

TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH

KHOA ĐIỆN - ĐIỆN TỬ

BỘ MÔN ĐIỀU KHIỂN TỰ ĐỘNG

Trịnh Quang Huy

Nguyễn Đăng Khoa

Nguyễn Thanh Khôi

ĐỒ ÁN TỐT NGHIỆP

ỨNG DỤNG THỊ GIÁC MÁY CHO XE TỰ HÀNH TRONG
GIAO THÔNG

COMPUTER VISION APPLICATION FOR AUTONOMOUS
VEHICLES IN TRAFFIC

CỬ NHÂN NGÀNH KỸ THUẬT ĐIỀU KHIỂN & TỰ ĐỘNG HÓA

TP. HỒ CHÍ MINH, 2025

TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN ĐIỀU KHIỂN TỰ ĐỘNG

Trịnh Quang Huy – 2110211

Nguyễn Đăng Khoa – 2111529

Nguyễn Thanh Khôi – 2111559

ĐỒ ÁN TỐT NGHIỆP
ỨNG DỤNG THỊ GIÁC MÁY CHO XE TỰ HÀNH TRONG
GIAO THÔNG

COMPUTER VISION APPLICATION FOR AUTONOMOUS
VEHICLES IN TRAFFIC

CỬ NHÂN NGÀNH KỸ THUẬT ĐIỀU KHIỂN & TỰ ĐỘNG HÓA

GIẢNG VIÊN HƯỚNG DẪN
PGS.TS HUỲNH THÁI HOÀNG

TP. HỒ CHÍ MINH, 2025

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến Quý Thầy Cô Trường Đại học Bách Khoa – Đại học Quốc gia Thành phố Hồ Chí Minh nói chung, Quý Thầy Cô Bộ môn Điều khiển Tự động nói riêng, đã truyền đạt cho em những kiến thức quý báu thông qua bài giảng của các môn học, giúp em có tiền đề vững vàng để hoàn thành đề tài Đồ án Tốt nghiệp này.

Chúng em xin được bày tỏ lòng biết ơn sâu sắc đến Thầy Huỳnh Thái Hoàng. Trong quá trình thực hiện đề tài, Thầy luôn tận tình giúp đỡ, theo sát tiến độ cũng như định hướng cho em để có thể thực hiện được Đồ án này một cách tốt nhất, đáp ứng được các mục tiêu ban đầu đặt ra cho đề tài. Có được sự dẫn dắt của Thầy là một may mắn rất lớn trên chặng đường học tập của em. Qua quá trình làm Đồ án này, em đã học hỏi thêm được rất nhiều điều bổ ích và đó sẽ trở thành hành trang quý giá cho em sau khi tốt nghiệp.

Chúng em cũng xin gửi lời cảm ơn sâu sắc đến bạn Trịnh Quang Huy – mã số sinh viên: 2110211, người đã tận tình hỗ trợ nhóm, chia sẻ kinh nghiệm thực tế và cùng nhóm làm việc để nhóm có thêm góc nhìn chuyên sâu và ứng dụng được lý thuyết vào thực tiễn.

Mặc dù đã cố gắng trong quá trình nghiên cứu, song do khả năng và kinh nghiệm của chúng em vẫn còn nhiều thiếu sót, nên Đồ án này không thể tránh khỏi những hạn chế. Vì vậy, em rất mong được nhận được sự góp ý từ Thầy để có thể sửa đổi và bổ sung tiếp để Đồ án này được hoàn thiện hơn.

Em xin chân thành cảm ơn!

ĐỀ CƯƠNG CHI TIẾT

TÊN ĐỒ ÁN: ỨNG DỤNG THỊ GIÁC MÁY CHO XE TỰ HÀNH TRONG GIAO THÔNG

Nội dung đề tài:

Mục tiêu:

Mục tiêu đề tài là xây dựng một hệ thống xe tự hành có khả năng bám theo làn đường, nhận diện và chấp hành biển báo giao thông. Hệ thống sử dụng camera Gopro Hero 8 và Jetson TX2 để xử lý ảnh trực tiếp, kết hợp các thuật toán điều khiển Fuzzy logic và PID nhằm điều hướng chính xác. Giao diện người dùng hỗ trợ giám sát trạng thái hoạt động và điều chỉnh tham số thời gian thực. Đảm bảo rằng hệ thống có thể bám làn ổn định, phản ứng đúng với biển báo nhờ YOLOv4-tiny, ra quyết định điều hướng phù hợp, và cung cấp giao diện điều khiển trực quan cho người dùng.

Phương pháp thực hiện:

- Xử lý ảnh và học máy:
 - Thu thập ảnh và huấn luyện mô hình YOLOv4-tiny với 8 loại biển báo phổ biến.
 - Áp dụng các kỹ thuật xử lý ảnh như làm mờ, nhị phân hóa, nội suy đường để phát hiện đặc trưng làn đường.
- Điều khiển xe:
 - Xây dựng bộ điều khiển Fuzzy cho góc lái dựa trên sai số tọa độ, đạo hàm sai số và góc lệch làn.
 - Kết hợp bộ Fuzzy thứ hai để điều chỉnh tốc độ dựa trên góc lái và vận tốc hiện tại.

- Sử dụng PID để điều khiển chính xác tốc độ động cơ DC.
- Giao tiếp và GUI:
 - Jetson TX2 xử lý ảnh, gửi dữ liệu sang STM32F407 qua UART.
 - STM32 xử lý điều khiển và phản hồi dữ liệu cho GUI thông qua Bluetooth.
 - GUI được lập trình bằng Python + PyQt, có khả năng điều chỉnh tham số và hiển thị trạng thái.

Kết quả mong muốn:

- Hệ thống nhận diện làn đường chính xác trong môi trường giả lập, duy trì trạng thái bám làn ổn định.
- Nhận diện và chấp hành đúng các loại biển báo giao thông đã định nghĩa.
- Giao diện trực quan, dễ sử dụng, hỗ trợ điều chỉnh tham số điều khiển trong thời gian thực.
- Mô hình xe hoạt động liên tục, tự động chuyển trạng thái giữa các chế độ điều khiển.

Kế hoạch thực hiện

Giai đoạn 1: Tìm hiểu tổng quan và xác định hướng triển khai

- Tìm hiểu các mô hình xe tự hành, nguyên lý điều khiển bám làn, phát hiện biển báo.
- Khảo sát sơ đồ hệ thống, lựa chọn phần cứng chính: Jetson TX2, STM32F407, Servo, Encoder, Camera Gopro Hero 8.
- Phân tích bài toán, xác định yêu cầu hệ thống, sơ đồ kết nối, thiết kế kiến trúc giao tiếp Jetson ↔ STM32 ↔ GUI.

Giai đoạn 2: Triển khai phần cứng và hệ thống điện

- Thiết kế, lắp đặt mô hình cơ khí dựa trên khung gầm RC1.
- Kết nối và cấp nguồn cho các thành phần: servo, encoder, motor, camera, Jetson TX2, module Bluetooth.
- Cấu hình hệ thống nguồn sử dụng pin LiPo và các mạch ổn áp phù hợp.

- Kiểm tra kết nối UART (Jetson – STM32), kết nối Bluetooth (STM32 – GUI).

Giai đoạn 3: Xây dựng thuật toán xử lý ảnh

- Cài đặt Jetson TX2, thiết lập môi trường Python và thư viện xử lý ảnh.
- Xử lý frame ảnh từ camera Gopro 8: phát hiện trung tâm làn, tính toán góc lệch (phi).
- Phát triển thuật toán nội suy khi mất lane, chuyển trạng thái qua giao lộ.
- Huấn luyện mô hình YOLOv4-tiny cho 8 loại biển báo và kiểm tra kết quả trên Jetson.

Giai đoạn 4: Phát triển các chương trình điều khiển trên STM32

- Triển khai chương trình đọc encoder, tính tốc độ động cơ, PID điều tốc.
- Triển khai bộ điều khiển Fuzzy điều chỉnh góc lái từ sai số tọa độ, đạo hàm sai số, góc lệch.
- Thiết kế thêm bộ Fuzzy điều tốc theo góc lái và tốc độ hiện tại.
- Triển khai chương trình phát hiện giá trị đầu vào bất thường, chương trình xoay camera theo góc nghiêng.
- Viết chương trình điều khiển rẽ trái/phải dựa vào dữ liệu Jetson.
- Tìm hiểu về chuẩn giao tiếp Protocol Buffer và áp dụng

Giai đoạn 5: Thiết kế giao diện người dùng (GUI)

- Xây dựng GUI bằng PyQt: hiển thị thông tin điều khiển, sai số, góc lái, tốc độ,...
- Cài đặt kết nối Bluetooth với STM32, hiển thị dữ liệu theo thời gian thực.
- Cho phép người dùng điều chỉnh tham số Fuzzy, PID, gửi lệnh bắt đầu, dừng robot.
- Triển khai Protocol Buffer

Giai đoạn 6: Tích hợp hệ thống và kiểm tra thực nghiệm

- Tích hợp xử lý ảnh, điều khiển STM32 và GUI vào một hệ thống hoàn chỉnh.

- Tạo sa bàn đáp ứng các trường hợp kiểm tra: bám làn, rẽ trái/phải tại giao lộ, phản ứng với biển báo.
- Tối ưu lại các ngưỡng điều kiện chuyển trạng thái, các hệ số điều khiển.

Giai đoạn 7: Đánh giá, hoàn thiện và viết báo cáo luận văn

- Chạy nhiều kịch bản để đánh giá độ chính xác, độ ổn định.
- Đo đặc sai số định lượng: sai số làn, tỷ lệ nhận diện biển báo, phản ứng hành vi.
- Chụp ảnh, ghi nhận kết quả, vẽ sơ đồ và viết hoàn thiện báo cáo luận văn.
- Chuẩn bị slide bảo vệ và trình bày đề tài.

Xác nhận của Cán bộ hướng dẫn

(Kí tên và ghi rõ họ tên)

TP. HCM, ngày ... tháng ... năm ...

Sinh viên

(Kí tên và ghi rõ họ tên

DANH SÁCH HỘI ĐỒNG BẢO VỆ LUẬN VĂN

Hội đồng chấm luận văn tốt nghiệp, thành lập theo Quyết định số

ngày của Hiệu trưởng Trường Đại học Bách khoa TP.HCM.

1. – Chủ tịch.

2. – Thư ký.

3. – Ủy viên.

4. – Ủy viên.

5. – Ủy viên.

Mục lục

Chương 1: Giới thiệu	1
1.1. Giới thiệu bài toán hoặc đối tượng nghiên cứu.....	1
1.2. Ý nghĩa khoa học và thực tiễn của việc nghiên cứu đề tài	2
1.3. Tổng quan về các nghiên cứu liên quan.....	2
1.4. Mục tiêu đồ án tốt nghiệp và phương pháp thực hiện.....	3
1.5. Sơ lược về nội dung báo cáo đồ án tốt nghiệp	3
Chương 2: Cơ sở lý thuyết.....	5
2.1. Mô hình điều khiển.....	5
2.1.1. Cấu trúc hệ thống điều khiển	5
2.1.2. Mục tiêu của mô hình điều khiển.....	8
2.2. Xử lý ảnh.....	8
2.2.1. Phương pháp Otsu Thresholding	8
2.2.2. Phương pháp Median Blur.....	10
2.2.3. Phương pháp phát hiện đường biên	11
2.3. YOLO cho nhận diện đối tượng (Object Detection)	12
2.3.1. Giới thiệu chung về YOLO	12
2.3.2. Cách hoạt động của YOLO	15
2.3.3. YOLOv4-Tiny – Phiên bản nhẹ và hiệu quả	18
2.4. Bộ điều khiển PID	20
2.5. Bộ điều khiển Fuzzy	24
Chương 3: Thiết kế và thi công phần cứng	27
3.1. Sơ đồ khái mô hình.....	27
3.1.1. Sơ đồ tổng quát mô hình.....	27
3.1.2. Sơ đồ chi tiết mô hình.....	28

3.1.3. Sơ đồ kết nối phần công suất	29
3.1.4. Sơ đồ kết nối phần giao tiếp và điều khiển	30
3.2. Các cơ cấu cơ khí của mô hình.....	31
 3.2.1. Cơ cấu đánh lái.....	31
 3.2.2. Cơ cấu truyền động tạo tốc độ dài cho mô hình.....	32
 3.2.3. Cơ cấu xoay camera	32
3.3. Tổng quan các thiết bị.....	33
 3.3.1. Các module, thiết bị được sử dụng.....	33
 3.3.2. Board giao tiếp giữa STM32F407 và các khối khác	40
3.4. Sản phẩm hoàn thiện	42
Chương 4: Thiết kế phần mềm.....	43
4.1. Thiết kế giao tiếp giữa các hệ thống	43
 4.1.1. Chuẩn giao tiếp chung	43
 4.1.2. Kiến trúc luồng giao tiếp tổng quát.....	44
 4.1.3. Định dạng gói tin truyền thông.....	48
4.2. Thiết kế Firmware trên vi điều khiển STM32F407	49
 4.2.1. Kiến trúc Firmware	49
 4.2.2. Các tính năng của Firmware	53
4.3. Thiết kế phần mềm của GUI.....	54
 4.3.1. Giao diện người dùng	54
 4.3.2. Giao diện biểu đồ.....	55
 4.3.3. Giao diện thiết lập.....	57
4.4. Thiết kế phần mềm trên Jetson TX-2.....	58
4.5. Tổng quan hệ thống mô hình xe tự hành	60
Chương 5: Xây dựng giải thuật	62

5.1. Giải thuật xử lý ảnh làn đường.....	62
5.1.1. Chương trình chính	62
5.1.2. Chương trình con	64
5.2. Giải thuật nhận diện biển báo sử dụng YOLOv4-tiny.....	71
5.2.1. Giới thiệu bài toán và tập dữ liệu	71
5.2.2. Giải thuật chương trình chính	76
5.2.3. Giải thuật chương trình con	78
5.3. Giải thuật điều khiển xe tự hành	80
5.3.1. Giải thuật chương trình chính	80
5.3.2. Mô hình máy trạng thái điều khiển xe tự hành.....	81
5.3.3. Giải thuật chương trình con	83
Chương 6: Kết quả và hướng phát triển	106
6.1. Kết quả thực hiện	106
6.1.1. Kết quả xử lý ảnh.....	106
6.1.2. Kết quả nhận diện biển báo giao thông	112
6.1.3. Kết quả vận hành của mô hình xe tự hành	115
6.2. Đánh giá kết quả	117
6.3. Hướng phát triển	120
Tài liệu tham khảo.....	121

Danh mục hình ảnh

Hình 1. Mô hình xe trong các đề tài trước.....	2
Hình 2. Kết quả phương pháp Otsu.....	10
Hình 3. Mô tả hoạt động của bộ lọc trung vị.....	10
Hình 4. Hiệu quả của bộ lọc trung vị trong khử nhiễu muối tiêu.....	11
Hình 5. Các phương pháp phát hiện vật thể.....	13
Hình 6. Bảng so sánh hiệu suất các model	13
Hình 7. Kiến trúc YOLO.....	14
Hình 8. Mô tả tổng quan YOLO.....	15
Hình 9. Đầu ra mỗi ảnh của YOLO	15
Hình 10. Tính chỉ số IoU.....	16
Hình 11. Kết quả của NMS	17
Hình 12. Hàm loss của YOLO	18
Hình 13. Kiến trúc YOLOv4-tiny.....	18
Hình 14. TensorRT.....	19
Hình 15. Sơ đồ bộ điều khiển PID.....	21
Hình 16. Đáp ứng của hệ thống khi tăng dần KP	22
Hình 17. Đáp ứng của hệ thống khi tăng dần K_I	23
Hình 18. Đáp ứng của hệ thống khi tăng dần K_D	24
Hình 19. Sơ đồ khối tổng quát bộ điều khiển Fuzzy	25
Hình 20. Sơ đồ khối tổng quát các bước bộ điều khiển mờ cơ bản	25
Hình 21. Phương pháp giải mờ theo trọng tâm	26
Hình 22. Phương pháp giải trung bình mờ	26
Hình 23. Sơ đồ khối tổng quát phần cứng	27
Hình 24. Sơ đồ vị trí và kết nối của các thành phần trên mô hình xe.....	28
Hình 25. Sơ đồ kết nối phần công suất	29
Hình 26. Sơ đồ kết nối phần dữ liệu và điều khiển	30
Hình 27. Cơ cấu đánh lái	31
Hình 28. Cơ cấu truyền động tạo tốc độ dài	32
Hình 29. Cơ cấu xoay camera	32

<i>Hình 30. Jetson TX-2 Developer Kit</i>	33
<i>Hình 31. Kit STM32F407 Discovery.....</i>	34
<i>Hình 32. Camera hành trình Gopro Hero 8.....</i>	35
<i>Hình 33. Pin Lipo Power 11.1VDC.....</i>	35
<i>Hình 34. Module giảm áp LM2596</i>	36
<i>Hình 35. Module giảm áp XL4005</i>	36
<i>Hình 36. Động cơ DC Servo GA25-370</i>	37
<i>Hình 37. Servo MG996R.....</i>	38
<i>Hình 38. Module lái BTS7960</i>	39
<i>Hình 39. Encoder</i>	39
<i>Hình 40. Sơ đồ nguyên lý board giao tiếp.....</i>	40
<i>Hình 41. Sơ đồ mạch in</i>	41
<i>Hình 42. Sản phẩm phần cứng mô hình xe tự hành hoàn thiện.....</i>	42
<i>Hình 43. Quá trình hoạt động của giao thức Protocol Buffer trong hệ thống</i>	43
<i>Hình 44. Sơ đồ trình tự giao tiếp – STM32F407 gửi tin nhắn.....</i>	45
<i>Hình 45. Sơ đồ trình tự giao tiếp – GUI gửi tin nhắn.....</i>	46
<i>Hình 46. Sơ đồ trình tự giao tiếp –Jetson TX-2 gửi tin nhắn.....</i>	47
<i>Hình 47. Kiến trúc Firmware tổng thể</i>	49
<i>Hình 48. Kiến trúc Firmware chi tiết.....</i>	51
<i>Hình 49. Giao diện người dùng của GUI.....</i>	54
<i>Hình 50. Giao diện biểu đồ của GUI.....</i>	55
<i>Hình 51. Giao diện thiết lập của GUI</i>	57
<i>Hình 52. Sơ đồ hoạt động của chương trình chính trên Jetson TX-2</i>	59
<i>Hình 53. Sơ đồ trình tự giao tiếp – tổng quan chương trình hệ thống.....</i>	60
<i>Hình 54. Mô tả hoạt động của chương trình chính xử lý ảnh.....</i>	62
<i>Hình 55. Mô tả hoạt động của chương trình tiền xử lý ảnh.....</i>	64
<i>Hình 56. Mô tả hoạt động của chương trình xác định góc lệch và vị trí trung tâm (trực tiếp).....</i>	65
<i>Hình 57. Mô tả hoạt động của chương trình xác định góc lệch và vị trí trung tâm (nội suy).....</i>	66

<i>Hình 58. Mô tả hoạt động của chương trình xác định vùng ảnh chứa mặt đường ..</i>	67
<i>Hình 59. Mô tả hoạt động của chương trình tìm vạch kẻ đường (trực tiếp)</i>	68
<i>Hình 60. Mô tả hoạt động của chương trình tìm vạch kẻ đường (nội suy).....</i>	70
<i>Hình 61. Quy trình huấn luyện mô hình</i>	73
<i>Hình 62. Ảnh dùng để huấn luyện model</i>	74
<i>Hình 63. Ảnh sau khi gắn nhãn.....</i>	74
<i>Hình 64. Tổng quan tập dữ liệu.....</i>	75
<i>Hình 65. Chương trình nhận diện biển báo</i>	76
<i>Hình 66. Chương trình lọc nhiễu nhận diện biển báo.....</i>	78
<i>Hình 67. Chương trình lọc nhiễu chấp hành biển báo.....</i>	79
<i>Hình 68. Lưu đồ giải thuật chương trình chính điều khiển xe tự hành.....</i>	80
<i>Hình 69. Mô hình máy trạng thái điều khiển xe tự hành.....</i>	82
<i>Hình 70. Lưu đồ giải thuật điều khiển góc xoay động cơ Servo</i>	84
<i>Hình 71. Lưu đồ giải thuật đọc kết quả encoder.....</i>	86
<i>Hình 72. Lưu đồ giải thuật điều khiển tốc độ bằng bộ điều khiển PID rời rạc</i>	87
<i>Hình 73. Lưu đồ giải thuật phát hiện giá trị bất thường.....</i>	88
<i>Hình 74. Lưu đồ máy trạng thái điều khiển góc quay camera.....</i>	90
<i>Hình 75. Sơ đồ chương trình điều khiển di chuyển theo làn đường</i>	91
<i>Hình 76. Sơ đồ khởi bộ điều khiển mờ điều chỉnh góc lái (phần được khoanh đỏ)..</i>	94
<i>Hình 77. Định nghĩa giá trị ngôn ngữ cho sai số tọa độ điểm trung tâm làn đường</i>	95
<i>Hình 78. Định nghĩa giá trị ngôn ngữ cho tốc độ biến thiên sai số tọa độ điểm trung tâm làn đường.....</i>	95
<i>Hình 79. Định nghĩa giá trị ngôn ngữ cho sai số độ nghiêng làn đường</i>	95
<i>Hình 80. Định nghĩa giá trị ngôn ngữ cho kết quả giá trị góc lái của bộ điều khiển ..</i>	96
<i>Hình 81. Quy tắc suy luận bộ điều khiển mờ điều chỉnh góc servo đánh lái</i>	96
<i>Hình 82. Sơ đồ khởi bộ điều khiển mờ điều chỉnh tốc độ đặt (phần được khoanh đỏ)</i>	97
<i>Hình 83. Định nghĩa giá trị ngôn ngữ cho góc của động cơ servo đánh lái đã chuẩn hóa.....</i>	98
<i>Hình 84. Định nghĩa giá trị ngôn ngữ cho tốc độ hiện tại của xe tự hành</i>	98

<i>Hình 85. Định nghĩa giá trị ngôn ngữ cho tốc độ đặt của xe tự hành</i>	98
<i>Hình 86. Quy tắc suy luận bộ điều khiển mờ điều chỉnh tốc độ đặt cho xe tự hành</i> .	99
<i>Hình 87. Sơ đồ chương trình điều khiển rẽ trái, rẽ phải</i>	100
<i>Hình 88. Lưu đồ giải thuật chương trình điều khiển rẽ trái, rẽ phải</i>	101
<i>Hình 89. Sơ đồ khởi bộ điều khiển mờ điều chỉnh góc lái (phần được khoanh đỏ)</i> 102	
<i>Hình 90. Định nghĩa giá trị ngôn ngữ cho sai số tọa độ điểm trung tâm làn đường</i>	103
<i>Hình 91. Định nghĩa giá trị ngôn ngữ cho tốc độ biến thiên sai số tọa độ điểm trung tâm làn đường</i>	103
<i>Hình 92. Định nghĩa giá trị ngôn ngữ cho kết quả giá trị góc lái của bộ điều khiển</i>	103
<i>Hình 93. Quy tắc suy luận bộ điều khiển mờ điều chỉnh góc servo đánh lái</i>	104
<i>Hình 94. Lưu đồ giải thuật xử lý biển báo giao thông</i>	104
<i>Hình 95. Ảnh gốc thu được từ camera của xe</i>	106
<i>Hình 96. Khu vực ảnh được khoanh vùng</i>	106
<i>Hình 97. Ảnh sau khi chuyển thành ảnh xám</i>	107
<i>Hình 98. Ảnh sau khi lọc nhiễu</i>	107
<i>Hình 99. Ảnh sau khi đảo</i>	107
<i>Hình 100. Ảnh sau bước phân ngưỡng</i>	107
<i>Hình 101. Biểu đồ cường độ sáng</i>	108
<i>Hình 102. Vùng ảnh mặt đường tìm được</i>	108
<i>Hình 103. Góc lệch và vị trí trung tâm</i>	108
<i>Hình 104. Ảnh gốc thu được từ camera của xe khi đi thẳng qua giao lộ</i>	109
<i>Hình 105. Khu vực ảnh được khoanh vùng khi đi qua giao lộ</i>	109
<i>Hình 106. Ảnh sau bước tiền xử lý</i>	110
<i>Hình 107. Biểu đồ cường độ sáng</i>	110
<i>Hình 108. Các vạch kẻ đường được tìm ra</i>	110
<i>Hình 109. Góc lệch và vị trí trung tâm của mặt đường</i>	111
<i>Hình 110. Ảnh gốc thu được từ camera của xe khi rẽ trái ở giao lộ</i>	111
<i>Hình 111. Vạch kẻ đường bên trái được tìm ra</i>	111

<i>Hình 112. Ảnh gốc thu được từ camera của xe khi đi rẽ phải ở giao lộ</i>	<i>111</i>
<i>Hình 113. Vạch kẻ đường bên phải được tìm ra</i>	<i>112</i>
<i>Hình 114. Đồ thị F1-score (trái) và Precision, Recall (phải)</i>	<i>112</i>
<i>Hình 115. Đồ thị Train_loss</i>	<i>113</i>
<i>Hình 116. Đồ thị mAP50</i>	<i>113</i>
<i>Hình 117. Nhận diện biển báo Children_Crossing</i>	<i>115</i>
<i>Hình 118. Nhận diện đèn đỏ</i>	<i>115</i>
<i>Hình 119. Sa bàn 1</i>	<i>116</i>
<i>Hình 120. Sa bàn 2</i>	<i>116</i>
<i>Hình 121. Giá trị đầu vào bất thường</i>	<i>118</i>
<i>Hình 122. Giá trị đầu vào bất thường sau khi được xử lý</i>	<i>118</i>
<i>Hình 123. Đồ thị thể hiện sai số tọa độ, sai số góc nghiêng, góc của động cơ servo đánh lái, tốc độ mô hình theo thời gian</i>	<i>119</i>

Danh mục bảng biểu

<i>Bảng 1. Cấu trúc gói tin truyền thông.....</i>	<i>48</i>
<i>Bảng 2. Danh sách tín hiệu và biến báo giao thông</i>	<i>73</i>
<i>Bảng 3. Ý nghĩa các kí hiệu.....</i>	<i>92</i>
<i>Bảng 4. Confusion matrix.....</i>	<i>114</i>

Chương 1: Giới thiệu

1.1. Giới thiệu bài toán hoặc đối tượng nghiên cứu

Xe tự hành, hay còn gọi là xe tự lái, là loại phương tiện có khả năng di chuyển và điều khiển mà không cần sự can thiệp trực tiếp của con người. Dựa vào các cảm biến, hệ thống xử lý tín hiệu và thuật toán trí tuệ nhân tạo, xe tự hành có thể nhận biết môi trường xung quanh, đưa ra quyết định và điều hướng an toàn. Sự phát triển của xe tự hành được kỳ vọng sẽ mang lại nhiều lợi ích cho xã hội như giảm thiểu tai nạn giao thông do lỗi con người, tối ưu hóa lưu thông đô thị, và mang đến giải pháp vận tải hiệu quả cho tương lai.

Để xe có thể tham gia giao thông một cách tự động, việc nhận diện chính xác làn đường và biển báo giao thông đóng vai trò then chốt trong việc đảm bảo an toàn và vận hành hiệu quả. Xe tự hành cần có khả năng phân tích hình ảnh từ hệ thống camera gắn trên thân xe để hiểu rõ cấu trúc đường đi, xác định vị trí của làn đường và đọc hiểu các loại biển báo. Đây là một bài toán phức tạp bởi điều kiện môi trường thường xuyên thay đổi như ánh sáng ban đêm, mưa lớn, sương mù hoặc mặt đường bị hư hỏng.

Nội dung đề tài tập trung vào việc xây dựng hệ thống xử lý hình ảnh và áp dụng các kỹ thuật học sâu (deep learning) nhằm tăng độ chính xác trong việc nhận diện. Hệ thống cần xử lý các hình ảnh đầu vào, trích xuất các đặc trưng hình học của làn đường, phát hiện và phân loại biển báo giao thông một cách nhanh chóng và chính xác. Các phương pháp dựa trên mạng nơ-ron tích chập (CNN) đang cho thấy hiệu quả nổi bật trong việc nhận diện các đối tượng và sẽ được khai thác trong nghiên cứu này.

Bên cạnh đó, đề tài xây dựng và phát triển một bộ điều khiển phù hợp để xử lý các tín hiệu đầu vào giúp xe vận hành một cách chính xác. Việc tích hợp kết quả nhận diện vào hệ thống điều khiển thời gian thực của xe cũng là một thách thức lớn. Xe không chỉ cần "nhìn thấy" mà còn phải "hiểu" và phản ứng kịp thời trước các tín hiệu giao thông. Do đó, nghiên cứu cũng hướng tới việc tối ưu hóa tốc độ xử

lý, giảm thiểu độ trễ và đảm bảo độ tin cậy cao, nhằm tạo ra một nền tảng vững chắc cho các ứng dụng xe tự hành trong tương lai.

1.2. Ý nghĩa khoa học và thực tiễn của việc nghiên cứu đề tài

Nghiên cứu xe tự hành bám làn đường và nhận diện, chấp hành biển báo giao thông có ý nghĩa khoa học quan trọng trong việc ứng dụng các công nghệ thị giác máy tính (Computer Vision) và học sâu (Deep Learning) vào thực tiễn điều khiển phương tiện. Việc sử dụng mô hình YOLO (You Only Look Once) để nhận diện biển báo kết hợp với thuật toán điều khiển bám làn giúp xe tự động xử lý tình huống giao thông gần giống với hành vi lái xe của con người. Thực tiễn cho thấy, việc phát triển và ứng dụng các hệ thống này góp phần giảm thiểu tai nạn, tăng hiệu quả vận hành giao thông, đồng thời mở ra nhiều tiềm năng trong việc xây dựng các hệ thống giao thông thông minh trong tương lai.

1.3. Tổng quan về các nghiên cứu liên quan

Trong nhiều nghiên cứu trước đây về hệ thống xe tự hành, camera thường được gắn cố định, giới hạn trường quan sát của hệ thống trong một góc nhìn nhất định, chủ yếu tập trung về phía trước. Điều này có thể làm giảm hiệu quả trong việc phát hiện các biển báo hoặc vật cản nằm lệch về hai bên làn đường, đặc biệt trong môi trường giao thông phức tạp hoặc có tầm nhìn hạn chế.



Hình 1. Mô hình xe trong các đề tài trước

Ngược lại, hệ thống xe tự hành trong nghiên cứu của nhóm được trang bị camera có khả năng xoay linh hoạt theo hướng di chuyển, giúp mở rộng vùng quan sát và cải thiện khả năng phát hiện làn đường. Đây là một ưu điểm nổi bật, giúp nâng cao độ chính xác và khả năng thích ứng với các tình huống giao thông thực tế.

Việc chủ động điều chỉnh góc nhìn còn hỗ trợ hệ thống trong việc theo dõi làn đường giúp giảm thiểu sai lệch khi xe rẽ hoặc di chuyển qua các đoạn đường cong - điều mà nhiều hệ thống sử dụng camera cố định còn hạn chế.

1.4. Mục tiêu đồ án tốt nghiệp và phương pháp thực hiện

Mục tiêu của đồ án là xây dựng một hệ thống xe tự hành có khả năng:

- Bám theo làn đường dựa trên dữ liệu từ camera.
- Phát hiện và nhận diện biển báo giao thông bằng mô hình YOLOv4-tiny.
- Có thể điều khiển và quan sát các thông tin trạng thái của xe thông qua GUI.
- Ra quyết định và điều chỉnh hành vi lái xe tự động (ví dụ: giảm tốc độ, rẽ, dừng, ...) theo yêu cầu của biển báo nhận diện được hoặc theo yêu cầu của người dùng.

Phương pháp thực hiện gồm:

- Thu thập và tiền xử lý dữ liệu hình ảnh làn đường và biển báo.
- Huấn luyện mô hình YOLOv4-tiny để nhận diện các loại biển báo phổ biến.
- Xây dựng thuật toán xử lý ảnh để phát hiện và theo dõi làn đường.
- Phát triển hệ thống điều khiển xe dựa trên kết quả nhận diện.
- Thủ nghiệm mô hình trên mô hình xe thực tế.

1.5. Sơ lược về nội dung báo cáo đồ án tốt nghiệp

Đồ án bao gồm 6 chương:

Chương 1: Tổng quan về đề tài, cung cấp những thông tin cơ bản về xe tự hành và các ứng dụng thực tiễn của chúng. Giới thiệu mục tiêu, nhiệm vụ và cấu trúc tổng quan của đề tài.

Chương 2: Trình bày cơ sở lý thuyết của các bộ điều khiển, phương pháp xử lý ảnh, các thuật toán sử dụng trong đề tài,

Chương 3: Giới thiệu cấu hình phần cứng, các thiết bị sử dụng trong đề tài. Trình bày chi tiết về cấu trúc, chức năng và nhiệm vụ của từng khối trong phần cứng.

Chương 4: Trình bày cách kiến trúc phần mềm được thiết kế.

Chương 5: Trình bày các giải thuật và phương pháp điều khiển.

Chương 6: Trình bày kết quả thực nghiệm, đánh giá và đưa ra phương hướng phát triển cho đề tài..

Chương 2: Cơ sở lý thuyết

2.1. Mô hình điều khiển

Mô hình xe tự hành được thiết kế dựa trên mô hình xe bốn bánh. Cụ thể, hai bánh trước của robot được điều khiển góc lái bởi động cơ Servo, trong khi hai bánh sau chịu trách nhiệm điều khiển tốc độ thông qua động cơ DC.

Mô hình điều khiển của hệ thống được xây dựng với mục tiêu đảm bảo xe tự hành di chuyển ổn định trong làn đường, sử dụng các thông số đã qua xử lý thị giác máy làm đầu vào.

2.1.1. Cấu trúc hệ thống điều khiển

2.1.1.1. Máy tính nhúng xử lý thị giác máy và camera

Máy tính nhúng NVIDIA Jetson TX-2 được kết nối với camera hành trình Gopro Hero 8 qua đường USB. Có nhiệm vụ xử lý hình ảnh cho phép nhận diện làn đường, vạch kẻ đường và xác định tọa độ của điểm chính giữa làn đường trong khung hình, độ nghiêng của làn đường và nhận diện biển báo giao thông.

Trong trường hợp bình thường như khi đi theo làn đường, ta cần phải đảm bảo rằng lúc nào khung hình cũng có thông tin đầy đủ về hai vạch kẻ đường ở cả hai bên trái và phải. Điểm chính giữa làn đường và góc nghiêng của làn đường được sử dụng làm đầu vào cho hệ thống điều khiển.

Còn trong trường hợp khi rẽ trái, rẽ phải tại ngã tư, ta sẽ ưu tiên về việc nhìn thấy vạch kẻ đường của phía tương ứng. Vị trí của vạch kẻ đường của phía tương ứng khi rẽ trái, rẽ phải sẽ là đầu vào cho hệ thống điều khiển.

Các giá trị đầu ra của hệ thống xử lý ảnh sẽ được truyền đến hệ thống điều khiển qua đường UART.

2.1.1.2. Bộ điều khiển Fuzzy điều chỉnh góc lái

a. Trường hợp đi theo làn đường

Điểm trung tâm làn đường, và góc nghiêng của làn đường tìm được khi đặt mô hình xe tự hành ở vị trí trung tâm và dọc theo làn đường thẳng không nghiên đóng vai trò là hai giá trị đặt (setpoint) của bộ điều khiển.

Bộ điều khiển Fuzzy điều chỉnh góc lái trong trường hợp đi theo làn đường gồm ba ngõ vào và một ngõ ra.

Sau khi nhận được tọa độ của điểm giữa làn đường và góc nghiêng làn đường từ máy tính nhúng thông qua đường dữ liệu UART, dựa vào các giá trị đặt đã xác định từ trước, sẽ tính toán được sai số tọa độ, tốc độ biến thiên sai số tọa độ và sai số góc nghiêng.

Bộ điều khiển Fuzzy được sử dụng để điều chỉnh góc lái dựa vào:

- Ngõ vào 1: Sai số tọa độ.
- Ngõ vào 2: Tốc độ biến thiên sai số tọa độ.
- Ngõ vào 3: Sai số góc nghiêng
- Ngõ ra: Góc lái cần thiết để duy trì tọa độ điểm giữa làn đường và góc nghiêng của làn đường luôn duy trì ở giá trị đặt. Lưu ý rằng góc lái ở đây chính là góc lái của động cơ DC Servo để điều khiển cơ cấu đánh lái cho mô hình xe tự hành.

b. Trường hợp rẽ trái, rẽ phải tại ngã tư hoặc ngã ba

Một điểm cố định trong khung hình lệch về phía mà xe muốn rẽ được chọn làm điểm đặt. Ứng với mỗi phía phải và trái sẽ có điểm đặt tương ứng và được chọn dựa vào thực nghiệm.

Bộ điều khiển Fuzzy điều chỉnh góc lái trong trường hợp rẽ trái, rẽ phải tại ngã tư hoặc ngã ba gồm hai ngõ vào và một ngõ ra.

Sau khi nhận được tọa độ của cửa vạch kẻ đường từ máy tính nhúng thông qua đường dữ liệu UART, dựa vào giá trị đặt đã xác định từ trước, sẽ tính toán được sai số tọa độ, tốc độ biến thiên sai số tọa độ.

Bộ điều khiển Fuzzy được sử dụng để điều chỉnh góc lái dựa vào:

- Ngõ vào 1: Sai số tọa độ.
- Ngõ vào 2: Tốc độ biến thiên sai số tọa độ.
- Ngõ ra: Góc lái cần thiết để duy trì tọa độ vạch kẻ đường xác định được luôn duy trì ở giá trị đặt. Lưu ý rằng góc lái ở đây chính là góc lái của động cơ DC Servo để điều khiển cơ cấu đánh lái cho mô hình xe tự hành.

2.1.1.3. Bộ điều khiển Fuzzy điều chỉnh tốc độ đặt

Để đảm bảo mô hình xe tự hành chạy ổn định, tốc độ đặt của robot cần phải thay đổi dựa vào độ lớn của góc lái và tốc độ hiện tại.

Bộ điều khiển Fuzzy được sử dụng để điều chỉnh tốc độ đặt dựa vào:

- Ngõ vào 1: Tốc độ hiện tại của robot.
- Ngõ vào 2: Góc lái hiện tại được tính toán từ bộ Fuzzy trước đó.
- Ngõ ra: Tốc độ xe tự hành cần chạy để phù hợp với điều kiện di chuyển, nhằm duy trì sự ổn định và chính xác trong khi di chuyển.

2.1.1.4. Bộ điều khiển PID tốc độ động cơ

Để điều khiển tốc độ quay của hai bánh sau, sử dụng một bộ điều khiển PID với mục tiêu duy trì tốc độ di chuyển của xe tự hành ổn định đúng theo tốc độ đặt.

Bộ điều khiển PID được sử dụng để điều khiển tốc độ dựa vào:

- Ngõ vào 1: Tốc độ thực tế của động cơ DC, đo được bằng encoder.
- Ngõ vào 2: Giá trị đặt của tốc độ được tính toán từ bộ điều khiển Fuzzy trước đó.
- Ngõ ra: Tín hiệu điều khiển động cơ, đảm bảo tốc độ của xe tự hành đạt được giá trị mong muốn với độ chính xác cao.

2.1.1.5. Giao diện người dùng (GUI)

Giao diện người dùng (GUI) được thiết kế nhằm hỗ trợ quá trình vận hành, giám sát và hiệu chỉnh hệ thống xe tự hành một cách thuận tiện. GUI kết nối với hệ thống xe thông qua module Bluetooth HC-05, sử dụng giao thức UART để trao đổi dữ liệu.

Trong quá trình vận hành, GUI thực hiện chức năng thu thập và hiển thị các thông tin trạng thái của mô hình xe tự hành, bao gồm các thông số như tốc độ thực tế của động cơ, góc lái hiện tại, sai số tọa độ trung tâm làn đường, góc nghiêng của làn đường, trạng thái nhận diện biển báo giao thông và trạng thái các bộ điều khiển. Các thông tin này được cập nhật liên tục theo thời gian thực, cho phép người dùng theo dõi toàn bộ hoạt động của hệ thống một cách trực quan.

Ngoài chức năng giám sát, GUI còn cho phép người dùng thay đổi các thông số cài đặt của các bộ điều khiển Fuzzy. Người dùng có thể chỉnh sửa các tham số như giới hạn giá trị vào/ra, giá trị điểm đặt (setpoint).

2.1.2. Mục tiêu của mô hình điều khiển

Mục tiêu chính của mô hình điều khiển là đảm bảo mô hình xe tự hành di chuyển ổn định, chính xác, và linh hoạt trong làn đường. Sự kết hợp giữa bộ điều khiển Fuzzy cho góc lái, bộ điều khiển Fuzzy cho tốc độ đặt và bộ điều khiển PID cho điều khiển tốc độ động cơ giúp tối ưu hóa khả năng thích ứng của xe tự hành với các điều kiện môi trường thực tế, đồng thời nâng cao hiệu suất và độ chính xác của hệ thống.

2.2. Xử lý ảnh

2.2.1. Phương pháp Otsu Thresholding

Otsu Thresholding là một phương pháp phân ngưỡng tự động trong xử lý ảnh, nhằm mục tiêu phân tách ảnh thành hai vùng: foreground (đối tượng cần quan tâm) và background (nền). Phương pháp này hoạt động dựa trên việc phân tích histogram (biểu đồ tần suất mức xám) của ảnh và tìm ra ngưỡng sao cho sự phân biệt giữa hai vùng là rõ rệt nhất.

Về mặt lý thuyết, Otsu tìm một ngưỡng t sao cho phương sai giữa các lớp đạt giá trị lớn nhất, hay tương đương với việc tổng phương sai trong lớp là nhỏ nhất.

Cụ thể, giả sử ta chọn một ngưỡng t , ảnh sẽ được chia thành hai lớp:

- Lớp C_0 gồm các pixel có mức xám từ 0 đến t .
- Lớp C_1 gồm các pixel có mức xám từ $t + 1$ đến $L - 1$ (với L là mức xám tối đa, thường là 255).

Ta định nghĩa:

- $\omega_0(t)$: xác suất (tỉ lệ) các pixel thuộc lớp C_0 ,
- $\omega_1(t)$: xác suất các pixel thuộc lớp C_1 ,
- $\mu_0(t)$: trung bình mức xám của lớp C_0 ,
- $\mu_1(t)$: trung bình mức xám của lớp C_1 .

