

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT
KHOA ĐIỆN – ĐIỆN TỬ



BÁO CÁO ĐỒ ÁN TỐT NGHIỆP

NGÀNH: KỸ THUẬT ĐIỀU KHIỂN VÀ TỰ ĐỘNG HÓA

ĐỀ TÀI:

**ĐIỀU KHIỂN VÀ GIÁM SÁT MÔ HÌNH ROBOT DELTA
SẮP XẾP SẢN PHẨM ỨNG DỤNG HỌC SÂU**

Giảng viên hướng dẫn: TS. Cao Nguyễn Khoa Nam

Sinh viên thực hiện: Cao Văn Nhật Triều

Châu Ngọc Vinh

Đà Nẵng, tháng 01 năm 2025

NHẬN XÉT CỦA NGƯỜI HƯỚNG DẪN

Đà Nẵng, ngày tháng năm

(Ký, ghi rõ họ tên)

NHẬN XÉT CỦA NGƯỜI PHẢN BIỆN

Đà Nẵng, ngày tháng năm

(Ký, ghi rõ họ tên)

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ
THUẬT
KHOA ĐIỆN-ĐIỆN TỬ

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

ĐỀ CƯƠNG ĐỒ ÁN TỐT NGHIỆP

1. Họ và tên sinh viên: Cao Văn Nhật Triều Mã sinh viên: 211155120168
Châu Ngọc Vinh Mã sinh viên: 211150120

2. Llop: 125DATNTDHKS01

3. GVHD : TS. Cao Nguyễn Khoa Nam

4. Đề tài

Tên đề tài: *Điều khiển và giám sát hệ thống sắp xếp vật ứng dụng học sâu*

Thời gian thực hiện: Từ tháng 8 đến tháng 12

5. Mục tiêu

- **Thiết kế và chế tạo:** Thiết kế và chế tạo thành công một mô hình robot Delta ba bậc tự do có khả năng hoạt động ổn định.
 - **Phát triển hệ thống thị giác máy:** Úng dụng công nghệ xử lý ảnh kết hợp với Deep Learning, sử dụng mạng nơ-ron tích chập (CNN) như ResNet hoặc YOLO, để nhận diện và phân loại sản phẩm lỗi.
 - **Lập trình hệ thống điều khiển:** Xây dựng chương trình điều khiển tích hợp để robot có thể thực hiện chính xác các thao tác gấp, điều chỉnh hướng của sản phẩm và thả sản phẩm theo kết quả phân loại từ hệ thống thị giác.
 - **Mô phỏng và thực nghiệm:** Tiến hành mô phỏng và thử nghiệm trên mô hình thực tế, đánh giá hiệu suất của toàn bộ hệ thống về tốc độ, độ chính xác và độ ổn định.

6. NỘI DUNG CHÍNH

- **Chương 1: Tổng quan đề tài:** Giới thiệu về bối cảnh, lý do chọn đề tài, mục tiêu, đối tượng, phạm vi, ý nghĩa và phương pháp nghiên cứu.
 - **Chương 2: Cơ sở lý thuyết:** Trình bày các kiến thức nền tảng liên quan đến robot công nghiệp, robot Delta, xử lý ảnh và Deep Learning.
 - **Chương 3: Thiết kế và chế tạo hệ thống:** Mô tả chi tiết quá trình thiết kế cơ khí,

thiết kế hệ thống điện và thi công chế tạo mô hình.

- **Chương 4: Lập trình điều khiển:** Nêu quá trình huấn luyện mô hình học sâu, xây dựng giao diện điều khiển, viết code điều khiển robot.
- **Chương 5: Kết quả – đánh giá – hướng phát triển:** Phân tích kết quả thu được, đưa ra đánh giá về hiệu suất của hệ thống và đề xuất các hướng phát triển trong tương lai.

7. Kết quả dự kiến đạt được

- Hệ thống robot Delta dùng xử lý ảnh ứng dụng học sâu để gấp và sắp xếp vật
- Huấn luyện được mô hình học sâu Yolov8 để phân loại sản phẩm lỗi.
- Giao diện điều khiển và giám sát trên PC
- Truyền nhận tín hiệu ổn định giữa PC và STM32

8. Tiến độ thực hiện

TT	Thời gian	Nội dung công việc	Kết quả dự kiến đạt được
1	Tuần 1	Trao đổi, lựa chọn đề tài.	Tìm được đề tài mong muốn thực hiện.
2	Tuần 2	Tiến hành tìm hiểu hướng đi đề tài, phương án thiết kế, phần mềm liên quan để lập trình.	Tìm hiểu được cách làm, hướng đi của đề tài, nguyên lý hoạt động, biết lập trình cơ bản các phần mềm liên quan.
3	Tuần 3	Thiết kế phần cơ khí và điện của hệ thống	Thiết kế 3D được mô hình Robot Delta, thiết kế được mạch điều khiển
4	Tuần 4	Chế tạo và mua các linh kiện cần thiết	Mua đầy đủ linh kiện
5	Tuần 5	Lắp ráp mô hình	Lắp ráp, đấu nối các thành phần như đã thiết kế
6	Tuần 6	Dữ liệu huấn luyện mô hình học sâu	Thu thập đầy đủ dữ liệu và huấn luyện được một mô hình học sâu
7	Tuần 7	Xây dựng giao diện điều khiển và giám sát trên PC	Xây dựng được một giao diện trực quan đầy đủ thông tin
8	Tuần 8-9-10	Xây dựng chương trình xử lý ảnh trên PC, chương trình điều khiển trình điều khiển	Xây dựng được chương trình điều khiển cho hệ thống

		robot trên STM32, chương trình giao tiếp giữa PC và STM32	
9	Tuần 11-14	Thực nghiệm, sửa lỗi, đánh giá, hoàn thiện hệ thống	Hoàn thiện được mô hình chạy ổn định
10	Tuần 15	Viết báo cáo	Hoàn thiện đồ án

Đà Nẵng, ngày.....tháng 9 năm 2025

BỘ MÔN DUYỆT

NGƯỜI HƯỚNG DẪN

SINH VIÊN

a.

LỜI MỞ ĐẦU

Trong sản xuất hiện nay, việc thay thế các khâu thủ công bằng máy móc tự động là rất cần thiết để tăng năng suất và giảm sai sót. Đặc biệt, với những công việc cần sự nhanh nhẹn và chính xác như phân loại sản phẩm, sự kết hợp giữa robot tốc độ cao và trí tuệ nhân tạo đang trở thành giải pháp hiệu quả.

Nhận thấy tầm quan trọng đó, nhóm chúng em thực hiện đề tài: "**Điều khiển và giám sát mô hình Robot Delta sắp xếp sản phẩm ứng dụng học sâu**". Mục tiêu của nhóm là tự thiết kế và lắp ráp viết chương trình điều khiển một hệ thống có thể tự "nhìn" và phân loại sản phẩm, giúp quy trình sản xuất thông minh và chính xác hơn.

Quá trình làm đồ án đã giúp chúng em áp dụng các kiến thức về điện tử, vi điều khiển và lập trình vào thực tế. Việc tự tay thi công mô hình không chỉ giúp nhóm hiểu sâu lý thuyết mà còn rèn luyện kỹ năng giải quyết vấn đề thực tế.

Nhóm xin chân thành cảm ơn Thầy TS. Cao Nguyễn Khoa Nam đã tận tình hướng dẫn. Đồng thời, cảm ơn các Thầy, Cô trong Khoa Điện – Điện tử, Trường Đại học Sư phạm Kỹ thuật – Đại học Đà Nẵng đã tạo điều kiện cho chúng em hoàn thành đồ án này.

Dù đã cố gắng hết sức, nhưng do kinh nghiệm còn hạn chế nên đồ án chắc chắn vẫn còn thiếu sót. Nhóm rất mong nhận được sự góp ý của các Thầy, Cô để đề tài được hoàn thiện hơn.

Xin chân thành cảm ơn!

Người thực hiện đề tài

LỜI CAM ĐOAN

Nhóm xin cam đoan rằng đề tài " Điều khiển và giám sát mô hình robot delta sắp xếp sản phẩm ứng dụng học sâu " được thực hiện dưới sự hướng dẫn của thầy TS. Cao Nguyễn Khoa Nam. Trong quá trình nghiên cứu, nhóm đã chủ động tìm tòi, nghiên cứu, trau dồi và tổng hợp các kiến thức lý thuyết để giải quyết vấn đề đặt ra.

Nhóm xin cam đoan rằng tất cả các kết quả trong đồ án này đều là sản phẩm trung thực của quá trình nghiên cứu, không sao chép từ bất kỳ đồ án nào trước đó. Em hoàn toàn chịu trách nhiệm về nội dung của đồ án này.

