tương tác cao với Canvas API để hiển thị video stream |
| Backend Server | Node.js + Express.js | Máy chủ trung gian xử lý routing, middleware và chuyển tiếp lệnh giữa User và Robot |
| Real-time Protocol | WebSocket (thư viện ws) | Truyền thông hai chiều với độ trễ thấp < 200ms cho điều khiển và video streaming ​ |
| Database | MongoDB + Mongoose ODM | Lưu trữ dữ liệu phi cấu trúc (Alert logs, User sessions) với schema linh hoạt |
| AI Processing | TensorFlow.js + MediaPipe Pose | Phân tích video frame để phát hiện hành vi nguy hiểm ​ |
| Image Processing | Canvas API / Sharp | Vẽ bounding box, keypoints và xử lý ảnh cảnh báo |

c) Chức năng Dashboard

1. Màn hình Video Streaming

* Hiển thị Video: Khu vực trung tâm kích thước 640×480 pixels sử dụng HTML5 Canvas để vẽ từng JPEG frame nhận được qua WebSocket​
* FPS Counter: Hiển thị góc trên bên trái để theo dõi tốc độ truyền hình ảnh (mục tiêu 10-15 FPS)
* Alert Overlay: Lớp phủ màu đỏ xuất hiện khi AI phát hiện hành vi nguy hiểm, hiển thị cảnh báo " PHÁT HIỆN BẤT THƯỜNG!" trong 5 giây
* Pose Visualization: Vẽ 17 điểm keypoints màu đỏ và các đường nối skeleton màu xanh lá khi có detection

2. Bảng điều khiển (Control Panel)

Chế độ hoạt động:

* Toggle switch để chuyển đổi giữa "Manual" (Thủ công) và "Auto" (Tự động)
* Khi bật Auto, các nút điều khiển bị vô hiệu hóa (disabled), xe tự hành theo thuật toán tránh vật cản

Joystick 5 hướng:

|  |  |  |
| --- | --- | --- |
| **Nút điều khiển** | **Command gửi** | **Chức năng** |
| ↑ (Forward) | {type: "control", command: "MoveCar", value: 1} | Tiến về phía trước |
| ↓ (Backward) | {type: "control", command: "MoveCar", value: 2} | Lùi về phía sau |
| ← (Left) | {type: "control", command: "MoveCar", value: 3} | Rẽ trái |
| → (Right) | {type: "control", command: "MoveCar", value: 4} | Rẽ phải |
| ⬛ (Stop) | {type: "control", command: "MoveCar", value: 0} | Dừng lại |

Thanh trượt tốc độ:

* Range slider từ 0-255 (giá trị PWM)
* Mặc định: 150 (tốc độ trung bình ~59% công suất)
* Gửi lệnh {type: "control", command: "Speed", value: <giá\_trị>} khi thay đổi

Panel cảm biến:

* Khoảng cách vật cản: Hiển thị giá trị từ HC-SR04 cập nhật mỗi 1-2 giây, đổi màu đỏ khi < 30cm
* Tốc độ hiện tại: Giá trị PWM đang áp dụng (0-255)
* Trạng thái kết nối: Chấm tròn màu xanh (Online) / đỏ (Offline)

3. Lịch sử cảnh báo AI

Bảng dữ liệu (Table):

|  |  |
| --- | --- |
| **Cột** | **Nội dung** |
| Thời gian | Format: "DD/MM/YYYY HH:mm:ss" (múi giờ GMT+7) |
| Loại sự kiện | Badge màu: Xanh dương (Vượt rào) / Đỏ (Bị ngã) |
| Độ tin cậy | Phần trăm confidence của model (70-100%) |
| Hình ảnh | Thumbnail 80×60 pixels với keypoints đã vẽ |
| Thao tác | Nút " Xem chi tiết" và " Xóa" |

Chức năng lọc:

* Dropdown filter theo loại: "Tất cả", "Vượt rào", "Bị ngã"
* Nút " Làm mới" để reload dữ liệu từ API
* Tự động cập nhật real-time khi có cảnh báo mới qua WebSocket

d) Luồng giao tiếp WebSocket

Server Node.js đóng vai trò trung gian chuyển tiếp giữa User Client và Robot Client:​

Hướng User → Robot (Điều khiển):

[User Browser] → WebSocket → [Server Node.js] → WebSocket → [ESP32]

Message: {"type": "control", "command": "MoveCar", "value": 1}

Hướng Robot → User (Dữ liệu):

[ESP32] → WebSocket → [Server Node.js] → Broadcasting → [All User Browsers]

Message: {"type": "video\_frame", "frame": "<base64\_jpeg>"}

Cơ chế Broadcasting:

* Server duy trì danh sách tất cả User WebSocket đang kết nối
* Khi nhận video frame từ Robot, Server broadcast đến tất cả Users đồng thời (multicast)
* Module AI chạy song song phân tích frame, không chặn luồng broadcasting

e) API Endpoints

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method** | **Endpoint** | **Tham số** | **Mô tả** | **Response** |
| GET | /api/alerts | ?type=<loại>&startDate=<timestamp> | Lấy danh sách cảnh báo có bộ lọc | Mảng JSON chứa các alert objects |
| GET | /api/alerts/:id | id (MongoDB ObjectId) | Xem chi tiết 1 cảnh báo cụ thể | Object JSON với đầy đủ thông tin và keypoints |
| DELETE | /api/alerts/:id | id (MongoDB ObjectId) | Xóa cảnh báo khỏi database | {success: true} |
| GET | /api/robot/status | Không có | Kiểm tra trạng thái Robot | {connected: true, lastSeen: <timestamp>} |

f) Database Schema

Collection: alerts

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Mô tả** |
| \_id | ObjectId | Primary key tự động |
| type | String (enum) | "climbing" hoặc "fall" |
| confidence | Number (0-1) | Độ tin cậy của model AI |
| imag\*eUrl | String | Đường dẫn tương đối: /alerts/IMG\_xxx.jpg |
| keypoints | Array | Mảng 17 objects {x, y, score} tọa độ điểm khớp xương |
| timestamp | Date | Thời gian phát hiện (ISODate) |

### **4.3.3 Hệ thống AI Giám sát An ninh**

a. Pipeline tổng thể của hệ thống AI

Pipeline xử lý AI bao gồm 6 giai đoạn chính:

- Nhận dữ liệu hình ảnh từ Robot (ESP32-CAM → Server)

+ ESP32-CAM gửi liên tục các khung hình MJPEG thông qua Server trung gian.

+ Server tách từng frame, chuyển vào hàng đợi xử lý (Frame Queue).

+ Mỗi frame được gán timestamp để phục vụ nhận diện và lưu log.

- Tiền xử lý ảnh (Preprocessing)

+ Chuyển đổi khung hình về định dạng chuẩn (RGB).

+ Resize về kích thước chuẩn mô hình pose estimation yêu cầu (ví dụ: 256×256 hoặc 320×320).

+ Chuẩn hóa giá trị pixel (Normalization).

+ Loại bỏ nhiễu nhẹ bằng bộ lọc Gaussian nếu ảnh từ ESP32-CAM bị nhiễu.

- Trích xuất Pose Skeleton bằng mô hình AI

+ Sử dụng mô hình Pose Estimation (MoveNet / BlazePose / OpenPose tùy triển khai).

+ Mô hình trả về 17–33 điểm khớp (keypoints) của cơ thể người: đầu, vai, khuỷu tay, hông, đầu gối, cổ chân,…

+ Trích xuất vector tọa độ (x, y) và độ tin cậy (confidence) của mỗi keypoint.

+ Lọc các keypoint có độ tin cậy thấp nhằm giảm nhiễu.

- Phân tích hành vi dựa trên Pose Skeleton

Hệ thống áp dụng hai nhánh phân tích độc lập:

+ Nhánh phát hiện vượt rào (Climbing Detection)

Server thực hiện:

* Kiểm tra vị trí các khớp chân, khớp hông, trọng tâm cơ thể so với vạch rào do người dùng thiết lập trên giao diện.
* Tính chiều cao tương đối của keypoint: Nếu chân hoặc trọng tâm vượt quá ngưỡng (rào), đánh dấu trạng thái climbing\_suspect.
* Tính vận tốc chuyển động theo trục đứng: nếu tốc độ tăng đột ngột + vượt vạch → kết luận Vượt rào.

+ Nhánh phát hiện ngã (Fall Detection)

Server sử dụng các chỉ số:

* Aspect Ratio (AR): tỷ lệ giữa chiều cao và chiều rộng bounding box cơ thể.
* Hướng vector trục cơ thể (Tư thế đứng – tư thế nằm).
* Độ cao của đầu theo thời gian: nếu giảm nhanh trong < 1s.
* Tốc độ thay đổi tư thế: từ đứng → nằm.
* Điều kiện xác nhận ngã: AR giảm mạnh Đầu/torso chạm gần mặt đất Hành vi chuyển đổi nhanh (đột ngột)
* Nếu cả 3 điều kiện thỏa → xác định Fall Detected.

- Ra quyết định & kích hoạt cảnh báo

Khi một hành vi bất thường được phát hiện:

+ Server sinh sự kiện cảnh báo (Event Push) và gửi real-time qua WebSocket đến Dashboard người dùng.