Công thức tính phương sai giữa các lớp được cho bởi:

$$\sigma_b^2(t) = \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2$$

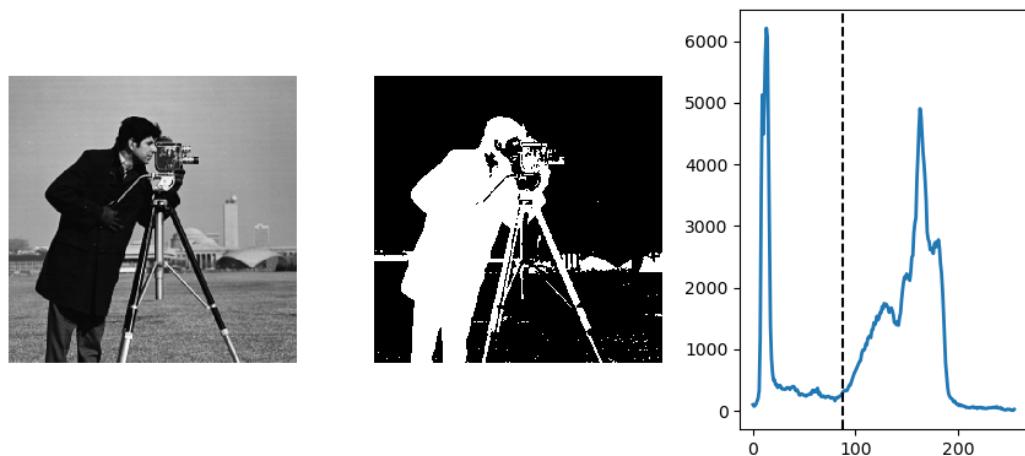
Ý nghĩa của công thức trên là: phương sai giữa hai lớp sẽ lớn nếu tỉ lệ hai lớp cân bằng và mức xám trung bình giữa hai lớp chênh lệch nhiều.

Thuật toán Otsu tiến hành lần lượt tính $\sigma_b^2(t)$ cho tất cả các giá trị ngưỡng t từ 0 đến 255, sau đó chọn ra giá trị t^* sao cho:

$$t^* = \arg_t \max \sigma_b^2(t)$$

Nói cách khác, ngưỡng tối ưu là ngưỡng mà phương sai giữa các lớp đạt giá trị cực đại.

Phương pháp Otsu hoạt động rất hiệu quả khi histogram của ảnh có hai đỉnh rõ rệt (ảnh dạng bimodal). Tuy nhiên, nếu ảnh có nhiều vùng xám hoặc histogram phức tạp, Otsu có thể không tìm được ngưỡng phân tách lý tưởng.



Hình 2. Kết quả phương pháp Otsu

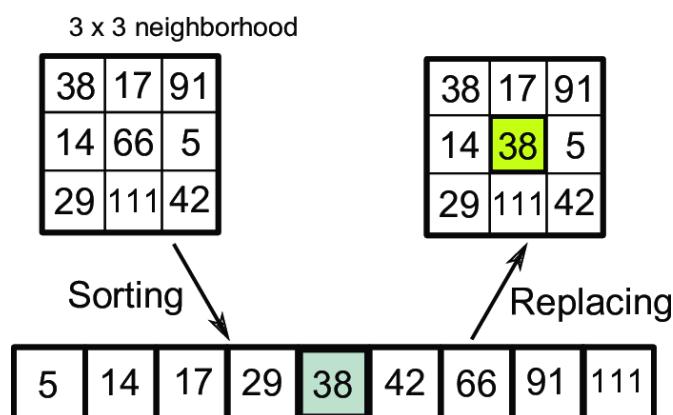
2.2.2. Phương pháp Median Blur

Median Blur là một kỹ thuật lọc phi tuyến trong xử lý ảnh, được dùng để làm mịn ảnh và đặc biệt hiệu quả trong việc khử nhiễu muối tiêu.

Khác với các phương pháp làm mờ tuyến tính như Gaussian Blur (lọc bằng trung bình có trọng số), Median Blur hoạt động dựa trên việc thay thế mỗi pixel bằng giá trị trung vị trong vùng lân cận. Trung vị là giá trị nằm giữa dãy số sau khi các giá trị pixel trong cửa sổ được sắp xếp theo thứ tự tăng dần.

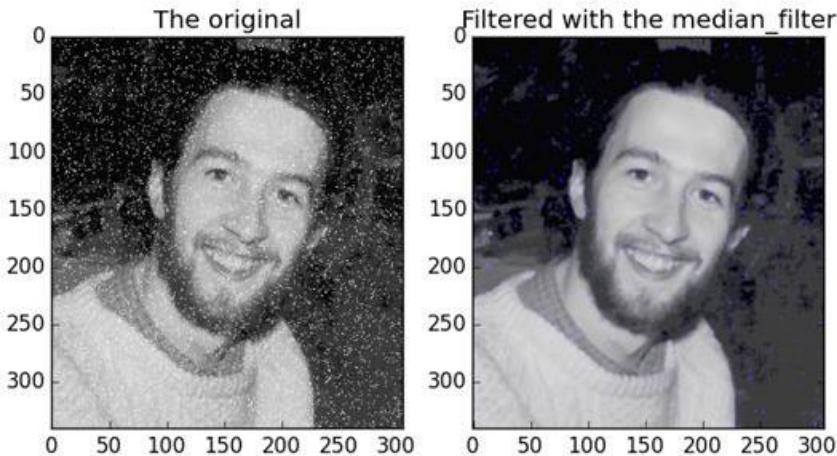
Cách thực hiện Median Blur có thể mô tả theo các bước sau:

- Đối với mỗi pixel trên ảnh gốc, lấy tất cả các giá trị pixel trong một cửa sổ $k \times k$ (ví dụ $3 \times 3, 5 \times 5, \dots$)
- Sắp xếp các giá trị trong cửa sổ theo thứ tự tăng dần.
- Lấy giá trị ở giữa (trung vị) và thay thế pixel trung tâm bằng giá trị đó.



Hình 3. Mô tả hoạt động của bộ lọc trung vị

Ưu điểm lớn của Median Blur là khả năng loại bỏ các điểm nhiễu cực trị (rất nhỏ hoặc rất lớn) mà không làm mờ các chi tiết cạnh trong ảnh, vì trung vị không bị ảnh hưởng mạnh bởi các giá trị ngoại lai như giá trị trung bình. Điều này đặc biệt hữu ích khi xử lý ảnh chứa nhiễu muối tiêu.



Hình 4. Hiệu quả của bộ lọc trung vị trong khử nhiễu muối tiêu

Tuy nhiên, nếu lựa chọn kích thước cửa sổ quá lớn, ảnh sau lọc có thể bị làm mờ quá mức và mất chi tiết. Đồng thời, Median Blur cũng không xử lý hiệu quả đối với các loại nhiễu Gaussian.

2.2.3. Phương pháp phát hiện đường biên

Trong xử lý ảnh số, việc tìm đường bao (contour detection) là một bước quan trọng nhằm xác định biên giới giữa các vùng khác nhau trong ảnh, đặc biệt là giữa đối tượng và nền. Về tổng quát, phương pháp tìm đường bao dựa trên việc phát hiện những sự thay đổi giá trị cường độ đột ngột trong ảnh hoặc dựa trên việc nhóm các pixel có tính chất tương đồng và liên thông.

Đầu tiên, ảnh đầu vào thường được chuyển đổi thành ảnh nhị phân thông qua các kỹ thuật như phân ngưỡng (thresholding) hoặc phát hiện biên (edge detection). Mục tiêu của bước này là làm cho đối tượng và nền được phân tách rõ ràng, giúp dễ dàng nhận diện các khu vực cần tìm đường bao.

Tiếp theo, thuật toán tiến hành quét từng pixel trong ảnh để xác định các pixel thuộc về vật thể (pixel foreground, giá trị 255 trong ảnh nhị phân). Khi một

pixel thuộc vật thể được phát hiện, thuật toán sẽ kiểm tra các pixel lân cận theo nguyên tắc liên thông (connectivity), có thể là liên thông 4 chiều (trên, dưới, trái, phải) hoặc 8 chiều (thêm các đường chéo). Dựa trên thông tin liên thông này, các pixel được nhóm lại thành các cụm liên kết, và đường bao của mỗi cụm được xác định bằng cách nối các pixel biên liên tiếp.

Về mặt hình thức, một contour được định nghĩa như tập hợp các điểm (x, y) thỏa mãn điều kiện:

$$C = \{(x, y) \mid I(x, y) = k, \exists (x', y') \in \text{lân cận } I(x', y') \neq k\}$$

Trong đó:

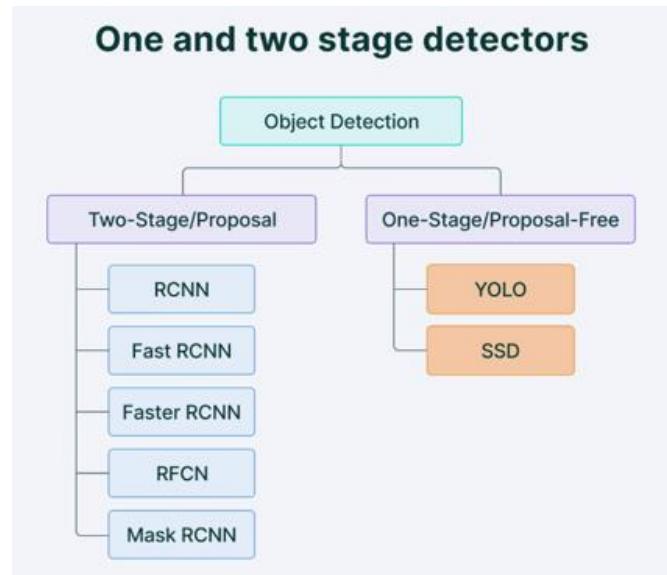
- $I(x, y)$ là giá trị cường độ tại điểm ảnh (x, y)
- k là ngưỡng phân biệt vật thể với nền. Điều kiện này đảm bảo rằng pixel nằm tại biên của một vùng, vì trong lân cận của nó có sự thay đổi giá trị.

Sau khi phát hiện được toàn bộ các điểm trên đường bao, dữ liệu contour có thể được biểu diễn dưới dạng một chuỗi các điểm liên tiếp, hoặc được rút gọn để tối ưu hóa lưu trữ bằng cách chỉ lưu những điểm đặc trưng (ví dụ: góc ngoặt).

2.3. YOLO cho nhận diện đối tượng (Object Detection)

2.3.1. Giới thiệu chung về YOLO

Có lẽ trong vài năm trở lại đây nhận diện đối tượng là một trong những đề tài nổi bật của deep learning bởi khả năng ứng dụng cao. Các thuật toán mới của object detection như YOLO, SSD có tốc độ khá nhanh và độ chính xác cao nên giúp cho nhận diện đối tượng có thể thực hiện được các tác vụ gần như là thời gian thực. Các mô hình cũng trở nên nhẹ hơn nên có thể hoạt động trên các thiết bị IoT để tạo nên các thiết bị thông minh.



Hình 5. Các phương pháp phát hiện vật thể

Các phương pháp phát hiện vật thể được chia thành hai nhóm chính: phát hiện hai giai đoạn (two-stage) và phát hiện một giai đoạn (one-stage).

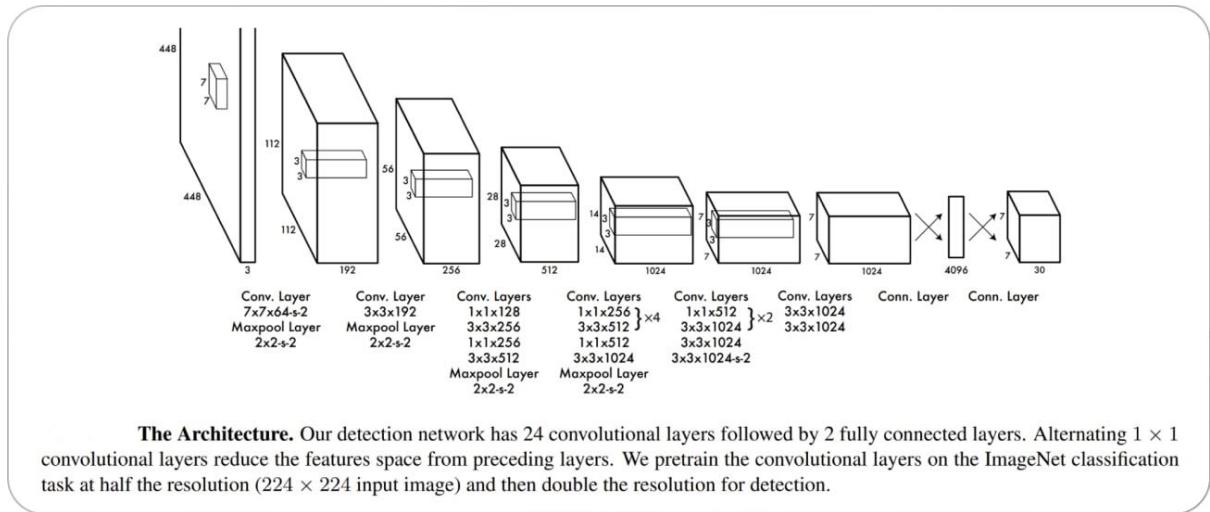
- **Phát hiện hai giai đoạn:** các mô hình như RCNN, Fast RCNN, Faster RCNN, RFCN và Mask RCNN. Các mô hình này đầu tiên tạo ra các vùng đề xuất (region proposals), sau đó thực hiện phân loại và định vị chính xác hơn. Do đó, chúng đạt độ chính xác cao nhưng yêu cầu tài nguyên tính toán lớn hơn.
- **Phát hiện một giai đoạn:** tiêu biểu là YOLO và SSD, bỏ qua bước đề xuất vùng và xử lý toàn bộ ảnh trong một lượt duy nhất. Điều này giúp tăng tốc độ xử lý, phù hợp với các ứng dụng thời gian thực, tuy nhiên độ chính xác có thể giảm, đặc biệt với các vật thể nhỏ.

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19

Hình 6. Bảng so sánh hiệu suất các model

Với các đặc điểm nổi bật như nhanh, chạy được real-time, end-to-end, không tách riêng bước region proposal như R-CNN, phù hợp cho ứng dụng trên thiết bị nhúng và hệ thống yêu cầu tốc độ cao. Để lựa chọn mô hình phù hợp với mục tiêu

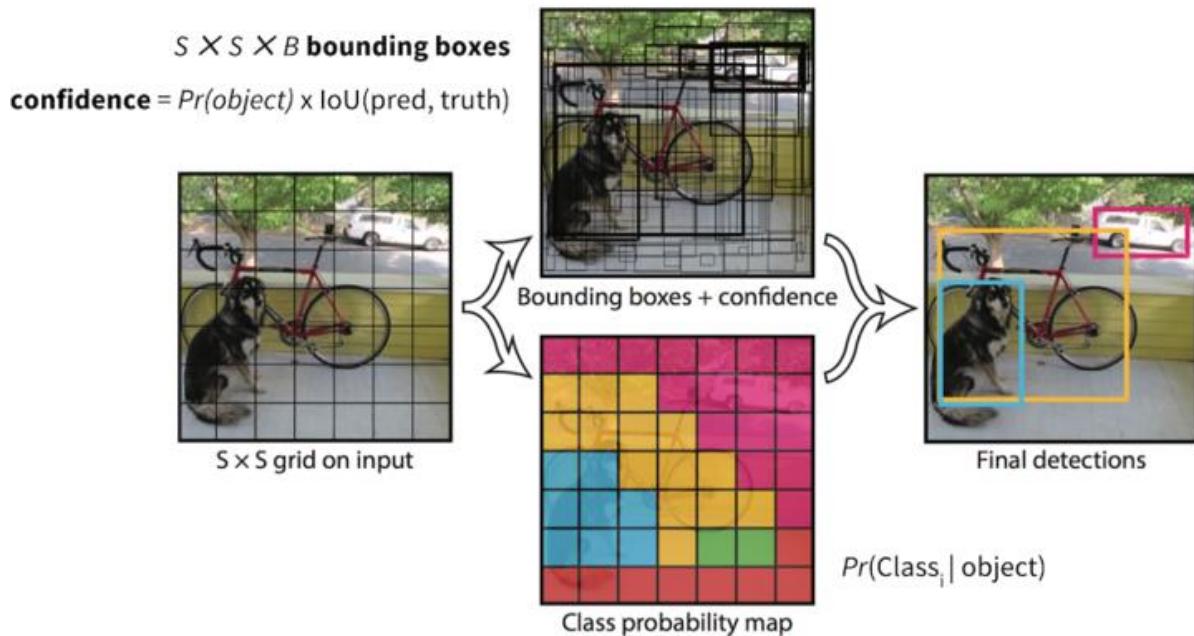
ứng dụng ưu tiên tốc độ xử lí như xe tự hành tham gia giao thông nhóm quyết định chọn one-stage YOLO.



Hình 7. Kiến trúc YOLO

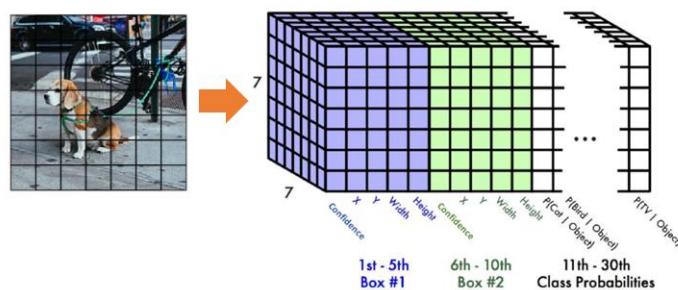
Kiến trúc mạng phát hiện vật thể của YOLO bao gồm 24 lớp tích chập (convolutional layers) và 2 lớp kết nối đầy đủ (fully connected layers). Mạng sử dụng xen kẽ các lớp tích chập 1×1 và 3×3 để giảm không gian đặc trưng mà vẫn giữ lại thông tin quan trọng từ các lớp trước. Các lớp maxpool được chèn vào nhằm giảm kích thước đầu ra không gian theo từng giai đoạn. Mô hình được huấn luyện ban đầu trên tập ImageNet cho bài toán phân loại với ảnh đầu vào kích thước 224×224 , sau đó mở rộng lên độ phân giải 448×448 để thực hiện phát hiện vật thể. Cấu trúc này cho phép YOLO xử lý toàn bộ ảnh trong một lần duy nhất, đưa ra dự đoán đồng thời về vị trí và lớp của các vật thể trong ảnh.

2.3.2. Cách hoạt động của YOLO



Hình 8. Mô tả tổng quan YOLO

Kiến trúc này chia ảnh đầu vào thành một lưới có kích thước $S \times S$. Nếu tâm của hộp chứa vật thể (bounding box) nằm trong một ô lưới nào đó, thì ô lưới đó sẽ chịu trách nhiệm phát hiện vật thể đó. Mỗi ô lưới sẽ dự đoán các hộp chứa cùng với điểm tin cậy (confidence score) tương ứng. Mỗi điểm tin cậy thể hiện mức độ chắc chắn rằng hộp dự đoán thực sự chứa một vật thể, đồng thời phản ánh độ chính xác của các tọa độ hộp so với nhãn thực tế (ground truth).



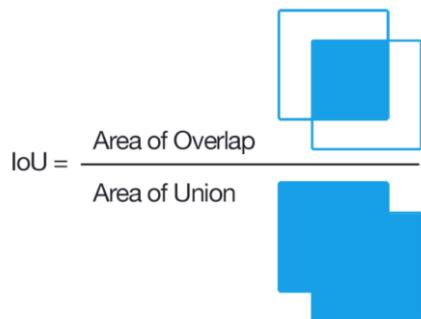
Hình 9. Đầu ra mỗi ảnh của YOLO

Mỗi ô lưới sẽ chịu trách nhiệm dự đoán các đối tượng có tâm nằm trong phạm vi của ô đó. Cụ thể, mỗi ô sẽ dự đoán B hộp giới hạn, với mỗi hộp bao gồm 5 giá trị: tọa độ trung tâm (x, y), chiều rộng w , chiều cao h , và độ tin cậy (confidence

score). Trong đó, (x,y) được tính tương đối với ô lưới, còn w và h được tính tương đối với toàn bộ ảnh đầu vào.

Độ tin cậy phản ánh hai yếu tố: khả năng có vật thể trong hộp và mức độ khớp giữa hộp dự đoán với hộp thực, được đo bằng chỉ số Intersection over Union (IoU). Nếu không có vật thể nào trong ô, giá trị confidence phải bằng 0; ngược lại, confidence chính bằng giá trị IoU.

$$\text{Confidence Score} = P(\text{Object}) \times \text{IoU}$$



Hình 10. Tính chỉ số IoU

Bên cạnh đó, mỗi ô cũng dự đoán xác suất có điều kiện ứng với C lớp đối tượng, ký hiệu là $\text{Pr}(\text{Class i}|\text{Object})$. Khi suy luận, xác suất từng lớp được nhân với độ tin cậy của từng hộp để thu được xác suất có điều kiện theo lớp:

$$\text{Pr}(\text{Class i}) \times \text{IoU} = \text{Pr}(\text{Class i}|\text{Object}) \times \text{Pr}(\text{Object}) \times \text{IoU}$$

Kết quả đầu ra của toàn bộ mô hình được biểu diễn dưới dạng một tensor có kích thước $S \times S \times (B \times 5 + C)$, trong đó tổng hợp đầy đủ thông tin về vị trí, kích thước, độ tin cậy và xác suất lớp cho từng đối tượng trong ảnh.

Non-Maximum Suppression (NMS) là một bước hậu xử lý quan trọng trong các thuật toán phát hiện đối tượng như YOLO. Do đặc tính mỗi ảnh đầu vào có thể sinh ra rất nhiều bounding box, đặc biệt với các cell lân cận trong ảnh, các bounding box dễ bị chồng lặp lên nhau. Vì vậy, NMS được sử dụng để loại bỏ các bounding box dư thừa, chỉ giữ lại các box có khả năng cao nhất đại diện cho đối tượng thực sự. Quy trình của NMS diễn ra qua hai bước chính:

Đầu tiên, tất cả các bounding box có độ tin cậy (confidence score) nhỏ hơn một ngưỡng định trước (thường là 0.5) sẽ bị loại bỏ, nhằm giảm bớt các dự đoán yếu và không chính xác. Tiếp theo, trong số các bounding box còn lại, NMS chọn ra box có confidence cao nhất và tính IoU giữa box này với các box còn lại. Nếu IoU vượt quá một ngưỡng (thường khoảng 0.5), các box bị cho là trùng lặp và sẽ bị loại bỏ, chỉ giữ lại box chính xác nhất. Quá trình này lặp lại cho đến khi không còn box nào cần xử lý.



Hình 11. Kết quả của NMS

Kết quả là mỗi đối tượng trong ảnh chỉ được xác định bởi một bounding box duy nhất, giúp tăng độ chính xác và giảm nhiễu trong quá trình dự đoán.

Hàm mất mát trong mô hình YOLO được thiết kế để tối ưu hóa hiệu suất phát hiện đối tượng bằng cách kết hợp ba thành phần chính: loss tọa độ, loss đối tượng (objectness) và loss phân loại. Biểu thức tổng thể của hàm mất mát bao gồm nhiều thành phần phản ánh sai số trong việc dự đoán vị trí, kích thước, sự tồn tại của đối tượng và phân loại.

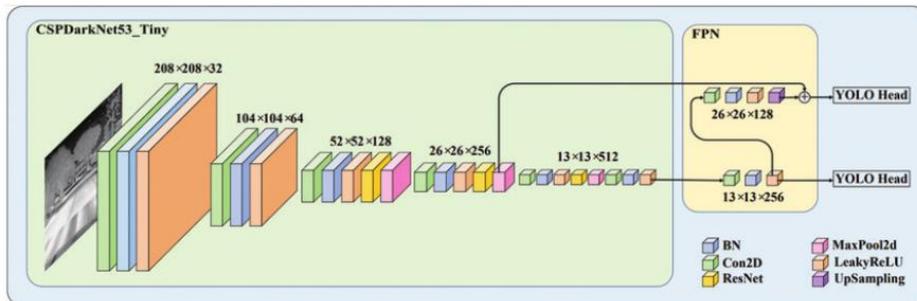
$$\begin{aligned}
& \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
& + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] \\
& + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
& + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
& + \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}$$

Hình 12. Hàm loss của YOLO

- Loss tọa độ đánh giá sai số giữa vị trí và kích thước bounding box dự đoán và thực tế, chỉ áp dụng cho các ô có chứa vật thể, được điều chỉnh bằng hệ số λ_{coord} .
- Loss đối tượng phản ánh độ chính xác của confidence score, gồm cả box có và không có vật thể, với box không chứa vật thể được điều chỉnh bằng λ_{noobj}
- Loss phân loại đo sai số giữa phân phối xác suất các lớp được dự đoán và nhãn thật tại các ô chứa vật thể.

Tổng thể, hàm mất mát của YOLO là một biểu thức bình phương kết hợp nhiều thành phần, giúp mô hình học được cách phát hiện chính xác vị trí, kích thước và loại của các đối tượng trong ảnh đầu vào. Thiết kế này đảm bảo mô hình được huấn luyện hiệu quả trên cả ba khía cạnh: định vị, nhận diện sự tồn tại và phân loại.

2.3.3. YOLOv4-Tiny – Phiên bản nhẹ và hiệu quả

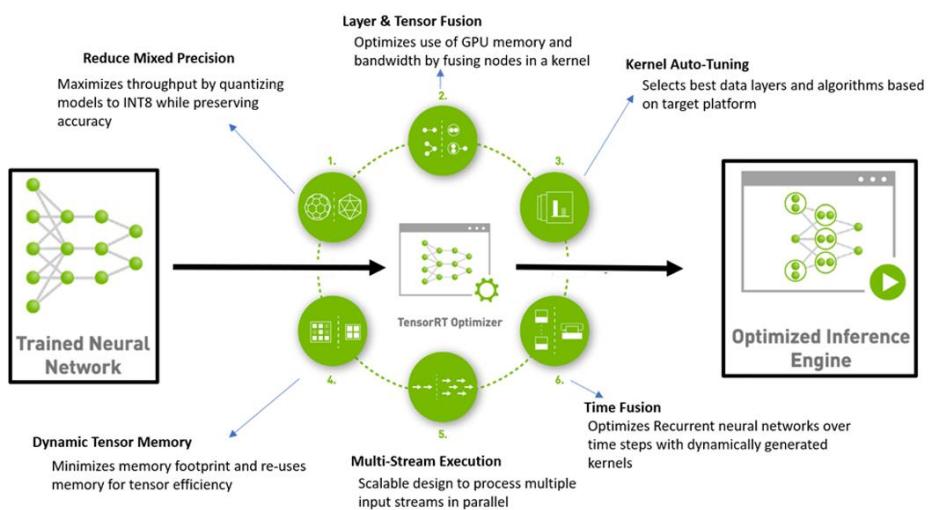


YOLOv4-Tiny network architecture. Extracted from the input image by CSPDarkNet53_Tiny, features will be fused in the FPN module and passed into YOLO head

Hình 13. Kiến trúc YOLOv4-tiny

YOLOv4-Tiny là phiên bản rút gọn của YOLOv4, được thiết kế đặc biệt để phù hợp với các thiết bị có tài nguyên phần cứng hạn chế như Jetson TX2, Nano. Mục tiêu chính của phiên bản này là giảm tải tính toán, tăng tốc độ xử lý trong khi vẫn duy trì hiệu suất nhận dạng ở mức chấp nhận được. YOLOv4-Tiny sử dụng backbone CSPDarknet-19, một kiến trúc nhẹ và hiệu quả. Phiên bản này áp dụng các kỹ thuật đơn giản hóa mô hình, cắt giảm số lượng tầng và tham số nhằm giảm thiểu độ phức tạp, nhưng vẫn giữ được hiệu quả phát hiện vật thể ở mức tốt. YOLOv4-Tiny có tốc độ suy luận (inference) rất nhanh, có thể đạt hàng chục khung hình mỗi giây (FPS) ngay cả trên GPU yếu. Dù độ chính xác không bằng bản đầy đủ YOLOv4, nhưng sự đánh đổi giữa tốc độ và độ chính xác là hợp lý, đặc biệt phù hợp cho các ứng dụng thời gian thực như xe tự hành, robot hoặc hệ thống giám sát thông minh.

Triển khai mô hình YOLOv4-Tiny bằng TensorRT



Hình 14. TensorRT

Để tối ưu hóa hiệu suất suy luận (inference) trên máy tính nhúng Jetson, NVIDIA cung cấp TensorRT—một SDK chuyên dụng giúp tăng tốc độ và giảm độ trễ trong quá trình triển khai mô hình học sâu trên GPU. TensorRT thực hiện các kỹ thuật như:

- Hợp nhất các lớp (layer fusion): Kết hợp các lớp mạng để giảm số lượng phép toán cần thực hiện.

- Giảm độ chính xác (precision): Chuyển đổi từ FP32 sang FP16 hoặc INT8 để giảm kích thước mô hình và tăng tốc độ xử lý.
- Tối ưu hóa bố cục bộ nhớ (memory layout): Sắp xếp dữ liệu trong bộ nhớ để truy cập nhanh hơn.

Lợi ích khi sử dụng TensorRT trên thiết bị nhúng:

- Tăng tốc độ suy luận: TensorRT có thể tăng tốc độ suy luận lên gấp 2-4 lần so với mô hình gốc.
- Giảm độ trễ (latency): Thời gian phản hồi của hệ thống được rút ngắn, phù hợp với các ứng dụng thời gian thực.
- Tiết kiệm năng lượng: Việc tối ưu hóa mô hình giúp giảm mức tiêu thụ điện năng, kéo dài thời gian hoạt động của thiết bị.

Quy trình triển khai YOLOv4-Tiny với TensorRT là xuất mô hình sang định dạng ONNX bằng cách sử dụng các công cụ chuyển đổi để xuất mô hình YOLOv4-Tiny sang định dạng ONNX, một định dạng trung gian phổ biến cho các mô hình học sâu. Sử dụng TensorRT để chuyển đổi mô hình ONNX thành TensorRT Engine, một định dạng tối ưu hóa cho suy luận trên GPU. Sau đó triển khai mô hình TensorRT Engine trên các thiết bị như Jetson Nano, TX2 hoặc Xavier để thực hiện suy luận. Thực tế việc triển khai YOLOv4-Tiny bằng TensorRT rất phù hợp cho các ứng dụng yêu cầu xử lý thời gian thực và tiết kiệm năng lượng.

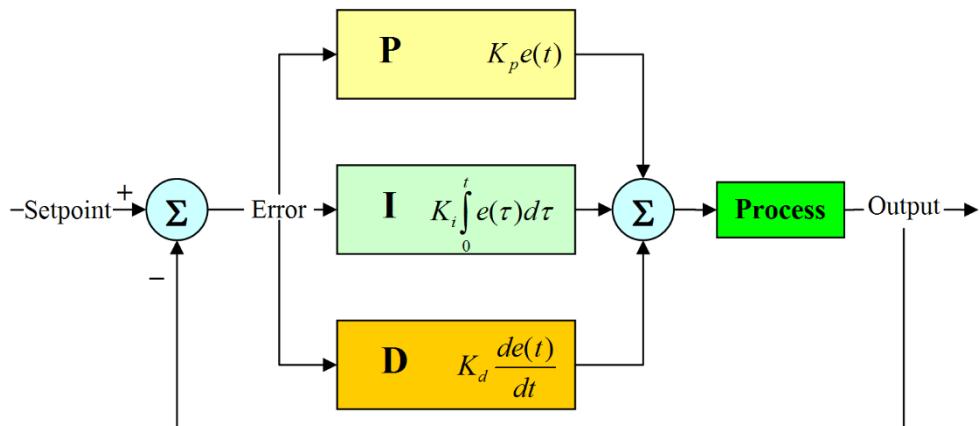
2.4. Bộ điều khiển PID

Bộ điều khiển vi tích phân tỉ lệ (bộ điều khiển PID - Proportional Integral Derivative) là một cơ chế phản hồi vòng điều khiển (bộ điều khiển) tổng quát được sử dụng rộng rãi trong các hệ thống điều khiển – bộ điều khiển PID là bộ điều khiển được sử dụng nhiều nhất trong các bộ điều khiển phản hồi.

Bộ điều khiển PID sẽ tính toán giá trị "sai số" là hiệu số giữa giá trị đo thông số biến đổi và giá trị đặt mong muốn. Bộ điều khiển sẽ thực hiện giảm tối đa sai số bằng cách điều chỉnh giá trị điều khiển đầu vào. Trong trường hợp không có kiến thức cơ bản (mô hình toán học) về hệ thống điều khiển thì bộ điều khiển PID là sẽ bộ điều khiển tốt nhất.

Tuy nhiên, để đạt được kết quả tốt nhất, các thông số PID sử dụng trong tính toán phải điều chỉnh theo tính chất của hệ thống - trong khi kiểu điều khiển là giống nhau, các thông số phải phụ thuộc vào đặc thù của hệ thống.

Bộ điều khiển PID gồm 3 thông số đặc trưng: các giá trị tỉ lệ, tích phân và đạo hàm, viết tắt là P, I, và D. Giá trị tỉ lệ xác định tác động của sai số hiện tại, giá trị tích phân xác định tác động của tổng các sai số quá khứ, và giá trị vi phân xác định tác động của tốc độ biến đổi sai số.



Hình 15. Sơ đồ bộ điều khiển PID

Gọi e là sai số của quá trình, được xác định: $e = SP - PV$

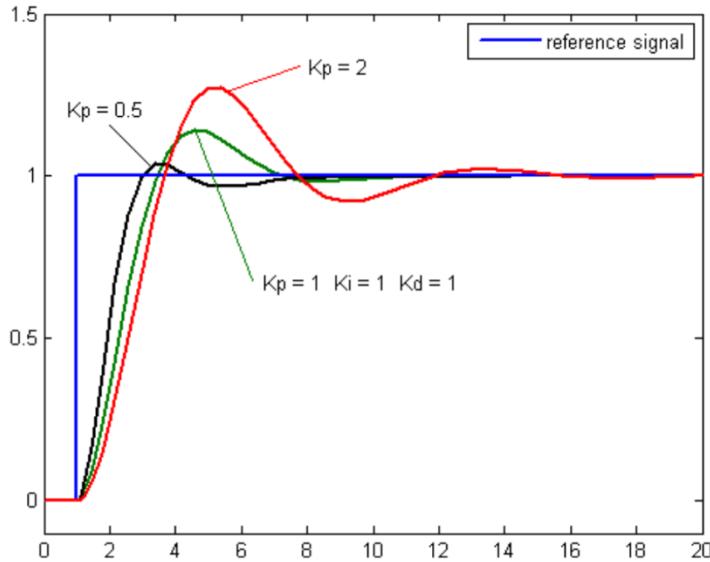
Trong đó: SP là giá trị đặt, PV là tín hiệu đầu ra.

Nhiệm vụ của bộ điều khiển phản hồi là điều chỉnh giá trị đầu vào theo giá trị e đo được để giá trị đầu ra tiến tới giá trị đặt. Tuy nhiên nếu chỉ cộng hoặc trừ e vào giá trị điều khiển thì thời gian đáp ứng của hệ thống có thể sẽ nhanh hơn hoặc chậm hơn thời gian đáp ứng mong muốn.

Do đó cần đến khâu tỉ lệ P. Khâu tỉ lệ làm thay đổi giá trị đầu ra, tỉ lệ với giá trị sai số hiện tại. Đáp ứng tỉ lệ có thể được điều chỉnh bằng cách nhân sai số đó với một hằng số K_p , được gọi là hệ số tỉ lệ.

Khâu tỉ lệ được cho bởi: $P = K_p \times e$

Nếu hệ số tỉ lệ quá cao, hệ thống sẽ không ổn định. Nếu hệ số tỉ lệ quá thấp, tác động điều khiển có thể sẽ quá bé khi đáp ứng với các nhiễu của hệ thống.



Hình 16. Đáp ứng của hệ thống khi tăng dần KP

Khi sai số tồn tại trong một khoảng thời gian dài về một phía của giá trị đặt, ta cần tăng tốc chuyển động của quá trình tới điểm đặt. Nhờ có khâu tích phân ta có thể giải quyết được vấn đề này.

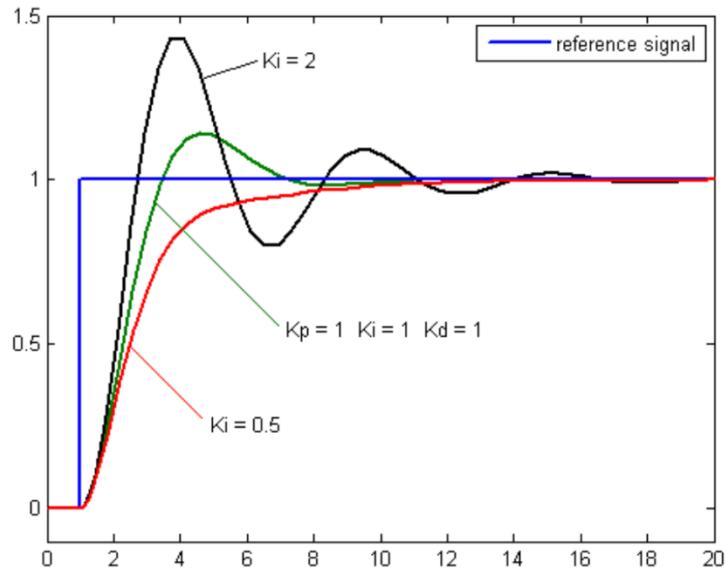
Khâu tích phân tỉ lệ thuận với cả biên độ sai số lẫn quãng thời gian xảy ra sai số. Tổng sai số tức thời theo thời gian cho ta tích lũy bù đã được hiệu chỉnh trước đó. Tích lũy sai số sau đó được nhân với độ lợi tích phân và cộng với tín hiệu đầu ra của bộ điều khiển. Biên độ phân phối của khâu tích phân trên tất cả tác động điều chỉnh được xác định bởi hệ số tích phân K_I .

Thìa số tích phân được cho bởi: $I = K_I \int_0^t e(\tau) d\tau$

Trong đó: K_I là hệ số tích phân

$e(t)$ là hàm sai lệch theo thời gian

Tuy nhiên, vì khâu tích phân là đáp ứng của sai số tích lũy trong quá khứ, nó có thể khiến giá trị hiện tại vọt lố qua giá trị đặt (ngang qua điểm đặt và tạo ra một độ lệch với các hướng khác), do đó cần phải chú ý khi sử dụng khâu tích phân.



Hình 17. Đáp ứng của hệ thống khi tăng dần K_I

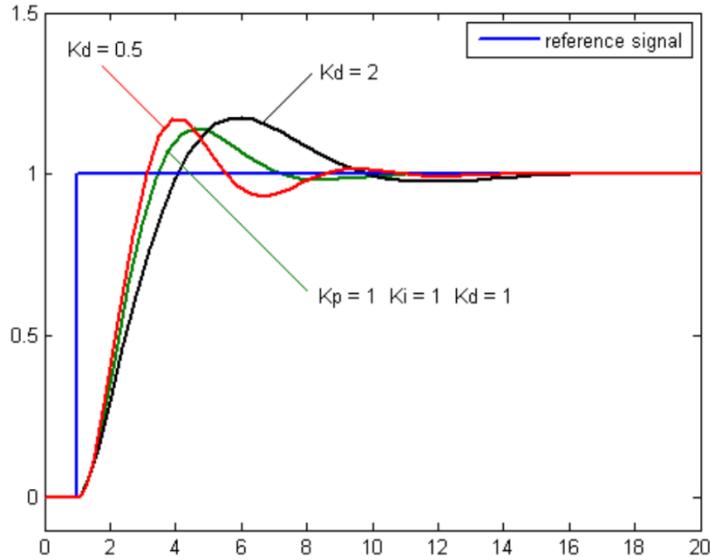
Tốc độ thay đổi của sai số qua trình được tính toán bằng cách xác định độ dốc của sai số theo thời gian (tức là đạo hàm bậc một theo thời gian) và nhân tốc độ này với hệ số vi phân K_D .

Thìa số vi phân được cho bởi: $D = K_D \frac{d}{dt} e(t)$

Trong đó: KD là hệ số vi phân

$e(\tau)$ là hàm sai lệch theo thời gian

Khâu vi phân làm chậm tốc độ thay đổi của đầu ra bộ điều khiển và đặc tính này là đang chú ý nhất để đạt tới điểm đặt của bộ điều khiển. Từ đó, điều khiển vi phân được sử dụng để làm giảm biên độ vọt lố được tạo ra bởi thành phần tích phân và tăng cường độ ổn định của bộ điều khiển hỗn hợp.



Hình 18. Đáp ứng của hệ thống khi tăng dần K_D

Khâu tỉ lệ, tích phân, vi phân được cộng lại với nhau để tính toán đầu ra của bộ điều khiển PID. Gọi u là tín hiệu điều khiển, ta có:

$$u = K_p \times e + K_I \int_0^t e(\tau) d\tau + K_D \frac{d}{dt} e(t)$$

Công thức tính toán bộ điều khiển PID ròng rạc: $\int_0^t e(t) dt = \sum_{k=0}^k \frac{e(kT) + e(kT - T)}{2}$

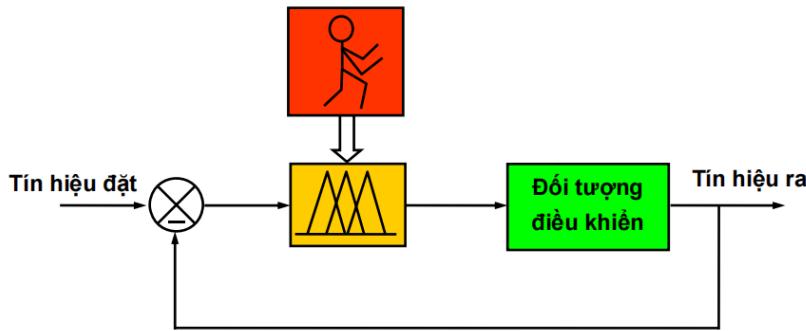
Sử dụng công thức lấy xấp xỉ với chu kỳ lấy mẫu T : $\frac{de(t)}{dt} = \frac{e(kT) - e(kT - T)}{T}$

Từ đó, ta được:

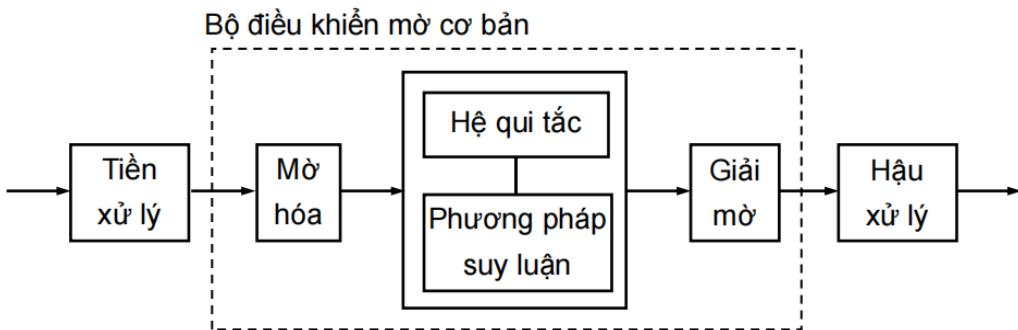
$$u(kT) = u(kT - T) + K_p \times [e(kT) - e(kT - T)] + K_I \times \frac{e(kT) - e(kT - T)}{2T} + K_D \times \frac{e(kT) - 2e(kT - T) + e(kT - 2T)}{T}$$

2.5. Bộ điều khiển Fuzzy

Bộ điều khiển Fuzzy (Fuzzy Logic Controller - FLC) là một phương pháp điều khiển dựa trên logic mờ. Thay vì sử dụng các mô hình toán học cứng nhắc, FLC áp dụng các quy tắc logic mờ để mô tả mối quan hệ giữa đầu vào và đầu ra, cho phép xử lý hiệu quả trong các hệ thống có nhiều yếu tố bất định hoặc phi tuyến.



