BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HỒ CHÍ MINH

ĐỒ ÁN MÔN HỌC

ĐỒ ÁN ROBOT

HỆ THỐNG XE THÔNG MINH TỰ HÀNH

ỨNG DỤNG CẢM BIẾN VÀ THUẬT TOÁN ĐIỀU KHIỂN

Ngành: Robot và trí tuệ nhân tạo

Lớp: 22DRTA1

Giảng viên hướng dẫn : Ths. PHẠM QUỐC PHƯƠNG

Sinh viên thực hiện:	**MSSV:**	**Lớp:**
Nguyễn Văn Đạt	2286300010	22DRTA1
Huỳnh Long	2286300028	22DRTA1
Nguyễn Chấn Huy	2286300020	22DRTA1

Tp.HCM, ngày ... tháng ... năm

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HỒ CHÍ MINH

ĐỒ ÁN MÔN HỌC

ĐỒ ÁN ROBOT

HỆ THỐNG XE THÔNG MINH TỰ HÀNH ỨNG DỤNG CẢM BIẾN VÀ THUẬT TOÁN ĐIỀU KHIỂN

Ngành: Robot và trí tuệ nhân tạo

Lớp: 22DRTA1

Giảng viên hướng dẫn : Ths. PHẠM QUỐC PHƯƠNG

Sinh viên thực hiện:	**MSSV:**	**Lớp:**
Nguyễn Văn Đạt	2286300010	22DRTA1
Huỳnh Long	2286300028	22DRTA1
Nguyễn Chấn Huy	2286300020	22DRTA1

Tp.HCM, ngày ... tháng ... năm

VIỆN KỸ THUẬT HUTECH

PHIẾU GIAO ĐỀ TÀI

TÊN MÔN HỌC : Đồ án ROBOT

NGÀNH: Robot và Trí tuệ nhân tạo

1. **Họ và tên sinh viên/ nhóm sinh viên được giao đề tài** (sĩ số trong nhóm: 03):
   (1) Nguyễn Văn Đạt .....MSSV: 2286300010..... Lớp: 22DRTA1
   (2) Huỳnh Long.....MSSV: 2286300028..... Lớp: 22DRTA1
   (3) Nguyễn Chấn Huy .....MSSV: 2286300020..... Lớp: 22DRTA1
2. **Tên đề tài**: Hệ thống xe thông minh tự hành ứng dụng cảm biến và thuật toán điều khiển.....
   .....
   .....
3. **Các dữ liệu ban đầu**:

- Kiến thức nền tảng về hệ thống điều khiển, cảm biến, robot di động và IoT.
- Linh kiện phần cứng: vi điều khiển (ESP32/Arduino), cảm biến line, cảm biến siêu âm/IR, động cơ DC/servo, nguồn pin, mạch điều khiển động cơ.
- Công cụ phần mềm: Arduino IDE, MATLAB/LabVIEW (hoặc Python), phần mềm thiết kế PCB.
- Tài liệu tham khảo về thuật toán điều khiển PID, fuzzy logic, AI trong điều khiển robot tự hành.

4. **Nội dung nhiệm vụ** :

- Nghiên cứu, thiết kế và chế tạo robot xe tự hành có khả năng di chuyển độc lập.
- Tích hợp cảm biến để robot có thể bám line hoặc tránh vật cản.
- Xây dựng thuật toán điều khiển (PID, fuzzy logic hoặc AI) cho phép robot đưa ra quyết định dựa trên dữ liệu cảm biến.
- Thiết kế cơ khí và mạch điều khiển đảm bảo gọn nhẹ, dễ mở rộng.
- Xây dựng hệ thống lưu trữ và phân tích dữ liệu hành trình để tối ưu lộ trình di chuyển.
- Nâng cao (tùy điều kiện):
  - Tích hợp camera để robot nhận diện môi trường hoặc vật thể.
  - Ứng dụng AI (Computer Vision/Deep Learning) để hỗ trợ điều hướng.

5. **Kết quả tối thiểu phải có**:
   1) Robot có khả năng tự hành theo line hoặc tránh vật cản dựa trên cảm biến.
   2) Thuật toán điều khiển được triển khai và chứng minh hoạt động ổn định.
   3) Cơ khí và mạch điều khiển hoàn chỉnh, gọn nhẹ, có khả năng mở rộng.
   4) Hệ thống lưu và phân tích dữ liệu hành trình phục vụ tối ưu di chuyển.
   5) Báo cáo đầy đủ: mô tả, thiết kế, thực nghiệm, kết quả đạt được, tài liệu tham khảo.

Ngày giao đề tài: ……./……../…… Ngày nộp báo cáo: ……./……../……

TP. HCM, ngày … tháng … năm …….

Sinh viên thực hiện
(Ký và ghi rõ họ tên các thành viên)

Nguyễn Văn Đạt

Huỳnh Long

Nguyễn Chấn Huy

Giảng viên hướng dẫn
(Ký và ghi rõ họ tên)

Phạm Quốc Phương

LỜI CẢM ƠN

Trong suốt quá trình thực hiện và hoàn thành đồ án Robot, nhóm chúng em đã nhận được rất nhiều sự hỗ trợ, định hướng và động viên quý báu. Nhìn lại dấu chân từ những ngày đầu còn loay hoay với ý tưởng, đến lúc từng chi tiết của mô hình dần hoàn thiện, chúng em càng cảm nhận rõ ràng rằng đồ án này không chỉ là kết quả của riêng nhóm, mà còn là thành quả chung của những người đã âm thầm giúp đỡ phía sau.

Trước hết, chúng em xin gửi lời cảm ơn chân thành nhất đến Thầy Phạm Quốc Phương phụ trách hướng dẫn đồ án. Thầy/Cô không chỉ giúp chúng em định hình hướng đi phù hợp, gợi ý những giải pháp kỹ thuật khả thi, mà còn kiên nhẫn góp ý từng lỗi nhỏ trong bản thiết kế, trong code và trong bản thuyết minh. Mỗi buổi trao đổi, dù ngắn hay dài, đều giúp chúng em nhìn ra những thiếu sót của mình và tiến bộ hơn một chút. Nếu không có sự hướng dẫn tận tình và nghiêm túc của Thầy/Cô, đồ án Robot của nhóm khó có thể hoàn thành theo đúng tiến độ và đạt được chất lượng như hiện tại.

Chúng em cũng xin cảm ơn các Thầy Cô trong khoa đã truyền đạt cho chúng em nền tảng kiến thức về lập trình, điều khiển, cảm biến, cơ khí… trong suốt quá trình học tập. Chính những kiến thức tưởng chừng rất "lý thuyết" đó đã trở thành công cụ để chúng em áp dụng vào thực tế khi thiết kế, lắp ráp và lập trình cho Robot. Mặc dù trong lời cảm ơn này chúng em không thể nhắc đến từng Thầy Cô một cách cụ thể, nhưng chúng em luôn trân trọng những bài giảng, những giờ thực hành và cả những lời nhắc nhở mà mình đã nhận được.

Bên cạnh đó, nhóm xin chân thành cảm ơn các anh/chị và bạn bè đã trực tiếp hỗ trợ chúng em trong quá trình thử nghiệm và hoàn thiện Robot: giúp mượn linh kiện, góp ý các lỗi phát sinh khi chạy thử, hỗ trợ kiểm tra lại các trường hợp ngoại lệ mà nhóm dễ bỏ sót. Những góp ý thẳng thắn, những giờ cùng nhau "vật lộn" trong phòng lab, hay chỉ đơn giản là động viên mỗi khi nhóm bị "tắc" ý tưởng đã giúp chúng em có thêm động lực để không bỏ cuộc giữa chừng.

Cuối cùng, chúng em muốn gửi lời cảm ơn đến gia đình – những người không trực tiếp tham gia vào chuyên môn, nhưng luôn là chỗ dựa tinh thần vững chắc. Sự tin tưởng, thông cảm khi chúng em bận rộn với đồ án, những câu hỏi đơn giản "Làm tới đâu rồi?" hay "Có cần gì không?" đã giúp chúng em cảm thấy mình không đơn độc trong hành trình này.

Vì tin rằng lời cảm ơn chỉ thật sự có ý nghĩa khi dành cho những người thực sự đóng góp vào quá trình hoàn thành đồ án, nên trong phạm vi này, chúng em chỉ xin nhắc đến những tập thể và cá nhân đã trực tiếp hỗ trợ, đồng hành cùng nhóm trong suốt thời gian qua. Một lần nữa, nhóm 3 chúng em xin chân thành tri ân tất cả những sự giúp đỡ đó.

Xin trân trọng cảm ơn.

MỤC LỤC

DANH MỤC CÁC TỪ VIẾT TẮT

Từ viết tắt	***Thuật ngữ đầy đủ (Tiếng Anh)***	***Giải thích (Tiếng Việt)***
CPU	*Central Processing Unit*	*Bộ xử lý trung tâm*
CNN	*Convolutional Neural Network*	*Mạng nơ-ron tích chập*
DDS	*Data Distribution Service*	*Giao thức phân phối dữ liệu trong ROS2*
Docker	*Docker Container Platform*	*Nền tảng ảo hóa container*
FPS	*Frames Per Second*	*Số khung hình xử lý mỗi giây*
IMU	*Inertial Measurement Unit*	*Cảm biến đo quán tính*
INT8	*Integer 8-bit*	*Định dạng số nguyên 8 bit*
IoT	*Internet of Things*	*Internet vạn vật*
LiDAR	*Light Detection and Ranging*	*Cảm biến đo khoảng cách bằng laser*
Micro-ROS	*Micro Robot Operating System*	*ROS cho vi điều khiển*
OpenCV	*Open Source Computer Vision Library*	*Thư viện xử lý ảnh mã nguồn mở*
PTZ	*Pan–Tilt–Zoom*	*Cơ cấu quay – nghiêng – phóng to camera*
QoS	*Quality of Service*	*Chất lượng dịch vụ truyền thông ROS2*
ROS	*Robot Operating System*	*Hệ điều hành cho robot*
RViz	*ROS Visualization Tool*	*Công cụ trực quan hóa trong ROS*
SFTP	*SSH File Transfer Protocol*	*Giao thức truyền file bảo mật*
SPI	*Serial Peripheral Interface*	*Chuẩn giao tiếp nối tiếp SPI*
YOLO	*You Only Look Once*	*Thuật toán nhận diện vật thể thời gian thực*
TFLite	*TensorFlow Lite*	*Định dạng mô hình AI cho thiết bị nhúng*
Teleop	*Teleoperation*	*Điều khiển từ xa*
Twist	*geometry_msgs/Twist*	*Bản tin vận tốc trong ROS*
Odometry	*Odometry*	*Ước lượng vị trí robot*

DANH MỤC CÁC BẢNG VÀ BIỂU ĐỒ, HÌNH ẢNH

CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI

1.1 Đặt vấn đề

Trong bối cảnh cuộc Cách mạng công nghiệp lần thứ tư, robot và trí tuệ nhân tạo (AI) đang trở thành những công nghệ nòng cốt thúc đẩy quá trình tự động hóa và thông minh hóa trong nhiều lĩnh vực như sản xuất công nghiệp, logistics, giao thông và dịch vụ. Đặc biệt, các hệ thống robot di động tích hợp thị giác máy tính (Computer Vision) cho phép robot không chỉ di chuyển mà còn "nhìn", "hiểu" và phản ứng linh hoạt với môi trường xung quanh. Đây là nền tảng quan trọng cho sự phát triển của các hệ thống xe tự hành, robot dịch vụ và robot thông minh trong tương lai.