+ Đồng thời chụp lại khung hình (Snapshot) từ pipeline tại thời điểm phát hiện.

+ Ghi log vào Database gồm:

* Timestamp
* Loại sự kiện (Climbing / Fall)
* Ảnh snapshot
* Giá trị keypoints phục vụ truy vết
* ID Robot / người dùng

- Lưu trữ & hiển thị lịch sử sự kiện

+ Database lưu toàn bộ sự kiện để người dùng xem lại.

+ Dashboard truy xuất qua API để hiển thị dạng bảng, có lọc theo ngày / loại cảnh báo.

+ Người dùng có thể phóng to ảnh log hoặc xuất dữ liệu.

b. Đặc tả tác nhân tham gia

|  |  |
| --- | --- |
| Thành phần | Vai trò |
| ESP32-CAM | Gửi khung hình liên tục |
| Server AI | Tiền xử lý, chạy mô hình Pose, phân tích hành vi |
| Client người dùng | Nhận cảnh báo, xem lịch sử, thiết lập vùng cấm |
| Database | Lưu snapshot, log sự kiện |

c. Đặc tả yêu cầu hoạt động của pipeline AI

- Độ trễ xử lý mỗi frame không vượt quá 150–250 ms để đảm bảo thời gian thực.

- Mô hình pose phải chạy ổn định trên CPU (không GPU).

- Tỷ lệ nhận diện đúng > 85% trong môi trường ánh sáng tốt.

- Với Climbing Detection: sai số biên < 5 px so với vạch rào.

- Với Fall Detection: ngưỡng thay đổi AR phải tối ưu để giảm báo sai.

- Dữ liệu cảnh báo phải được lưu tức thì, không mất dữ liệu khi Server quá tải.

## 4.4 Tích hợp thiết bị

## 

### **4.4.1. Khối Động Cơ & Driver L298N (Nối với ESP32 Chính)**

Hệ thống xe 4 bánh sử dụng 2 module L298N. Lưu ý: Các chân **ENA** và **ENB** dùng để điều chỉnh tốc độ (PWM).

Theo sơ đồ dây đầy đủ trong hình, bảng dưới đây bao gồm cả chức năng điều tốc:

|  |  |  |  |
| --- | --- | --- | --- |
| **Linh Kiện** | **Tên Chân (Pin)** | **Nối vào ESP32 (GPIO)** | **Chức năng** |
| **L298N Trái** | ENA | GPIO 32 | Điều tốc (PWM) động cơ trái trước |
|  | IN1 | GPIO 33 | Chiều quay động cơ trái trước |
|  | IN2 | GPIO 25 | Chiều quay động cơ trái trước |
|  | IN3 | GPIO 26 | Chiều quay động cơ trái sau |
|  | IN4 | GPIO 27 | Chiều quay động cơ trái sau |
|  | ENB | GPIO 14 | Điều tốc (PWM) động cơ trái sau |
| **L298N Phải** | ENA | GPIO 22 | Điều tốc (PWM) động cơ phải trước |
|  | IN1 | GPIO 16 | Chiều quay động cơ phải trước |
|  | IN2 | GPIO 17 | Chiều quay động cơ phải trước |
|  | IN3 | GPIO 18 | Chiều quay động cơ phải sau |
|  | IN4 | GPIO 19 | Chiều quay động cơ phải sau |
|  | ENB | GPIO 23 | Điều tốc (PWM) động cơ phải sau |

### **4.4.2. Khối Camera & Servo (Giao tiếp giữa 2 ESP32)**

Trong sơ đồ này, Servo được điều khiển bởi **ESP32-CAM** (để xoay camera), còn ESP32 chính sẽ gửi lệnh cho ESP32-CAM thông qua giao tiếp Serial.

|  |  |  |  |
| --- | --- | --- | --- |
| **Linh Kiện** | **Tên Chân** | **Nối Vào Đâu** | **Chức năng** |
| ESP32-CAM | TX2 (GPIO 4) | Nối vào chân **U0R** (RX) của ESP32-CAM | Gửi lệnh từ xe lên Camera |
|  | RX2 (GPIO 15) | Nối vào chân **U0T** (TX) của ESP32-CAM | Nhận dữ liệu từ Camera về xe |
|  | GND | Nối chung với GND của ESP32-CAM | Đồng bộ tín hiệu (Bắt buộc) |
| **Servo (Tilt+)** | Dây Xanh dương (Signal) | GPIO 13 (hoặc 15) của ESP32 | Điều khiển góc quay camera |
|  | Dây Đỏ (VCC) | 5V (Từ nguồn UBEC) | Cấp nguồn Servo |
|  | Dây đen(GND) | GND (Nối chung hệ thống) |  |