Hình 19. Sơ đồ khối tổng quát bộ điều khiển Fuzzy



Hình 20. Sơ đồ khối tổng quát các bước bộ điều khiển mờ cơ bản

Bộ điều khiển mờ cơ bản sẽ gồm có các khâu được minh họa như trên hình:

Đầu tiên là khâu tiền xử lý. Khâu này xử lý tín hiệu ngõ vào của bộ điều khiển, có thể thực hiện các công việc bao gồm lượng tử hóa tín hiệu, chuẩn hóa giá trị, lọc nhiễu.

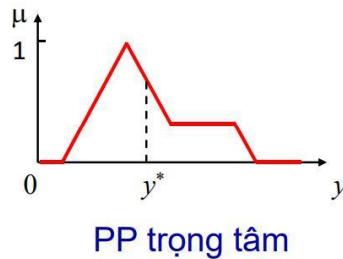
Tiếp đến khâu mờ hóa, từ các tín hiệu rõ ở ngõ vào của hệ mờ, ta sẽ tính các giá trị hàm liên thuộc của các giá trị ngôn ngữ được định nghĩa cho các biến vào. Từ đó ta suy ra độ đúng của các mệnh đề điều kiện trong hệ quy tắc mờ.

Hệ quy tắc mờ được đặt ra dựa trên sự suy luận hợp logic, kinh nghiệm điều khiển của con người. Thông qua hệ quy tắc, bộ điều khiển sử dụng phương pháp suy luận để giải ra độ đúng của mệnh đề kết luận ứng với mỗi quy tắc. Có hai phương pháp suy luận là phương pháp MAX-PROD và phương pháp MAX-MIN.

Sau khi có được kết quả suy luận mờ, ta thực hiện giải mờ để có được giá trị rõ cho biến ra. Tùy thuộc vào loại hệ mờ mà ta lựa chọn, ta sẽ có phương pháp giải mờ tương ứng. Có hai loại hệ mờ là hệ mờ Mamdani và hệ mờ Sugeno. Đối với hệ

mờ Mamdani, trong các bài toán điều khiển, ta thường dùng các phương pháp giải mờ: phương pháp giải mờ theo trọng tâm, phương pháp trung bình mờ.

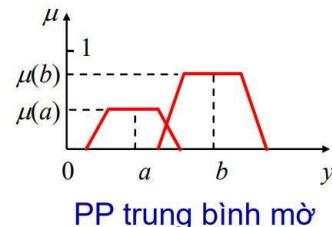
Công thức giải mờ cho phương pháp trọng tâm:



Hình 21. Phương pháp giải mờ theo trọng tâm

$$y^* = \frac{\int_Y y \mu(y) dy}{\int_Y \mu(y) dy} \approx \frac{\sum_k y_k \mu(y_k)}{\sum_k \mu(y_k)}$$

Công thức giải mờ cho phương pháp trung bình mờ:



Hình 22. Phương pháp giải trung bình mờ

$$y^* = \frac{a\mu(a) + b\mu(b)}{\mu(a) + \mu(b)}$$

Đối với hệ mờ Sugeno, ta có các phương pháp giải mờ trung bình có trọng số:

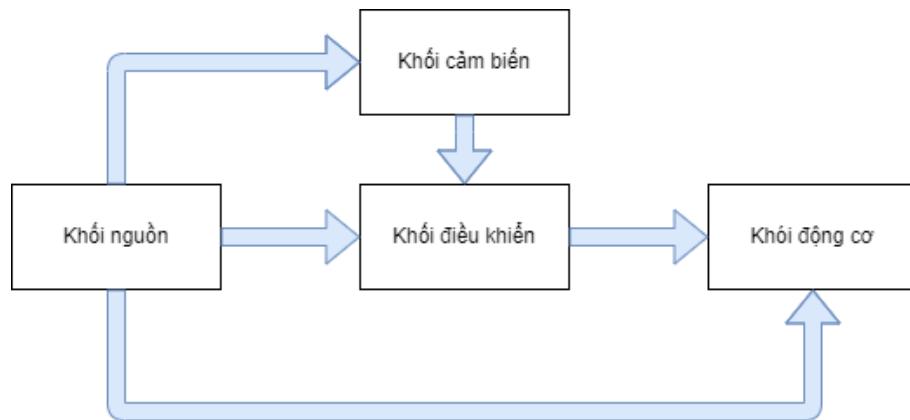
$$y^* = \frac{\sum_k \beta_k y_k}{\sum_k \beta_k}$$

Trong đó: β_k là kết quả suy luận của quy tắc thứ k trong hệ quy tắc mờ, y_k là hằng số ứng với giá trị ngôn ngữ của biến ra tại kết quả suy luận quy tắc thứ k trong hệ quy tắc mờ.

Chương 3: Thiết kế và thi công phần cứng

3.1. Sơ đồ khái niệm

3.1.1. Sơ đồ tổng quát mô hình



Hình 23. Sơ đồ khái niệm tổng quát phần cứng

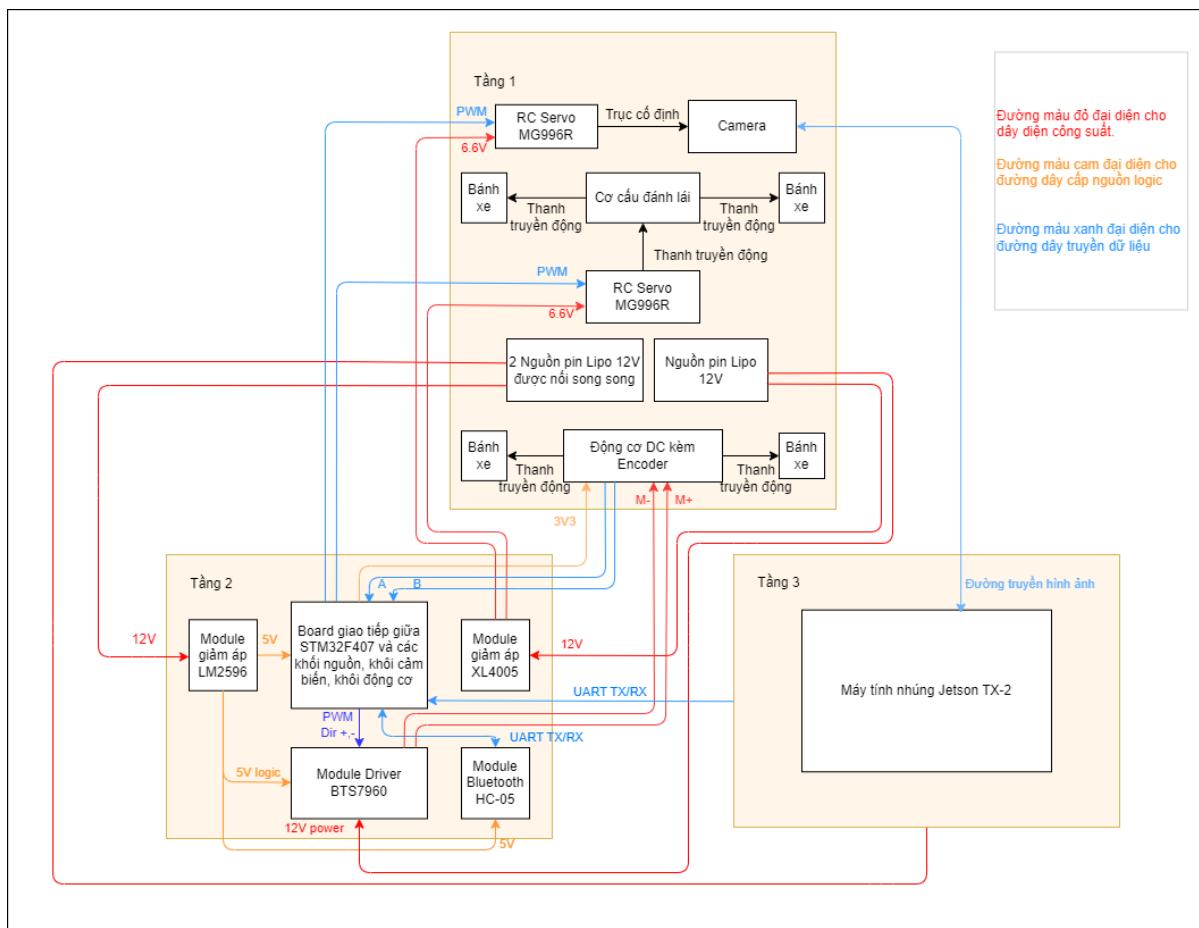
Khối Nguồn: Cung cấp năng lượng cho các thành phần của robot, bao gồm pin và mạch ổn áp.

Khối Cảm biến: Camera Gopro Hero 8 và Encoder.

Khối Điều khiển: Board Jetson TX-2 và kit phát triển STM32F407.

Khối động cơ: Module lái BTS7960 và động cơ DC Servo GA25-370 DC, RC Servo MG996.

3.1.2. Sơ đồ chi tiết mô hình



Hình 24. Sơ đồ vị trí và kết nối của các thành phần trên mô hình xe

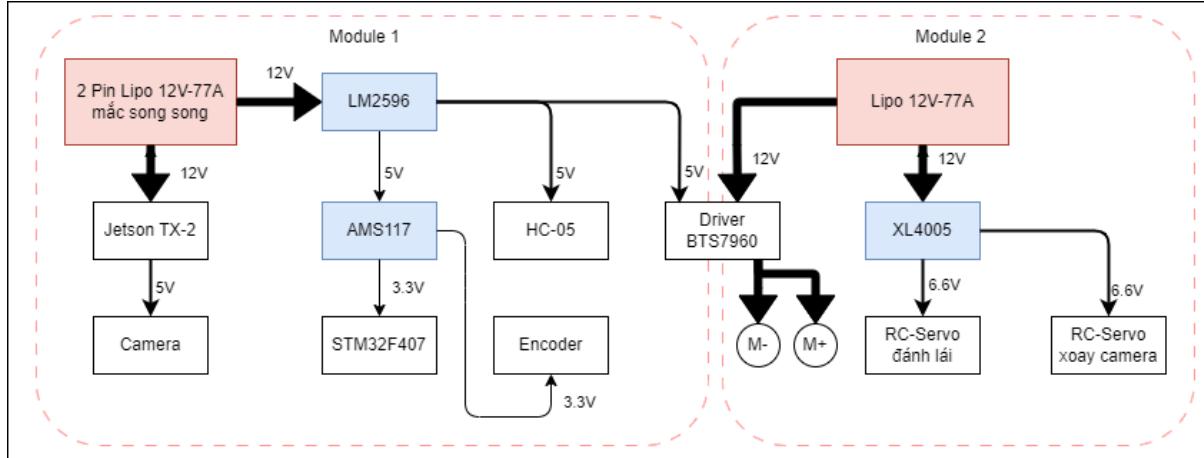
Mô hình xe tự hành được phân chia thành ba tầng như sơ đồ hình trên.

Tầng một bao gồm các viên pin Lipo 12V cung cấp nguồn cho toàn bộ mô hình. Cùng với đó là cơ cấu cơ khí đánh lái được nối với hai bánh trước nhờ các thanh truyền động và được điều khiển bởi một động cơ RC Servo, động cơ DC được nối với hai bánh sau bằng thanh truyền động để cung cấp tốc độ dài. Ngoài ra còn một động cơ RC Servo được gắn với Camera phía trên, nhằm thay đổi góc nhìn của Camera.

Tầng hai bao gồm module giảm áp LM2596, module giảm áp XL4005, module Bluetooth HC-05, module Driver BTS7960 và quan trọng nhất là board giao tiếp giữa STM32F407. Board giao tiếp này có các ngõ ra I/O để điều khiển các module Driver, động cơ RC Servo và các ngõ giao tiếp dữ liệu UART.

Tầng ba chỉ là nơi đặt máy tính Jetson TX-2, giao tiếp với các module khác thông qua cổng giao tiếp USB và giao tiếp UART.

3.1.3. Sơ đồ kết nối phần công suất



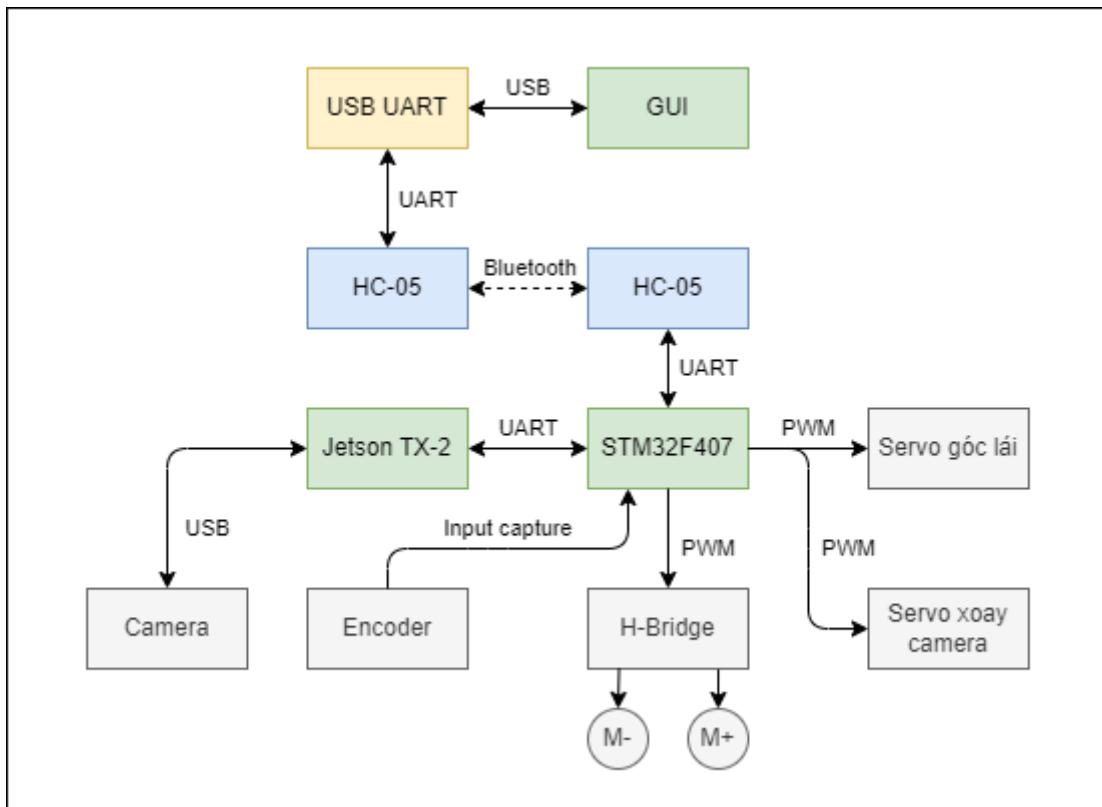
Hình 25. Sơ đồ kết nối phần công suất

Nguồn cấp gồm 3 pin Lipo 11.1VDC. Module 1 sử dụng 2 pin Lipo làm ngõ vào, module 2 sử dụng 1 pin Lipo làm ngõ vào.

Module 1 là module nguồn cung cấp cho Jetson TX-2, Development Kit STM32F407, mạch logic của Driver BTS7960. Với 2 pin Lipo được mắc song song với nhau, cung cấp điện áp 12V trực tiếp cho máy tính nhúng Jetson TX-2 và cho module giảm áp LM2596. Ngõ ra của module giảm áp LM2596 dùng để cấp nguồn cho board Dev Kit STM32F407, mạch logic của Driver BTS7960. Board Dev Kit STM32F407 được tích hợp sẵn module giảm áp AMS117, ngõ ra của module giảm áp này sẽ cấp nguồn cho vi điều khiển được tích hợp trên Board và cấp nguồn cho Encoder được gắn ở động cơ.

Module 2 là module nguồn cung cấp cho DC Motor và động cơ DC Servo. Với 1 pin Lipo cung cấp điện áp 12V trực tiếp cho động cơ DC thông qua mạch lái động cơ và cho module giảm áp XL4005 với dòng tối đa lên đến 5A. Ngõ ra của module giảm áp XL4005 dùng để cấp nguồn cho động cơ Servo điều khiển góc lái của xe và động cơ Servo điều khiển góc xoay Camera.

3.1.4. Sơ đồ kết nối phần giao tiếp và điều khiển



Hình 26. Sơ đồ kết nối phần dữ liệu và điều khiển

Sơ đồ trên mô tả chi tiết kiến trúc kết nối hệ thống giao tiếp và điều khiển của mô hình xe tự hành. Hệ thống bao gồm ba thành phần chính: giao diện người dùng GUI trên máy tính, bộ xử lý nhúng Jetson TX-2, và vi điều khiển STM32F407VGT6, cùng với các thiết bị ngoại vi như động cơ, encoder và các module truyền thông.

Giao diện GUI trên máy tính được kết nối với module Bluetooth HC-05 (Master) thông qua bộ chuyển đổi USB UART. GUI đóng vai trò giao tiếp giữa người dùng và hệ thống, cho phép người dùng gửi lệnh điều khiển, cấu hình thông số và theo dõi trạng thái vận hành của xe. Dữ liệu từ GUI được truyền tới module HC-05 Master, sau đó thiết lập kết nối Bluetooth không dây với module HC-05 Slave gắn trên mô hình xe.

Module HC-05 (Slave) trên xe kết nối với vi điều khiển STM32F407 qua giao tiếp UART. Tại đây, STM32F407 tiếp nhận các lệnh điều khiển từ GUI và phản hồi

trạng thái vận hành xe về giao diện người dùng, STM32F407 đồng thời đóng vai trò trung tâm điều phối các tác vụ điều khiển khác trong hệ thống.

Jetson TX-2 là máy tính nhúng chịu trách nhiệm chính trong việc xử lý hình ảnh từ camera GoPro Hero 8 gắn trên xe. Các khung hình video được phân tích trực tiếp trên Jetson TX-2. Sau khi xử lý hình ảnh, các giá trị sẽ được gửi về STM32F407 thông qua một đường UART riêng biệt.

STM32F407 sử dụng các tín hiệu nhận từ Jetson TX-2 để thực hiện điều khiển các phần tử chấp hành. Cụ thể, STM32F407 phát ra các tín hiệu PWM để điều khiển:

- Servo góc lái: Thực hiện thay đổi góc đánh lái của hai bánh trước, giúp xe bám theo lane hoặc rẽ tại các ngã tư.
- Servo xoay camera: Điều chỉnh góc nhìn của camera GoPro theo từng trạng thái vận hành, đảm bảo camera luôn quan sát được lane đường.

Bên cạnh đó, STM32F407 cũng điều khiển động cơ DC thông qua module driver H-Bridge BTS7960. Tín hiệu PWM điều chỉnh điện áp đầu ra của H-Bridge, từ đó điều khiển tốc độ và hướng quay của động cơ bánh sau. Để thực hiện điều khiển tốc độ chính xác, STM32F407 sử dụng chức năng Input Capture để thu thập xung từ Encoder gắn trên động cơ. Các xung này phản ánh tốc độ quay thực tế của động cơ, và được đưa vào bộ điều khiển PID để điều chỉnh tốc độ động cơ theo yêu cầu vận hành.

3.2. Các cơ cấu cơ khí của mô hình

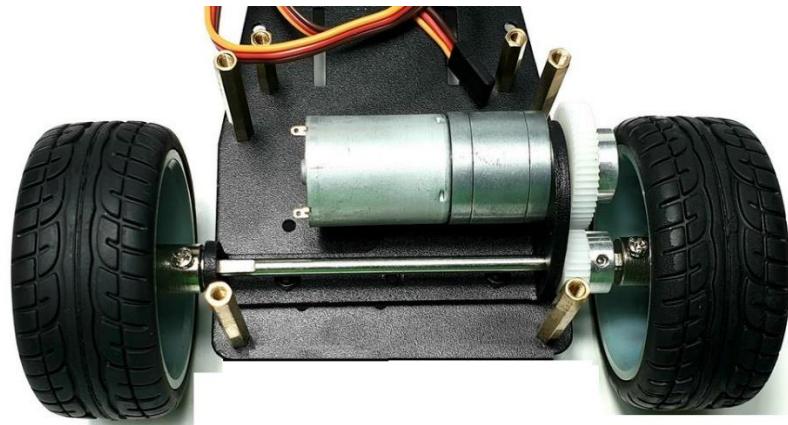
3.2.1. Cơ cấu đánh lái



Hình 27. Cơ cấu đánh lái

Cơ cấu đánh lái của mô hình được điều khiển bằng một động cơ RC Servo. Với độ lớn góc quay tối đa của Servo về mỗi phía là 45 độ so với vị trí ban đầu.

3.2.2. Cơ cấu truyền động tạo tốc độ dài cho mô hình



Hình 28. Cơ cấu truyền động tạo tốc độ dài

Cơ cấu truyền động tạo tốc độ dài được điều khiển bởi một động cơ DC. Động cơ DC kết nối với thanh dẫn động thông qua hai bánh răng có tỉ số giữa bánh răng lớn so với bánh răng nhỏ là 54/30.

3.2.3. Cơ cấu xoay camera



Hình 29. Cơ cấu xoay camera

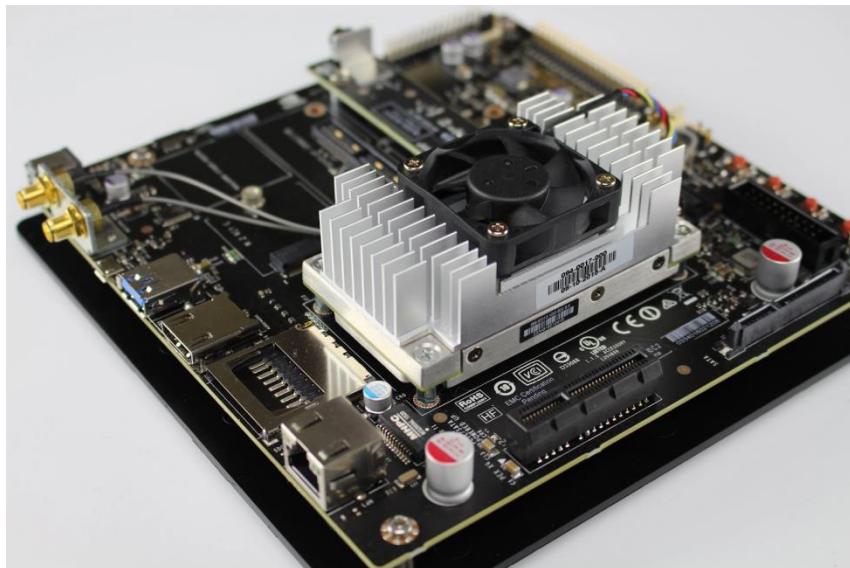
Camera được cố định vào một trục. Trục này được gắn với đĩa xoay của động cơ RC Servo, nhờ vậy nên khi động cơ RC Servo xoay, Camera có thể xoay hướng thay đổi góc nhìn được. Độ lớn góc xoay tối đa được đặt là 45 độ.

3.3. Tổng quan các thiết bị

3.3.1. Các module, thiết bị được sử dụng

3.3.1.1. Jetson TX2 Developer Kit

Jetson TX2 Developer Kit là một bộ công cụ phát triển nhúng AI mạnh mẽ được NVIDIA thiết kế, nhằm hỗ trợ các nhà phát triển xây dựng, kiểm tra và triển khai các ứng dụng AI/ML trong môi trường nhúng. Đây là một nền tảng nổi bật nhờ khả năng tính toán hiệu suất cao, tiêu thụ năng lượng thấp và kích thước nhỏ gọn.



Hình 30. Jetson TX-2 Developer Kit

Module Jetson TX2:

- CPU: Bộ xử lý 6 nhân, gồm 4 nhân ARM Cortex-A57 và 2 nhân
- GPU: NVIDIA Pascal với 256 lõi CUDA, hỗ trợ xử lý đồ họa và tính toán AI.
- RAM: 8GB LPDDR4, băng thông 59.7 GB/s.
- Bộ nhớ: 32GB eMMC 5.1.
- Hiệu suất: Tối đa 1.3 TFLOPS (FP16).
- Tiêu thụ điện năng: Chỉ từ 7.5W đến 15W, tùy chế độ.

3.3.1.2. STM32F407 Discovery

STM32F407 Discovery là một bô phát triển (development board) của STMicroelectronics, được thiết kế để hỗ trợ các nhà phát triển làm quen và xây dựng các ứng dụng với dòng vi điều khiển STM32F407. Đây là một sản phẩm mạnh mẽ, được tích hợp nhiều tính năng hữu ích để phát triển và thử nghiệm trong nhiều lĩnh vực, đặc biệt là nhúng và điều khiển.



Hình 31. Kit STM32F407 Discovery

Thông số kỹ thuật chính của STM32F407 Discovery:

- CPU: ARM Cortex-M4 32-bit, tần số lên tới 168 MHz.
- RAM: 192 KB SRAM.
- ROM: 1 MB Flash.
- Tích hợp FPU (Floating Point Unit): Tăng cường hiệu suất cho các phép toán số thực.
- Hỗ trợ DSP (Digital Signal Processing): Phù hợp cho xử lý tín hiệu.

3.3.1.3. Camera hành trình Gopro Hero 8

Camera Gopro Hero 8 được sử dụng trong mô hình xe tự hành để ghi nhận hình ảnh phục vụ các tác vụ thị giác máy. Camera có khả năng quay video với chất lượng cao, ổn định, và hoạt động tốt trong nhiều điều kiện môi trường khác nhau.



Hình 32. Camera hành trình Gopro Hero 8

Thông số kỹ thuật chính của Camera Gopro Hero 8:

- Cảm biến hình ảnh có độ phân giải 12MP.
- Hỗ trợ quay video ở nhiều độ phân giải và tốc độ khung hình: 4K ở 60fps, 2.7K ở 120fps, và 1080p ở 240fps.
- Tốc độ truyền dữ liệu video tối đa lên tới 100 Mbps.
- Hỗ trợ chuẩn mã hóa video H.264 và H.265.
- Công nghệ chống rung HyperSmooth 2.0 giúp ổn định hình ảnh khi xe di chuyển.

3.3.1.4. Pin sạc Lipo Lion Power 11.1VDC 2200mAh 3S 35C XT60

Là loại pin Lithium Polymer được sử dụng phổ biến trong các thiết bị công suất cao như máy bay mô hình, xe tự hành, và xe đua RC. Pin gồm 3 cell Lipo mắc nối tiếp (3S), với dung lượng 2200mAh, điện áp trung bình từ 11.1V đến 12.6V, và dòng xả tối đa lên đến 77A (35C).



Hình 33. Pin Lipo Power 11.1VDC

- Điện áp : 11.1V, khi sạc đầy có thể đạt đến 12.6V

- Dung lượng : 2200 mAh
- Dòng xả tối đa: 77A

3.3.1.5. Mạch giảm áp DC-DC Buck LM2596

Là mạch chuyển đổi DC-DC hiệu suất cao, phù hợp cho các ứng dụng cần giảm áp và cung cấp dòng ổn định lên đến 3A.



Hình 34. Module giảm áp LM2596

- Điện áp đầu vào: 4~30VDC.
- Điện áp đầu ra: Có thể điều chỉnh từ 1.5~29VDC.
- Dòng ngõ ra tối đa: 3A.

3.3.1.6. Mạch giảm áp DC-DC Buck XL4005 5A

Mạch giảm áp DC-DC Buck XL4005 5A sử dụng phương pháp giảm áp xung để giảm áp DC với dòng đầu ra lên đến 5A, mạch có tần số xung lên đến 300Khz cho độ ổn định cao, nhiễu thấp



Hình 35. Module giảm áp XL4005

- Điện áp đầu vào: 5~32VDC
- Điện áp đầu ra: điều chỉnh từ 1.25V đến 30V.

- Dòng đỉnh tối đa là 5A (dòng trung bình 3.5A).

3.3.1.7. Động Cơ DC Servo GA25-370

Động Cơ DC Servo GA25-370 DC Geared Motor được tích hợp thêm Encoder hai kênh AB giúp đọc và điều khiển chính xác vị trí, chiều quay của động cơ trong các ứng dụng cần độ chính xác cao: điều khiển PID, Robot tự hành,....

Động Cơ DC Servo GA25-370 DC Geared Motor có cấu tạo bằng kim loại cho độ bền và độ ổn định cao, được sử dụng trong các mô hình robot, xe, thuyền,..., hộp giảm tốc của động cơ có nhiều tỉ số truyền giúp bạn dễ dàng lựa chọn giữa lực kéo và tốc độ (lực kéo càng lớn thì tốc độ càng chậm và ngược lại), động cơ sử dụng nguyên liệu chất lượng cao.



Hình 36. Động cơ DC Servo GA25-370

Thông số kỹ thuật:

- Điện áp sử dụng: 12VDC
- Đường kính: 25mm
- Tỉ số truyền 171:1 (động cơ quay 171 vòng trục chính hộp giảm tốc quay 1 vòng).
- Dòng không tải: 60mA
- Dòng chịu đựng tối đa khi có tải: 1.3A
- Tốc độ không tải: 35RPM (35 vòng 1 phút)
- Tốc độ chịu đựng tối đa khi có tải: 27RPM (27 vòng 1 phút)
- Lực kéo Moment định mức: 4KG.CM
- Lực leo Moment tối đa: 9KG.CM

- Chiều dài hộp số L: 25mm

3.3.1.8. Động cơ RC Servo MG996.

Động cơ RC Servo MG996 là loại thường được sử dụng nhiều nhất trong các thiết kế Robot hoặc dẫn hướng xe. Động cơ RC Servo MG996 có lực kéo mạnh, các khớp và bánh răng được làm hoàn toàn bằng kim loại nên có độ bền cao, động cơ được tích hợp sẵn driver điều khiển động cơ bên trong theo cơ chế phát xung - quay góc nên rất dễ sử dụng.



Hình 37. Servo MG996R

Thông số kỹ thuật:

- Chủng loại: Analog RC Servo.
- Điện áp hoạt động: 4.8~6.6VDC
- Góc quay của trục: 0-180°
- Lực kéo:
 - ❖ 9.4kg/cm (4.8VDC).
 - ❖ 11kg/cm (6.0VDC).
- Tốc độ quay:
 - ❖ 0.19sec/60degree (4.8VDC).
 - ❖ 0.15sec/60degree (6.0VDC).

3.3.1.9. Module lái BTS7960

Mạch điều khiển động cơ DC BTS7960 43A High-power Motor Driver có khả năng điều khiển 1 động cơ DC sử dụng IC điều khiển động cơ BTS7960 với công

suất tối đa lên đến 43A theo thông số nhà sản xuất, ngoài ra mạch còn được thiết kế thêm IC buffer chuyển mức tín hiệu 74HC244 giúp bạn kết nối an toàn với vi điều khiển khi sử dụng.

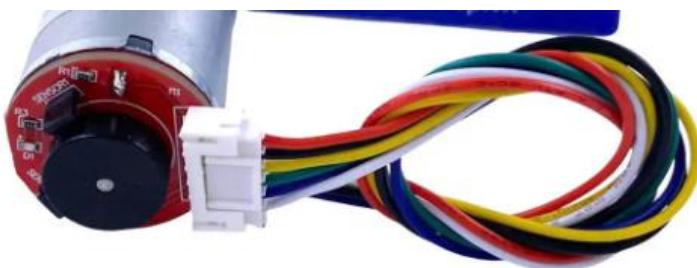


Hình 38. Module lái BTS7960

Thông số kỹ thuật:

- Nguồn cấp: 6 ~ 27VDC
- Dòng điện tải mạch: 43A (Tải trớ) hoặc 15A (Tải cảm)
- Tín hiệu logic điều khiển: 3.3 ~ 5VDC.
- Tần số điều khiển tối đa: 25KHz.

3.3.1.10. Encoder



Hình 39. Encoder

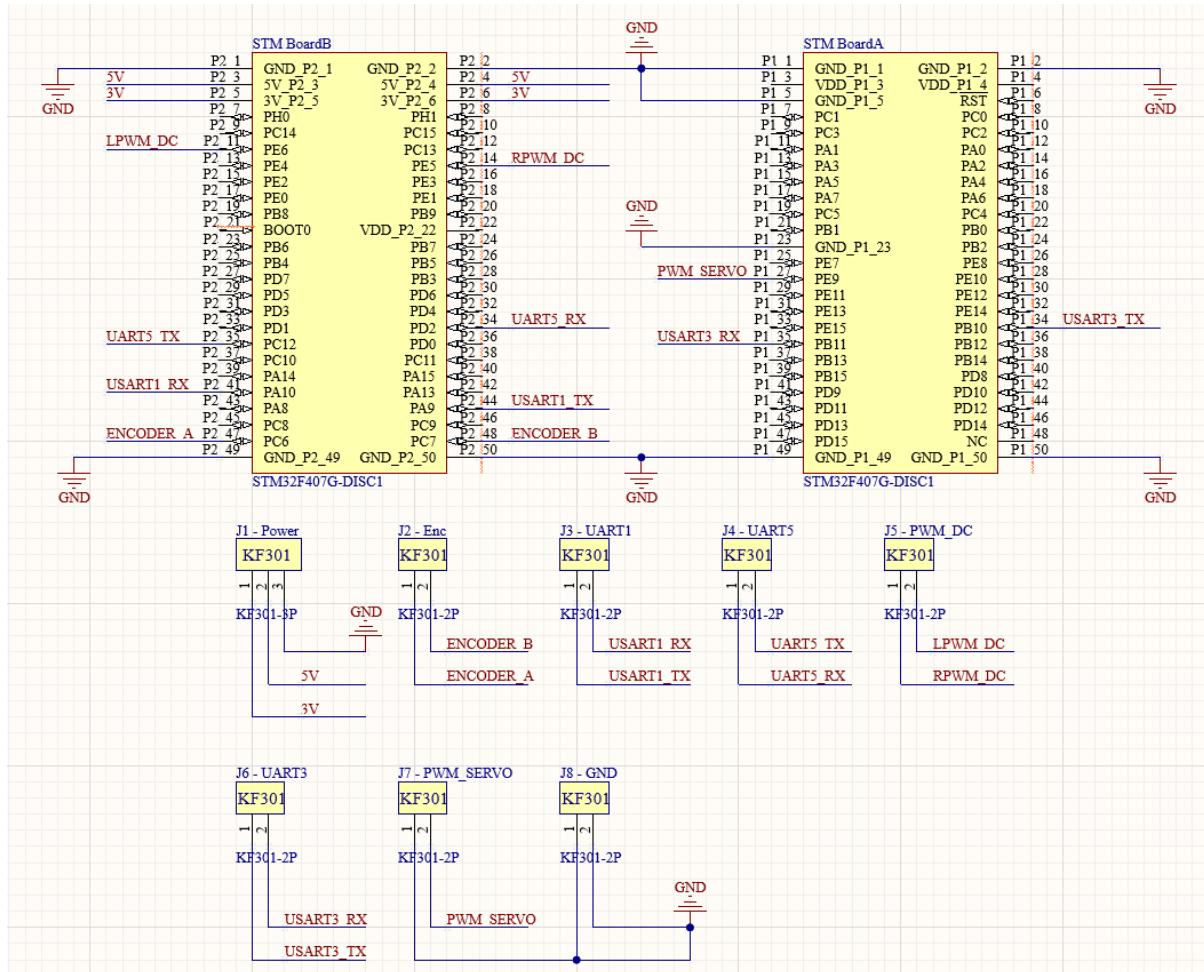
Cảm biến từ trường Hall, có 2 kênh AB lệch nhau giúp xác định chiều quay và vận tốc của động cơ, đĩa Encoder trả ra 11 xung/1 kênh/ 1 vòng (nếu đo tín hiệu đồng thời của cả hai kênh sẽ thu được tổng 22 xung / 1 vòng quay của Encoder).

Số xung của mỗi kênh trên 1 vòng quay của trục chính động cơ = Tỉ số truyền x số xung của Encoder, ví dụ tỉ số 150:1 thì số xung Encoder trả ra cho 1 vòng quay của trục chính động cơ sẽ là $11 \times 150 = 1650$ xung / 1 kênh.

Điện áp cấp cho Encoder hoạt động: 3.3~5VDC

3.3.2. Board giao tiếp giữa STM32F407 và các khối khác

3.3.2.1. Sơ đồ nguyên lý



Hình 40. Sơ đồ nguyên lý board giao tiếp

Hình trên thể hiện sơ đồ nguyên lý kết nối của vi điều khiển STM32F407 với các ngoại vi điều khiển và cảm biến trong hệ thống xe tự hành.

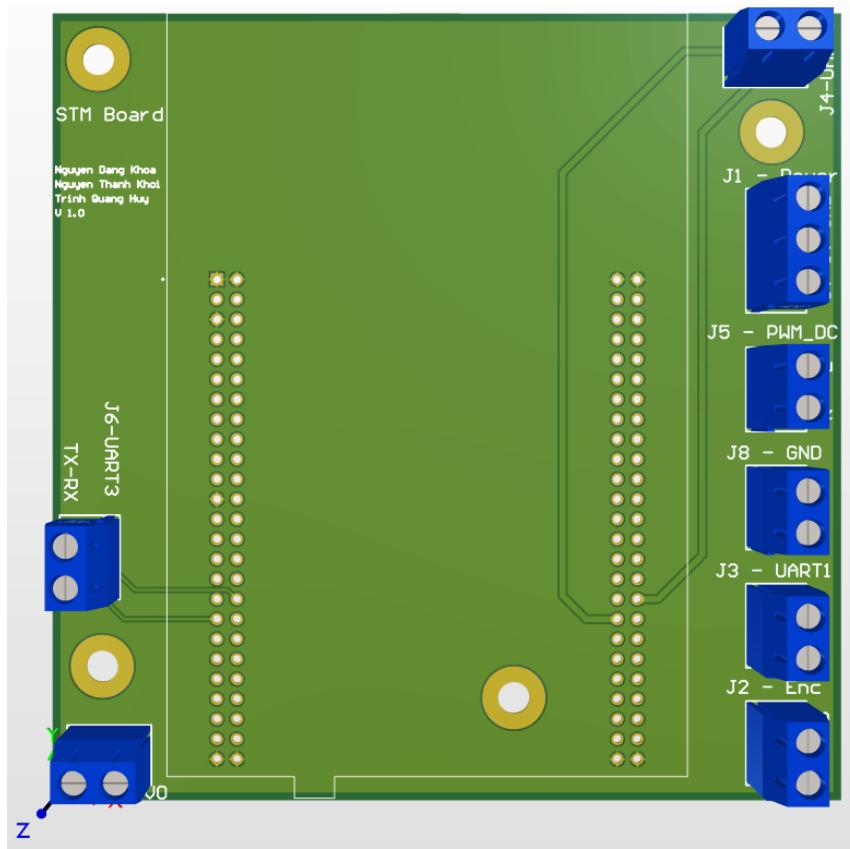
Vi điều khiển STM32F407 sử dụng các chân PWM để xuất tín hiệu điều khiển tốc độ động cơ DC (PWM_DC) và điều khiển góc lái cho Servo đánh lái (PWM_SERVO). Tín hiệu PWM được cấp ra qua các cổng kết nối KF301-2P. Ngoài ra, tín hiệu điều khiển góc lái cho Servo Camera được kết nối trực tiếp với chân PE14.

Giao tiếp UART được sử dụng để trao đổi dữ liệu với các thiết bị khác trong hệ thống. Các chân USART1_RX và USART1_TX dùng để kết nối truyền nhận dữ liệu chính, trong khi các chân UART5_RX và UART5_TX dành cho các kênh truyền thông bổ sung.

Encoder được kết nối đến vi điều khiển thông qua các chân ENCODER_A và ENCODER_B, cho phép STM32F407 đo lường tốc độ và vị trí của động cơ bằng tín hiệu xung.

Nguồn cấp cho hệ thống bao gồm đường 5V và 3V được phân phối tới các cổng giao tiếp và vi điều khiển. Các chân GND đảm bảo đất chung cho toàn bộ mạch.