Tuy nhiên, việc triển khai các thuật toán trí tuệ nhân tạo, đặc biệt là Deep Learning, trên các nền tảng phần cứng nhúng như Raspberry Pi vẫn gặp nhiều thách thức do hạn chế về tài nguyên tính toán, bộ nhớ và năng lượng. Bên cạnh đó, yêu cầu về thời gian thực, độ ổn định và khả năng mở rộng của hệ thống robot đòi hỏi một kiến trúc phần mềm phù hợp. Xuất phát từ thực tiễn đó, đề tài "Xây dựng hệ thống Robot Vision trên nền tảng MicroROS – Raspberry Pi 5 – ROS2" được lựa chọn nhằm nghiên cứu và triển khai một hệ thống robot xe tích hợp AI nhận diện hình ảnh, đáp ứng yêu cầu vận hành ổn định trên thiết bị nhúng và có khả năng mở rộng cho các bài toán robot tự hành trong tương lai.

1.2 Mục tiêu nghiên cứu

Mục tiêu tổng quát của đề tài là xây dựng và triển khai thành công một hệ thống robot xe tích hợp thị giác máy tính hoạt động trên nền tảng Raspberry Pi 5 chạy ROS2, có khả năng thu nhận hình ảnh từ camera, xử lý bằng mô hình Deep Learning và hiển thị kết quả nhận diện theo thời gian thực. Hệ thống hướng tới việc đảm bảo tính ổn định, hiệu năng phù hợp và khả năng tái lập môi trường triển khai trên phần cứng nhúng.

Cụ thể, đề tài đặt ra các mục tiêu sau: (1) xây dựng kiến trúc phần mềm robot dựa trên ROS2 với các node độc lập theo mô hình publish–subscribe; (2) triển khai mô hình YOLOv8 cho bài toán nhận diện vật thể trên Raspberry Pi 5; (3) xây dựng môi trường Docker nhằm đảm bảo tính đồng bộ và tránh xung đột thư viện; (4) nghiên cứu và áp dụng các kỹ thuật tối ưu hiệu năng để đáp ứng yêu cầu thời gian thực; và (5) tích hợp điều khiển robot thông qua tay cầm (teleoperation), tạo nền tảng cho việc phát triển các chức năng tự hành dựa trên kết quả nhận diện trong các nghiên cứu tiếp theo

1.3 Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu của đề tài là hệ thống robot xe di động tích hợp thị giác máy tính, được xây dựng trên nền tảng phần cứng MicroROS–Pi5 ROS2 Robot CarRobot Car. Hệ thống sử dụng Raspberry Pi 5 làm bộ xử lý trung tâm, camera USB/CSI để thu thập dữ liệu hình ảnh, khung xe robot thương mại và môi trường phần mềm ROS2 Humble chạy trong Docker Container. Mô hình trí tuệ nhân tạo được sử dụng trong nghiên cứu là YOLOv8 Nano, phù hợp với đặc thù phần cứng nhúng.

Về phạm vi nghiên cứu, đề tài tập trung vào các nội dung chính gồm: thiết kế kiến trúc phần mềm ROS2, triển khai và tối ưu mô hình AI nhận diện vật thể, xây dựng môi trường Docker cho robot và điều khiển xe thông qua tay cầm. Các nội dung như điều khiển tự hành hoàn toàn, lập bản đồ hay định vị robot chưa được triển khai trong phạm vi đề tài này, nhưng được định hướng là hướng phát triển trong tương lai dựa trên nền tảng hệ thống đã xây dựng.

1.4 Phương pháp thực hiện

Đề tài được thực hiện theo phương pháp nghiên cứu kết hợp giữa lý thuyết và thực nghiệm. Trước hết, nhóm tiến hành thu thập và nghiên cứu các tài liệu liên quan đến robot di động, ROS2, Micro-ROS, Computer Vision và các mô hình Deep Learning cho nhận diện hình ảnh. Trên cơ sở đó, các kiến thức được tổng hợp và phân tích nhằm lựa chọn giải pháp kỹ thuật phù hợp với điều kiện phần cứng và mục tiêu nghiên cứu.

Tiếp theo, hệ thống được thiết kế và triển khai theo từng giai đoạn, bao gồm xây dựng kiến trúc phần mềm ROS2, cấu hình môi trường Docker, triển khai mô hình YOLOv8 và tối ưu hiệu năng trên Raspberry Pi 5. Quá trình thực nghiệm được tiến hành thông qua việc đo đạc các chỉ số như tốc độ xử lý (FPS), độ trễ và mức độ ổn định của hệ thống. Cuối cùng, kết quả thực nghiệm được phân tích và đánh giá nhằm rút ra nhận xét, đồng thời đề xuất các hướng cải tiến và phát triển hệ thống trong tương lai.

CHƯƠNG 2: TỔNG QUAN HỆ THỐNG

2.1 Tổng quan robot xe tự hành tích hợp thị giác máy tính

Robot xe tự hành tích hợp thị giác máy tính là một dạng robot di động có khả năng thu nhận và xử lý thông tin hình ảnh từ môi trường xung quanh nhằm hỗ trợ quá trình ra quyết định và điều hướng. Không giống các robot tự hành truyền thống chỉ dựa vào cảm biến khoảng cách hoặc cảm biến quang đơn giản, robot tích hợp thị giác máy tính có thể nhận diện vật thể, phân tích ngữ cảnh và hiểu môi trường ở mức độ cao hơn. Đây là nền tảng quan trọng cho các ứng dụng robot thông minh như xe tự lái, robot giao hàng, robot dịch vụ và robot giám sát.

Trong các hệ thống robot hiện đại, thị giác máy tính thường được kết hợp với các thuật toán trí tuệ nhân tạo để xử lý dữ liệu hình ảnh theo thời gian thực. Việc tích hợp thị giác máy tính vào robot xe giúp mở rộng đáng kể khả năng ứng dụng của hệ thống, từ các bài toán cơ bản như phát hiện vật cản đến các bài toán nâng cao như nhận diện làn đường, theo dõi mục tiêu hoặc tương tác với con người. Do đó, robot xe tự hành tích hợp thị giác máy tính được xem là xu hướng phát triển tất yếu trong lĩnh vực robot di động và tự động hóa thông minh.

2.2 Nền tảng phần cứng MicroROS – Raspberry Pi 5 – ROS2

Hình 2.1. Model MicroROS-Pi5 ROS2 Robot Car

Nền tảng phần cứng MicroROS–Pi5 ROS2 Robot Car được thiết kế theo kiến trúc phân tầng nhằm kết hợp hiệu quả giữa xử lý thời gian thực và xử lý mức cao. Trong kiến trúc này, Raspberry Pi 5 đóng vai trò là bộ xử lý trung tâm, đảm nhiệm các tác vụ tính toán phức tạp như xử lý hình ảnh, suy luận mô hình Deep Learning và quản lý hệ

thống ROS2. Với CPU ARM Cortex-A76 hiệu năng cao, Raspberry Pi 5 đáp ứng tốt yêu cầu triển khai các ứng dụng robot vision trên thiết bị nhúng.

Micro-ROS được sử dụng để kết nối các vi điều khiển vào hệ sinh thái ROS2, cho phép các tác vụ điều khiển mức thấp và thời gian thực như đọc cảm biến, điều khiển động cơ được thực hiện hiệu quả. ROS2 đóng vai trò là nền tảng phần mềm trung gian, cung cấp cơ chế giao tiếp, quản lý node và mở rộng hệ thống. Sự kết hợp giữa Micro-ROS, Raspberry Pi 5 và ROS2 tạo nên một nền tảng linh hoạt, dễ mở rộng và phù hợp cho các hệ thống robot xe tích hợp thị giác máy tính.

2.3 Tổng quan Computer Vision và Deep Learning trong robot

Computer Vision là lĩnh vực nghiên cứu cho phép máy móc thu nhận, xử lý và phân tích thông tin từ hình ảnh hoặc video. Trong robot di động, Computer Vision đóng vai trò quan trọng trong việc giúp robot "nhìn" và hiểu môi trường, từ đó đưa ra các quyết định điều hướng phù hợp. Các bài toán Computer Vision phổ biến trong robot bao gồm phát hiện vật thể, theo dõi chuyển động, nhận diện làn đường và ước lượng không gian.

Sự phát triển của Deep Learning đã mang lại bước tiến lớn cho Computer Vision, đặc biệt là với các mô hình mạng nơ-ron tích chập (CNN). Các mô hình Deep Learning cho phép robot học trực tiếp từ dữ liệu hình ảnh và đạt độ chính xác cao hơn so với các phương pháp xử lý ảnh truyền thống. Tuy nhiên, việc triển khai Deep Learning trên robot đòi hỏi phải cân nhắc đến hiệu năng tính toán, độ trễ và tài nguyên phần cứng, đặc biệt khi triển khai trên các nền tảng nhúng như Raspberry Pi.

2.4 Mô hình YOLO và khả năng ứng dụng trong robot di động

YOLO (You Only Look Once) là một trong những mô hình nhận diện vật thể theo thời gian thực nổi bật trong lĩnh vực Computer Vision. Khác với các phương pháp nhận diện truyền thống thực hiện nhiều bước xử lý, YOLO tiếp cận bài toán theo hướng end-to-end, cho phép phát hiện và phân loại vật thể chỉ trong một lần suy luận. Nhờ đó, YOLO có tốc độ xử lý nhanh và phù hợp với các ứng dụng yêu cầu thời gian thực như robot di động.

Trong robot xe tích hợp thị giác máy tính, YOLO có thể được sử dụng để nhận diện vật cản, con người hoặc các đối tượng quan trọng trong môi trường hoạt động. Phiên bản YOLOv8 Nano được lựa chọn trong đề tài do có kích thước mô hình nhỏ và tốc độ xử lý phù hợp với phần cứng Raspberry Pi 5. Việc ứng dụng YOLO trong robot di động

không chỉ nâng cao khả năng nhận thức môi trường mà còn tạo tiền đề cho các chức năng tự hành thông minh trong tương lai.

2.5 ROS2 và mô hình truyền thông Publish–Subscribe

ROS2 (Robot Operating System 2) là nền tảng phần mềm được thiết kế nhằm hỗ trợ xây dựng các hệ thống robot phức tạp với kiến trúc phân tán. ROS2 sử dụng mô hình truyền thông Publish–Subscribe, trong đó các node giao tiếp với nhau thông qua các topic mà không cần biết trực tiếp vị trí hoặc trạng thái của node khác. Cơ chế này giúp hệ thống có tính linh hoạt cao, dễ mở rộng và dễ bảo trì.

