

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

CHÂU TRÍ VIỄN
VÕ HIỆP THÀNH

ĐỒ ÁN TỐT NGHIỆP
GIẢI PHÁP ĐỊNH VỊ VÀ ĐIỀU HƯỚNG XE TỰ
HÀNH SỬ DỤNG MÃ QR CODE

CỬ NHÂN NGÀNH KỸ THUẬT ĐIỀU KHIỂN VÀ TỰ ĐỘNG HÓA

TP.HỒ CHÍ MINH, 2023

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

**CHÂU TRÍ VIỄN - 1912434
VÕ HIỆP THÀNH - 1910535**

ĐỒ ÁN TỐT NGHIỆP
GIẢI PHÁP ĐỊNH VỊ VÀ ĐIỀU HƯỚNG XE TỰ HÀNH
SỬ DỤNG MÃ QR CODE

**POSITIONING AND NAVIGATION SOLUTION FOR
AUTOMATED GUIDED VEHICLE USING QR CODE**

CỬ NHÂN NGÀNH KỸ THUẬT ĐIỀU KHIỂN VÀ TỰ ĐỘNG HÓA

GIẢNG VIÊN HƯỚNG DẪN
TIẾN SĨ NGUYỄN VĨNH HẢO

TP.HỒ CHÍ MINH, 2023

**CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI
TRƯỜNG ĐẠI HỌC BÁCH KHOA -ĐHQG -HCM**

Cán bộ hướng dẫn Khóa luận tốt nghiệp :
(Ghi rõ họ, tên, học hàm, học vị và chữ ký)

Cán bộ chấm nhận xét 1 :
(Ghi rõ họ, tên, học hàm, học vị và chữ ký)

Cán bộ chấm nhận xét 2 :
(Ghi rõ họ, tên, học hàm, học vị và chữ ký)

Khóa luận tốt nghiệp được bảo vệ tại Trường Đại học Bách Khoa,
ĐHQG Tp.HCM ngày tháng năm

Thành phần Hội đồng đánh giá khóa luận tốt nghiệp gồm:
(Ghi rõ họ, tên, học hàm, học vị của Hội đồng chấm bảo vệ khóa luận
tốt nghiệp)

1.
2.
3.
4.
5.

Xác nhận của Chủ tịch Hội đồng đánh giá khóa luận tốt nghiệp và Chủ
nhiệm Bộ môn sau khi luận văn đã được sửa chữa (nếu có).

CHỦ TỊCH HỘI ĐỒNG

CHỦ NHIỆM BỘ MÔN.....

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

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT
NAM
Độc lập - Tự do - Hạnh phúc

TP.HCM, ngày.....tháng.....năm.....

NHẬN XÉT ĐỒ ÁN TỐT NGHIỆP CỦA CÁN BỘ HƯỚNG DẪN

Tên luận văn:

GIẢI PHÁP ĐỊNH VỊ VÀ ĐIỀU HƯỚNG XE TỰ HÀNH SỬ DỤNG MÃ QR CODE

Nhóm Sinh viên thực hiện:

Châu Trí Viễn

1912434 TS. Nguyễn Vĩnh Hảo

Võ Hiệp Thành

1910535 TS. Nguyễn Vĩnh Hảo

Đánh giá Luận văn

1. Về cuốn báo cáo:

Số trang _____

Số chương _____

Số bảng số liệu _____

Số hình vẽ _____

Số tài liệu tham khảo _____

Sản phẩm _____

Một số nhận xét về hình thức cuốn báo cáo:

.....
.....
.....
.....
.....
.....
.....
.....

2. Về nội dung luận văn:

.....
.....
.....
.....
.....
.....
.....
.....

3. Về tính ứng dụng:

.....

.....

.....

.....

.....

.....

.....

.....

4. Về thái độ làm vi:

.....

.....

.....

.....

.....

.....

.....

.....

Đánh giá chung: Luận văn đạt/không đạt yêu cầu của một luận văn tốt nghiệp kỹ sư, xếp loại Giỏi/ Khá/ Trung bình

Điểm từng sinh viên:

Châu Trí Viễn /10

Võ Hiệp Thành /10

Cán bộ hướng dẫn

(Ký tên và ghi rõ họ tên)

TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ

MINH

KHOA ĐIỆN - ĐIỆN TỬ

BỘ MÔN: ĐIỀU KHIỂN TỰ ĐỘNG

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

TP.HCM, ngày.....tháng.....năm.....

NHẬN XÉT ĐỒ ÁN TỐT NGHIỆP CỦA CÁN BỘ PHẢN BIỆN

Tên luận văn:

GIẢI PHÁP ĐỊNH VỊ VÀ ĐIỀU HƯỚNG XE TỰ HÀNH SỬ DỤNG MÃ QR CODE

Nhóm Sinh viên thực hiện:

Châu Trí Viễn

1912434 GV.

Võ Hiệp Thành

1910535 GV.

Cán bộ phản biện:

Đánh giá Luận văn

5. Về cuốn báo cáo:

Số trang _____

Số chương _____

Số bảng số liệu _____

Số hình vẽ _____

Số tài liệu tham khảo _____

Sản phẩm _____

Một số nhận xét về hình thức cuốn báo cáo:

.....
.....
.....
.....
.....
.....
.....

6. Về nội dung luận văn:

.....
.....
.....
.....
.....
.....

.....

7. Về tính ứng dụng:

.....

.....

.....

.....

.....

.....

.....

8. Về thái độ làm việc của sinh viên:

.....

.....

.....

.....

.....

.....

.....

.....

Đánh giá chung: Luận văn đạt/không đạt yêu cầu của một luận văn tốt nghiệp kỹ sư, xếp loại Giỏi/ Khá/ Trung bình

Điểm từng sinh viên:

Châu Trí Viễn /10

Võ Hiệp Thành /10

Cán bộ hướng dẫn

(Ký tên và ghi rõ họ tên)

Lời cảm ơn

Lời đầu tiên, chúng em xin được bày tỏ lòng biết ơn sâu sắc tới quý thầy cô Trường đại học Bách Khoa – Đại học Quốc gia Thành phố Hồ Chí Minh nói chung, quý thầy cô khoa Điện – Điện tử nói riêng, đã truyền đạt cho chúng em những kiến thức và hướng đi quý giá để phục vụ cho quá trình hoàn thành Luận văn tốt nghiệp.

Chúng em xin gửi lời cảm ơn đặc biệt đến thầy Nguyễn Vĩnh Hảo – Trưởng bộ môn Điều khiển tự động. Bên cạnh những kiến thức thầy giảng dạy, thầy còn hỗ trợ về không gian phòng Lab, địa điểm đi họp và làm việc. Có được sự hỗ trợ của thầy chính là một trong những cơ hội lớn nhất của chúng em có được trong những năm học khó khăn vừa qua.

Bên cạnh sự biết ơn đó đó, chúng em cũng xin chân thành cảm ơn thầy Trần Quốc Tiến Dũng, thầy đã giúp đỡ, động viên chúng em những lúc khó khăn và tình huống cấp bách, nhưng kiến thức lắn kinh nghiệm thầy chỉ bảo đã giúp chúng em vượt qua được những vấn đề trong lúc xây dựng phần cứng. Chúng em cũng muốn bày tỏ lòng biết ơn đến thầy Nguyễn Trung Hiếu bộ môn Điện tử. Cảm ơn thầy đã hướng dẫn, gợi ý chúng em trong quá trình lựa chọn đề tài và cả quá trình học tập khó khăn trong 4 năm qua.

Cảm ơn anh em đồng nghiệp ở CPE Lab – Trung tâm phát triển hạ tầng – FPT Telecom miền Nam đã tài trợ luận văn nhiều thiết bị cũng như kiến thức, góp phần không nhỏ để chúng em hoàn thành luận văn. Chính nhờ được làm việc ở phòng Lab mỗi ngày đã giúp chúng em tiếp thêm động lực hoàn thành luận văn và mong muốn ra trường cống hiến cho quý công ty.

Cảm ơn Nguyễn Gia Nguyên, người bạn đáng tin cậy đã cho những lời khuyên, sự hỗ trợ giá trị trong quá trình hoàn thành luận văn. Cảm ơn Phan Sơn Phúc, người bạn cùng khoa đã hỗ trợ chúng mình board mạch trong tình huống khẩn cấp cần thay thế linh kiện.

Cuối cùng, chúng em gửi cảm ơn đến gia đình đã chăm lo và hỗ trợ. Sự động viên, quan tâm, lo lắng đến từ gia đình là động lực để chúng em theo đuổi con đường học tập và không bỏ cuộc khi khó khăn. Trong những tình huống tâm lý không tốt, phải lựa chọn đi tiếp hay chuyển hướng, chính chỗ dựa gia đình là nền tảng vững chắc để hai sinh viên mạnh mẽ hơn từng ngày.

TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ
CHÍ MINH

KHOA ĐIỆN - ĐIỆN TỬ

BỘ MÔN: ĐIỀU KHIỂN TỰ ĐỘNG

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT
NAM

Độc lập - Tự do - Hạnh phúc

TP.HCM, ngày 12 tháng 5 năm
2023

ĐỀ CƯƠNG CHI TIẾT

TÊN LUẬN VĂN: GIẢI PHÁP ĐỊNH VỊ VÀ ĐIỀU HƯỚNG XE TỰ HÀNH SỬ DỤNG
MÃ QR CODE

Cán bộ hướng dẫn: TS. Nguyễn Vĩnh Hảo

Thời gian thực hiện: Từ ngày 02/01/2023 đến ngày 12/05/2023

Sinh viên thực hiện: Châu Trí Viễn – 1912434

Võ Hiệp Thành – 1910535

Nội dung đề tài

Mục tiêu:

Xây dựng mô hình xe tự hành trong công nghiệp với giải pháp sử dụng mã QR Code để điều hướng và định vị.

Phương pháp thực hiện:

- Thiết kế phần cứng, xây dựng mô hình xe 4 bánh (2 bánh mắt trâu đa hướng và 2 bánh xe có động cơ).
- Lập trình phần mềm trên STM32, giao tiếp Serial Port với Raspberry Pi.
- Lập trình Raspberry tiếp nhận lệnh từ giao diện điều khiển thông qua giao thức MQTT, thực hiện các bước xử lý mã QR Code, căn chỉnh và gửi tín hiệu điều khiển động cơ ngược về STM32, đồng thời gửi phản hồi về giao diện điều khiển.
- Lập trình giao diện điều khiển Qt C++, có thể kết nối từ xa để gửi lệnh và nhận phản hồi từ xe.

Kết quả mong đợi:

- Xây dựng thành công mô hình xe để chạy thử nghiệm.
- Trên máy tính nhúng, thực hiện được các quá trình xử lý ảnh trên mã

<p>QR để xác định được góc, độ lệch, tọa độ.</p> <ul style="list-style-type: none"> - Thực hiện thành công quá trình điều khiển, giao tiếp, điều hướng từ xe đến giao diện người dùng. - Lập trình giao diện điều khiển có phản hồi, trực quan và dễ sử dụng. 	
Kế hoạch thực hiện	
Châu Trí Viễn	Võ Hiệp Thành
<p>Giai đoạn 1: Tiếp nhận dự án, tìm hiểu tổng thể cấu trúc robot.</p> <ul style="list-style-type: none"> - Tìm hiểu thông tin phần cứng về nguồn, ổn áp, giảm áp, ắc quy. - Tìm hiểu AutoCad. - Tìm hiểu về lập trình C trên STM32, lập trình C++ và giao diện điều khiển Qt C++. <p>Giai đoạn 2: Tìm hiểu và bắt đầu thiết kế phần cứng.</p> <ul style="list-style-type: none"> - Vẽ AutoCad, thiết kế mô hình xe. - Dựng mô hình xe và liên kết các module cần thiết, hoàn thiện xe. <p>Giai đoạn 3: Tìm hiểu các thuật toán điều khiển xe, lập trình C trên vi điều khiển.</p> <ul style="list-style-type: none"> - Lập trình giao tiếp UART cho vi điều khiển. - Lập trình Timer, xử lý Encoder, điều khiển vị trí PID, vận tốc hình thang. <p>Giai đoạn 4: Lập trình giao diện người dùng.</p>	<p>Giai đoạn 1: Tiếp nhận dự án, tìm hiểu tổng thể cấu trúc robot.</p> <ul style="list-style-type: none"> - Tìm hiểu về các driver điều khiển động cơ, lựa chọn driver phù hợp. - Tìm hiểu về động cơ, encoder và các thông số giảm tốc. - Tìm hiểu về lập trình python trên máy tính nhúng và tìm hiểu về máy tính nhúng. <p>Giai đoạn 2: Tìm hiểu và bắt đầu thiết kế phần cứng</p> <ul style="list-style-type: none"> - Dựng mô hình xe và liên kết các module cần thiết, hoàn thiện xe. <p>Giai đoạn 3: Tìm hiểu các thuật toán xử lý ảnh, lập trình Python trên máy tính nhúng.</p> <ul style="list-style-type: none"> - Tiến hành xử lý mã QR Code qua camera dùng thư viện OpenCV. - Tính toán các góc lệch và cách calibrate khi xe vận hành với mã QR. - Hoàn thành giải thuật điều khiển với mã QR.

<ul style="list-style-type: none"> - Tìm hiểu giao thức MQTT và Opensource Mosquitto. - Lập trình giao diện Qt C ++ đa luồng, kết hợp giao tiếp MQTT với máy tính nhúng. <p>Giai đoạn 5: Chạy thử, ghi nhận số liệu và đánh giá.</p>	<p>Giai đoạn 4: Hoàn thiện ứng dụng trên máy tính nhúng.</p> <ul style="list-style-type: none"> - Giao tiếp MQTT với giao diện điều khiển. <p>Giai đoạn 5: Chạy thử, ghi nhận số liệu và đánh giá.</p>
<p>Xác nhận của Cán bộ hướng dẫn (Ký và ghi rõ họ tên)</p>	<p>TP.HCM, ngày 12 tháng 5 năm 2023 Sinh viên (Ký và ghi rõ họ tên)</p>

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. TỔNG QUAN VỀ ĐỀ TÀI.....	1
1.1. Giới thiệu xe tự hành trong công nghiệp.....	1
1.2. Ứng dụng trong thực tế	1
1.2.1. Xe tự hành Pegasus ở Amazon.....	1
1.2.2. AGV của nhà máy GigaFactory của Tesla.....	3
1.3. Mục tiêu đề tài	4
1.4. Cấu trúc tổng quát của hệ thống.....	4
1.5. Sơ lược nội dung luận văn.....	5
Chương 2. CHƯƠNG 2. CƠ SỞ LÝ THUYẾT	6
2.1. Mã QR.....	6
2.1.1. Khái niệm và cấu trúc mã QR.....	6
2.1.2. Các phương pháp định vị AGV.....	7
2.2. Điều hướng và định vị bằng mã QR trong điều khiển AGV....	14
2.3. Thuật toán xử lý ảnh	15
2.3.1. Giới thiệu thuật toán Optical Flow	15
2.3.2. Ứng dụng phương pháp Lucas-Kanade vào xử lý QR code.....	16
2.4. Quy luật điều khiển vận tốc	18
2.4.1. Bộ điều khiển PID	18
2.4.2. Ứng dụng vận tốc hình thang vào điều khiển xe	20
2.5. Giao tiếp điều khiển MQTT	22
2.5.1. Khái niệm và các thông số cơ bản của MQTT.....	22
2.5.2. Áp dụng MQTT vào mô hình điều khiển xe từ xa	24
Chương 3. THIẾT KẾ VÀ THI CÔNG PHẦN CỨNG	25
3.1. Sơ đồ kết nối phần cứng.	25
3.1.1. Sơ đồ nguồn	25
3.1.2. Sơ đồ kết nối các thành phần.....	27
3.1.3. AutoCad	28
3.2. Tổng quan các thiết bị.....	29
3.2.1. Vi xử lý STM32F407	29
3.2.2. Raspberry Pi 4B Model 8GB RAM	30

3.2.3. Driver điều khiển động cơ	32
3.2.4. USB Camera Rapoo	33
3.3. Thực tế phần cứng sau khi xây dựng hoàn thiện	34
Chương 4. THIẾT KẾ PHẦN MỀM.....	35
4.1. Giải thuật điều khiển.....	35
4.1.1. Khối PID Motor	35
4.1.2. Khối vận tốc hình thang	36
4.2. Hệ thống phần mềm điều khiển trên Máy tính nhúng	37
4.2.1. Tổng quan giải thuật.....	37
4.2.2. Lập trình đa luồng	38
4.2.3. Giao tiếp MQTT	38
4.2.4. Xử lý ảnh.....	39
4.2.5. Giao tiếp với vi điều khiển thông qua giao thức UART	44
4.3. Hệ thống giao diện điều khiển trên máy tính	45
4.3.1. Tổng quan thuật toán.....	45
4.3.2. Giao diện tổng quát.....	46
4.3.3. Đa luồng.....	50
4.3.4. Giao tiếp MQTT	52
Chương 5. KẾT QUẢ THỰC NGHIỆM	56
5.1. Kết quả điều khiển	56
5.1.1. PID & Vận tốc hình thang	56
5.1.2. Giao diện điều khiển sử dụng Qt C++ và MQTT	56
5.2. Kết quả điều khiển, định vị thông qua QR Code	58
5.2.1. Mô hình triển khai thực nghiệm.....	58
5.2.2. Đánh giá quỹ đạo với điều kiện ổn định.....	60
5.2.3. Kết quả chạy thử trên các vận tốc khác nhau	64
5.2.4. Kết quả chạy thử trên khoảng cách dài hơn.....	66
Chương 6. ĐÁNH GIÁ KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	69
6.1. Đánh giá kết quả	69
6.2. Kết luận.....	71
6.3. Hướng phát triển.....	72

DANH MỤC HÌNH ẢNH

Hình 1.1. Xe tự hành ở Amazon	2
Hình 1.1.2. Xe tự hành ở Amazon thay thế hàng nghìn công nhân.....	3
Hình 1.3. Xe tự hành của nhà máy Tesla.....	3
Hình 1.4. Cấu trúc tổng quan hệ thống.....	4
Hình 2.1. Cấu trúc của một mã QR.....	6
Hình 2.2 Cảm biến bằng các spot từ	8
Hình 2.3. Cách cảm biến từ hoạt động	9
Hình 2.4. Cảm biến bằng đường từ	9
Hình 2.5. Cảm biến bằng Laser	10
Hình 2.6. Định vị bằng RFID.....	11
Hình 2.7. Định vị bằng mã QR.....	12
Hình 2.8. Các xe định vị bằng ma trận mã QR.....	14
Hình 2.9. Sơ đồ khối xe định vị bằng QR Code.	15
Hình 2.10. Tuật toán Optical Flow.....	16
Hình 2.11. Sơ đồ bộ điều khiển PID	18
Hình 2.12. Mô hình vận hành code trong vi điều khiển.	22
Hình 2.13. Mô hình MQTT.....	23
Hình 2.14. Mô hình MQTT ứng dụng trong điều khiển từ xa.....	24
Hình 3.1. Các cụm nguồn dành cho bộ phận điều khiển và trợ sáng	25
Hình 3.2. Cụm nguồn dành cho động cơ và encoder	25
Hình 3.3. Cụm 3 pin 18650.....	26
Hình 3.4. Pin 18650 được xếp thành module 2 tầng.....	27
Hình 3.5. Sơ đồ kết nối các thành phần với nhau.....	27
Hình 3.6. Bản vẽ AutoCad tầng 1.....	28
Hình 3.7. Bảng vẽ AutoCad tầng 2	29
Hình 3.8. Vi điều khiển STM32F407	30
Hình 3.9. Máy tính nhúng Raspberry Pi 4.....	31
Hình 3.10. Driver điều khiển động cơ.....	32
Hình 3.11. USB Camera Rapoo.....	33
Hình 3.12. Hình ảnh thực tế mặc sau xe.	34
Hình 3.13. Hình ảnh thực tế mặc trước xe.....	34
Hình 4.1. Khối PID.....	35
Hình 4.2. Truyền message control thông qua MQTT.....	39
Hình 4.3. Truyền message feedback thông qua MQTT.....	39
Hình 4.4. Phát hiện mã QR và xác định các thông số.....	41
Hình 4.5. Thực hiện quá trình căn chỉnh đầu tiên.....	42
Hình 4.6. Thực hiện quá trình căn chỉnh ngay giữa khung hình.	43
Hình 4.7. Thực hiện quá trình căn chỉnh cuối cùng	44
Hình 4.8. Tổng hợp các bước căn chỉnh.....	44

Hình 4.9. Mô hình giao tiếp giữa STM32 và Máy tính nhúng.....	45
Hình 4.10. Logo của open source Mosquitto.....	46
Hình 4.11. Giao diện tổng quan của GUI.....	47
Hình 4.12. Khu vực set đường đường đi và gửi thông tin.....	48
Hình 4.13. Các loại log được thể hiện bằng màu sắc khác nhau.....	49
Hình 4.14. Khu vực theo dõi quá trình di chuyển của xe.....	49
Hình 4.15. Khu vực setup ban đầu GUI.....	50
Hình 4.16. Mô hình đa luồng được xây dựng cho giao diện điều khiển.....	51
Hình 4.17. Một đoạn code của task capture.....	52
Hình 4.18. Các thông tin cơ bản của EMQX được giới thiệu trên trang chủ.	53
Hình 4.19. Cách các topic được phân nhánh.....	53
Hình 4.20. Mô hình tin nhắn PROCESS được định dạng theo chuỗi JSON....	54
Hình 4.21 Mô hình tin nhắn COMMAND được định dạng theo chuỗi JSON..	54
Hình 4.22. Mô hình tin nhắn phản hồi STATUS.....	55
Hình 4.23. Mô hình tin nhắn phản hồi WARNING.....	55
Hình 5.1. Giao diện tuning PID và vận tốc hình thang.....	56
Hình 5.2. Tin nhắn được truyền đi bởi GUI (kiểm thử bằng terminal).....	57
Hình 5.3. Vùng phản hồi khi xe di chuyển lượt đi được thể hiện màu xanh.	57
Hình 5.4, Vùng phản hồi khi xe di chuyển lượt về được thể hiện qua màu đỏ.	
	58
Hình 5.5. GUI báo lỗi được thể hiện qua đoạn log màu đỏ.....	58
Hình 5.6. Triển khai thực hiện với camera được gắn trên trần.....	59
Hình 5.7. Cách tính độ lệch giữa mã QR Code và xe bằng Camera.....	59
Hình 5.8. Lần kiểm thử đầu tiên.....	60
Hình 5.9. Thử nghiệm chạy liên tục với số lần lặp là 5 lần.....	61
Hình 5.10. Chạy thử nghiệm với đường đi được cho trước.	62
Hình 5.11. Sai số sau mỗi lần check-in tại các mã QR tính theo mm.....	63
Hình 5.12. Thử nghiệm với vận tốc chậm.....	64
Hình 5.13. Thử nghiệm với vận tốc trung bình.....	64
Hình 5.14. Thử nghiệm với vận tốc nhanh.....	65
Hình 5.15. Thử nghiệm với quãng đường là 1m2, lần 1.	66
Hình 5.16. Thử nghiệm với quãng đường 1m2 lần 2.....	66
Hình 5.17. Thử nghiệm lần 3 bị thất bại.....	67
Hình 6.1. Các nhiệm vụ chia nhỏ cần phải làm.....	69
Hình 6.2. Các nhiệm vụ cần thực hiện của phần cứng.....	69
Hình 6.3. Các nhiệm vụ cần thực hiện của phần mềm.	70

DANH MỤC BẢNG

Bảng 2.1. Bảng đánh giá mức độ khôi phục của mã QR.....	7
Bảng 2.2. Bảng so sánh các phương pháp định vị tuyệt đối	13
Bảng 2.3. Bảng đánh giá các thông số ảnh hưởng lên PID.....	20
Bảng 3.1. Bảng thông số kỹ thuật của Robot AGV.....	34
Bảng 4.1. Bảng giá trị góc biên của các mã QR Code.....	43
Bảng 4.2. Bảng mô tả các nhóm lệnh của GUI đến AGV	54
Bảng 4.3. Mô tả các nhóm lệnh từ AGV đến GUI.....	54

DANH MỤC VIẾT TẮT

Tùy khóa	Tên tiếng anh
AGV	Automated guided vehicle
OF	Optical flow
MQTT	Message Queuing Telemetry Transport
GUI	Graphical User Interface
UI	User interface
QR	Quick Response code

TÓM TẮT ĐỒ ÁN

Luận văn này trình bày quá trình thiết kế, xây dựng và lập trình xe tự hành ứng dụng giải pháp định vị và điều hướng bằng QR Code. Trong quá trình xây dựng xe, chúng tôi đã sử dụng vi điều khiển STM32F407 để điều khiển 2 động cơ có encoder. Vi điều khiển này được lập trình bộ điều khiển PID và phương pháp điều khiển vận tốc hình thang, đồng thời nó cũng tiếp nhận thông tin từ máy tính nhúng và phản hồi lại trạng thái thông qua UART. Máy tính nhúng được dùng là Raspberry Pi 4, thiết bị này được lập trình để xử lý mã QR thông qua USB Camera, nhận lệnh từ giao diện điều khiển từ xa, gửi lệnh sau khi phân tích mã QR xuống vi điều khiển, cuối cùng là phản hồi về nơi nhận lệnh.

Luận văn đã xây dựng thành công mô hình xe tự hành di chuyển với vận tốc ổn định là 0.3m/s và khoảng cách lý tưởng là 0.6m giữa các QR Code. Đồng thời dự án cũng xây dựng thành công giao diện điều khiển, chứng minh được giải pháp dùng mã QR để định vị và điều hướng xe là có khả thi.

ABSTRACTION

This thesis presents the process of designing, constructing, and programming a self-driving vehicle that applies a solution based on QR Code location and navigation. During the vehicle construction process, we utilized the STM32F407 microcontroller to control two encoder-equipped motors. This microcontroller was programmed with a PID controller and a trapezoidal velocity control method, and it also received information from the embedded computer and transmitted its state back through UART. The embedded computer used was the Raspberry Pi 4, which was programmed to process QR codes via a USB camera, receive commands from a remote control interface, send commands after analyzing the QR code to the microcontroller, and finally respond to the sender.

The thesis has successfully constructed a self-driving vehicle model that moves at a stable speed of 0.3m/s and has an ideal distance of 0.6m between two QR Codes. Additionally, the project has also successfully constructed a control interface, demonstrating the feasibility of using QR codes for vehicle location and navigation.

Chương 1. TỔNG QUAN VỀ ĐỀ TÀI.

1.1. Giới thiệu xe tự hành trong công nghiệp

Xe tự hành AGV, Robot AGV, tiếng anh là Automated Guided Vehicle, là loại xe sử dụng các công nghệ dẫn đường để vận chuyển hàng hóa, nguyên vật liệu đến những địa điểm đã được đánh dấu sẵn mà không cần đến sự can thiệp của con người. Xe tự hành AGV hay còn được gọi là Robot kéo hàng, Robot vận chuyển hàng tự động.

Loại xe vận chuyển này là một phần trong quá trình phát triển những nhà máy thông minh của các doanh nghiệp cũng như tự động hóa trong công nghiệp. Trên thị trường hiện nay, dòng xe AGV này được sử dụng để vận chuyển nguyên vật liệu trong nhiều ngành nghề khác nhau. Chẳng hạn như công nghiệp ô tô, Logistic, dược phẩm, điện tử, hàng tiêu dùng, y tế... Xe tự hành thay thế cho con người, nhưng với một mức độ làm việc năng suất hơn, tải trọng cao hơn. Có thể kể đến một số lợi thế khi sử dụng AGV trong các nhà máy:

- Giảm chi phí nhân công, công việc vận chuyển sẽ được thay thế bằng các AGV làm việc không ngừng nghỉ.
- An toàn trong lao động, giảm thiểu nguy cơ tai nạn lao động và chi phí sai sót của con người.
- Nâng cao năng suất lao động, với khả năng tải hàng lên đến vài trăm kg.
- Dễ dàng thay đổi và mở rộng diện tích làm việc.