## 4.5 Phát triển ứng dụng

a) Giao diện đăng nhập

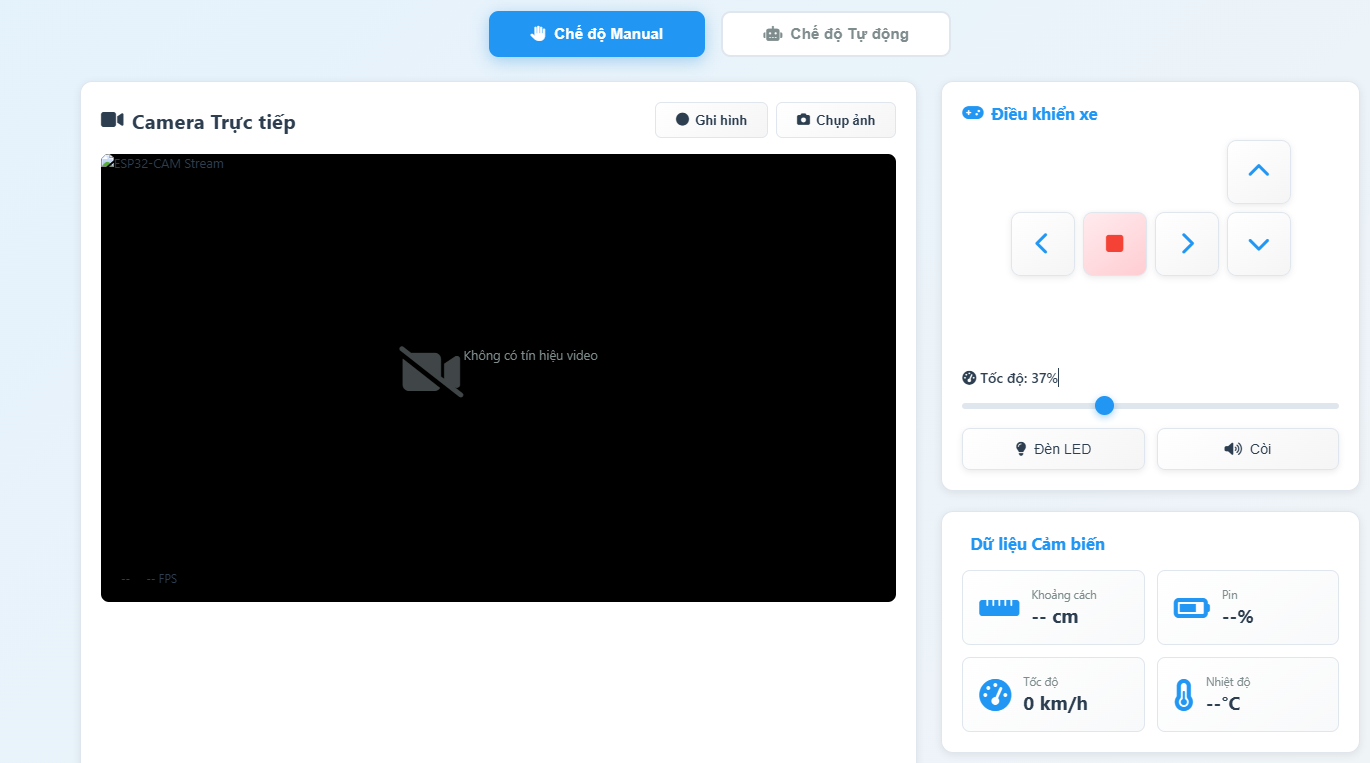
Trước khi sử dụng hệ thống, người dùng cần thực hiện đăng nhập để có thể truy cập vào hệ thống quản lý xe giám sát. Giao diện đăng nhập bao gồm:

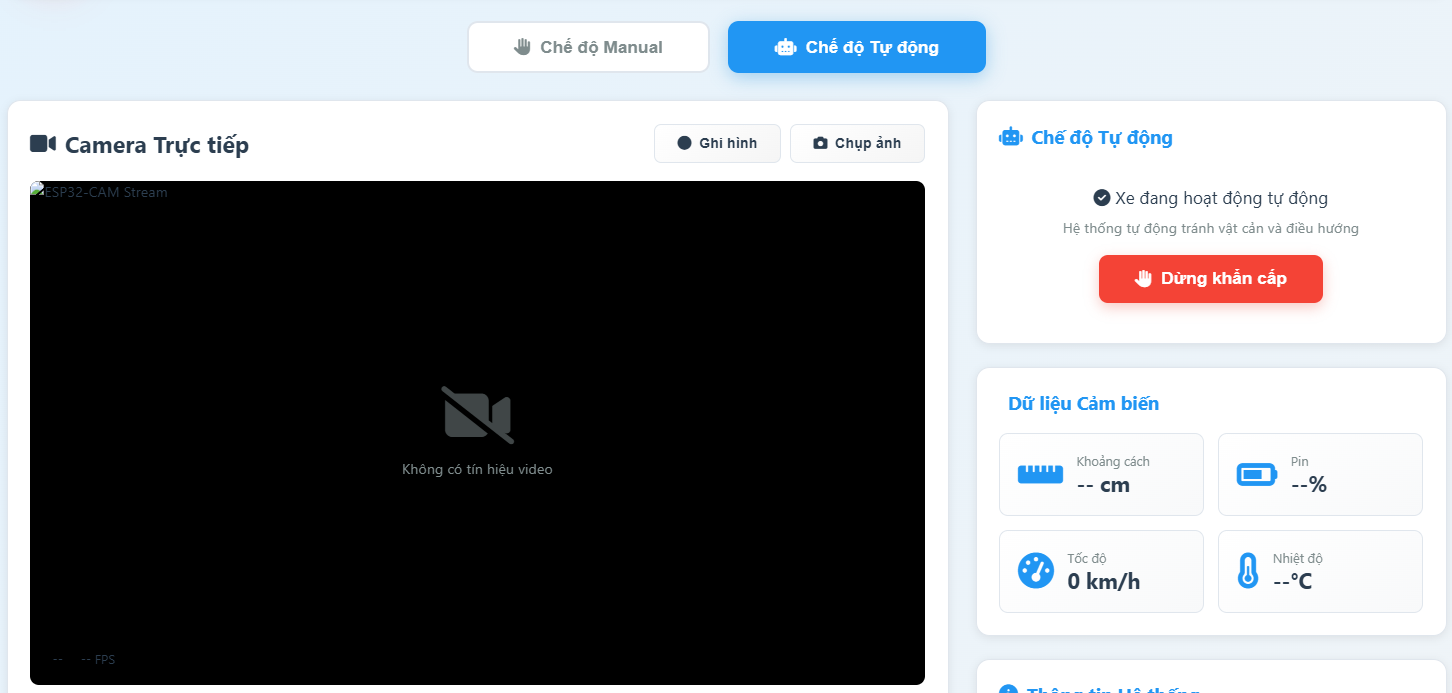
* Hai ô nhập liệu: Username (Tên đăng nhập) và Password (Mật khẩu)
* Một nút "Đăng nhập" để xác thực và truy cập vào dashboard
* Dữ liệu về tài khoản và mật khẩu được lưu trữ trong cơ sở dữ liệu MongoDB với 3 thuộc tính: id (Primary Key), username, password (được mã hóa)

Quy trình đăng nhập:

* Nếu đăng nhập thành công, hệ thống sẽ chuyển hướng tới giao diện Dashboard chính
* Nếu đăng nhập thất bại, hệ thống sẽ hiển thị thông báo "Sai tên tài khoản hoặc mật khẩu" và yêu cầu người dùng nhập lại

b) Dashboard điều khiển chính





Sau khi đăng nhập thành công, người dùng được chuyển đến trang Dashboard - trung tâm điều khiển của hệ thống. Dashboard được chia thành 3 khu vực chính:

Khu vực 1: Video Streaming

Hiển thị luồng video trực tiếp từ camera ESP32-CAM với các thông số:

* Kích thước khung hình: 640×480 pixels
* Tốc độ khung hình: Hiển thị ở góc trên trái (thường dao động 10-14 FPS)
* Trạng thái AI: Khi phát hiện hành vi nguy hiểm (vượt rào hoặc ngã), hệ thống sẽ:
  + Vẽ 17 điểm keypoints màu đỏ trên cơ thể người được phát hiện
  + Vẽ đường skeleton màu xanh lá nối các điểm khớp xương
  + Hiển thị overlay cảnh báo màu đỏ: " PHÁT HIỆN BẤT THƯỜNG!"
  + Phát âm thanh cảnh báo tự động

Khu vực 2: Bảng điều khiển

2.1. Chuyển đổi chế độ hoạt động

Toggle switch với 2 trạng thái:

* Manual (Thủ công): Người dùng điều khiển xe bằng các nút bấm
* Auto (Tự động): Xe tự động di chuyển và tránh vật cản

Khi chuyển sang chế độ Auto, các nút điều khiển sẽ bị vô hiệu hóa (disabled) và xe sẽ hoạt động theo thuật toán:

* Tiến về phía trước khi không có vật cản
* Dừng lại khi phát hiện vật cản < 30cm
* Lùi lại 0.8 giây
* Quay trái hoặc phải ngẫu nhiên
* Tiếp tục tiến