Trong hệ thống robot vision, mô hình Publish–Subscribe cho phép tách biệt các chức năng như thu thập hình ảnh, xử lý AI và điều khiển robot thành các node độc lập. Dữ liệu hình ảnh, kết quả nhận diện và lệnh điều khiển được truyền qua các topic một cách đồng bộ và hiệu quả. Nhờ đó, ROS2 trở thành nền tảng phù hợp để triển khai các hệ thống robot xe tích hợp thị giác máy tính, đặc biệt trong các ứng dụng yêu cầu khả năng mở rộng và tích hợp nhiều công nghệ khác nhau.

CHƯƠNG 3: KIẾN TRÚC HỆ THỐNG VÀ THIẾT KẾ PHẦN CỨNG

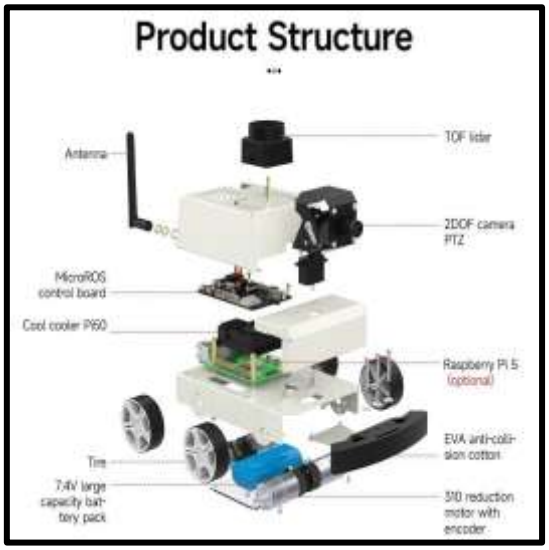
3.1 Phân tích bài toán và yêu cầu hệ thống

Bài toán đặt ra trong đề tài là xây dựng một hệ thống robot xe tích hợp thị giác máy tính (Robot Vision) có khả năng thu nhận hình ảnh từ môi trường, xử lý nhận diện vật thể bằng trí tuệ nhân tạo và hiển thị kết quả theo thời gian thực trên nền tảng phần cứng nhúng. Hệ thống cần đảm bảo khả năng vận hành ổn định trên Raspberry Pi 5 – một thiết bị có tài nguyên tính toán giới hạn so với máy tính cá nhân – đồng thời vẫn đáp ứng yêu cầu về tốc độ xử lý và độ trễ để phục vụ cho các ứng dụng robot di động.

Từ bài toán trên, hệ thống được đặt ra các yêu cầu chính gồm: (i) khả năng thu nhận hình ảnh liên tục từ camera; (ii) triển khai mô hình AI nhận diện vật thể hoạt động ổn định trên thiết bị nhúng; (iii) tổ chức phần mềm theo kiến trúc mô-đun, dễ mở rộng và bảo trì; (iv) hỗ trợ giao tiếp và tích hợp với các module điều khiển robot; và (v) có khả năng mở rộng để phát triển các chức năng tự hành trong tương lai. Những yêu cầu này là cơ sở cho việc xây dựng kiến trúc hệ thống và lựa chọn phần cứng phù hợp.

3.2 Kiến trúc tổng thể hệ thống Robot Vision

Hệ thống Robot Vision được thiết kế theo kiến trúc phân tầng, kết hợp giữa phần cứng nhúng và nền tảng phần mềm ROS2. Ở tầng thu thập dữ liệu, camera đảm nhiệm việc ghi nhận hình ảnh môi trường và truyền dữ liệu hình ảnh vào hệ thống. Ở tầng xử lý, Raspberry Pi 5 chạy ROS2 và mô hình Deep Learning thực hiện các tác vụ xử lý ảnh, suy luận AI và quản lý luồng dữ liệu. Ở tầng ứng dụng, kết quả nhận diện được sử dụng cho mục đích hiển thị, giám sát và làm tiền đề cho các thuật toán điều khiển robot.

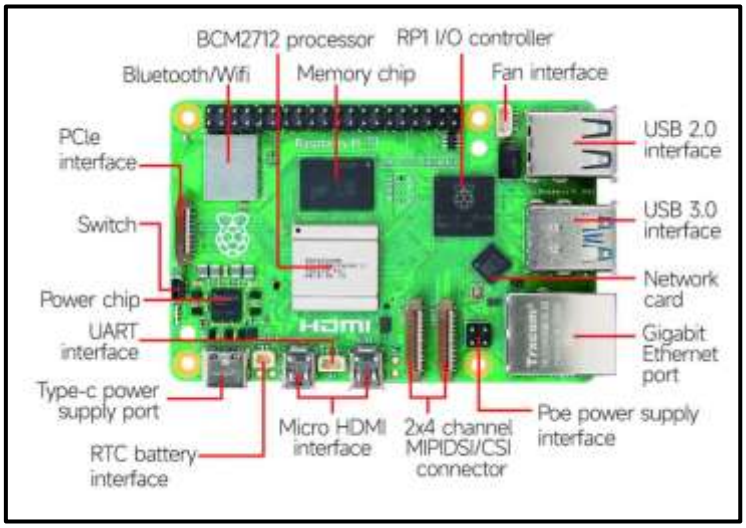


Hình 3.1. Mô hình tổng thể của robot MicroROS-Pi5 ROS2 Raspberry Pi 5

Kiến trúc hệ thống sử dụng mô hình **Publish–Subscribe của ROS2**, trong đó các chức năng được chia thành các node độc lập như camera node, AI inference node và các node điều khiển. Cách tiếp cận này giúp hệ thống đạt được tính mô-đun cao, dễ mở rộng và cho phép thay thế hoặc nâng cấp từng thành phần mà không ảnh hưởng đến toàn bộ hệ thống. Đây là kiến trúc phù hợp cho các hệ thống robot hiện đại, đặc biệt là các hệ thống tích hợp AI và thị giác máy tính.

3.3 nền tảng MicroROS–Pi5 ROS2 Robot Car

3.3.1 Khối xử lý trung tâm Raspberry Pi 5



Hình 2.2. Vi điều khiển Raspberry Pi 5

Raspberry Pi 5 được lựa chọn làm khối xử lý trung tâm của hệ thống Robot Vision nhờ hiệu năng tính toán vượt trội so với các thế hệ trước. Với CPU ARM Cortex-A76 đa nhân, Raspberry Pi 5 có khả năng xử lý các tác vụ nặng như xử lý ảnh, suy luận mô hình Deep Learning và quản lý hệ thống ROS2. Thiết bị này đóng vai trò trung tâm trong việc điều phối hoạt động của toàn bộ hệ thống robot.

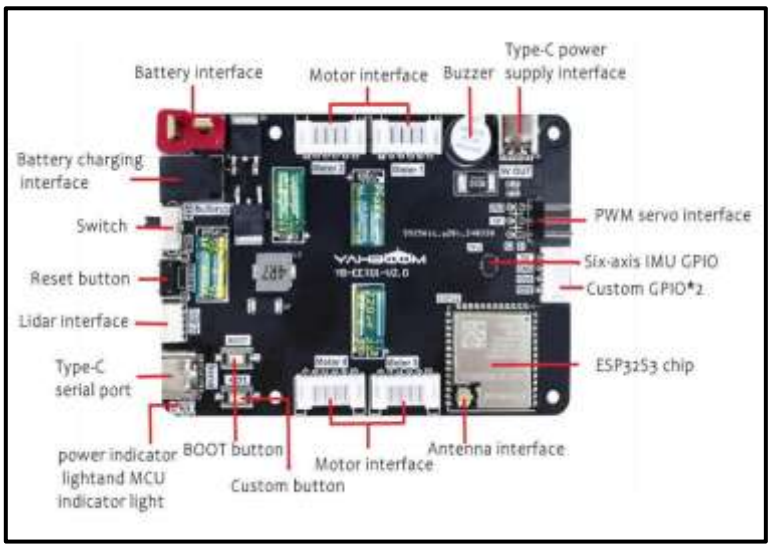
Ngoài ra, Raspberry Pi 5 hỗ trợ nhiều giao tiếp ngoại vi như USB, CSI, Ethernet và WiFi, cho phép kết nối linh hoạt với camera, mạng và các thiết bị ngoại vi khác. Việc triển khai ROS2 và Docker trên Raspberry Pi 5 giúp hệ thống đạt được tính ổn định, đồng bộ môi trường và dễ dàng tái lập khi cần thiết, đặc biệt quan trọng trong các hệ thống robot nghiên cứu và giáo dục.

3.3.2 Khối điều khiển động cơ ESP32

ESP32 được sử dụng làm khối điều khiển động cơ cấp thấp trong hệ thống robot xe. Khối này đảm nhiệm các tác vụ điều khiển thời gian thực như điều khiển tốc độ, chiều

quay của động cơ DC và thu thập dữ liệu trạng thái (encoder, dòng điện hoặc tín hiệu phản hồi nếu có). Việc tách khối điều khiển động cơ ra khỏi Raspberry Pi 5 giúp giảm tải cho khối xử lý trung tâm và đảm bảo tính ổn định cho các tác vụ điều khiển có yêu cầu thời gian thực cao.

ESP32 giao tiếp với Raspberry Pi 5 thông qua nền tảng micro-ROS, cho phép ESP32 hoạt động như một node trong hệ sinh thái ROS2. Các lệnh điều khiển vận tốc (ví dụ: /cmd_vel) được xuất bản từ ROS2 trên Raspberry Pi 5 và truyền xuống ESP32 để thực thi điều khiển động cơ. Ngược lại, ESP32 có thể gửi các dữ liệu phản hồi (vận tốc bánh xe, encoder, trạng thái động cơ) lên ROS2 để phục vụ giám sát và phát triển các thuật toán điều khiển nâng cao trong tương lai.



Hình 3.3. Board mạch Pcb ESP32-S3 tích hợp điều khiển động cơ và cảm biến

Việc sử dụng ESP32 mang lại các ưu điểm như: khả năng xử lý nhanh, tích hợp WiFi/Bluetooth, tiêu thụ năng lượng thấp và dễ dàng mở rộng thêm các cảm biến hoặc cơ cấu chấp hành khác. Đây là kiến trúc phổ biến và hiệu quả trong các hệ thống robot di động hiện đại

3.3.2 Camera USB/CSI

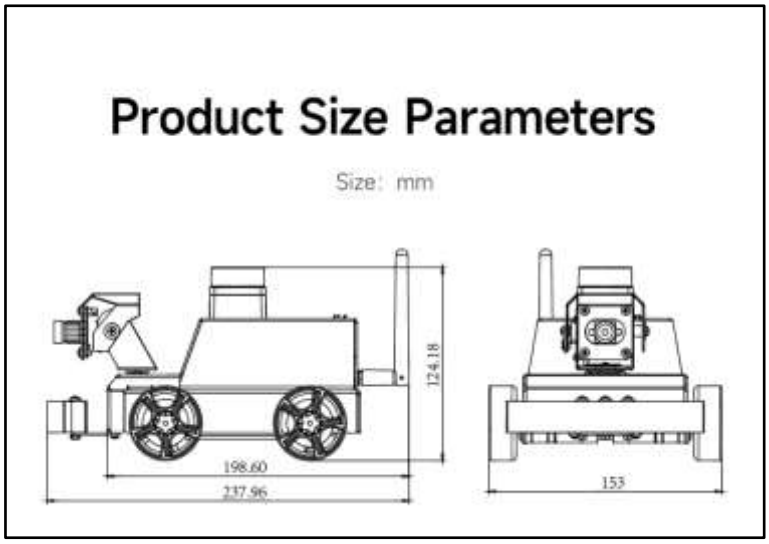
Camera là thành phần quan trọng trong hệ thống Robot Vision, chịu trách nhiệm thu thập dữ liệu hình ảnh từ môi trường xung quanh robot. Trong đề tài, camera USB hoặc CSI được sử dụng nhờ tính phổ biến, dễ tích hợp và tương thích tốt với Raspberry Pi 5. Camera cung cấp luồng dữ liệu hình ảnh liên tục, phục vụ trực tiếp cho các thuật toán xử lý ảnh và nhận diện vật thể.