1.2. Ứng dụng trong thực tế

1.2.1. Xe tự hành Pegasus ở Amazon

Để giải quyết vấn đề quá tải trong vận chuyển đơn hàng, sản phẩm trong kho bãi và đáp ứng được tốc độ, năng suất xử lý, công ty Amazon đã sử dụng các xe AGV trong quá trình xử lý hàng hóa và lưu kho tại các trung tâm phân phối của họ. Đây là một phần của việc tự động hóa quá trình xử lý hàng hóa và lưu kho của công ty. Các ứng dụng của AGV trong công ty Amazon bao gồm:



Hình 1.1. Xe tự hành ở Amazon

Vận chuyển hàng hóa: AGV được sử dụng để vận chuyển hàng hóa từ vị trí đóng gói đến vị trí lưu trữ trong kho. Việc sử dụng AGV giúp giảm thời gian và chi phí cho việc vận chuyển hàng hóa và tăng hiệu quả hoạt động của kho.

Định vị và vận chuyển sản phẩm: AGV được trang bị các hệ thống định vị và cảm biến để xác định vị trí của sản phẩm trong kho và di chuyển sản phẩm đến các khu vực lưu trữ tương ứng. Điều này giúp tăng tốc độ xử lý hàng hóa và giảm thời gian tìm kiếm sản phẩm.

Bên cạnh 300,000 nhân công đang làm việc ở các trung tâm kho bãi, Amazon công bố rằng 100,000 hệ thống Robot đã được phát triển và ứng dụng trên 25 trung tâm khắp đất nước Mỹ. Xe tự hành Pegasus cũng là một trong số đó, với độ dày khoảng 19cm nhưng xe có thể tải lên đến 560kg.



Hình 1.1.2. Xe tự hành ở Amazon thay thế hàng nghìn công nhân.

1.2.2. AGV của nhà máy GigaFactory của Tesla

Mặc dù nhà máy Tesla có hơn 10,000 nhân công khi vận hành, công ty vẫn trang bị các thiết bị tự động đầy đủ. Với phương châm “Máy móc xây dựng máy móc” (Machine building the machine), Tesla đã ứng dụng hàng loạt các máy móc như dây chuyền, cánh tay máy, ... Trong đó có xe tự hành AGV. Các AGV được giao nhiệm vụ di chuyển các sản phẩm pin năng lượng từ khu vực này đến khu vực khác. Các sản phẩm AGV của Tesla có đa dạng chủng loại và các chủng loại thường được phân loại theo cân nặng. Các sản phẩm có thể hoạt động từ 19-24 giờ liên tục và tải trọng cơ bản từ 60 – 100kg.

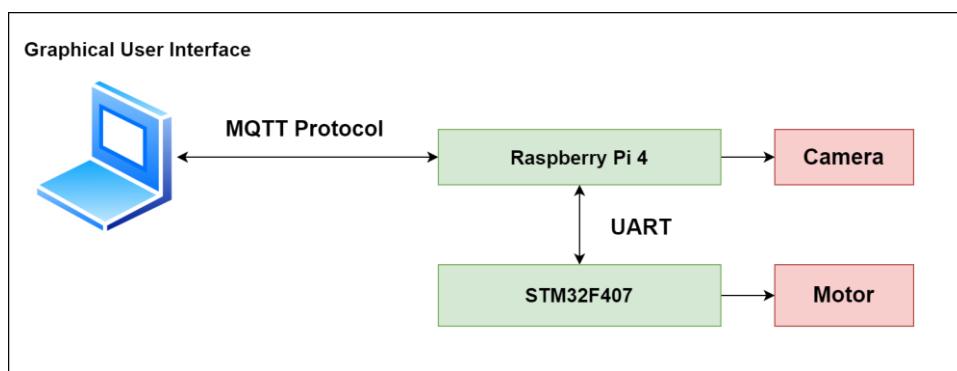


Hình 1.3. Xe tự hành của nhà máy Tesla

1.3. Mục tiêu đề tài

- Xây dựng mô hình xe tự hành trong công nghiệp ứng dụng giải pháp định vị và điều hướng dùng mã QR Code. Chứng minh được giải pháp có tính khả thi trong thực tế.
- Xây dựng được giao diện điều khiển giúp người dùng dễ dàng điều khiển được vị trí mong muốn và nhận được phản hồi chính xác từ xe. Điều chỉnh được vận tốc xe và các thông số khác từ xa.

1.4. Cấu trúc tổng quát của hệ thống



Hình 1.4. Cấu trúc tổng quan hệ thống

Cấu trúc của mô hình mô hình được chia ra ba phần chính:

- **Giao diện điều khiển:** là ứng dụng giúp người dùng có thể điều khiển và hiển thị các trạng thái phản hồi từ xe. Giao diện này được biết bằng Qt C++. Bên trong phần mềm sử dụng giao thức MQTT để phần mềm có thể giao tiếp với xe thông qua internet. Để sử dụng giao thức MQTT, giao diện sử dụng open source Mosquitto (một mã nguồn mở để lập trình client MQTT viết bằng C/C++).
- **Máy tính nhúng Raspberry Pi 4:** Máy tính nhúng được boot Raspberry Pi OS, là một distro của Linux, trên hệ điều hành này sinh viên tiến hành lập trình ngôn ngữ python với 3 tác vụ chính. Đầu tiên là truyền nhận MQTT để giao tiếp với giao diện người dùng, sau đó là tác vụ xử lý ảnh để quét mã QR Code, cuối cùng là giao tiếp UART để truyền nhận với vi điều khiển nhằm điều khiển tốc độ và vị trí mong muốn.
- **Vi điều khiển STM32F407:** Được lập trình với ngôn ngữ chính là C, vi điều

khiển được lập trình để nhận lệnh từ UART từ máy tính nhúng, sau đó tiến hành điều khiển động cơ với thuật toán PID và vận tốc hình thang.

1.5. Sơ lược nội dung luận văn

Luận văn gồm 6 chương:

Chương 1: Tổng quan về đề tài, cung cấp những thông tin cơ bản về xe AGV cũng như các ứng dụng thực tiễn của chúng, đồng thời nhóm sinh viên cũng giới thiệu mục tiêu và tổng quan về kiến trúc của mô hình.

Chương 2: Trình bày cơ sở lý thuyết của các bộ điều khiển, các thuật toán sử dụng trong đề tài như khái niệm về mã QR, các thuật toán xử lý ảnh, mô hình MQTT chi tiết.

Chương 3: Giới thiệu về quá trình thiết kế và xây dựng phần cứng, các nhóm sinh viên chia các khối nguồn, đấu nối các module và hình ảnh thực tế.

Chương 4: Nhóm sinh viên tiến hành trình bày các giải thuật điều khiển và các nhóm sinh viên lập trình.

Chương 5: Trình bày các kết quả thực nghiệm, đánh giá quá trình định vị và điều hướng của xe. Nhận xét về các quá trình thông qua các khoảng cách và vận tốc khác nhau.

Chương 6: Kết luận và định hướng phát triển cho đề tài.

Chương 2. CƠ SỞ LÝ THUYẾT

2.1. Mã QR

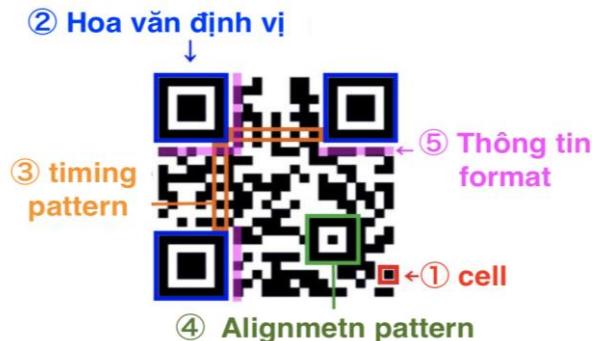
2.1.1. Khái niệm và cấu trúc mã QR

2.1.1.1. Khái niệm mã QR

QR code là kỹ thuật giúp chúng ta có thể đọc được thông tin khi quét bằng camera của điện thoại thông minh, tương tự như mã vạch. QR Code ra đời vào năm 1994 bởi công ty Denso Wave. So với mã vạch đáp ứng 20 ký tự thì QR Code đáp ứng được nhiều thông tin hơn, dữ liệu trong QR được lưu theo 2 chiều ngang dọc của một ma trận hình vuông, QR Code còn được gọi là 2D code.

2.1.1.2. Cấu trúc

Khi vừa nhìn vào, nhiều người sẽ nghĩ QR code là một hình vẽ được sắp xếp phức tạp không thể đọc hiểu được, tuy nhiên, QR code được cấu trúc dựa trên những nguyên tắc cơ bản được định sẵn.



Hình 2.1. Cấu trúc của một mã QR

- *Cell*: trong QR code có chứa nhiều ô hoa văn đen trắng, thực tế, các ô đen trắng này chứa các đoạn mã nhị phân. Các ô (cell) trắng đen này lần lượt mang giá trị 0 và 1, tập hợp các cell chính là các thông tin được lưu trữ vào QR code.
- *Hoa văn định vị*: Ở bốn góc của QR code bố trí các ô vuông gọi là hoa văn định vị. Nhờ vào hoa văn định vị này, camera có thể xác định được phạm vi QRcode cũng như đọc được thông tin ngay cả trong trường hợp QR code bị biến dạng, nhờ đó ta có thể quét được QR code một cách nhanh chóng ở bất

kỳ góc độ nào.

- *Timing pattern*: Các ô vuông đen trắng được đặt xen kẽ nhau nhằm giúp cho việc xác định tọa độ của QR code.
- *Alignment pattern*: Ở vùng phía dưới bên phải của QR code có một hình vuông chứa hình vuông nhỏ khác bên trong, hoa văn này có tác dụng quan trọng. Mục đích chính của mẫu căn chỉnh là đảm bảo rằng mã QR có thể quét chính xác, ngay cả khi nó bị méo hoặc bị che một phần.
- *Thông tin format*: Xung quanh hoa văn định vị là phần chứa thông tin format, quyết định mức độ sửa chữa lỗi của QR code. Khả năng sửa chữa lỗi của QR code được chia làm 4 mức độ: L, M, Q, H. Mức độ sửa chữa lỗi càng cao thì khả năng kháng lại lỗi rách, hỏng của QR code càng cao. Trong các trường hợp thông thường, QR code với mức độ sửa chữa lỗi M được sử dụng. Trong các môi trường QR code dễ bị bẩn, rách như công xưởng, công trường, code level Q hoặc H được sử dụng.

Bảng 2.1. Bảng đánh giá mức độ khôi phục của mã QR.

Mức độ sửa chữa lỗi của QR code	Độ khôi phục	Ứng dụng
Level L	Khoảng 7%	Sử dụng trong môi trường ít bị bẩn
Level M	Khoảng 15%	Sử dụng trong môi trường thông thường
Level Q	Khoảng 25%	Sử dụng trong môi trường dễ bị bẩn như xưởng sản xuất
Level H	Khoảng 30%	Sử dụng trong môi trường dễ bị bẩn như xưởng sản xuất

2.1.2. Các phương pháp định vị AGV

Trong bài toán định vị vị trí của xe tự hành, thông thường được chia ra 2 loại: vị trí tương đối và vị trí tuyệt đối, trong đó vị trí tương đối định vị bằng cách đo

quãng đường đi được từ điểm bắt đầu rồi tìm ra vị trí hiện tại (thường dùng IMU, Encoder, ...). Cách định vị này có khuyết điểm là sai số bị tích lũy theo thời gian. Chính vì thế để có vị trí chính xác, thông thường ta sẽ kết hợp với cách đo vị trí tuyệt đối (Gửi thông tin chính xác bằng các điểm được đánh dấu: QR Code, magnetic line, magnetic spot...).

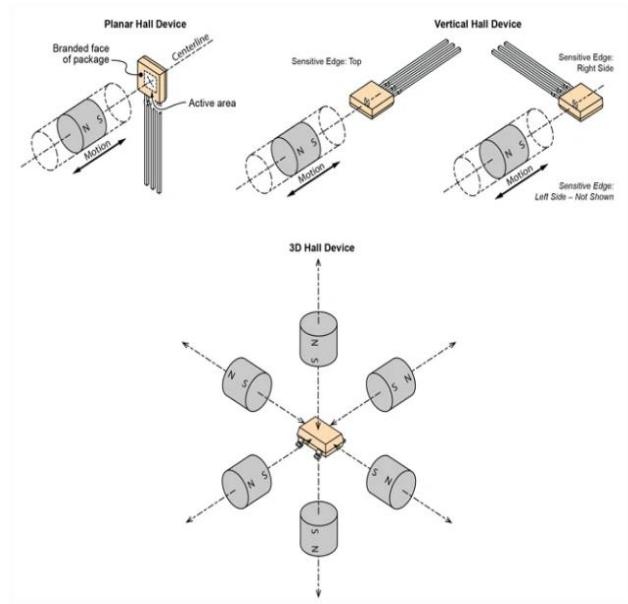
2.1.2.1. Dẫn đường bằng spot từ

Sử dụng cảm biến Hall để phát hiện viên nam châm có kích thước nhỏ được lắp đặt trên sàn và đo đặc sự phân bố mật độ từ thông của nam châm, từ đó tính toán quãng đường cần di chuyển của bánh xe (moving distances) và góc lái (steering angle).



Hình 2.2 Cảm biến bằng các spot từ

Là cảm biến tuyến tính có giá thành rẻ, hiệu quả cao. Cảm biến này có khả năng dò được chuyển động, vị trí hoặc sự thay đổi của từ. Cùng với đó, cảm biến này cũng ít tiêu tốn năng lượng. Hai thông số cần quan tâm về cảm biến Hall là “flux density” – “mật độ thông” (lượng từ đi qua) và các cực (the North and South poles).



Hình 2.3. Cách cảm biến từ hoạt động

2.1.2.2. Dẫn đường bằng đường từ

Với phương pháp dò đường này, AGV được hướng dẫn di chuyển thông qua các đường vật lý, như là băng keo từ (magnetic tape), dây cảm ứng (inductive wire) được lắp đặt dưới sàn.

Phương pháp này hoạt động theo cách sau: AGV sử dụng cảm biến hồng ngoại (đối với line màu sắc) hoặc cảm biến từ lắp theo module (line từ) dò đường line trên sàn. Tiếp đó, đo đặc sai số hướng di chuyển và sử dụng thông tin đó để điều chỉnh quỹ đạo của robot.



Hình 2.4. Cảm biến bằng đường từ.

2.1.2.3. Laser

Điều hướng bằng laser là một trong những công nghệ phổ biến nhất được sử dụng trong AGV để điều hướng chính xác và đáng tin cậy. Sử dụng máy quét laser được gắn trên đầu xe để tạo ra một bản đồ của môi trường xung quanh. Máy quét phát ra tia laser và phản xạ từ các vật thể trong môi trường, sau đó trở lại. Máy quét sau đó sử dụng thông tin này để tạo ra một bản đồ chi tiết về môi trường, bao gồm các chướng ngại vật, tường và các đối tượng khác.

Sau khi bản đồ được tạo ra, AGV sử dụng thông tin này để lập kế hoạch cho đường đi của mình qua môi trường. AGV liên tục cập nhật vị trí của mình và điều chỉnh đường đi dựa trên các thay đổi trong môi trường, chẳng hạn như vật thể di chuyển hoặc chướng ngại vật mới.

Điều hướng bằng laser cung cấp nhiều lợi ích hơn so với các công nghệ điều hướng khác. Nó rất chính xác, với độ chính xác vị trí trong vài milimet. Nó đáng tin cậy, ngay cả trong môi trường khó khăn với ánh sáng yếu hoặc mức độ bụi bẩn và mảnh vụn cao.

Một lợi thế khác của điều hướng bằng laser là tính linh hoạt của nó. Máy quét laser có thể được gắn ở nhiều vị trí khác nhau trên AGV, bao gồm phía trước, phía sau hoặc trên đầu xe. Điều này cho phép AGV điều hướng trong một loạt các môi trường và xử lý một loạt các nhiệm vụ.



Hình 2.5. Cảm biến bằng Laser.

2.1.2.3. RFID

RFID là viết tắt của "Radio-Frequency Identification", một công nghệ sử dụng các trường điện từ để tự động nhận diện và theo dõi các thẻ được gắn vào các đối tượng. Trong bối cảnh của xe AGV, công nghệ RFID thường được sử dụng để cải thiện độ chính xác và hiệu quả của quá trình điều hướng.

Công nghệ RFID thường được triển khai theo một trong hai cách: như một phương tiện để xác định vị trí của xe AGV, hoặc như một phương tiện để theo dõi và quản lý kho.

Trong trường hợp theo dõi vị trí, các thẻ RFID được đặt tại các điểm chiến lược dọc theo đường đi của xe AGV, chẳng hạn như ở các giao lộ, góc cua hoặc các điểm quan trọng khác. Đầu đọc RFID trên xe AGV phát hiện các thẻ khi nó đi qua, cho phép phần mềm trên xe xác định vị trí của nó trong thời gian thực.



Hình 2.6. Định vị bằng RFID.

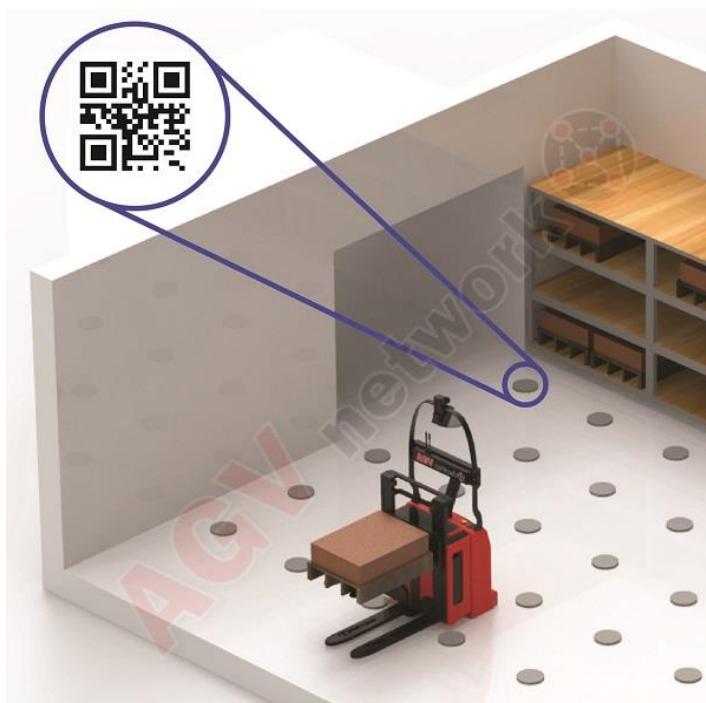
2.1.2.4. QR code

Mã QR là một mã vạch hai chiều có thể lưu trữ một lượng lớn dữ liệu. Trong các ứng dụng AGV, công nghệ mã QR thường được sử dụng cho mục đích điều hướng. Các mã QR có thể được đặt tại các vị trí khác nhau trên tuyến đường của AGV, chẳng hạn như tại các giao lộ hoặc dưới sàn nhà, để cung cấp thông tin về vị trí và hướng dẫn AGV đến đích của nó.

AGV được trang bị một máy ảnh hoặc máy quét có thể đọc mã QR, và thông tin được mã hóa trong mã QR được sử dụng để xác định vị trí và đích của

AGV. Phần mềm AGV có thể tính toán tuyến đường hiệu quả nhất để đến đích và điều khiển các chuyển động của AGV tương ứng.

Điều hướng bằng QR code rất có lợi trong các ứng dụng AGV, vì nó cung cấp một phương tiện điều hướng và xác định danh tính đáng tin cậy và tiết kiệm chi phí. Nó cũng có thể được tích hợp dễ dàng với các hệ thống quản lý kho và các công nghệ khác, làm cho nó trở thành một giải pháp đa năng cho nhiều loại hoạt động khác nhau.



Hình 2.7. Định vị bằng mã QR.

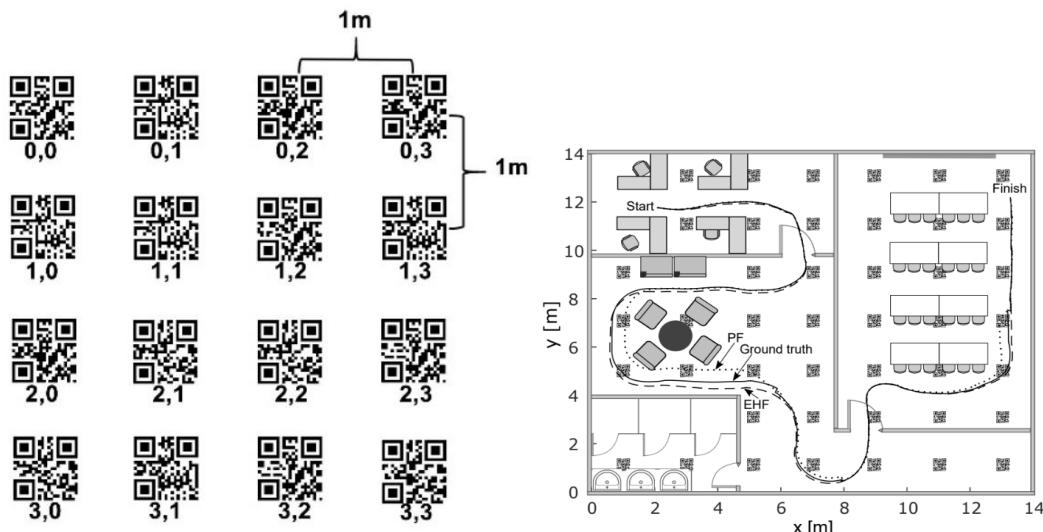
2.1.3. So sánh giữa các phương thức định vị tuyệt đối.

Bảng 2.2. Bảng so sánh các phương pháp định vị tuyệt đối

Loại định vị	Độ chính xác	Giá thành	Lắp đặt và Bảo trì
Spot từ	Độ chính xác cao nhưng cần thuật toán tốt.	Module dò từ có giá thành từ 3.000.000 vnd, các spot từ có giá thành cao.	Không có chi phí bảo trì (không hao mòn) nhưng khó khăn trong việc thay đổi. Khó khăn trong lắp đặt và thay thế.
Đường từ	Độ chính xác cao, thuật toán không quá phức tạp.	Các module cảm biến từ có giá thành từ 3.000.000 vnd, băng keo từ có giá thành bán lẻ từ 65.000vnd/1m	Dễ dàng lắp đặt và bảo trì.
QR Code	Độ chính xác cao, cần thuật toán tốt.	Camera xử lý ảnh/QR Code có giá thành từ 300.000vnd. Các máy quét mã vạch có giá từ 500.000vnd/ một sản phẩm.	Lắp đặt dễ dàng, cần có các biện pháp bảo vệ mã QR trong môi trường công nghiệp.

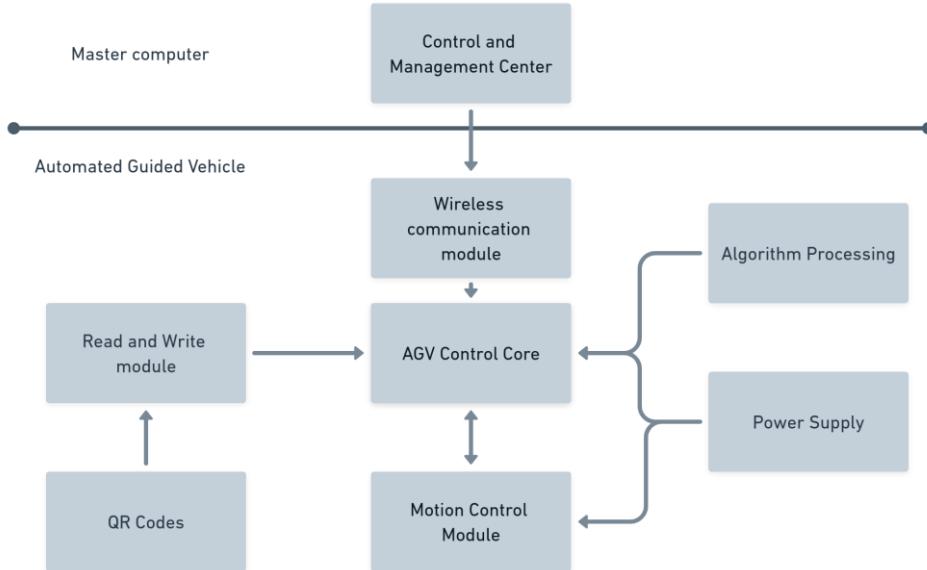
2.2. Điều hướng và định vị bằng mã QR trong điều khiển AGV

Bước đầu tiên để dẫn đường xe AGV là dán các mã QR Code trên nền đất theo dạng một ma trận, các mã QR Code này chứa dữ liệu bao gồm: Thông tin vị trí của xe trên ma trận QR Code dán trên đường đi.



Hình 2.8. Các xe định vị bằng ma trận mã QR.

Khi Xe AGV đi qua các khu vực, nó sẽ dùng camera hoặc các module Scan và phân tích mã code 2D này để lấy dữ liệu tiếp tục hành trình. Dữ liệu này được đưa về máy chủ qua WIFI hoặc các kết nối không dây khác. Một trong số các thuật toán thường được dùng là dựa vào dữ liệu AGV thu được từ mã QR Code, chính máy tính nhúng hoặc hoặc các thiết bị gắn trên AGV sẽ giao tiếp xuống bộ phận thực thi, trong phương pháp này, máy chủ chỉ đóng vai trò là trạm monitor và gửi các lệnh setup khi cần thiết.



Hình 2.9. Sơ đồ khái niệm xe định vị bằng QR Code.

2.3. Thuật toán xử lý ảnh

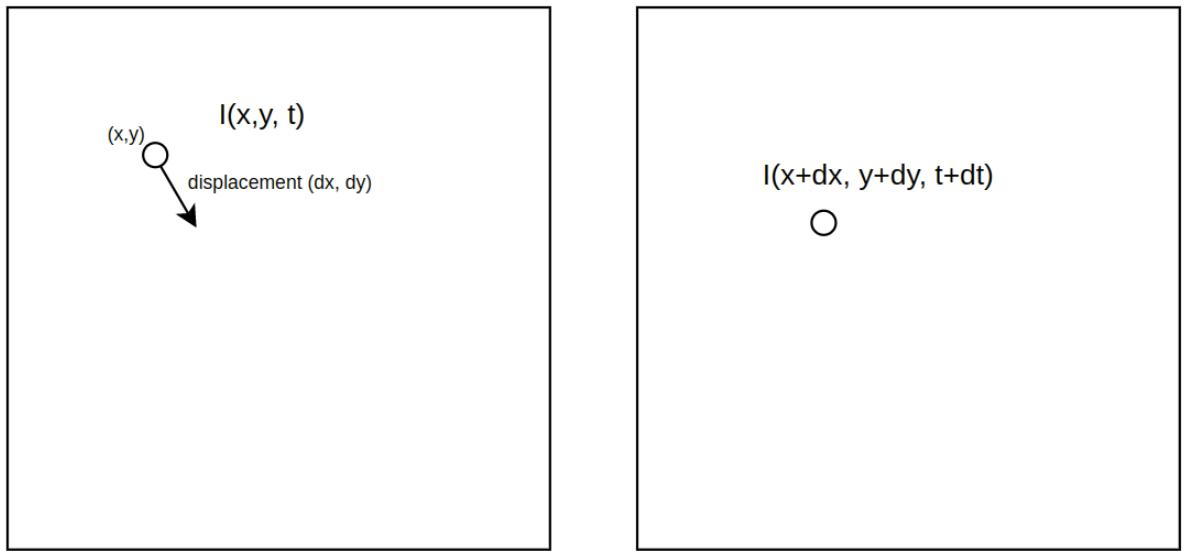
2.3.1. Giới thiệu thuật toán Optical Flow

Optical Flow là một thuật toán trong xử lý ảnh, được sử dụng để ước tính vận tốc của các điểm trong khung hình. Thuật toán này sử dụng các pixel của hai khung hình liên tiếp để tìm ra sự thay đổi vị trí của chúng và tính toán vận tốc chuyển động của các pixel đó. Kết quả của thuật toán Optical Flow có thể được sử dụng để phát hiện chuyển động, theo dõi đối tượng.

Cách hoạt động của thuật toán Optical Flow là sử dụng tính toán gradient của các điểm ảnh trên hai khung hình liên tiếp trong chuỗi hình ảnh. Gradient này sẽ giúp tính toán được sự thay đổi của các điểm ảnh theo thời gian.