2.2. Joystick điều khiển (Chỉ hoạt động ở chế độ Manual)

Bố cục 5 nút điều khiển theo hình chữ thập:

Cách sử dụng:

* Nhấn giữ nút: Xe di chuyển theo hướng tương ứng
* Thả nút: Xe dừng lại ngay lập tức
* Hỗ trợ keyboard: Có thể sử dụng phím mũi tên ↑↓←→ trên bàn phím, phím Space để dừng

2.3. Thanh trượt điều chỉnh tốc độ

* Phạm vi: 0 - 255 (giá trị PWM)
* Giá trị mặc định: 150 (~59% công suất)
* Kéo thanh trượt → Giá trị hiển thị cập nhật ngay lập tức
* Tốc độ thay đổi sẽ áp dụng cho lệnh điều khiển tiếp theo

2.4. Panel thông số cảm biến

Hiển thị thông tin thời gian thực:

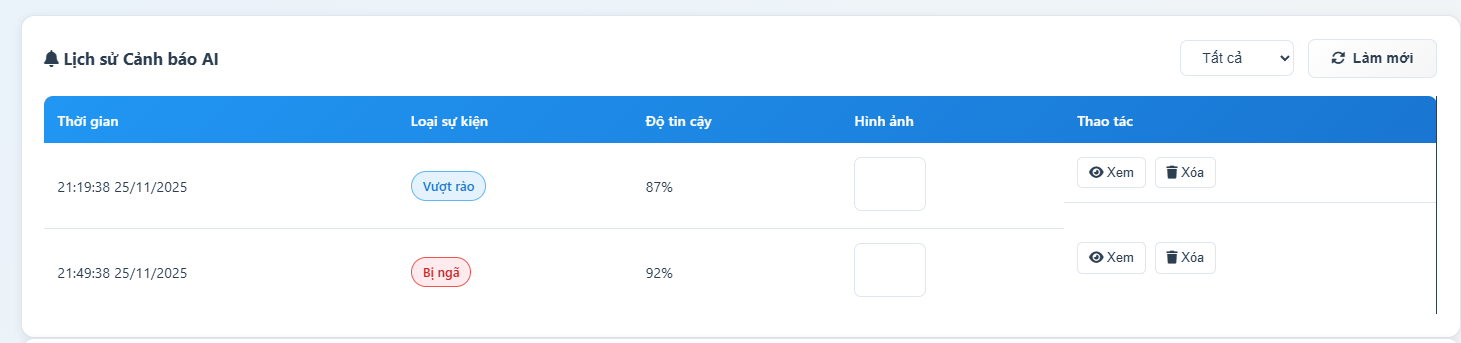
|  |  |  |
| --- | --- | --- |
| **Thông số** | **Đơn vị** | **Cập nhật** |
| Khoảng cách vật cản | cm | Mỗi 1-2 giây |
| Tốc độ hiện tại | 0-255 | Khi thay đổi |
| Trạng thái kết nối | Online/Offline | Real-time |
| Chế độ hoạt động | Manual/Auto | Khi chuyển đổi |

* Khi khoảng cách < 30cm, giá trị sẽ đổi màu đỏ cảnh báo
* Trạng thái kết nối hiển thị bằng chấm tròn: Xanh (Online) / Đỏ (Offline)

Khu vực 3: Lịch sử cảnh báo AI

Chức năng bộ lọc:

* Dropdown lọc theo loại: Tất cả / Vượt rào / Bị ngã
* Nút " Làm mới": Reload dữ liệu từ database
* Nút "Xem chi tiết": Mở modal hiển thị ảnh full size kèm tọa độ 17 keypoints
* Nút "Xóa": Xác nhận trước khi xóa vĩnh viễn khỏi database