Hình 3.4.Camera AI nhận dạng hình ảnh và chuyển động 2DOF

Việc lựa chọn camera USB/CSI cho phép linh hoạt trong cấu hình hệ thống và dễ dàng thay thế hoặc nâng cấp khi cần thiết. Dữ liệu hình ảnh từ camera được truyền vào ROS2 thông qua các node chuyên dụng, đảm bảo tính chuẩn hóa và thuận lợi cho việc xử lý ở các tầng tiếp theo của hệ thống.

3.3.3 Khung xe Yahboom và cơ cấu truyền động

Khung xe Yahboom được sử dụng làm nền tảng cơ khí cho hệ thống robot nhờ thiết kế gọn nhẹ, chắc chắn và phù hợp cho các ứng dụng robot di động. Khung xe cho phép lắp đặt Raspberry Pi 5, camera và các linh kiện điện tử khác một cách thuận tiện, đồng thời đảm bảo độ ổn định khi robot di chuyển.



Hình 3.5. Khung xe và kích thước của xe Raspberry PI 5

Cơ cấu truyền động của xe bao gồm các động cơ DC và hệ thống bánh xe, cho phép robot thực hiện các chuyển động tiến, lùi và quay tại chỗ. Mặc dù trong phạm vi đề tài, hệ thống chủ yếu tập trung vào Robot Vision và điều khiển bằng tay cầm, nhưng cơ cấu truyền động này đóng vai trò quan trọng trong việc tạo nền tảng cho các nghiên cứu và ứng dụng tự hành trong các giai đoạn phát triển tiếp theo.

3.4 Thiết kế cơ khí và bố trí linh kiện

Thiết kế cơ khí và bố trí linh kiện được thực hiện nhằm đảm bảo tính ổn định, cân bằng và thuận tiện cho việc vận hành hệ thống robot. Raspberry Pi 5 và các module điện tử được bố trí ở vị trí trung tâm của khung xe để giảm rung lắc và bảo vệ linh kiện trong quá trình di chuyển. Camera được đặt ở vị trí phía trước và có góc nhìn phù hợp để tối ưu khả năng quan sát môi trường.

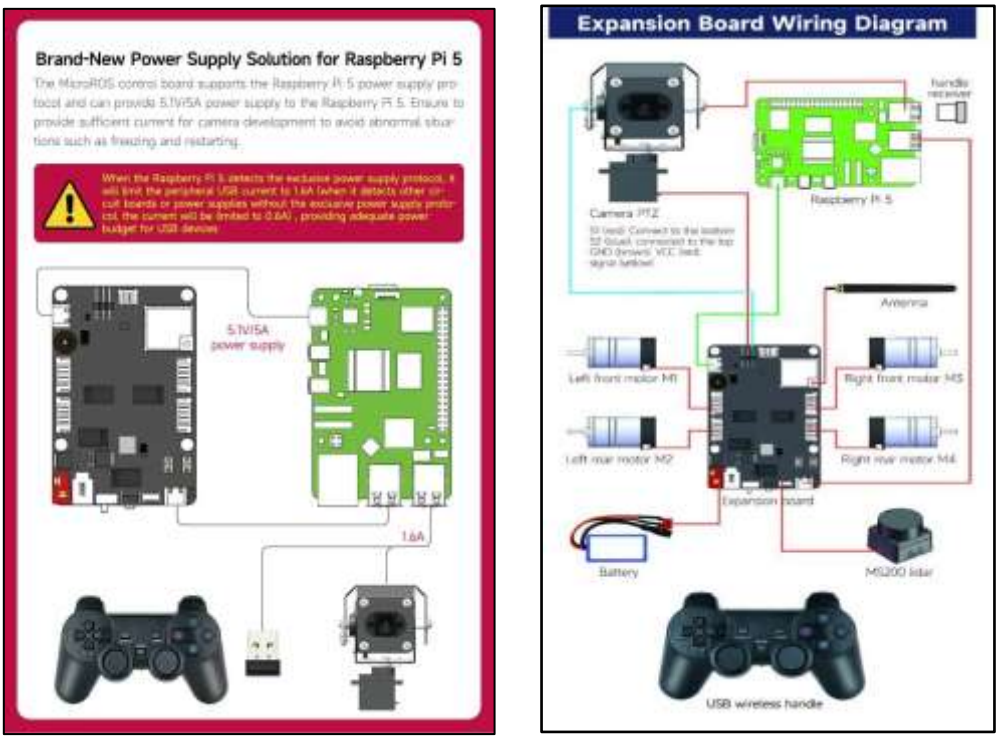
Việc bố trí linh kiện còn chú trọng đến khả năng tản nhiệt, đi dây gọn gàng và dễ dàng tháo lắp khi cần bảo trì hoặc nâng cấp. Thiết kế cơ khí hợp lý giúp hệ thống robot vận hành ổn định hơn, đồng thời tạo điều kiện thuận lợi cho việc mở rộng hệ thống trong tương lai.

3.5 Sơ đồ kết nối và mạch điện hệ thống

3.5.1 Sơ đồ kết nối tổng thể và sơ đồ nguồn hệ thống

Sơ đồ kết nối tổng thể và sơ đồ nguồn hệ thống mô tả mối quan hệ giữa các khối phần cứng chính, bao gồm Raspberry Pi 5, ESP32, camera, joystick, driver động cơ và hệ thống nguồn. Trong hệ thống, Raspberry Pi 5 đóng vai trò xử lý trung tâm, chạy ROS2 và thực hiện các thuật toán nhận diện làn đường dựa trên dữ liệu hình ảnh thu thập từ camera.

Camera USB/CSI được kết nối trực tiếp với Raspberry Pi 5 để truyền dữ liệu hình ảnh phục vụ cho bài toán thị giác máy tính. Raspberry Pi 5 giao tiếp với ESP32 thông qua kết nối USB hoặc UART, cho phép truyền dữ liệu giám sát và sẵn sàng mở rộng sang các chế độ điều khiển tự động trong tương lai. Trong phạm vi đề tài, việc điều khiển chuyển động chủ yếu được thực hiện thông qua joystick kết nối với ESP32.



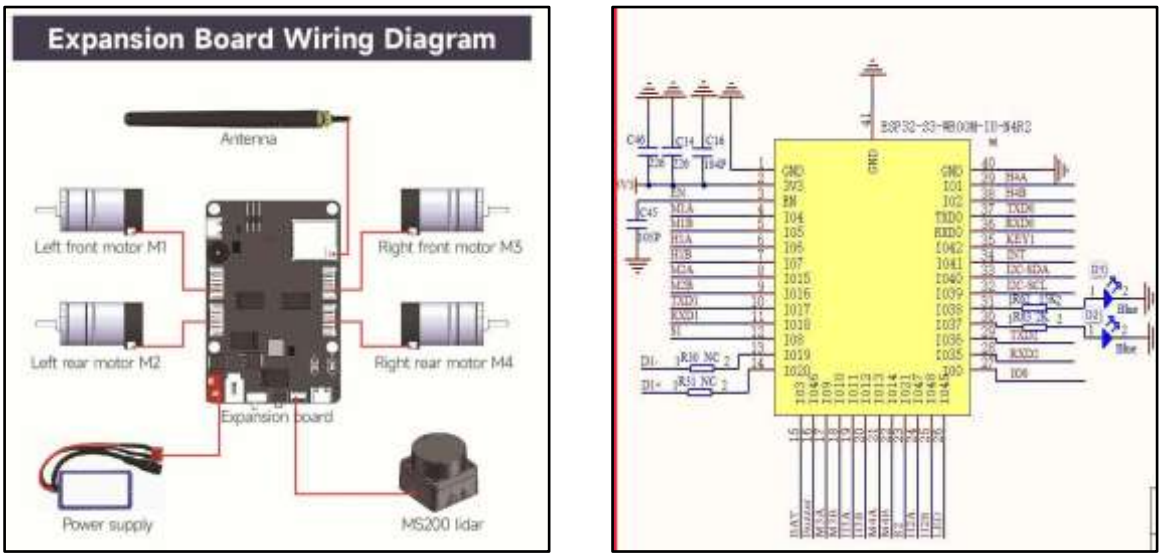
Hình 3.6. Sơ đồ kết nối tổng thể và sơ đồ nguồn hệ thống robot

ESP32 nhận tín hiệu điều khiển từ joystick, xử lý dữ liệu đầu vào và xuất các tín hiệu điều khiển đến driver động cơ. Driver động cơ cung cấp dòng công suất để điều khiển 4 động cơ DC có encoder, đảm bảo robot có thể di chuyển theo lệnh điều khiển thủ công.

Hệ thống sử dụng nguồn pin DC (Li-ion hoặc LiPo) làm nguồn cấp chính. Nguồn này được phân phối thông qua các bộ chuyển đổi DC–DC để tạo ra các mức điện áp phù hợp cho từng khối chức năng: 5V cho Raspberry Pi 5, 5V/3.3V cho ESP32 và joystick, và nguồn công suất riêng cho driver động cơ. Việc tách nguồn logic và nguồn động cơ giúp hạn chế nhiễu điện và đảm bảo sự ổn định cho hệ thống trong quá trình vận hành.

3.5.2 Sơ đồ mạch điện điều khiển động cơ sử dụng ESP32

Sơ đồ mạch điện điều khiển động cơ trong đề tài được xây dựng dựa trên bo mạch tích hợp ESP32-S3, trong đó mạch công suất điều khiển động cơ đã được tích hợp trực tiếp trên PCB, không sử dụng driver động cơ rời. Bo mạch cho phép điều khiển trực tiếp 4 động cơ DC có encoder, phù hợp cho các ứng dụng robot xe di động.



Hình 3.7. Sơ đồ mạch điện điều khiển động cơ sử dụng bo mạch ESP32-S3 tích hợp mạch công suất

ESP32-S3 đóng vai trò điều khiển trung tâm, thực hiện việc đọc tín hiệu từ joystick và sinh các tín hiệu PWM điều khiển tốc độ cùng tín hiệu logic điều khiển chiều quay động cơ. Quá trình khuếch đại dòng và cấp nguồn cho động cơ được thực hiện bởi mạch công suất tích hợp trên bo mạch.

Các cổng kết nối Motor 1 đến Motor 4 cho phép điều khiển độc lập từng động cơ. Tín hiệu encoder từ các động cơ DC được đưa trực tiếp về ESP32-S3 thông qua các chân GPIO, phục vụ giám sát trạng thái chuyển động của robot. ESP32-S3 đồng thời giao tiếp với Raspberry Pi 5 thông qua USB hoặc UART và nền tảng micro-ROS, cho phép truyền dữ liệu động cơ lên hệ thống ROS2 để theo dõi và mở rộng các thuật toán điều khiển trong tương lai.