Người thực hiện đề tài

MỤC LỤC

NHẬN XÉT CỦA NGƯỜI HƯỚNG DẪN	i
NHẬN XÉT CỦA NGƯỜI PHẢN BIỆN	ii
ĐỀ CƯƠNG ĐỒ ÁN TỐT NGHIỆP	iii
LỜI MỞ ĐẦU	vi
LỜI CAM ĐOAN	vii
MỤC LỤC	viii
DANH MỤC HÌNH ẢNH	xii
DANH MỤC BẢNG	xiii
CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI	1
1.1. Lý do chọn đề tài	1
1.1.1. Xu hướng tự động hóa và Robot công nghiệp trong sản xuất hiện đại.....	1
1.1.2. Nhu cầu phân loại sản phẩm lỗi và ứng dụng của Robot Delta	1
1.1.3. Tiềm năng của việc tích hợp xử lý ảnh với học sâu.....	2
1.2. Mục tiêu đề tài	2
1.3. Đối tượng và phạm vi nghiên cứu.....	2
1.3.1. Đối tượng nghiên cứu.....	2
1.3.2. Phạm vi nghiên cứu	3
1.4. Ý nghĩa khoa học và thực tiễn.....	3
1.4.1. Ý nghĩa khoa học	3
1.4.2. Ý nghĩa thực tiễn.....	3
1.5. Phương pháp nghiên cứu.....	3
1.6. Kết cấu báo cáo.....	4
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	5
2.1. Lý thuyết về Robot Delta	5
2.1.1. Tổng quan.....	5
2.1.2. Ứng dụng thực tế	5
2.1.3. Bài toán động học.....	6
2.2. Lý thuyết xử lý ảnh	13
2.2.1. Khái niệm về ảnh kỹ thuật số	13
2.2.2. Các bước trong xử lý ảnh	14
2.2.3. Ứng dụng trong công nghiệp.....	15
2.3. Lý thuyết về học sâu Deep Learning	15
2.3.1. Tổng quan về học sâu	15
2.3.2. Mạng nơron tích chập CNN.....	15
2.3.3. Các mô hình học sâu phổ biến	16
2.3.4. Ứng dụng học sâu trong xử lý công nghiệp.....	17
CHƯƠNG 3: THIẾT KẾ VÀ THI CÔNG HỆ THỐNG	18

3.1. Thiết kế tổng quan mô hình hệ thống.....	18
3.1.1. Phân tích yêu cầu mô hình	18
3.1.2. Sơ đồ tổng quát mô hình.....	18
3.1.3. Lựa chọn vi điều khiển và giao thức truyền thông	19
3.2. Thiết kế cơ khí Robot Delta.....	23
3.2.1. Thông số yêu cầu làm việc	23
3.2.2. Lựa chọn cấu hình Robot Delta	23
3.2.3. Ràng buộc thiết kế	23
3.2.4. Lựa chọn tỉ lệ.....	23
3.2.5. Tính toán thông số cơ bản	24
3.2.6. Tính toán lựa chọn động cơ.....	26
3.2.7. Thiết kế cơ khí chi tiết Robot Delta	28
3.3. Lựa chọn thiết bị cho mô hình	29
3.3.1. Module Driver TB6600	29
3.3.2. Module Relay.....	31
3.3.3. Đèn báo	33
3.3.4. Công tắc hành trình.....	34
3.3.5. Bơm hút chân không.....	34
3.3.6. Van điện từ.....	35
3.3.7. Nút nhấn	35
3.3.8. Camera Sony IMX258	36
3.4. Sơ đồ đi dây.....	37
3.4.1. Sơ đồ mạch điều khiển.....	37
3.4.2. Sơ đồ mạch động lực	37
3.5. Thiết kế tủ điện.....	38
3.5.1. Mặt trong tủ	38
3.5.2. Mặt ngoài tủ	39
3.6. Thi công mô hình.....	39
3.6.1. Khâu lắp ráp khung và đế Robot.....	39
3.6.2. Khâu lắp ráp cánh tay Robot	40
3.6.3. Khâu lắp đặt bệ di động và đầu gấp	41
3.6.4. Khâu lắp đặt hệ thống băng tải và cảm biến.....	41
3.6.5. Khâu lắp đặt hệ thống camera	41
3.6.6. Tủ điện	42
CHƯƠNG 4: LẬP TRÌNH ĐIỀU KHIỂN	44
4.1. Quy trình công nghệ.....	44
4.1.1. Chế độ thủ công và thiết lập ban đầu.....	44
4.1.2. Chế độ tự động (Auto)	44
4.2. Tổng quan kiến trúc điều khiển	45
4.2.1. Giao thức giao tiếp.....	45
4.2.2. Tổng quát về kiến trúc hệ thống	46
4.3. Xây dựng chương trình điều khiển STM32	48

4.3.1. Cấu trúc tổ chức mã nguồn	48
4.3.2. Phân công đầu ra đầu vào và cấu hình STM32 trên Cube MX	49
4.3.3. Xây dựng chương trình chính	50
4.3.4. Xử lý lệnh từ PC	53
4.3.5. Xử lý ngắt phát xung điều khiển động cơ	55
4.3.6. Quy trình xác định điểm gốc hệ tọa độ	57
4.3.7. Các cơ chế bảo vệ và an toàn hệ thống	59
4.3.8. Điều khiển các thiết bị ngoại vi	59
4.4. Xây dựng chương trình điều khiển trên máy tính	60
4.4.1. Môi trường phát triển và công cụ	60
4.4.2. Cấu trúc tổng thể chương trình	61
4.4.3. Thiết kế giao diện người dùng	62
4.4.4. Mô-đun thị giác máy tính và nhận diện	63
4.4.5. Mô-đun tính toán động học và quy hoạch quỹ đạo	66
4.4.6. Chiến lược điều khiển và Logic tự động	69
4.4.7. Quản lý trạng thái và xử lý lỗi	72
4.4.8. Quy trình hiệu chuẩn hệ thống	73
CHƯƠNG 5: THỬ NGHIỆM ĐÁNH GIÁ KẾT QUẢ	74
5.1. Thiết lập môi trường thực nghiệm	74
5.2. Đánh giá hệ thống nhận diện vật thể	74
5.3. Đánh giá khả năng điều khiển và độ chính xác gấp thê	74
5.4. Đánh giá hiệu suất tổng thể hệ thống	75
HƯỚNG PHÁT TRIỂN ĐỀ TÀI	76
TÀI LIỆU THAM KHẢO	1
PHỤ LỤC 1: lập trình động học	1
Chương trình trên file kinematics.py	1
PHỤ LỤC 2: CHƯƠNG TRÌNH CHÍNH TRÊN STM32	10
Chương trình file main.c	10
PHỤ LỤC 3: Chương trình chính trên máy tính	22
Chương trình file main.py	22

DANH MỤC HÌNH ẢNH

<i>Hình 2.1:Robot Delta</i>	5
<i>Hình 2.2:Robot Delta gấp bánh</i>	6
<i>Hình 2.3:Robot Delta được sử dụng trong các ứng dụng phẫu thuật.....</i>	6
<i>Hình 2.4:Robot Delta được sử dụng như máy in 3D.....</i>	6
<i>Hình 2.5:Ký hiệu các tham số tính toán.....</i>	7
<i>Hình 2.6:Hình chiếu lên mặt phẳng Oxy.....</i>	9
<i>Hình 2.7:Ký hiệu tham số tính toán</i>	11
<i>Hình 2.8:Hình chiếu lên mặt phẳng Oxz.....</i>	11
<i>Hình 2.9:Ký hiệu tính toán</i>	12
<i>Hình 2.10:Giới thiệu về CNN</i>	16
<i>Hình 2.11:Phát hiện đối tượng Yolo với OpenCV.....</i>	17
<i>Hình 3.1:Sơ đồ khối tổng quát hệ thống</i>	18
<i>Hình 3.2: STM32F103C8T6</i>	20
<i>Hình 3.3:Chức năng các chân STM32F103C8T6.....</i>	21
<i>Hình 3.4:Sơ đồ mạch nguyên lý.....</i>	21
<i>Hình 3.5: Ký hiệu tham số tính toán</i>	24
<i>Hình 3.6: Kiểm tra vùng làm việc Robot Delta</i>	26
<i>Hình 3.7: Thông số động cơ NENA 17.....</i>	27
<i>Hình 3.8: Tấm đế cố định</i>	28
<i>Hình 3.9:Giá đỡ động cơ và hộp số</i>	28
<i>Hình 3.10: Tấm đế di động</i>	29
<i>Hình 3.11: Cánh tay chủ động.....</i>	29
<i>Hình 3.12: Module Driver TB6600.....</i>	30
<i>Hình 3.13: Hình ảnh các Module Relay</i>	32
<i>Hình 3.14:Sơ đồ nguyên lý Module Relay.....</i>	32
<i>Hình 3.15: Đèn báo</i>	33
<i>Hình 3.16: Công tắc hành trình</i>	34
<i>Hình 3.17:Bơm hút chân không.....</i>	34
<i>Hình 3.18: Sơ đồ mạch điều khiển</i>	37
<i>Hình 3.19: Sơ đồ mạch động lực</i>	37
<i>Hình 3.20: Bản vẽ mặt trong của tủ</i>	38
<i>Hình 3.21: Bản vẽ mặt ngoài tủ</i>	39
<i>Hình 3.22:Gia cố lắp đặt động cơ.....</i>	40
<i>Hình 3.23:Khớp nối của cánh tay robot.....</i>	40
<i>Hình 3.24:Lắp đặt động servo và giác hút</i>	41
<i>Hình 3.25: Khâu lắp camera.....</i>	42
<i>Hình 3.26: Tủ điện</i>	42
<i>Hình 4.1: Kiến trúc của hệ thống robot Delta</i>	46