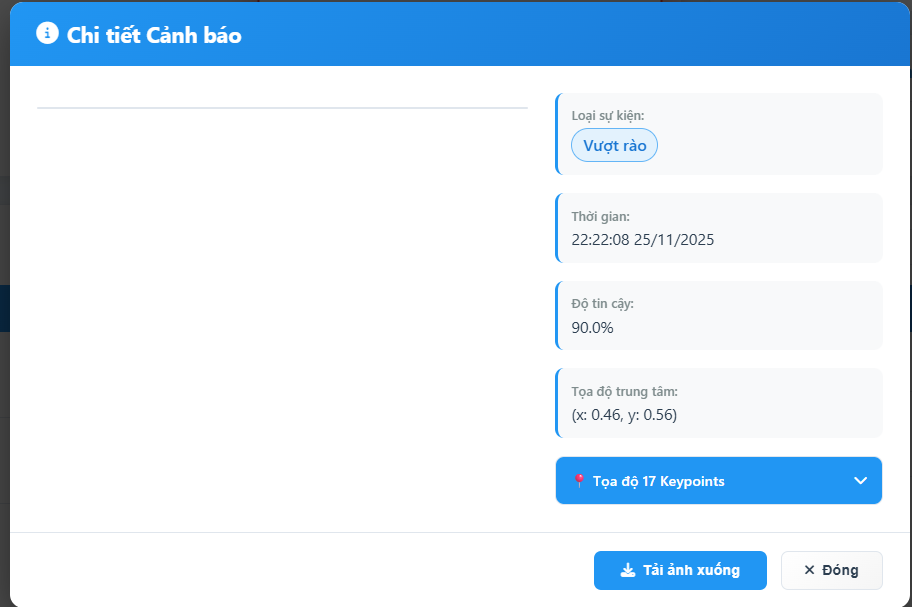
**

d) Trang chi tiết cảnh báo

Khi nhấn nút "Xem chi tiết" từ bảng lịch sử:

Hiển thị:

* Ảnh full size 640×480 pixels với tất cả keypoints và skeleton
* Thông tin chi tiết:
  + Loại sự kiện: Vượt rào / Bị ngã
  + Thời gian: 25/11/2025 lúc 17:23:05
  + Độ tin cậy: 87.3%
  + Tọa độ trung tâm cơ thể: (x: 0.45, y: 0.32)
* Bảng tọa độ 17 keypoints (có thể thu gọn/mở rộng)
* Nút [Tải ảnh xuống] và [Đóng]

**

# 5. KẾT LUẬN

## 5.1. Kết quả đạt được

### **5.1.1. Về mặt lý thuyết**

Thông qua quá trình nghiên cứu và triển khai, nhóm đã nắm vững toàn bộ kiến thức nền tảng liên quan đến hệ thống IoT, bao gồm phần cứng, truyền thông và xử lý dữ liệu. Cụ thể:

- Hiểu rõ cấu tạo và nguyên lý hoạt động của các vi điều khiển ESP32 và ESP32-CAM, bao gồm cách cấu hình Wi-Fi, xử lý GPIO, điều khiển PWM và cách đọc tín hiệu từ nhiều loại cảm biến.  
- Nắm chắc cơ chế hoạt động của cảm biến siêu âm HC-SR04, các yếu tố ảnh hưởng đến độ chính xác cũng như phương pháp lọc nhiễu khi đo khoảng cách trong môi trường có vật cản.  
- Hiểu rõ quy trình mã hóa hình ảnh MJPEG, cách ESP32-CAM thu và gửi các khung hình liên tục, đồng thời nhận thức được các giới hạn của bộ xử lý khi truyền video thời gian thực.  
- Xây dựng được một mô hình lý thuyết hoàn chỉnh về kiến trúc Client–Server, bao gồm phân tích luồng dữ liệu từ Robot đến Server và từ Server đến người dùng cuối.  
- Nắm được phương pháp áp dụng AI vào phân tích video trong thời gian thực, đặc biệt là kỹ thuật Pose Estimation để phát hiện hành vi vượt rào hoặc ngã, giúp bổ sung khía cạnh an ninh cho hệ thống.

Những hiểu biết này giúp nhóm hình thành một nền tảng vững chắc về IoT, truyền thông real-time, xử lý ảnh và tổ chức hệ thống phần mềm – phần cứng.

### **5.1.2. Về mặt thực tiễn**

Hệ thống đã được thiết kế, lắp ráp và vận hành thành công với đầy đủ các chức năng đề ra:

- Robot chạy ổn định ở hai chế độ: điều khiển thủ công và tự hành, với phản hồi nhanh, chính xác, độ trễ điều khiển duy trì dưới 200ms trong điều kiện thực tế.  
- Khả năng truyền hình ảnh trực tiếp từ ESP32-CAM thông qua Server hoạt động trơn tru. Hình ảnh được gửi liên tục giúp người dùng quan sát môi trường theo thời gian thực.  
- Cảm biến siêu âm đo khoảng cách chính xác, hỗ trợ Robot dừng khẩn cấp và tránh vật cản hiệu quả trong quá trình di chuyển.  
- Chế độ tự động vận hành đúng logic: tự động lùi – xoay – tìm đường mới khi phát hiện vật cản, giúp Robot có thể hoạt động độc lập mà không cần sự can thiệp của người dùng.  
- Hệ thống AI vận hành ổn định, phát hiện được hai tình huống quan trọng: vượt rào và ngã. Đồng thời chụp lại khung hình sự kiện và lưu vào lịch sử trong cơ sở dữ liệu, hỗ trợ công tác giám sát an ninh.  
- Dashboard Web hiển thị đầy đủ dữ liệu: video, trạng thái kết nối, điều khiển chuyển chế độ, điều chỉnh tốc độ, xem lịch sử cảnh báo,… mang lại trải nghiệm trực quan và dễ sử dụng.