3.3.2.2. Mạch in



Hình 41. Sơ đồ mạch in

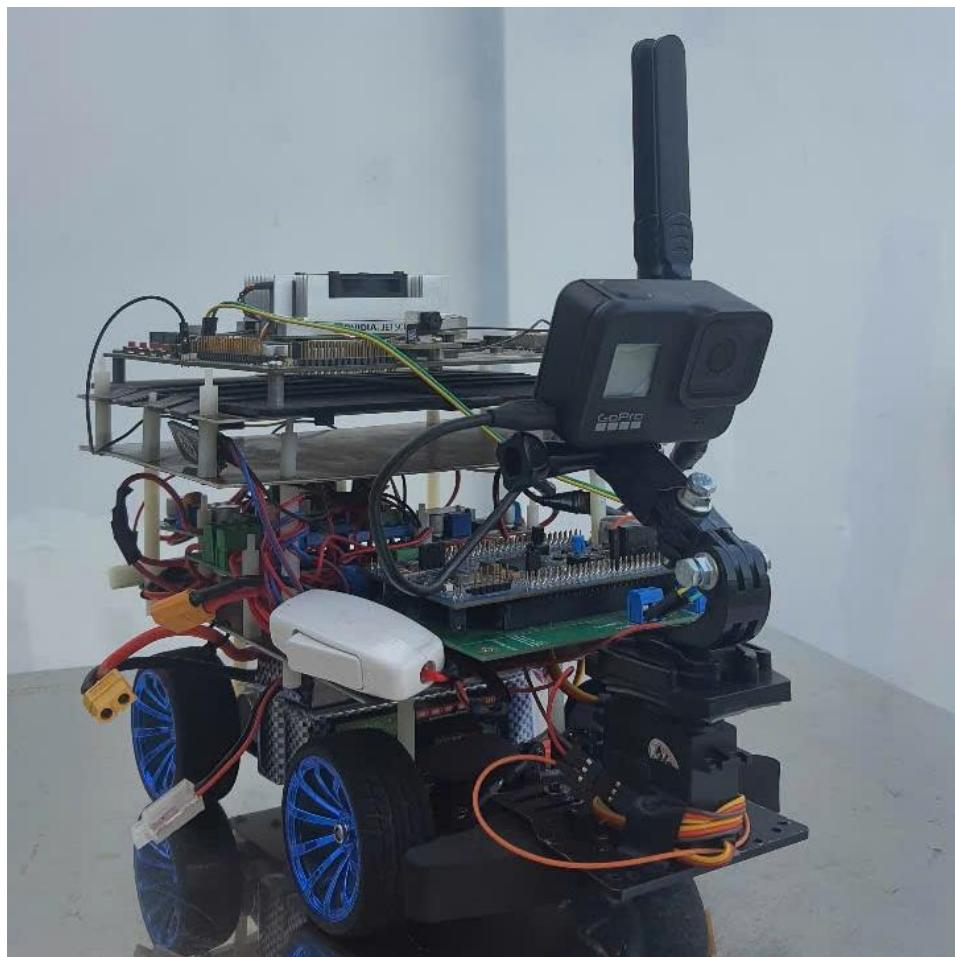
Hình trên thể hiện bố trí PCB của mạch in giao tiếp vi điều khiển STM32F407 với các ngoại vi.

Mạch in được thiết kế dạng shield, với hai hàng chân cắm tương thích với board STM32F407 Discovery.

Các cổng kết nối ngoại vi như Power, Encoder, UART, PWM_DC và PWM_SERVO được đưa ra qua các đầu nối KF301-2P bố trí dọc hai cạnh PCB, đảm bảo việc đấu nối và tháo lắp dễ dàng.

Đường mạch được thiết kế gọn gàng, phân luồng rõ ràng giữa các tín hiệu nguồn, tín hiệu điều khiển PWM và giao tiếp UART.

3.4. Sản phẩm hoàn thiện



Hình 42. Sản phẩm phần cứng mô hình xe tự hành hoàn thiện

Hình trên là sản phẩm mô hình xe tự hành hoàn thiện, tích hợp đầy đủ các thành phần gồm camera Gopro Hero 8, máy tính nhúng Jetson TX-2, vi điều khiển STM32F407, hệ thống động cơ, servo và các module giao tiếp.

Chương 4: Thiết kế phần mềm

4.1. Thiết kế giao tiếp giữa các hệ thống

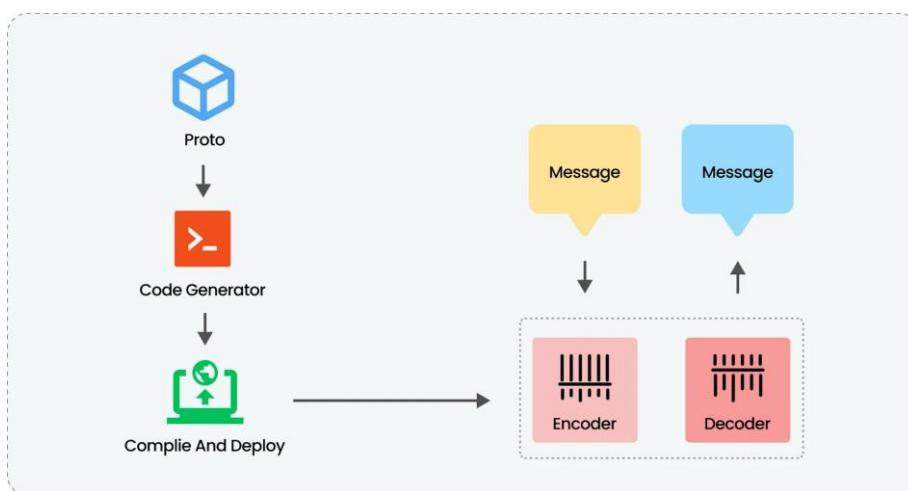
4.1.1. Chuẩn giao tiếp chung

Để giao tiếp dữ liệu giữa các thiết bị sử dụng các ngôn ngữ lập trình khác nhau (Python cho GUI và Jetson TX-2, C cho vi điều khiển STM32F407), hệ thống sử dụng Protocol Buffers (Protobuf) làm giao thức trung gian để mã hóa và truyền tải dữ liệu.

Protobuf là một giao thức tuân tự hóa dữ liệu dạng nhị phân do Google phát triển, mã nguồn mở và được hỗ trợ rộng rãi trên nhiều nền tảng như Python, Java, C++. Protobuf cho phép định nghĩa dữ liệu theo cách thống nhất và tự động sinh mã nguồn tương ứng cho từng ngôn ngữ lập trình từ một tệp duy nhất (.proto).

Tuy nhiên, phiên bản chuẩn của Protobuf có kích thước bộ thư viện tương đối lớn, gây khó khăn khi tích hợp vào các vi điều khiển có tài nguyên bộ nhớ hạn chế như STM32F407. Để khắc phục, hệ thống sử dụng Nanopb – một phiên bản tối ưu hóa của Protobuf dành riêng cho các nền tảng nhúng, với kích thước nhỏ gọn và dễ dàng tích hợp vào các chương trình nhúng sử dụng ngôn ngữ C.

Quá trình hoạt động của Protocol Buffer trong hệ thống diễn ra như sau:



Hình 43. Quá trình hoạt động của giao thức Protocol Buffer trong hệ thống

- Định nghĩa dữ liệu: Các loại dữ liệu và gói tin sử dụng trong giao tiếp giữa các thiết bị được định nghĩa tập trung trong một tệp tin định dạng .proto.
- Sinh mã nguồn: Từ file .proto, sử dụng trình biên dịch Protobuf để sinh ra các tệp mã nguồn tương ứng cho từng nền tảng: file .pb.h và .pb.c cho STM32 (Nanopb), file .py cho Python.
- Mã hóa và gửi dữ liệu (Encoding): Khi một thiết bị (vi điều khiển hoặc GUI hoặc Jetson TX-2) cần gửi dữ liệu, thiết bị sẽ tạo một đối tượng tin nhắn theo khuôn mẫu đã định nghĩa, sau đó mã hóa đối tượng này thành một chuỗi byte và truyền đi qua các kênh giao tiếp như UART, Bluetooth.
- Giải mã và xử lý dữ liệu (Decoding): Khi nhận được chuỗi byte, thiết bị đích sẽ giải mã theo cấu trúc Protobuf đã định nghĩa, khôi phục lại dữ liệu gốc để xử lý.

Nhờ việc sử dụng Protocol Buffer, hệ thống đảm bảo tính tương thích cao, dữ liệu truyền đi nhẹ, chính xác, dễ dàng mở rộng khi bổ sung các loại dữ liệu hoặc tin nhắn mới mà không ảnh hưởng đến các thiết bị đã triển khai.

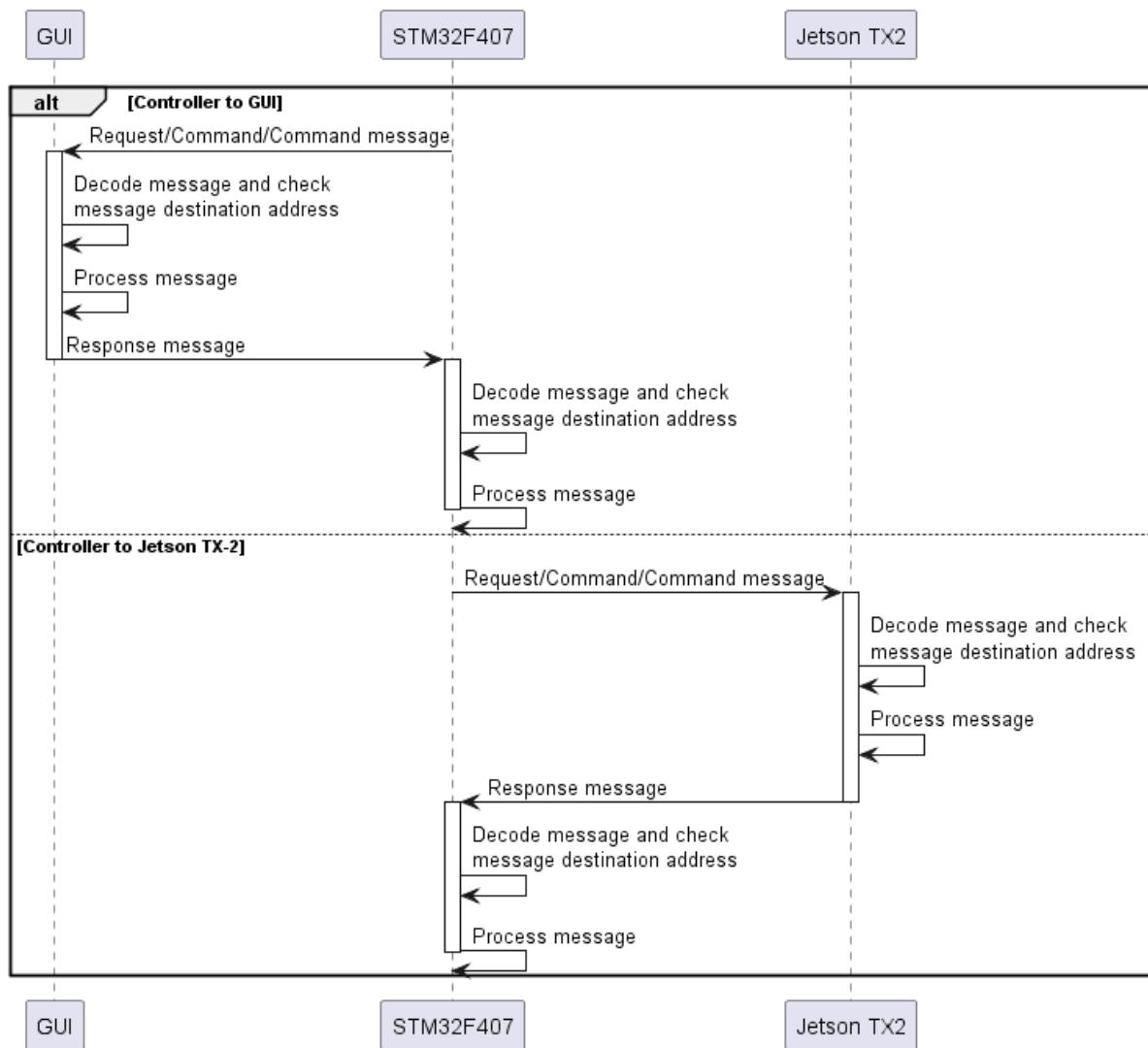
4.1.2. Kiến trúc luồng giao tiếp tổng quát

Hệ thống giao tiếp giữa ba thành phần chính gồm GUI, vi điều khiển STM32F407 và máy tính nhúng Jetson TX-2 được tổ chức theo mô hình trung gian thông qua vi điều khiển.

Vi điều khiển STM32F407 (Controller) được kết nối trực tiếp với GUI thông qua kết nối Bluetooth và kết nối trực tiếp với Jetson TX-2 thông qua UART. GUI và Jetson TX-2 không kết nối trực tiếp với nhau mà gửi và nhận dữ liệu thông qua STM32F407. Vi điều khiển có nhiệm vụ giải mã gói tin nhận được, kiểm tra địa chỉ đích của gói tin, và quyết định hành động:

- Nếu gói tin gửi tới chính vi điều khiển, STM32F407 sẽ xử lý gói tin tại chỗ.
- Nếu gói tin gửi tới thiết bị khác (GUI hoặc Jetson TX-2), STM32F407 sẽ chuyển tiếp gói tin mà không thay đổi nội dung.

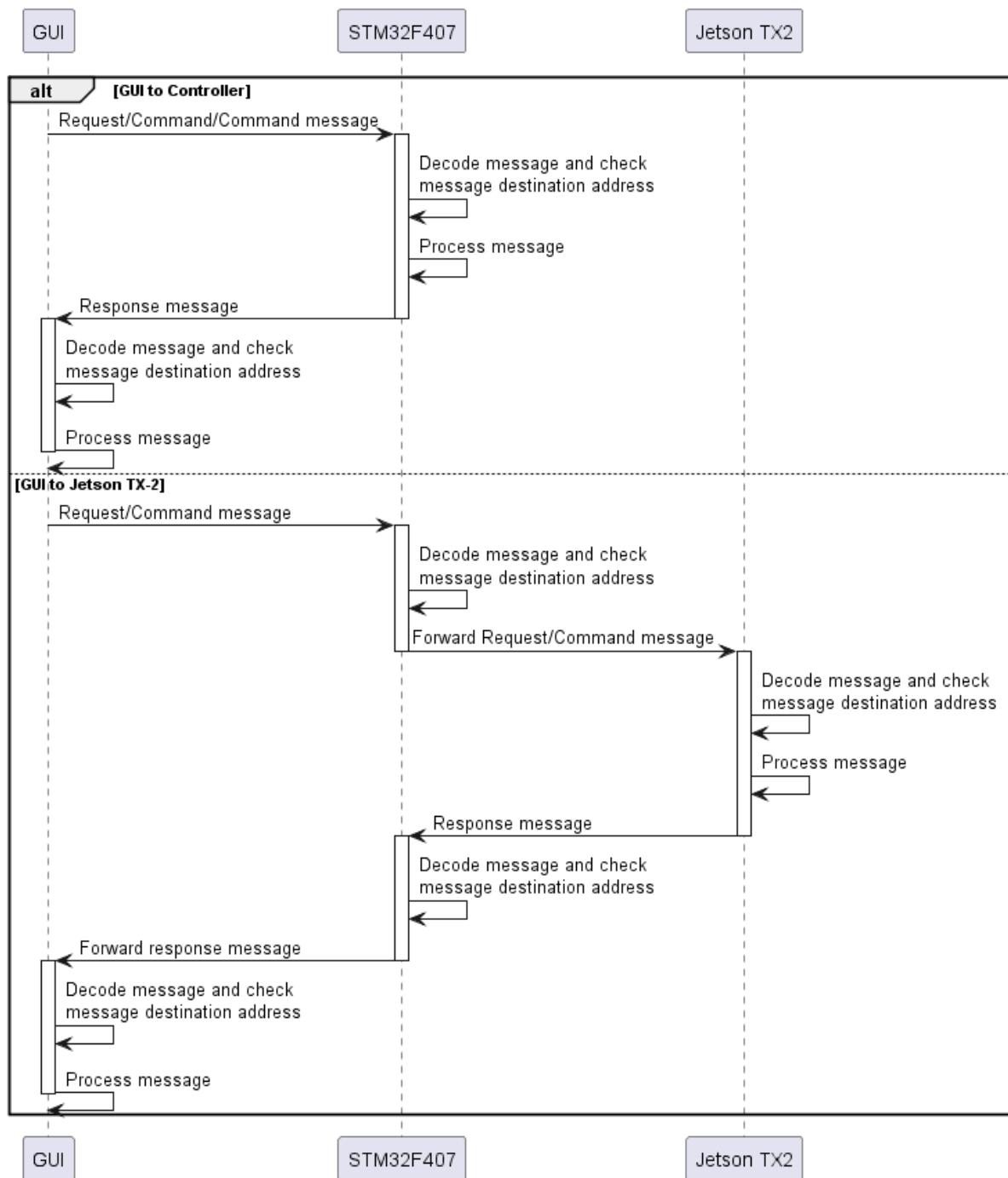
4.1.1. Giao tiếp từ Controller tới GUI và Jetson TX-2



Hình 44. Sơ đồ trình tự giao tiếp – STM32F407 gửi tin nhắn

STM32F407 gửi các gói tin yêu cầu (Request/Command) hoặc phản hồi (Response) đến GUI và Jetson TX-2 tùy theo chức năng và địa chỉ đích của gói tin.

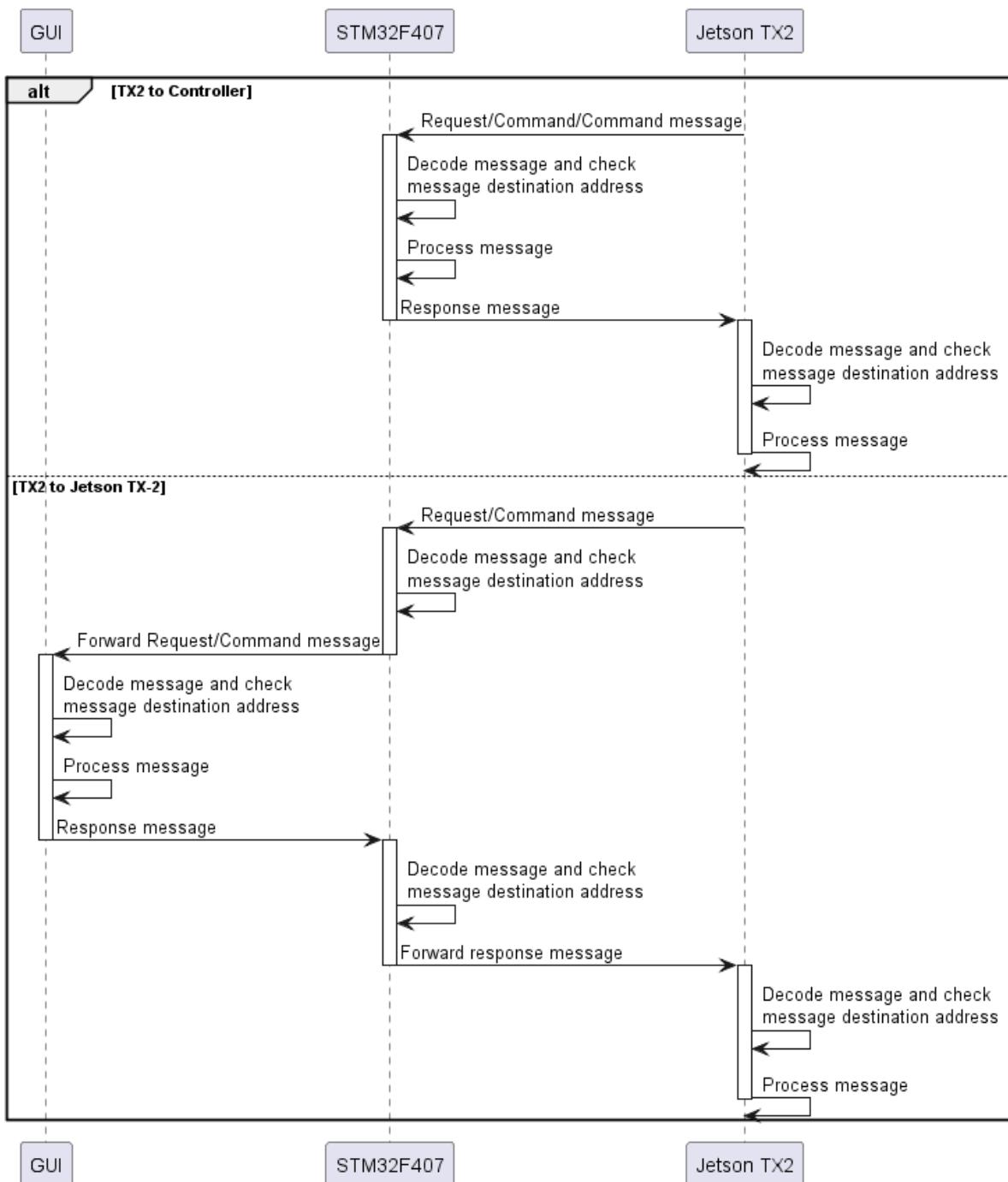
4.1.2. Giao tiếp từ GUI tới Controller và Jetson TX-2



Hình 45. Sơ đồ trình tự giao tiếp – GUI gửi tin nhắn

GUI gửi các yêu cầu tới STM32F407. Nếu cần gửi dữ liệu cho Jetson TX-2, STM32F407 sẽ thực hiện chuyển tiếp gói tin.

4.1.3. Giao tiếp từ Jetson TX-2 tới Controller và GUI



Hình 46. Sơ đồ trình tự giao tiếp -Jetson TX-2 gửi tin nhắn

Jetson TX-2 có thể gửi dữ liệu trực tiếp đến STM32F407 để yêu cầu thực thi, hoặc gửi dữ liệu tới GUI thông qua STM32F407 làm trung gian.

4.1.3. Định dạng gói tin truyền thông

Để đảm bảo việc phân định gói tin, xác thực dữ liệu và đồng bộ hóa trong quá trình giao tiếp giữa các thiết bị, mỗi gói tin truyền đi được đóng gói theo định dạng frame riêng, bao gồm phần header quản lý giao tiếp và phần payload dữ liệu được mã hóa bằng Protocol Buffer.

Cấu trúc tổng thể của một gói tin truyền thông như sau:

Thành phần	Kích thước	Mô tả
Start of Frame (SOF)	4 bytes	Giá trị cố định 0xAA55CDEF, đánh dấu bắt đầu gói tin
Length	4 bytes	Độ dài của trường Data tính bằng byte
Data	Không cố định	Nội dung gói tin đã được encode bằng Protocol Buffer
Checksum	1 byte	Tổng cộng tất cả các byte của Start of Frame, Length và Data

Bảng 1. Cấu trúc gói tin truyền thông

Trường Data sau khi giải mã từ Protocol Buffer sẽ bao gồm:

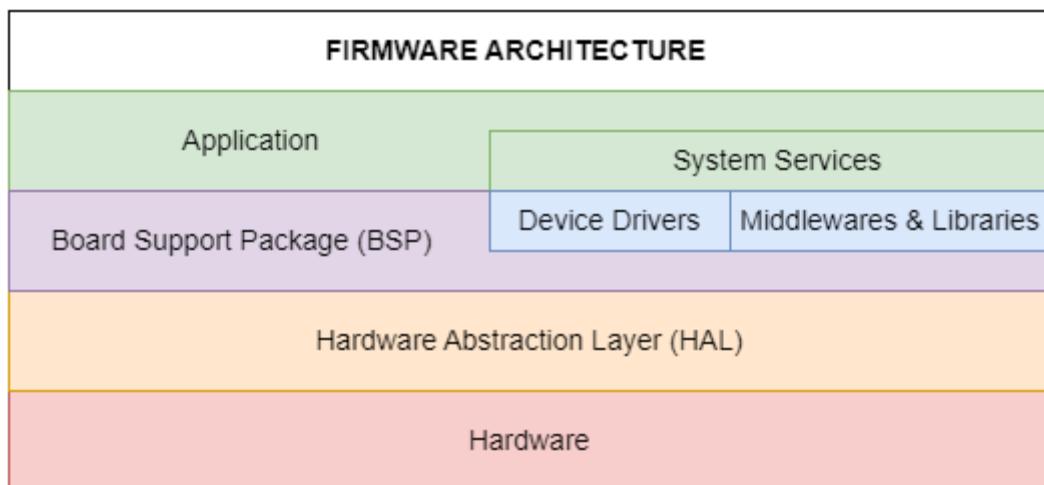
- Header:
 - Source Address: Địa chỉ thiết bị gửi gói tin.
 - Destination Address: Địa chỉ thiết bị nhận gói tin.
 - Epoch Time: Thời gian gói tin được đóng gói và gửi đi, tính theo đơn vị millisecond (ms) tại múi giờ GMT+7.
- Type Identifier:
 - Which Params: Một giá trị số nguyên (enum) xác định loại nội dung tin nhắn đang chứa bên trong gói dữ liệu, ví dụ: lệnh điều khiển, phản hồi trạng thái, cập nhật thông số, thông báo nhận dạng giao thông, v.v.

- Payload nội dung (Dữ liệu cụ thể tùy thuộc vào giá trị của which_params):
 - Lệnh yêu cầu (Request Command).
 - Phản hồi dữ liệu (Response Message).
 - Cập nhật trạng thái hệ thống (Info Response).
 - Thay đổi thiết lập bộ điều khiển (Setting/Configuration).
 - Dữ liệu giao thông và thị giác máy (Traffic Signs, Vision Coordinates).

4.2. Thiết kế Firmware trên vi điều khiển STM32F407

4.2.1. Kiến trúc Firmware

4.2.1.1. Tổng quan kiến trúc



Hình 47. Kiến trúc Firmware tổng thể

Firmware được thiết kế theo kiến trúc nhằm mục đích phân rõ nhiệm vụ cho từng lớp khác nhau, mỗi lớp sẽ quản lý một hoặc nhiều chức năng nhất định của hệ thống. Việc thiết kế Firmware theo một lối kiến trúc cụ thể hỗ trợ cho việc kiểm soát và phát triển dự án lâu dài. Với nguyên tắc thiết kế từ thiết kế tổng thể, sau đó chia bản thiết kế ra thành từng phần chi tiết hơn, quá trình thiết kế sẽ đi từ đỉnh của kiến trúc xuống. Tức là, xác định được yêu cầu hệ thống, sau đó lập ra các function hay các APIs của những lớp thấp hơn để đạt được yêu cầu đó.

Vì thế, trước hết phải định nghĩa được file C Header .h, chính là các struct, enum và các function (hay API) gán với một nhiệm vụ nhất định để có thể đạt được

chức năng ở lớp Application. Từ đó mới viết file C Source .c để thực hiện đúng nhiệm vụ đó.

4.2.1.2. Vai trò các lớp trong kiến trúc

Nhiệm vụ của từng lớp trong kiến trúc tổng thể như sau:

Hardware Abstraction Layer (HAL): đây là lớp thấp nhất của Firmware, giúp cho các lớp phía trên giao tiếp với phần cứng mà không cần quan tâm đến chi tiết của phần cứng, ví dụ như định dạng thanh ghi, địa chỉ thanh ghi, ...

Device Driver: Lớp này đóng vai trò quan trọng giao tiếp giữa thiết bị với hệ thống, bao gồm chức năng khởi tạo, quản lý dữ liệu nhập xuất, điều khiển,...

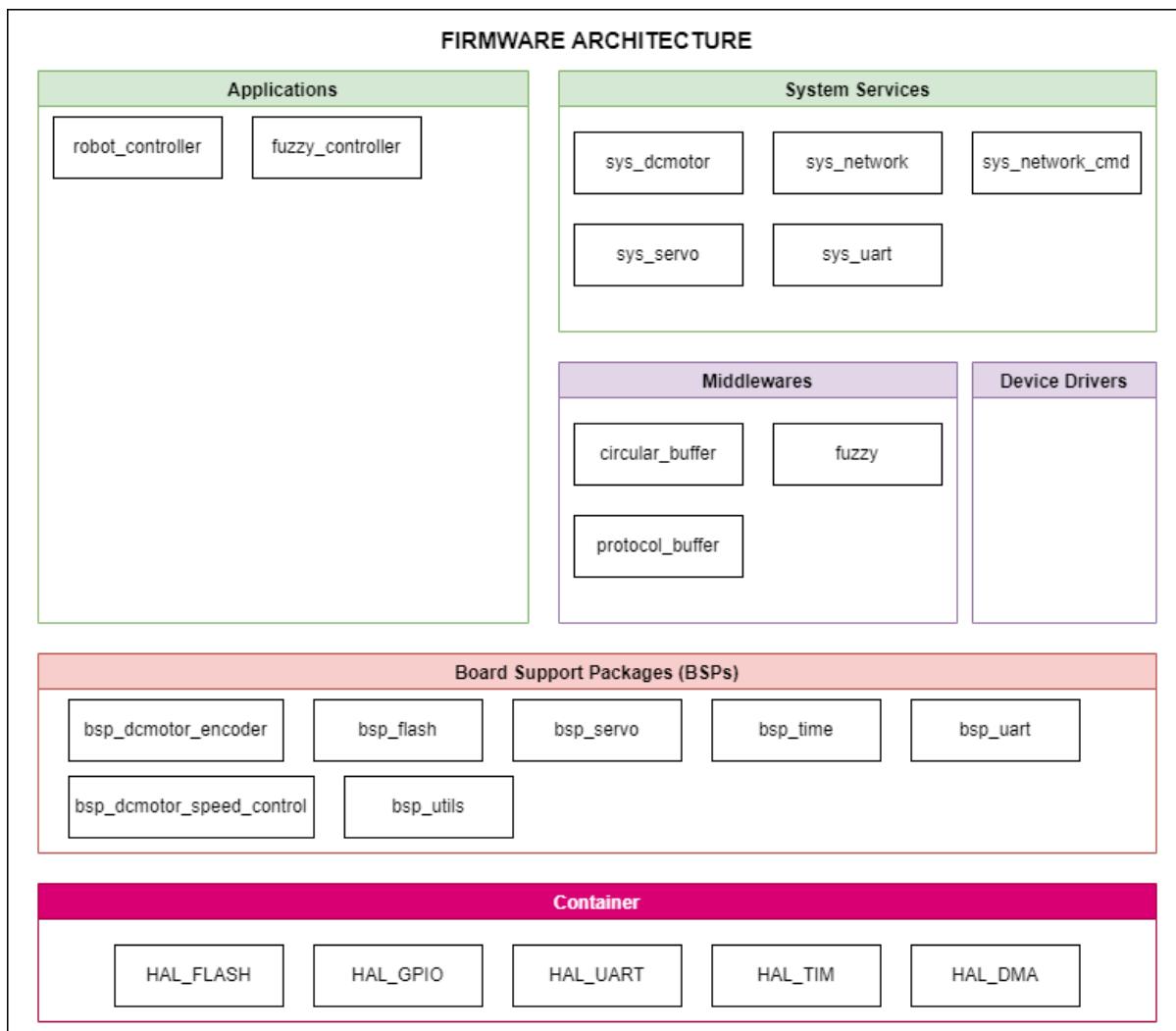
Board Support Package (BSP): Lớp này hướng tới việc quản lý bộ xử lý của hệ thống ví dụ như khởi động, cấu hình các ngoại vi cần thiết phục vụ chức năng của hệ thống.

Middlewares & Libraries: Lớp trung gian để quản lý giao tiếp giữa phần application và các device.

System Services: Lớp này có chức năng quản lý các thành phần liên quan đến chức năng của hệ thống ví dụ như bộ nhớ, công suất tiêu thụ, bắt lỗi, xử lý lỗi,...

Application: Đây là lớp trên cùng của kiến trúc, thực hiện chức năng hệ thống. Từ góc độ chức năng, tất cả các cấp module trong ứng dụng đều nhằm mục đích thực hiện các chức năng của lớp này.

4.2.1.3. Kiến trúc Firmware chi tiết



Hình 48. Kiến trúc Firmware chi tiết

Hình trên mô tả chi tiết kiến trúc Firmware của hệ thống. Firmware được chia thành nhiều lớp rõ ràng, mỗi lớp đảm nhận một nhóm chức năng riêng biệt, đảm bảo tính độc lập, dễ bảo trì và mở rộng.

a. Application Layer:

Là lớp cao nhất, trực tiếp thực hiện các chức năng điều khiển hệ thống. Bao gồm hai module chính:

- robot_controller: Điều khiển hoạt động tổng thể của mô hình xe tự hành.
- fuzzy_controller: Xử lý các thuật toán điều khiển Fuzzy phục vụ việc điều chỉnh góc lái.

b. System Services Layer:

Quản lý các dịch vụ hệ thống hỗ trợ hoạt động của các lớp trên:

- sys_dcmotor: Quản lý động cơ DC và tốc độ di chuyển.
- sys_servo: Quản lý động cơ Servo điều khiển góc lái và camera.
- sys_network và sys_network_cmd: Xử lý giao tiếp mạng nội bộ giữa các thành phần (GUI, Jetson TX-2, STM32).
- sys_uart: Quản lý giao tiếp UART.

c. Middlewares & Libraries Layer:

Cung cấp các thư viện hỗ trợ trung gian giữa Application và các Device Drivers:

- circular_buffer: Thư viện quản lý bộ đệm vòng cho giao tiếp dữ liệu.
- fuzzy: Thư viện thực thi các thuật toán Fuzzy Logic.
- protocol_buffer: Bộ xử lý dữ liệu Protocol Buffer mở rộng cho hệ thống nhúng.

d. Board Support Package (BSP) Layer:

Thực hiện khởi tạo, cấu hình các ngoại vi cơ bản của vi điều khiển:

- bsp_dcmotor_encoder: Quản lý tín hiệu Encoder động cơ DC.
- bsp_dcmotor_speed_control: Quản lý điều khiển tốc độ động cơ DC.
- bsp_flash: Quản lý lưu trữ Flash.
- bsp_servo: Quản lý tín hiệu PWM cho Servo.
- bsp_time: Quản lý bộ định thời Timer.
- bsp_uart: Quản lý module UART.
- bsp_utils: Các hàm hỗ trợ tiện ích chung.

e. Container Layer (HAL Layer):

Cung cấp các API trừu tượng hóa việc giao tiếp phần cứng:

- HAL_FLASH, HAL_GPIO, HAL_UART, HAL_TIM, HAL_DMA: Các module cơ bản của thư viện HAL dùng để truy cập ngoại vi vi điều khiển.

4.2.2. Các tính năng của Firmware

Firmware trên vi điều khiển STM32F407 được xây dựng để thực hiện các chức năng điều khiển và quản lý toàn bộ hoạt động của mô hình xe tự hành.

4.2.2.1. Điều khiển động cơ DC:

Firmware thực hiện điều khiển tốc độ hai bánh sau của xe thông qua module H-Bridge, sử dụng thuật toán điều khiển PID để đảm bảo tốc độ thực tế bám sát tốc độ đặt với độ chính xác cao. Tốc độ tối đa có thể đạt được của mô hình hiện tại là 20 cm/s.

4.2.2.2. Điều khiển góc lái động cơ Servo:

Firmware điều chỉnh góc lái của hai bánh trước bằng cách điều khiển động cơ Servo góc, dựa trên kết quả đầu ra từ bộ điều khiển Fuzzy. Góc lái được cập nhật liên tục theo tín hiệu thu thập từ thị giác máy. Độ lớn góc lái của động cơ là 45 độ.

4.2.2.3. Điều khiển góc quay camera:

Firmware quản lý Servo xoay camera thông qua máy trạng thái, cho phép thay đổi góc nhìn của camera theo yêu cầu vận hành (chế độ rẽ trái, rẽ phải, băng qua ngã tư...).

4.2.2.4. Giao tiếp dữ liệu qua UART và Bluetooth:

STM32F407 đảm nhiệm vai trò trung gian giao tiếp giữa GUI và Jetson TX-2. Dữ liệu giao tiếp sử dụng Protocol Buffer để đảm bảo tính gọn nhẹ, đồng nhất và dễ dàng mở rộng.

4.2.2.5. Xử lý gói tin và chuyển tiếp thông minh:

Firmware giải mã các gói tin nhận được, kiểm tra địa chỉ đích và thực hiện xử lý hoặc chuyển tiếp gói tin phù hợp đến GUI hoặc Jetson TX-2.

4.2.2.6. Thu thập và xử lý tín hiệu từ Encoder:

Firmware đo tốc độ thực tế của xe thông qua bộ Encoder gắn trên bánh sau, sử dụng tín hiệu xung để tính toán vận tốc hiện tại, phục vụ điều chỉnh PID.

4.2.2.7. Quản lý thời gian hệ thống:

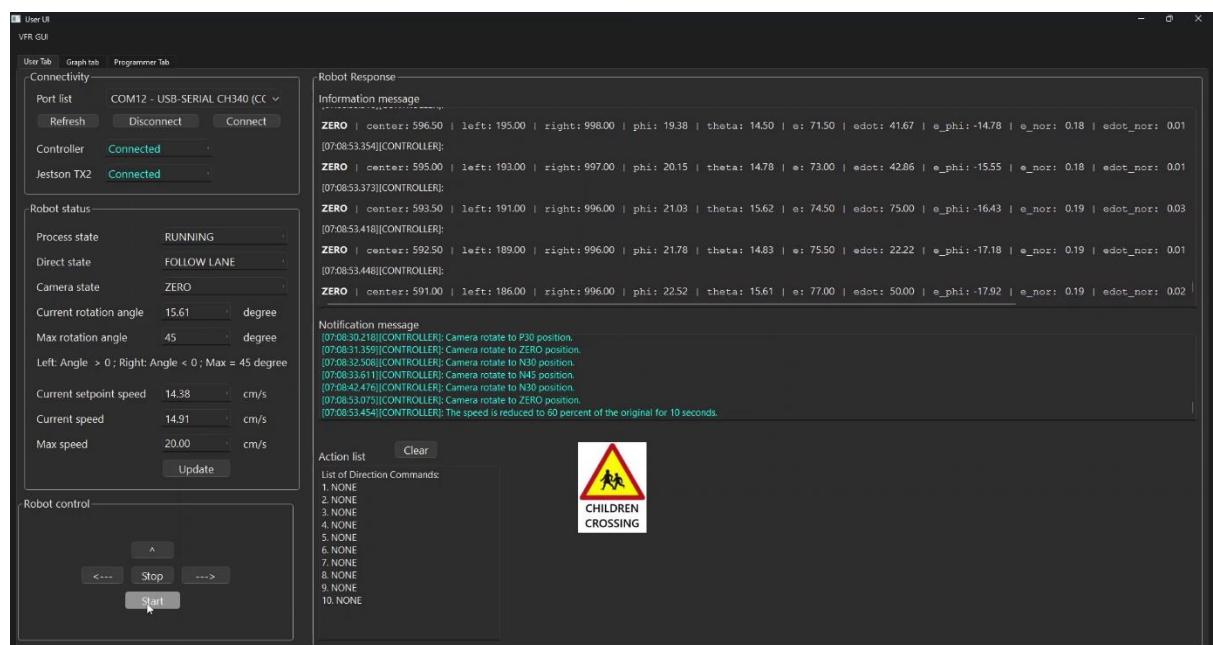
Firmware đồng bộ thời gian nội bộ với GUI qua các gói tin chứa thông tin epoch time, đảm bảo dữ liệu vận hành được ghi nhận chính xác theo thời gian thực.

4.2.2.8. Cập nhật và lưu trữ tham số điều khiển:

Các hệ số của bộ điều khiển Fuzzy có thể được cập nhật động từ GUI. Firmware hỗ trợ lưu trữ các tham số này vào bộ nhớ Flash để giữ nguyên sau khi khởi động lại.

4.3. Thiết kế phần mềm của GUI

4.3.1. Giao diện người dùng



Hình 49. Giao diện người dùng của GUI

Giao diện người dùng giúp người dùng tương tác, theo dõi và điều khiển mô hình xe tự hàn. Toàn bộ giao diện được chia thành nhiều khối chức năng chính, mỗi khối đảm nhiệm một vai trò cụ thể trong hệ thống.

Khối Connectivity: cho phép người dùng chọn cổng giao tiếp với module Bluetooth, thực hiện các thao tác kết nối hoặc ngắt kết nối. Các cổng khả dụng sẽ được liệt kê và làm mới bằng nút “Refresh”. Đồng thời, giao diện cũng hiển thị trạng thái kết nối hiện tại giữa GUI với vi điều khiển và Jetson TX-2.

Khối Robot Status: hiển thị các thông số hoạt động hiện tại của xe tự hành. Bao gồm trạng thái xử lý (process state), trạng thái điều hướng (direct state), góc xoay hiện tại và góc xoay tối đa của camera, tốc độ đặt, tốc độ thực tế và tốc độ tối đa của robot. Ngoài ra, người dùng có thể điều chỉnh giới hạn góc đánh lái servo và giới hạn tốc độ tối đa thông qua nút “Update”.

Khối Robot Control: cho phép người dùng khởi động hoặc dừng quá trình hoạt động của robot thông qua các nút điều khiển cơ bản. Ngoài ra, người dùng có thể thêm các hành động điều hướng như đi thẳng, rẽ trái, rẽ phải vào hàng đợi hành động để robot thực hiện tại các ngã ba, ngã tư tiếp theo.

Khối Robot Response: Hiển thị các thông số đầu ra từ hệ thống xử lý ảnh và các giá trị điều khiển được tính toán như vị trí trung tâm làn đường, góc nghiêng, sai số và đạo hàm sai số. Ngoài ra còn thể hiện các thông báo trạng thái và hành vi đang được thực hiện bởi robot, giúp người dùng theo dõi hoạt động thực tế. Kèm với đó còn hiển thị các hành động điều hướng đã được thêm vào hàng đợi và đang chờ thực thi và bên cạnh hiển thị hình ảnh các biển báo giao thông đã được mô hình nhận diện trong thời gian thực.

4.3.2. Giao diện biểu đồ



Hình 50. Giao diện biểu đồ của GUI

Giao diện bao gồm bốn biểu đồ chính, mỗi biểu đồ thể hiện sự thay đổi của một thông số cụ thể theo trục thời gian (đơn vị giây):

Biểu đồ sai số tọa độ điểm trung tâm làn đường theo thời gian: biểu diễn độ lệch giữa vị trí trung tâm của xe so với trung tâm làn đường. Đơn vị sai số là pixel.