Hình 4.2: Lưu đồ thuật toán vòng lặp xử lý chính của STM32.....	51
Hình 4.3: Lưu đồ thuật toán xử lý lệnh từ PC.....	55
Hình 4.4: Lưu đồ thuật toán phát xung điều khiển động cơ.....	57
Hình 4.5: Quy trình về home	58
Hình 4.6: Lưu đồ cơ chế dừng an toàn của hệ thống	59
Hình 4.7: Lưu đồ thuật toán điều khiển ngoại vi	60
Hình 4.8: Giao diện điều khiển trên máy tính	62
Hình 4.9: Lưu đồ xử lý thị giác của hệ thống.....	64
Hình 4.10: Lưu đồ thuật toán quy hoạch quỹ đạo di chuyển.....	67
Hình 4.11: Lưu đồ thuật toán lập lịch gấp và đồng bộ hoá.....	70

DANH MỤC BẢNG

Bảng 3.1: Thông số STM32F103C8T6	20
Bảng 3.2: So sánh giao thức UART và USB CDC	22
Bảng 3.3: Tóm tắt thông số	25
Bảng 3.4: Thông số Module Driver	30
Bảng 3.5: Thông số cài đặt vi bước	30
Bảng 3.6: Thông số cài đặt dòng	31
Bảng 3.7: Thông số kỹ thuật đèn báo	33
Bảng 3.8: Thông số công tắc hành trình	34
Bảng 3.9: Thông số kỹ thuật Pump	35
Bảng 3.10: Thông số nút nhấn	36
Bảng 3.11: Thông số IXM258	36

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

1.1. Lý do chọn đề tài

1.1.1. Xu hướng tự động hóa và Robot công nghiệp trong sản xuất hiện đại

Trong bối cảnh Cuộc cách mạng Công nghiệp 4.0, tự động hóa đã vượt xa các hệ thống truyền thống và trở thành một yếu tố cốt lõi, mang tính đột phá trong sản xuất. Sự hội tụ của các công nghệ tiên tiến như Trí tuệ nhân tạo (AI), Internet vạn vật (IoT), dữ liệu lớn (Big Data) và robot tự hành đã tạo ra một môi trường sản xuất thông minh, linh hoạt và kết nối mạnh mẽ. Các hệ thống tự động hóa này giúp tăng năng suất, nâng cao hiệu quả quy trình sản xuất, cải thiện tính nhất quán và độ chính xác của sản phẩm, đồng thời giảm thiểu sai sót và rủi ro do yếu tố con người.

Các phương pháp tự động hóa truyền thống, vốn chỉ có khả năng phát hiện các vấn đề đơn giản như chiều dài hoặc trọng lượng sản phẩm, không còn đủ sức đáp ứng trước sự bùng nổ của dữ liệu và nhu cầu ngày càng cao về tính linh hoạt, tùy biến trong sản xuất. Điều này đã thúc đẩy một sự chuyển đổi lớn, tạo ra một môi liên kết nhân quả giữa nhu cầu của thị trường và việc ứng dụng các công nghệ mới. Các doanh nghiệp giờ đây cần các hệ thống có khả năng xử lý các vấn đề phức tạp hơn, tự học và thích ứng với các biến thể tự nhiên của sản phẩm. Việc tích hợp AI và Machine Learning, đặc biệt là Deep Learning, vào các quy trình sản xuất trở thành một giải pháp trực diện, nhằm đạt được mục tiêu tiết kiệm chi phí và ổn định chất lượng sản phẩm một cách tối ưu. Đồ án này ra đời với mục đích tạo ra một mô hình thực nghiệm để chứng minh tính hiệu quả của chuỗi giải pháp đó.

1.1.2. Nhu cầu phân loại sản phẩm lỗi và ứng dụng của Robot Delta

Trong các dây chuyền sản xuất công nghiệp, công đoạn kiểm tra chất lượng và phân loại sản phẩm đóng vai trò then chốt, quyết định giá trị và uy tín của doanh nghiệp. Việc giảm thiểu sản phẩm lỗi không chỉ tiết kiệm chi phí mà còn đảm bảo chất lượng đồng đều và nâng cao khả năng cạnh tranh. Các phương pháp kiểm tra thủ công thường gặp phải rủi ro do sự phân tâm, thiếu nhất quán và không thể hoạt động liên tục, dẫn đến sai sót và lãng phí. Robot Delta, còn được biết đến với tên gọi robot nhện, là một giải pháp tự động hóa hiệu quả cho các ứng dụng gấp đặt (pick-and-place) yêu cầu tốc độ và độ chính xác cao. Với cấu trúc song song độc đáo, các động cơ được đặt cố định trên bệ phía trên, giúp giảm khối lượng di chuyển và cho phép robot đạt tốc độ cực nhanh, có thể lên đến 300 sản phẩm mỗi phút. Thiết kế nhỏ gọn này còn giúp tối ưu hóa không gian làm việc, một yếu tố quan trọng trong các nhà máy sản xuất. Do đó, robot Delta là lựa chọn lý tưởng để giải quyết bài toán phân loại sản phẩm tốc độ cao, đặc biệt trong các ngành công nghiệp như thực phẩm, dược phẩm và mỹ phẩm.

1.1.3. Tiềm năng của việc tích hợp xử lý ảnh với học sâu

Mặc dù robot Delta có tốc độ và độ chính xác cơ học vượt trội, khả năng ứng dụng của nó sẽ bị hạn chế nếu không có một hệ thống thị giác thông minh. Các phương pháp xử lý ảnh truyền thống thường yêu cầu lập trình chi tiết cho từng loại lỗi có thể xảy ra, vốn rất tốn thời gian và không thể xử lý hiệu quả các lỗi phức tạp hoặc có nhiều biến thể tự nhiên.

Sự xuất hiện của Deep Learning đã giải quyết triệt để vấn đề này. Là một phân nhánh của Machine Learning, Deep Learning sử dụng các mạng nơ-ron sâu để tự động học và trích xuất các đặc trưng phức tạp từ dữ liệu hình ảnh. Điều này cho phép hệ thống nhận diện và phân loại các lỗi tinh vi, thậm chí những lỗi mà mắt thường không thể nhận ra, một cách nhất quán và đáng tin cậy. Đề tài này không chỉ đơn thuần là việc kết hợp hai công nghệ riêng lẻ, mà còn là sự cộng hưởng sức mạnh, tạo ra một hệ thống vượt trội. Tốc độ và độ chính xác của robot Delta sẽ được phát huy tối đa nhờ khả năng "nhìn" và "suy nghĩ" thông minh, linh hoạt của hệ thống học sâu. Ngược lại, robot tốc độ cao cho phép hệ thống kiểm tra thông minh xử lý một lượng lớn sản phẩm trong thời gian thực, biến nó trở thành một giải pháp toàn diện cho bài toán phân loại sản phẩm công nghiệp.

1.2. Mục tiêu đề tài

Dựa trên bối cảnh và lý do đã trình bày, đề tài đặt ra các mục tiêu cụ thể như sau:

- Thiết kế và chế tạo: Thiết kế và chế tạo thành công một mô hình robot Delta ba bậc tự do có khả năng hoạt động ổn định.
- Phát triển hệ thống thị giác máy: Ứng dụng công nghệ xử lý ảnh kết hợp với Deep Learning, sử dụng mạng nơ-ron tích chập (CNN) như ResNet hoặc YOLO, để nhận diện và phân loại sản phẩm lỗi.
- Lập trình hệ thống điều khiển: Xây dựng chương trình điều khiển tích hợp để robot có thể thực hiện chính xác các thao tác gấp, điều chỉnh hướng của sản phẩm và thả sản phẩm theo kết quả phân loại từ hệ thống thị giác.
- Mô phỏng và thực nghiệm: Tiến hành mô phỏng và thử nghiệm trên mô hình thực tế, đánh giá hiệu suất của toàn bộ hệ thống về tốc độ, độ chính xác và độ ổn định.