Trong luận văn của chúng tôi, bài toán là xử lý các frame từ video của camera. Để đơn giản, ta xét 2 frames kế tiếp, rồi tính toán tương tự cho một chuỗi frame.

Xét sự chuyển động của một pixel giữa 2 frame liên tiếp trong 1 đoạn video. Hình ảnh dưới mô tả sự chuyển động của điểm pixel có tọa độ (x, y) ở thời điểm t đến thời điểm $t + 1$, điểm pixel này dịch chuyển đến tọa độ $(x + dx, y + dy)$ và vector (dx, dy) chính là vector mô tả sự dịch chuyển đó.



Hình 2.10. Tuật toán Optical Flow.

Hình X. Pixel trong 2 frame kế tiếp tại thời điểm t và $t + dt$

Vì thuật toán Optical Flow giả sử pixel dịch chuyển là nhỏ trong khoảng thời gian dt và mức xám là không thay đổi, cho nên:

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \quad (1)$$

$$I(x + dx, y + dy, t + dt) = I(x, y, t) + \frac{\partial I}{\partial x} dx + \frac{\partial I}{\partial y} dy + \frac{\partial I}{\partial t} dt \quad (2)$$

Từ (1) và (2), ta có:

$$\frac{\partial I}{\partial x} dx + \frac{\partial I}{\partial y} dy + \frac{\partial I}{\partial t} dt = 0$$

Chia 2 vế cho dt , ta được:

$$\frac{\partial I}{\partial x} v_x + \frac{\partial I}{\partial y} v_y + \frac{\partial I}{\partial t} = 0$$

2.3.2. Ứng dụng phương pháp Lucas-Kanade vào xử lý QR code

Có nhiều phương pháp để trực quan hóa kết quả của thuật toán Optical Flow, như Lucas-Kanade, Horn-Schunck, Buxton-Buxton, Black-Jepson, ... Bài toán của chúng tôi là bài toán real-time, cho nên thuật toán Lucas-Kanade được lựa chọn.

Ưu điểm của thuật toán Lucas-Kanade là nó rất nhanh và tối ưu, có thể hoạt động với tốc độ cao ngay cả với các video có độ phân giải cao. Ngoài ra, thuật toán

này cũng rất ổn định và độ chính xác cao, giúp đưa ra kết quả tracking chính xác hơn. Thuật toán Lucas-Kanade cũng có khả năng đối phó với các thay đổi về ánh sáng, màu sắc, hình dạng và quang cảnh, giúp cho việc tracking vật thể trở nên linh hoạt và chính xác hơn trong nhiều điều kiện khác nhau.

Phương pháp Lucas-Kanade dựa trên giả định rằng các điểm ảnh trong vùng xung quanh điểm ảnh cần ước tính đều có cùng tốc độ chuyển động. Phương pháp này được đặt tên theo hai nhà toán học Robert Lucas và Takeo Kanade.

Phương pháp Lucas-Kanade giải quyết vấn đề ước tính Optical Flow bằng cách tối thiểu hóa sai số giữa đạo hàm riêng của hàm mức xám và tốc độ chuyển động của các điểm ảnh trong vùng quan sát. Xét các điểm ảnh trong vùng quan tâm, kết hợp phương trình tổng quát tìm được trong thuật toán Optical Flow:

$$I_x(p_1)v_x + I_y(p_1)v_y = -I_t(p_1)$$

$$I_x(p_2)v_x + I_y(p_2)v_y = -I_t(p_2)$$

...

$$I_x(p_n)v_x + I_y(p_n)v_y = -I_t(p_n)$$

Từ đây, ta suy ra:

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \dots & \dots \\ I_x(p_n) & I_y(p_n) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} -I_t(p_1) \\ -I_t(p_2) \\ \dots \\ -I_t(p_n) \end{bmatrix}$$

Để tìm được vector trượt (v_x, v_y) có nhiều cách, trong trường hợp này được sử dụng phương pháp bình phương tối thiểu để giải quyết.

$$Av = b$$

$$A^T A v = A^T b$$

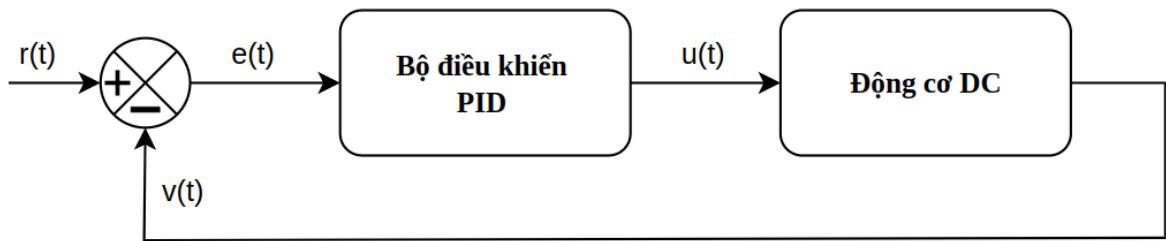
$$v = (A^T A)^{-1} A^T b$$

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x^2(p_i) & \sum_i I_x(p_i)I_y(p_i) \\ \sum_i I_x(p_i)I_y(p_i) & \sum_i I_y^2(p_i) \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(p_i)I_t(p_i) \\ -\sum_i I_y(p_i)I_t(p_i) \end{bmatrix}$$

2.4. Quy luật điều khiển vận tốc

2.4.1. Bộ điều khiển PID

Vận dụng thuật toán bộ điều khiển PID trong việc điều khiển động cơ DC, sử dụng hàm truyền của bộ điều khiển PID rời rạc từ đó tìm được phương trình điều khiển.



Hình 2.11. Sơ đồ bộ điều khiển PID.

Trong đó:

- r : Giá trị đặt
- v : Giá trị hiện tại
- e : Sai số giá trị đặt và giá trị hiện tại $e = r - v$

Ta có hàm truyền của bộ điều khiển PID rời rạc có dạng như sau:

$$G(z) = \frac{U(z)}{E(z)} = K_p + \frac{K_I T}{2} \frac{z+1}{z-1} + \frac{K_D}{T} \frac{z-1}{z}$$

Viết lại phương trình trên ta được:

$$G(z) = \frac{\left(K_p + \frac{K_I T}{2} + \frac{K_D}{T}\right) + \left(-K_p + \frac{K_I T}{2} - 2\frac{K_D}{T}\right)z^{-1} + \frac{K_D}{T}z^{-2}}{1 - z^{-1}}$$

Đặt:

$$a_0 = K_p + \frac{K_I T}{2} + \frac{K_D}{T}$$

$$a_1 = -K_p + \frac{K_I T}{2} - 2\frac{K_D}{T}$$

$$a_2 = \frac{K_D}{T}$$

Ta suy ra:

$$G(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 - z^{-1}}$$

Từ đó, ta tính được giá trị ngõ ra của bộ điều khiển PID $u(k)$ khi giá trị ngõ vào là $e(k)$ như sau:

$$u(k) = G(z)e(k) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 - z^{-1}} e(k)$$

Thay các giá trị a_0 , a_1 và a_2 vào phương trình trên và áp dụng tính chất dời thời gian theo biến đổi Z ta có:

$$\begin{aligned} u(k) &= u(k-1) + (K_P + \frac{K_I T}{2} + \frac{K_D}{T})e(k) + (-K_p + \frac{K_I T}{2} - \frac{-2K_D}{T})e(k-1) \\ &\quad + \frac{K_D}{T}e(k-2) \end{aligned}$$

Viết lại phương trình trên ta tìm được giá trị tại các khâu P, I và D như sau:

$$P_{Part} = K_P(e(k) - e(k-1))$$

$$I_{Part} = \frac{K_I T}{2}(e(k) + e(k-1))$$

$$D_{Part} = \frac{K_D}{T}(e(k) - 2e(k-1) + e(k-2))$$

Từ đó ta có được phương trình bộ điều khiển PID ròng rạc dùng cho điều khiển động cơ DC như sau:

$$u(k) = u(k-1) + P_{Part} + I_{Part} + D_{Part}$$

Điều khiển tỉ lệ (K_P) có ảnh hưởng làm giảm thời gian lên và sẽ làm giảm nhưng không loại bỏ sai số xác lập. Điều khiển tích phân (K_I) sẽ loại bỏ sai số xác lập nhưng có thể là đáp ứng quá độ xấu đi. Điều khiển vi phân (K_D) có tác dụng là tăng sự ổn định của hệ thống, giảm vọt lố và cải thiện đáp ứng quá độ. Ảnh hưởng

của mỗi bộ điều khiển K_p , K_I và K_D lên hệ thống vòng kín được tóm tắt ở bảng dưới đây.

Bảng 2.3. Bảng đánh giá các thông số ảnh hưởng lên PID

Đáp ứng vòng kín	Thời gian lên	Vọt lố	Thời gian xác lập	Sai số xác lập
Kp	Giảm	Tăng	Thay đổi nhỏ	Giảm
Ki	Giảm	Tăng	Tăng	Loại bỏ
Kd	Thay đổi nhỏ	Giảm	Giảm	Thay đổi nhỏ

2.4.2. Ứng dụng vận tốc hình thang vào điều khiển xe

2.3.2.1. Vận tốc hình thang

Trong điều khiển xe (robot), vận tốc hình thang (trapezoidal velocity) là một phương pháp điều khiển chuyển động của xe để di chuyển từ một vị trí đến vị trí khác với tốc độ và gia tốc được giữ ở mức ổn định, nhằm giảm thiểu sai số và dao động trong quá trình di chuyển. Nó cho phép xe di chuyển với tốc độ tăng dần lên một mức tối đa, giữ vận tốc ở mức đó trong một khoảng thời gian và sau đó giảm tốc độ xuống một cách dần dần để đến vị trí đích một cách chính xác và ổn định hơn.

Công thức vận tốc hình thang được biểu diễn bằng các thông số sau:

- Tổng quãng đường di chuyển: s
- Vận tốc tối đa: v_{max}
- Gia tốc: a
- Thời gian tăng tốc: t_1
- Thời gian giảm tốc: t_2
- Thời gian di chuyển với vận tốc tối đa t_3
- Tổng thời gian di chuyển: $T = t_1 + t_2 + t_3$

Công thức tính toán vận tốc hình thang được mô tả như sau:

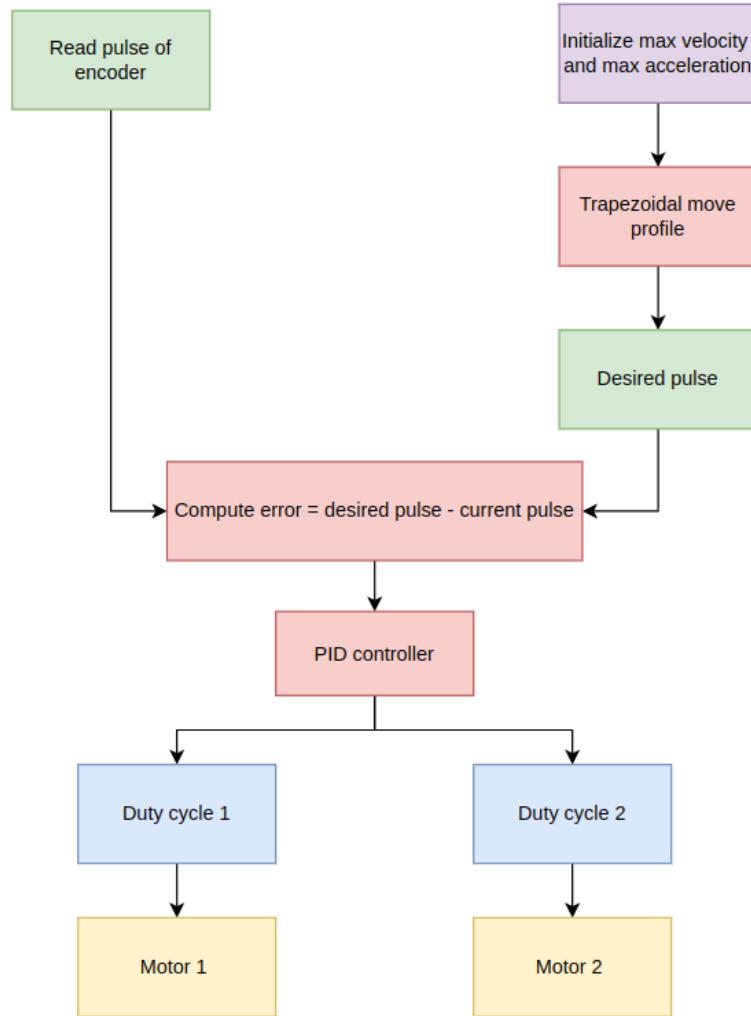
- Quãng đường di chuyển trong giai đoạn tăng tốc: $s_1 = \frac{1}{2}at_1^2$
- Vận tốc tại cuối giai đoạn tăng tốc và đầu giai đoạn giảm tốc: $v_1 = at_1$
- Thời gian giảm tốc: $t_2 = \frac{v_{1max}}{a}$
- Quãng đường di chuyển trong giai đoạn giảm tốc: $s_2 = \frac{1}{2}at_2^2$
- Quãng đường di chuyển ở vận tốc tối đa: $s_3 = v_{3max}$

Mô tả cách hoạt động của vận tốc hình thang:

- Trong giai đoạn tăng tốc, vận tốc của xe tăng từ 0 đến v_1 theo một đường thẳng với gia tốc là a . Do đó, quãng đường di chuyển trong giai đoạn này được tính bằng công thức $s_1 = \frac{1}{2}at_1^2$
- Sau khi vận tốc đạt đến v_1 , xe di chuyển với vận tốc tối đa v_{max} trong một khoảng thời gian t_3 . Trong giai đoạn này, quãng đường di chuyển được tính bằng công thức $s_3 = v_{3max}$.
- Khi xe tiếp cận đến điểm kết thúc của hành trình, nó cần giảm tốc để dừng lại ở vị trí mong muốn. Do đó, trong giai đoạn giảm tốc, xe giảm tốc độ từ v_{max} xuống 0 theo một đường thẳng với gia tốc là a . Quãng đường di chuyển trong giai đoạn này được tính bằng công thức $s_2 = \frac{1}{2}at_2^2$

2.3.2.2. Kết hợp vận tốc hình thang và bộ điều khiển PID

Vận tốc hình thang là một phương pháp điều khiển chuyển động đơn giản và hiệu quả cho robot di chuyển trên một đường thẳng. Tuy nhiên, nó có thể gây ra dao động và lỗi ở các điểm đầu cuối của quãng đường di chuyển, do đó để cải thiện độ chính xác và ổn định của robot trong quá trình di chuyển, ta có thể kết hợp phương pháp điều khiển vận tốc hình thang với bộ điều khiển PID.

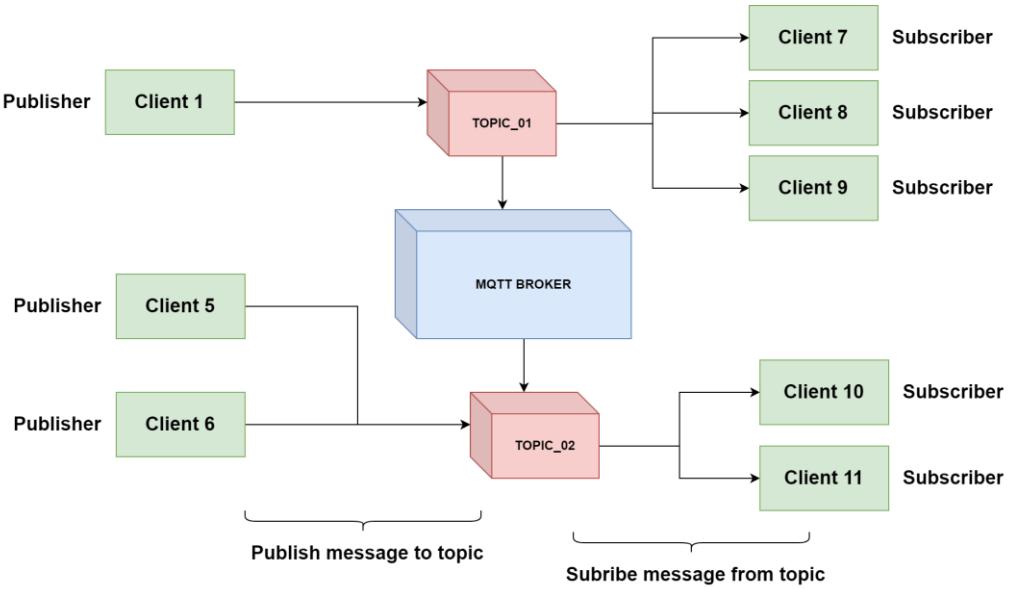


Hình 2.12. Mô hình vận hành code trong vi điều khiển.

2.5. Giao tiếp điều khiển MQTT

2.5.1. Khái niệm và các thông số cơ bản của MQTT

Giao thức MQTT (Message Queuing Telemetry Transport) là một giao thức truyền thông nhẹ và độc lập nền tảng được thiết kế cho các ứng dụng IoT (Internet of Things) và M2M (Machine to Machine). MQTT sử dụng mô hình đăng ký và đăng bài viết để truyền tải các thông tin dạng tin nhắn (message) giữa các thiết bị. Dưới đây là các định nghĩa chi tiết của MQTT:

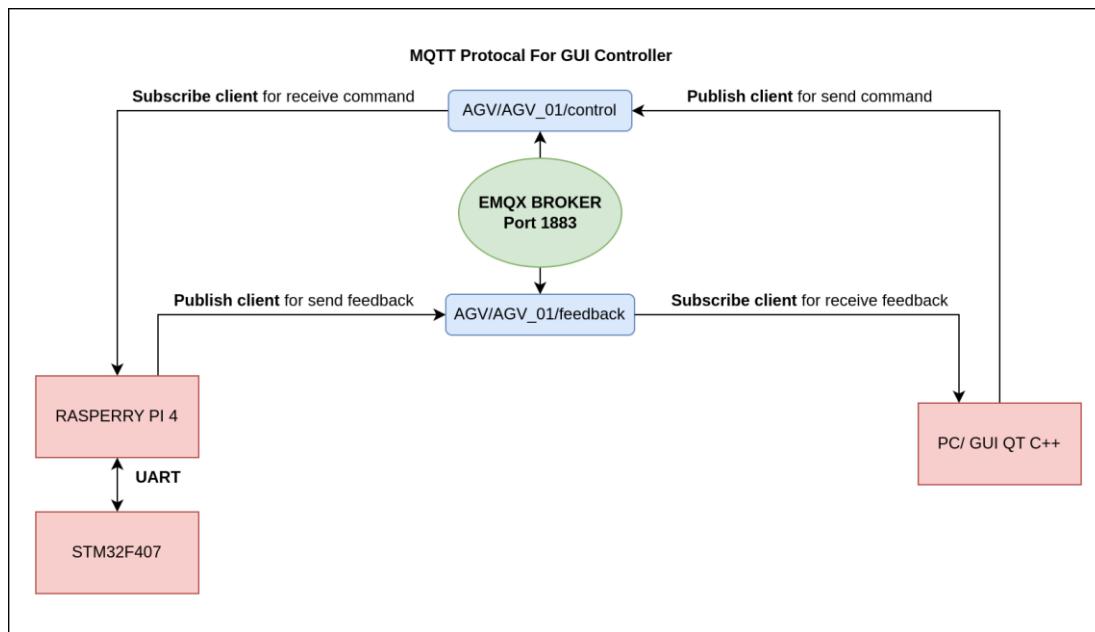


Hình 2.13. Mô hình MQTT.

- Thiết bị MQTT: MQTT sử dụng các thiết bị (device) để truyền tải và nhận các thông tin. Thiết bị MQTT có thể là các cảm biến, bộ điều khiển hoặc các ứng dụng khác.
- Mô hình Publisher and Subscriber: MQTT sử dụng mô hình subscribe và publish message để truyền tải thông tin giữa các thiết bị. Các thiết bị có thể đăng ký vào các kênh (topic) để nhận các thông tin liên quan đến kênh đó.
- Kênh (Topic): Kênh trong MQTT là nơi các thiết bị đăng ký để nhận thông tin. Các thông tin được truyền tải thông qua các kênh được định danh bởi các chuỗi ký tự (topic string). Kênh cũng có thể được phân cấp để quản lý thông tin.
- Tin nhắn (Message): MQTT sử dụng tin nhắn (message) để truyền tải các thông tin giữa các thiết bị. Tin nhắn MQTT có thể là bất kỳ định dạng nào, từ dữ liệu văn bản đến các định dạng dữ liệu phức tạp như JSON hoặc XML.
- Broker MQTT: Broker MQTT là trung tâm của hệ thống MQTT. Nó nhận tin nhắn từ các thiết bị đăng bài viết và chuyển đến các thiết bị đăng ký vào kênh tương ứng. Broker cũng có thể lưu trữ thông tin để cho các thiết bị đăng ký sau này có thể nhận được thông tin đã được đăng trên kênh trước đó.
- QoS (Quality of Service): MQTT hỗ trợ ba mức độ QoS để đảm bảo tin nhắn được truyền tải chính xác và đáng tin cậy giữa các thiết bị. Ba mức độ QoS

của MQTT là QoS 0 (chỉ gửi tin nhắn một lần), QoS 1 (gửi tin nhắn ít nhất một lần) và QoS 2 (gửi tin nhắn chính xác một lần).

2.5.2. Áp dụng MQTT vào mô hình điều khiển xe từ xa



Hình 2.14. Mô hình MQTT ứng dụng trong điều khiển từ xa.

Mô hình điều khiển xe được thiết kế với hai clients, trong đó một client là xe và client còn lại là giao diện điều khiển người dùng. Mỗi client điều đóng vai trò là publisher và subscriber trong 2 topic khác nhau.

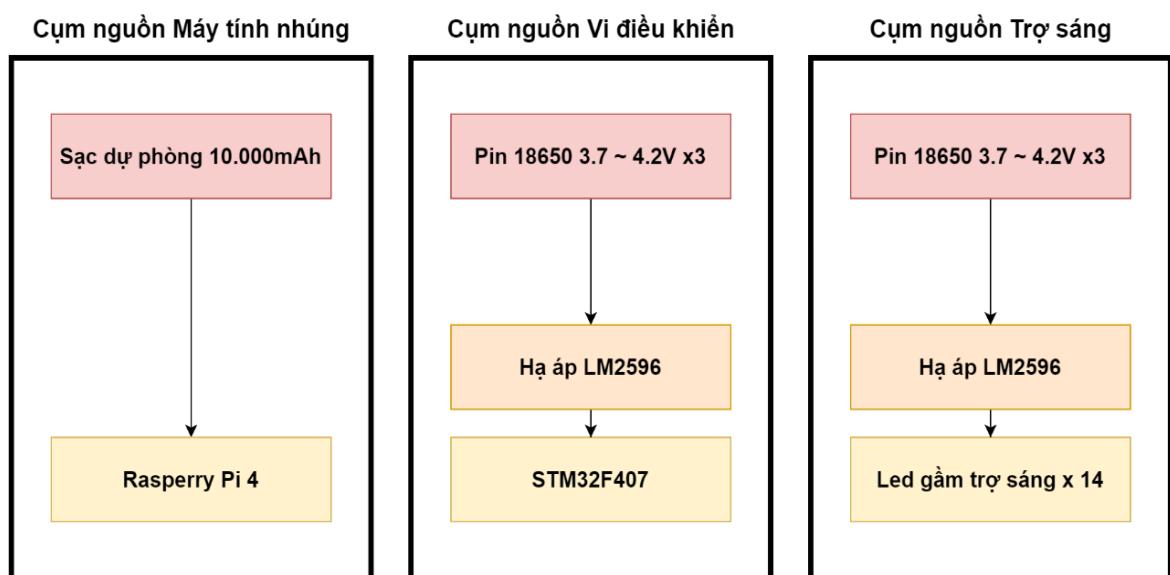
- Topic “**AGV/<AGV_ID>/control**”: Topic này được giao diện người dùng đóng vai trò là publisher. Người dùng có thể gửi các mã lệnh tự định nghĩ từ phần mềm đến xe AGV thông qua topic này. Ngược lại, xe AGV đóng vai trò là một subscriber đợi nhận tin nhắn câu lệnh. Sau khi nhận lệnh, xe AGV xe phân tích và gửi các lệnh điều khiển vận tốc và vị trí xuống STM32.
- Topic “**AGV/<AGV_ID>/feedback**”: Topic này là topic có vai trò nhận các tin nhắn phản hồi trạng thái từ xe AGV và gửi đến giao diện người dùng. Với sự phân chia này, giao diện người dùng đóng vai trò là subscriber đăng ký vào topic feedback, còn xe AGV đóng vai trò là publisher gửi tin nhắn trạng thái.

Chương 3. THIẾT KẾ VÀ THI CÔNG PHẦN CỨNG

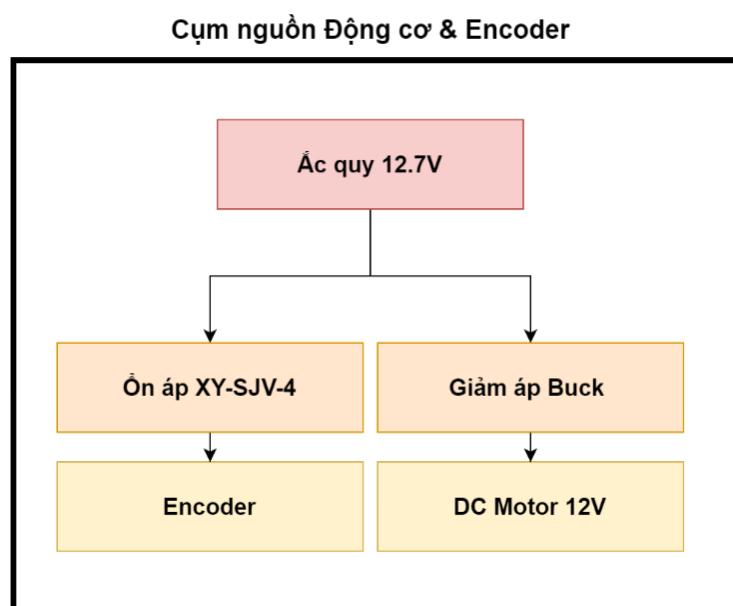
3.1. Sơ đồ kết nối phần cứng.

3.1.1. Sơ đồ nguồn

Cụm nguồn được chia thành 4 khối chính với 3 loại nguồn khác nhau, 4 khối được thiết kế tách biệt để tránh các trường hợp rò rỉ cũng như điện áp trả về ngoài ý muốn từ động cơ. Sơ đồ nguồn được mô tả theo mô hình sau:



Hình 3.1. Các cụm nguồn dành cho bộ phận điều khiển và trợ sáng

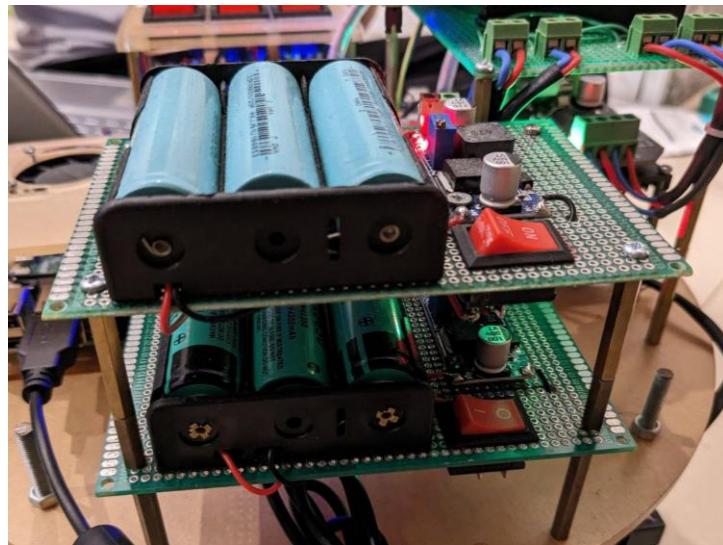


- Cụm nguồn máy tính nhúng: Máy tính nhúng Raspberry Pi 4 là một module có giá trị cao, kèm với đó máy tính này cũng đòi hỏi một nguồn type C có chất lượng ổn định từ 15W trở xuống. Để đáp ứng được điều này, nhóm sinh viên chọn nguồn sạc dự phòng để cấp cho máy tính nhúng.
- Cụm nguồn vi điều khiển: Vì điều khiển yêu cầu nguồn cấp 3.3V đến 5V, để tránh các trường hợp rò rỉ các dòng ngoài ý muốn, vi điều khiển được cấp nguồn riêng với 3 pin 18650, điện áp trung bình 12V, điện áp này đi qua module giảm áp xuống 5V.