Thiết kế mạch tích hợp giúp hệ thống gọn nhẹ, ổn định và phù hợp cho các ứng dụng robot di động.

CHƯƠNG 4: THIẾT KẾ PHẦN MỀM ROS2 VÀ MÔI TRƯỜNG DOCKER

4.1 Kiến trúc Node trong ROS2

Hệ thống phần mềm Robot Vision được xây dựng trên nền tảng ROS2 Humble Hawksbill, trong đó các chức năng được tổ chức dưới dạng các node độc lập và giao tiếp với nhau thông qua mô hình truyền thông publish–subscribe. Việc phân tách hệ thống thành các node riêng biệt giúp giảm sự phụ thuộc giữa các thành phần, tăng tính mô-đun và thuận lợi cho việc phát triển, bảo trì cũng như mở rộng hệ thống.

Hình 4.1. Sơ đồ khối thuật toán hệ thống

Trong kiến trúc này, mỗi node đảm nhiệm một nhiệm vụ cụ thể như thu nhận hình ảnh, xử lý nhận diện vật thể hoặc hiển thị kết quả. Các node có thể được triển khai độc lập trong môi trường Docker nhưng vẫn giao tiếp liền mạch thông qua ROS2, tạo nên một hệ thống phần mềm linh hoạt và dễ mở rộng cho robot di động tích hợp thị giác máy tính.

4.1.1 Camera Node

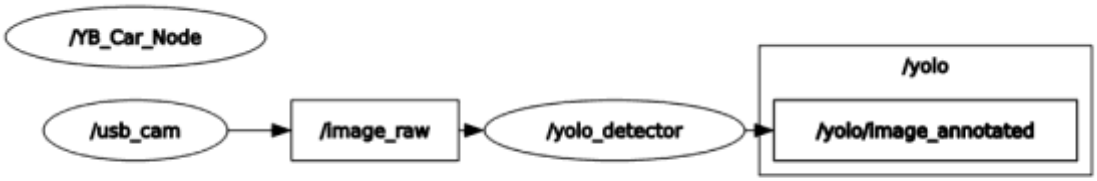
Camera Node có nhiệm vụ thu thập dữ liệu hình ảnh thô từ camera USB hoặc CSI được kết nối với Raspberry Pi 5. Node này sử dụng các driver camera tương thích với ROS2 để đọc dữ liệu hình ảnh theo thời gian thực và đóng gói dữ liệu dưới dạng bản tin sensor_msgs/Image. Đây là nguồn dữ liệu đầu vào chính cho toàn bộ hệ thống Robot Vision

Để giảm tải cho hệ thống và tối ưu hiệu năng, Camera Node được cấu hình sử dụng định dạng nén như MJPEG ngay tại tầng phần cứng, sau đó chuyển đổi sang định dạng màu chuẩn (BGR8) trước khi publish lên topic ROS2. Dữ liệu hình ảnh được publish

liên tục lên topic (ví dụ: /image_raw), cho phép các node xử lý phía sau có thể subscribe và sử dụng mà không phụ thuộc trực tiếp vào phần cứng camera.

4.1.2 YOLO Detection Node

YOLO Detection Node là node chịu trách nhiệm xử lý nhận diện vật thể bằng mô hình Deep Learning YOLOv8. Node này subscribe dữ liệu hình ảnh từ topic do Camera Node cung cấp, sau đó thực hiện các bước tiền xử lý, suy luận mô hình và hậu xử lý để xác định vị trí, nhãn và độ tin cậy của các vật thể được phát hiện trong khung hình.



Hình 4.2. YOLOv8 node

Kết quả nhận diện được vẽ trực tiếp lên hình ảnh gốc dưới dạng bounding box và nhãn đối tượng, sau đó được publish lên một topic khác (ví dụ: /yolo/image_annotated) để phục vụ cho việc hiển thị hoặc các mục đích xử lý tiếp theo. Việc tách riêng YOLO Detection Node giúp hệ thống dễ dàng thay thế mô hình AI khác hoặc mở rộng thêm các chức năng xử lý thị giác mà không ảnh hưởng đến Camera Node.

4.2 Xây dựng môi trường Docker cho robot

Việc triển khai các thư viện AI và ROS2 trực tiếp trên hệ điều hành gốc của Raspberry Pi có thể gây ra xung đột thư viện và khó khăn trong việc tái lập môi trường. Do đó, đề tài sử dụng **Docker Container** để xây dựng một môi trường phần mềm cô lập, đảm bảo tính đồng bộ, ổn định và dễ triển khai lại trên các thiết bị khác.

Do tính chất phức tạp của các thư viện AI (PyTorch, TensorFlow, OpenCV), việc cài đặt trực tiếp lên hệ điều hành gốc (Host OS) dễ gây xung đột. Phương pháp tối ưu là sử dụng Docker với cấu hình đặc biệt:

- **Network Host (--network host):** Cho phép Docker sử dụng trực tiếp card mạng của Raspberry Pi, giúp giao tiếp ROS 2 (DDS Multicast) hoạt động mượt mà với các thiết bị bên ngoài (Laptop).
- **Volume Mapping (-v):** Ánh xạ thư mục mã nguồn từ thẻ nhớ vào container để đảm bảo an toàn dữ liệu (Persistence).

- **Privileged Mode (--privileged):** Cấp quyền truy cập trực tiếp vào /dev/video0 (Camera).

Môi trường Docker được xây dựng dựa trên image có sẵn tích hợp ROS2 Humble, sau đó được cấu hình bổ sung các thư viện cần thiết như OpenCV, Ultralytics YOLOv8 và các thư viện Python hỗ trợ. Toàn bộ hệ thống ROS2 và các node xử lý được chạy bên trong container, trong khi vẫn có khả năng truy cập trực tiếp vào phần cứng của robot.

4.2.1 Cấu hình Network Host cho ROS2

ROS2 sử dụng middleware DDS để thực hiện cơ chế khám phá node và truyền thông dữ liệu, trong đó multicast UDP đóng vai trò quan trọng. Khi chạy trong Docker với chế độ mạng mặc định (bridge), các gói multicast thường bị chặn hoặc không được định tuyến đúng, dẫn đến việc các node không thể phát hiện lẫn nhau.

Để khắc phục vấn đề này, container Docker được khởi chạy với chế độ **--network host**, cho phép container sử dụng trực tiếp ngăn xếp mạng của Raspberry Pi. Cấu hình này đảm bảo các node ROS2 bên trong container có thể giao tiếp bình thường với các node ROS2 khác bên ngoài, đồng thời giảm độ trễ truyền dữ liệu – yếu tố quan trọng đối với hệ thống Robot Vision thời gian thực

4.2.2 Quyền truy cập phần cứng và Camera

Để các node ROS2 bên trong Docker có thể truy cập trực tiếp vào camera và các thiết bị phần cứng khác, container được khởi chạy với tham số **--privileged**. Tham số này cho phép container có quyền truy cập đầy đủ vào các thiết bị trong thư mục /dev của hệ điều hành chủ, bao gồm thiết bị camera như /dev/video0.

Nếu không cấu hình quyền truy cập phù hợp, hệ thống sẽ gặp lỗi không thể mở camera hoặc bị từ chối quyền truy cập. Do đó, việc cấp quyền phần cứng cho container là yêu cầu bắt buộc để đảm bảo Camera Node hoạt động ổn định trong môi trường Docker.

4.2.3 Cơ chế Volume Mapping và bảo toàn dữ liệu

Một hạn chế lớn của Docker là dữ liệu bên trong container sẽ bị mất khi container bị dừng hoặc xóa. Để đảm bảo mã nguồn và dữ liệu không bị mất, hệ thống sử dụng cơ chế **Volume Mapping** thông qua tham số -v khi khởi chạy container. Thư mục mã nguồn ROS2 trên Raspberry Pi được ánh xạ trực tiếp vào thư mục làm việc bên trong container.

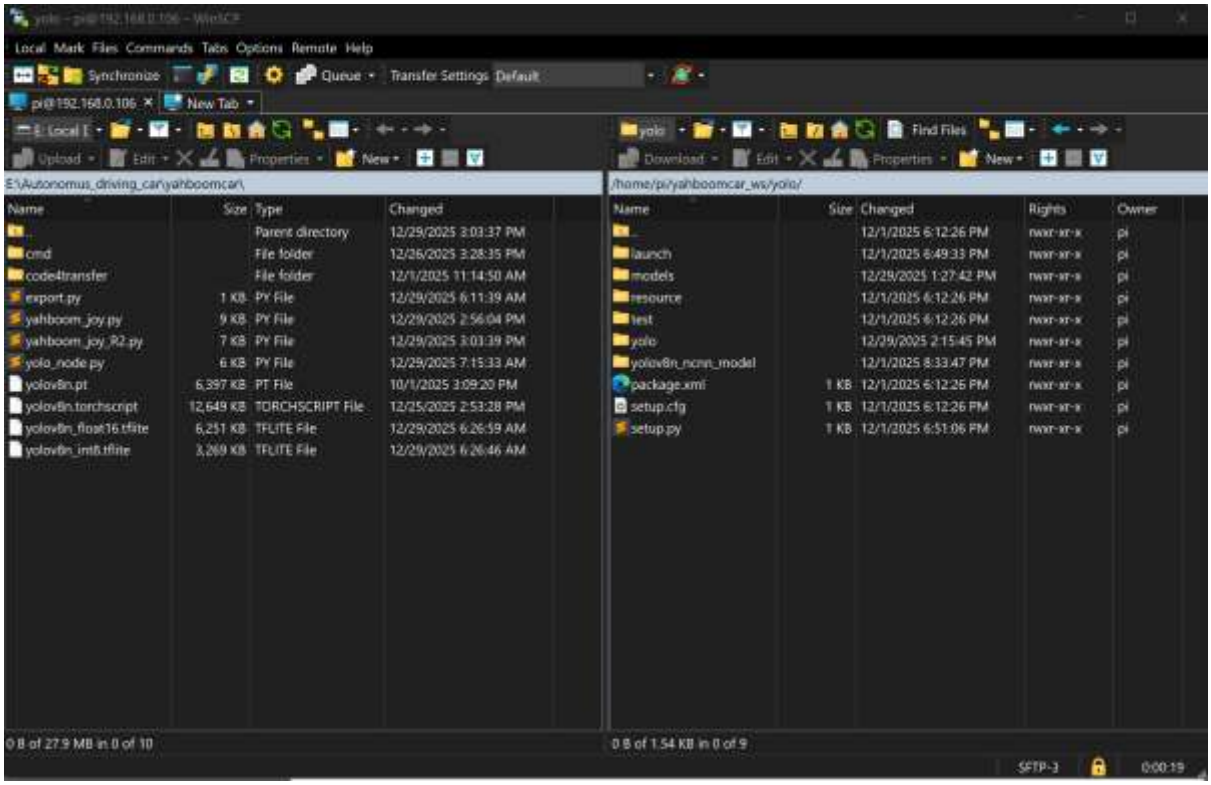
Nhờ cơ chế này, mọi thay đổi về mã nguồn hoặc dữ liệu thực hiện bên trong container đều được lưu trực tiếp trên hệ thống file của Raspberry Pi. Điều này giúp bảo toàn dữ liệu, thuận tiện cho việc phát triển, cập nhật và sao lưu hệ thống trong quá trình thực hiện đồ án.