1.3. Đối tượng và phạm vi nghiên cứu

1.3.1. Đối tượng nghiên cứu

Đồ án tập trung nghiên cứu và phát triển ba thành phần chính của hệ thống:

- Robot Delta: Cấu trúc cơ khí của robot Delta ba bậc tự do, bao gồm khung, tay máy, và cơ cấu truyền động.

- Hệ thống xử lý ảnh: Quy trình thu nhận, xử lý, và phân tích hình ảnh sản phẩm.
- Mô hình phân loại: Mô hình học sâu (Deep Learning) được huấn luyện để nhận diện và phân loại sản phẩm đạt và sản phẩm không đạt.
- Hệ thống điều khiển: Lập trình và tích hợp các bộ điều khiển (PC control/vi điều khiển) để đồng bộ hóa hoạt động của các thành phần trong hệ thống.

1.3.2. Phạm vi nghiên cứu

- Về cơ khí: Đồ án giới hạn ở việc thiết kế và chế tạo một mô hình robot Delta thu nhỏ.
- Về điều khiển: Nghiên cứu và lập trình các bài toán động học của robot Delta, đặc biệt là động học nghịch, để điều khiển vị trí chính xác của đầu thao tác.
- Về thị giác máy: Tập trung vào việc xây dựng và huấn luyện mô hình Deep Learning sử dụng mạng CNN để phân loại đơn giản, cụ thể là phân biệt sản phẩm đạt và sản phẩm không đạt.

1.4. Ý nghĩa khoa học và thực tiễn

1.4.1. Ý nghĩa khoa học

Để tài thể hiện một sự kết hợp liên ngành mang tính học thuật cao giữa Cơ khí, Tự động hóa, Thị giác máy tính và Trí tuệ nhân tạo. Việc tích hợp các thuật toán phức tạp như Deep Learning vào một hệ thống vật lý cụ thể không chỉ là một ứng dụng kỹ thuật đơn thuần mà còn là một minh chứng cho khả năng tạo ra các "robot thông minh" có thể ra quyết định và phản ứng linh hoạt dựa trên dữ liệu cảm biến. Đồ án góp phần vào việc mở rộng và đào sâu kiến thức trong lĩnh vực robot song song và các giải pháp AI trong tự động hóa.

1.4.2. Ý nghĩa thực tiễn

Đồ án là một ví dụ minh họa rõ nét cho tính khả thi và hiệu quả của việc tự động hóa các công đoạn kiểm tra và phân loại trong sản xuất công nghiệp. Ứng dụng mô hình này vào thực tiễn sẽ góp phần nâng cao năng suất, giảm thiểu sai sót do con người, và cắt giảm đáng kể chi phí vận hành cho các doanh nghiệp. Hệ thống có khả năng hoạt động liên tục, nhất quán và đáng tin cậy, phù hợp với yêu cầu của các dây chuyền sản xuất hiện đại.

1.5. Phương pháp nghiên cứu

Các phương pháp nghiên cứu được áp dụng trong đồ án bao gồm:

- Nghiên cứu tài liệu: Tổng hợp và phân tích các kiến thức nền tảng về robot Delta, xử lý ảnh và các mô hình Deep Learning từ các sách, bài báo khoa học, và các đồ án tốt nghiệp có liên quan.

- Mô hình hóa và thiết kế: Sử dụng phần mềm thiết kế hỗ trợ máy tính (CAD) để tạo mô hình cơ khí và sử dụng các công cụ mô phỏng để tính toán động học và kiểm tra chuyển động của robot.
- Huấn luyện mô hình Deep Learning: Thu thập dữ liệu hình ảnh, xử lý và gắn nhãn, sau đó huấn luyện mô hình CNN để thực hiện bài toán phân loại.
- Chế tạo và tích hợp: Lắp ráp các thành phần cơ khí, điện tử, và lập trình để tích hợp toàn bộ hệ thống thành một mô hình hoạt động hoàn chỉnh.
- Thủ nghiệm và đánh giá: Tiến hành các bài kiểm tra thực tế trên mô hình để đo lường và đánh giá hiệu suất, từ đó đưa ra những nhận xét và cải tiến cần thiết.

1.6. Kết cấu báo cáo

Báo cáo được trình bày gồm năm chương chính, với nội dung được bô cục như sau:

- Chương 1: Tổng quan đề tài: Giới thiệu về bối cảnh, lý do chọn đề tài, mục tiêu, đối tượng, phạm vi, ý nghĩa và phương pháp nghiên cứu.
- Chương 2: Cơ sở lý thuyết: Trình bày các kiến thức nền tảng liên quan đến robot công nghiệp, robot Delta, xử lý ảnh và Deep Learning.
- Chương 3: Thiết kế và chế tạo hệ thống: Mô tả chi tiết quá trình thiết kế cơ khí, thiết kế hệ thống điện và thi công chế tạo mô hình.
- Chương 4: Lập trình điều khiển: Nêu quá trình huấn luyện mô hình học sâu, xây dựng giao diện điều khiển, viết chương trình điều khiển robot.
- Chương 5: Kết quả – đánh giá – hướng phát triển: Phân tích kết quả thu được, đưa ra đánh giá về hiệu suất của hệ thống và đề xuất các hướng phát triển trong tương lai.

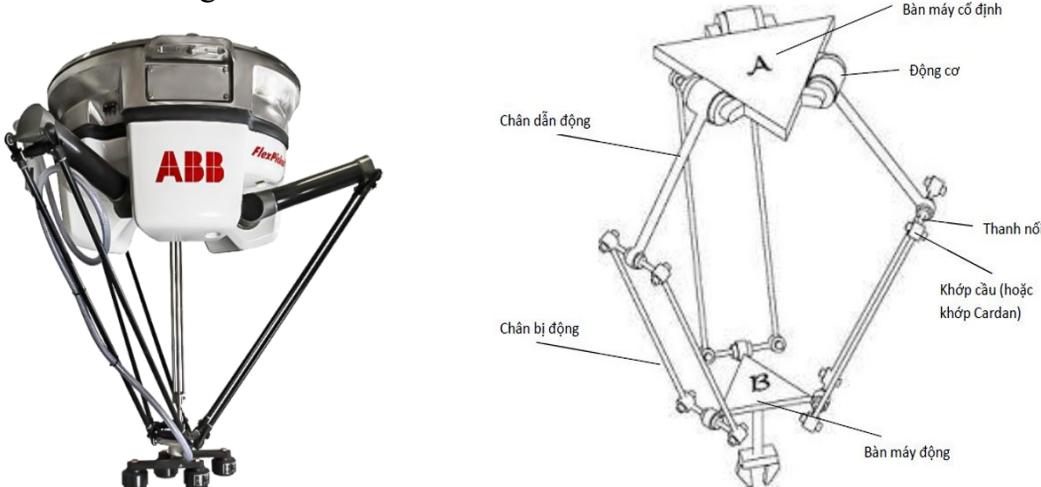
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Lý thuyết về Robot Delta

2.1.1. Tổng quan

Robot Delta được phát minh trong đầu thập niên 80 ở Thụy Sĩ của giáo sư Reymond Clavel và đã được nhận Bằng sáng chế Hoa Kỳ 4.976.582 [1]. Là một loại robot song song có cấu trúc độc đáo, Robot Delta bao gồm một bệ cố định ở trên và một bệ di ở dưới, được kết nối bằng ba (hoặc hơn) cánh tay hình bình hành. Các động cơ điều khiển chuyển động được đặt trên bệ cố định, giúp giảm khối lượng di chuyển và cho phép robot đạt được tốc độ cực nhanh, gia tốc lớn và độ chính xác cao. Nhờ cơ cấu hình bình hành, bệ di động luôn giữ được hướng cố định, chỉ di chuyển tịnh tiến theo các trục không gian X, Y và Z.

Robot Delta của hãng ABB



a, Robot delta của hãng ABB b) Cấu trúc Robot Delta

Hình 2.1: Robot Delta

2.1.2. Ứng dụng thực tế

Với tốc độ và độ chính xác vượt trội, robot Delta đã trở thành một công cụ không thể thiếu trong nhiều ngành công nghiệp. Các ứng dụng tiêu biểu bao gồm:

- Công nghiệp thực phẩm và đồ uống: Gấp và đặt sản phẩm nhẹ như bánh quy, kẹo, snack với tốc độ cao để đóng gói và phân loại.



Hình 2.2:Robot Delta gấp bánh

- Công nghiệp hóa dược, mỹ phẩm và ứng dụng trong y học: Thực hiện các tác vụ gấp đặt, đóng gói sản phẩm trong môi trường sạch, yêu cầu độ chính xác cao, một số được ứng dụng cho việc phẫu thuật.



Hình 2.3:Robot Delta được sử dụng trong các ứng dụng phẫu thuật

- Công nghiệp điện tử: Lắp ráp các linh kiện nhỏ, yêu cầu độ chính xác tuyệt đối, hoặc dùng như một máy in 3D.