Hình 3.3. Cụm 3 pin 18650.

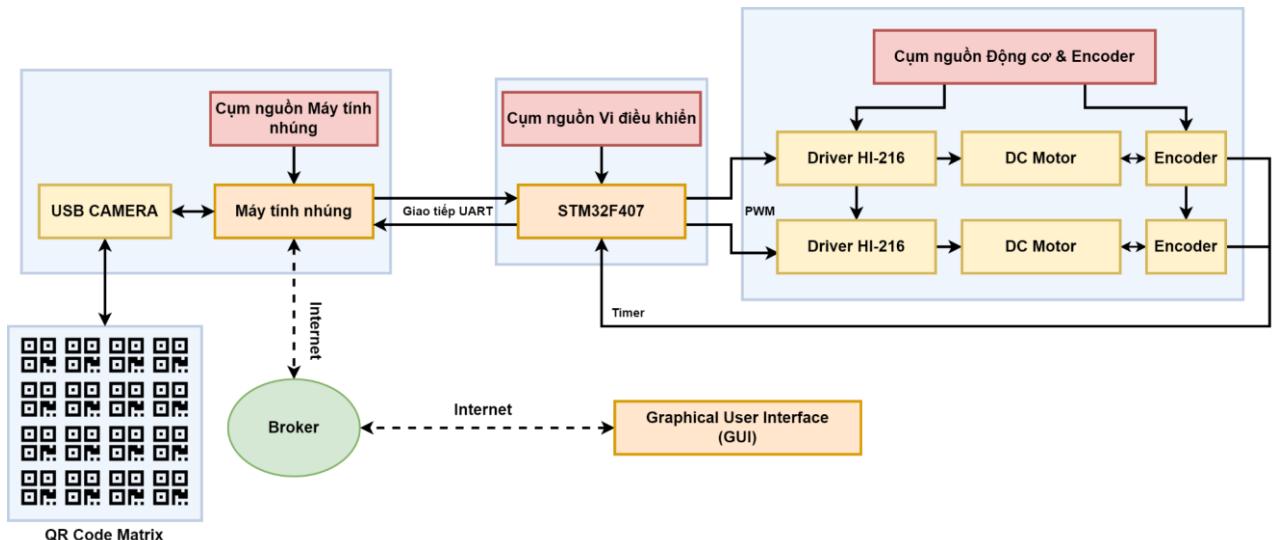
- Cụm nguồn trợ sáng led gầm: Để tăng tính ổn định trong môi trường thiếu sáng, dưới gầm xe được trang bị 2 module led bao gồm 14 led, các đèn này được cấp nguồn 5V từ nguồn 3 pin 18650 12V qua giảm áp để hoạt động trợ sáng cho xe trong lúc vận hành.



Hình 3.4. Pin 18650 được xếp thành module 2 tầng.

- Cụm nguồn động cơ và Encoder: Động cơ và encoder luôn hoạt động song song nhau, đồng thời cần đủ lớn để xe hoạt động lâu dài, nên nhóm sinh viên đã chọn người ắc quy 12.7V. Điện áp này qua giảm áp còn 12V cho DC Motor, đồng thời cũng qua ổn áp 5V cho Encoder.

3.1.2. Sơ đồ kết nối các thành phần



Hình 3.5. Sơ đồ kết nối các thành phần với nhau.

Sơ đồ kết nối phần cứng với 2 cụm trung tâm là máy tính nhúng, STM32F407. Qua đó, hệ thống kết nối các module như sau:

- Giao diện giao tiếp với Raspberry thông qua Internet bằng giao thức MQTT.
- Raspberry nhận lệnh và bắt đầu công việc phân tích mã QR, định vị, xác định

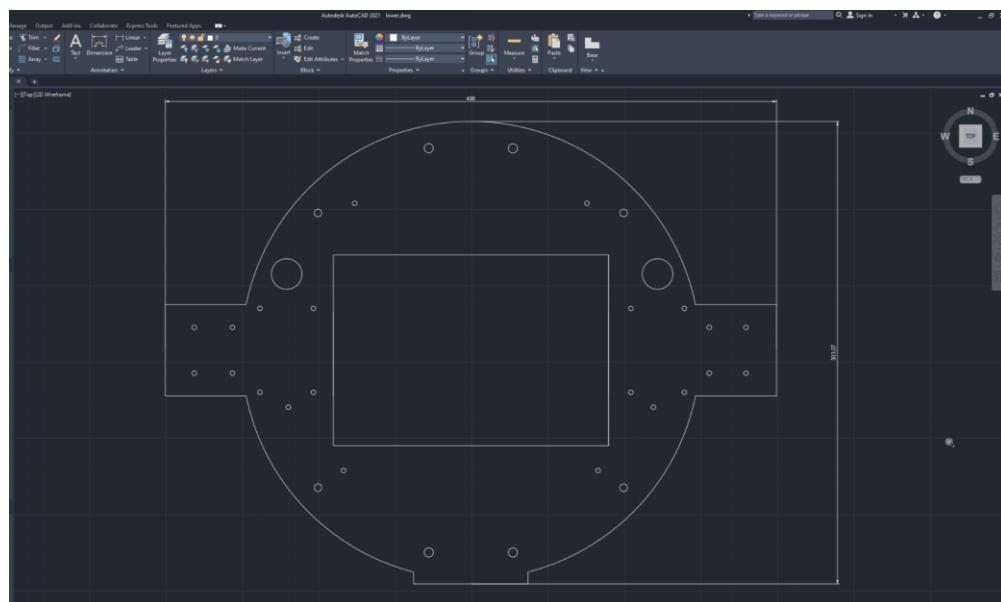
vận tốc và góc quay cần thiết, “nén” thông tin thành frame truyền UART.

- Vi điều khiển kết nối và giao tiếp với Raspberry Pi 4 thông qua cổng giao tiếp UART. Cả 2 bên đều giao tiếp hai chiều Tx và Rx để thuận tiện cho việc “send command” và “feedback status”.
- Vi điều khiển điều khiển động cơ với thông tin feedback từ 2 encoder kèm động cơ.

3.1.3. AutoCad

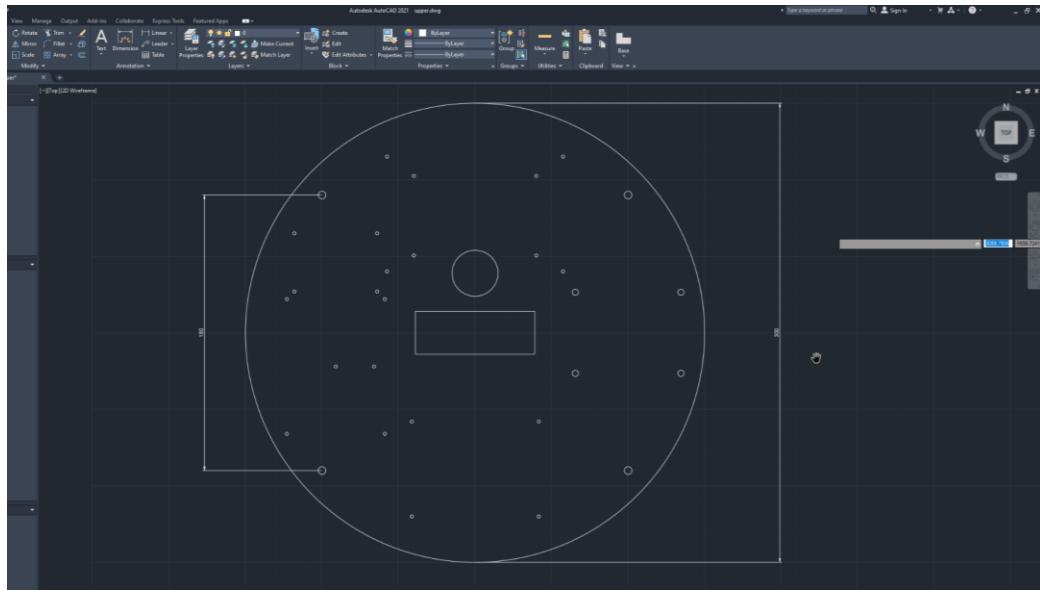
Một trong những phần quan trọng nhất của xe là bản vẽ kỹ thuật để có thể đấu nối các module theo định hình và cố định chắc chắn. Để có thể hoàn thiện bước cuối là liên kết các cụm, nhóm sinh viên đã thiết kế xe với 2 tầng bằng AutoCad, sau khi có bản vẽ, nhóm sinh viên đã gia công bảng vẽ với mica trong suốt, độ dày 8mm. Dưới đây là bản vẽ AutoCad.

- Bản vẽ tầng 1 với các thông số đường kính 30cm, khoảng cách 2 bánh xe là 40cm kèm các lỗ ốc vít.



Hình 3.6. Bản vẽ AutoCad tầng 1.

- Bản vẽ tầng 2 với thông số bán đường kính 30cm, khoảng cách 4 ốc trụ nối 2 tầng là 18cm:



Hình 3.7. Bảng vẽ AutoCad tầng 2.

3.2. Tổng quan các thiết bị

3.2.1. Vi xử lý STM32F407

Vi điều khiển nhúng STM32F407 là một trong những loại vi điều khiển nhúng (microcontroller) thuộc họ STM32 của STMicroelectronics. Nó là một vi điều khiển ARM Cortex-M4F với tần số hoạt động tối đa 168 MHz và được tích hợp sẵn các tính năng như:

- Bộ nhớ Flash lên đến 1MB, bộ nhớ SRAM 192 KB.
- DMA (Direct Memory Access).
- 3 bộ chuyển đổi ADC (Analog-to-Digital Converter) 12-bit.
- 2 DAC (Digital-to-Analog Converter) 12-bit.
- Các interface giao tiếp như USB (Universal Serial Bus), UART, SPI, I2C.



Hình 3.8. Vi điều khiển STM32F407

Vi điều khiển STM32F407 có khả năng xử lý cao, được thiết kế để hỗ trợ các ứng dụng yêu cầu xử lý số liệu phức tạp và tốc độ cao. Nó cũng có khả năng xử lý tín hiệu nhanh và hiệu quả, làm cho nó trở thành một lựa chọn phù hợp cho các ứng dụng như điều khiển động cơ, điều khiển các thiết bị điện tử và điều khiển các hệ thống thông minh.

Để lập trình cho vi điều khiển STM32F407, có thể sử dụng các công cụ như STM32CubeIDE, một môi trường phát triển tích hợp (IDE) miễn phí của STMicroelectronics, hoặc các công cụ phát triển bên thứ ba như Keil MDK hoặc IAR Embedded Workbench. Các ngôn ngữ lập trình được hỗ trợ bao gồm C và C++.

3.2.2. Raspberry Pi 4B Model 8GB RAM

Raspberry Pi 4 Model B là một phiên bản mới nhất của dòng sản phẩm Raspberry Pi, được giới thiệu vào tháng 6 năm 2019 bởi Raspberry Pi Foundation. Raspberry Pi 4 Model B là một board máy tính nhúng có hiệu năng cao, với nhiều tính năng mới hơn so với các phiên bản trước đó.



Hình 3.9. Máy tính nhúng Raspberry Pi 4.

Raspberry Pi 4 Model B được trang bị bộ vi xử lý Broadcom BCM2711 64-bit quad-core ARM Cortex-A72 tốc độ 1.5GHz, là bộ vi xử lý mạnh nhất từng được sử dụng trên một board Raspberry Pi. Phiên bản này cũng được cung cấp với 4GB hoặc 8GB RAM, tăng khả năng xử lý dữ liệu và cải thiện hiệu suất cho các ứng dụng đòi hỏi bộ nhớ nhiều.

Raspberry Pi 4 Model B:

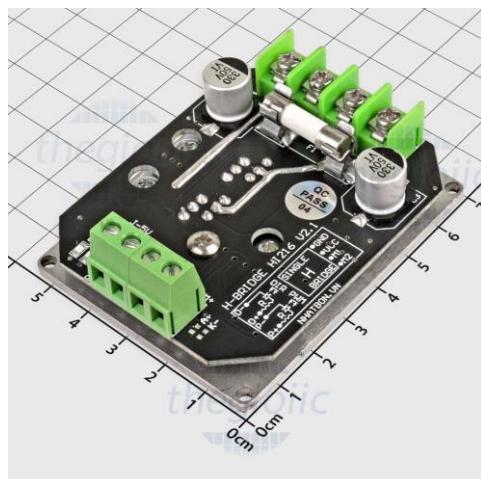
- Hỗ trợ kết nối Gigabit Ethernet, Bluetooth 5.0, Wi-Fi chuẩn 802.11ac và các cổng USB 3.0 và USB 2.0. Nó cũng có các cổng kết nối video HDMI 2.0, cho phép hiển thị 4K trên màn hình và âm thanh 7.1 kênh.
- Trang bị hai cổng Micro-HDMI để hiển thị hình ảnh 4K ở tốc độ 30 khung hình/giây hoặc 1080p ở tốc độ 60 khung hình/giây. Nó cũng có cổng âm thanh 3.5 mm và khe cắm thẻ nhớ MicroSD cho khả năng mở rộng bộ nhớ lưu trữ.
- Hỗ trợ nhiều hệ điều hành như Raspbian, Ubuntu, Windows 10 IoT Core và các hệ điều hành Linux khác. Nó cũng hỗ trợ nhiều ngôn ngữ lập trình phổ biến như Python, C++, Scratch và Java.
- Có kích thước nhỏ gọn, tiêu thụ điện năng thấp và giá thành phải chăng, là một giải pháp phù hợp cho các ứng dụng IoT, máy tính

nhúng, điều khiển thiết bị và các ứng dụng khác yêu cầu xử lý dữ liệu và kết nối mạng cao.

- Trang bị 40 chân GPIO, cung cấp cho người dùng khả năng tương tác với các thiết bị ngoại vi như cảm biến, motor, LED, đầu vào analog và digital. Chân GPIO được hỗ trợ bởi các thư viện phần mềm và các công cụ lập trình cho phép người dùng tương tác với các thiết bị này.

3.2.3. Driver điều khiển động cơ

Mạch H HI216 dùng IC kích Fet chuyên dụng, cho Fet luôn dẫn tốt nhất, tránh hiện tượng trùng dẫn, giúp công suất và hiệu năng của board luôn đạt ngưỡng dẫn tốt nhất và ít hư hỏng Fet, Module này chuyên dùng cho điều khiển tốc độ động cơ công suất lớn, điều khiển vị trí, vận tốc DC Motor, có thể lựa chọn mức kích âm hoặc dương để dễ dàng điều khiển.



Hình 3.10. Driver điều khiển động cơ.

Thông số kỹ thuật	Mô tả
Điện áp kích	3.3V đến 5.5V
Điện áp công suất	12V đến 48V
Dòng liên tục	15A
Dòng đỉnh	20A
Công suất tối đa	600W

Tần số hoạt động tối đa	100 KHz
Động rộng xung tối đa	100%
Kích thước	54x64x22 mm

3.2.4. USB Camera Rapoo

Rapoo C260 là một trong những thiết bị webcam sở hữu độ phân giải 1080P cho hình ảnh rõ ràng và sắc nét, điều chỉnh tốc độ khung hình một cách phù hợp với điều kiện ánh sáng xung quanh.

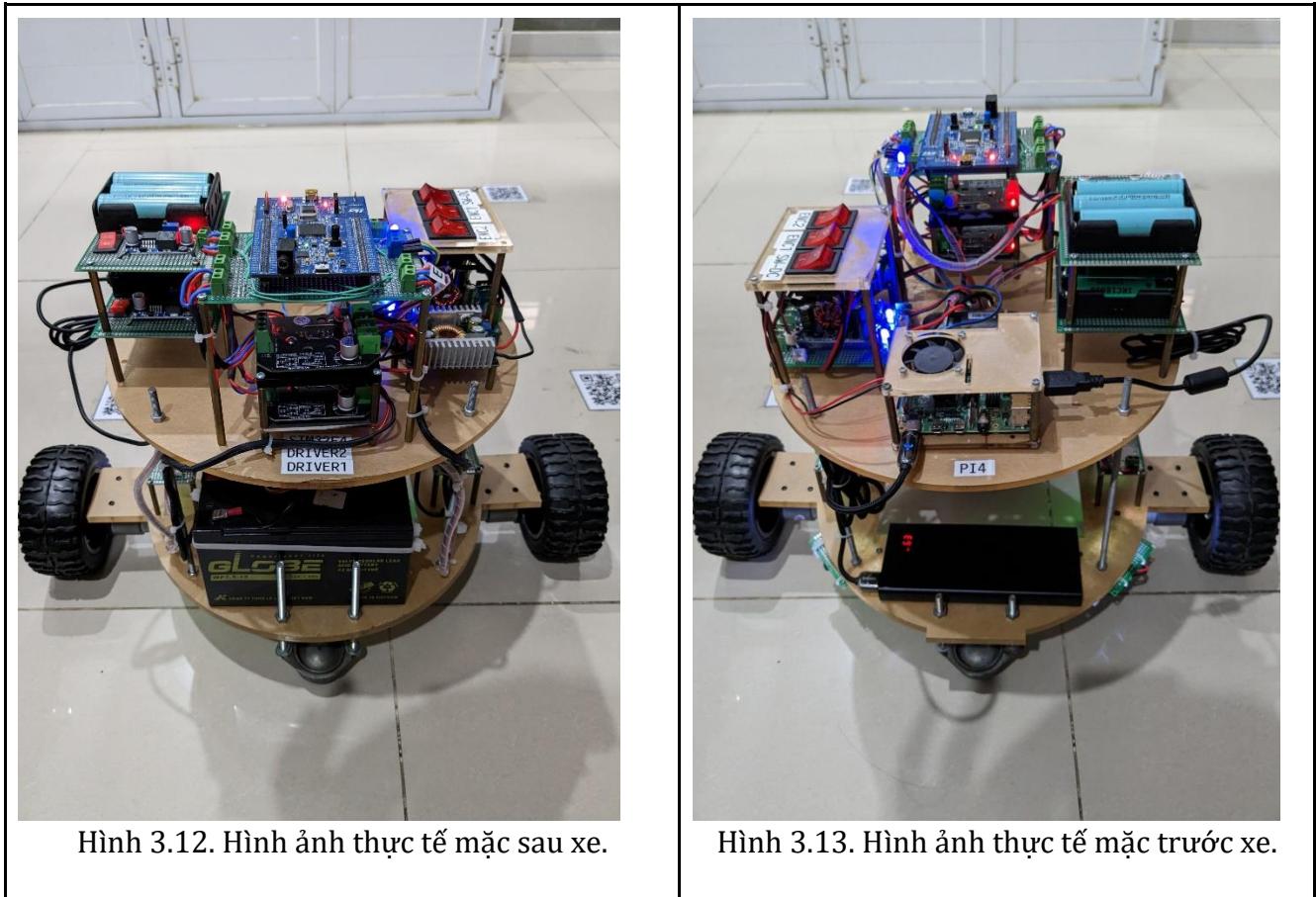


Hình 3.11. USB Camera Rapoo.

Thông số kỹ thuật	Mô tả
Hãng sản xuất	Rapoo
Model	C260 FullHD 1080p
Góc nhìn	80 độ
Video Encoding	H.264
Độ phân giải	FHD 1080P / HD 720P
Hiệu chỉnh ánh sáng tự động	Có
Kết nối	USB 2.0
Kích thước	77x48.7x45mm

3.3. Thực tế phần cứng sau khi xây dựng hoàn thiện

- Mô hình robot:



Bảng 3.1. Bảng thông số kỹ thuật của Robot AGV

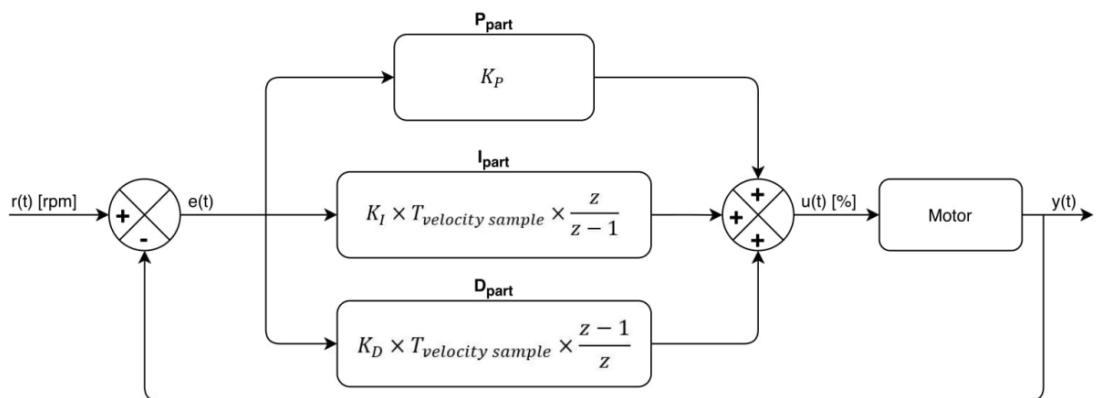
Bảng thông số kỹ thuật	
Trọng lượng	3.5kg
Kích thước bán kính	30cm
Chiều cao	35cm
Điện áp hoạt động	12V
Tốc độ chạy tối đa	0.3m/s
Tốc độ chạy ổn định	0.15m/s
Mang tải tối đa	2kg

Chương 4. THIẾT KẾ PHẦN MỀM

4.1. Giải thuật điều khiển

4.1.1. Khối PID Motor

Từ vận tốc đặt cho từng bánh, chúng ta cần điều khiển vận tốc động cơ sử dụng bộ điều khiển PID rời rạc, với thời gian lấy mẫu $T_{velocity_sample} = 5$ ms.



Hình 4.1. Khối PID.

Cách điều chỉnh hệ số K_P, K_I, K_D

- Cho K_I và K_D bằng 0, tăng dần K_P cho tới khi hệ thống đạt giá trị xác lập và có thể có vọt lố.
- Tăng dần K_I với K_P ở bước trên. Tăng đến khi đạt tới điểm xác lập, có thể có vọt lố. Chọn K_D sao cho mất đi độ vọt lố.
- Tùy chỉnh K_P để có được thời gian đáp ứng mong muốn.

Bước 1: Xử lý tín hiệu Encoder, tính toán vận tốc từng bánh.

Vận tốc hiện tại của động cơ được tính dựa trên số xung trên vòng của encoder trong một chu kỳ lấy mẫu vận tốc, được tính theo công thức:

$$v(k) = \frac{(Encoder(k) - Encoder(k-1))}{Encoder_Per_Resolution} \frac{60}{T_{velocity_sample}} [rpm]$$

Trong đó:

- Encoder(k): Giá trị xung hiện tại được đọc từ Encoder của động cơ.
- Encoder(k-1): Giá trị Encoder trước đó 1 chu kỳ.
- Encoder Per Resolution: Tổng số xung encoder trên 1 vòng quay động cơ.
- $T_{velocity_sample}$: Thời gian lấy mẫu vận tốc.

Mô hình xe sử dụng đơn vị m/s cho việc cấu hình vận tốc, vì thế cần chuyển đổi đơn vị vận tốc m/s sang rpm để động bộ với bộ điều khiển PID bằng công thức:

$$v_{rpm} = \frac{v_{m/s} * 60}{2\pi R}$$

Trong đó: R là bán kính của bánh xe.

Bước 2: Cập nhật luật điều khiển PID rời rạc, công thức luật điều khiển

$$P_{Part} = K_P(e(k) - e(k-1))$$

$$I_{Part} = \frac{K_I T_{velocity_sample}}{2} (e(k) + e(k-1))$$

$$D_{Part} = \frac{K_D}{T_{velocity_sample}} (e(k) - 2e(k-1) + e(k-2))$$

$$u(k) = u(k-1) + P_{Part} + I_{Part} + D_{Part}$$

Ngõ ra $u(k)$ được cho qua khâu bão hòa (-100, 100) để tránh trường hợp vọt lố và vượt ngoài tầm giá trị của PWM.

Bước 3: Xuất xung điều khiển PWM

Vận tốc động cơ điều chỉnh dựa vào giá trị độ rộng xung PWM trong 1 chu kỳ với tần số phù hợp với động cơ đang sử dụng (thường là 20 kHz)

$$PWM = \frac{u(k)*N}{100}$$

Với N là tổng số xung trong 1 chu kỳ xuất xung đầy PWM.

4.1.2. Khối vận tốc hình thang

- **Bước 1:** Khởi tạo giá trị v_{max} , a_{max}

- **Bước 2:** Xem xét giá trị thời gian đang thuộc khoảng nào trong đồ thị vận tốc hình thang để tính được giá trị quãng đường hiện tại. Đặt khoảng thời gian tăng tốc, giảm tốc và xe có vận tốc cực đại là t_1 , t_3 và t_2 .

$$\text{Nếu } 0 \leq t \leq t_1: s = \frac{1}{2}at^2$$

$$\text{Nếu } t_1 \leq t \leq t_2: s = \frac{1}{2}at_1^2 + v1_{max}$$

$$\text{Nếu } t_2 \leq t \leq t_3: s = \frac{1}{2}at_1^2 + v2\frac{\frac{1}{2}}{v_{12}}_{max}^2$$

Từ giá trị quãng đường này, chuyển đổi sang giá trị xung.

- **Bước 3:** Đọc tín hiệu xung từ Encoder 1 và Encoder 2.
- **Bước 4:** Tính toán sai số giữa hai giá trị xung có được từ bước 2 và bước 3. Ta được hai giá trị sai số tương ứng của động cơ 1 và động cơ 2.
- **Bước 5:** Cho hai giá trị sai số này đi qua bộ điều khiển PID với thời gian lấy mẫu $T = 5$ ms để tính toán chu kỳ xung cho động cơ 1 và động cơ 2.
- **Bước 6:** Gửi giá trị chu kỳ xung tương ứng xuống hai động cơ.

4.2. Hệ thống phần mềm điều khiển trên Máy tính nhúng

4.2.1. Tổng quan giải thuật

Giải thuật triển khai trên máy tính nhúng Raspberry Pi 4 có 4 tasks, mỗi task là một thread, tất cả được lập trình multi-thread, tóm tắt như sau:

- Xử lý ảnh: sử dụng thuật toán Optical Flow, camera đi tìm QR code trong khung hình với độ chính xác cao, sau đó giải mã data chứa trong QR code. Tính toán góc lệch giữa tâm QR code và hướng đi của xe, khoảng cách giữa tâm khung hình tới tâm QR code để ứng dụng vào quá trình calib giữa xe AGV với QR code, mục tiêu là đưa QR code vào chính giữa khung hình.
- Giao tiếp giữa máy tính nhúng Raspberry Pi 4 và vi điều khiển STM32F407 thông qua giao thức UART: chiều dữ liệu từ máy tính nhúng xuống vi điều khiển là frame data của hành động calib, cách thức di chuyển từ QR code này tới QR code khác. Còn dữ liệu vi điều khiển gửi lên máy tính nhúng là tín

hiệu thông báo đã thực hiện xong hành động mà Pi 4 đã gửi xuống trước đó.

- Giao tiếp với hệ thống điều khiển trên máy tính thông qua giao thức MQTT: được lập trình bằng 2 threads. Thread đầu, máy tính nhúng là publisher (gửi dữ liệu từ máy tính nhúng lên hệ thống). Thread còn lại, máy tính nhúng là subscriber (nhận dữ liệu từ hệ thống).