Nhìn chung, đề tài không chỉ hoàn thiện về mặt kỹ thuật mà còn chứng minh được giá trị ứng dụng thực tiễn cao trong lĩnh vực giám sát, an ninh và điều khiển từ xa.

## 5.2. Hạn chế

Bên cạnh những kết quả đạt được, hệ thống vẫn còn một số điểm cần tiếp tục cải thiện:

- Việc truyền dữ liệu hình ảnh phụ thuộc nhiều vào chất lượng mạng Wi-Fi 2.4GHz. Khi tín hiệu yếu, video sẽ bị giật hoặc mất khung hình.  
- ESP32-CAM có khả năng xử lý và chất lượng camera còn hạn chế, đặc biệt là trong điều kiện ánh sáng yếu. Hệ thống chưa hỗ trợ quan sát ban đêm.  
- Thuật toán tránh vật cản của Robot mới chỉ ở mức đơn giản, chưa xử lý được môi trường phức tạp hoặc nhiều vật cản gần nhau.  
- Cảm biến siêu âm có thể sai số khi gặp vật mềm hoặc bề mặt hấp thụ sóng.  
- Server chưa hỗ trợ khả năng chịu tải cao khi nhiều người dùng cùng truy cập hoặc nhiều Robot cùng kết nối.  
- Cơ chế bảo mật hiện tại hoạt động nhưng chưa thật sự tối ưu: WebSocket chưa được mã hóa hoàn toàn, chưa có xác thực hai lớp hoặc theo dõi nhật ký bảo mật chuyên sâu.

-Ai khó khăn trong việc nhận diện và phân biệt các vật thể 3d giả con người .

Những hạn chế này tạo tiền đề để nhóm tiếp tục phát triển và hoàn thiện hệ thống trong tương lai.

## 5.3. Hướng phát triển tương lai

### **5.3.1. Về hệ thống**

- Tích hợp 4G/5G hoặc LoRa để mở rộng phạm vi hoạt động của Robot, không còn phụ thuộc vào Wi-Fi nội bộ.  
- Nâng cấp chất lượng camera, bổ sung đèn hồng ngoại hoặc sử dụng mô-đun hỗ trợ quay đêm để cải thiện khả năng giám sát trong điều kiện ánh sáng thấp.  
- Mở rộng thuật toán tự hành bằng các phương pháp nâng cao như SLAM, bản đồ môi trường hoặc AI-based Navigation để Robot có thể xử lý môi trường phức tạp và tự tìm đường tối ưu.  
- Nâng cấp khả năng xử lý video bằng chuẩn H.264 để giảm băng thông.  
- Cho phép quản lý nhiều Robot từ một Dashboard duy nhất, phù hợp cho hệ thống giám sát quy mô lớn.  
- Bổ sung chức năng cảnh báo qua ứng dụng di động hoặc SMS ngay khi phát hiện vượt rào/ngã hoặc mất kết nối.  
- Tối ưu tiêu thụ năng lượng, nghiên cứu tích hợp pin công suất lớn hoặc sạc năng lượng mặt trời.

### **5.3.2. Về bảo mật**

- Áp dụng mô hình phân quyền theo vai trò (RBAC) để đảm bảo người dùng chỉ thao tác đúng phạm vi được phép.  
- Mã hóa toàn bộ kênh truyền WebSocket bằng TLS để chống nghe lén, giả mạo lệnh điều khiển và tấn công trung gian (MITM).  
- Bổ sung xác thực hai lớp (2FA) khi đăng nhập vào hệ thống.  
- Triển khai kiểm soát truy cập nâng cao cho Robot nhằm ngăn việc gửi lệnh từ nguồn không hợp lệ.  
- Ghi nhật ký truy cập (Access Log) và phân tích bảo mật tự động để phát hiện hành vi nghi ngờ.  
- Tăng cường bảo vệ Database bằng các cơ chế mã hóa và phân quyền chi tiết.

# Bảng phân công nhiệm vụ

|  |  |
| --- | --- |
| Họ tên | Nhiệm vụ |
| Nguyễn Mạnh Cường-B22DCCN100 | Thiết kế mạch và code esp32 |
| Bùi Văn Hiến - B22DCCN292 | Thiết kế web |
| Trần Đức Phương -B22DCCN640 | Thiết kế ai |
| Đỗ Thành Đạt -B22DCCN184 | Thiết kế ai |