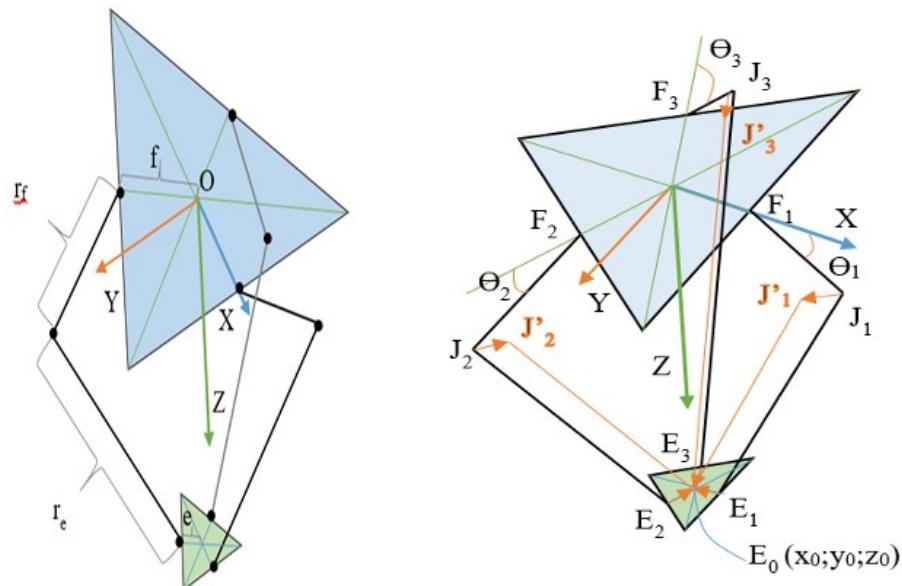
Hình 2.4:Robot Delta được sử dụng như máy in 3D

2.1.3. Bài toán động học

Trước hết, ta xác định một số thông số hình học quan trọng trong thiết kế robot như sau:

- f: là khoảng cách từ điểm O – tâm của bàn máy cố định – đến trực của mỗi động cơ. Trong mô hình này, ta giả định rằng điểm O nằm trên cùng mặt phẳng với trực các động cơ.
- e: là khoảng cách từ điểm E – tâm của bàn máy di động – đến các trực nối của các thanh liên kết trên bàn máy này.
- rf: chiều dài của khâu chủ động, tức là phần truyền động chính từ động cơ.
- re: chiều dài của khâu bị động, hay chính là chiều dài của các thanh trong cấu trúc hình bình hành.

Những thông số trên là các đại lượng vật lý được lựa chọn và xác định trong quá trình thiết kế robot. Hệ tọa độ tham chiếu được đặt tại trọng tâm của khung tam giác cố định, vì vậy tọa độ z của điểm gắn với cơ cấu chấp hành cuối cùng của robot luôn mang giá trị dương, như minh họa bên dưới:



Hình 2.5: Ký hiệu các tham số tính toán

Để chương trình xử lý động học robot được hiệu quả giảm sai số và dễ dàng thiết lập cho vi điều khiển, nên ở đây ta sẽ thiết lập đơn giản động học robot bằng phương pháp hình học [3].

2.1.3.1. Số bậc tự do của Robot Delta

Bậc tự do là số khả năng chuyển động của một cơ cấu (chuyển động quay hoặc tịnh tiến). Để dịch chuyển được một vật thể trong không gian, cơ cấu chấp hành của robot phải đạt được một số bậc tự do.

Thay vì đếm toàn bộ thanh của hình bình hành, ta giản lược mỗi nhánh thành:

- Động cơ quay → 1 khớp R (1-DOF)

- Upper arm → 1 khâu
- Forearm hình bình hành → thay bằng 1 khâu ảo
- Bệ di động (end-effector) → 1 khâu

Robot Delta 3 nhánh sẽ có mô hình tối thiểu:

- $N=8N$ khâu (gồm base + 3 upper arms + 3 virtual links + end-effector)
- $J_1=3J_1$ khớp quay chủ động (R-joint tại động cơ)
- $J_2=6J_2$ khớp universal 2-DOF ảo tại 3 đầu nối bệ di động

Bước 2 – Áp dụng công thức Kutzbach 3D [2]

$$M = 6(N - 1) - 5J_1 - 4J_2 - 3J_3$$

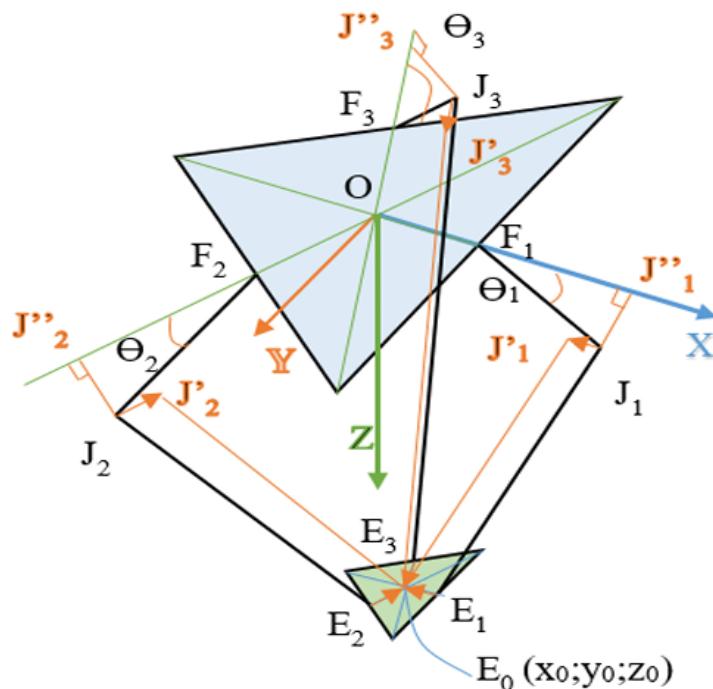
Thé số vào:

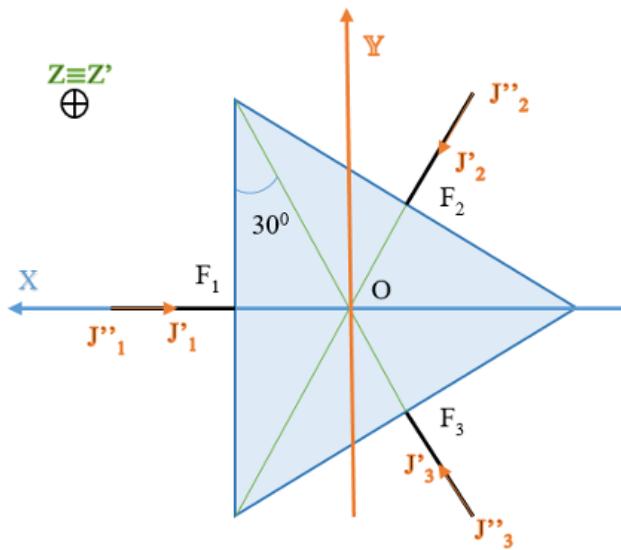
$$M = 6(8 - 1) - 5(3) - 4(6) - 3(0) = 42 - 15 - 24 = 3$$

Vậy ta chứng minh được robot Delta có 3 bậc tự do.

2.1.3.2. Động học thuận

Ví dụ nếu chúng ta biết các góc của khâu chủ động (chúng ta đã tính toán được góc quay hiện tại của động cơ), chúng ta cần xác định vị trí của cơ cấu chấp hành, quá trình xác định như vậy được biết giải động học thuận robot.





Hình 2.6:Hình chiếu lên mặt phẳng Oxy

Bài toán động học thuận là xác định tọa độ của tâm bàn máy di động $E_0(x_0, y_0, z_0)$ khi biết trước ba góc quay của khâu chủ động ($\theta_1, \theta_2, \theta_3$).

Quá trình tính toán được thực hiện bằng cách giải hệ phương trình giao điểm của ba mặt cầu. Tâm của các mặt cầu này là các khớp khuỷu J_i , và bán kính của chúng chính là chiều dài khâu bị động r_e .