4.2.2. Lập trình đa luồng

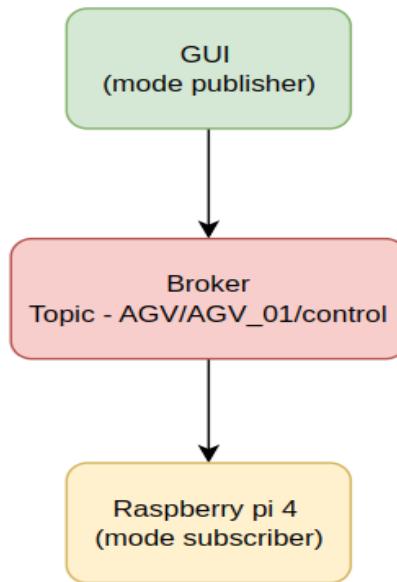
Python hỗ trợ lập trình đa luồng (Multithreading) thông qua module threading. Lập trình đa luồng cho phép chúng ta tạo nhiều luồng thực thi song song để tăng tốc độ xử lý của chương trình. Mỗi luồng sẽ có thể thực hiện một tác vụ độc lập và chúng ta có thể đồng bộ hóa các luồng bằng cách sử dụng các phương thức liên quan đến luồng.

Để có thể giao tiếp dữ liệu giữa các thread, chúng tôi sử dụng một cấu trúc dữ liệu có tên là Queue. Queue là nơi các phần tử được thêm vào và xóa khỏi theo cơ chế đầu vào đầu ra (FIFO - First In First Out). Nghĩa là phần tử đầu tiên thêm vào hàng đợi sẽ được xử lý trước, và các phần tử thêm vào sau sẽ phải đợi cho tới khi phần tử đầu tiên được xử lý xong và được lấy ra khỏi hàng đợi.

4.2.3. Giao tiếp MQTT

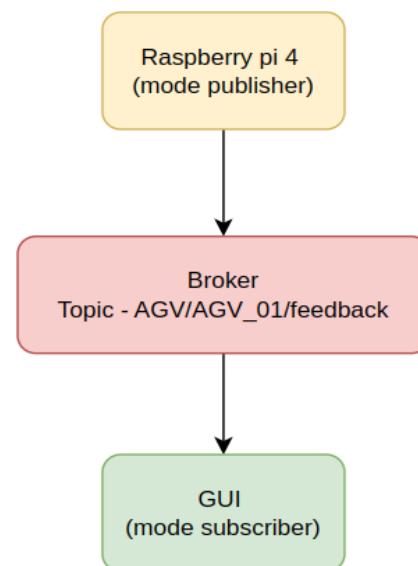
Ở nhiệm vụ này, ta có 2 threads hoạt động

- Thread 1: máy tính nhúng Raspberry Pi 4 có mode subscriber. Khi nhận được dữ liệu gửi từ hệ thống điều khiển trên máy tính thông qua topic *AGV/AGV_01/control*. Nó sẽ tiếp nhận, phân tích dữ liệu sau đó tính toán lộ trình di chuyển giữa hai mã QR bất kỳ trên ma trận.



Hình 4.2. Truyền message control thông qua MQTT.

- Thread 2: máy tính nhúng Raspberry Pi 4 có mode là publisher. Nó có nhiệm vụ gửi thông báo lên hệ thống điều khiển khi xe đi tới QR code, mục tiêu của việc này nhằm giúp hệ thống có thể theo dõi tiến trình di chuyển của xe AGV, giúp nắm bắt được QR code đang ở đâu theo thời gian thực.



Hình 4.3. Truyền message feedback thông qua MQTT.

4.2.4. Xử lý ảnh

Khi có được lộ trình đường đi từ thread *PubMessage()*, xe sẽ có thông tin các QR code nào sẽ phải di chuyển qua và di chuyển như thế nào, kết hợp các hành động đi thẳng, xoay trái góc 90° , xoay phải góc 90° . Ngoài ra, có một hành động quan trọng trước khi thực hiện di chuyển từ QR code này tới QR code kế tiếp đó là *calib* giữa xe và QR code.

Trước khi đi vào các thuật toán xử lý, tính toán calib, ta cần nắm các ý sau:

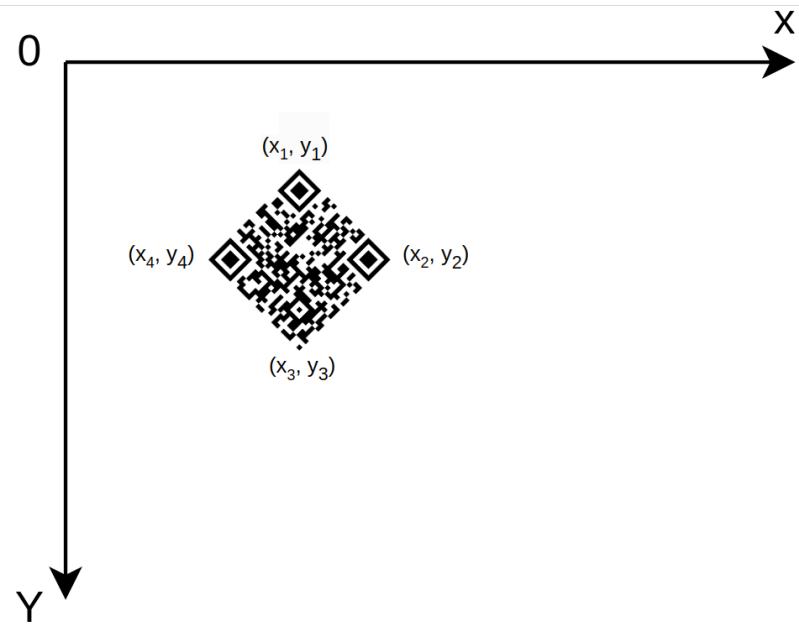
- Kích thước của khung hình: height = 320 pixels, width = 500 pixels.
- Tâm của khung hình có tọa độ: x = 315, y = 245.
- Khoảng cách từ Camera gắn trên xe tới QR code đặt dưới sàn: 23 cm.
- Đo đạc và tính toán ra được trong trường hợp này, trong frame của camera:

$$1 \text{ pixel} = 0.5 \text{ mm}$$

- Hệ trục tọa độ Oxy được gắn với frame của camera, khi robot, nó cũng sẽ di chuyển theo.
- Tọa độ của QR code trong khung hình được xác định và quy ước như sau, tại góc không có hoa văn định vị luôn là point 3, đối diện point 3 sẽ là point 1. Từ đó dễ dàng nhận biết point 2 và point 4 như hình dưới.

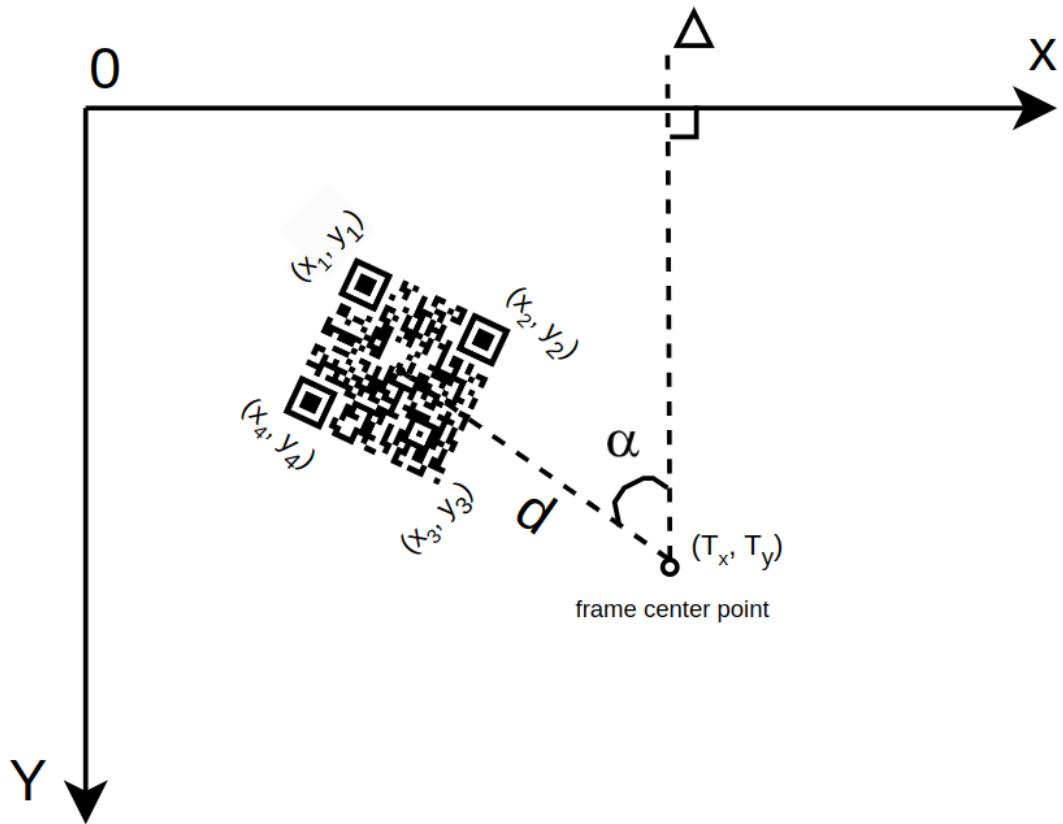
Các bước thực hiện xử lý QR code để xác định các dữ liệu cần thiết gửi xuống vi điều khiển:

Bước 1: Phát hiện và giải mã QR code có áp dụng thuật toán Optical Flow để tăng độ chính xác. Sau khi giải mã, ta có tọa độ của 4 góc QR code và thông tin mà QR code đang chứa như hình dưới.



Hình 4.4. Phát hiện mã QR và xác định các thông số.

Bước 2: Thực hiện quá trình căn chỉnh đầu tiên. Tính toán góc, khoảng cách giữa tâm khung hình và QR code, ý tưởng ở đây là xoay xe tại chỗ sao cho góc α giảm dần về 0, sau đó cho xe di chuyển một đoạn d để tiến tới QR code. Với quy ước tâm QR code nằm bên trái đường thẳng Δ thì góc lệch dương, ngược lại nếu nằm phía bên phải là góc lệch âm, nằm trên đường Δ thì góc lệch bằng 0.



Hình 4.5. Thực hiện quá trình căn chỉnh đầu tiên.

- Từ tọa độ 4 điểm đã tìm được từ bước 1, ta xác định tọa độ trọng tâm QR code

$$G_x = \frac{x_1 + x_3}{2}$$

$$G_y = \frac{y_1 + y_3}{2}$$

- Tính toán góc lệch từ QR code và tâm khung hình với hướng di chuyển:

$$\tan(\alpha) = \frac{T_x - G_x}{T_y - G_y}$$

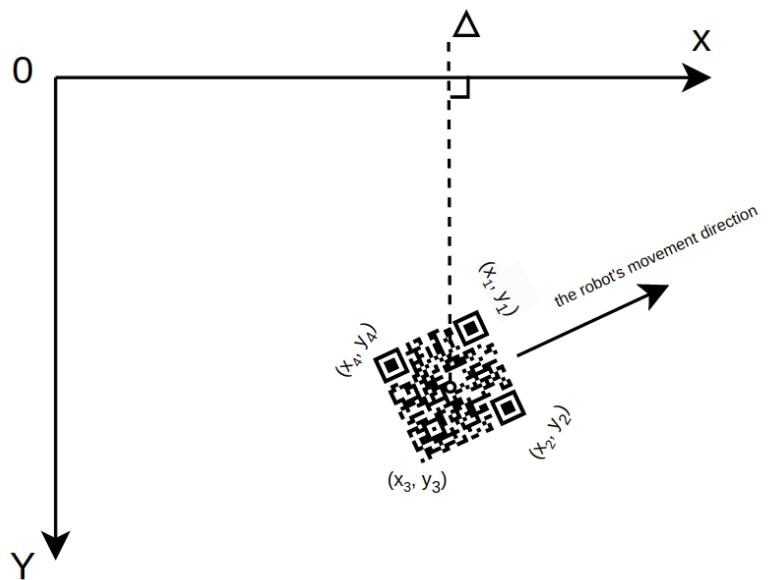
$$\Rightarrow \alpha = \arctan\left(\frac{T_x - G_x}{T_y - G_y}\right)$$

- Tính toán khoảng cách từ tâm khung hình với QR code:

$$d = \sqrt{(T_x - G_x)^2 + (T_y - G_y)^2}$$

Bước 3: Sau khi đã đưa được tâm khung hình tới QR code, nhiệm vụ cǎn

chỉnh tiếp theo là làm cho QR code ngay với khung hình (đưa Δ tiến tới trùng với *the robot's movement direction*).



Hình 4.6. Thực hiện quá trình căn chỉnh ngay giữa khung hình.

- Với quy ước *the robot's movement direction* thuộc vùng bên phải của Δ thì góc lệch có giá trị âm, ngược lại góc lệch có giá trị dương. Vậy ta có biểu thức góc lệch giữa QR code và camera (β - góc giữa Δ và *the robot's movement direction*):

$$\tan(\beta) = \frac{y_1 - y_2}{x_2 - x_1}$$

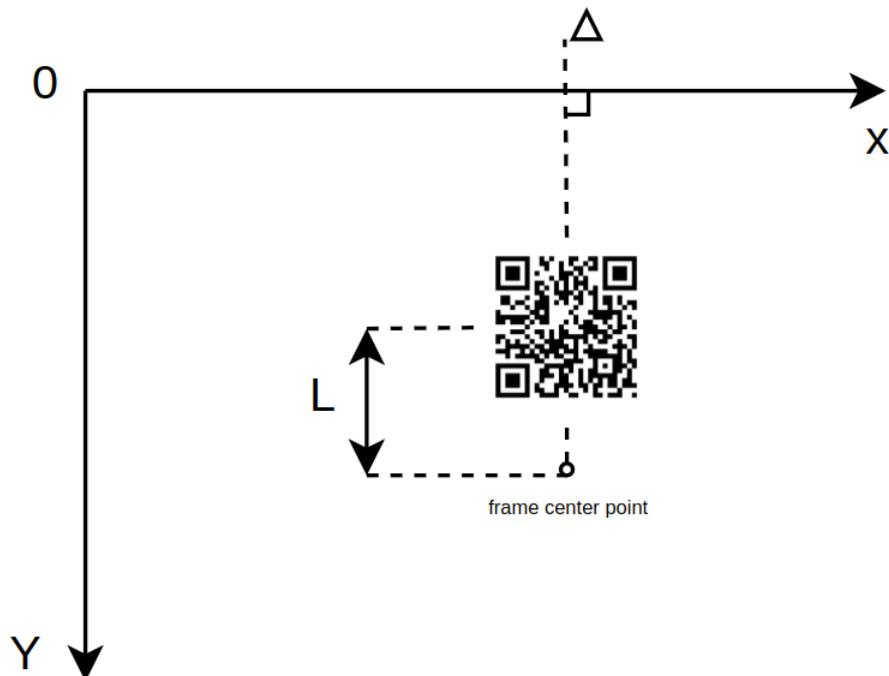
$$\beta = \arctan\left(\frac{y_1 - y_2}{x_2 - x_1}\right)$$

- Ta có các giá trị góc biên của QR code với camera như sau:

Bảng 4.1. Bảng giá trị góc biên của các mã QR Code

0°	90°	180°	-90°	-180°

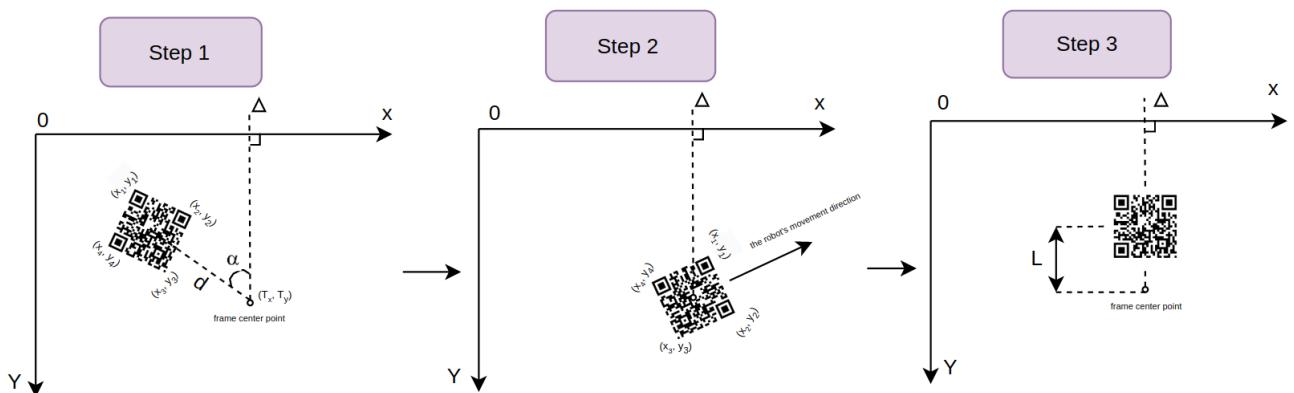
Bước 4: Sau khi thực hiện ba bước trên, nếu QR code vẫn chưa ngay hoàn toàn với khung hình, thì ta thực hiện thêm bước tịnh tiến tới/lùi.



Hình 4.7. Thực hiện quá trình căn chỉnh cuối cùng.

$$L = |G_y - T_y|$$

Tổng kết lại các bước, xe thực hiện các hành động sau:

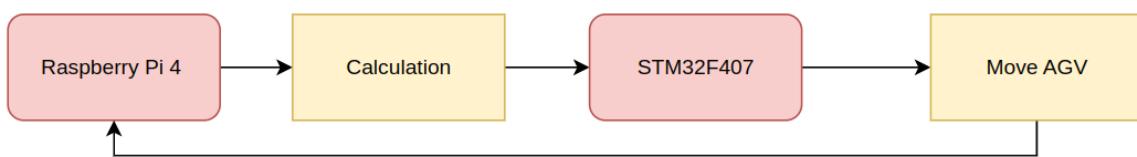


Hình 4.8. Tổng hợp các bước căn chỉnh.

Vậy sau khi thực hiện xong các bước căn chỉnh giữa khung hình và QR code, theo giải thuật đã được lập trình, xe sẽ được phép thực hiện các hành động như đi thẳng, rẽ trái 90° , rẽ phải 90° để di chuyển tới QR code kế tiếp.

4.2.5. Giao tiếp với vi điều khiển thông qua giao thức UART

Sau mỗi lần calib giữa QR code và xe, ta có giá trị của các tham số như góc, khoảng cách. Xe di chuyển khi máy tính nhúng Raspberry Pi 4 gửi xuống vi điều khiển các giá trị của tham số đó. Sau khi xe thực hiện xong một hành động máy tính nhúng gửi xuống vi điều khiển để yêu cầu, vi điều khiển sẽ gửi tín hiệu thông báo lên máy tính nhúng là nó đã thực hiện xong. Để thực hiện hành động tiếp theo, Pi 4 sẽ tiếp tục tính toán rồi gửi dữ liệu phù hợp xuống vi điều khiển. Lặp lại quá trình truyền và nhận dữ liệu của cả hai cho đến khi xe đi được đến đích.



Hình 4.9. Mô hình giao tiếp giữa STM32 và Máy tính nhúng.

4.3. Hệ thống giao diện điều khiển trên máy tính

4.3.1. Tổng quan thuật toán

Giao diện điều khiển người dùng được lập trình multithread với 4 task, trong đó task init và khởi tạo đầu tiên là Main Application. Sau khi giao diện đã được khởi tạo, chương trình cũng sẽ khởi tạo Capture Task, các task còn lại sẽ được khởi tạo sau khi có các event cần thiết trên giao diện người dùng.

Các Open Source chính được dùng trong giao diện:

- Mosquitto: là một thư viện mã nguồn mở cho phép lập trình viên có thể lập trình các MQTT Client như Publisher, Subscriber và cả Broker bằng ngôn ngữ C/C++

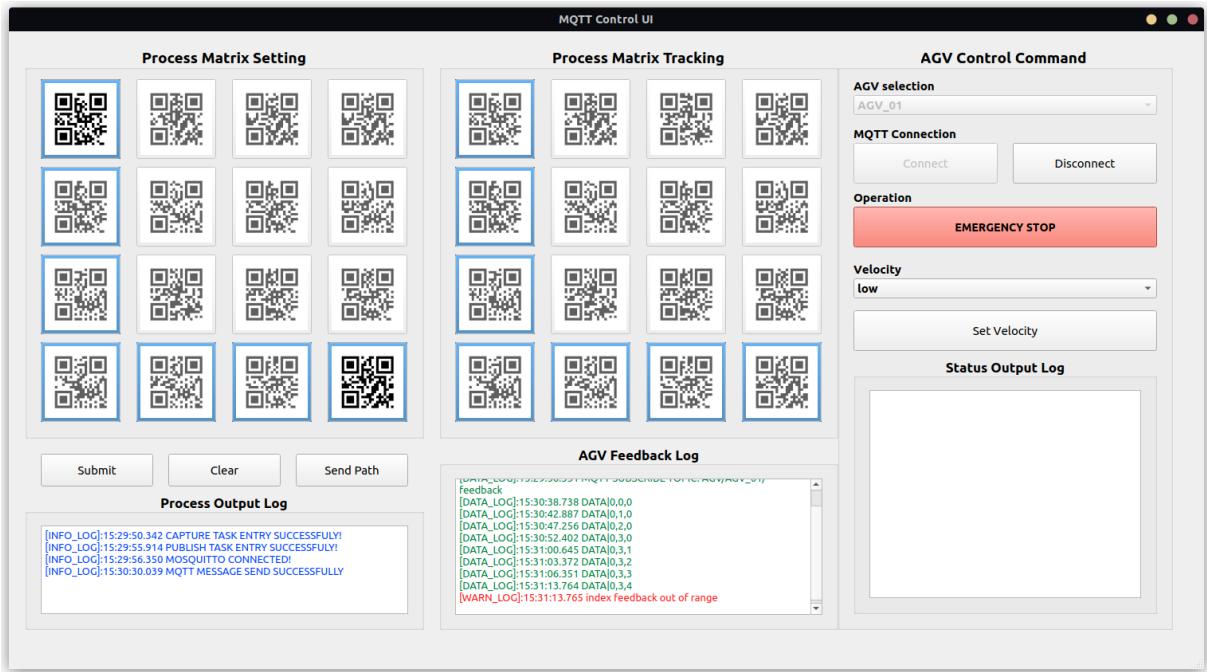


Hình 4.10. Logo của open source Mosquitto.

- Ak Linux Base: Là một Open Source đa luồng, quản lý các tác vụ theo hướng “Event Driven”, các tác vụ sẽ trao đổi với nhau thông qua “signal”. Thư viện này được viết và chia sẻ từ FPT Telecom.
- Jansson: Là một thư viện dùng để đóng gói và làm việc với data type dạng JSON.

4.3.2. Giao diện tổng quát

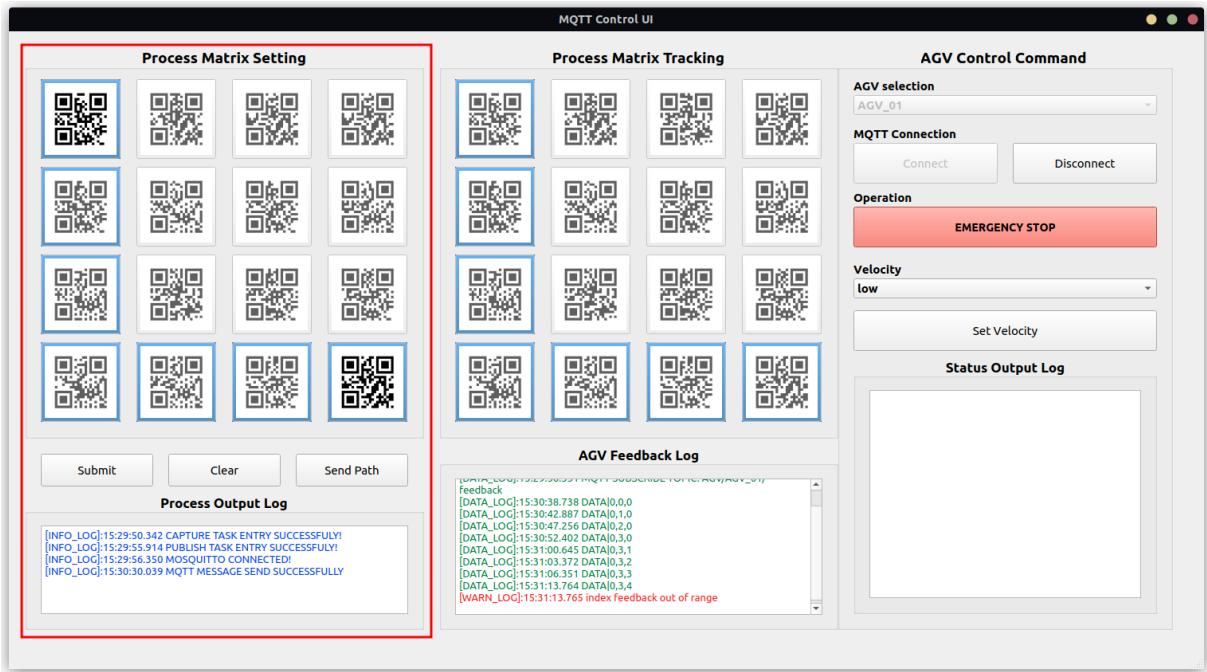
Giao diện tổng quát bao gồm 3 vùng làm việc chính, bao gồm: Vùng làm việc ra lệnh, chọn đường đi cho xe AGV, vùng làm việc nhận phản hồi và hiển thị từ xe AGV, vùng cấu hình một số command tự định nghĩ như reset máy tính nhúng, reset STM32F407, cấu hình tốc độ động cơ....



Hình 4.11. Giao diện tổng quan của GUI.

4.2.2.1. Vùng “Process Matrix Setting”

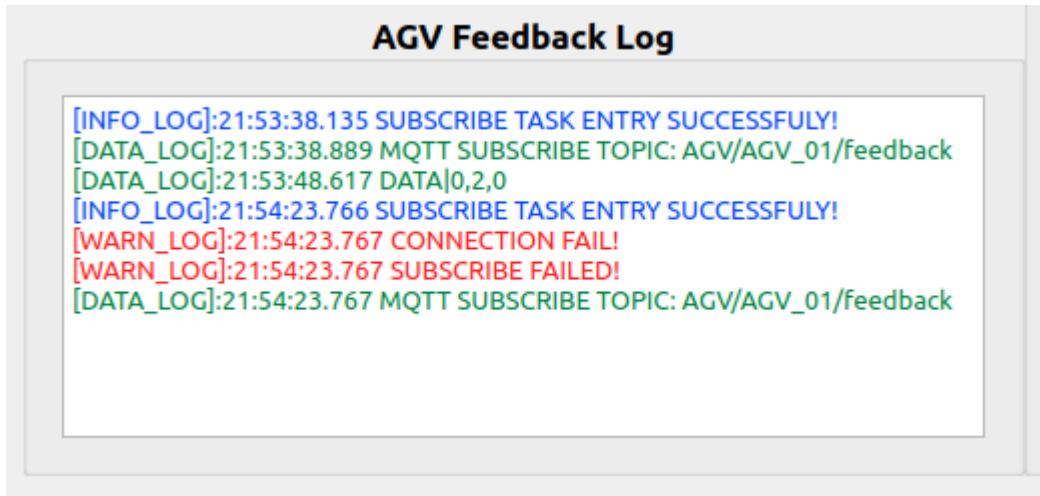
Vùng “Process Matrix Setting” là vùng để người dùng cấu hình đường đi của xe AGV dựa vào điểm bắt đầu và điểm kết thúc. Sau khi chọn điểm bắt đầu và điểm kết thúc trong 16 ô hiển thị trên GUI, người dùng nhấn “Submit” button để GUI tính toán đường đi bằng thuật toán lập trình trước. Sau khi có được đường đi hiển thị trên GUI, người dùng có thể ấn “Send Path” nếu đã xác nhận, hoặc bấm “Clear” nếu muốn chọn lại.



Hình 4.12. Khu vực set đường đi và gửi thông tin.