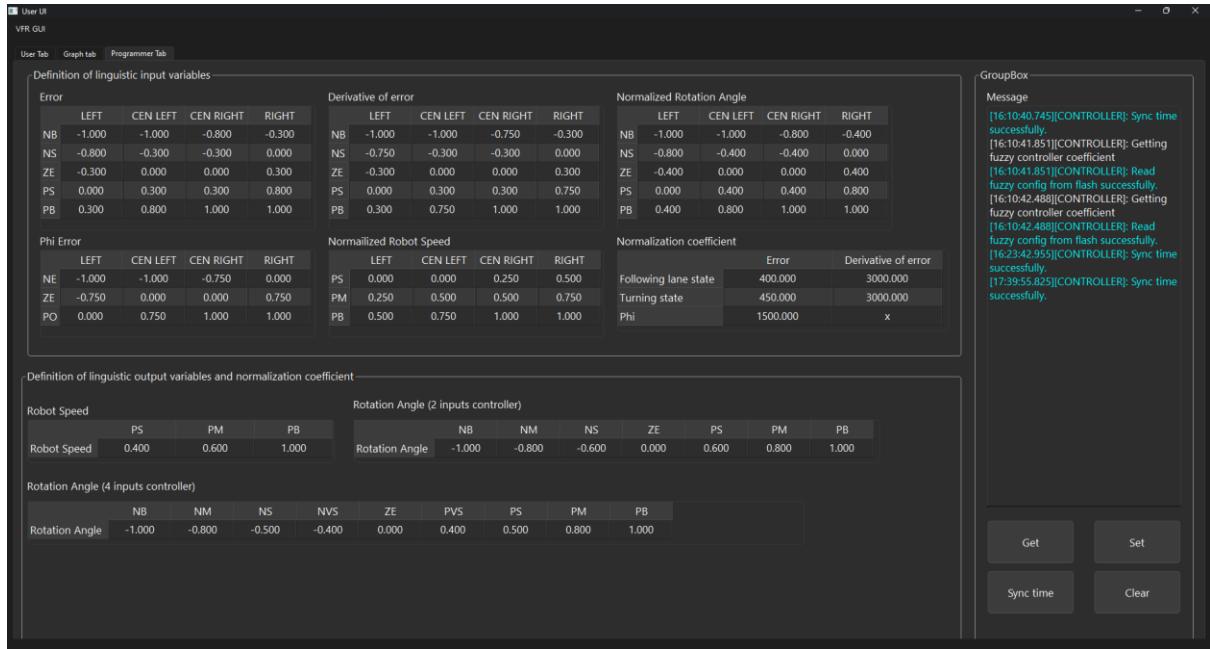
Biểu đồ sai số góc nghiêng làn đường theo thời gian: biểu thị góc lệch giữa góc nhìn hiện tại của camera và hướng song song với làn đường, đơn vị là độ. Thông số này phản ánh mức độ "nghiêng" của làn đường trong khung hình so với hướng di chuyển mong muốn.

Biểu đồ góc lái theo thời gian: thể hiện giá trị góc điều khiển của động cơ servo đánh lái, đơn vị là độ.

Biểu đồ tốc độ xe theo thời gian: thể hiện tốc độ thực tế của mô hình xe tự hành, đơn vị là cm/s.

Mỗi đường biểu diễn trên biểu đồ được tô màu khác nhau tùy theo trạng thái xoay của camera tại thời điểm tương ứng. Nhờ đó, người dùng có thể dễ dàng dõi chiếu biến động của hệ thống điều khiển với từng góc nhìn cụ thể của camera trong quá trình di chuyển.

4.3.3. Giao diện thiết lập



Hình 51. Giao diện thiết lập của GUI

Giao diện thiết lập cho phép người dùng can thiệp trực tiếp vào các thông số của hệ thống điều khiển mờ, bao gồm việc điều chỉnh các giá trị ngôn ngữ đầu vào, đầu ra và hệ số chuẩn hóa. Giao diện được chia thành ba khối chính như sau:

Khối định nghĩa giá trị ngôn ngữ của biến đầu vào: hiển thị các bảng thiết lập giá trị của các biến mờ như sai số tâm làn (Error), đạo hàm sai số (Derivative of Error), sai số góc nghiêng (Phi Error), góc lái chuẩn hóa (Normalized Rotation Angle) và tốc độ robot chuẩn hóa (Normalized Robot Speed). Người dùng có thể nhập thủ công các giá trị cho từng mức ngôn ngữ như NB (Negative Big), ZE (Zero), PB (Positive Big),... nhằm điều chỉnh quá trình suy luận mờ. Các hệ số chuẩn hóa (Normalization coefficient) tương ứng cũng được cấu hình tại đây.

Khối định nghĩa giá trị ngõ ra: bao gồm bảng thiết lập các giá trị đầu ra của bộ điều khiển mờ. Cụ thể là giá trị tốc độ robot ứng với các mức ngôn ngữ (PS, PM, PB), góc đánh lái điều khiển hai ngõ vào (góc lệch và sai số vị trí), và góc đánh lái điều khiển bốn ngõ vào (kết hợp sai số, đạo hàm sai số và các biến khác).

Khối hiển thị thông báo và chức năng hệ thống: thể hiện các thông báo được gửi về từ STM32F407 như trạng thái đọc/ghi cấu hình điều khiển. Đồng thời, cung cấp các nút chức năng như:

- Get: lấy dữ liệu cấu hình hiện tại từ bộ điều khiển.
- Set: gửi cấu hình đang hiển thị lên bộ điều khiển để cập nhật.
- Sync time: đồng bộ thời gian giữa máy tính và vi điều khiển.
- Clear: xóa toàn bộ log thông báo trên giao diện.

4.4. Thiết kế phần mềm trên Jetson TX-2

Phần mềm vận hành trên máy tính nhúng Jetson TX-2 được thiết kế theo kiến trúc đa tiến trình, nhằm tối ưu hóa hiệu suất xử lý và đảm bảo khả năng phản hồi nhanh trong quá trình di chuyển thực tế của xe tự hành.

Sau khi nhận lệnh bắt đầu xử lý ảnh từ vi điều khiển STM32F407, chương trình chính trên Jetson TX-2 sẽ liên tục thu nhận các khung hình từ camera Gopro Hero 8. Mỗi khung hình sau khi thu nhận được đồng thời đẩy vào hai hàng đợi:

- Một hàng đợi phục vụ tiến trình xử lý xác định làn đường (Lane Detection).
- Một hàng đợi phục vụ tiến trình nhận diện biển báo giao thông bằng mô hình YOLO v4.

Hai tiến trình xử lý hoạt động song song như sau:

- Tiến trình xác định làn đường nhận ảnh từ hàng đợi, thực hiện thuật toán xác định vạch kẻ đường, tính toán vị trí trung tâm làn đường và góc nghiêng, sau đó ghi kết quả vào một hàng đợi kết quả riêng.
- Tiến trình nhận diện biển báo nhận ảnh từ hàng đợi, phát hiện biển báo giao thông, cập nhật các cờ trạng thái và ghi kết quả vào một hàng đợi kết quả riêng.

Chương trình chính sẽ liên tục kiểm tra các hàng đợi kết quả. Khi phát hiện có dữ liệu mới từ bất kỳ tiến trình nào, chương trình chính sẽ lấy dữ liệu, đóng gói bằng Protocol Buffer và gửi về STM32F407 thông qua giao tiếp UART.

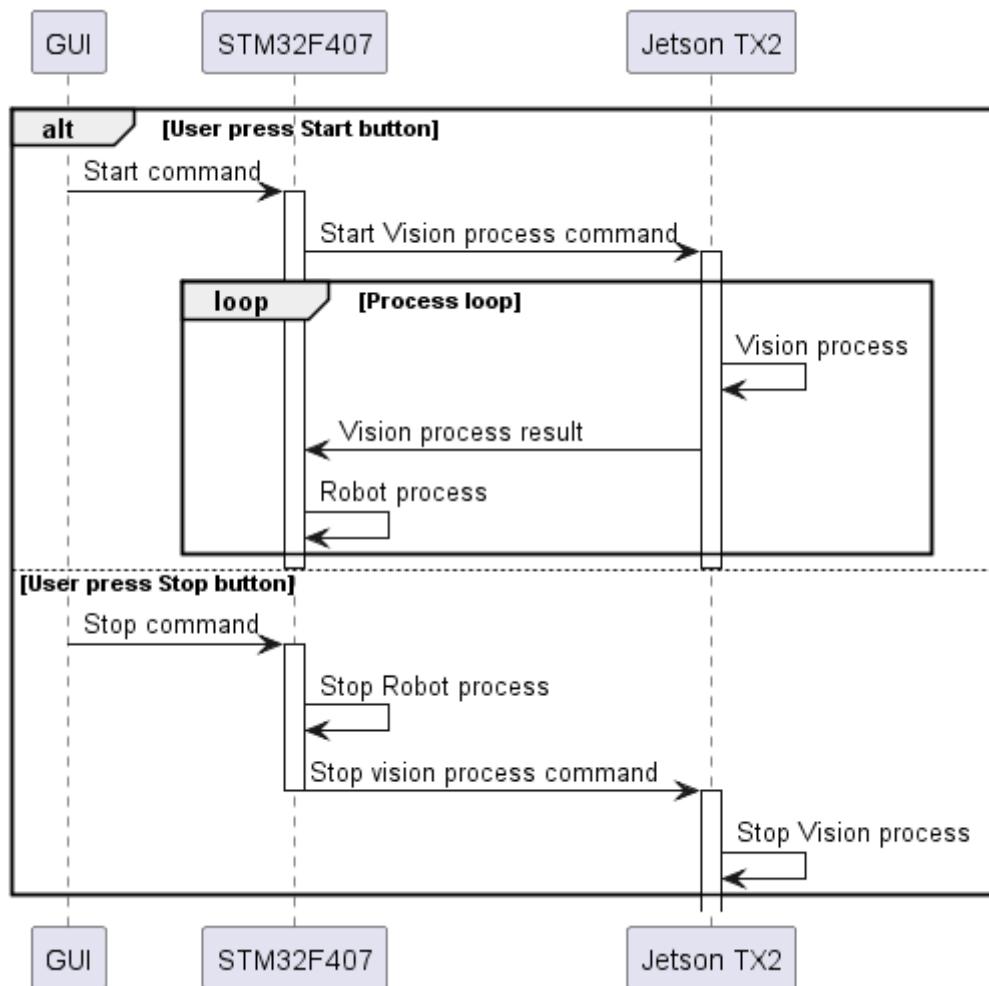


Hình 52. Sơ đồ hoạt động của chương trình chính trên Jetson TX-2

Chương trình chính chịu trách nhiệm thu nhận ảnh, phân phôi vào các hàng đợi cho tiến trình con xử lý, đồng thời kiểm tra kết quả từ các tiến trình để đóng gói và gửi dữ liệu về vi điều khiển.

4.5. Tổng quan hệ thống mô hình xe tự hành

Việc phối hợp giữa các thành phần GUI, vi điều khiển STM32F407 và máy tính nhúng Jetson TX-2 được thực hiện thông qua giao tiếp tuần tự và cơ chế phản hồi liên tục.



Hình 53. Sơ đồ trình tự giao tiếp – tổng quan chương trình hệ thống

Khi người dùng nhấn nút bắt đầu (Start) trên giao diện GUI, một gói lệnh điều khiển ra lệnh bắt đầu chương trình sẽ được gửi đến vi điều khiển STM32F407 thông qua kết nối Bluetooth. Sau khi nhận được lệnh, vi điều khiển sẽ truyền tiếp lệnh khởi động xử lý ảnh đến Jetson TX-2 qua giao tiếp UART. Jetson TX-2 khi nhận lệnh sẽ bắt đầu chu trình thu nhận và xử lý ảnh, đồng thời sử dụng đa tiến trình để thực hiện đồng thời hai tác vụ chính: xác định làn đường và nhận diện biển báo giao thông. Quá trình cho việc khi người dùng nhấn nút dừng (Stop) trên GUI cũng diễn ra tương tự như vậy.

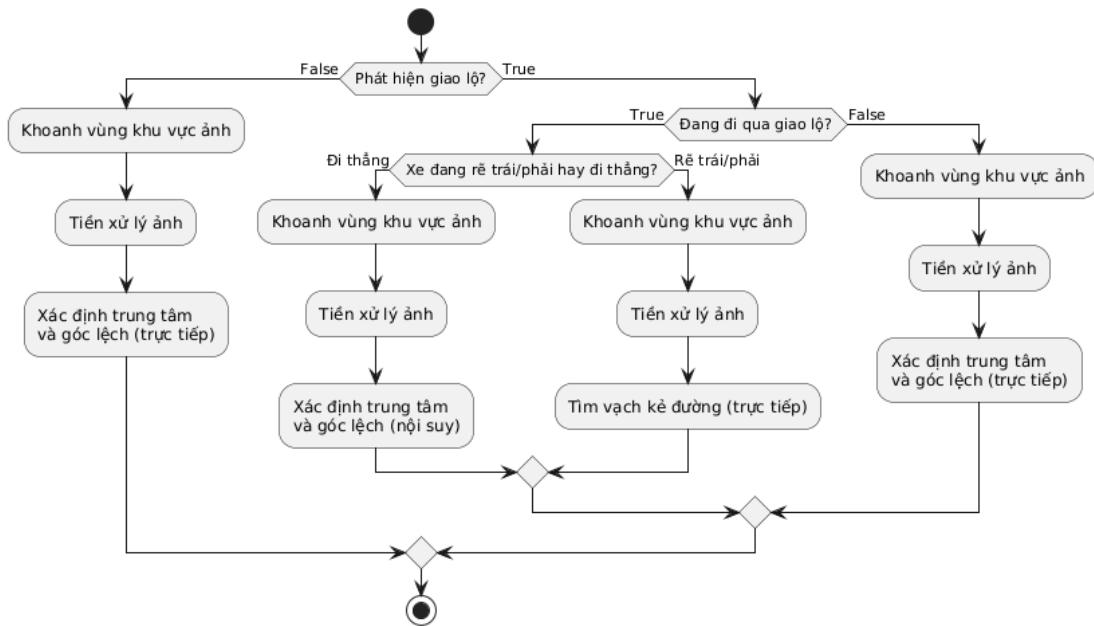
Kết quả xử lý bao gồm tọa độ trung tâm làn đường, góc nghiêng của làn đường và các thông tin liên quan đến biển báo sẽ được đóng gói và gửi trở lại cho vi điều khiển. STM32F407 sử dụng các thông tin này làm đầu vào cho các khối xử lý điều khiển, bao gồm bộ điều khiển mờ (Fuzzy Controller), các bộ điều khiển PID, logic trạng thái điều hướng và thuật toán xác định tốc độ phù hợp. Từ đó, hệ thống tính toán và gửi lệnh điều khiển góc lái và tốc độ đến các cơ cấu cơ khí trên xe.

Quá trình này được lặp liên tục trong suốt thời gian hoạt động của xe. Ngoài ra, GUI cũng liên tục nhận phản hồi từ vi điều khiển để hiển thị thông tin trạng thái hiện tại như tốc độ, góc lái, trạng thái xử lý ảnh, các thông số hệ thống và kết quả nhận diện biển báo.

Chương 5: Xây dựng giải thuật

5.1. Giải thuật xử lý ảnh làn đường

5.1.1. Chương trình chính



Hình 54. Mô tả hoạt động của chương trình chính xử lý ảnh

Chương trình chính xử lý ảnh hoạt động dựa trên các điều kiện cốt lõi: “phát hiện giao lộ” và “trạng thái đang đi qua giao lộ”. Dựa vào hai điều kiện này, chương trình phân chia luồng xử lý thành hai nhánh chính để phù hợp với từng tình huống cụ thể khi xe tham gia giao thông.

Trong trường hợp chưa phát hiện giao lộ, tức là phương tiện không nằm trong khu vực ngã ba hoặc ngã tư, chương trình sẽ thực hiện lần lượt các bước sau: khoanh vùng khu vực ảnh nhằm xác định vùng quan tâm trong ảnh đầu vào; tiếp theo là tiền xử lý ảnh với các thao tác cơ bản như lọc nhiễu, phân ngưỡng (thresholding), v.v.; cuối cùng là xác định vị trí trung tâm và góc lệch của đoạn đường bằng phương pháp trực tiếp, đảm bảo độ chính xác cao khi dữ liệu ảnh còn đầy đủ.

Ngược lại, khi phát hiện có giao lộ phía trước, chương trình sẽ kiểm tra thêm trạng thái hiện tại của phương tiện là đã hay chưa đi vào giao lộ. Nếu chưa đi qua,

chương trình tiếp tục thực hiện khoanh vùng và tiền xử lý ảnh như bình thường, sau đó sẽ xác định trung tâm và góc lệch cũng bằng phương pháp trực tiếp.

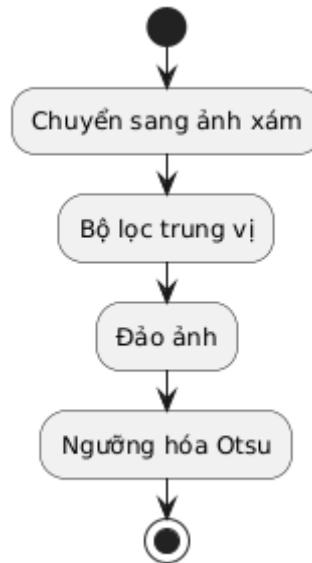
Tuy nhiên, khi phương tiện đang thực sự đang đi qua giao lộ chương trình sẽ kiểm tra xem xe đang muốn rẽ trái, rẽ phải hay đi thẳng để lựa chọn phương pháp xử lý phù hợp.

Nếu xe muốn rẽ trái hoặc rẽ phải chương trình sẽ thực hiện các bước tương tự với các bước xử lý khi xe đang đi ở khu vực không có giao lộ, tuy nhiên thay vì xác định trung tâm và góc lệch của mặt đường, chương trình sẽ chỉ tìm vạch kẻ đường bên trái (nếu xe rẽ trái) hoặc vạch kẻ đường bên phải (nếu xe rẽ phải). Vì khi rẽ trái hoặc rẽ phải bộ điều khiển sẽ điều khiển xe bám theo vạch kẻ đường bên trái hoặc phải.

Còn nếu xe đi thẳng qua giao lộ – nơi dữ liệu ảnh có thể thiếu do đặc điểm của giao lộ – chương trình sẽ tiến hành khoanh vùng lại vùng quan tâm, vùng ảnh này thường sẽ lớn hơn vùng ảnh sử dụng trong phương pháp trực tiếp nhằm thu thập nhiều dữ liệu hơn, tiếp đến chương trình sẽ thực hiện tiền xử lý ảnh và sử dụng phương pháp nội suy để ước lượng vị trí trung tâm và góc lệch. Phương pháp này cho phép hệ thống duy trì khả năng xác định hướng di chuyển trong nhiều điều kiện khác nhau.

5.1.2. Chương trình con

5.1.2.1. Tiền xử lý ảnh



Hình 55. Mô tả hoạt động của chương trình tiền xử lý ảnh

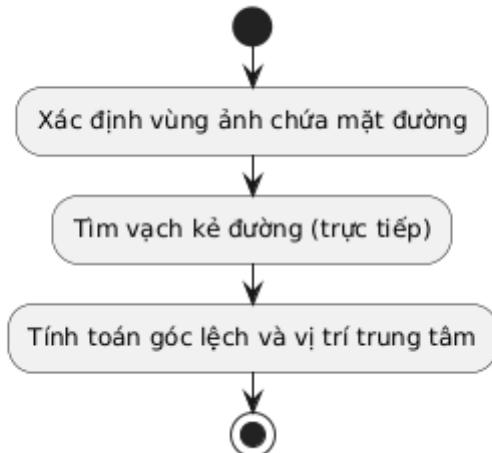
Quá trình tiền xử lý ảnh đóng vai trò quan trọng trong việc chuẩn bị dữ liệu đầu vào cho các thuật toán phát hiện vạch kẻ đường. Mục tiêu chính của giai đoạn này là làm nổi bật các đặc trưng hình ảnh liên quan đến vạch kẻ, đồng thời loại bỏ nhiễu và các chi tiết không cần thiết, giúp nâng cao độ chính xác trong việc xác định vị trí của vạch kẻ đường.

Quy trình được thực hiện qua các bước tuần tự như sau: đầu tiên, ảnh đầu vào được chuyển sang ảnh xám (grayscale) nhằm đơn giản hóa dữ liệu và giảm thời gian tính toán. Tiếp theo, ảnh được đi qua bộ lọc trung vị (median blur) nhằm loại bỏ nhiễu, đặc biệt là nhiễu muối tiêu mà không làm mất thông tin biên của các vạch kẻ. Ảnh sau đó được đảo ngược (inverted) để thuận tiện hơn trong việc lập trình. Cuối cùng, chương trình áp dụng ngưỡng hóa Otsu (Otsu Thresholding) để chuyển đổi ảnh sang dạng nhị phân, qua đó tách biệt rõ ràng các vạch kẻ đường so với phần nền còn lại.

Toàn bộ quy trình tiền xử lý này được thiết kế nhằm tối ưu hóa khả năng nhận diện vạch kẻ đường trong môi trường có điều kiện ánh sáng và chất lượng

ảnh khác nhau, qua đó nâng cao độ tin cậy của hệ thống phát hiện vạch kẻ đường giúp hỗ trợ các bước tiếp theo trong hệ thống điều hướng.

5.1.2.2. Xác định góc lệch và vị trí trung tâm (trực tiếp)



Hình 56. Mô tả hoạt động của chương trình xác định góc lệch và vị trí trung tâm (trực tiếp)

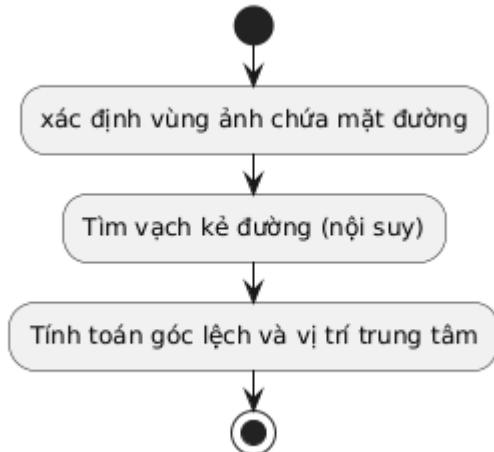
Chức năng xác định góc lệch và vị trí trung tâm theo phương pháp trực tiếp được sử dụng trong các tình huống mà chất lượng ảnh đủ tốt, không bị thiếu dữ liệu. Mục tiêu của bước này là xác định chính xác vị trí của trung tâm mặt đường so với phương tiện, cũng như góc lệch giữa vị trí hiện tại của mặt đường so với đoạn đường phía trước nhằm thể hiện hướng đi chuẩn của đoạn đường.

Quá trình được thực hiện thông qua bốn bước chính. Trước tiên, chương trình tiến hành xác định vùng ảnh chỉ chứa mặt đường để tập trung vào khu vực lân cận có khả năng chứa vạch kẻ. Tiếp theo, chương trình lần lượt tìm vạch kẻ đường bên trái và vạch kẻ đường bên phải trong vùng lân cận mặt đường bằng phương pháp trực tiếp. Việc phát hiện này thường dựa trên các đặc trưng hình dạng và độ tương phản rõ nét giữa vạch kẻ và mặt đường sau bước tiền xử lý.

Sau khi xác định được vị trí hai vạch kẻ, chương trình tiến hành tính toán góc lệch và vị trí trung tâm. Góc lệch được xác định dựa trên hướng tạo bởi trung tâm của mặt đường ở vị trí hiện tại so với trung tâm của mặt đường ở đoạn đường phía trước, trong khi vị trí trung tâm được tính bằng trung điểm giữa hai vạch. Thông tin này đóng vai trò đầu vào quan trọng cho hệ thống điều hướng hoặc hiệu chỉnh hướng lái của phương tiện.

Phương pháp trực tiếp mang lại độ chính xác cao khi ảnh đầu vào rõ nét, đồng thời cho phép hệ thống phản ứng nhanh với thay đổi trong điều kiện di chuyển thực tế.

5.1.2.3. Xác định góc lệch và vị trí trung tâm (nội suy)



Hình 57. Mô tả hoạt động của chương trình xác định góc lệch và vị trí trung tâm (nội suy)

Chức năng xác định góc lệch và vị trí trung tâm theo phương pháp nội suy được áp dụng trong các tình huống khi thông tin mà ảnh đầu vào cung cấp không đủ để có thể tìm ra các vạch kẻ đường theo phương pháp trực tiếp, nguyên nhân có thể xuất phát từ các đặc điểm của giao lộ khiến xe không thể nhìn được các vạch kẻ đường làm dữ liệu đầu vào bị thiếu hụt. Mục tiêu chính của phương pháp nội suy này là ước lượng tương đối chính xác vị trí trung tâm mặt đường và góc lệch, giúp duy trì khả năng điều hướng ngay cả khi ảnh không cung cấp đầy đủ thông tin hình ảnh.

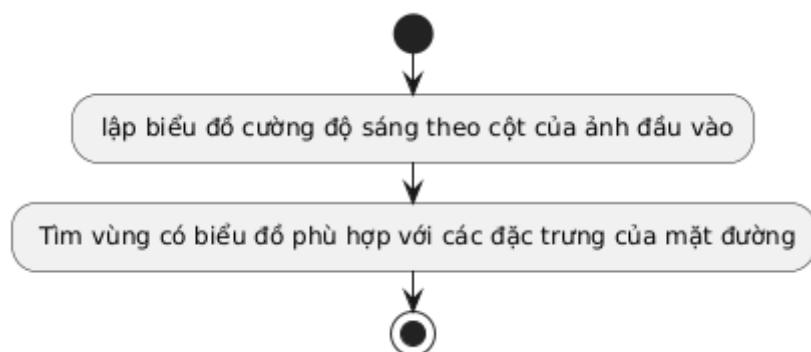
Quá trình xử lý gồm bốn bước chính. Đầu tiên, chương trình khoanh vùng ảnh chỉ chứa mặt đường. Sau đó, chương trình tiến hành tìm làn đường bên trái và làn đường bên phải bằng phương pháp nội suy dựa trên các thông tin thu thập được.

Sau khi ước lượng được hai vạch làn đường, chương trình tiến hành tính toán góc lệch và vị trí trung tâm tương tự như trong phương pháp trực tiếp. Cụ thể, góc lệch được xác định dựa trên sự thay đổi tương đối giữa trung tâm hiện tại và

trung tâm dự kiến của mặt đường phía trước. Vị trí trung tâm được tính theo trung điểm giữa hai làn đường đã nội suy.

Phương pháp gián tiếp giúp hệ thống duy trì khả năng định hướng ổn định trong các điều kiện khó khăn như giao lộ, khu vực bị mờ, hoặc khi một phần vách kè bị che khuất. Dù độ chính xác có thể thấp hơn so với phương pháp trực tiếp, nhưng phương pháp này đóng vai trò quan trọng trong việc đảm bảo tính liên tục cho hệ thống điều hướng.

5.1.2.4. Xác định vùng ảnh chứa mặt đường



Hình 58. Mô tả hoạt động của chương trình xác định vùng ảnh chứa mặt đường

Chức năng xác định vùng ảnh chứa mặt đường có vai trò quan trọng trong quá trình xử lý ảnh nhằm phục vụ cho bước phát hiện vách kè đường.

Quá trình này được thực hiện thông qua hai bước chính. Đầu tiên, chương trình sẽ lập biểu đồ cường độ sáng theo cột của ảnh đầu vào. Biểu đồ này biểu diễn tổng mức độ sáng trên từng cột của ảnh, từ đó phản ánh các khu vực có sự phân bố đặc trưng về độ sáng. Trong thực tế, mặt đường thường có cường độ sáng thấp, độ sáng tương đối ổn định theo phương ngang, và không có nhiều thay đổi đột ngột về ánh sáng như các vùng có vật thể khác (vỉa hè, xe cộ, cây cối...).

Sau khi có biểu đồ, chương trình sẽ tiến hành tìm và chọn ra vùng có biểu đồ phù hợp nhất với đặc trưng của mặt đường. Cụ thể, vùng ảnh được chọn là nơi có độ sáng trung bình thấp, ít dao động, đảm bảo rằng chỉ bao gồm phần mặt đường, tránh nhiễu từ các khu vực không liên quan. Việc khoanh vùng chính xác này là

điều kiện tiên quyết để đảm bảo hiệu quả cho thuật toán phát hiện vạch kẻ được thực hiện ở bước sau.

5.1.2.5. Tìm vạch kẻ đường (trực tiếp)



Hình 59. Mô tả hoạt động của chương trình tìm vạch kẻ đường (trực tiếp)

Chức năng tìm vạch kẻ đường là một thành phần then chốt trong hệ thống phát hiện làn đường, với mục tiêu nhận diện chính xác các vạch kẻ đường bên trái và bên phải trong ảnh đã qua xử lý. Quá trình này được thực hiện tuần tự qua các bước như sau:

Đầu tiên, chương trình tiến hành trích xuất một số hàng ảnh nằm gần đáy của khung hình, nơi mà vạch kẻ đường thường xuất hiện rõ ràng trong tầm nhìn từ camera gắn trên phương tiện. Việc lựa chọn hàng gần đáy (ví dụ: hàng 1079 hoặc 1279 trong ảnh có độ phân giải 720×1280) giúp đảm bảo rằng các vạch kẻ chưa bị biến dạng do góc nhìn và nằm gần vùng điều khiển của phương tiện.

Tiếp theo, chương trình duyệt qua từng điểm ảnh trên hàng ảnh vừa trích xuất để tìm các điểm có khả năng thuộc vạch kẻ đường. Các điểm này thường có cường độ sáng cao, tương phản rõ với mặt đường xung quanh.

Sau khi đánh dấu các điểm nghi ngờ, chương trình tiến hành phân biệt các điểm này là điểm thuộc vạch kẻ đường bên trái hay bên phải, dựa trên vị trí tương đối của các điểm sáng so với vùng ảnh đã được xác định là mặt đường. Điều này giúp định hướng lại thông tin để các bước xử lý sau có thể sử dụng một cách chính xác theo ngữ cảnh thực tế (bên trái xe và bên phải xe).

Cuối cùng, để tăng độ tin cậy, chương trình sẽ kiểm tra các điểm ảnh lân cận của từng điểm đã được đánh dấu. Việc xác minh này nhằm loại bỏ nhiễu và đảm bảo rằng điểm đang xét không phải là một điểm sáng ngẫu nhiên hạy vật thể ngoài vùng mặt đường, từ đó tăng độ chính xác trong nhận dạng vạch kẻ.

Toàn bộ quá trình giúp tìm các điểm đại diện cho vạch kẻ đường, đóng vai trò quan trọng trong việc tính toán vị trí trung tâm chính xác của mặt đường và góc lệch thực tế.

5.1.2.6. Tìm vạch kẻ đường (nội suy)



Hình 60. Mô tả hoạt động của chương trình tìm vạch kẻ đường (nội suy)

Phương pháp nội suy được sử dụng trong trường hợp phương tiện đang đi qua giao lộ – nơi các vạch kẻ đường gần xe không còn hiển thị rõ ràng trong ảnh đầu vào. Trong bối cảnh này, chương trình không thể xác định trực tiếp vạch kẻ đường tại vùng gần bánh xe, nên cần tập trung vào việc tìm các vạch kẻ ở xa phía trước và sử dụng nội suy để ước lượng lại quỹ đạo vạch kẻ đường, từ đó hỗ trợ điều hướng.

Quá trình được thực hiện như sau: đầu tiên, chương trình trích xuất biên trái và biên phải của vùng mặt đường đã được xác định từ bước xử lý trước. Hai biên này vừa đóng vai trò là giới hạn cho vùng tìm kiếm, vừa được sử dụng để khởi tạo hai giá trị x ban đầu – tương ứng với vị trí của vạch kẻ đường trái và phải.

Tiếp theo, chương trình sẽ duyệt ảnh theo chiều từ trên xuống dưới, bắt đầu từ $y = 0$. Tại mỗi bước lặp, một vùng tìm kiếm hình chữ nhật nhỏ sẽ được tạo ra quanh vị trí x hiện tại, nằm giữa hai biên giới hạn và có chiều cao cố định. Vùng này được dùng để kiểm tra xem có sự xuất hiện của vạch kẻ đường hay không. Nếu có vạch kẻ trong vùng, tọa độ x sẽ được cập nhật theo vị trí vạch vừa phát hiện; nếu không, x được giữ nguyên. Sau đó, giá trị y mới được tăng lên để tiếp tục duyệt vùng tiếp theo. Việc tăng y sau khi xử lý tại vị trí hiện tại giúp đảm bảo rằng không có vùng nào bị bỏ sót trong quá trình duyệt.

Toàn bộ quá trình được lặp lại cho đến khi vùng tìm kiếm bao phủ toàn bộ chiều cao mặt đường. Kết quả là một tập hợp các điểm được đánh dấu là thuộc về vạch kẻ đường, dù có thể không hoàn toàn liên tục. Từ các điểm thu được, chương trình tiến hành nội suy để tái dựng lại hình dạng vạch kẻ, tạo thành một đường mờ phỏng gần đúng với vạch thực tế trong ảnh. Đầu ra của bước này sẽ được sử dụng trong các giai đoạn tiếp theo như xác định trung tâm mặt đường và tính toán góc lệch.

Phương pháp nội suy tuy có thể cho độ chính xác thấp hơn so với phát hiện trực tiếp, nhưng lại đảm bảo tính liên tục, khả năng phục hồi và hoạt động ổn định trong các điều kiện thiếu dữ liệu – đặc biệt là tại giao lộ hoặc khi ảnh hưởng bởi điều kiện ánh sáng không thuận lợi.

5.2. Giải thuật nhận diện biển báo sử dụng YOLOv4-tiny

5.2.1. Giới thiệu bài toán và tập dữ liệu

Trong hệ thống xe tự hành mini, việc nhận diện chính xác các biển báo giao thông là yếu tố then chốt giúp phương tiện hiểu và tuân thủ đúng luật khi di chuyển. Để đạt được điều này, đề tài tiến hành huấn luyện mô hình YOLOv4-Tiny nhằm phát

hiện và phân loại các loại biển báo phổ biến như biển cấm, biển cảnh báo, biển hiệu lệnh,... Việc lựa chọn YOLOv4-Tiny – một phiên bản nhẹ và tối ưu cho thiết bị nhúng - giúp đảm bảo khả năng xử lý thời gian thực trong khi vẫn duy trì độ chính xác chấp nhận được.

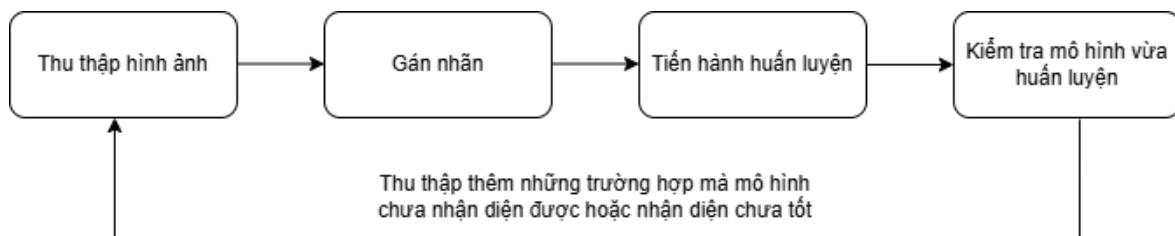
Dưới đây là danh sách 8 biển báo nhóm chọn để thực hiện trong đề tài:

STT	Biển báo	Tên
1		Bus_Stop
2		Children_Crossing
3		Green_Light
4		Left_Turn_Only
5		No_Stopping
6		Red_Light

7	 SPEED LIMIT 40	Speed_Limit_40
8	 STOP	Stop

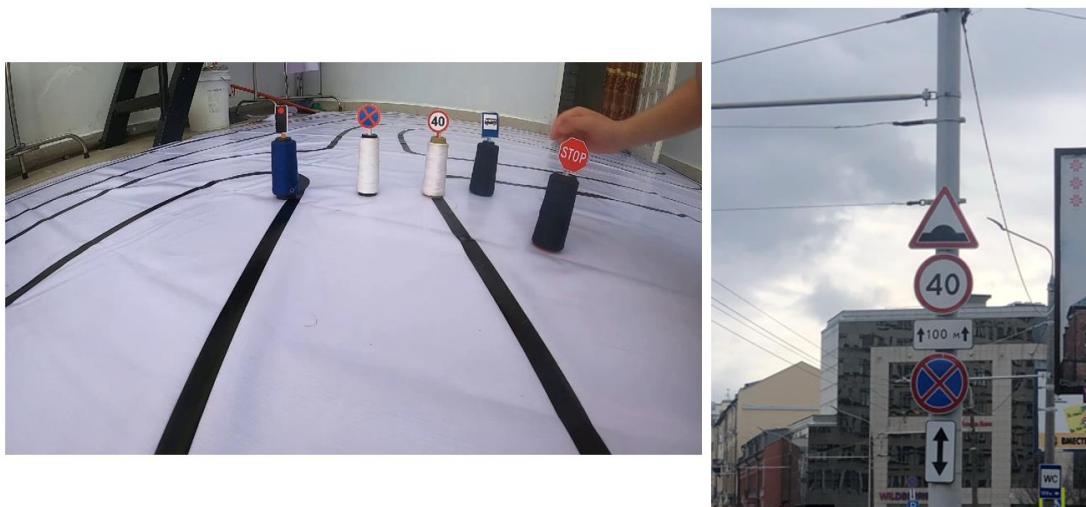
Bảng 2. Danh sách tín hiệu và biển báo giao thông

Cần phải huấn luyện và chạy thử nhiều lần để kiểm ra độ ổn định và chính xác của mô hình, phải xác định tất cả các trường hợp có thể xảy ra với vật thể ở vùng cần nhận dạng, lưu lại dữ liệu. Sau mỗi lần huấn luyện, ngoài việc kiểm tra các thông số, ta còn cần phải kiểm tra thực tế xem có trường hợp nào dữ liệu chưa đủ để mô hình học được, từ đó tiến hành thu thập thêm nhiều dữ liệu và làm giàu dữ liệu thêm nữa cho mô hình.



Hình 61. Quy trình huấn luyện mô hình

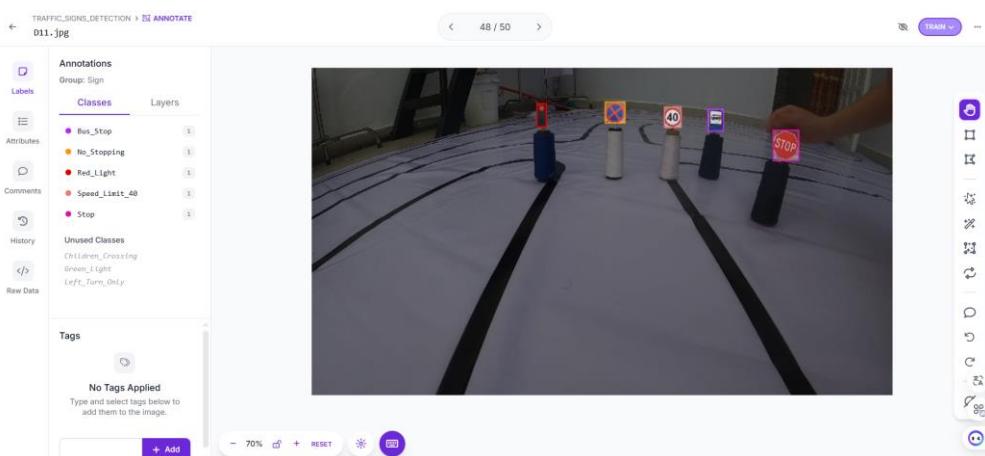
Quá trình thu thập dữ liệu được thực hiện nhằm xây dựng một tập ảnh đầu vào phong phú, phục vụ cho việc huấn luyện mô hình nhận diện biển báo giao thông. Tập dữ liệu bao gồm hàng nghìn hình ảnh được lấy từ các nguồn thực tế hoặc công khai trên Internet, đảm bảo tính đa dạng cao về góc nhìn, điều kiện ánh sáng cũng như kích thước và vị trí của các biển báo trong khung hình. Việc đảm bảo độ phong phú của dữ liệu là yếu tố quan trọng giúp mô hình học được các đặc trưng tổng quát, từ đó nâng cao khả năng nhận diện chính xác trong các tình huống thực tế sau này.



Hình 62. Ảnh dùng để huấn luyện model

Bên trái là hình được lấy từ góc nhìn camera của xe, bên phải là hình của các dự án công khai có sẵn trên Roboflow.

Sau khi thu thập dữ liệu, bước tiếp theo là gán nhãn cho các đối tượng trong ảnh nhằm phục vụ quá trình huấn luyện mô hình. Công cụ Roboflow được sử dụng để thực hiện việc này, cho phép đánh dấu chính xác từng biển báo giao thông xuất hiện trong mỗi bức ảnh bằng cách tạo các bounding box tương ứng với vị trí và loại biển báo. Sau khi hoàn tất việc gán nhãn, dữ liệu được xuất ra ở định dạng YOLO (.txt), trong đó mỗi dòng chứa thông tin về lớp đối tượng (class) và tọa độ khung giới hạn (bounding box) được chuẩn hóa theo kích thước ảnh. Việc sử dụng đúng định dạng giúp quá trình huấn luyện với mô hình YOLOv4-Tiny diễn ra suôn sẻ và hiệu quả.



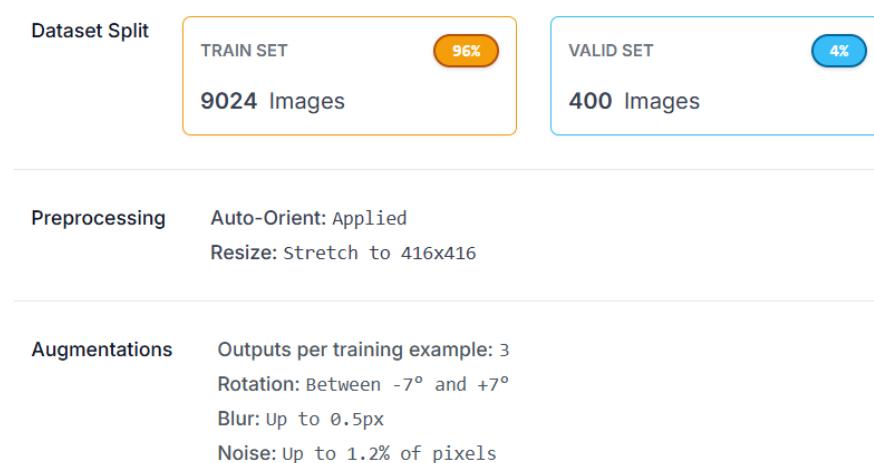
Hình 63. Ảnh sau khi gắn nhãn

Sau khi hoàn thành bước gán nhãn cho các ảnh trong tập dữ liệu, dữ liệu tiếp tục được tổ chức lại một cách có hệ thống nhằm phục vụ cho quá trình huấn luyện mô hình YOLOv4-Tiny. Tổng số ảnh ban đầu thu được là 3.408 ảnh, bao gồm các biển báo giao thông thuộc nhiều loại khác nhau. Tập dữ liệu này được chia thành hai phần:

- Tập huấn luyện (training set) gồm 3.008 ảnh. (Chưa tăng cường dữ liệu)
- Tập kiểm tra (validation set) gồm 400 ảnh.