Bước 1: Xác định tọa độ các khớp khuỷu J_i (Đã biết)

Dựa trên các tham số thiết kế f, r_f và các góc θ_i (đầu vào), tọa độ của 3 khớp khuỷu J_1, J_2, J_3 được xác định trong hệ tọa độ tổng $OXYZ$ như sau:

$$J_1 = ((f + r_f \cos \theta_1), 0, r_f \sin \theta_1)$$

$$J_2 = ((f + r_f \cos \theta_2) \cos(120^\circ), (f + r_f \cos \theta_2) \sin(120^\circ), r_f \sin \theta_2)$$

$$J_3 = ((f + r_f \cos \theta_3) \cos(240^\circ), (f + r_f \cos \theta_3) \sin(240^\circ), r_f \sin \theta_3)$$

Bước 2: Xác định tọa độ các khớp trên bàn máy động E_i (Ẩn số)

Tọa độ của 3 khớp E_1, E_2, E_3 trên bàn máy di động phụ thuộc vào vị trí của tâm $E_0(x_0, y_0, z_0)$ (ẩn số) và tham số thiết kế e :

$$E_1 = (x_0 + e, y_0, z_0)$$

$$E_2 = (x_0 + e \cos(120^\circ), y_0 + e \sin(120^\circ), z_0)$$

$$E_3 = (x_0 + e \cos(240^\circ), y_0 + e \sin(240^\circ), z_0)$$

Bước 3: Thiết lập hệ phương trình

Ràng buộc vật lý của hệ thống là khoảng cách từ mỗi khớp J_i đến khớp E_i tương ứng

luôn không đổi và bằng r_e .

$$\| J_i - E_i \|^2 = r_e^2$$

Điều này cho chúng ta một hệ 3 phương trình, với 3 ẩn là x_0, y_0, z_0 :

$$\left((f + r_f \cos \theta_1) - (x_0 + e) \right)^2 + (0 - y_0)^2 + \left(r_f \sin \theta_1 - z_0 \right)^2 = r_e^2 \quad (1)$$

$$\left((f + r_f \cos \theta_2) \cos(120^\circ) - (x_0 + e \cos(120^\circ)) \right)^2 + \left((f + r_f \cos \theta_2) \sin(120^\circ) - (y_0 + e \sin(120^\circ)) \right)^2 + \left(r_f \sin \theta_2 - z_0 \right)^2 = r_e^2 \quad (2)$$

$$\left((f + r_f \cos \theta_3) \cos(240^\circ) - (x_0 + e \cos(240^\circ)) \right)^2 + \left((f + r_f \cos \theta_3) \sin(240^\circ) - (y_0 + e \sin(240^\circ)) \right)^2 + \left(r_f \sin \theta_3 - z_0 \right)^2 = r_e^2 \quad (3)$$

Bước 4: Phương pháp giải hệ phương trình

Khi khai triển, cả ba phương trình trên đều có dạng phương trình mặt cầu chung:

$$x_0^2 + y_0^2 + z_0^2 + A_i x_0 + B_i y_0 + C_i z_0 + D_i = 0 \quad (\text{với } i = 1, 2, 3)$$

Trong đó A_i, B_i, C_i, D_i là các hằng số được tính toán từ các tham số đã biết (f, e, r_f, r_e, θ_i).

Để giải hệ 3 phương trình phi tuyến này, ta sử dụng phương pháp trừ vé để loại bỏ các số hạng bậc hai:

- Lấy phương trình (1) trừ phương trình (2).
- Lấy phương trình (1) trừ phương trình (3).

Thao tác này triệt tiêu các số hạng x_0^2, y_0^2, z_0^2 , tạo ra một hệ 2 phương trình tuyến tính (phương trình mặt phẳng) có dạng:

$$(A_1 - A_2)x_0 + (B_1 - B_2)y_0 + (C_1 - C_2)z_0 = (D_2 - D_1)$$

$$(A_1 - A_3)x_0 + (B_1 - B_3)y_0 + (C_1 - C_3)z_0 = (D_3 - D_1)$$

Từ hệ tuyến tính này, ta có thể rút x_0 và y_0 theo z_0 (tương tự như phương pháp tại):

b. $x_0 = a_1 z_0 + b_1$

c. $y_0 = a_2 z_0 + b_2$

Cuối cùng, thế các biểu thức x_0 và y_0 này vào lại một trong ba phương trình mặt cầu ban đầu (ví dụ, phương trình (1)). Ta sẽ thu được một phương trình bậc hai với ẩn duy nhất là z_0 , có dạng tương tự như phương trình :

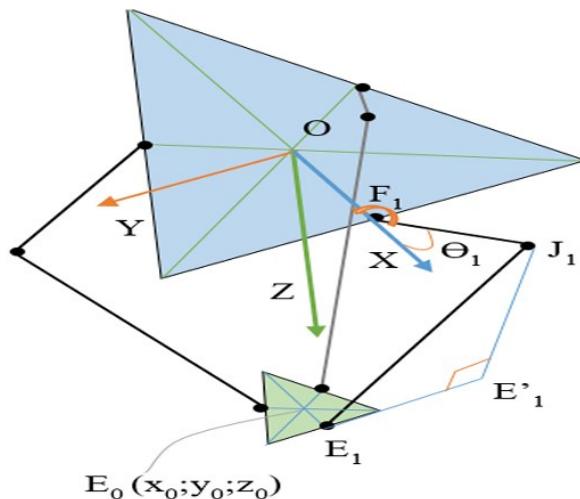
$$Pz_0^2 + Qz_0 + R = 0$$

Giải phương trình bậc hai này, ta tìm được z_0 . Bài toán thường cho hai nghiệm z_0 , ta cần chọn nghiệm phù hợp với vùng làm việc của robot (thường là nghiệm có giá trị âm và nằm trong giới hạn vật lý).

Sau khi có z_0 , ta tính được x_0 và y_0 thông qua các biểu thức $x_0 = a_1 z_0 + b_1$ và $y_0 = a_2 z_0 + b_2$ để tìm ra tọa độ $E_0(x_0, y_0, z_0)$.

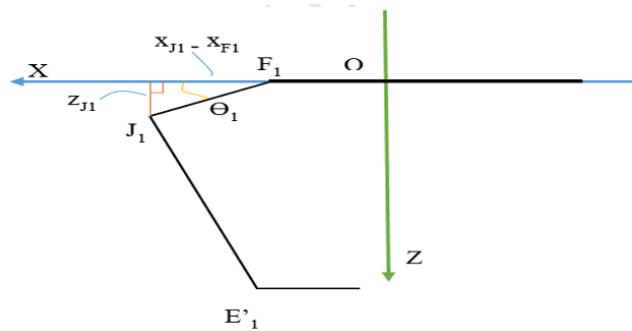
2.1.3.3. Động học nghịch

Khi vật chạy qua camera thì qua quá trình xử lý ảnh chúng ta sẽ có được tọa độ của vật (X, Y, Z), ví dụ chúng ta muốn gấp vật ở điểm có tọa độ (X, Y, Z), chúng ta cần phải xác định góc tương ứng của mỗi tay (góc của khâu chủ động) thiết lập động cơ ở đúng vị trí để chọn. Quá trình xác định như vậy được biết đến như là phép tính ngược hay chính là giải động học ngược robot. Quá trình tính toán như sau : Ta gọi hình chiếu của E_1 lên mặt phẳng XZ là E'_1 .



Hình 2.7: Ký hiệu tham số tính toán

Ta xét mặt phẳng XZ và tính góc theta1:



Hình 2.8: Hình chiếu lên mặt phẳng Oxz

Gọi tọa độ $E_0(x_0, y_0, z_0)$.

$$EE_1 = e$$

$$E_1(x_0 + e, y_0, z_0) \Rightarrow E_1'(x_0 + e, 0, z_0)$$

$$E_1'J_1 = \sqrt{E_1J_1^2 - E_1E_1'^2} = \sqrt{r_e^2 - y_0^2}$$

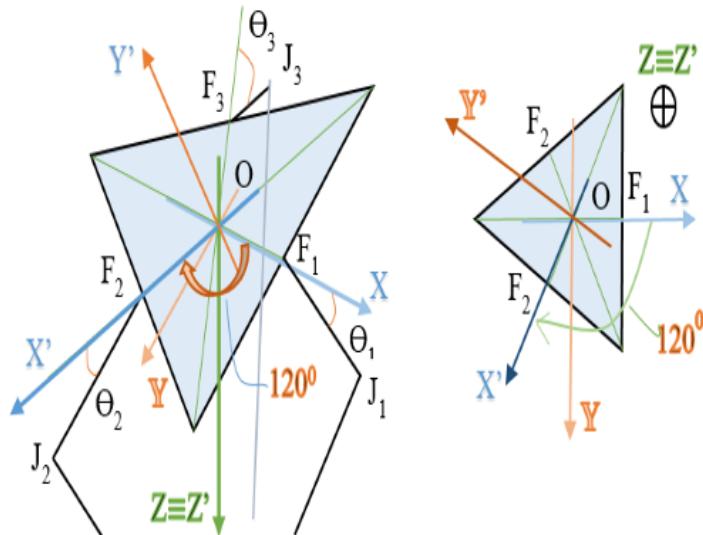
$$F_1 = (f, 0, 0)$$

Ta có hệ sau:

$$\begin{aligned} & \begin{cases} (x_{J_1} - x_{F_1})^2 + (z_{J_1} - z_{F_1})^2 = r_f^2 \\ (x_{J_1} - x_{E'_1})^2 + (z_{J_1} - z_{E'_1})^2 = r_e^2 - y_0^2 \end{cases} \\ & \Rightarrow \begin{cases} (x_{J_1} - f)^2 + z_{J_1}^2 = r_f^2 \\ (x_{J_1} - x_0 - e)^2 + (z_{J_1} - z_0)^2 = r_e^2 - y_0^2 \end{cases} \\ & \Rightarrow J_1(x_{J_1}, 0, z_{J_1}) \Rightarrow \theta_1 = \arctan\left(\frac{z_{J_1}}{x_{J_1} - x_{F_1}}\right) \end{aligned}$$

Tính góc θ_2 .