Sau khi ấn Send Path, giao diện người dùng sẽ gửi một command có chứa thông tin theo frame tự định nghĩa lên MQTT Broker (sẽ được mô tả chi tiết trong phần Giao tiếp MQTT). Ngoài các nút nhấn, vùng làm việc này còn có 1 textbox để hiển thị thông tin log có chứa ngày tháng. Thông tin log có dạng “[<loại thông báo>]:<timestampl> <nội dung>”. Có 3 loại log được hiển thị trên GUI:

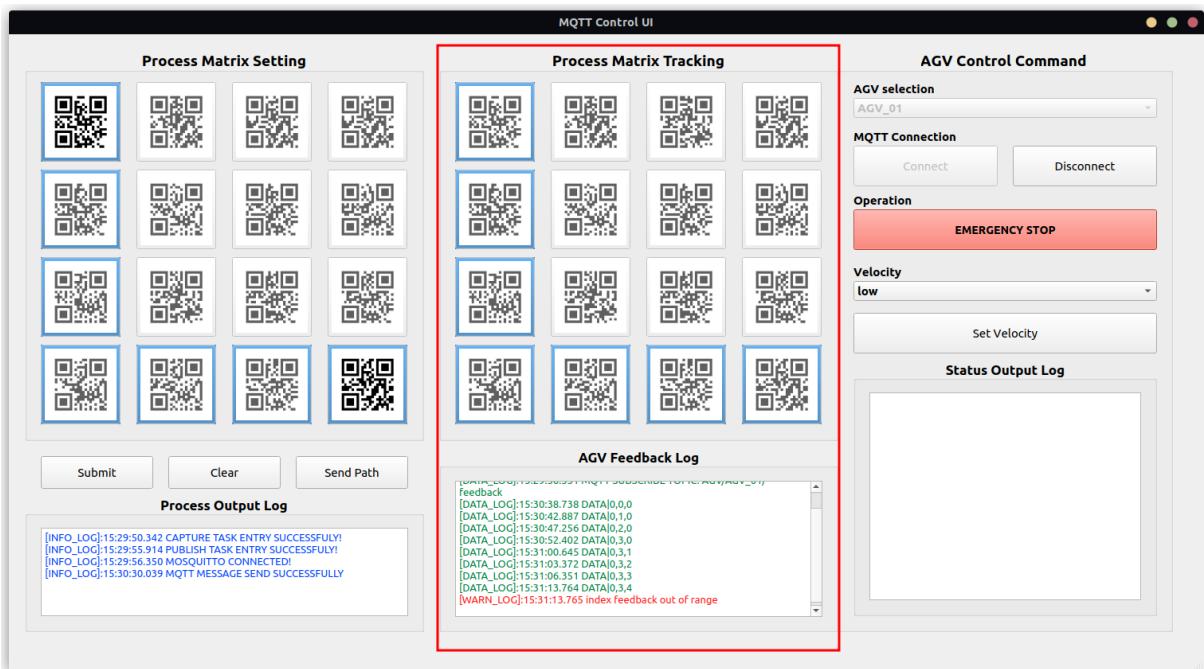
- Log ghi nhận thông tin [INFO_LOG]: Thông báo màu xanh dương ghi nhận các trạng thái của GUI như các task đã khởi chạy thành công, đã kết nối được MQTT Broker...
- Log ghi nhận lỗi [WARN_LOG]: Thông báo màu đỏ ghi nhận các lỗi đã diễn ra như mất kết nối với internet, subscribe topic không thành công...
- Log ghi nhận dữ liệu [DATA_LOG]: Bao gồm các thông báo xanh lá ghi nhận các dữ liệu trả về từ MQTT, thông tin topic, các message nhận được từ AGV...



Hình 4.13. Các loại log được thể hiện bằng màu sắc khác nhau.

4.2.2.2. Vùng nhận phản hồi “Process Matrix Tracking”

Vùng phản hồi là vùng làm việc không được phép chọn hoặc chỉnh sửa từ người dùng. Đây là vùng hiển thị các phản hồi của xe AGV gửi về thông qua MQTT. Cứ mỗi điểm checkpoint QR Code, xe sẽ gửi phản hồi để người dùng có thể nhận biết vị trí của xe trong thời điểm gần nhất.

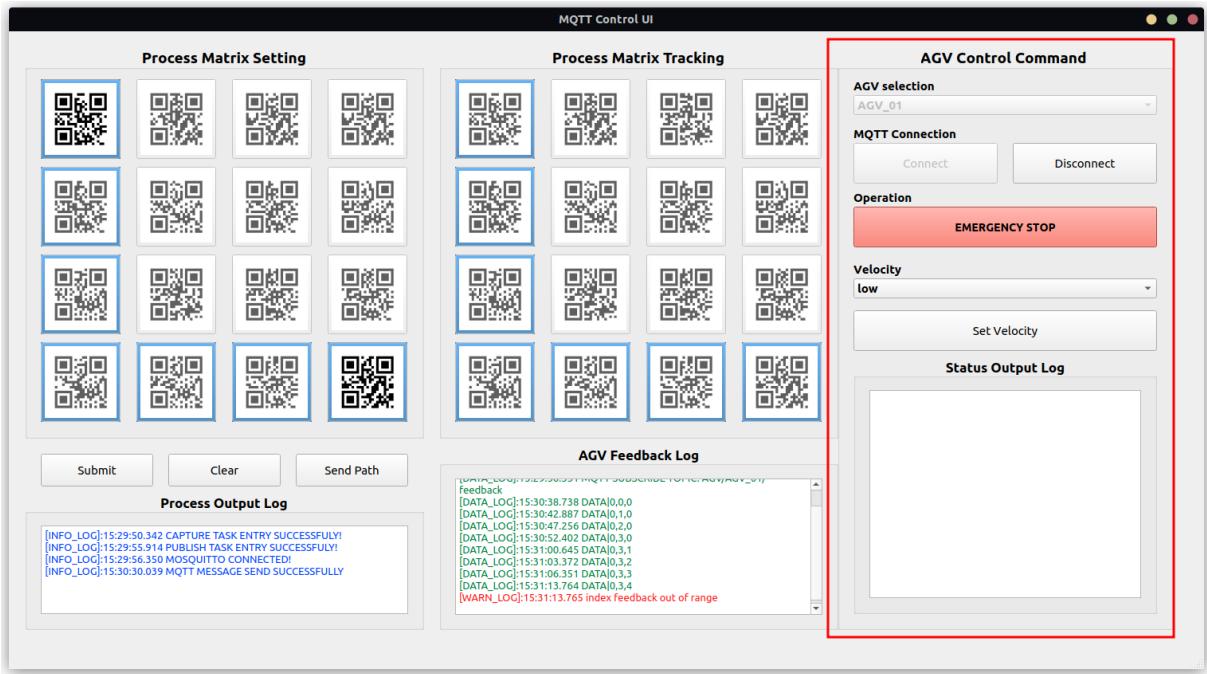


Hình 4.14. Khu vực theo dõi quá trình di chuyển của xe.

4.2.2.3. Vùng điều khiển Setup và các nút nhấn gửi câu lệnh tự định nghĩa “AGV Control Command”

Vùng làm này cho phép người dùng thay đổi topic để có thể điều khiển các

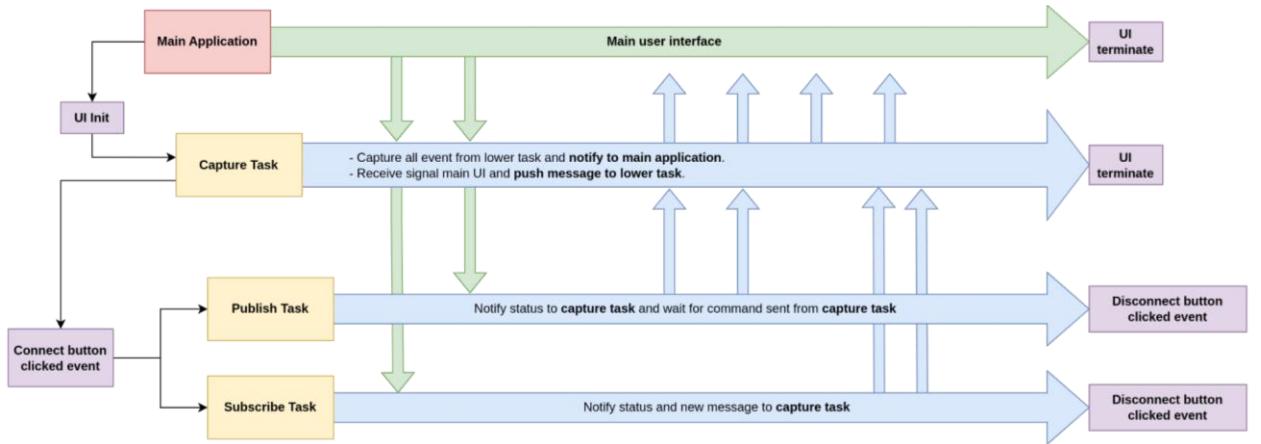
AGV khác. Các topic được thay đổi theo ID của xe AGV trong mục "AGV selection". Khi bắt đầu điều khiển và kết nối với MQTT, người dùng chọn button "Connect", sau khi điều khiển hoặc dừng điều khiển, người dùng có thể chọn "Disconnect". Ngoài ra vùng này cũng có các câu lệnh tự định nghĩa như reset máy tính nhúng Raspberry Pi, reset STM32F407, lựa chọn tốc độ của xe theo 3 mức: low, normal, high.



Hình 4.15. Khu vực setup ban đầu GUI.

4.3.3. Đa luồng

Để có thể thực hiện cùng lúc 3 tác vụ: cập nhật và hiển thị lên UI, publish tin nhắn và subscribe topic, giao diện được viết với 4 thread chính. Nhóm sinh viên gọi 4 thread này là các "task". Các task giao tiếp với nhau qua các "signal" và được lập trình, xây dựng theo hướng "event driven".



Hình 4.16. Mô hình đa luồng được xây dựng cho giao diện điều khiển.

Đầu tiên, task hiển thị UI (Main Application) được khởi tạo, task này cũng sẽ khởi tạo Capture task. Các task publish/subscribe bắt đầu khởi tạo sau khi người dùng ấn nút “Connect”. Main Application và Capture sẽ dừng khi người dùng tắt GUI, còn hai task publish/subscribe sẽ dừng sau khi người dùng ấn “Disconnect”. Nhiệm vụ các task được mô tả như sau:

- Main Application: Là task chính của framework Qt C++, task này có nhiệm vụ quản lý, cập nhật trạng thái của giao diện người dùng.
- Capture task: là tác vụ dùng để “hứng” tất cả các tín hiệu, dữ liệu của hai task MQTT và chính nó, sau khi hứng các tín hiệu này, capture task sẽ truyền lên main application để cập nhật UI. Tác vụ này chính là cầu nối UI và hai tác vụ giao tiếp MQTT.

```

1 void* task_app_capture_entry(void){
2     textBrowser.emitTaskTextBrowser(Logger(INFO_PUB, QString::fromStdString(CAP_ENTRY)));
3     while(1) {
4         ak_msg_t *msg;
5         msg = ak_msg_rev(TASK_APP_3);
6         QString insertText;
7         switch(msg->header->sig){
8             case INFO_PUB:
9                 insertText = QString::fromStdString((char*)(msg->header->payload));
10                textBrowser.emitTaskTextBrowser(Logger(INFO_PUB, insertText));
11                break;
12            case WARNING_PUB:
13                insertText = QString::fromStdString((char*)(msg->header->payload));
14                textBrowser.emitTaskTextBrowser(Logger(WARNING_PUB, insertText));
15                break;
16            case DATA_PUB:
17                insertText = QString::fromStdString((char*)(msg->header->payload));
18                textBrowser.emitTaskTextBrowser(Logger(DATA_PUB, insertText));
19                break;
20            case INFO_SUB:
21                insertText = QString::fromStdString((char*)(msg->header->payload));
22                textBrowser.emitFeedbackTextBrowser(Logger(INFO_SUB, insertText));
23                break;
24            case WARNING_SUB:
25                insertText = QString::fromStdString((char*)(msg->header->payload));
26                textBrowser.emitFeedbackTextBrowser(Logger(WARNING_SUB, insertText));
27                break;
28            case DATA_SUB:
29                insertText = QString::fromStdString((char*)(msg->header->payload));
30                textBrowser.emitFeedbackTextBrowser(Logger(DATA_SUB, insertText));
31                break;
32            case MATRIX_SUB:
33                char row[2], column[2], direct[2];
34                sscanf((char*)msg->header->payload, "%[^,]*[,%[^,]*[,%[^,]*[,%[^,]*]", direct, row, column);
35                matrixTracking.emitMatrixUpdate(atoi(direct), atoi(row), atoi(column));
36                break;
37            default:
38                break;
39        }
40        /* free message */
41        ak_msg_free(msg);
42    }
43    return (void *)0;
44 }
45 }

```

Hình 4.17. Một đoạn code của task capture.

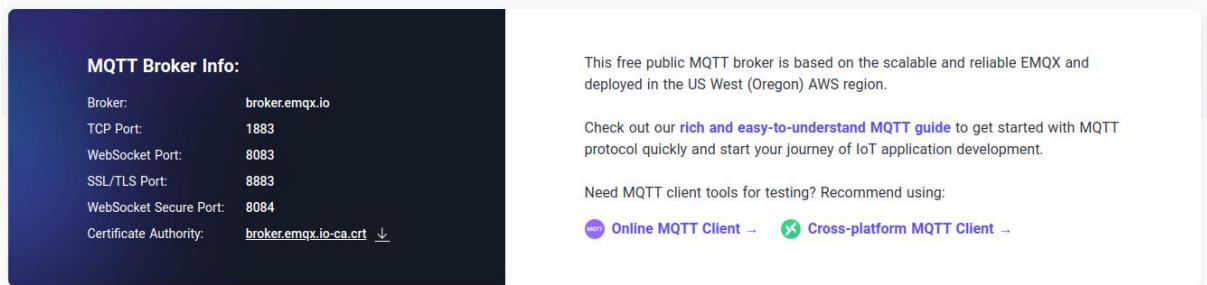
- Publish task: là tác vụ gửi các tin nhắn thông qua giao tiếp MQTT. Tác vụ này nhận nội dung từ “process matrix command” hoặc các câu lệnh tự định nghĩa trên GUI và gửi đến topic “AGV/<AGV_ID>/control”. Sau khi gửi tin nhắn, publish task cũng sẽ cập nhật trạng thái ngược lại UI để người điều khiển nhận biết kết quả.
- Subscribe task: là tác vụ nhận các tin nhắn, sau khi người dùng ấn nút “Connect”, task này sẽ lập tức kết nối và subscribe đến topic “AGV/<AGV_ID>/feedback” và chờ tin nhắn phản hồi từ AGV. Task này sẽ truyền tín hiệu đến UI thông qua capture task sau mỗi lần nhận tin nhắn từ xe.

4.3.4. Giao tiếp MQTT

Khi đã lập trình được các client publish/subscribe, có ba vấn đề nhóm sinh viên phải lựa chọn và thiết kế đó là: Broker sử dụng trong mô hình là gì? Các topic

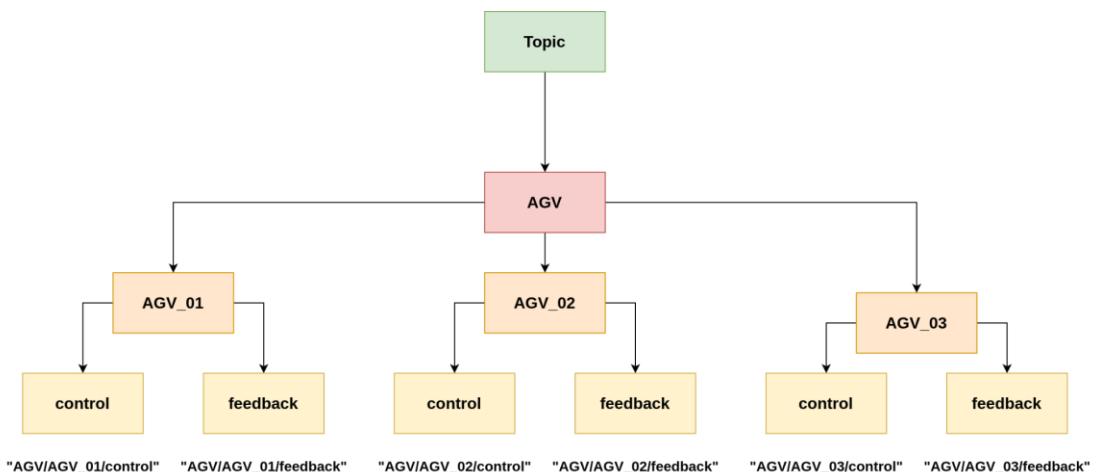
được thiết kế có khả năng thay đổi và mở rộng như thế nào? Cuối cùng là, các tin nhắn được truyền đi với dạng nào?

Về Broker, nhóm sinh viên lựa chọn EMQX Free Broker, một server broker miễn phí với nhiều port giao tiếp TLS/SSL, no TLS/SSL... Nhóm sinh viên lựa chọn EMQX với port 1883 no TLS/SSL. Broker này phục vụ cho mục đích kiểm tra, chạy thử, và một số dự án cá nhân.



Hình 4.18. Các thông tin cơ bản của EMQX được giới thiệu trên trang chủ.

Các topic MQTT được thiết kế đặt tên theo quy tắc phân nhánh để dễ dàng quản lý và mở rộng hệ thống. Hệ thống phân nhánh của các topic được mô tả như hình sau:

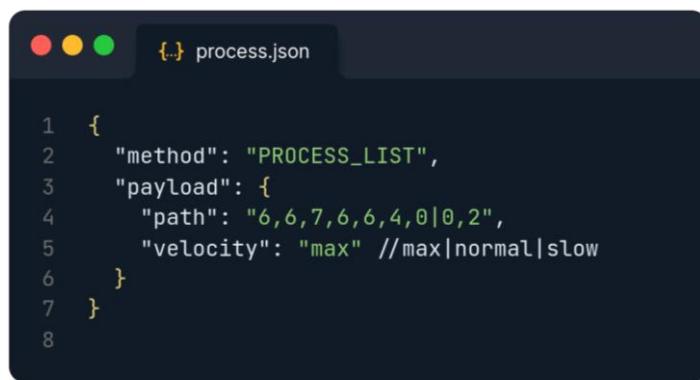


Hình 4.19. Cách các topic được phân nhánh.

Cuối cùng là tin nhắn được truyền tải bằng giao thức MQTT. Các tin nhắn này được truyền đi dưới dạng JSON, và được định dạng theo các nhóm sau:

Bảng 4.2. Bảng mô tả các nhóm lệnh của GUI đến AGV

Dạng câu lệnh	Mô tả
PROCESS_LIST	Thực thi danh sách đường đi và tốc độ mô tả trong “payload”.
COMMAND	Thực thi một command lập trình từ trước chứa trong “payload”.

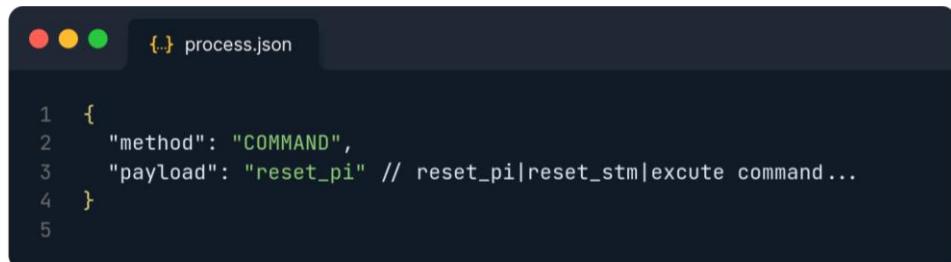


```

1  {
2      "method": "PROCESS_LIST",
3      "payload": {
4          "path": "6,6,7,6,6,4,0|0,2",
5          "velocity": "max" //max|normal|slow
6      }
7  }
8

```

Hình 4.20. Mô hình tin nhắn PROCESS được định dạng theo chuỗi JSON.



```

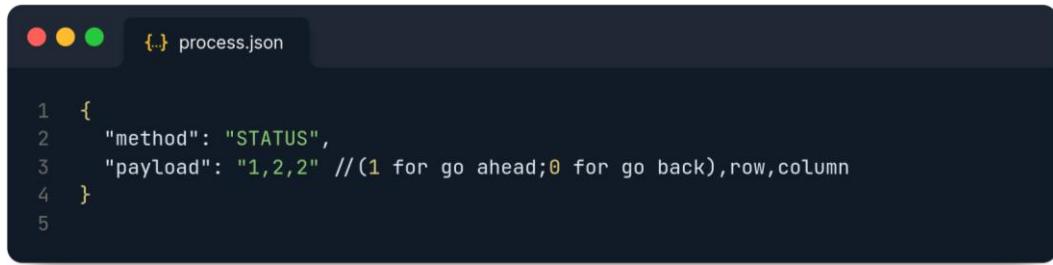
1  {
2      "method": "COMMAND",
3      "payload": "reset_pi" // reset_pi|reset_stm|excute command...
4  }
5

```

Hình 4.21 Mô hình tin nhắn COMMAND được định dạng theo chuỗi JSON.

Bảng 4.3. Mô tả các nhóm lệnh từ AGV đến GUI

Dạng câu lệnh	Mô tả
STATUS	Cập nhật trạng thái và vị trí, hướng.
WARNING	Cảnh báo lỗi.



```
1  {
2      "method": "STATUS",
3      "payload": "1,2,2" // (1 for go ahead; 0 for go back), row, column
4  }
5
```

Hình 4.22. Mô hình tin nhắn phản hồi STATUS.



```
1  {
2      "method": "WARNING",
3      "payload": "00" // warning code
4  }
5
```

Hình 4.23. Mô hình tin nhắn phản hồi WARNING.

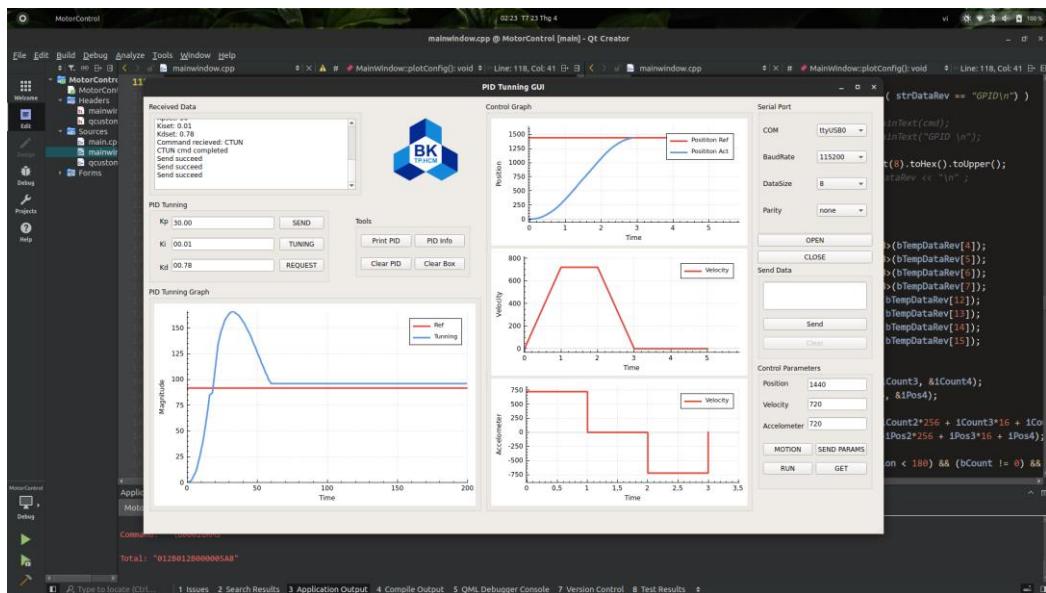
Chương 5. KẾT QUẢ THỰC NGHIỆM

5.1. Kết quả điều khiển

5.1.1. PID & Vận tốc hình thang

Nhóm sinh viên đã tiến hành sử dụng Qt và giao tiếp Serial Port để dựng một giao diện để có thể tuning PID và theo dõi quá trình điều khiển theo phương pháp hình thang, giao diện này cũng một phần thừa hưởng từ quá trình nghiên cứu và thực hiện đồ án 1.

Trong giao diện này, chúng ta dễ dàng xem được các đáp ứng PID và đáp ứng vị trí theo thời gian. Qua đồ thị trong hình nhóm đã thành công xử lý thuật toán vận tốc hình thang, làm cho động cơ xe hoạt động tốt hơn trong quá trình di chuyển.

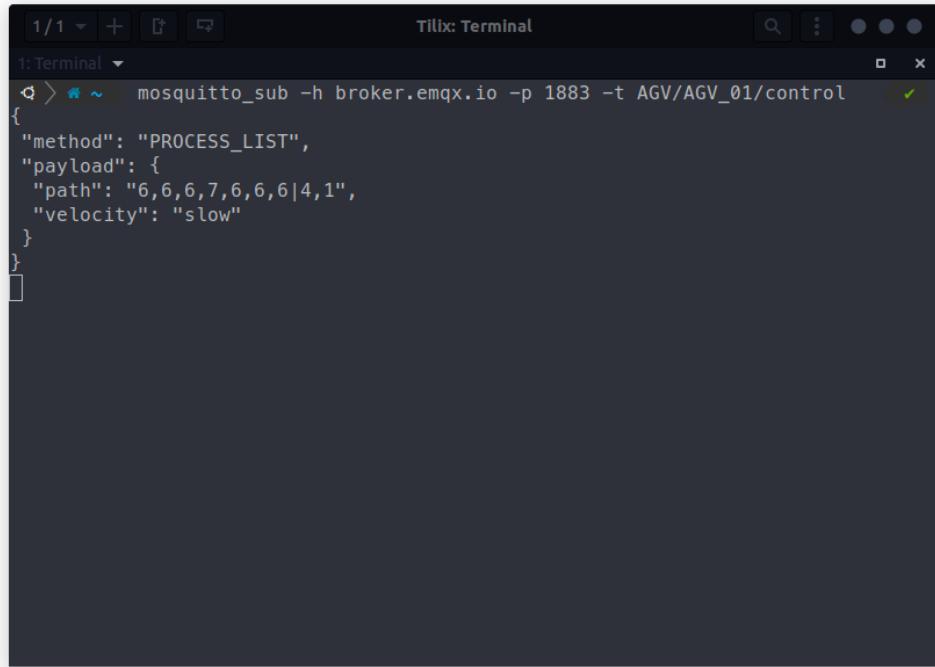


Hình 5.1. Giao diện tuning PID và vận tốc hình thang.

5.1.2. Giao diện điều khiển sử dụng Qt C++ và MQTT

Giao diện điều khiển chính sử dụng Qt và MQTT cũng hoàn thành tốt trong quá trình làm luận văn. Như yêu cầu đã được đề ra, giao diện thể hiện được khả năng có thể gửi câu lệnh chuỗi đường đi từ máy chủ xuống robot AGV. Đồng thời giao diện cũng tiếp nhận được và xử lý thông tin phản hồi từ xe.

Ngoài ra giao diện cũng có khả năng điều chỉnh tốc độ cơ bản. Đáp ứng được nhu cầu điều khiển, theo dõi và vận hành của người dùng.