4.3 Quy trình xử lý dữ liệu ảnh trong ROS2

Quy trình xử lý dữ liệu ảnh trong hệ thống Robot Vision được thiết kế theo một pipeline rõ ràng. Trước hết, Camera Node thu thập hình ảnh và publish lên topic ROS2. YOLO Detection Node subscribe dữ liệu hình ảnh này, sau đó sử dụng thư viện cv_bridge để chuyển đổi dữ liệu từ định dạng ROS Image sang ma trận ảnh OpenCV.

Tiếp theo, hình ảnh được tiền xử lý (resize, chuẩn hóa) để phù hợp với đầu vào của mô hình YOLOv8. Sau khi thực hiện suy luận, kết quả nhận diện được hậu xử lý và vẽ lên hình ảnh gốc. Cuối cùng, hình ảnh đã được chú thích được publish trở lại ROS2, hoàn thiện một vòng xử lý dữ liệu ảnh theo thời gian thực trong hệ thống robot.

4.4 Giao tiếp và quản lý dữ liệu từ xa (WinSCP – SFTP)



Hình 4.3. Hệ thống vận hành luồng dữ liệu trên WinSCP

Do hệ thống robot sử dụng Raspberry Pi hoạt động ở chế độ không màn hình (headless) và môi trường Docker hạn chế truy cập Internet, việc quản lý mã nguồn và

dữ liệu được thực hiện thông qua **WinSCP** sử dụng giao thức **SFTP**. WinSCP cho phép kết nối an toàn từ máy tính cá nhân đến Raspberry Pi thông qua mạng nội bộ hoặc WiFi.

Thông qua WinSCP, người dùng có thể dễ dàng tải lên mã nguồn, mô hình AI hoặc các tệp cấu hình trực tiếp vào thư mục được ánh xạ với container Docker. Cách tiếp cận này giúp quá trình phát triển, thử nghiệm và cập nhật hệ thống Robot Vision diễn ra thuận tiện, nhanh chóng và an toàn.

Hệ thống vận hành luồng dữ liệu như sau:

1. **Tầng Máy trạm (Windows):** Thực hiện export model (.pt -> .tflite) và tải thư viện (.whl).
2. **Tầng Host (Raspberry Pi):** * Kết nối WinSCP vào địa chỉ IP của xe.
   - **Xử lý quyền truy cập (Permission):** Sử dụng lệnh sudo chown -R pi:pi /home/pi/yahboomcar_ws/ để cấp quyền ghi cho tài khoản pi, cho phép kéo thả file trực tiếp vào thư mục workspace.
3. **Tầng Container (ROS 2):** * Sử dụng cơ chế **Volume** Mapping của Docker (-v). Thư mục trên Host được ánh xạ trực tiếp vào Container.
   - **Kết quả:** File vừa copy từ Windows sẽ xuất hiện tức thời trong môi trường ROS 2 để biên dịch và chạy mà không cần copy lần 2.

CHƯƠNG 5: TRIỂN KHAI VÀ TỐI ƯU MÔ HÌNH YOLOv8

5.1 Tổng quan YOLOv8 Nano

YOLOv8 (You Only Look Once phiên bản 8) là thế hệ mới nhất của dòng mô hình nhận diện vật thể YOLO do Ultralytics phát triển, được thiết kế theo hướng cải thiện đồng thời độ chính xác và tốc độ suy luận. YOLOv8 áp dụng kiến trúc mạng nơ-ron tích chập hiện đại, cho phép thực hiện bài toán phát hiện vật thể theo phương pháp end-to-end, trong đó toàn bộ quá trình nhận diện được thực hiện trong một lần suy luận duy nhất.

Trong các phiên bản của YOLOv8, **YOLOv8 Nano (YOLOv8n)** là phiên bản có kích thước mô hình nhỏ nhất, được tối ưu cho các nền tảng phần cứng hạn chế về tài nguyên như thiết bị nhúng hoặc Edge Device. Phiên bản này có số lượng tham số thấp, dung lượng mô hình nhỏ và tốc độ suy luận cao, phù hợp với các ứng dụng robot di động yêu cầu xử lý thời gian thực. Do đó, YOLOv8 Nano được lựa chọn làm mô hình cốt lõi cho hệ thống Robot Vision trong đề tài.

5.2 Triển khai YOLOv8 trên Raspberry Pi 5

Việc triển khai YOLOv8 trên Raspberry Pi 5 đòi hỏi phải cân nhắc kỹ lưỡng giữa hiệu năng xử lý và giới hạn tài nguyên phần cứng. Raspberry Pi 5 sử dụng CPU ARM Cortex-A76, không được trang bị GPU chuyên dụng cho Deep Learning, do đó toàn bộ quá trình suy luận của mô hình YOLOv8 được thực hiện trên CPU. Điều này đặt ra thách thức lớn về tốc độ xử lý khi áp dụng các mô hình học sâu.

Trong hệ thống, mô hình YOLOv8 được triển khai dưới dạng một **ROS2 Node** riêng biệt (YOLO Detection Node) chạy trong môi trường Docker. Node này sử dụng thư viện Ultralytics để tải mô hình, nhận dữ liệu hình ảnh từ Camera Node thông qua topic ROS2 và thực hiện suy luận. Kết quả nhận diện sau đó được đóng gói và publish lên các topic khác để hiển thị hoặc phục vụ cho các tác vụ xử lý tiếp theo. Việc tích hợp YOLOv8 dưới dạng node ROS2 giúp hệ thống đạt được tính mô-đun và dễ dàng mở rộng.

5.3 Tối ưu hiệu năng suy luận

Để đảm bảo hệ thống Robot Vision có thể hoạt động ổn định theo thời gian thực trên Raspberry Pi 5, việc tối ưu hiệu năng suy luận của mô hình YOLOv8 là yêu cầu bắt buộc. Nếu sử dụng mô hình gốc ở định dạng FP32, tốc độ xử lý đạt được chỉ ở mức 3–

5 FPS, không đáp ứng yêu cầu của robot di động. Do đó, đề tài áp dụng nhiều chiến lược tối ưu kết hợp nhằm nâng cao hiệu năng tổng thể của hệ thống.

Các kỹ thuật tối ưu được triển khai bao gồm lượng tử hóa mô hình, giảm tần suất suy luận, tối ưu hiển thị và tổ chức lại luồng xử lý dữ liệu. Các chiến lược này không chỉ giúp tăng tốc độ xử lý mà còn giảm độ trễ và cải thiện độ ổn định của hệ thống trong quá trình vận hành thực tế.

5.3.1 Lượng tử hóa mô hình TFLite INT8

Lượng tử hóa mô hình là kỹ thuật chuyển đổi các tham số của mô hình từ dạng số thực 32-bit (FP32) sang dạng số nguyên 8-bit (INT8). Đối với CPU ARM, việc xử lý số nguyên hiệu quả hơn nhiều so với số thực, do đó lượng tử hóa giúp tăng đáng kể tốc độ suy luận và giảm dung lượng mô hình.

Trong đề tài, mô hình YOLOv8 Nano được chuyển đổi sang định dạng **TensorFlow Lite INT8 (TFLite INT8)** bằng công cụ export của Ultralytics. Việc cố định kích thước đầu vào ảnh (ví dụ: 320×320) giúp loại bỏ các phép tính động không cần thiết và tăng hiệu quả suy luận. Kết quả cho thấy dung lượng mô hình giảm đáng kể và tốc độ xử lý tăng nhiều lần so với mô hình FP32 ban đầu.

5.3.2 Chiến thuật Frame Skipping

Chiến thuật **Frame Skipping** được áp dụng nhằm giảm số lần thực hiện suy luận AI trong một đơn vị thời gian. Thay vì thực hiện nhận diện trên tất cả các khung hình từ camera (thường là 30 FPS), hệ thống chỉ thực hiện suy luận trên mỗi khung hình thứ N, trong khi các khung hình còn lại sử dụng kết quả nhận diện của khung hình trước đó.

Cách tiếp cận này giúp giảm tải đáng kể cho CPU mà vẫn đảm bảo trải nghiệm hiển thị mượt mà cho người dùng. Mặc dù tần suất suy luận giảm, nhưng do tốc độ thay đổi của môi trường không quá nhanh trong các ứng dụng robot di động, chiến thuật Frame Skipping vẫn đảm bảo độ chính xác chấp nhận được và cải thiện rõ rệt độ trễ phản hồi của hệ thống.

5.3.3 Tối ưu hiển thị Bounding Box

Trong quá trình hiển thị kết quả nhận diện, việc sử dụng các hàm vẽ mặc định của thư viện Ultralytics có thể gây tiêu tốn nhiều tài nguyên xử lý. Các hàm này thường thực hiện nhiều phép toán đồ họa phức tạp và tạo ra các đối tượng tạm thời trong bộ nhớ, làm giảm hiệu năng tổng thể.

Để khắc phục, hệ thống sử dụng các hàm vẽ nguyên thủy của OpenCV như cv2.rectangle và cv2.putText để hiển thị bounding box và nhãn đối tượng. Việc tối ưu hiển thị giúp giảm đáng kể thời gian xử lý mỗi khung hình, góp phần nâng cao tốc độ hiển thị và độ ổn định của hệ thống Robot Vision.

5.3.4 Đa luồng đọc Camera

Một vấn đề thường gặp trong các hệ thống Robot Vision là hiện tượng nghẽn hoặc treo luồng khi đọc dữ liệu từ camera, đặc biệt khi hệ thống phải đồng thời xử lý mạng ROS2 và suy luận AI. Để giải quyết vấn đề này, đề tài áp dụng kỹ thuật **đa luồng (multi-threading)** cho việc đọc camera.

Cụ thể, việc đọc khung hình từ camera được thực hiện trong một luồng riêng biệt, liên tục cập nhật khung hình mới nhất vào bộ nhớ đệm. Luồng chính của ROS2 chỉ lấy dữ liệu từ bộ nhớ đệm để xử lý, tránh việc bị chặn bởi quá trình đọc camera. Kỹ thuật này giúp hệ thống hoạt động ổn định hơn, giảm nguy cơ treo và cải thiện độ mượt của luồng hình ảnh.

5.4 Đánh giá hiệu năng (FPS, độ trễ, độ ổn định)

Hiệu năng của hệ thống Robot Vision được đánh giá dựa trên các chỉ tiêu chính gồm tốc độ xử lý (FPS), độ trễ và độ ổn định trong quá trình vận hành. Kết quả thực nghiệm cho thấy, khi sử dụng mô hình YOLOv8 Nano ở định dạng FP32, hệ thống chỉ đạt khoảng 3–5 FPS, không đáp ứng yêu cầu thời gian thực.