Để đơn giản chúng ta hãy xoay tọa độ trong mặt phẳng XY quanh trục Z thông qua góc 120 độ ngược chiều kim đồng hồ để trục Y trùng với OF_2 , như thể hiện dưới đây.



Hình 2.9: Ký hiệu tính toán

Ta có tọa độ E_0 trong tọa độ mới Oxy là:

$$E_0' = Rot(z, 120)^{-1}E_0 = \begin{bmatrix} \cos 120 & \sin 120 & 0 & 0 \\ -\sin 120 & \cos 120 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} x_0 \cos 120 + y_0 \sin 120 \\ -x_0 \sin 120 + y_0 \cos 120 \\ z_0 \\ 1 \end{bmatrix}$$

$$\Rightarrow E_{0'} = (x_0 \cos 120 + y_0 \sin 120, -x_0 \sin 120 + y_0 \cos 120, z_0)$$

Tương tự các bước tính θ_1 :

$$\theta_2 = \arctan\left(\frac{z_{J_2}}{x_{J_1} - x_{F_1}}\right)$$

Để tính góc θ_3 tương tự xoay tọa độ trong mặt phẳng XY quanh trục Z thông qua góc 240 độ ngược chiều kim đồng hồ để trục Y trùng với OF_3 .

Sau đó tìm ra $E_{0''} \Rightarrow$ Tương tự các bước tính θ_1 :

$$\theta_3 = \arctan\left(\frac{z_{J_3}}{x_{J_1} - x_{F_1}}\right)$$

\Rightarrow Vậy từ tọa độ $E_0(x_0, y_0, z_0)$ ta tìm ra được 3 góc theta.

Từ đây, ta có thể tạo ra chương trình tính động học robot giúp robot hoạt động.

2.2. Lý thuyết xử lý ảnh

2.2.1. Khái niệm về ảnh kỹ thuật số

Xử lý ảnh kỹ thuật số là một lĩnh vực trong khoa học máy tính và kỹ thuật điện tử, tập trung vào việc sử dụng các thuật toán để thao tác và phân tích hình ảnh được biểu diễn dưới dạng số [4].

Định nghĩa ảnh số: Ảnh số là hình ảnh được biểu diễn bởi một ma trận các pixel (điểm ảnh), trong đó mỗi pixel chứa giá trị số đại diện cho cường độ sáng hoặc màu sắc tại vị trí đó. Ảnh số được tạo ra từ các thiết bị thu nhận như camera kỹ thuật số, nơi ánh sáng liên tục được chuyển đổi thành dữ liệu rời rạc thông qua quá trình lấy mẫu và lượng tử hóa.

Các loại ảnh:

- Ảnh xám (grayscale): Chỉ chứa thông tin về cường độ sáng, thường biểu diễn bằng giá trị từ 0 (đen) đến 255 (trắng) cho mỗi pixel.
- Ảnh màu: Bao gồm nhiều kênh màu, thường là RGB (Red, Green, Blue), nơi mỗi kênh là một ma trận grayscale đại diện cho cường độ của một màu cơ bản.
- Ảnh nhị phân (binary): Chỉ có hai giá trị (0 hoặc 1, đen hoặc trắng), thường

dùng cho phân đoạn hoặc nhận dạng đối tượng đơn giản.

Mục tiêu và ứng dụng của xử lý ảnh: Mục tiêu chính là cải thiện chất lượng ảnh, trích xuất thông tin hữu ích, và hỗ trợ ra quyết định. Ứng dụng rộng rãi trong y tế (chẩn đoán hình ảnh), an ninh (nhận dạng khuôn mặt), nông nghiệp (phát hiện bệnh cây trồng), và công nghiệp (kiểm tra chất lượng sản phẩm).

2.2.2. Các bước trong xử lý ảnh

Quá trình xử lý ảnh thường bao gồm các giai đoạn chính để chuyển đổi ảnh thô thành thông tin có ý nghĩa.

Thu nhận ảnh: Đây là bước đầu tiên, sử dụng cảm biến quang học như CCD hoặc CMOS trong camera để chuyển đổi ánh sáng thành tín hiệu điện, sau đó số hóa thành định dạng ảnh như JPEG, PNG, hoặc TIFF. Định dạng ảnh ảnh hưởng đến chất lượng và kích thước file, ví dụ JPEG nén có mất mát, phù hợp cho lưu trữ nhưng có thể làm giảm chi tiết.

Tiền xử lý ảnh: Nhằm cải thiện chất lượng ảnh trước khi phân tích sâu hơn.

- Lọc nhiễu: Sử dụng bộ lọc Gaussian để làm mờ nhiễu ngẫu nhiên bằng cách tính trung bình có trọng số, hoặc Median để loại bỏ nhiễu muối tiêu bằng cách thay thế pixel bằng median của vùng lân cận.
- Cân bằng sáng và tăng cường độ tương phản: Sử dụng histogram equalization để phân bố đều cường độ, giúp tăng độ tương phản ở các vùng tối hoặc sáng quá mức.

Phân đoạn ảnh: Tách biệt các đối tượng khỏi nền.

- Phân ngưỡng (thresholding): Chuyển ảnh xám thành nhị phân bằng cách đặt ngưỡng, ví dụ Otsu's method tự động chọn ngưỡng tối ưu.
- Phát hiện biên: Sử dụng toán tử Sobel để tính gradient theo hướng ngang và dọc, hoặc Canny để phát hiện biên chính xác hơn với các bước làm mịn, tìm gradient, non-maximum suppression và hysteresis thresholding.
- Phân tích contour: Sau phát hiện biên, sử dụng thuật toán như findContours trong OpenCV để xác định đường viền đối tượng, hỗ trợ đo lường kích thước hoặc hình dạng.
- Trích chọn đặc trưng: Rút trích các thuộc tính đặc trưng để phân loại hoặc nhận dạng.
- Hình dạng (shape): Như diện tích, chu vi, eccentricity.
- Màu sắc: Histogram màu hoặc trung bình RGB.

- Kết cấu (texture): Phân tích sự biến đổi cường độ, ví dụ GLCM (Gray-Level Co-occurrence Matrix).
- Các phương pháp nâng cao: Histogram of Oriented Gradients (HOG) để mô tả hình dạng dựa trên gradient; Scale-Invariant Feature Transform (SIFT) và Speeded Up Robust Features (SURF) để trích xuất keypoint bất biến với tỷ lệ và xoay.

2.2.3. Ứng dụng trong công nghiệp

Xử lý ảnh được áp dụng rộng rãi để tự động hóa quy trình sản xuất.

- Nhận dạng đối tượng: Sử dụng các kỹ thuật trên để xác định vị trí và loại đối tượng trong ảnh.
- Phân loại sản phẩm trên băng tải: Phân tích hình ảnh thời gian thực để sắp xếp sản phẩm dựa trên kích thước, màu sắc hoặc hình dạng.
- Kiểm tra lỗi sản phẩm: Phát hiện khuyết điểm như vết nứt, sai kích thước qua so sánh với mẫu chuẩn.

2.3. Lý thuyết về học sâu Deep Learning

2.3.1. Tổng quan về học sâu

Học sâu là một nhánh của học máy, sử dụng mạng nơ-ron nhân tạo với nhiều lớp để học biểu diễn dữ liệu phức tạp [5].