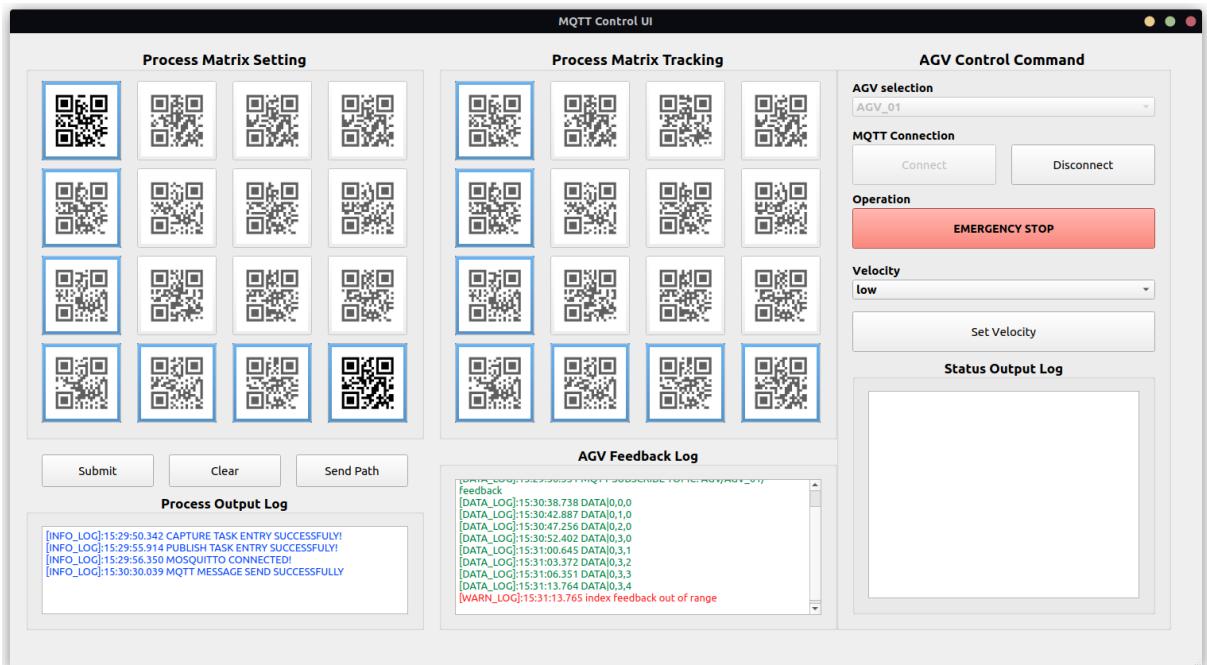


```

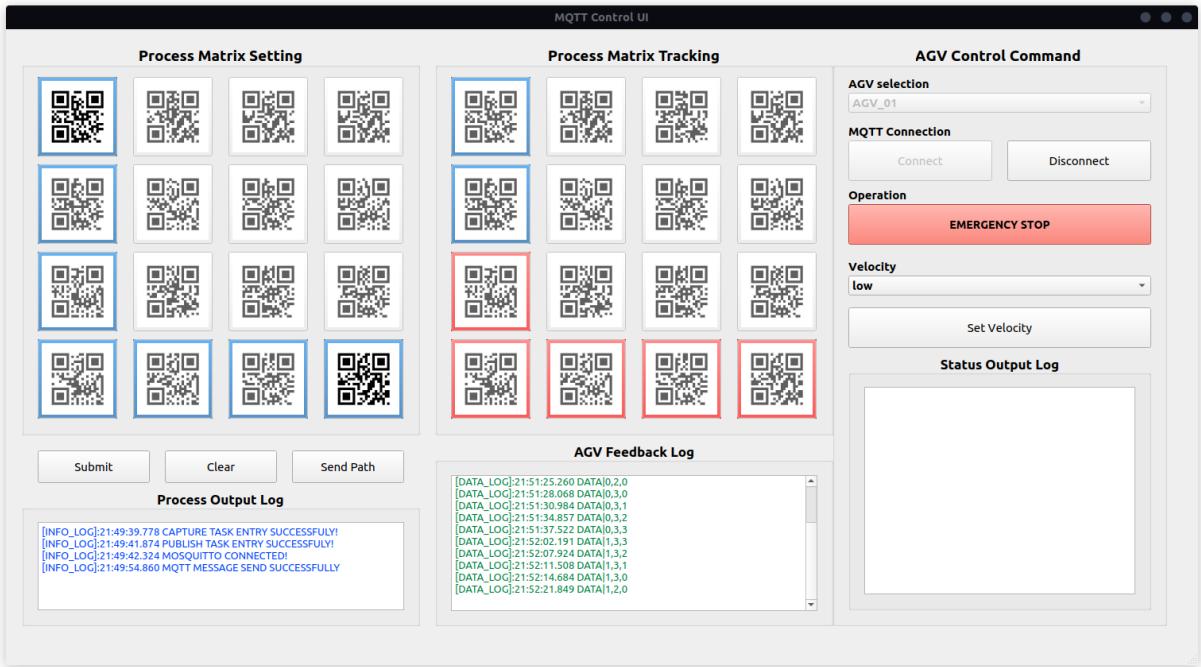
Tilix: Terminal
1: Terminal ▾
> mosquitto_sub -h broker.emqx.io -p 1883 -t AGV/AGV_01/control
{
  "method": "PROCESS_LIST",
  "payload": {
    "path": "6,6,6,7,6,6,6|4,1",
    "velocity": "slow"
  }
}

```

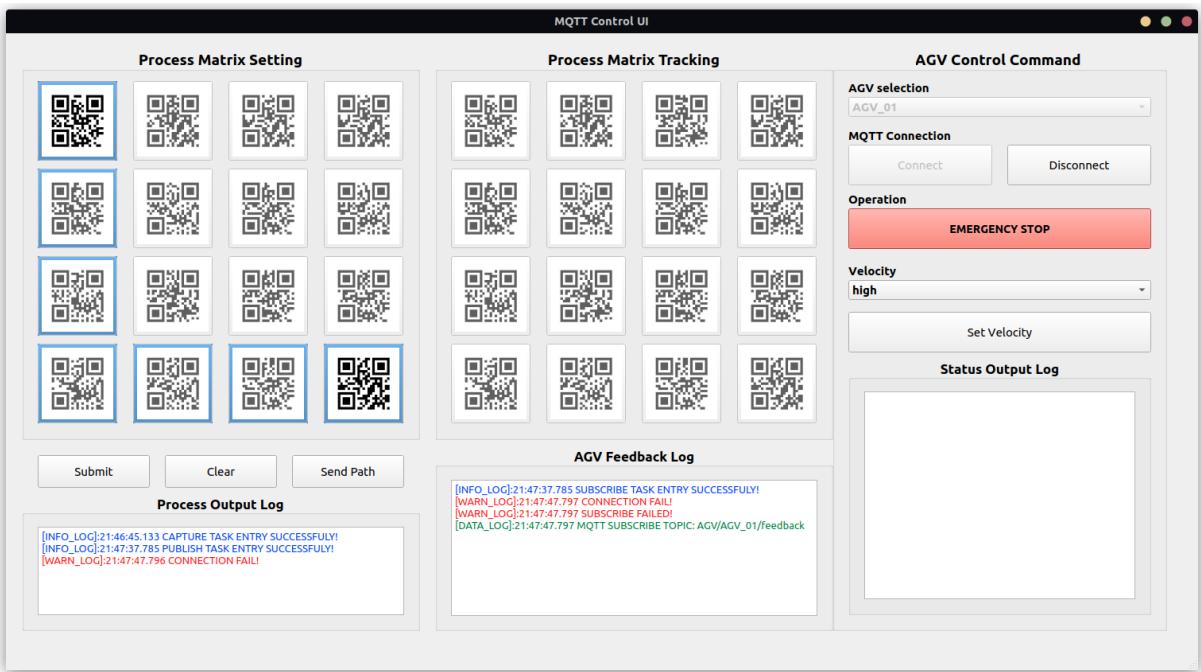
Hình 5.2. Tin nhắn được truyền đi bởi GUI (kiểm thử bằng terminal).



Hình 5.3. Vùng phản hồi khi xe di chuyển lượt đi được thể hiện màu xanh.



Hình 5.4, Vùng phản hồi khi xe di chuyển lượt về được thể hiện qua màu đỏ.



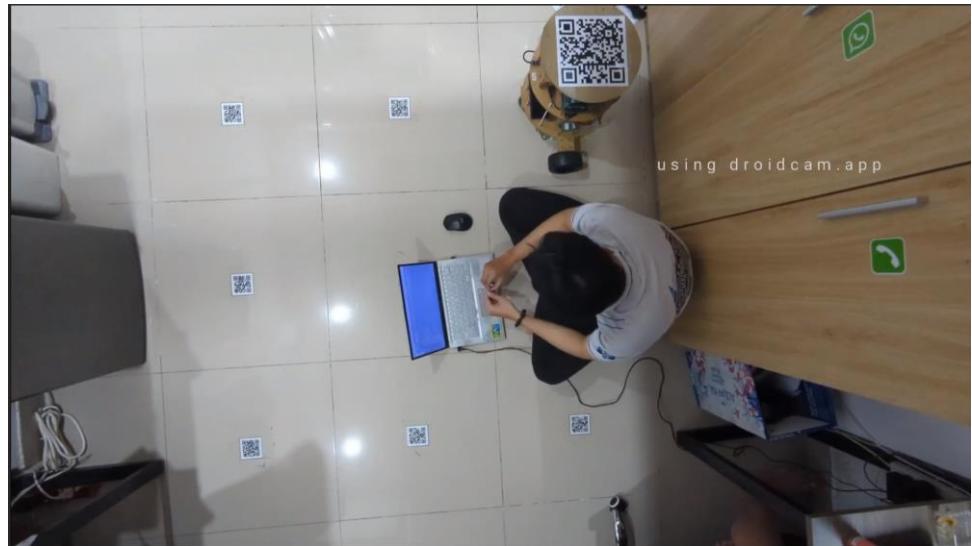
Hình 5.5. GUI báo lỗi được thể hiện qua đoạn log màu đỏ.

5.2. Kết quả điều khiển, định vị thông qua QR Code

5.2.1. Mô hình triển khai thực nghiệm

Triển khai mô hình đánh giá quỹ đạo, nhóm sinh viên thực hiện đính camera góc rộng lên trần nhà, sau đó sử dụng phần mềm đánh giá quỹ đạo đã lập trình ở

mục 4.4, nhóm sinh viên tiến hành thu thập và đánh giá trên cùng một đường đi với các điều kiện khác nhau như khoảng cách giữa các mã khác nhau, vận tốc thay đổi từ chậm đến nhanh, cường độ ánh sáng thay đổi.



Hình 5.6. Triển khai thực hiện với camera được gắn trên trần.

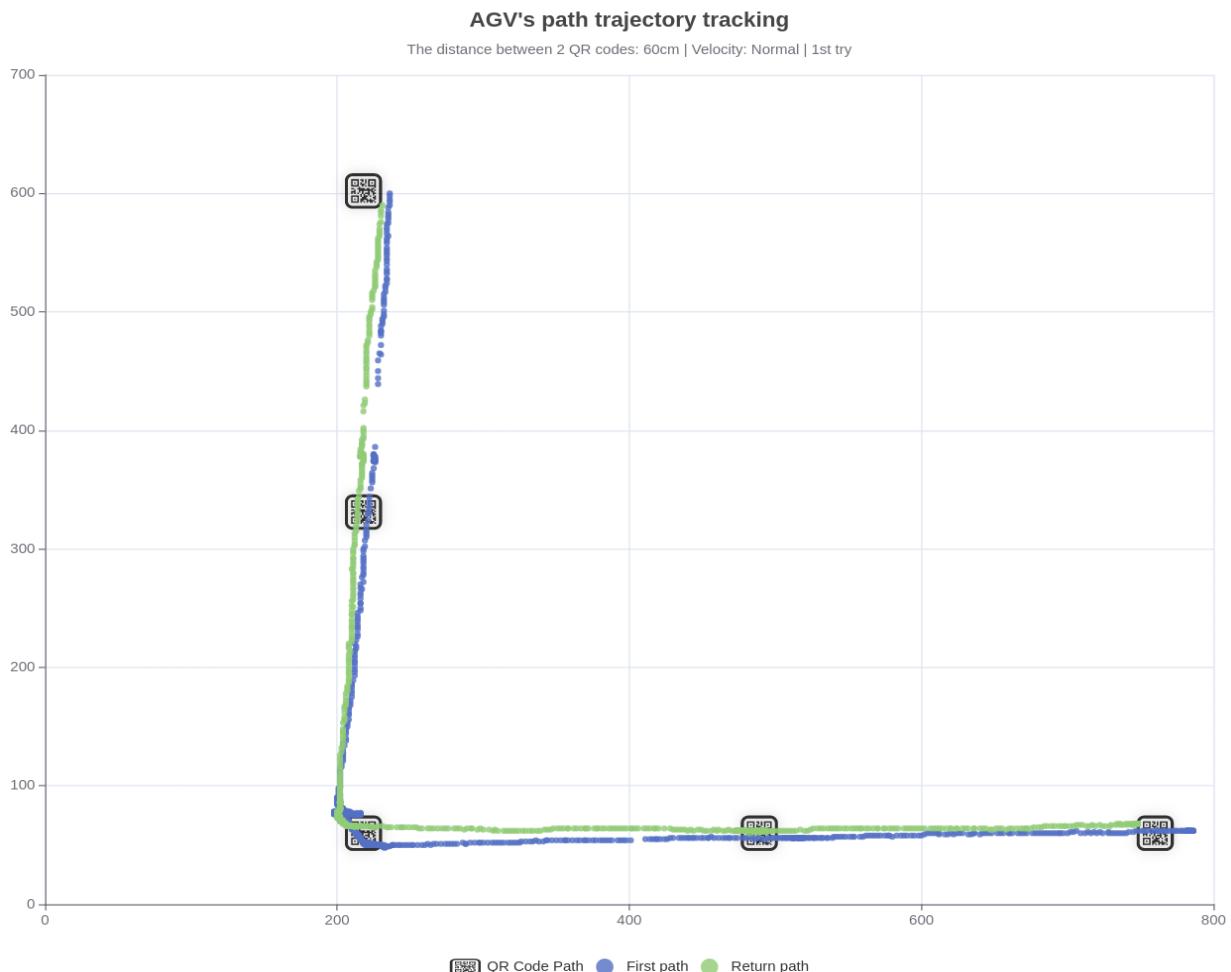
Sai số khi đánh giá các lần thực nghiệm được xét theo độ lệch giữa tâm QR Code và tâm frame ảnh của xe.



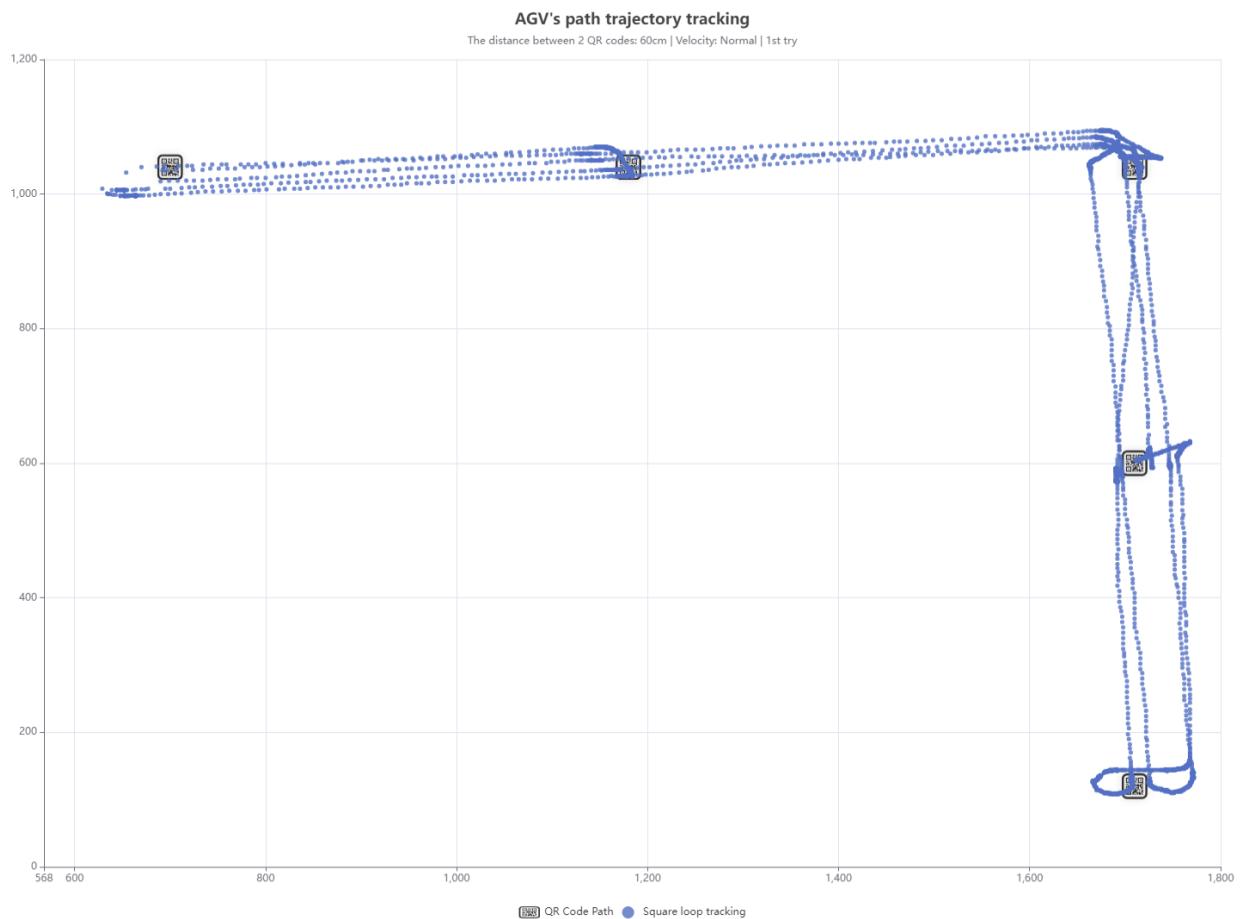
Hình 5.7. Cách tính độ lệch giữa mã QR Code và xe bằng Camera.

5.2.2. Đánh giá quỹ đạo với điều kiện ổn định

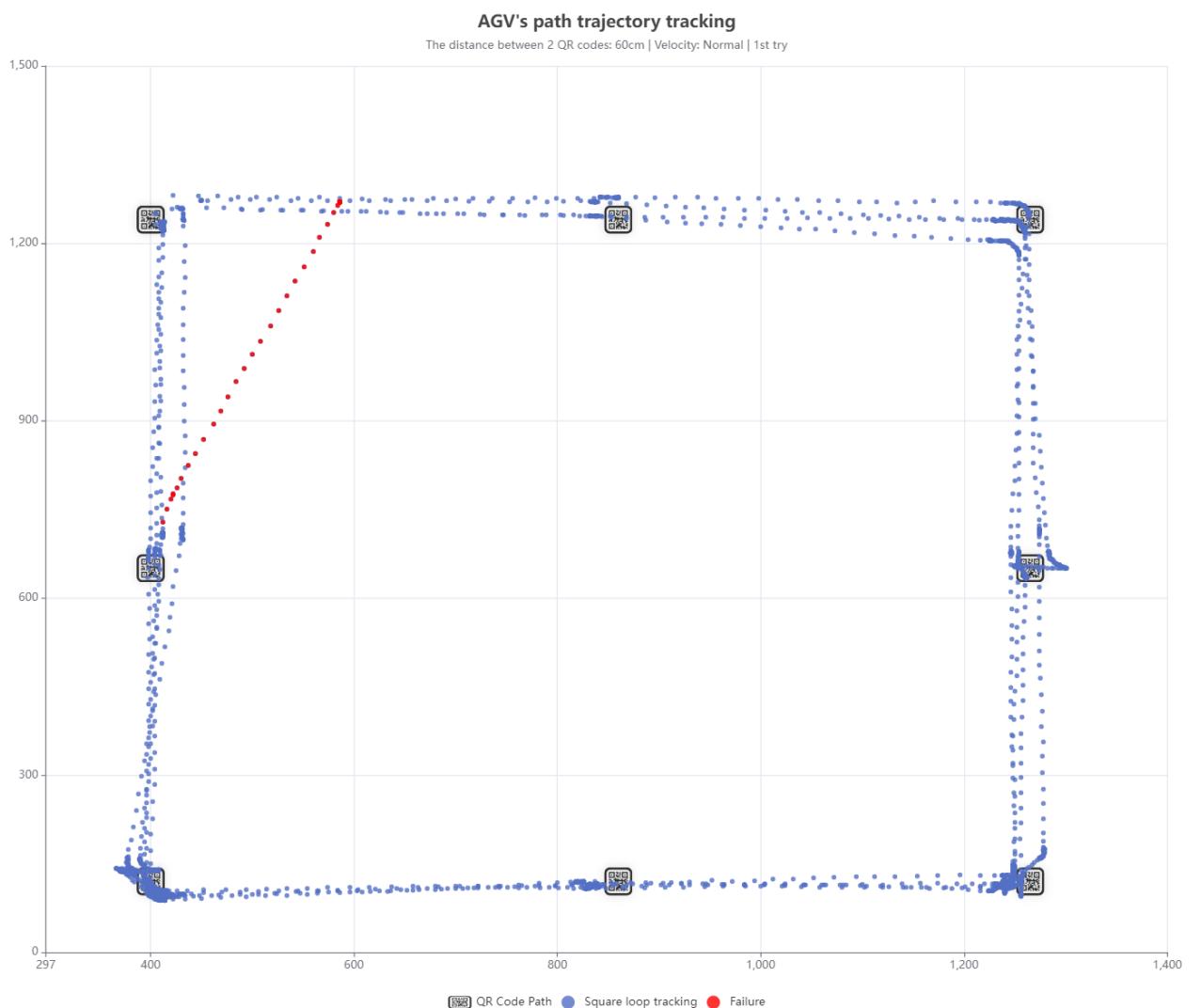
Mô hình thực nghiệm đầu tiên, xe chạy với khoảng cách giữa các mã ổn định nhất là 60cm, với tốc độ quay trung bình của bánh xe là 80 độ/giây.



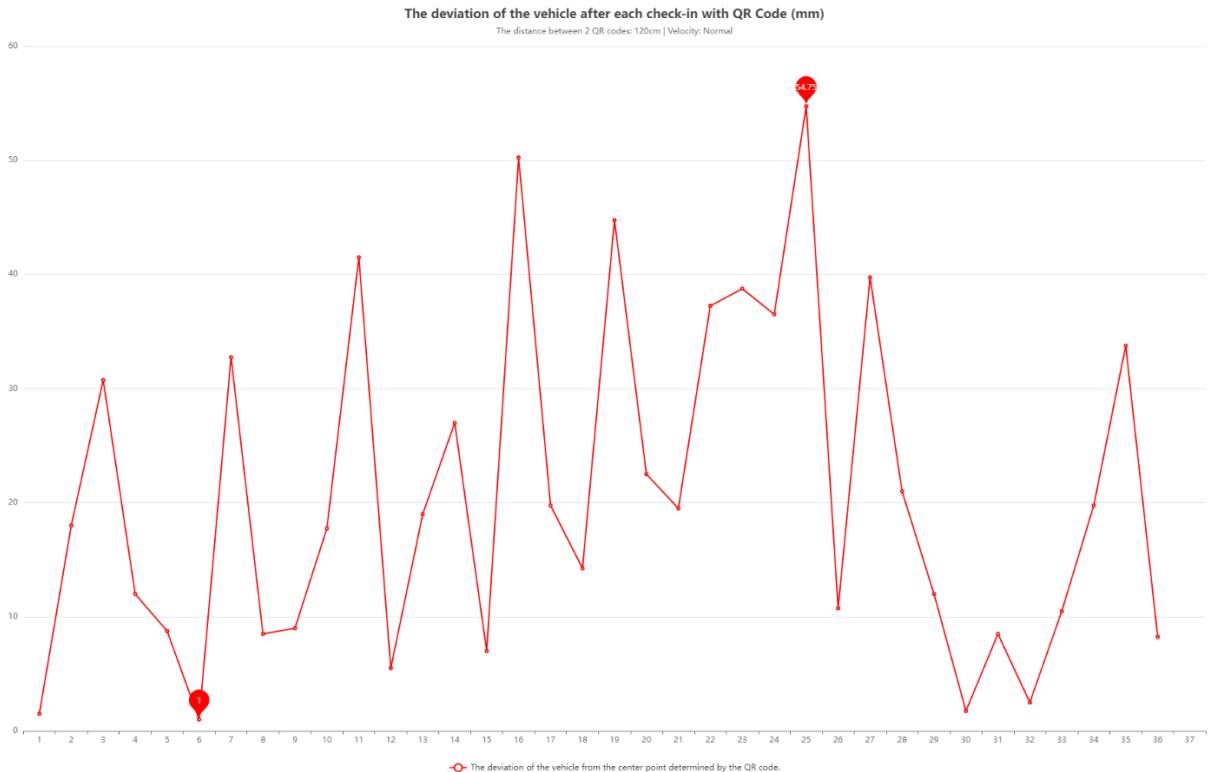
Hình 5.8. Lần kiểm thử đầu tiên.



Hình 5.9. Thử nghiệm chạy liên tục với số lần lặp là 5 lần.



Hình 5.10. Chạy thử nghiệm với đường đi được cho trước.



Hình 5.11. Sai số sau mỗi lần check-in tại các mã QR tính theo mm.

Đánh giá tổng quan về quỹ đạo của xe trong suốt quá trình di chuyển giữa 5 mã QR Code:

Ưu điểm:

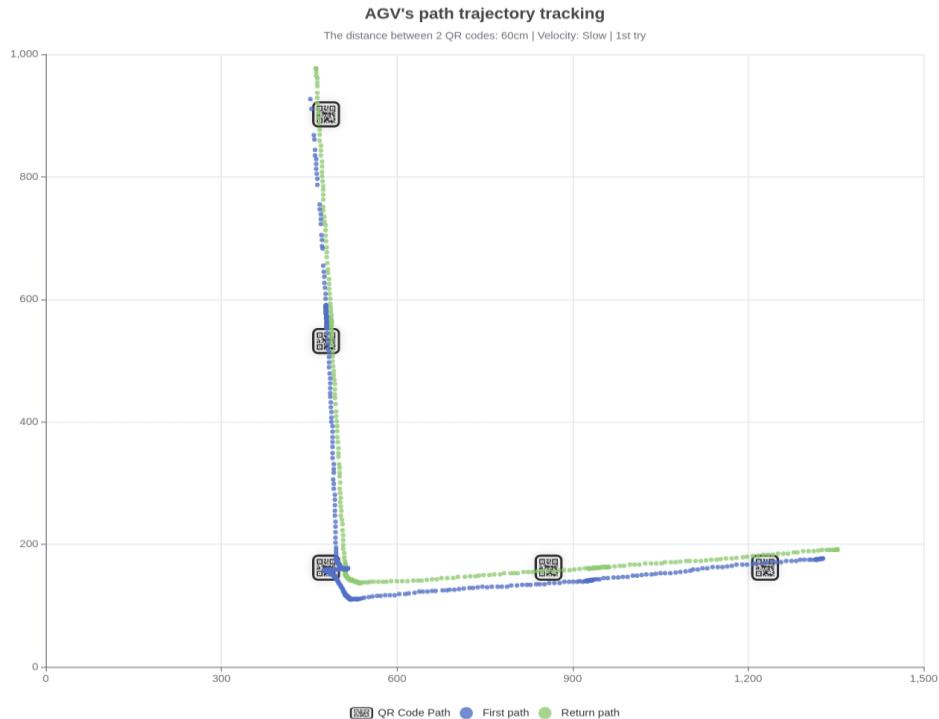
- Xe có quỹ đạo đi và quỹ đạo về khá ổn định, tỷ lệ hoàn thành đường đi thành công trong các lần thực nghiệm là 100%.
- Trong giai đoạn đầu, xe gần như chạy thẳng mà không cần lấy lại góc chuẩn giữa các quá trình.
- Góc lệch của robot sau khi di chuyển giữa hai mã QR so với đường thẳng đề ra luôn dao động dưới 5 ~ 10 độ.