Việc phân chia này tuân theo tỷ lệ phổ biến trong học máy, đảm bảo đủ dữ liệu cho mô hình học và kiểm tra hiệu quả tổng quát hóa.

Để tăng cường khả năng học của mô hình cũng như giảm thiểu hiện tượng overfitting, các kỹ thuật tăng cường dữ liệu (data augmentation) đã được áp dụng đối với tập huấn luyện. Cụ thể, các ảnh được xử lý qua nhiều phép biến đổi như xoay, làm mờ, nhiễu. Các phương pháp này giúp mô hình học được các đặc trưng phong phú hơn từ cùng một ảnh gốc, từ đó cải thiện đáng kể khả năng nhận diện các đối tượng trong điều kiện thực tế đa dạng.



Hình 64. Tổng quan tập dữ liệu

Sau khi áp dụng các kỹ thuật tăng cường dữ liệu, số lượng ảnh trong tập huấn luyện đã tăng lên gấp ba lần. Tập kiểm tra được giữ nguyên ở mức 400 ảnh để đảm bảo tính nhất quán trong đánh giá mô hình.

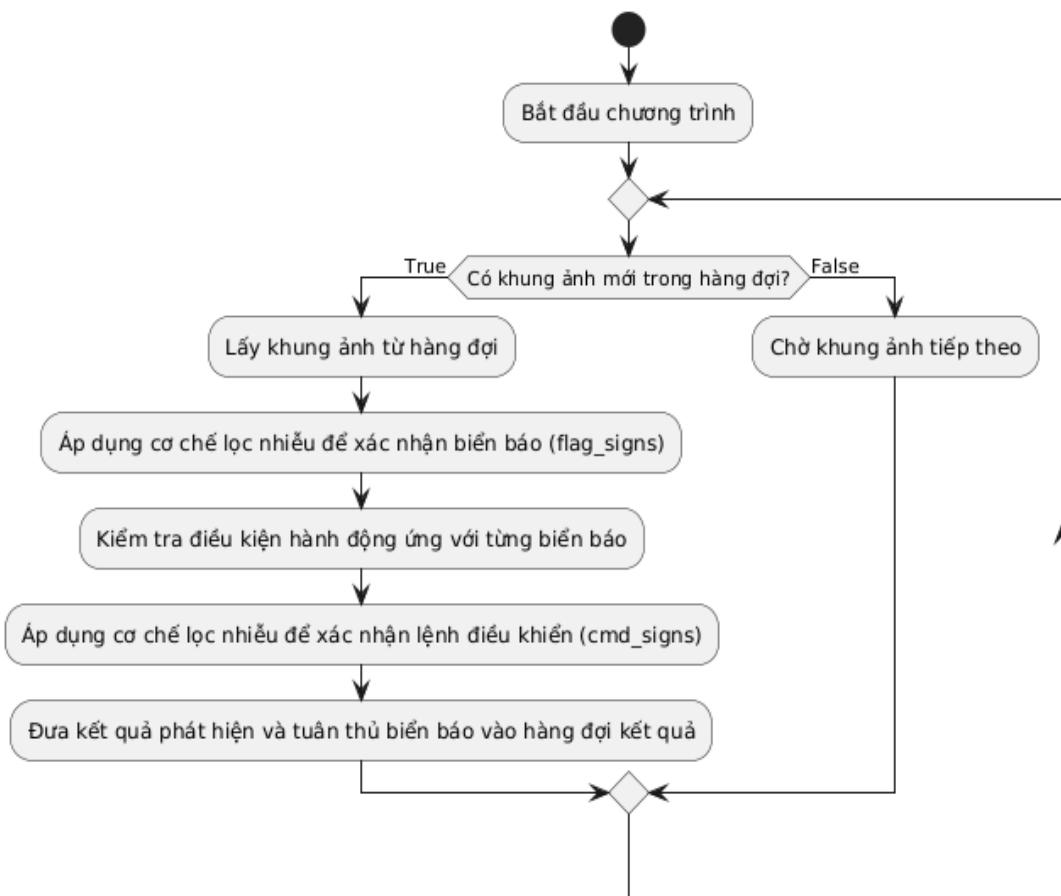
- Tập huấn luyện (training set) gồm 9.024 ảnh.

- Tập kiểm tra (validation set) gồm 400 ảnh.

Như vậy, tổng số ảnh đầu vào cho quá trình huấn luyện và đánh giá sau cùng là 9.424 ảnh. Việc tổ chức dữ liệu theo cách này không chỉ đảm bảo chất lượng đầu vào cho mô hình mà còn góp phần nâng cao hiệu suất và tính chính xác khi triển khai mô hình trong các hệ thống thực tế như xe tự hành hoặc camera giám sát thông minh.

Quá trình huấn luyện mô hình YOLOv4-Tiny được thực hiện trên nền tảng Google Colab với cấu hình batch size là 32, số lượng *max_batches* đặt ở mức 16.000, tương đương khoảng 57 epoch và mất khoảng 5 tiếng để huấn luyện. Việc huấn luyện trên Colab giúp tận dụng tài nguyên GPU miễn phí, rút ngắn thời gian xử lý và phù hợp với môi trường phát triển giới hạn phần cứng.

5.2.2. Giải thuật chương trình chính



Hình 65. Chương trình nhận diện biển báo

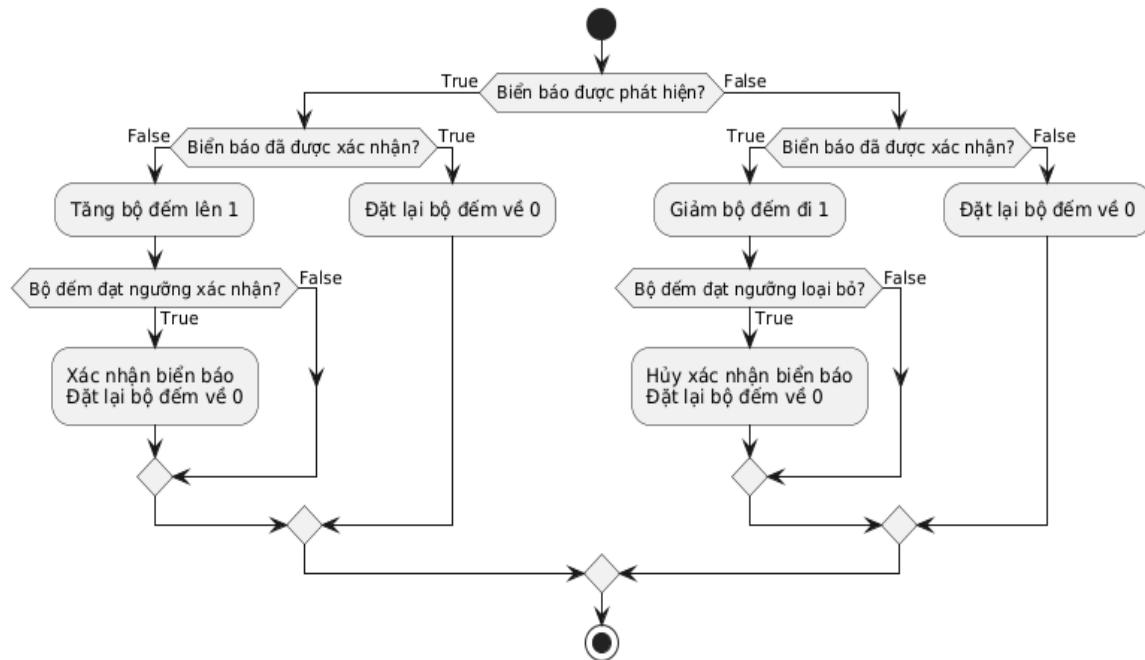
Lưu đồ trong hình mô tả quá trình xử lý chính của hệ thống nhận diện và tuân thủ biển báo giao thông trong xe tự hành mini. Toàn bộ chương trình vận hành theo cơ chế vòng lặp liên tục, bắt đầu từ nút khởi động chương trình (*Bắt đầu chương trình*). Ở mỗi vòng lặp, hệ thống đầu tiên kiểm tra xem có khung ảnh mới trong hàng đợi hay không. Nếu không có, chương trình tiếp tục chờ khung ảnh tiếp theo. Ngược lại, nếu có khung ảnh, hệ thống sẽ lấy khung ảnh ra để xử lý.

Bước đầu tiên là áp dụng cơ chế lọc nhiễu để xác nhận xem có biển báo nào thực sự xuất hiện trong ảnh hay không (*Áp dụng cơ chế lọc nhiễu để xác nhận biển báo*). Cơ chế lọc này sử dụng các thuật toán kiểm tra liên tục qua nhiều khung hình để đảm bảo rằng biển báo xuất hiện là ổn định, nhằm loại bỏ các phát hiện sai do nhiễu (sẽ được nói rõ ở phần sau). Sau đó, hệ thống sẽ kiểm tra điều kiện hành động đối với từng loại biển báo, ví dụ như vị trí, kích thước, độ tin cậy của đối tượng trong ảnh. Nếu thỏa mãn các điều kiện này, hệ thống sẽ tiếp tục áp dụng một cơ chế lọc nhiễu khác để xác nhận lệnh điều khiển tương ứng (*xác nhận lệnh điều khiển*), như “dừng lại”, “rẽ trái”,... Cuối cùng, sau khi biển báo đã được xác nhận và lệnh điều khiển đã được thiết lập rõ ràng, hệ thống sẽ đưa kết quả vào hàng đợi đầu ra (*Đưa kết quả phát hiện và tuân thủ biển báo vào hàng đợi kết quả*), để truyền cho module điều khiển hành vi của xe.

Toàn bộ quy trình này giúp đảm bảo rằng xe chỉ phản ứng với các biển báo hợp lệ và ổn định, tránh các hành vi sai lệch do phát hiện sai hoặc không chắc chắn trong môi trường thực tế nhiều biến động.

5.2.3. Giải thuật chương trình con

5.2.3.1. Chương trình lọc nhiễu nhận diện biển báo



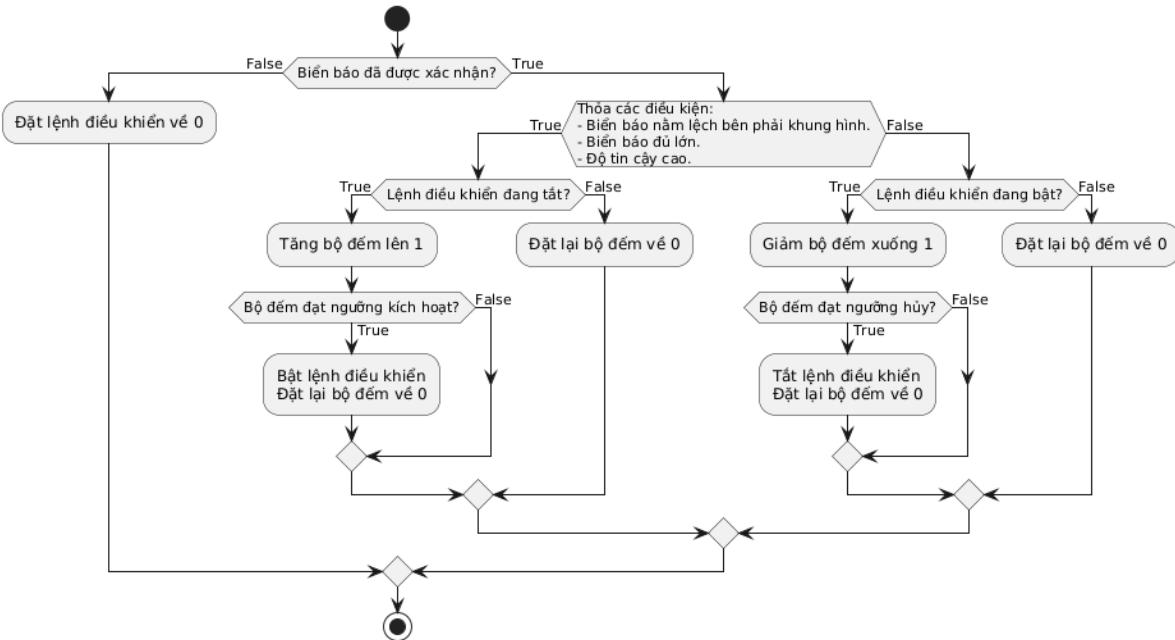
Hình 66. Chương trình lọc nhiễu nhận diện biển báo

Lưu đồ trên mô tả cơ chế lọc nhiễu để xác nhận hoặc loại bỏ một biển báo giao thông, giúp đảm bảo rằng hệ thống không phản ứng với các phát hiện sai lệch hoặc tạm thời. Quá trình vận hành dựa trên bộ đếm và cờ trạng thái (flag) được cập nhật liên tục theo từng khung hình (frame). Khi một biển báo được phát hiện trong ảnh, hệ thống kiểm tra xem biển báo đó đã được xác nhận trước đó chưa. Nếu chưa, bộ đếm tương ứng sẽ tăng lên một đơn vị. Khi bộ đếm đạt đến ngưỡng xác nhận (ví dụ: 8 frame liên tiếp đều phát hiện biển báo), hệ thống sẽ xác nhận sự hiện diện của biển báo và thiết lập cờ trạng thái *flag =1*. Điều này giúp tránh việc xác nhận quá sớm do các phát hiện nhất thời hoặc nhiễu.

Ngược lại, nếu trong một khung hình không phát hiện thấy biển báo, hệ thống cũng kiểm tra xem biển báo đó đã từng được xác nhận hay chưa. Nếu đã được xác nhận, bộ đếm sẽ giảm dần. Khi bộ đếm đạt đến một ngưỡng âm nhất định (ví dụ: -5), hệ thống sẽ hủy xác nhận biển báo (tức là thiết lập *flag =0*) và đưa bộ đếm về 0. Nếu biển báo chưa từng được xác nhận, bộ đếm cũng được đặt lại để tránh tích lũy sai lệch.

Cơ chế lọc này hoạt động như một bộ lọc thời gian, giúp hệ thống chỉ phản ứng với những biển báo thực sự ổn định và xuất hiện liên tục trong nhiều khung hình, qua đó tăng tính tin cậy cho hệ thống nhận diện và điều khiển trên xe tự hành.

5.2.3.2. Chương trình lọc nhiễu chấp hành biển báo



Hình 67. Chương trình lọc nhiễu chấp hành biển báo

Lưu đồ trên minh họa cơ chế lọc nhiễu lần hai nhằm đưa ra quyết định thực thi lệnh điều khiển dựa trên biển báo giao thông đã được xác nhận. Cơ chế này đóng vai trò quan trọng trong hệ thống xe tự hành, đảm bảo rằng các lệnh điều khiển chỉ được kích hoạt khi biển báo thực sự rõ ràng, ổn định và đáng tin cậy.

Quá trình bắt đầu khi một biển báo đã được xác nhận từ bước lọc nhiễu đầu tiên (flag bật). Hệ thống sau đó kiểm tra xem biển báo có thỏa các điều kiện không gian như: nằm lệch bên phải khung hình, có kích thước đủ lớn (bounding box đủ cao >80 pixel), và điểm tin cậy (confidence > 0.6) vượt ngưỡng. Nếu các điều kiện này được đáp ứng, bộ đếm sẽ tăng dần theo mỗi khung hình liên tiếp. Khi bộ đếm đạt đến ngưỡng kích hoạt (ví dụ 10), hệ thống sẽ bật cờ $cmd = 1$ – tức là bắt đầu thực thi hành động điều khiển tương ứng với biển báo (như rẽ trái, dừng lại...).

Tuy nhiên, nếu ở các khung hình sau, biển báo không còn thỏa điều kiện hoặc không còn xuất hiện, hệ thống tiếp tục theo dõi bằng cách giảm bộ đếm. Nếu bộ

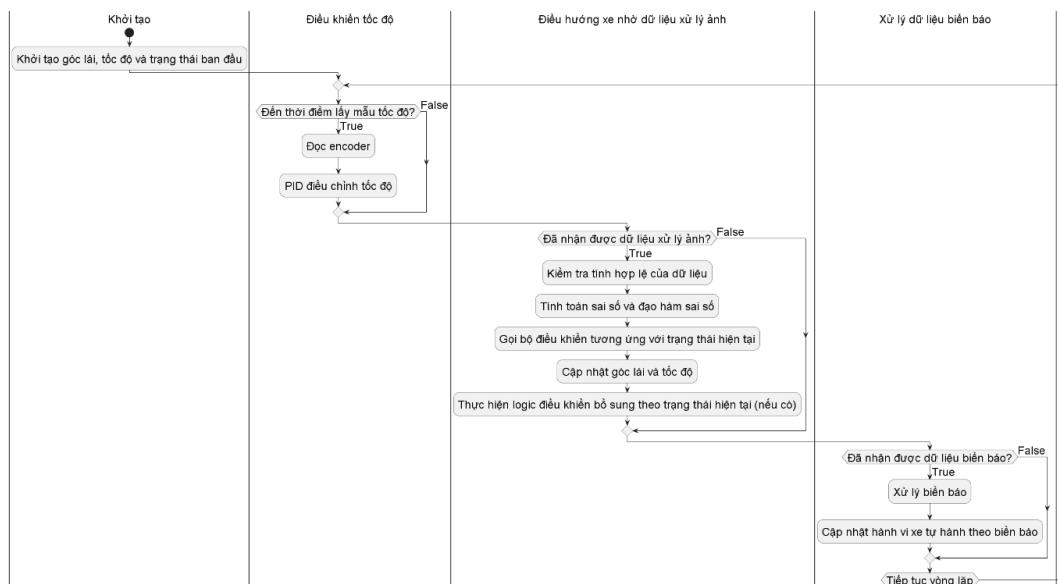
đếm giảm liên tục đến ngưỡng hủy bỏ (ví dụ -60), hệ thống sẽ tắt cờ $cmd = 0$, ngừng thực hiện lệnh điều khiển và đặt lại bộ đếm.

Cơ chế này giúp đảm bảo rằng lệnh điều khiển chỉ được thực thi nếu biến báo được phát hiện ổn định trong thời gian đủ dài và không bị ảnh hưởng bởi nhiễu hoặc sai lệch nhất thời. Sự kết hợp giữa xác nhận (flag) và thực thi (cmd) tạo nên một quy trình kiểm soát hai bước vừa chính xác vừa an toàn cho hệ thống xe tự hành.

5.3. Giải thuật điều khiển xe tự hành

5.3.1. Giải thuật chương trình chính

Giải thuật chương trình chính của hệ thống điều khiển robot được thể hiện trực quan qua sơ đồ sau:



Hình 68. Lưu đồ giải thuật chương trình chính điều khiển xe tự hành

Chương trình chính của hệ thống điều khiển robot được tổ chức dưới dạng một vòng lặp điều khiển. Ngay khi hệ thống khởi động, các thông số điều khiển như góc lái, tốc độ và trạng thái ban đầu của robot sẽ được khởi tạo để đảm bảo hệ thống có một trạng thái ổn định và xác định trước khi bước vào chu trình vận hành. Góc lái của động cơ servo đánh lái và góc xoay camera, tốc độ động cơ sẽ được đưa về không.

Trong mỗi chu kỳ lặp, hệ thống trước tiên sẽ kiểm tra điều kiện liên quan đến thời điểm lấy mẫu tốc độ. Nếu điều kiện được thỏa mãn, dữ liệu từ encoder sẽ được thu thập và sử dụng trong bộ điều khiển phản hồi nhằm tính toán và cập nhật tốc độ động cơ, bảo đảm rằng robot di chuyển ổn định theo tốc độ mong muốn.

Tiếp theo, hệ thống kiểm tra dữ liệu xử lý ảnh làn đường mới nào được gửi từ Jetson TX-2 hay không. Nếu có, quá trình xử lý sẽ tiếp tục với việc sàng lọc và kiểm tra tính hợp lệ của dữ liệu, loại bỏ các trường hợp bất thường hoặc không đáng tin cậy. Sau đó, các đại lượng điều khiển như sai số và đạo hàm sai số sẽ được tính toán dựa trên dữ liệu đã kiểm tra. Đây là các đầu vào của bộ điều khiển tương ứng với trạng thái hiện tại của robot (ví dụ: bấm làn, rẽ trái/phải, đi thẳng...). Kết quả điều khiển sẽ được sử dụng để cập nhật giá trị góc lái và tốc độ. Ngoài ra, hệ thống có thể thực hiện thêm một số logic điều khiển bổ sung tùy theo từng trạng thái cụ thể, chẳng hạn như điều chỉnh góc quay camera trong trạng thái bám theo làn đường.

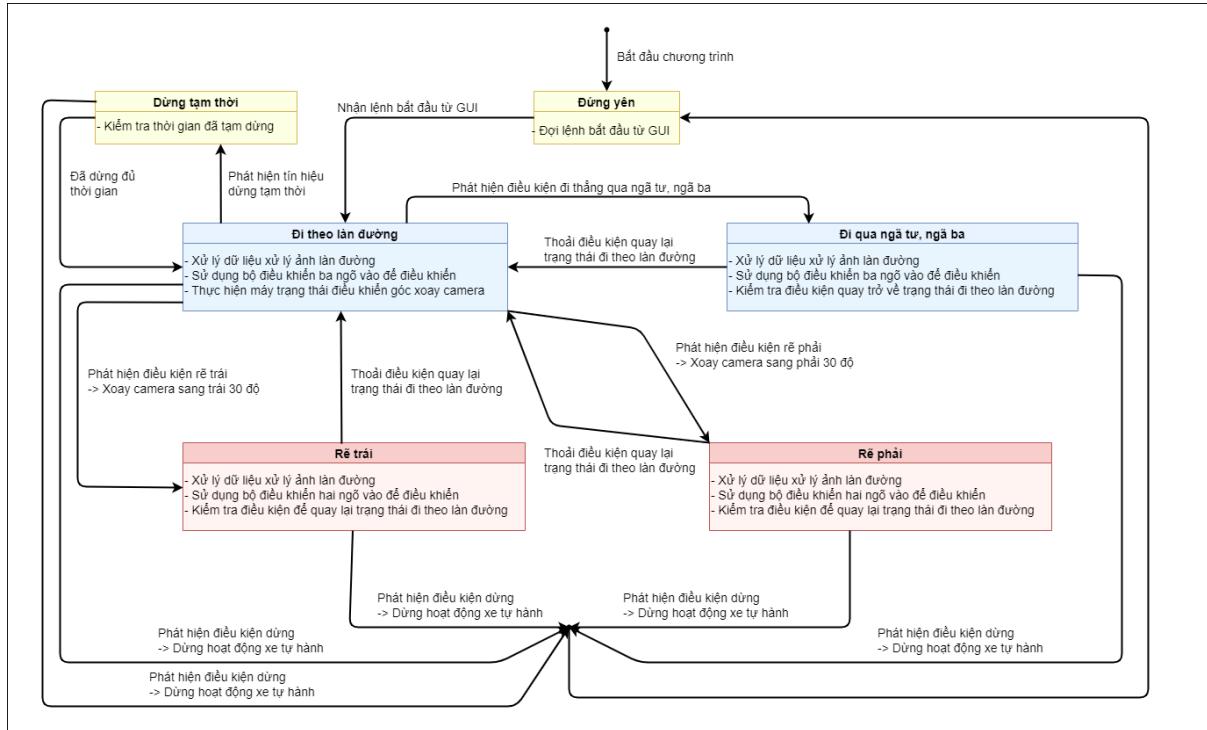
Cuối cùng, hệ thống sẽ kiểm tra xem có dữ liệu nhận dạng và chấp hành biển báo giao thông mới nào được gửi về hay không. Nếu có, các biển báo như đèn đỏ, điểm dừng xe buýt, biển cấm rẽ hoặc giới hạn tốc độ sẽ được xử lý tương ứng để cập nhật lại hành vi điều khiển của xe tự hành.

5.3.2. Mô hình máy trạng thái điều khiển xe tự hành

Chương trình điều khiển mô hình xe tự hành được xây dựng dựa trên mô hình máy trạng thái với sáu trạng thái lần lượt là:

- Đứng yên
- Đi theo làn đường
- Đi qua ngã tư, ngã ba
- Rẽ trái
- Rẽ phải
- Dừng tạm thời

Tại mỗi thời điểm, mô hình chỉ tồn tại trong đúng một trạng thái duy nhất, và có thể chuyển đổi giữa các trạng thái dựa trên dữ liệu xử lý ảnh hoặc tín hiệu từ biển báo giao thông.



Hình 69. Mô hình máy trạng thái điều khiển xe tự hành

Xe tự hành khởi đầu tại trạng thái “Đứng yên”, chờ lệnh bắt đầu từ GUI. Khi lệnh được gửi đến, hệ thống sẽ chuyển sang trạng thái “Đi theo làn đường”, nơi mô hình xử lý dữ liệu đầu vào từ camera để xác định vị trí và hướng đi theo làn đường. Tại đây, mô hình sử dụng bộ điều khiển ba ngõ vào để điều chỉnh góc lái và tốc độ, đồng thời thực hiện logic điều khiển bổ sung thông qua máy trạng thái điều khiển góc xoay camera (bộ điều khiển ba gõ vào và máy trạng thái điều khiển góc xoay camera sẽ được nêu rõ ở các phần sau). Ngoài ra, khi phát hiện các biển báo yêu cầu dừng như đèn đỏ hoặc trạm dừng, mô hình sẽ chuyển sang trạng thái “Đừng tạm thời”, chờ trong một khoảng thời gian định trước. Sau khi hết thời gian, xe tự hành tự động tiếp tục di chuyển.

Khi phát hiện điều kiện rẽ trái hoặc rẽ phải, mô hình sẽ chuyển sang trạng thái “Rẽ trái” hoặc “Rẽ phải”. Trong hai trạng thái này, xe tự hành sử dụng bộ điều khiển hai ngõ vào, tập trung vào việc bám theo một phía làn duy nhất. Sau khi thỏa

điều kiện, xe tự hành sẽ quay lại trạng thái “Đi theo làn đường”. (Bộ điều khiển hai ngõ vào và điều kiện để xe tự hành quay lại trạng thái “Đi theo làn đường” sẽ được nói rõ ở các phần sau)

Khi gặp ngã ba hoặc ngã tư, nếu không rẽ mà tiếp tục đi thẳng, xe tự hành sẽ chuyển sang trạng thái “Đi qua ngã tư, ngã ba”. Trong trạng thái này, mô hình vẫn sử dụng bộ điều khiển ba ngõ vào nhưng không thực hiện điều khiển góc quay camera. Khi đã băng qua giao lộ và thấy lại làn đường phía trước, xe tự hành quay trở lại trạng thái “Đi theo làn đường”.

Bất kỳ lúc nào, mô hình cũng có thể nhận lệnh dừng từ GUI hoặc qua biến báo, hệ thống sẽ chuyển về trạng thái “Đứng yên” và chờ lệnh tiếp theo.

5.3.3. Giải thuật chương trình con

5.3.3.1. Chương trình điều khiển góc của động cơ Servo

Hệ thống sử dụng động cơ Servo MG996R để thực hiện điều khiển góc đánh lái và điều khiển góc xoay camera. Động cơ này được điều khiển thông qua xung PWM với tần số 50 Hz, tương ứng với chu kỳ 20 ms. Độ rộng xung (t_{on} ms) trong mỗi chu kỳ sẽ xác định góc quay của servo. Theo tài liệu kỹ thuật của nhà sản xuất, $t_{on} = 2.5$ ms tương ứng với góc 0° , và $t_{on} = 12.5$ ms tương ứng với góc 180° .

Để thuận tiện trong quá trình điều khiển, hệ quy chiếu được quy ước tại góc servo = 90° , tương ứng với góc 0° tương đối trong hệ thống. Từ đó, góc điều khiển X được định nghĩa là độ lệch tương đối so với mốc 90° , nhóm đã giới hạn lại độ lớn tối đa góc xoay với $|X| \leq 45^\circ$.

Góc X có thể âm (camera hoặc bánh xe sẽ xoay sang phải) hoặc dương (camera hoặc bánh xe sẽ xoay sang trái). Tổng góc servo thực tế được tính là:

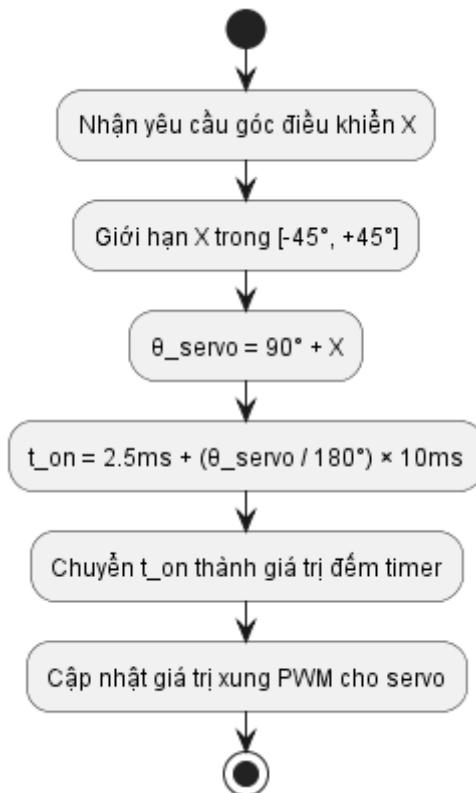
$$\theta_{servo} = 90^\circ + X \text{ (độ)}$$

Từ θ_{servo} này, chương trình sẽ nội suy tuyến tính để tính toán độ rộng xung PWM cần thiết theo công thức:

$$t_{on} = 2.5 + (\theta_{servo} / 180^\circ) \times 10 \text{ (ms)}$$

Có được độ rộng xung PWM cần thiết, chương trình quy đổi giá trị này thành thông số tương ứng với bộ Timer đã cài đặt sẵn của vi điều khiển STM32F407 để phát xung PWM phù hợp, điều khiển được góc mong muốn của động cơ Servo.

Quá trình có thể tóm tắt qua lưu đồ giải thuật sau:



Hình 70. Lưu đồ giải thuật điều khiển góc xoay động cơ Servo

5.3.3.2. Chương trình đọc và điều khiển tốc độ động cơ

Chương trình điều khiển tốc độ của động cơ DC bao gồm hai giai đoạn chính: đọc tốc độ thực tế từ encoder và điều khiển tốc độ bằng bộ điều khiển PID. Hai giai đoạn này được thực hiện định kỳ với tần số 50 Hz (chu kỳ 20 ms). Sóng PWM điều khiển động cơ có tần số 10 kHz để đảm bảo phản hồi nhanh và ổn định.

a. Đọc tốc độ từ encoder

Động cơ DC sử dụng encoder với 11 xung mỗi vòng quay. Khi cấu hình ở chế độ x4 (bắt cả 4 cạnh), tổng số xung mỗi vòng là:

$$EncoderPulsePerRev = 11 \times 4 = 44$$

Hệ thống truyền động gồm một hộp số có tỉ số truyền:

$$GearMotorRatio = 171$$

Động cơ DC và trục xoay bánh xe được kết nối bởi một cặp bánh răng, trong đó bánh chủ động có 54 răng và bánh bị động có 30 răng:

$$GearWheelRatio = 54 / 30 = 1.8$$

Từ đó, tỉ số truyền tổng là:

$$TotalGearRatio = GearMotorRatio \times GearWheelRatio = 171 \times 1.8 = 307.8$$

Đường kính bánh xe:

$$WheelDiameter = 6.2(cm)$$

Chu kỳ lấy mẫu tốc độ:

$$SampleTime = 0.02(s)$$

Tại mỗi chu kỳ, số xung encoder thay đổi là ΔN . Từ đó, ta tính:

$$n_{motor} = \frac{\Delta N}{EncoderPulsePerRev} = \frac{\Delta N}{44} \text{ và } rps_{motor} = \frac{n_{motor}}{SampleTime} = \frac{\Delta N}{44 \times 0.02}$$

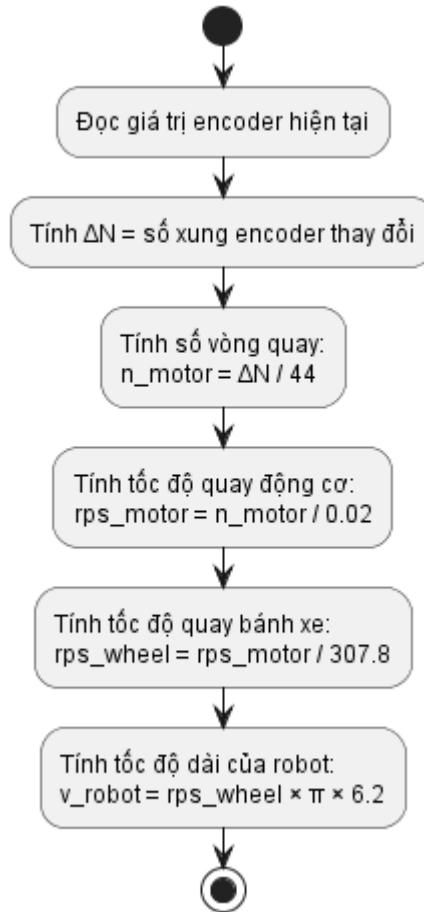
Tốc độ quay của bánh xe:

$$rps_{wheel} = \frac{rps_{motor}}{TotalGearRatio} = \frac{rps_{motor}}{307.8}$$

Tốc độ dài của xe tự hành (cm/s):

$$v_{robot} = rps_{wheel} \times \pi \times WheelDiameter = rps_{wheel} \times \pi \times 6.2$$

Quá trình trên có thể được thể hiện thông qua lưu đồ giải thuật sau:



Hình 71. Lưu đồ giải thuật đọc kết quả encoder.

b. Điều khiển tốc độ bằng bộ điều khiển PID

Khi muốn xe tự hành di chuyển với tốc độ dài v_{set} (cm/s), chương trình sẽ quy đổi về tốc độ quay mong muốn của động cơ:

$$rps_{set} = \left(\frac{v_{set}}{\pi \times \text{WheelDiameter}} \right) \times \text{TotalGearRatio} = \left(\frac{v_{set}}{\pi \times 6.2} \right) \times 307.8$$

Bộ điều khiển PID được sử dụng ở dạng rời rạc, với sai số tốc độ tại thời điểm k :

$$e_k = rps_{set} - rps_{motor}$$

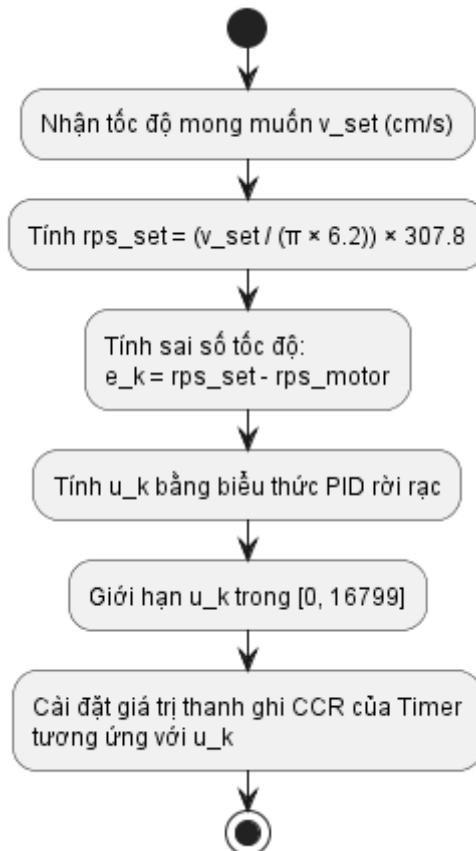
Giá trị điều khiển đầu ra được tính theo công thức:

$$u_k = u_{k-1} + K_p(e_k - e_{k-1}) + K_i(e_k + e_{k-1}) + K_d(e_k - 2e_{k-1} + e_{k-2})$$

Kết quả u_k sẽ được giới hạn trong miền: $u_k \in [u_{\min}, u_{\max}]$

Với $u_{\min} = 0$ và $u_{\max} = 16799$. Đây là hai giá trị thấp nhất và cao nhất của biến đếm Timer phụ trách xuất xung PWM điều khiển động cơ.

Quá trình trên có thể được thể hiện thông qua lưu đồ giải thuật sau:



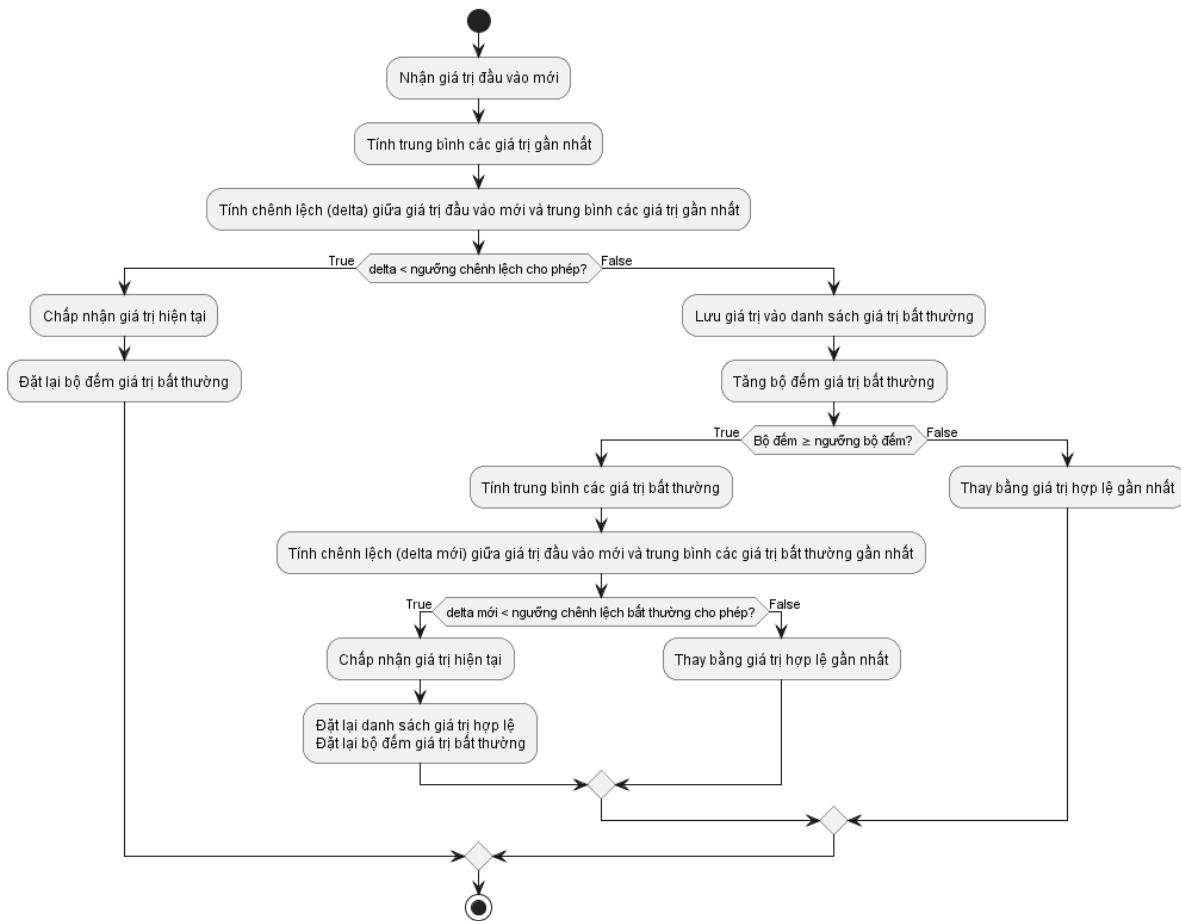
Hình 72. Lưu đồ giải thuật điều khiển tốc độ bằng bộ điều khiển PID rời rạc

Qua quá trình thực nghiệm, nhóm đã tinh chỉnh được các hệ sau K_p , K_i , K_d hoạt động hiệu quả như sau: $K_p = 1500$, $K_i = 3000$, $K_d = 300$.

5.3.3.3. Chương trình phát hiện các giá trị đầu vào bất thường

Trong quá trình điều khiển xe tự hành dựa trên dữ liệu xử lý ảnh, các sai số được trích xuất như tọa độ trung tâm làn đường, biên trái/phải và góc nghiêng có thể xuất hiện nhiều hoặc giá trị không đáng tin cậy do điều kiện ánh sáng, vật cản, hoặc lỗi xử lý. Để đảm bảo độ ổn định và độ tin cậy cho các đầu vào của bộ điều khiển, hệ thống triển khai một cơ chế phát hiện và xử lý các giá trị bất thường.

Chương trình có thể được thể hiện qua lưu đồ giải thuật sau:



Hình 73. Lưu đồ giải thuật phát hiện giá trị bất thường

Mỗi khi nhận được một giá trị đầu vào mới, hệ thống sẽ tính trung bình của một dãy các giá trị gần nhất đã được ghi nhận trước đó. Giá trị hiện tại sau đó được so sánh với trung bình này. Nếu độ chênh lệch giữa hai giá trị nhỏ hơn một ngưỡng cho phép được cấu hình từ trước, hệ thống sẽ chấp nhận giá trị hiện tại là hợp lệ và đặt lại bộ đếm số lần xuất hiện giá trị bất thường.

Trường hợp độ chênh lệch vượt quá ngưỡng cho phép, hệ thống tạm thời xem đây là một giá trị bất thường và đưa nó vào danh sách lưu trữ riêng biệt, đồng thời tăng bộ đếm theo dõi số lần liên tiếp xuất hiện giá trị bất thường. Nếu số lượng giá trị bất thường liên tiếp này vẫn nằm dưới giới hạn cho phép, hệ thống sẽ không chấp nhận giá trị hiện tại mà thay thế nó bằng giá trị hợp lệ gần nhất trước đó.

Tuy nhiên, nếu bộ đếm vượt quá ngưỡng được thiết lập, hệ thống sẽ đánh giá lại tính ổn định của chính các giá trị bất thường gần nhất. Cụ thể, hệ thống sẽ tính trung bình của các giá trị bất thường đã lưu, rồi so sánh giá trị đầu vào hiện tại

với trung bình này. Nếu độ chênh lệch nhỏ hơn ngưỡng chênh lệch thứ hai dành riêng cho nhóm giá trị bất thường, hệ thống xem đây là tín hiệu thay đổi hợp lệ và sẽ chấp nhận giá trị hiện tại, đồng thời đặt lại danh sách lưu và bộ đếm. Ngược lại, nếu chênh lệch vẫn quá lớn, giá trị đầu vào vẫn bị loại bỏ và hệ thống tiếp tục sử dụng lại giá trị hợp lệ gần nhất để duy trì ổn định cho bộ điều khiển.