Sau khi áp dụng các chiến lược tối ưu như lượng tử hóa TFLite INT8, Frame Skipping, tối ưu hiển thị và đa luồng camera, tốc độ xử lý được nâng lên khoảng **18–22 FPS**, đáp ứng tốt yêu cầu của robot di động. Độ trễ được giảm đáng kể và hệ thống hoạt động ổn định trong thời gian dài mà không xảy ra hiện tượng treo hoặc mất kết nối, chứng minh tính hiệu quả của các phương pháp tối ưu đã triển khai.

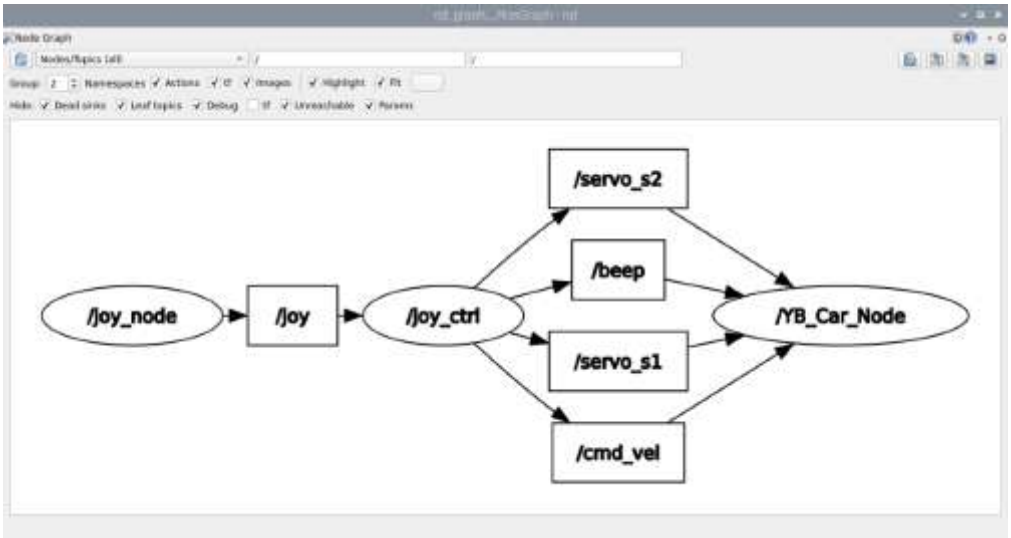
CHƯƠNG 6: ĐIỀU KHIỂN XE VÀ THỰC NGHIỆM

6.1 Kiến trúc điều khiển xe bằng tay cầm (Teleoperation)

Trong phạm vi đề tài, việc điều khiển chuyển động của robot xe được thực hiện thông qua phương pháp **Teleoperation**, tức là điều khiển từ xa bằng tay cầm. Phương pháp này cho phép người vận hành trực tiếp điều khiển robot di chuyển trong môi trường thực nghiệm, đồng thời quan sát phản ứng của hệ thống Robot Vision trong các tình huống khác nhau. Teleoperation đóng vai trò quan trọng trong giai đoạn nghiên cứu và thử nghiệm, giúp kiểm chứng độ ổn định của hệ thống trước khi tiến tới các chức năng tự hành hoàn toàn.

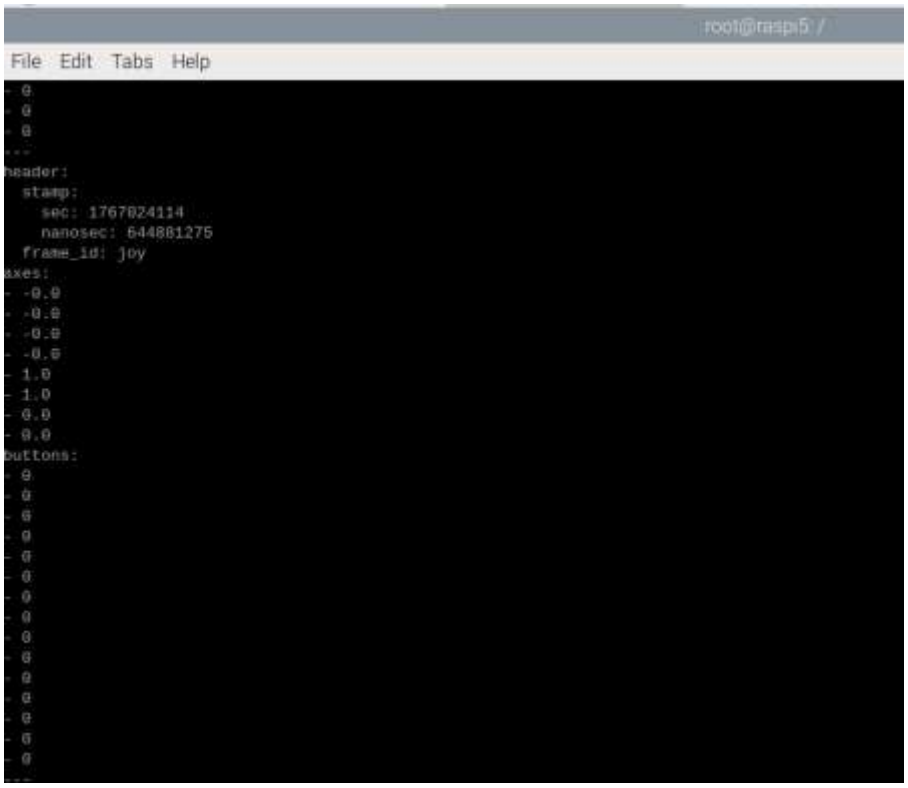
Kiến trúc điều khiển teleoperation được xây dựng trên nền tảng ROS2, trong đó tay cầm điều khiển gửi tín hiệu đầu vào đến robot thông qua các topic ROS. Hệ thống điều khiển được tách biệt rõ ràng với hệ thống xử lý thị giác máy tính, đảm bảo việc điều khiển xe và nhận diện vật thể có thể hoạt động song song mà không gây xung đột tài nguyên. Cách tiếp cận này giúp hệ thống linh hoạt, dễ kiểm soát và thuận lợi cho việc mở rộng sang các thuật toán điều khiển tự động trong tương lai.

6.2 Node JoyTeleop và ánh xạ vận tốc



Hình 6.1. Joystick điều khiển bằng tay node

Node JoyTeleop đóng vai trò trung gian giữa tay cầm điều khiển và hệ thống truyền động của robot. Node này subscribe dữ liệu từ topic /joy với kiểu bản tin sensor_msgs/Joy, trong đó chứa thông tin về trạng thái các trục (axes) và nút bấm (buttons) của tay cầm. Dữ liệu này phản ánh trực tiếp thao tác điều khiển của người dùng.



Hình 6.2. Hiển thị dữ liệu ánh xạ bằng analog sáng các thông số tuyến tính

Trên cơ sở dữ liệu đầu vào, Node JoyTeleop thực hiện ánh xạ vận tốc bằng cách chuyển đổi giá trị analog từ tay cầm sang các thông số vận tốc tuyến tính (linear.x) và vận tốc góc (angular.z) theo chuẩn geometry_msgs/Twist. Quá trình ánh xạ này cho phép điều chỉnh độ nhạy của tay cầm, đảm bảo robot di chuyển mượt mà và phù hợp với điều kiện thực nghiệm. Việc sử dụng chuẩn bản tin ROS2 giúp hệ thống điều khiển dễ dàng tích hợp với các module khác trong kiến trúc robot

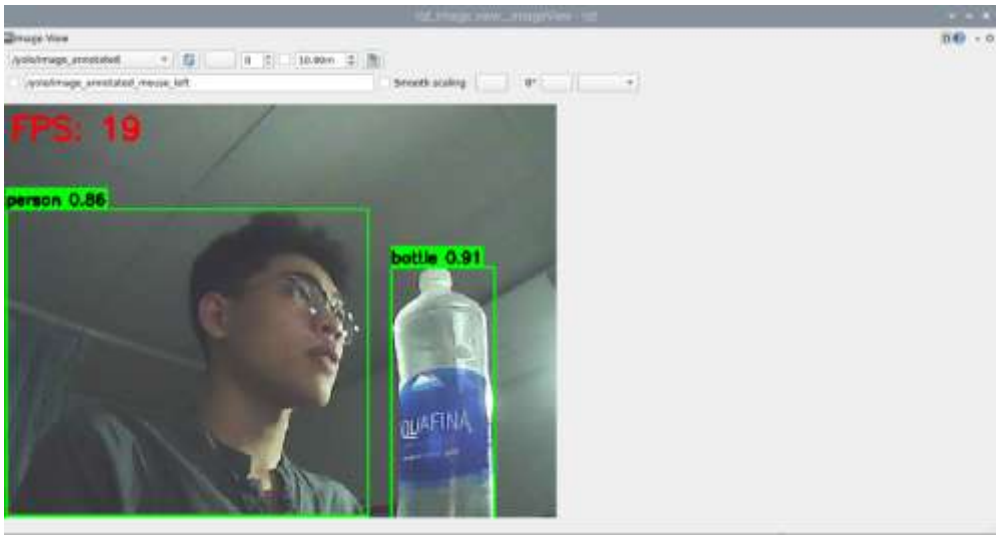
6.3 Cơ chế an toàn và giới hạn vận tốc

Để đảm bảo an toàn trong quá trình vận hành robot, hệ thống điều khiển được tích hợp các **cơ chế an toàn** nhằm hạn chế rủi ro cho thiết bị và môi trường xung quanh. Một trong những cơ chế quan trọng là **bộ lọc deadzone**, giúp loại bỏ các tín hiệu nhiễu nhỏ từ tay cầm khi cần gạt không trở về chính xác vị trí trung tâm. Nhờ đó, robot không bị trôi hoặc di chuyển ngoài ý muốn khi người dùng không chủ động điều khiển.

Bên cạnh đó, hệ thống áp dụng **giới hạn vận tốc tối đa** cho cả vận tốc tuyến tính và vận tốc góc, đảm bảo robot không vượt quá khả năng cơ khí và động cơ. Các nút chức năng trên tay cầm còn được sử dụng để thực hiện dừng khẩn cấp hoặc thay đổi chế độ điều khiển. Những cơ chế này giúp tăng độ an toàn, nâng cao độ tin cậy của hệ thống và tạo điều kiện thuận lợi cho các thử nghiệm kéo dài trong môi trường thực tế.