Nhược điểm:

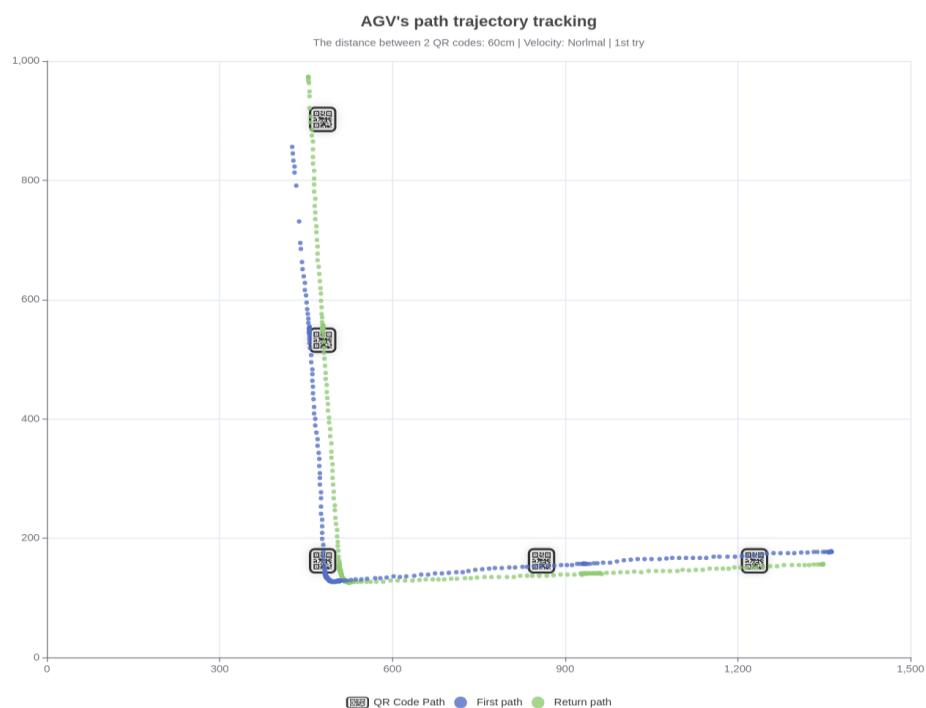
- Trong giai đoạn quay 90 độ, xe khá mất thời gian trong việc lấy lại góc chuẩn vì nhóm sinh viên lập trình cho xe tiến hành thực hiện 3 bước trong quá trình này, bao gồm: Lấy góc chuẩn, xoay, lấy lại góc chuẩn lần 2 nếu camera xác định góc lệch lớn hơn giá trị cho phép.

- Sau khi quay 90 độ, mặc dù đã thực hiện các động tác căn chỉnh nhưng vẫn tồn tại xu hướng đi lệch khỏi quỹ đạo cao hơn trước đó.

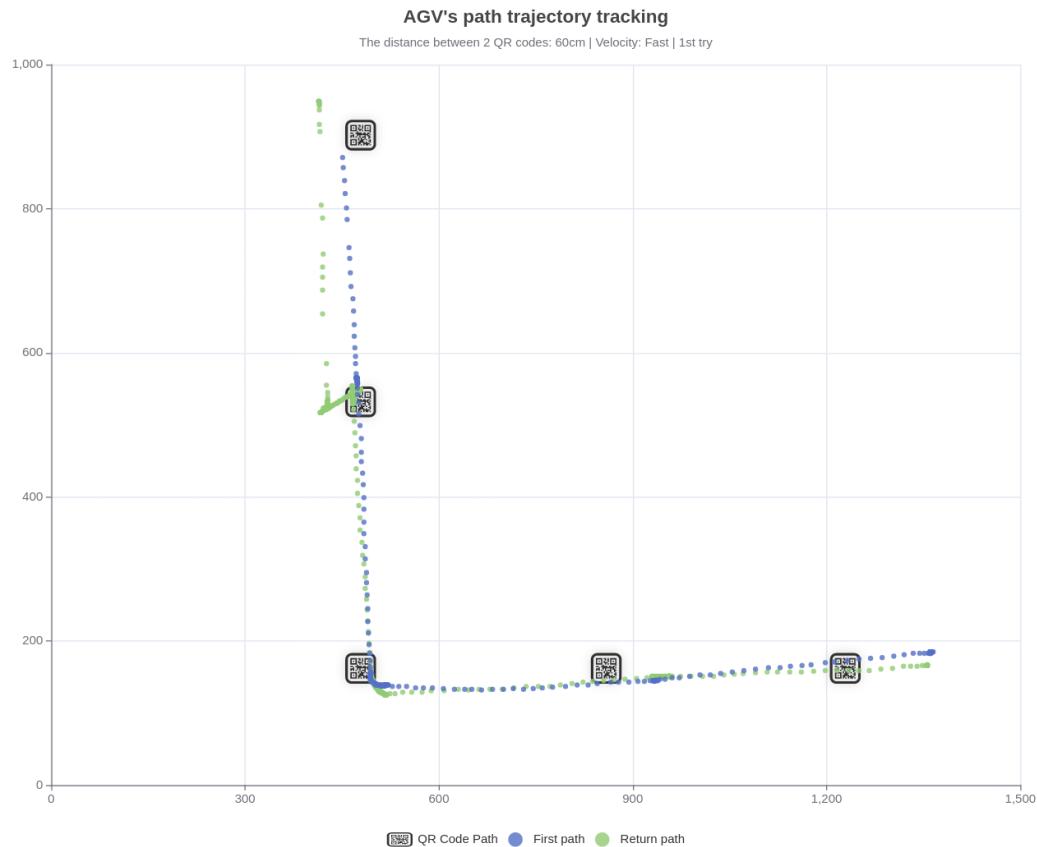
5.2.3. Kết quả chạy thử trên các vận tốc khác nhau



Hình 5.12. Thử nghiệm với vận tốc chậm.



Hình 5.13. Thử nghiệm với vận tốc trung bình.



Hình 5.14. Thử nghiệm với vận tốc nhanh.

Đánh giá tổng quan về quỹ đạo của xe trong suốt quá trình di chuyển giữa 5 mã QR Code:

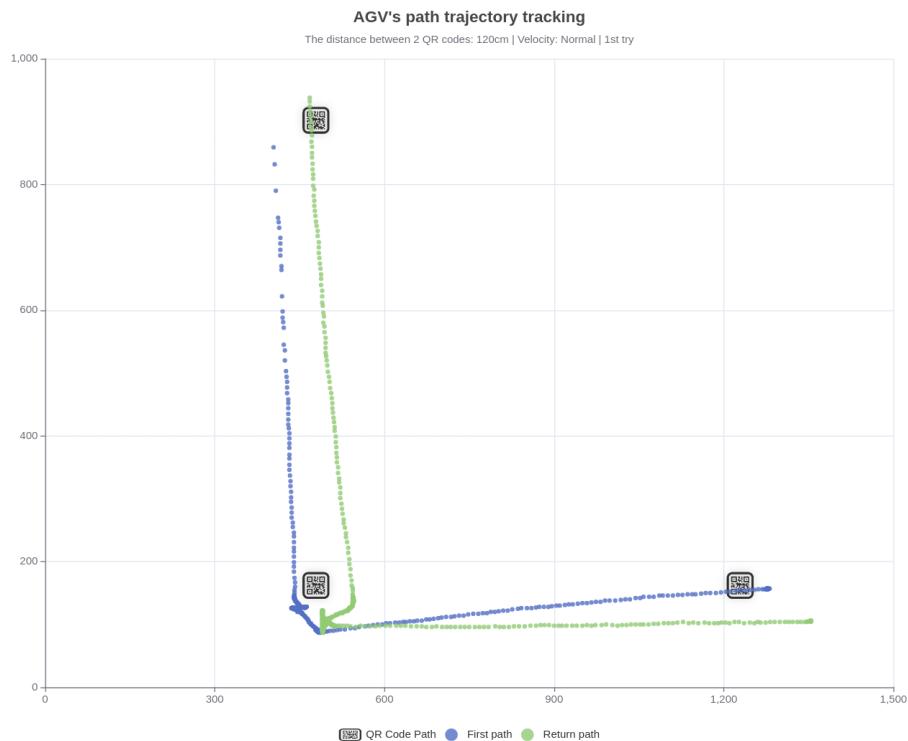
Ưu điểm:

- Xe vẫn hoàn thành tốt quỹ đạo trong các vận tốc khác nhau. Ngưỡng vận tốc cao nhất có thể đạt được là khi bánh xe xoay với vận tốc 120 độ/giây. Qua ngưỡng này, xe có dấu hiệu không đáp ứng được do giới hạn phần cứng cũng như khả năng lập trình xử lý của nhóm sinh viên chưa tốt.

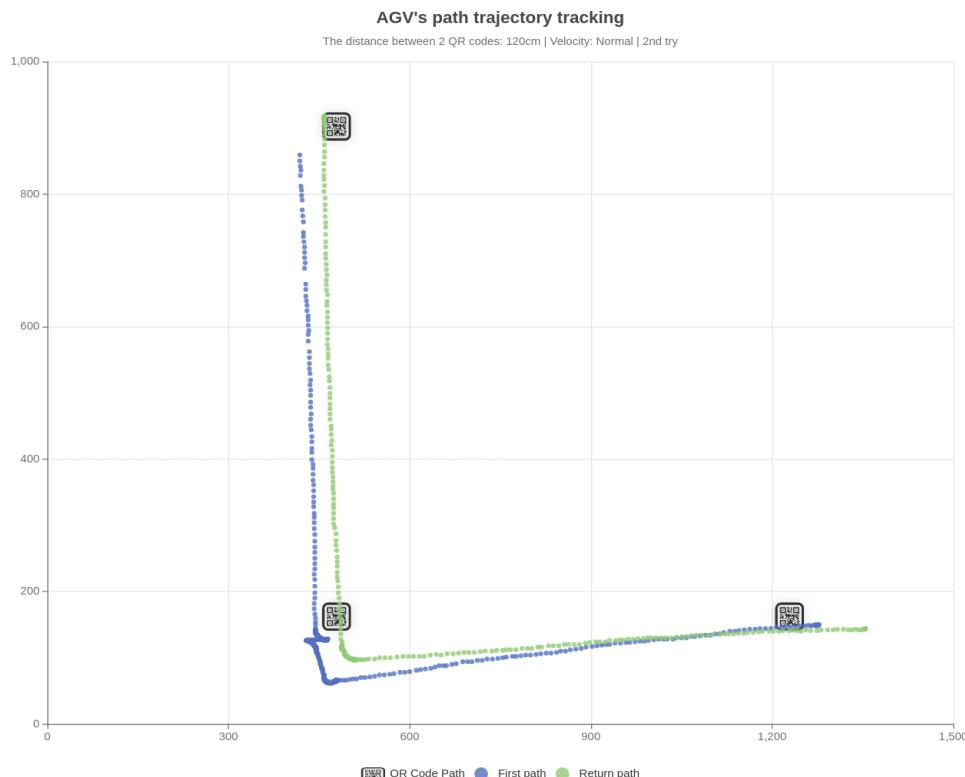
Nhược điểm:

- Với tốc độ cao nhất là 0.3m/s, xe vẫn chưa đáp ứng được tốc độ cần thiết trong công nghiệp, trung bình 0.5m/s.
- Mặc dù xe có thể hoàn thành được quỹ đạo với tỉ lệ thành công 80%, nhưng trong suốt quá trình di chuyển xe còn nhiều điểm bất ổn định.

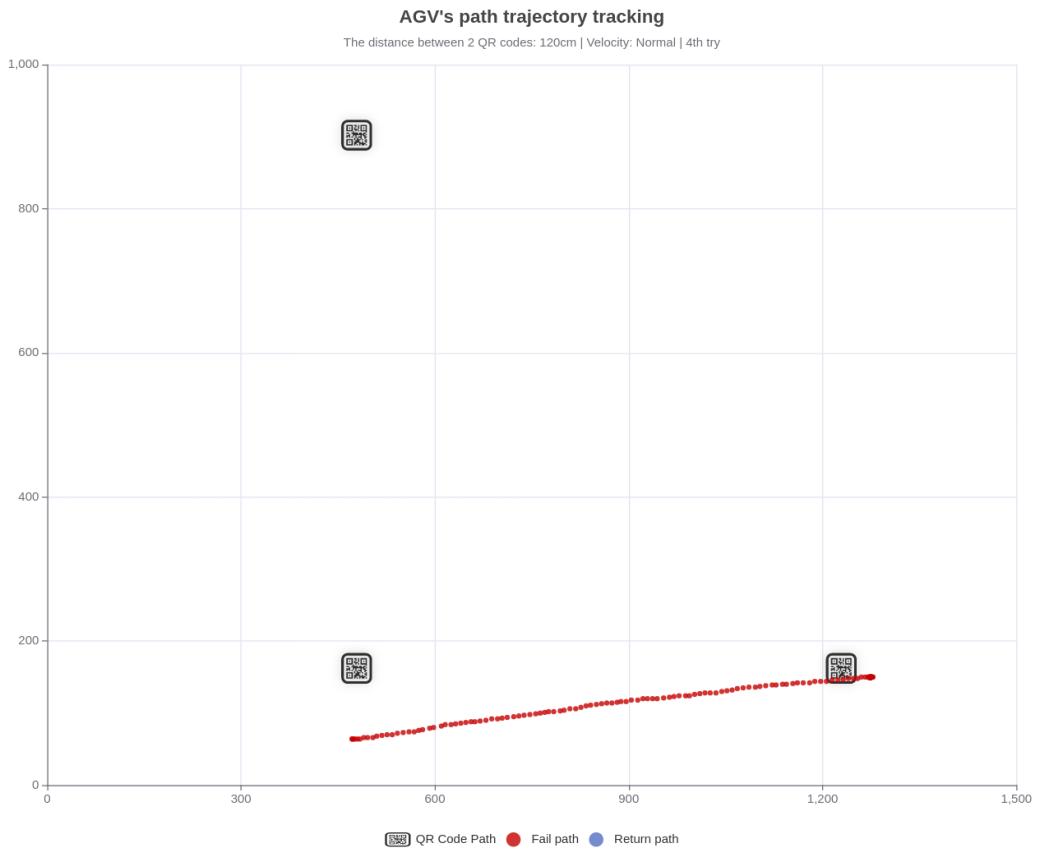
5.2.4. Kết quả chạy thử trên khoảng cách dài hơn



Hình 5.15. Thử nghiệm với quãng đường là 1m2, lần 1.



Hình 5.16. Thử nghiệm với quãng đường 1m2 lần 2.



Hình 5.17. Thử nghiệm lần 3 bị thất bại.

Đánh giá tổng quan về quỹ đạo của xe trong suốt quá trình di chuyển giữa 3 mã QR Code, khoảng cách 120 cm:

Ưu điểm:

- Xe vẫn hoàn thành quỹ đạo được với tỉ lệ thành công là 60%.

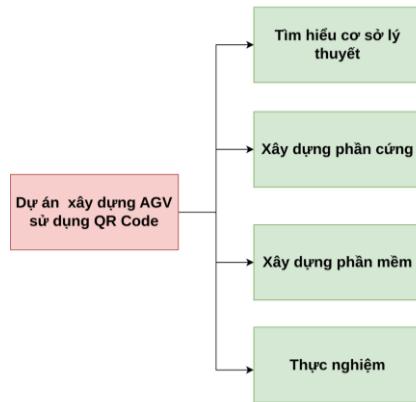
Nhược điểm:

- Với góc lệch cam cho phép là 10 độ, nhưng vì khoảng cách quá dài, nên độ lệch tích lũy cao, dẫn đến xe dễ dàng mất đi quỹ đạo và không nhận diện được mã QR tiếp theo.
- Xe mất khá nhiều thời gian để căn chỉnh góc độ giữa xe và mã QR trước khi di chuyển tới mã QR tiếp theo.

Tổng kết		
		Nhận xét
Vận tốc	0.15m/s	Xe chạy ổn định và có thể chạy với tần suất liên tục mà không bị lỗi với tốc độ này. Xe có thể đáp ứng 0.3m/s nhưng với độ ổn định không cao.
Khoảng cách giữa các mã QR	0.6m	Khoảng cách xa nhất có thể đạt được là 1.2m, tuy nhiên với khoảng cách này xe có độ lệch tích lũy cao.
Độ ổn định khi chạy lắp lại	80% trong trường hợp ổn định.	Trong các trường hợp xe chạy với vận tốc và khoảng cách giữa hai mã là 0.15m/s và 0.6m, xe có khả năng chạy liên tục với độ ổn định cao.
Sai số nhỏ nhất	1mm	Sai số được giới hạn ở mức thấp vì trước mỗi lần chạy, xe sẽ tiến hành đọc và xác định độ lệch của xe và mã, từ đó tính toán căn chỉnh độ lệch về mức thấp nhất.
Sai số lớn nhất	54mm	
Sai số trung bình giữa 2 mã	20.74mm	

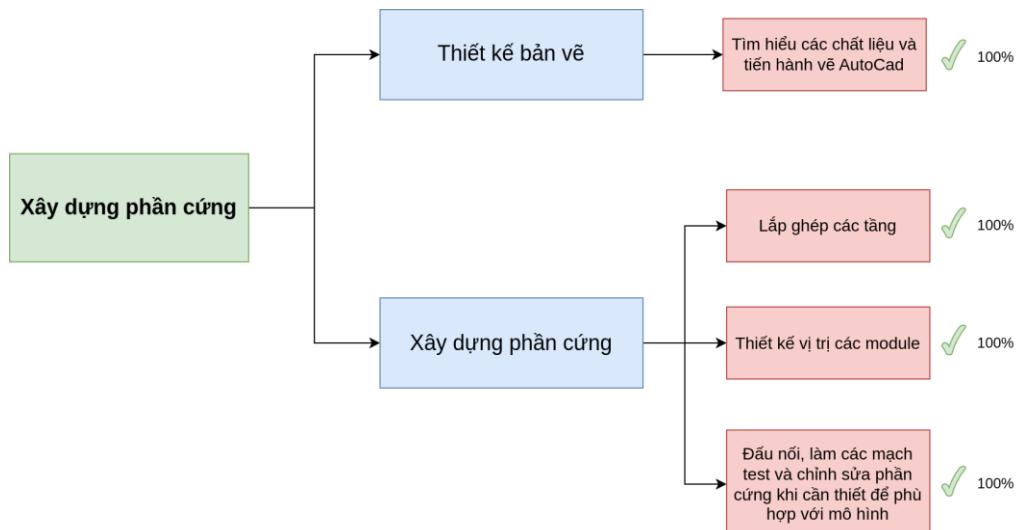
Chương 6. ĐÁNH GIÁ KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1. Đánh giá kết quả



Hình 6.1. Các nhiệm vụ chia nhỏ cần phải làm.

- Dựng được mô hình phần cứng



Hình 6.2. Các nhiệm vụ cần thực hiện của phần cứng.

Qua quá trình thiết kế, xây dựng, thử nghiệm nhiều lần để có thể dựng được phiên bản phần cứng ổn định và tốt nhất, nhóm sinh viên đã xây dựng thành công mô hình phần cứng có thể sử dụng không chỉ để thử nghiệm với mã QR mà có thể tích hợp thêm nhiều giải pháp khác.

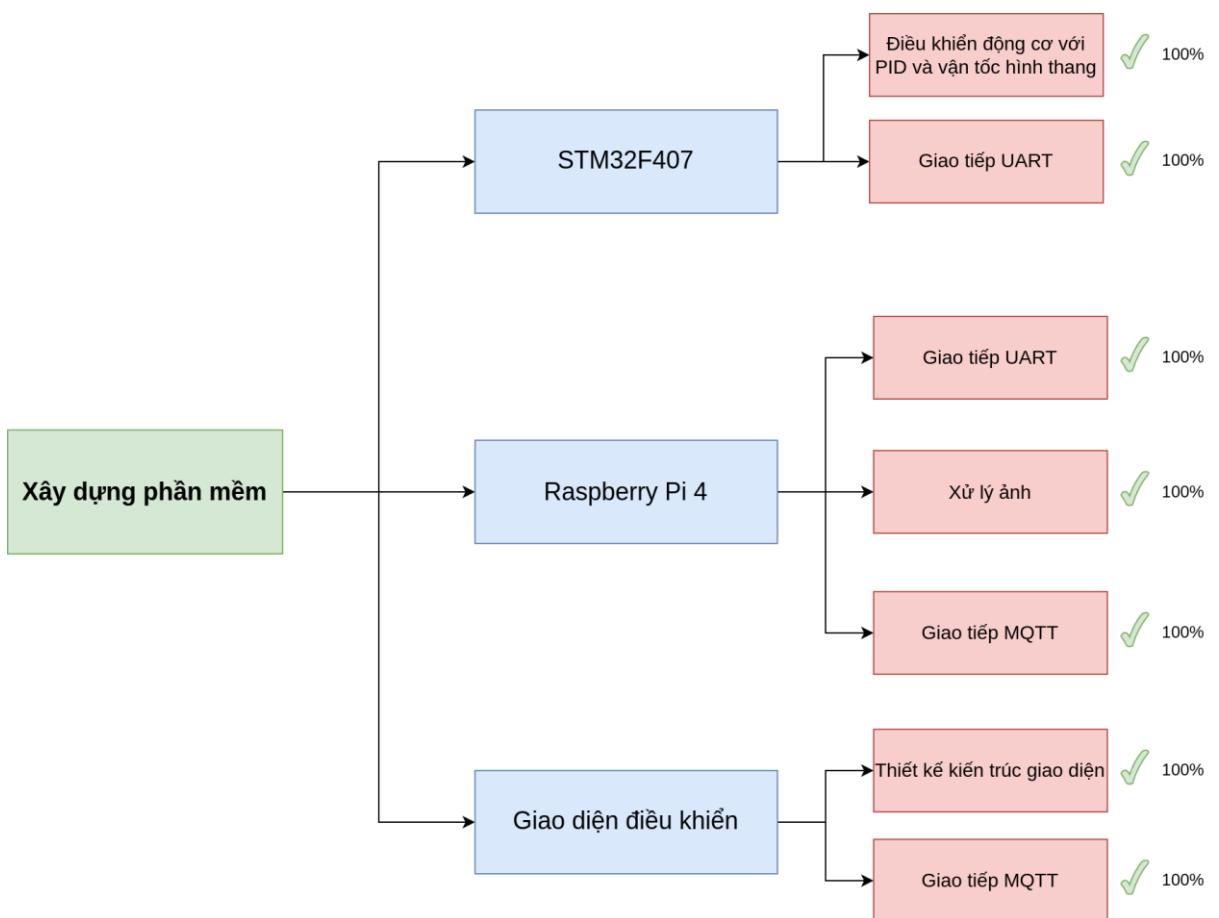
Mô hình sau khi xây dựng có cấu tạo từ mica. Ban đầu, nhóm sinh viên đã sử dụng mica độ dày 3mm. Tuy nhiên, trong quá trình thực nghiệm, mica với độ dày nhỏ là xe dễ bị vênh nếu chưa vật nặng, sau nhiều lần chỉnh sửa thì khung xe có

độ dày mica là 8mm cho tầng 1 và 5mm cho tầng 2. Với độ dày mica đáng tin cậy này và 4 trụ ốc 6 ly, xe có độ hoàn thiện cao và không bị cong vênh.

Tuy xe đã cứng cáp nhưng trong quá trình thực nghiệm, nhóm sinh viên nhận thấy xe còn chưa đủ kích thước so với một chiếc AGV thông thường, ngoài ra trong quá trình duy chuyển, cần thiết kế số lượng bánh xe và các phương pháp đặt bánh xe tốt hơn để vận hành chuẩn xác hơn.

Động cơ được dùng trong xe là động cơ loại nhỏ, có tỉ lệ hộp số lớn để có thể kéo xe cùng với ắc quy, nên vận tốc xe chưa được cao. Các kit để kết nối giữa STM32F407 và các module động cơ Encoder hiện tại vẫn là mạch test đục lỗ thủ công. Mặc dù đã có những kết nối chắc chắn thông qua Domino, nhưng nhóm sinh viên vẫn cần hoàn thiện mạch in để tránh trường hợp thiết ổn định sau này.

- **Triển khai thành công lý thuyết sang lập trình chạy thực tế:**



Hình 6.3. Các nhiệm vụ cần thực hiện của phần mềm.

Trong quá trình lập trình phần mềm, nhóm sinh viên nhận thấy việc tổng

thể kế hoạch bao gồm 3 phần: nhiệm vụ điều khiển động cơ dựa trên lệnh từ Raspberry Pi, nhiệm vụ xử lý và phân tích hình ảnh trên máy tính nhúng và lập trình giao diện người dùng.

Về lập trình điều khiển động cơ, nhóm sinh viên đã thành công trong việc lập trình STM32 với C, điều khiển vòng kín PID và sử dụng vận tốc hình thang. Các câu lệnh từ trên máy tính nhúng truyền xuống điều được xử lý thành công và chuyển đổi thanh các thông số cần vận hành.

Về lập trình máy tính nhúng để truyền nhận và xử lý ảnh, nhóm sinh viên đã thành công trong việc lập trình truyền nhận giữa vi điều khiển và máy tính nhúng thông qua UART và giao tiếp giữa xe và giao diện điều khiển Qt C++ thông qua MQTT. Bên cạnh đó, công việc xử lý và tính toán mã QR như góc nghiêng của mã so với xe, chuyển đổi góc nghiêng thành các thông số vận tốc, hướng quay, góc quay.

Công việc cuối cùng là giao diện điều khiển, nhóm sinh viên đã hoàn thành giao diện điều khiển dễ nhìn, dễ điều khiển và đầy đủ các tính năng như: điều chỉnh tốc độ, thiết lập được đường đi bằng hai điểm đầu cuối do người dùng chọn, dừng xe khẩn cấp, theo dõi được quỹ đạo khi xe tới các điểm QR, ghi lại log.

- **Triển khai thực nghiệm, lấy mẫu quỹ đạo**

Sau khi hoàn thiện được phần cứng, phần mềm. Nhóm sinh viên đã thành công trong việc chạy thử trên các ma trận 2x2, 3x3, 4x4. Ngoài việc ghi lại được các video thực nghiệm, sinh viên còn tiến hành dùng xử lý ảnh để trích xuất đường đi để dễ dàng đánh giá. Kết quả thực nghiệm tốt và chứng minh được giải pháp có tính khả thi.

6.2. Kết luận

Tuy vẫn còn một số hạn chế về giải thuật điều khiển và giới hạn phần cứng (Tốc độ xe chưa đạt được tốc độ phù hợp với thực tế, chưa tích hợp các cảm biến khác để xe có thể đi chính xác hơn...), nhưng mô hình đã cơ bản đáp ứng được các mục tiêu của đề tài là xây dựng mô hình xe tự hành trong công nghiệp với giải pháp sử dụng mã QR Code để điều hướng và định vị. Bao gồm hoàn thành các tác vụ chia

nhỏ như sau:

- Thiết kế và xây dựng thành công mô hình phần cứng phù hợp với việc thử nghiệm, chạy thử, lấy mẫu.
- Thiết kế và lập trình mô hình phần mềm có thể điều khiển động cơ thông qua xử lý mã QR Code, xác định được góc, độ lệch, tọa độ.
- Thực hiện giao tiếp và điều khiển từ xa.

Tuy vẫn có khả năng định vị và điều hướng, nhưng để tăng tính ổn định trong công nghiệp, xe vẫn cần phải kết hợp với các phương pháp điều khiển khác..

6.3. Hướng phát triển

- Tích hợp thêm các cảm biến định vị khác như: Lidar, IMU nhằm tăng được độ chính xác khi xe di chuyển, đồng thời có khả năng phục hồi sau khi xe có xu hướng đi lệch, đồng thời cải thiện thuật toán tốt hơn nhờ các cảm biến này, giảm thời gian cấn chỉnh góc sau mỗi lần đến mã QR Code.
- Cải thiện khả năng điều khiển động cơ, tích hợp các bộ lọc nhiễu Encoder và các cảm biến khác.
- Xây dựng thêm phần cứng có thể chở hàng, nâng, bốc tải và tiến hành thực nghiệm trong các nhà máy.
- Xây dựng server monitor để các thông số và trạng thái xe, đồng thời lưu dữ liệu hành trình và các thông số trên vào database để việc sửa lỗi, debug dễ dàng hơn nếu xe có gặp lỗi. Việc monitor theo thời gian cũng có thể giúp đội ngũ phát triển biết được điểm yếu và điểm mạnh của xe, dễ dàng nâng cấp hoặc cải thiện.

TÀI LIỆU THAM KHẢO

- [1]. Huijuan Zhang, Chin Yin Chen, “Localization and navigation using QR code for mobile robots in indoor environment”, 2015.
- [2]. Payam Nazemzadeh, Daniele Fontanelli, David Macii, Luigi Palopoli, “Indoor Localization of Mobile Robots through QR Code Detection and Dead Reckoning Data Fusion”, 2017.
- [3] Gokhan ATALI, Zeynep GARIP, S.Serdar OZKAN, Durmuş KARAYEL, “Path planning of mobile robots based on QR Code”, 2018.
- [4] Tansuriyavong Suriyon, Higa Keisuke, Boonmee Choompol, “Development of Guide Robot by Using QR Code Recognition”, 2011.
- [5] Dhara Patel, Saurabh Upadhyay, "Optical Flow Measurement using Lucas-Kanade Method", 2013.