5.3.3.4. Chương trình xoay camera

Trong trạng thái "Đi theo làn đường", hệ thống sẽ kích hoạt một chương trình con phụ trách điều khiển góc xoay của camera. Mục tiêu chính là đảm bảo rằng camera luôn duy trì góc nhìn phù hợp để quan sát đồng thời được cả hai vạch kẻ làn đường, từ đó đảm bảo độ tin cậy của dữ liệu đầu vào cho hệ thống điều khiển.

Chương trình hoạt động dưới dạng máy trạng thái hữu hạn, gồm tổng cộng năm trạng thái góc quay camera như sau:

- Nhìn thẳng (0°)
- Nhìn trái 30°
- Nhìn trái 45°
- Nhìn phải 30°
- Nhìn phải 45°

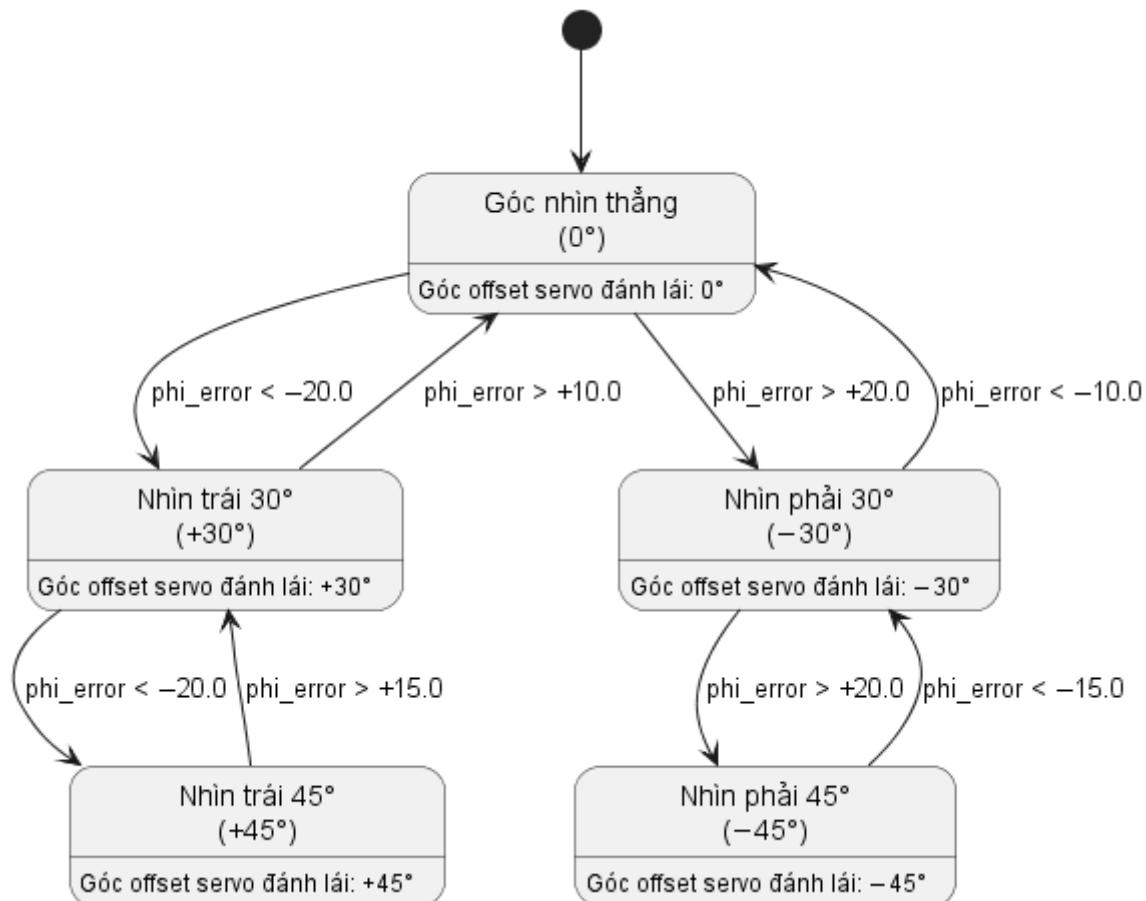
Quá trình chuyển đổi giữa các trạng thái được quyết định dựa trên giá trị sai số góc nghiêng (phi_error) của làn đường thu được từ xử lý ảnh. Tại mỗi chu kỳ điều khiển, sau khi hệ thống đã xử lý dữ liệu ảnh và tính toán xong góc lái, chương trình sẽ sử dụng phi_error để xác định có cần thay đổi trạng thái xoay camera hay không.

Mỗi trạng thái đều có các ngưỡng điều kiện cụ thể để chuyển sang trạng thái kế tiếp hoặc trở về trạng thái trước đó. Các ngưỡng này được xác định thông qua quá trình thực nghiệm, đảm bảo phù hợp với biến thiên thực tế của góc nhìn làn đường.

Ngoài ra, mỗi trạng thái quay camera còn được gán kèm một giá trị góc offset của động cơ Servo đánh lái, nhằm bù trừ cho sự sai lệch giữa phương quan sát của

camera và phương chuyển động thực tế của xe. Góc offset này sẽ được cộng thêm vào kết quả đầu ra của bộ điều khiển góc lái để đạt được hướng đánh lái chính xác.

Cách hoạt động của chương trình con xoay camera có thể được biểu diễn qua lưu đồ máy trạng thái sau:

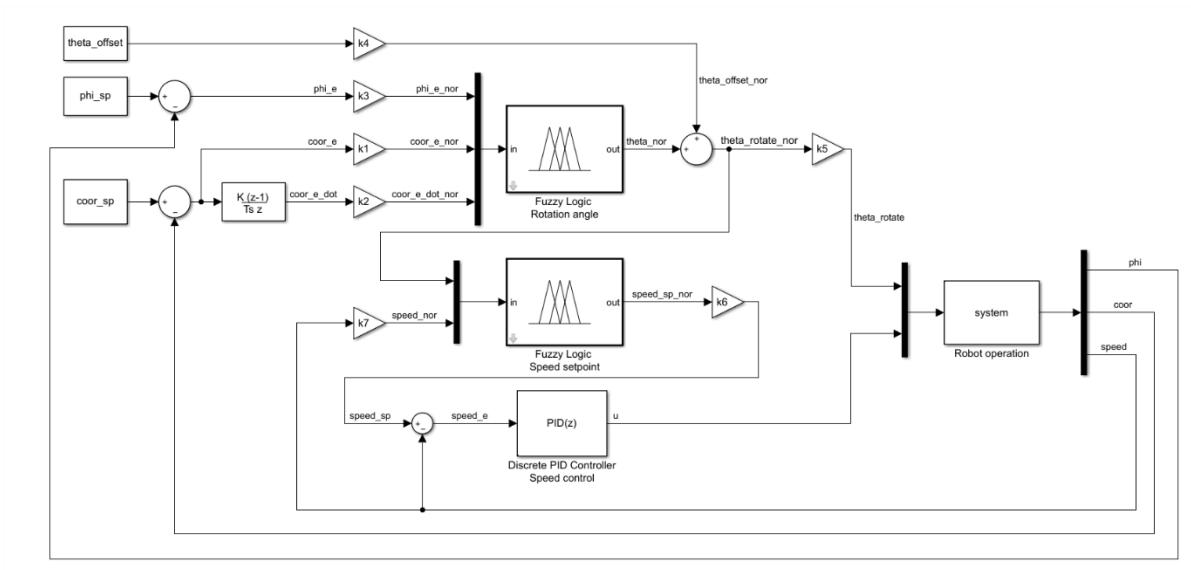


Hình 74. Lưu đồ máy trạng thái điều khiển góc quay camera

5.3.3.5. Chương trình điều khiển di chuyển theo làn đường

a. Phương pháp điều khiển

Chương trình điều khiển di chuyển theo làn đường được thiết kế với cấu trúc ba tầng điều khiển, hoạt động phối hợp để đảm bảo xe luôn duy trì hướng đi ổn định, song song với làn đường, đồng thời di chuyển ở tốc độ phù hợp với độ cong của làn đường.



Hình 75. Sơ đồ chương trình điều khiển di chuyển theo làn đường

Các kí hiệu có trong sơ đồ trên:

Tên gọi	Ý nghĩa
theta_offset	Giá trị góc offset của động cơ Servo đánh lái
phi	Độ nghiêng làn đường
phi_sp	Giá trị đặt của độ nghiêng làn đường
phi_e	Sai số độ nghiêng làn đường
phi_e_nor	Sai số độ nghiêng làn đường đã chuẩn hóa
coor	Giá trị điểm trung tâm làn đường
coor_sp	Giá trị đặt của điểm trung tâm làn đường
coor_e	Sai số tọa độ điểm trung tâm làn đường
coor_e_nor	Sai số tọa độ điểm trung tâm làn đường đã chuẩn hóa
coor_e_dot	Tốc độ biến thiên Sai số tọa độ điểm trung tâm làn đường

coor_e_dot_nor	Tốc độ biến thiên Sai số tọa độ điểm trung tâm làn đường đã chuẩn hóa
speed	Tốc độ dài của xe tự hành
speed_sp	Giá trị đặt của tốc độ dài xe tự hành
speed_e	Sai số tốc độ dài của xe tự hành
speed_nor	Tốc độ dài của xe tự hành đã chuẩn hóa
speed_sp_nor	Giá trị đặt của tốc độ dài xe tự hành đã chuẩn hóa
theta_nor	Kết quả giá trị góc lái của bộ điều khiển đã chuẩn hóa
theta_rotate	Kết quả giá trị góc lái của bộ điều khiển sau không cộng một lượng góc offset
theta_rotate_nor	Kết quả giá trị góc lái của bộ điều khiển sau không cộng một lượng góc offset đã chuẩn hóa
k1, k2, k3, k4, k7	Hệ số chuẩn hóa
k5, k6	Hệ số phi chuẩn hóa

Bảng 3. Ý nghĩa các kí hiệu

Tầng đầu tiên là bộ điều khiển mờ ba ngõ vào, có nhiệm vụ tính toán góc đánh lái phù hợp cho xe. Các ngõ vào của bộ điều khiển này bao gồm:

- Sai số vị trí của điểm trung tâm làn đường so với điểm đặt lý tưởng (coor_e).
- Đạo hàm sai số vị trí (coor_e_dot), phản ánh xu hướng thay đổi vị trí trung tâm.
- Sai số góc nghiêng làn đường (phi_e), phản ánh độ lệch giữa hướng chuyển động của xe và trực dọc của làn đường.

Các giá trị trên được chuẩn hóa (norm) để đưa vào bộ điều khiển mờ. Ngõ ra là giá trị góc đánh lái đã chuẩn hóa (theta_nor). Sau đó, góc này được cộng thêm một lượng offset (theta_offset) phụ thuộc vào trạng thái xoay hiện tại của camera. Giá trị góc offset này được xác định thực nghiệm nhằm bù trừ sai lệch giữa hướng

nhìn của camera và hướng thực tế của xe. Kết quả cuối cùng (`theta_rotate`) sẽ được phi chuẩn hóa để điều khiển trực tiếp động cơ servo đánh lái.

Tầng thứ hai là bộ điều khiển mờ hai ngõ vào điều khiển tốc độ đặt (`speed_sp`). Bộ điều khiển này lấy góc đánh lái hiện tại và tốc độ thực tế của xe làm đầu vào. Với mục đích đảm bảo tính an toàn và ổn định khi xe vào cua hay rẽ gấp. Nguyên tắc hoạt động là:

- Khi xe đang đánh lái gấp (góc lớn), bộ điều khiển sẽ giảm tốc độ đặt.
- Khi xe đang chạy ổn định ở hướng thẳng (góc lái nhỏ), tốc độ đặt sẽ được tăng lên dần về mức tối đa.

Ngõ ra từ bộ điều khiển tốc độ đặt (`speed_sp`) được đưa vào bộ điều khiển PID rồi rạc, kết hợp với giá trị tốc độ thực tế để tính toán tín hiệu điều khiển (u) điều khiển động cơ DC thông qua xung PWM. Bộ điều khiển này có nhiệm vụ điều chỉnh tốc độ động cơ sao cho bám sát tốc độ mong muốn được đặt ra từ tầng fuzzy.

b. Thiết kế bộ điều khiển mờ điều chỉnh góc lái của động cơ servo đánh lái dựa vào kinh nghiệm chuyên gia

Các giá trị tín hiệu đặt được tìm dựa vào việc đặt mô hình xe tự hành ở ví trí chuẩn (ở trung tâm làn đường và dọc song song theo làn đường) và tiến hành lấy kết quả xử lý ảnh.

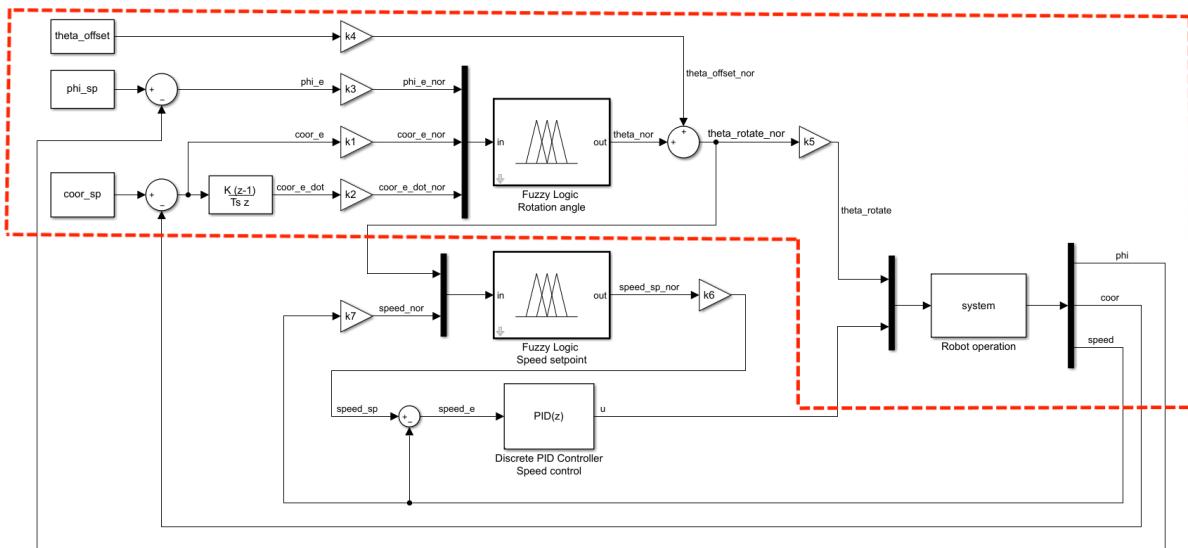
Tín hiệu đặt:

- `coor_sp` = 668 (gần ở giữa của bề ngang khung ảnh có độ phân giải 1280x720)
- `phi_sp` = 4.6 ° (do ví trí đặt thấu kính của camera không nằm ở trung tâm dọc theo mô hình xe)

Các biến vào, ra của bộ điều khiển:

- Ba biến vào: sai số độ nghiêng làn đường, sai số và tốc độ biến thiên sai số tọa độ
- Biến ra: góc lái của động cơ servo đánh lái

Sơ đồ khái niệm hệ thống điều khiển:



Hình 76. Sơ đồ khái bô điều khiển mờ điều chỉnh góc lái (phần được khoanh đỏ)

Chuẩn hóa biến vào, ra của bộ điều khiển:

- Sai số tọa độ trung tâm làn đường (đơn vị pixel):

$$coor_e = coor_sp - coor \Leftrightarrow 400 \leq coor_e \leq 400 \Leftrightarrow k_1 = 1/400$$
- Biến thiên sai số tọa độ trung tâm làn đường (đơn vị pixel/s):

$$3000 \leq coor_e_dot \leq 3000 \Leftrightarrow k_2 = 1/3000$$
- Sai số độ nghiêng của làn đường (đơn vị độ):

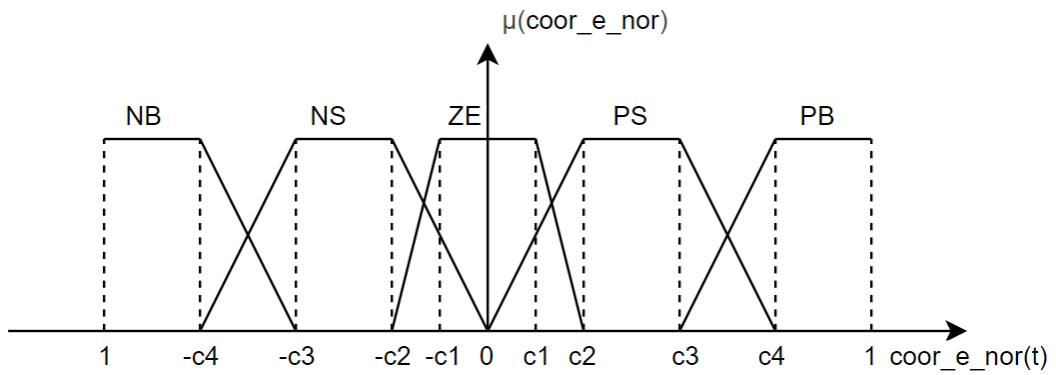
$$phi_e = phi_sp - coor \Leftrightarrow 45 \leq phi_e \leq 45 \Leftrightarrow k_3 = 1/45$$
- Góc lái của động cơ servo đánh lái (đơn vị độ):

$$-45 \leq theta_rotate \leq 45 \Leftrightarrow k_5 = 45$$
- Góc offset của động cơ servo đánh lái (đơn vị độ):

$$-45 \leq theta_offset \leq 45 \Leftrightarrow k_4 = 1/45$$

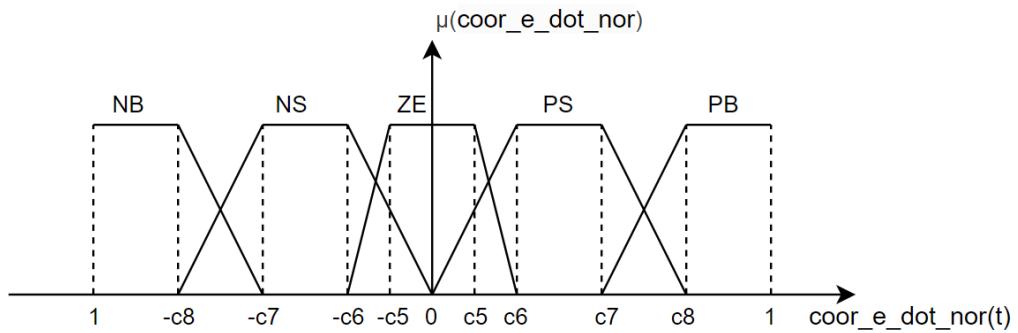
Định nghĩa các giá trị ngôn ngữ cho các biến vào, ra:

- Sai số tọa độ trung tâm coor_e_nor được định nghĩa bởi 5 giá trị ngôn ngữ:
NB, NS, ZE, PS, PB



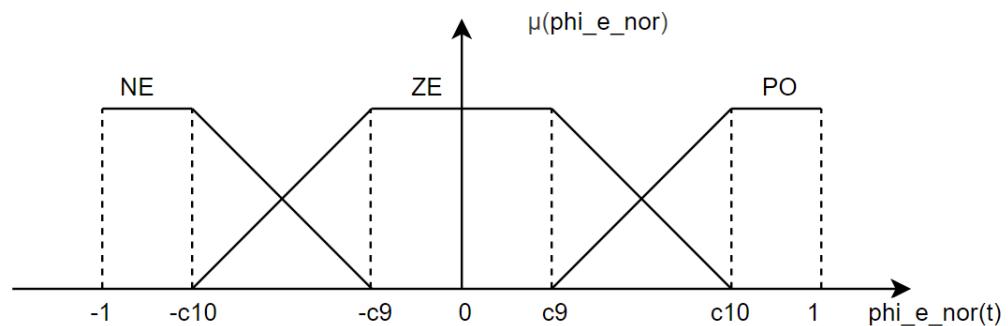
Hình 77. Định nghĩa giá trị ngôn ngữ cho sai số tọa độ điểm trung tâm làn đường

- Tốc độ biến thiên sai số tọa độ trung tâm coor_e_dot_nor được định nghĩa bởi 5 giá trị ngôn ngữ: NB, NS, ZE, PS, PB



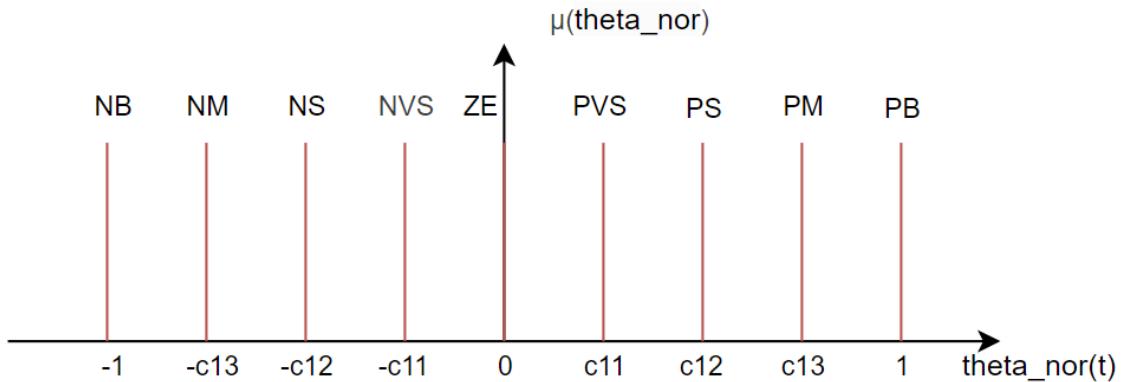
Hình 78. Định nghĩa giá trị ngôn ngữ cho tốc độ biến thiên sai số tọa độ điểm trung tâm làn đường

- Sai số độ nghiêng của làn đường được định nghĩa bởi 3 giá trị ngôn ngữ: NE, ZE, PO



Hình 79. Định nghĩa giá trị ngôn ngữ cho sai số độ nghiêng làn đường

- Kết quả giá trị góc lái của bộ điều khiển được định nghĩa bởi 9 giá trị ngôn ngữ: NB, NM, NS, NVS, ZE, PVS, PS, PM, PB



Hình 80. Định nghĩa giá trị ngôn ngữ cho kết quả giá trị góc lái của bộ điều khiển

Các hệ số c_1, c_2, \dots, c_{13} sẽ được tinh chỉnh trong quá trình điều khiển.

Ta có bảng quy tắc suy luận của bộ điều khiển mờ như sau:

Góc lái động cơ servo đánh lái		Sai số góc nghiêng của làn đường															
		NE				ZE				PO							
		Sai số tọa độ điểm trung tâm làn đường															
Tốc độ biến thiên sai số tọa độ điểm trung tâm	NB	NB	NE	ZE	PO	NB	NE	ZE	PO	NB	NE	ZE	PO	NB	NE	ZE	PO
	NS	NS	NVS	ZE	PVS	PS	NM	NS	NVS	ZE	PVS	PS	NM	NS	NVS	ZE	PVS
	ZE	NVS	ZE	PVS	PS	PM	NS	NVS	ZE	PVS	PS	PM	NM	NS	NVS	ZE	PVS
	PS	ZE	PVS	PS	PM	PB	NVS	ZE	PVS	PS	PM	NS	NVS	ZE	PVS	PS	PM
	PB	PVS	PS	PM	PB	PB	ZE	PVS	PS	PM	PB	NVS	ZE	PVS	PS	PM	

Hình 81. Quy tắc suy luận bộ điều khiển mờ điều chỉnh góc servo đánh lái

Sử dụng phương pháp giải mờ trung bình có trọng số để giải mờ.

Thông qua quá trình tinh chỉnh, ta có các hệ số sau:

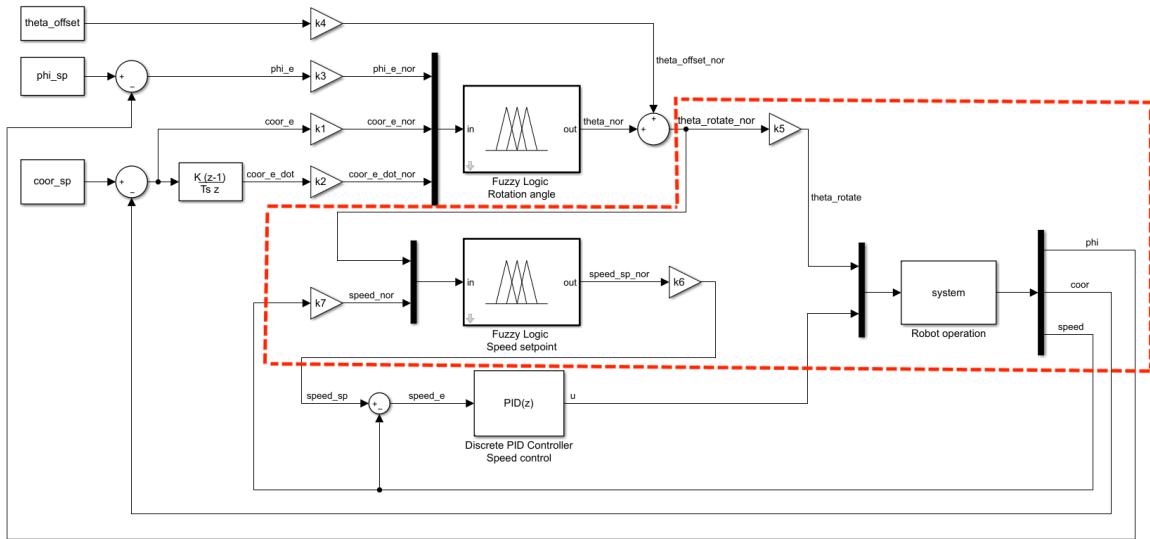
$$c_1 = 0, c_2 = c_3 = 0.3, c_4 = 0.8, c_5 = 0, c_6 = c_7 = 0.3, c_8 = 0.75, c_9 = 0, c_{10} = 0.75, c_{11} = 0.4, c_{12} = 0.5, c_{13} = 0.8$$

c. Thiết kế bộ điều khiển mờ điều chỉnh tốc độ đặt dựa vào kinh nghiệm chuyên gia

Các biến vào, ra của bộ điều khiển:

- Ba biến vào: góc của động cơ servo đánh lái, tốc độ dài hiện tại của xe tự hành
- Biến ra: tốc độ đặt cho xe tự hành

Sơ đồ khái niệm khai triển:



Hình 82. Sơ đồ khối bộ điều khiển mờ điều chỉnh tốc độ đặt (phần được khoanh đỏ)

Chuẩn hóa biến vào, ra của bộ điều khiển:

- Góc của động cơ servo đánh lái đã chuẩn hóa:

$$-1 \leq \text{theta_rotate_nor} \leq 1$$

- Tốc độ dài hiện tại của xe tự hành (cm/s):

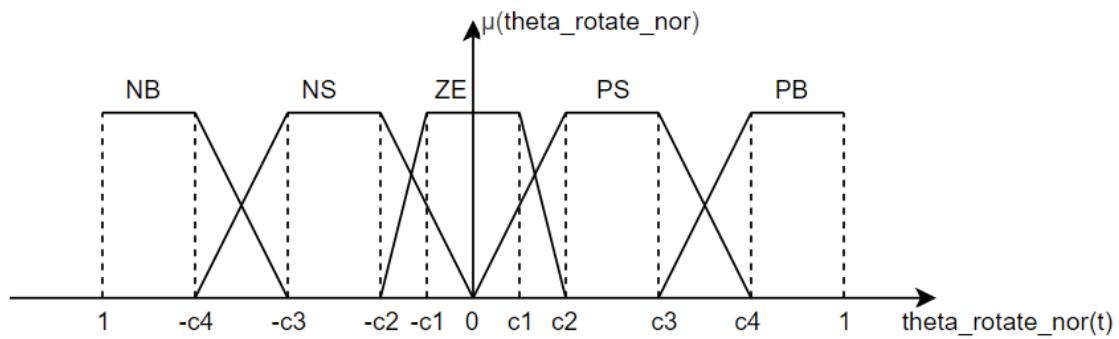
$$0 \leq \text{speed} \leq 20 \Leftrightarrow k_7 = 1/20$$

- Tốc độ đặt của xe tự hành (cm/s):

$$0 \leq \text{speed_sp} \leq 20 \Leftrightarrow k_6 = 20$$

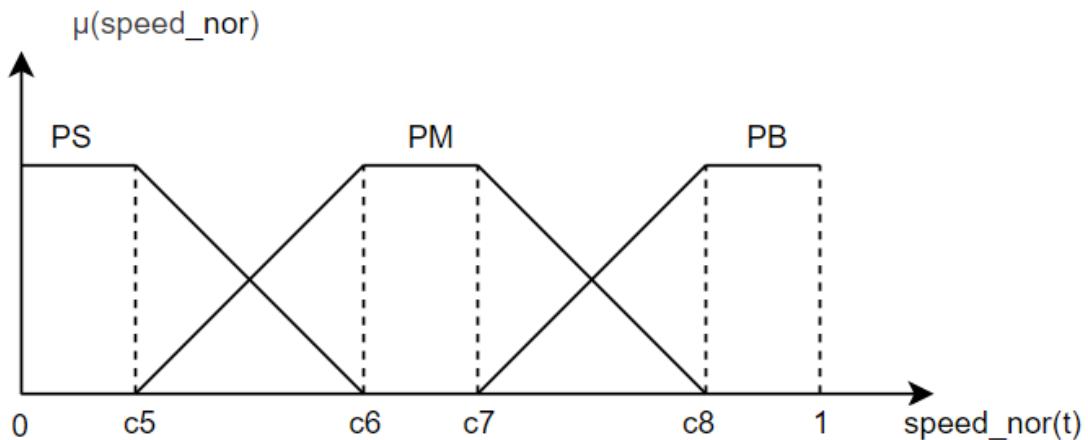
Định nghĩa các giá trị ngôn ngữ cho các biến vào, ra

- Góc của động cơ servo đánh lái đã chuẩn hóa được định nghĩa bởi 5 giá trị ngôn ngữ: NB, NS, ZE, PS, PB



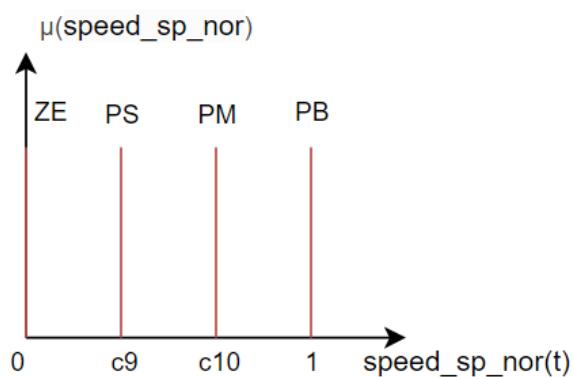
Hình 83. Định nghĩa giá trị ngôn ngữ cho góc của động cơ servo đánh lái đã chuẩn hóa

- Tốc độ hiện tại của xe tự hành được định nghĩa bởi 3 giá trị ngôn ngữ: PS, PM, PB



Hình 84. Định nghĩa giá trị ngôn ngữ cho tốc độ hiện tại của xe tự hành

- Kết quả tốc độ đặt của xe tự hành được định nghĩa bởi 4 giá trị ngôn ngữ: ZE, PS, PM, PB



Hình 85. Định nghĩa giá trị ngôn ngữ cho tốc độ đặt của xe tự hành

Các hệ số c1, c2, ..., c10 sẽ được tinh chỉnh trong quá trình điều khiển.

Ta có bảng quy tắc suy luận của bộ điều khiển mờ như sau:

Tốc độ đặt		Góc xoay của động cơ servo đánh lái				
		NB	NS	ZE	PS	PB
Tốc độ hiện tại	PS	PS	PS	PM	PS	PS
	PM	PS	PM	PB	PM	PS
	PB	PS	PM	PB	PM	PS

Hình 86. Quy tắc suy luận bộ điều khiển mờ điều chỉnh tốc độ đặt cho xe tự hành

Sử dụng phương pháp giải mờ trung bình có trọng số để giải mờ.

Thông qua quá trình tinh chỉnh, ta có các hệ số sau:

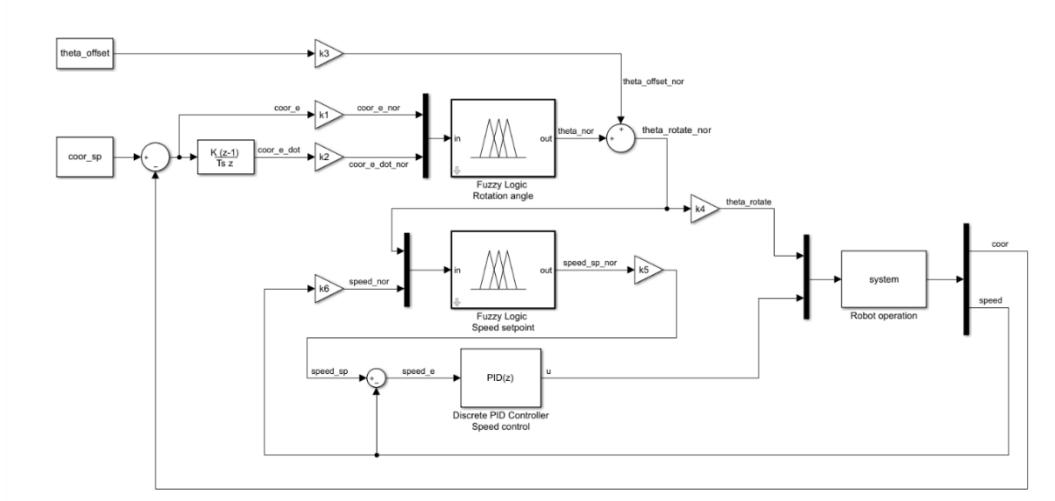
$$c_1 = 0, c_2 = c_3 = 0.4, c_4 = 0.8, c_5 = 0.25, c_6 = c_7 = 0.5, c_8 = 0.3, c_9 = 0.75$$

5.3.3.6. Chương trình điều khiển rẽ trái, rẽ phải

a. Phương pháp điều khiển

Chương trình điều khiển rẽ trái, rẽ phải có cơ chế hoạt động tương tự như chương trình điều khiển di chuyển theo làn đường đã trình bày ở mục 5.3.3.5. Trong mô hình điều khiển theo làn đường, hệ thống được chia thành ba tầng chính: tầng đầu xử lý tín hiệu điều hướng, tầng thứ hai điều chỉnh tốc độ đặt dựa trên góc lái, và tầng thứ ba là bộ điều khiển PID thực thi điều khiển tốc độ. Ở chương trình rẽ trái, rẽ phải, hai tầng sau vẫn được giữ nguyên như trong phần 5.3.3.5. nên sẽ không trình bày chi tiết phần đó. Điểm khác biệt duy nhất nằm ở tầng đầu tiên, nơi bộ điều khiển Fuzzy ba ngõ vào được thay thế bằng bộ điều khiển Fuzzy hai ngõ vào. Cụ thể, bộ điều khiển này chỉ sử dụng sai số tọa độ và đạo hàm sai số tọa độ để tính toán góc lái, trong khi sai số góc nghiêng sẽ không được xét đến.

Khi vạch kẻ này được phát hiện lại, hệ thống hiểu rằng quá trình rẽ đã hoàn tất và sẽ chuyển về trạng thái điều khiển theo làn đường thông thường.



Hình 87. Sơ đồ chương trình điều khiển rẽ trái, rẽ phải

Ý nghĩa của các biến trong sơ đồ vẫn tương tự như đã trình bày ở mục 5.3.3.5.

Ngay khi nhận được lệnh rẽ trái hoặc rẽ phải, hệ thống sẽ xoay camera tương ứng sang trái hoặc phải 30 độ, đồng thời cập nhật giá trị offset góc lái tương ứng cho động cơ Servo. Điểm đặt (setpoint) khi rẽ không còn là trung tâm ảnh mà được dịch lệch về phía hướng rẽ, cho phép xe bám sát vạch kẻ đường bên trái (khi rẽ trái) hoặc bên phải (khi rẽ phải).

Sau khi thiết lập, hệ thống liên tục thu nhận dữ liệu vạch kẻ phía rẽ, tính toán các sai số cần thiết, và thực hiện điều khiển góc lái dựa trên bộ điều khiển Fuzzy hai ngõ vào. Trong suốt quá trình này, hệ thống luôn theo dõi tín hiệu vạch kẻ đường phía đối diện. Khi phát hiện lại vạch kẻ đối diện (trạng thái từ không nhìn thấy sang nhìn thấy), hệ thống sẽ hiểu rằng quá trình rẽ đã hoàn tất và chuyển về trạng thái điều khiển di chuyển theo làn đường thông thường.

Quá trình trên có thể tóm lược thông qua sơ đồ giải thuật sau:



Hình 88. Lưu đồ giải thuật chương trình điều khiển rẽ trái, rẽ phải

b. Thiết kế bộ điều khiển mờ điều chỉnh góc lái của động cơ servo đánh lái dựa vào kinh nghiệm chuyên gia

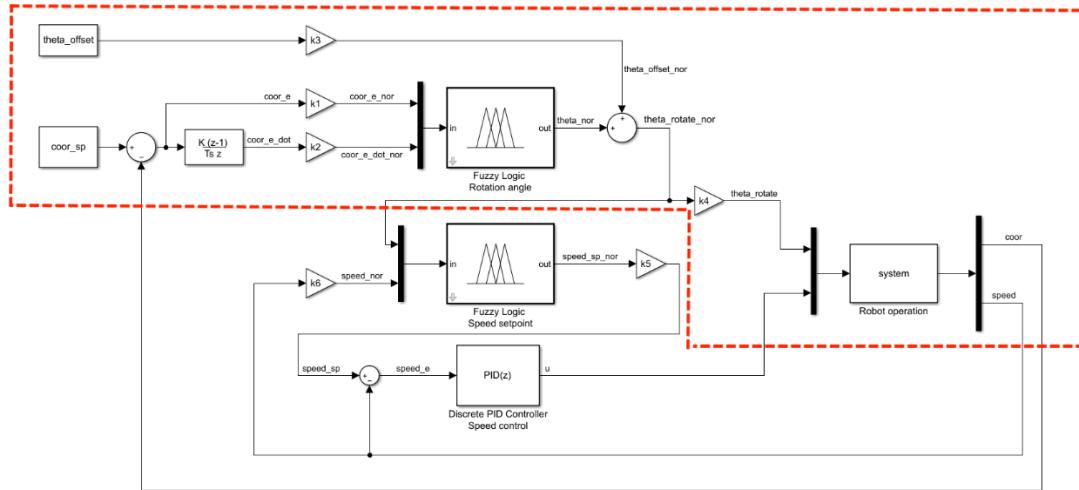
Tín hiệu đặt:

- coor_sp = 250 (nếu rẽ trái)
- coor_sp = 1279 – 250 = 1029 (nếu rẽ phải)

Các biến vào, ra của bộ điều khiển:

- Ba biến vào: sai số và tốc độ biến thiên sai số tọa độ
- Biến ra: góc lái của động cơ servo đánh lái

Sơ đồ khôi hệ thống điều khiển:



Hình 89. Sơ đồ khối bộ điều khiển mờ điều chỉnh góc lái (phần được khoanh đỏ)

Chuẩn hóa biến vào, ra của bộ điều khiển:

- Sai số tọa độ trung tâm làn đường (đơn vị pixel):

$$coor_e = coor_sp - coor \Leftrightarrow 250 \leq coor_e \leq 250 \Leftrightarrow k_1 = 1/250$$

- Biến thiên sai số tọa độ trung tâm làn đường (đơn vị pixel/s):

$$3000 \leq coor_e_dot \leq 3000 \Leftrightarrow k_2 = 1/3000$$

- Góc lái của động cơ servo đánh lái (đơn vị độ):

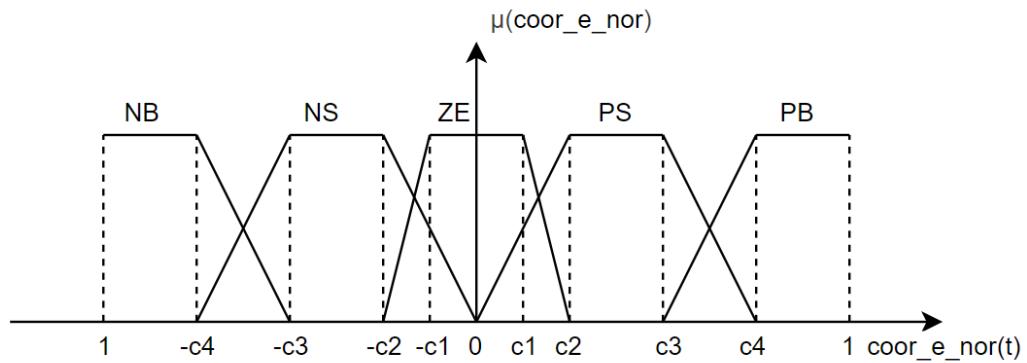
$$-45 \leq theta_rotate \leq 45 \Leftrightarrow k_4 = 45$$

- Góc offset của động cơ servo đánh lái (đơn vị độ):

$$-45 \leq theta_offset \leq 45 \Leftrightarrow k_3 = 1/45$$

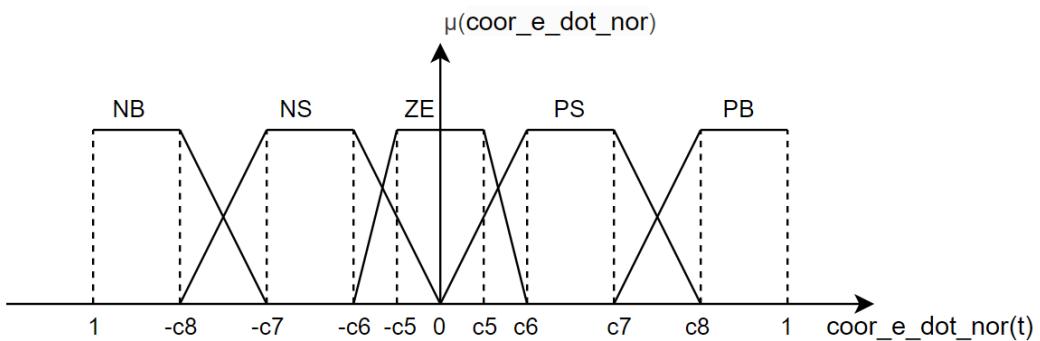
Định nghĩa các giá trị ngôn ngữ cho các biến vào, ra:

- Sai số tọa độ trung tâm coor_e_nor được định nghĩa bởi 5 giá trị ngôn ngữ: NB, NS, ZE, PS, PB



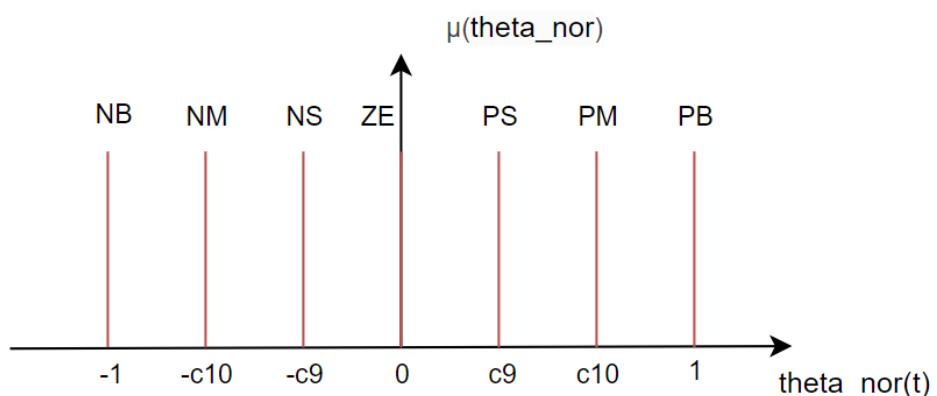
Hình 90. Định nghĩa giá trị ngôn ngữ cho sai số tọa độ điểm trung tâm làn đường

- Tốc độ biến thiên sai số tọa độ coor_e_dot_nor được định nghĩa bởi 5 giá trị ngôn ngữ: NB, NS, ZE, PS, PB



Hình 91. Định nghĩa giá trị ngôn ngữ cho tốc độ biến thiên sai số tọa độ điểm trung tâm làn đường

- Kết quả giá trị góc lái của bộ điều khiển được định nghĩa bởi 7 giá trị ngôn ngữ: NB, NM, NS, ZE, PS, PM, PB



Hình 92. Định nghĩa giá trị ngôn ngữ cho kết quả giá trị góc lái của bộ điều khiển

Các hệ số $c1, c2, \dots, c10$ sẽ được tính chỉnh trong quá trình điều khiển.