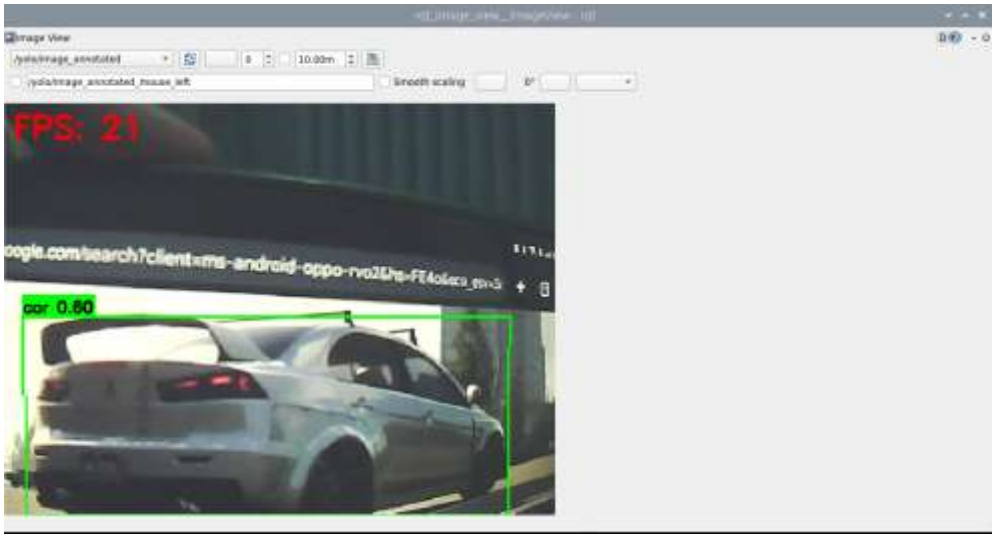
6.4 Thực nghiệm hệ thống

6.4.1 Thực nghiệm nhận diện vật thể



Hình 6.3. Kết quả hiển thị nhận diện vật thể

Thực nghiệm nhận diện vật thể được tiến hành nhằm đánh giá khả năng hoạt động của hệ thống Robot Vision trong điều kiện thực tế. Robot được vận hành trong môi trường có nhiều đối tượng khác nhau, camera thu thập hình ảnh liên tục và YOLO Detection Node thực hiện suy luận theo thời gian thực. Kết quả nhận diện được hiển thị trực tiếp dưới dạng bounding box và nhãn đối tượng trên luồng hình ảnh.



Hình 6.4. Hiển thị khung nhận diện cho vật thể và fps

Kết quả thực nghiệm cho thấy hệ thống có khả năng nhận diện ổn định các vật thể phổ biến trong tầm quan sát của camera. Sau khi áp dụng các kỹ thuật tối ưu hiệu năng,

tốc độ xử lý đạt mức đáp ứng yêu cầu thời gian thực, hình ảnh hiển thị mượt mà và không xuất hiện hiện tượng trễ hoặc treo hệ thống trong quá trình thử nghiệm kéo dài.

6.4.2 Thực nghiệm điều khiển xe

Thực nghiệm điều khiển xe được thực hiện nhằm đánh giá sự phối hợp giữa hệ thống điều khiển teleoperation và hệ thống xử lý thị giác máy tính. Trong quá trình robot di chuyển theo lệnh từ tay cầm, hệ thống Robot Vision vẫn duy trì khả năng nhận diện vật thể và hiển thị kết quả liên tục. Điều này chứng minh khả năng hoạt động song song của các node ROS2 trong môi trường Docker.

Robot phản hồi tốt với các thao tác điều khiển từ tay cầm, chuyển động mượt mà và ổn định trong suốt quá trình thực nghiệm. Không ghi nhận hiện tượng xung đột tài nguyên nghiêm trọng giữa tác vụ điều khiển và tác vụ suy luận AI, cho thấy kiến trúc phần mềm và các phương pháp tối ưu đã được triển khai hiệu quả.

6.4.3 Đánh giá khả năng vận hành thực tế

Từ các kết quả thực nghiệm, có thể đánh giá rằng hệ thống robot xe tích hợp thị giác máy tính hoạt động ổn định trong môi trường thực tế. Hệ thống đáp ứng tốt yêu cầu về tốc độ xử lý, độ trễ và khả năng điều khiển, đồng thời duy trì sự ổn định trong các phiên vận hành kéo dài. Việc kết hợp điều khiển teleoperation với hệ thống Robot Vision giúp người vận hành dễ dàng kiểm soát robot và quan sát trực quan kết quả nhận diện.

Mặc dù hệ thống chưa triển khai chức năng tự hành hoàn toàn, nhưng nền tảng phần cứng và phần mềm đã được xây dựng đầy đủ để phục vụ cho các nghiên cứu nâng cao trong tương lai. Kết quả của chương thực nghiệm cho thấy đề tài đã đạt được các mục tiêu đề ra, đồng thời tạo tiền đề vững chắc cho việc phát triển các hệ thống robot tự hành thông minh dựa trên thị giác máy tính và trí tuệ nhân tạo.

CHƯƠNG 7: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

7.1 Kết luận

Trong khuôn khổ đồ án, nhóm đã nghiên cứu, thiết kế và triển khai thành công một **hệ thống Robot Vision** hoạt động trên nền tảng **MicroROS – Raspberry Pi 5 – ROS2**, tích hợp mô hình trí tuệ nhân tạo YOLOv8 cho bài toán nhận diện vật thể theo thời gian thực. Hệ thống được xây dựng với kiến trúc phần mềm rõ ràng, mô-đun hóa theo chuẩn ROS2 và được triển khai trong môi trường Docker nhằm đảm bảo tính ổn định và khả năng tái lập.

Kết quả thực nghiệm cho thấy hệ thống hoạt động ổn định trên phần cứng nhúng, đáp ứng được các yêu cầu cơ bản về tốc độ xử lý, độ trễ và khả năng vận hành liên tục. Việc kết hợp điều khiển xe bằng tay cầm (teleoperation) với hệ thống Robot Vision đã chứng minh tính khả thi của kiến trúc đề xuất, đồng thời tạo nền tảng vững chắc cho các nghiên cứu và ứng dụng robot thông minh trong tương lai.

7.2 Các kết quả đạt được

Hình 7.1. Xây dựng phần cứng theo model Raspberry-Pi5 Robor Car

Qua quá trình thực hiện đồ án, nhóm đã đạt được các kết quả chính sau:

- Xây dựng thành công **hệ thống Robot Vision** chạy trên Raspberry Pi 5, sử dụng ROS2 Humble và triển khai trong Docker Container.
- Triển khai mô hình **YOLOv8 Nano** cho bài toán nhận diện vật thể trên thiết bị nhúng, phù hợp với điều kiện tài nguyên hạn chế.

- Áp dụng hiệu quả các **kỹ thuật tối ưu hiệu năng** như lượng tử hóa mô hình TFLite INT8, chiến thuật Frame Skipping, tối ưu hiển thị và đa luồng đọc camera.
- Nâng tốc độ xử lý từ mức 3–5 FPS lên khoảng **18–22 FPS**, đáp ứng yêu cầu xử lý gần thời gian thực cho robot di động.
- Xây dựng hệ thống **điều khiển xe bằng tay cầm** trên nền tảng ROS2, đảm bảo an toàn và ổn định trong quá trình thực nghiệm.
- Thiết lập quy trình quản lý mã nguồn và dữ liệu từ xa thông qua **WinSCP – SFTP**, thuận tiện cho phát triển và bảo trì hệ thống.

Những kết quả này cho thấy đề tài không chỉ mang tính học thuật mà còn có giá trị thực tiễn trong việc xây dựng các hệ thống robot tích hợp trí tuệ nhân tạo trên nền tảng phần cứng nhúng.

7.3 Hạn chế của hệ thống

Bên cạnh các kết quả đạt được, hệ thống vẫn tồn tại một số hạn chế nhất định. Trước hết, việc suy luận mô hình AI hoàn toàn dựa trên CPU của Raspberry Pi 5 khiến hiệu năng vẫn còn giới hạn so với các hệ thống sử dụng GPU hoặc phần cứng tăng tốc chuyên dụng. Điều này ảnh hưởng đến khả năng mở rộng mô hình AI phức tạp hơn hoặc xử lý các bài toán thị giác nâng cao.

Ngoài ra, trong phạm vi đồ án, hệ thống mới dừng lại ở mức **nhận diện và hiển thị thông tin**, chưa triển khai điều khiển tự hành hoàn toàn dựa trên kết quả nhận diện. Việc điều khiển xe vẫn phụ thuộc vào người vận hành thông qua teleoperation, do đó chưa khai thác hết tiềm năng của hệ thống Robot Vision trong các bài toán tự hành thông minh.

7.4 Hướng phát triển

Tích hợp điều khiển tự hành dựa trên YOLO

Trong các nghiên cứu tiếp theo, hệ thống có thể được mở rộng bằng cách tích hợp trực tiếp kết quả nhận diện từ YOLO vào khối điều khiển chuyển động. Thông tin về vị trí và loại vật thể có thể được sử dụng làm đầu vào cho các thuật toán ra quyết định, cho phép robot tự động điều chỉnh hướng di chuyển, tránh vật cản hoặc thực hiện các nhiệm vụ cụ thể mà không cần sự can thiệp của người vận hành.

Bám theo người / vật thể

Một hướng phát triển tiềm năng khác là xây dựng chức năng **bám theo người hoặc vật thể** dựa trên kết quả nhận diện của YOLO. Bằng cách kết hợp thông tin bounding

box với dữ liệu điều khiển vận tốc, robot có thể tự động duy trì khoảng cách và hướng di chuyển phù hợp với mục tiêu. Ứng dụng này có giá trị thực tiễn cao trong các lĩnh vực như robot dịch vụ, giám sát hoặc hỗ trợ con người.

Kết hợp Micro-ROS điều khiển mức thấp

Để nâng cao khả năng điều khiển thời gian thực và độ chính xác của hệ thống, Micro-ROS có thể được tích hợp sâu hơn vào khối điều khiển mức thấp. Vi điều khiển đảm nhiệm các tác vụ điều khiển động cơ, xử lý cảm biến và phản hồi nhanh, trong khi Raspberry Pi 5 tập trung vào xử lý AI và ra quyết định mức cao. Sự kết hợp này sẽ giúp hệ thống đạt được kiến trúc hoàn chỉnh hơn, phù hợp cho các hệ thống robot tự hành thông minh trong tương lai.

TÀI LIỆU THAM KHẢO

1. M. Reke et al., “A Self-Driving Car Architecture in ROS2,” in 2020 International SAUPEC/RobMech/PRASA Conference, IEEE, Jan. 2020, pp. 1–6. doi: 10.1109/saupec/robmech/prasa48453.2020.9041020.
2. H. Amano et al., “A dataset generation for object recognition and a tool for generating ROS2 FPGA node,” in 2021 International Conference on Field-Programmable Technology (ICFPT), IEEE, Dec. 2021, pp. 1–4. doi: 10.1109/icfpt52863.2021.9609880.
3. M. Shah, R. D. Eastman, and T. Hong, “An overview of robot-sensor calibration methods for evaluation of perception systems,” in Proceedings of the Workshop on Performance Metrics for Intelligent Systems, ACM, Mar. 2012, pp. 15–20. doi: 10.1145/2393091.2393095.
4. W. Wang, H. Lin, and J. Wang, “CNN based lane detection with instance segmentation in edge-cloud computing,” Journal of Cloud Computing, vol. 9, no. 1, May 2020, doi: 10.1186/s13677-020-00172-z.
5. D.-H. Lee and J.-L. Liu, “End-to-end deep learning of lane detection and path prediction for real-time autonomous driving,” Signal, Image and Video Processing, vol. 17, no. 1, pp. 199–205, Apr. 2022, doi: 10.1007/s11760-022-02222-2.
6. B.-C.-Z. Blaga, M.-A. Deac, R. W. Y. Al-doori, M. Negru, and R. Danescu, “Miniature Autonomous Vehicle Development on Raspberry Pi,” in 2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP), IEEE, Sep. 2018, pp. 229–236. doi: 10.1109/iccp.2018.8516589.