Ta có bảng quy tắc suy luận của bộ điều khiển mờ như sau:

Góc lái động cơ servo đánh lái		Sai số tọa độ				
		NB	NS	ZE	PS	PB
Tốc độ biến thiên sai số tọa độ	NB	NB	NM	NS	ZE	
	NS	NB	NM	NS	ZE	PS
	ZE	NM	NS	ZE	PS	PM
	PS	NS	ZE	PS	PM	PB
	PB	ZE	PS	PM	PB	PB

Hình 93. Quy tắc suy luận bộ điều khiển mờ điều chỉnh góc servo đánh lái

Sử dụng phương pháp giải mờ trung bình có trọng số để giải mờ.

Thông qua quá trình tinh chỉnh, ta có các hệ số sau:

$$c1 = 0, c2 = c3 = 0.3, c4 = 0.8, c5 = 0, c6 = c7 = 0.3, c8 = 0.75, c9 = 0.6, c10 = 0.8$$

5.3.3.7. Chương trình chấp hành biển báo giao thông

Hệ thống xe tự hành sử dụng kết quả nhận dạng biển báo giao thông từ Jetson TX2 để quyết định thay đổi hành vi di chuyển. Mỗi khung hình xử lý, Jetson TX2 gửi về gói dữ liệu trạng thái gồm hai nhóm thông tin: danh sách các biển báo đã phát hiện và danh sách các biển báo cần tuân thủ.



Hình 94. Lưu đồ giải thuật xử lý biển báo giao thông

Tại mỗi chu kỳ, hệ thống so sánh giá trị cờ tuân thủ hiện tại với cờ của chu kỳ trước. Nếu phát hiện sự chuyển đổi từ “chưa phát hiện” sang “đã phát hiện” (tức False → True), hệ thống sẽ tiến hành hành động tương ứng với từng loại biển báo.

Các hành vi cụ thể được thiết kế như sau:

- Đèn xanh: Lấy lệnh hành động kế tiếp từ hàng đợi điều hướng và thực hiện
- Đèn đỏ: Dừng xe trong 10 giây, sau đó thực hiện hành động kế tiếp giống như với đèn xanh
- Trạm dừng xe buýt: Dừng xe trong 5 giây, sau đó tiếp tục di chuyển bám theo làn đường
- Bắt buộc rẽ trái: Chuyển hướng sang trạng thái rẽ trái
- Khu vực trẻ em băng qua đường: Giảm tốc độ còn 60% tốc độ tối đa hiện tại trong 10 giây
- Giới hạn tốc độ 40: Giảm tốc độ còn 80% tốc độ tối đa hiện tại trong 10 giây
- Biển dừng: Dừng xe ngay lập tức, đưa về trạng thái đứng yên
- Cấm dừng: Không thay đổi hành vi, chỉ hiển thị cảnh báo trên GUI

Chương 6: Kết quả và hướng phát triển

6.1. Kết quả thực hiện

6.1.1. Kết quả xử lý ảnh

6.1.1.1. Trường hợp xe chưa đi qua giao lộ

Đầu tiên, camera của xe sẽ thu thập dữ liệu hình ảnh xung quanh phương tiện. *Hình 95* minh họa ảnh gốc thu được từ camera gắn phía trước xe, trong điều kiện ánh sáng bình thường. Ảnh này chứa đầy đủ thông tin về mặt đường và các vạch kẻ phân làn, đóng vai trò là dữ liệu đầu vào quan trọng cho chuỗi xử lý tiếp theo.



Hình 95. Ảnh gốc thu được từ camera của xe

Tiếp đến, vùng cần quan tâm của ảnh gốc sẽ được lựa chọn. *Hình 96* thể hiện khu vực cần quan tâm của ảnh gốc, có thể thấy vùng này nằm gần bánh xe và chứa đầy đủ hai vạch kẻ đường, các yếu tố quan trọng để đảm bảo dữ liệu đầu ra của xử lý ảnh là chính xác phục vụ cho bộ điều khiển xe.



Hình 96. Khu vực ảnh được khoanh vùng

Sau đó chương trình sẽ tiến hành tiền xử lý ảnh. Các bước thực hiện gồm: chuyển ảnh thành ảnh xám (*Hình 97*); Lọc nhiễu (*Hình 98*); Đảo ảnh (*Hình 99*); Phân ngưỡng (*Hình 100*).



Hình 97. Ảnh sau khi chuyển thành ảnh xám

Việc chuyển đổi ảnh màu thành ảnh xám sẽ giúp đơn giản và tăng tốc độ xử lý của chương trình.



Hình 98. Ảnh sau khi lọc nhiễu

Bộ lọc trung vị (median blur) giúp loại bỏ nhiễu trong ảnh, đặc biệt là nhiễu muối tiêu. Nhờ đó xe có thể hoạt động trong nhiều điều kiện ánh sáng khác nhau.



Hình 99. Ảnh sau khi đảo

Việc đảo ảnh giúp thuận tiện trong việc xây dựng logic lập trình ở các bước sau. Do trong thực tế các vạch kẻ đường thường có màu sáng và mặt đường có màu tối, tuy nhiên trong sa bàn để tiến hành thực nghiệm thì vạch kẻ đường có màu tối và mặt đường có màu sáng.

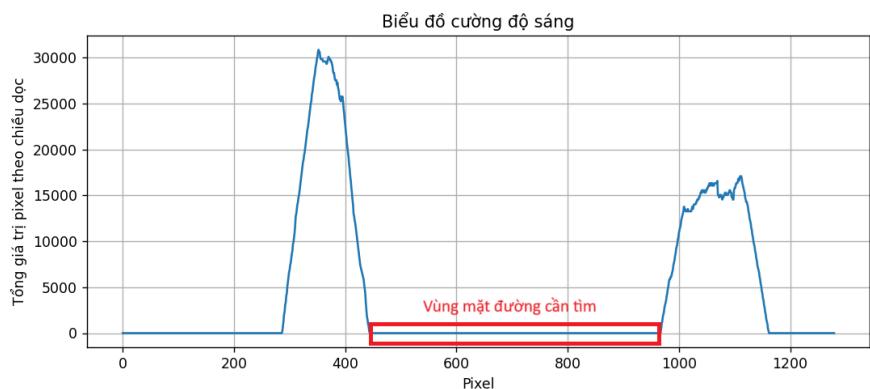


Hình 100. Ảnh sau bước phân ngưỡng

Thực hiện phân ngưỡng sử dụng phương pháp Otsu (Otsu Thresholding), giúp xe có thể hoạt động được trong nhiều điều kiện ánh sáng khác nhau, do thay

vì phải chọn một giá trị cố định để phân ngưỡng thì phương pháp Ostu sẽ tự động chọn giá trị ngưỡng phù hợp.

Kế tiếp chương trình tạo một biểu đồ cường độ sáng theo cột như *hình 101*, để phục cho việc tìm vùng chứa mặt đường.



Hình 101. Biểu đồ cường độ sáng



Hình 102. Vùng ảnh mặt đường tìm được

Có thể thấy vùng mặt đường là vùng được bao quanh giữa 2 làn đường có cường độ sáng thấp, độ sáng tương đối ổn định theo phương ngang, và không có nhiều thay đổi đột ngột về ánh sáng như các vùng khác.



Hình 103. Góc lệch và vị trí trung tâm

Cuối cùng chương trình xác định các điểm thuộc vạch kẻ đường, từ đó tìm ra vị trí trung tâm của mặt đường và góc lệch (*hình 103*), các giá trị này được đưa vào bộ điều khiển giúp xe đưa ra tín hiệu điều khiển phù hợp.

6.1.1.2. Trường hợp xe đang đi thẳng qua giao lộ



Hình 104. Ảnh gốc thu được từ camera của xe khi đi thẳng qua giao lộ

Từ hình ảnh đầu vào chúng ta có thể thấy khi xe đi qua giao lộ, camera của xe chỉ có thể nhìn thấy những vạch kẻ đường ở xa xe, do đặc điểm ở khu vực giao lộ. Vì vậy cần phải sử dụng phương pháp nội suy tính toán ra vị trí trung tâm và góc lệch của mặt đường.

Tiếp đến, vùng cần quan tâm của ảnh gốc sẽ được lựa chọn. *Hình 105* thể hiện khu vực cần quan tâm của ảnh gốc, có thể thấy vùng này thường sẽ lớn hơn vùng ảnh sử dụng khi xe chưa đi qua giao lộ nhằm thu thập nhiều dữ liệu hơn phục vụ cho việc nội suy.

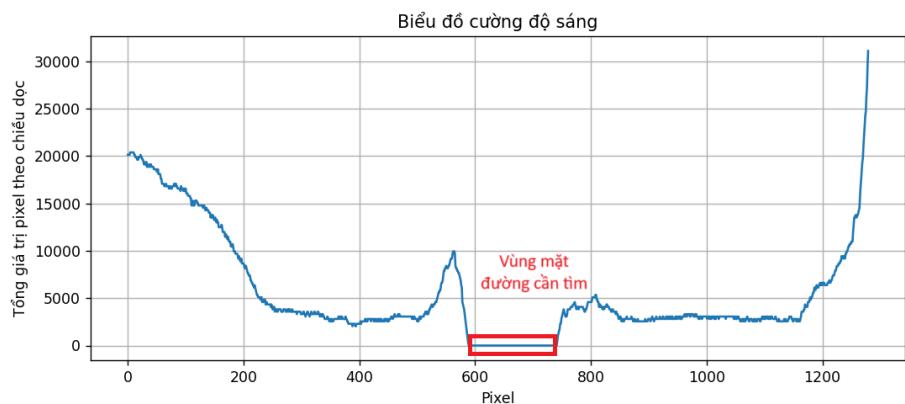


Hình 105. Khu vực ảnh được khoanh vùng khi đi qua giao lộ

Kế tiếp, chương trình sẽ tiến hành tiền xử lý ảnh với thuật toán tương tự với thuật toán dùng trong trường hợp xe chưa đi qua giao lộ (*Hình 106*). Sau đó sẽ tiếp tục tìm ra vùng ảnh chứa mặt đường (*Hình 107*).

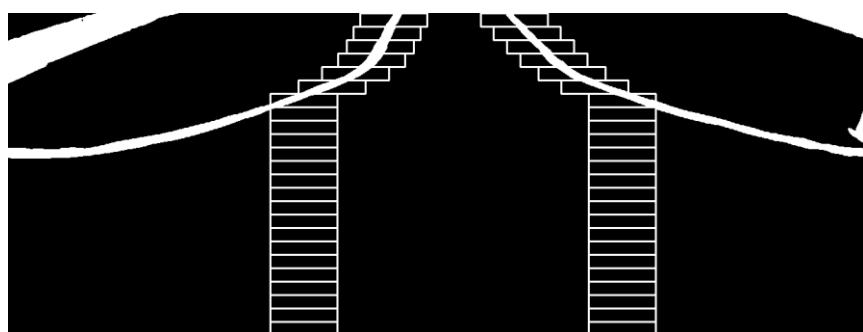


Hình 106. Ảnh sau bước tiền xử lý

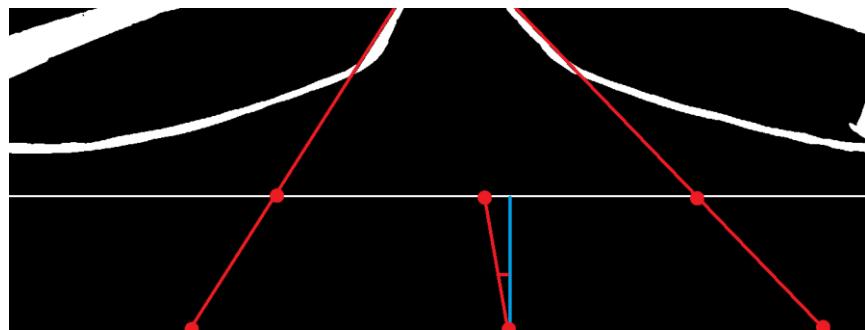


Hình 107. Biểu đồ cường độ sáng

Chương trình tìm ra các đoạn vạch kẻ đường nhờ vào kỹ thuật sliding window (*Hình 108*). từ đó sẽ nội suy ra đoạn thẳng đại diện cho vạch kẻ đường nằm gần bánh xe. Từ đó sẽ tìm ra vị trí trung tâm của mặt đường và góc lệch (*Hình 109*).



Hình 108. Các vạch kẻ đường được tìm ra

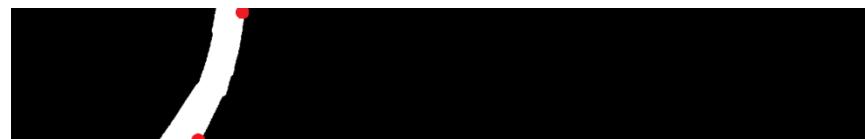


Hình 109. Góc lệch và vị trí trung tâm của mặt đường

6.1.1.3. Trường hợp xe đang rẽ trái/phải ở giao lộ



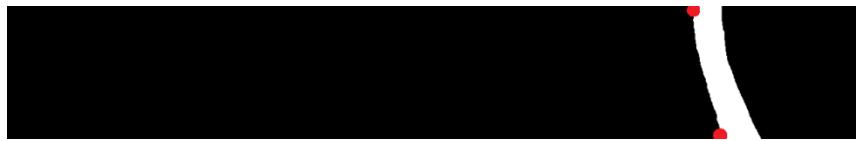
Hình 110. Ảnh gốc thu được từ camera của xe khi đi rẽ trái ở giao lộ



Hình 111. Vạch kẻ đường bên trái được tìm ra



Hình 112. Ảnh gốc thu được từ camera của xe khi đi rẽ phải ở giao lộ

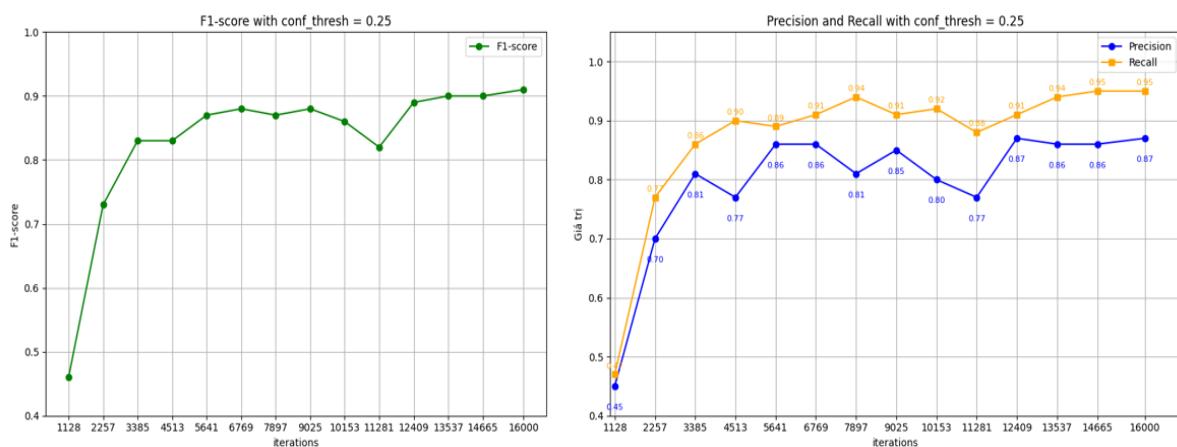


Hình 113. Vạch kẻ đường bên phải được tìm ra

Khi xe tiến hành rẽ trái và rẽ phải ở giao lộ, chúng ta sẽ tiến hành xử lý ảnh tương tự với lúc xe đang đi ở khu vực không phải giao lộ, tuy nhiên lúc này chúng ta chỉ quan tâm đến vạch kẻ đường bên trái nếu xe rẽ trái hoặc vạch kẻ đường bên phải nếu xe rẽ phải. Xe sẽ bám theo các vạch này để rẽ trái, rẽ phải ở giao lộ.

6.1.2. Kết quả nhận diện biển báo giao thông

Mục tiêu của quá trình đánh giá là phân tích và đo lường chất lượng của mô hình nhận diện biển báo giao thông sau khi hoàn tất quá trình huấn luyện. Đánh giá được thực hiện dựa trên các chỉ số hiệu suất phổ biến trong lĩnh vực thị giác máy tính, bao gồm F1-score, Precision, Recall, mAP@0.5 và train loss. Các chỉ số này cung cấp cái nhìn tổng quan về khả năng phát hiện chính xác, mức độ bao phủ và độ tin cậy của mô hình trong việc nhận dạng biển báo trong môi trường thực tế.

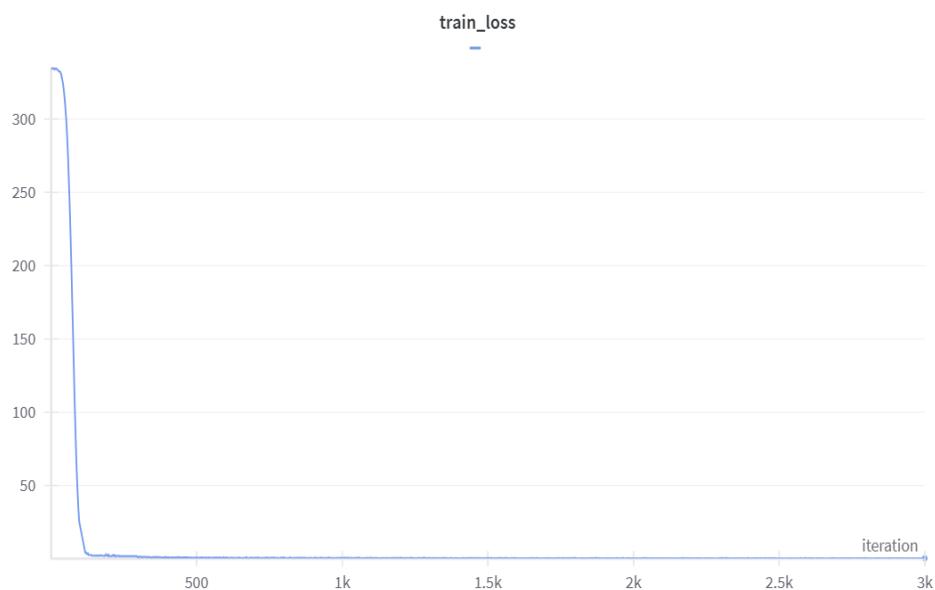


Hình 114. Đồ thị F1-score (trái) và Precision, Recall (phải)

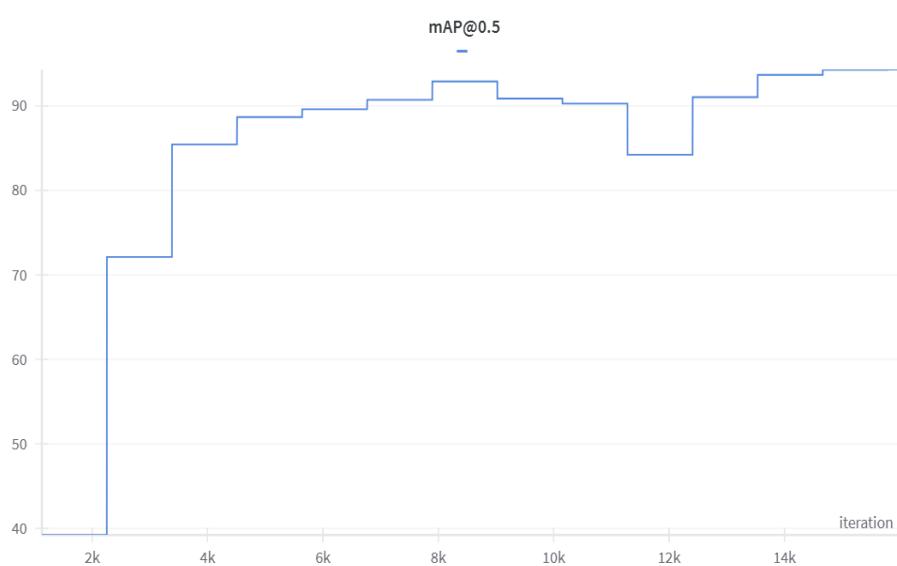
Biểu đồ F1-score cho thấy chỉ số này tăng nhanh từ khoảng 0.45 lên gần 0.9 trong 2.000 vòng lặp, sau đó dao động nhẹ quanh mức 0.9–0.95. Điều này cho thấy mô hình học nhanh và đạt hiệu suất cao, tuy nhiên vẫn có một số dao động nhỏ.

Về Precision và Recall, cả hai đều tăng ổn định trong giai đoạn đầu, sau đó duy trì ở mức khá cao (Precision: ~0.85–0.9, Recall: ~0.8–0.95). Precision có xu

hướng dao động nhẹ và ổn định hơn so với Recall, trong khi Recall có một số điểm giảm nhẹ xuống gần 0.8. Điều này cho thấy mô hình duy trì được sự cân bằng tương đối giữa hai chỉ số. Mô hình cho thấy hiệu suất nhận diện tốt, với F1-score trung bình cao (~0.9), đồng thời cân bằng được giữa độ chính xác và độ bao phủ. Tuy nhiên, sự biến động nhẹ ở các giai đoạn sau gợi ý rằng việc tinh chỉnh thêm mô hình có thể giúp cải thiện tính ổn định.



Hình 115. Đồ thị Train_loss



Hình 116. Đồ thị mAP50

Biểu đồ mAP@0.5 cho thấy chỉ số này tăng nhanh từ khoảng 40 lên trên 80 trong 2.000 vòng lặp đầu, sau đó dao động nhẹ và ổn định trong khoảng 85–90 từ

sau 8.000 vòng lặp. Điều này phản ánh khả năng nhận diện đối tượng chính xác và ổn định của mô hình ở ngưỡng IoU 0.5, vốn là một tiêu chí đánh giá quan trọng trong các bài toán phát hiện đối tượng.

Biểu đồ Train Loss ghi nhận giá trị giảm mạnh từ khoảng 350 xuống gần 0 trong 500 vòng lặp đầu tiên và duy trì ở mức rất thấp trong suốt quá trình huấn luyện. Mặc dù điều này cho thấy mô hình đã học tốt trên tập huấn luyện và tối ưu hóa hiệu quả hàm mất mát, train loss gần bằng 0.

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	TP = 480	FP = 72
Predicted Negative (0)	FN = 25	

Bảng 4. Confusion matrix

Dựa trên ba thông số chính từ confusion matrix gồm: True Positive (TP) = 480, False Positive (FP) = 72, và False Negative (FN) = 25, có thể đưa ra đánh giá hiệu suất của mô hình như sau:

$$Precision = \frac{TP}{TP + FP} = \frac{480}{480 + 72} \approx 0.87$$

$$Recall = \frac{TP}{TP + FN} = \frac{480}{480 + 25} \approx 0.95$$

$$F1_core = 2 \times \frac{Precision \times Recall}{Precision + Recall} \approx 0.91$$

Mô hình đạt Precision (độ chính xác khi dự đoán lớp dương) khoảng 0.87, cho thấy khoảng 87% các trường hợp được dự đoán là biến báo thực sự chính xác, trong khi vẫn tồn tại một tỷ lệ 13% dự đoán sai dương tính (False Positive). Điều này hàm ý mô hình có thể đưa ra một số cảnh báo không cần thiết trong ứng dụng thực tế.

Recall (khả năng phát hiện đầy đủ các trường hợp dương tính) đạt khoảng 0.95, phản ánh năng lực rất tốt trong việc nhận diện đầy đủ các biến báo thực sự, với tỷ lệ bỏ sót thấp (False Negative chỉ chiếm ~5%). Đây là yếu tố quan trọng trong các hệ thống yêu cầu ưu tiên độ nhạy cao, chẳng hạn như các ứng dụng an toàn trên phương tiện tự hành.

F1-score – đại diện cho mức độ cân bằng giữa Precision và Recall – đạt khoảng 0.91, thể hiện rằng mô hình duy trì hiệu suất tổng thể tốt, vừa đảm bảo khả năng phát hiện chính xác, vừa hạn chế bỏ sót.

Kết quả thực nghiệm:



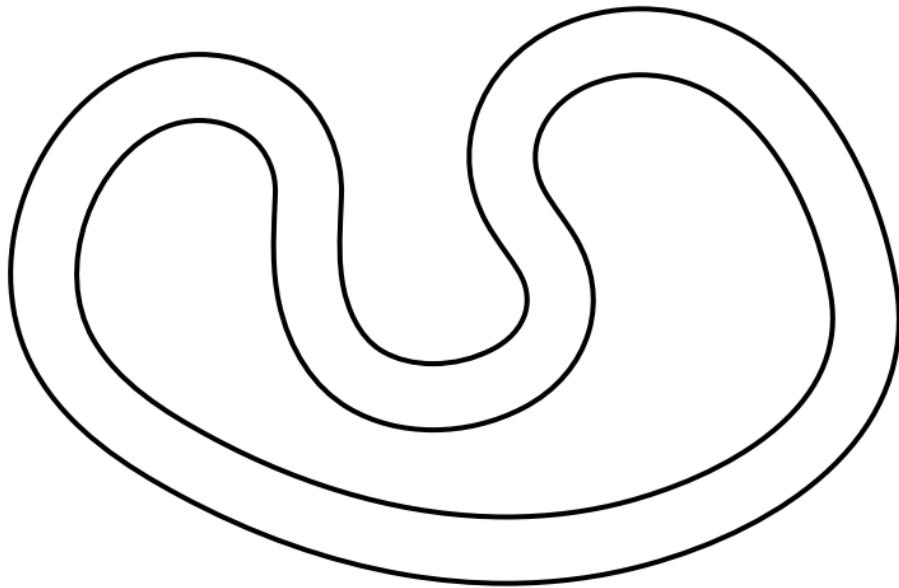
Hình 117. Nhận diện biển báo Children_Crossing



Hình 118. Nhận diện đèn đỏ

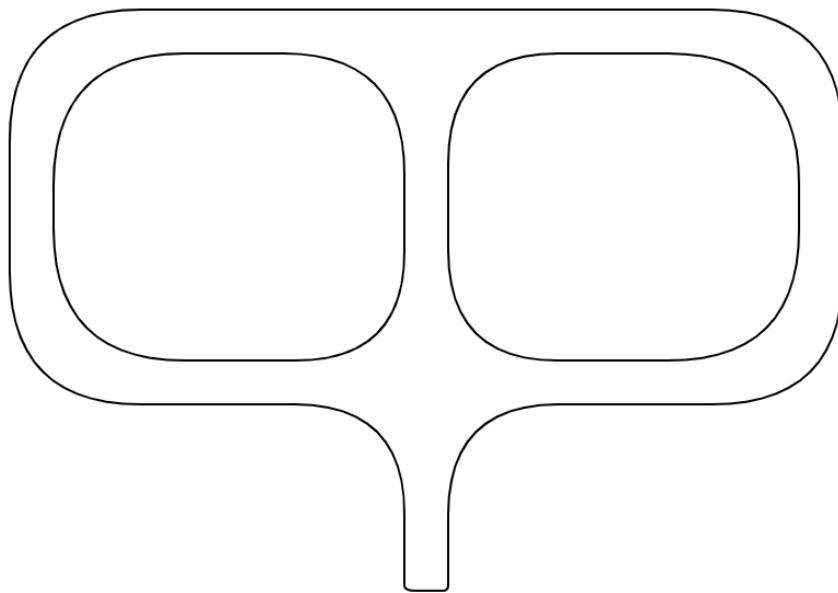
6.1.3. Kết quả vận hành của mô hình xe tự hành

Để tiến hành kiểm tra kết quả vận hành của mô hình xe tự hành, nhóm đã chuẩn bị hai sa bàn với hai mục đích cụ thể khác nhau. Cả hai sa bàn đều được thiết kế với độ rộng của làn đường là 30 cm. Kích thước của cả hai sa bàn đều được vẽ trong kích thước của vùng hình chữ nhật 5m x 3m.



Hình 119. Sa bàn 1

Hình 119 là hình dạng của sa bàn đầu tiên, thiết kế sa bàn này dùng để kiểm tra chủ yếu khả năng đi theo làn đường của xe tự hành, cũng như khả năng thay đổi góc xoay của camera dựa vào độ nghiêng của làn đường.



Hình 120. Sa bàn 2

Hình 120 là hình dạng của san bàn thứ hai, thiết kế sa bàn này dùng để kiểm tra khả năng đi theo làn đường và khả năng đi qua ngã tư, ngã ba, cũng như khả năng rẽ trái, rẽ phải tại các vùng giao nhau.

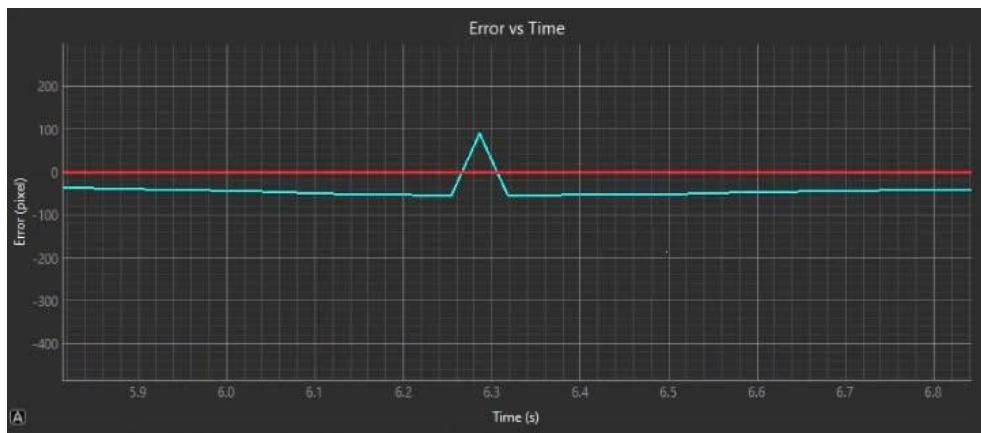
Qua quá trình thực nghiệm, có thể nhận thấy được rằng chương trình xoay camera đã hoạt động như mong đợi, tự động xoay theo tình huống cụ thể, giúp mở rộng vùng quan sát và đảm bảo rằng có thể thu thập dữ liệu hình ảnh của làn đường một cách đầy đủ. Từ đó làm cơ sở để thấy rằng bộ điều khiển được thế kế đã hoạt động được trong việc điều hướng chuyển động của xe. Cụ thể, xe có khả năng chạy ổn định trong làn đường, với nhiều độ cong khác nhau trong suốt quá trình di chuyển. Tại các khu vực giao lộ, bộ điều khiển cũng thể hiện khả năng xử lý khi xe thực hiện các thao tác rẽ trái, rẽ phải hoặc đi thẳng. Tại các thời điểm góc lái của động cơ Servo đánh lái lớn, tốc độ của xe cũng đã giảm đúng theo ý đồ của chương trình điều khiển.

Video kết quả thực nghiệm được đính kèm tại đường dẫn sau (đường link rút gọn, dẫn tới Google Drive): <http://bit.ly/438Mr7c>

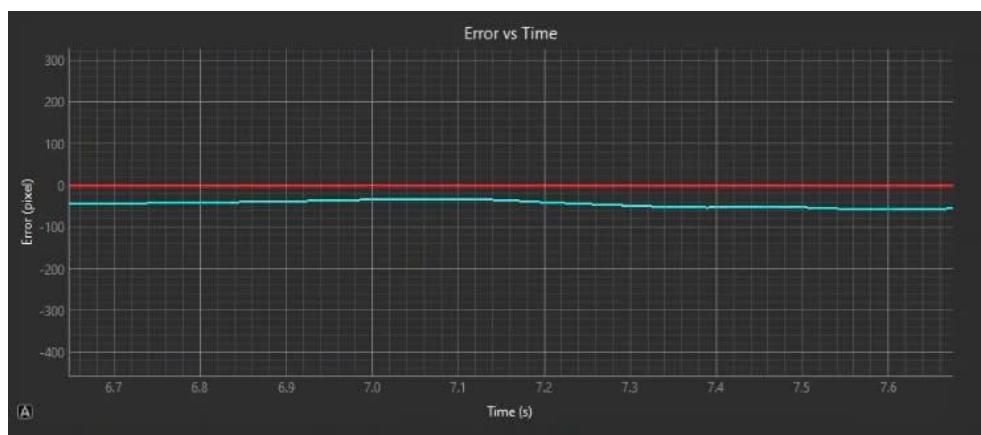
6.2. Đánh giá kết quả

Qua quá trình thực nghiệm, hệ thống điều khiển được triển khai đã cho thấy khả năng vận hành ổn định trong môi trường thực tế. Xe có thể di chuyển bám sát theo làn đường và thực hiện được các thao tác rẽ trái, rẽ phải và đi thẳng tại các giao lộ. Ngoài ra, chức năng nhận diện biển báo giao thông cũng được hệ thống xử lý và phản hồi ở mức độ chấp nhận được.

Tuy nhiên tín hiệu đầu vào của bộ điều khiển đôi khi xuất hiện những giá trị bất thường, không mong muốn do kết quả của việc xử lý ảnh bị ảnh hưởng bởi nhiều và điều kiện ánh sáng. Để giải quyết tình trạng này nhóm đã thực hiện so sánh giá trị đầu vào thời điểm hiện tại với các giá trị đầu vào trước đó để đảm bảo không xuất hiện giá trị đầu vào bất thường.



Hình 121. Giá trị đầu vào bất thường



Hình 122. Giá trị đầu vào bất thường sau khi được xử lý

Do sa bàn nhóm thực hiện không có đoạn đường dài và thẳng (hoặc góc nghiêng ít), dẫn đến việc trạng thái của góc xoay camera thường xuyên thay đổi. Mỗi khi góc nhìn camera thay đổi, do điểm đặt tọa độ vẫn giữ nguyên nên sai số tọa độ và sai số độ nghiêng sẽ tăng lên cao tại mỗi giai đoạn chuyển trạng thái của góc nhìn camera. Điều này dẫn đến việc sai số tọa độ và sai số góc nghiêng của quá trình vận hành của xe vẫn còn tương đối lớn, chưa thể về mức sai số thấp như mong muốn. Có thể nhìn thấy rõ điều đó qua hình sau.



Hình 123. Đồ thị thể hiện sai số tọa độ, sai số góc nghiêng, góc của động cơ servo đánh lái, tốc độ mô hình theo thời gian

Trong hình trên, phía trên bên trái là đồ thị biểu thị sai số tọa độ theo thời gian, phía trên bên phải là đồ thị biểu thị sai số góc nghiêng theo thời gian, phía dưới bên trái là đồ thị biểu thị góc của động cơ servo đánh lái theo thời gian và phía dưới bên phải là đồ thị biểu thị tốc độ của mô hình theo thời gian. Màu sắc của đường vẽ của biểu đồ phụ thuộc vào trạng thái của góc xoay camera hiện tại. Trong hình trên, có hai màu là màu xanh và màu xanh lá cây. Màu xanh thể hiện rằng đó là dữ liệu lúc camera nhìn thẳng phía trước (tức góc 0 độ), màu xanh lá cây thể hiện dữ liệu lúc camera nhìn qua trái 30 độ.

Dữ liệu trên được thu thập khi mô hình đang vận hành ở trạng thái “Di chuyển theo làn đường”. Ta có thể thấy rằng tại mỗi thời điểm chuyển trạng thái của góc xoay camera, sai số tọa độ và sai số góc nghiêng lại có sự thay đổi, đó là do ảnh hưởng của việc thay đổi góc nhìn đột ngột. Tuy vậy, sau khi thay đổi trạng thái góc nhìn, sai số tọa độ luôn có xu hướng giảm dần về không, mặc dù sai số vẫn còn khá lớn. Kèm với đó, ta có thể thấy được mối liên hệ giữa hai đồ thị góc của động cơ servo đánh lái và đồ thị tốc độ với nhau. Mỗi khi góc của động cơ servo đánh lái càng cao thì tốc độ càng giảm và ngược lại.

Ngoài ra, vẫn còn tồn tại một số hạn chế cần lưu ý. Việc phát hiện làn đường chủ yếu dựa vào xử lý ảnh truyền thống nên độ chính xác bị ảnh hưởng đáng kể trong điều kiện ánh sáng thay đổi, ví dụ như khi có bóng râm, ánh sáng ngược hoặc

trời âm u. Đặc biệt, trong trường hợp xe đi qua ngã tư, hệ thống phải sử dụng phương pháp nội suy để ước lượng làn đường do không còn thông tin trực tiếp về vạch kẻ, điều này khiến độ chính xác trong việc xác định trung tâm và góc lệch có thể bị sai lệch.

Bên cạnh đó, khả năng nhận diện biển báo giao thông cũng chưa thực sự ổn định trong mọi điều kiện. Hệ thống hoạt động tốt khi biển báo được chiếu sáng đầy đủ và không bị che khuất, tuy nhiên dễ bị suy giảm hiệu quả khi ánh sáng yếu hoặc biển báo bị mờ, nghiêng.

6.3. Hướng phát triển

Để nâng cao hiệu suất và độ tin cậy của hệ thống trong điều kiện thực tế đa dạng hơn, các hướng phát triển tiềm năng trong tương lai bao gồm:

Ứng dụng học sâu (Deep Learning) trong nhận diện làn đường: Thay vì sử dụng các kỹ thuật xử lý ảnh truyền thống vốn nhạy cảm với điều kiện môi trường, việc áp dụng các mô hình học sâu như Semantic Segmentation (ví dụ: ERFNet, ENet, hoặc LaneNet) có thể giúp cải thiện độ chính xác trong nhận diện làn đường, đặc biệt trong các tình huống phức tạp như ngã tư, vạch mờ, hay ánh sáng không đồng đều.

Cải tiến chương trình nhận diện biển báo giao thông: Hướng phát triển tiềm năng là sử dụng những mô hình Deep Learning mới nhất hiện nay như mô hình YOLO v11, YOLO v12, ... Những mô hình này có khả năng nhận diện biển báo trong nhiều điều kiện ánh sáng, góc nhìn và kích thước khác nhau với độ chính xác cao hơn. Ngoài ra, việc huấn luyện mô hình với tập dữ liệu đa dạng về môi trường thực tế tại Việt Nam cũng là một yếu tố quan trọng để tăng tính ứng dụng.

Tài liệu tham khảo

- [1] “*Otsu Thresholding with OpenCV*”, [Trực tuyến]. Truy cập từ: <https://learnopencv.com/otsu-thresholding-with-opencv/>
- [2] “*The Ultimate Guide to Real-Time Lane Detection Using OpenCV*”, [Trực tuyến]. Truy cập từ: <https://automaticaddison.com/the-ultimate-guide-to-real-time-lane-detection-using-opencv/>
- [3] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, *You Only Look Once: Unified, Real-Time Object Detection*, University of Washington, Allen Institute for AI, Facebook AI Research.
- [4] Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao, *YOLOv4: Optimal Speed and Accuracy of Object Detection*, Institute of Information Science, Academia Sinica, Taiwan.
- [5] “*YOLO: You Only Look Once*”, [Trực tuyến]. Truy cập từ: <http://pjreddie.com/yolo/>
- [6] “*TensorRT Demos by jkjung-avt*”, [Trực tuyến]. Truy cập từ: https://github.com/jkjung-avt/tensorrt_demos
- [7] “*Train YOLOv4-tiny on Custom Data – Lightning Fast Detection*”, [Trực tuyến]. Truy cập từ: <https://blog.roboflow.com/train-yolov4-tiny-on-custom-data-lightning-fast-detection/>
- [8] H. T. Hoàng, “*Chương 2_NMDKTM*”.
- [9] H. T. Hoàng, “*Chương 3_NMDKTM*”.
- [10] “*Protocol Buffers (Protobuf) – Google Developers*”, [Trực tuyến]. Truy cập từ: <https://protobuf.dev/>
- [11] “*MG995 Robot Servo – 180° Rotation*”, [Trực tuyến]. Truy cập từ: <https://towerpro.com.tw/product/mg995-robot-servo-180-rotation/>
- [12] ITR VN, “*Coding Standard and Strategic Approaches*”.

[13] Phúc Đỗ Minh, “*Self-driving car using CNN + YOLOv4-tiny + TensorRT on Jetson nano kit*”, [Trực tuyến]. Truy cập từ:

https://www.youtube.com/watch?v=v734HtE8bno&ab_channel=Ph%C3%BAc%C4%90%BB%97Minh