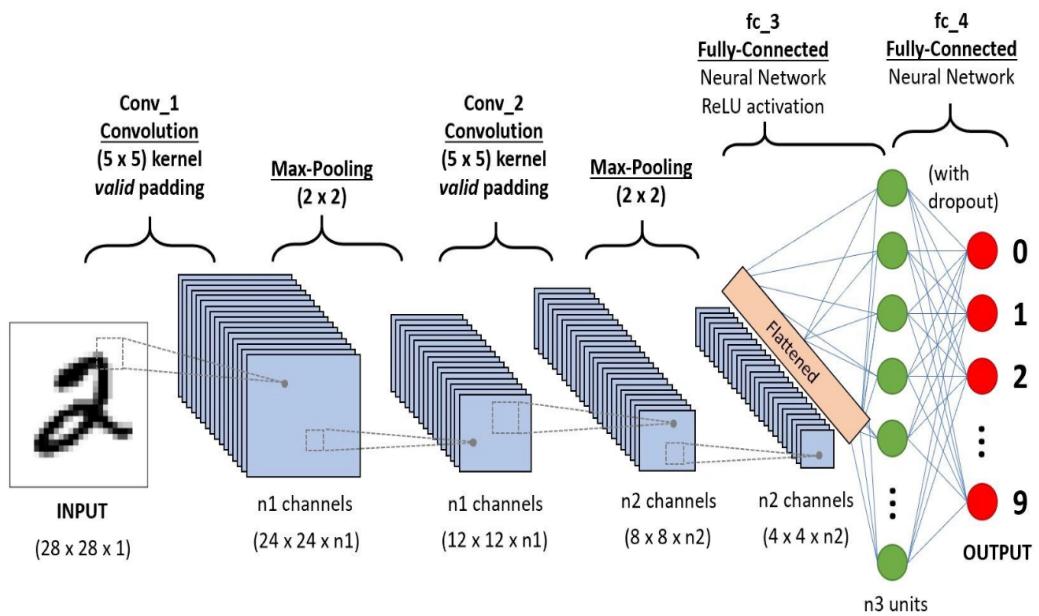
Khái niệm học sâu và sự khác biệt với học máy truyền thống: Học sâu dựa trên mạng nơ-ron sâu để tự động trích xuất đặc trưng từ dữ liệu thô, khác với học máy truyền thống yêu cầu trích xuất đặc trưng thủ công (như HOG). Học sâu hiệu quả hơn với dữ liệu lớn, nhưng đòi hỏi tài nguyên tính toán cao.

Lịch sử phát triển và xu hướng ứng dụng: Bắt đầu từ perceptron năm 1950, phát triển mạnh từ 2010 nhờ GPU và dữ liệu lớn (ImageNet). Xu hướng hiện nay bao gồm học chuyển giao, học không giám sát, và ứng dụng trong tự lái xe, y tế, và công nghiệp 4.0.

2.3.2. Mạng nơron tích chập CNN

CNN là mô hình học sâu chuyên cho dữ liệu lưới như ảnh, tận dụng cấu trúc không gian.

Cấu trúc mạng CNN: Bao gồm lớp tích chập để trích đặc trưng bằng kernel; lớp pooling để giảm kích thước; và lớp fully connected để phân loại.



Hình 2.10: Giới thiệu về CNN

Cách hoạt động: Trích đặc trưng qua tích chập và pooling; lan truyền tiến tính output; lan truyền ngược cập nhật trọng số bằng gradient descent để giảm lỗi.

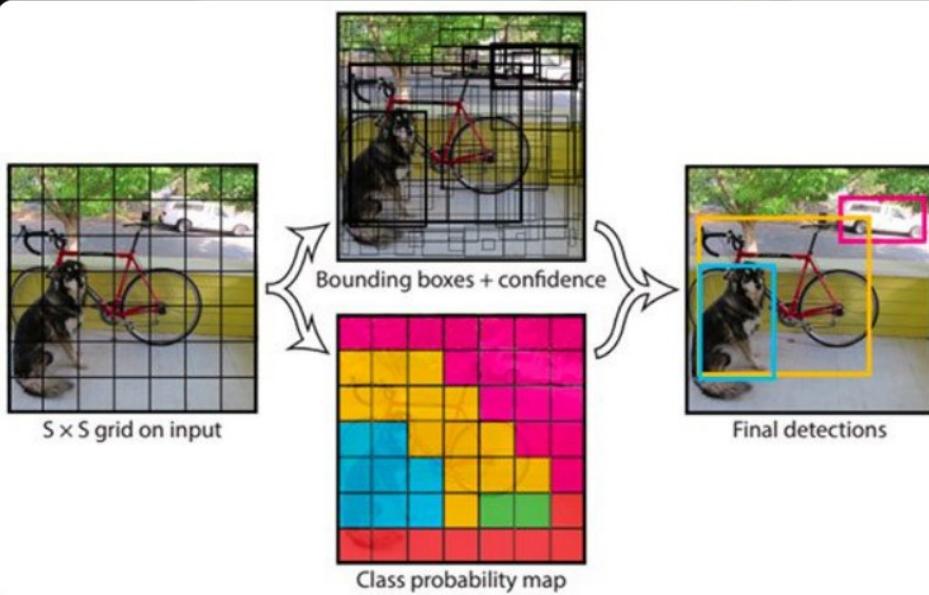
Các hàm kích hoạt: ReLU (Rectified Linear Unit) để giới thiệu phi tuyến tính nhanh chóng; Sigmoid cho output nhị phân; Softmax cho phân loại đa lớp bằng cách chuyển thành xác suất.

2.3.3. Các mô hình học sâu phổ biến

YOLO (You Only Look Once): Là mô hình phát hiện đối tượng thời gian thực [6].

Cách hoạt động: Chia ảnh thành lưới $S \times S$, mỗi ô dự đoán bounding box, độ tin cậy và class. Sử dụng non-max suppression để loại bỏ trùng lặp.

Ưu điểm: Tốc độ nhanh (hàng trăm FPS), phù hợp ứng dụng thời gian thực như giám sát.



Hình 2.11: Phát hiện đối tượng Yolo với OpenCV

SSD (Single Shot Detector): Tương tự YOLO nhưng cải tiến.

- So sánh với YOLO: SSD sử dụng nhiều lớp đặc trưng để phát hiện đối tượng ở các tỷ lệ khác nhau, chính xác hơn với đối tượng nhỏ, nhưng có thể chậm hơn một chút.
- Cách xử lý đa tỷ lệ đối tượng: Áp dụng default boxes ở các lớp khác nhau, dự đoán offset và class cho từng box.

2.3.4. Ứng dụng học sâu trong xử lý công nghiệp

Học sâu nâng cao hiệu quả xử lý ảnh bằng cách học từ dữ liệu lớn.

- Phát hiện và phân loại đối tượng: Sử dụng CNN như YOLO để nhận dạng sản phẩm chính xác cao.
- Gấp vật trên băng tải bằng robot: Tích hợp với robot delta, sử dụng output bounding box để tính tọa độ gấp.
- Tích hợp với hệ thống điều khiển tự động: Kết nối với PLC hoặc IoT để tự động hóa toàn bộ dây chuyền, giảm lỗi con người và tăng năng suất.

CHƯƠNG 3: THIẾT KẾ VÀ THI CÔNG HỆ THỐNG

3.1. Thiết kế tổng quan mô hình hệ thống

3.1.1. Phân tích yêu cầu mô hình

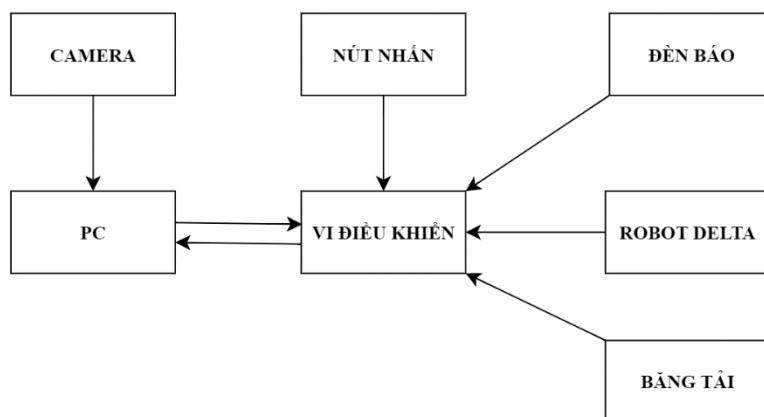
Thiết kế hệ thống robot Delta xử lý ảnh trong phạm vi đề tài này tập trung vào việc chế tạo một mô hình thu nhỏ để chứng minh nguyên lý hoạt động của hệ thống thị giác và điều khiển tốc độ cao. Các quyết định thiết kế cơ khí, lựa chọn vật tư và đấu nối điện được thực hiện nhằm tối ưu hóa tính kinh tế, dễ lắp ráp, đồng thời đáp ứng các tiêu chí về độ chính xác và tốc độ cần thiết cho việc gấp và đặt mô hình vật phẩm nhỏ.

Mục tiêu chính là thiết kế một mô hình đáp ứng các tiêu chí sau:

- Vùng làm việc khả thi: Vùng làm việc đủ rộng để che phủ khu vực băng tải xử lý ảnh và khu vực thả sản phẩm, yêu cầu robot phải bao phủ được một vùng hình trụ D = 300mm, Z=100mm.
- Tải trọng: Robot có thể gấp được vật nặng Pmax = 0.1 kg.
- Có tích hợp cơ cấu xoay để điều chỉnh hướng của vật được gấp.
- Độ chính xác: Đạt độ lặp lại vị trí chấp nhận được ($\pm 2mm$) để gấp các mô hình vật phẩm nhẹ.
- Tốc độ: Robot phải hoàn thành 20 lần gấp thả trong 1 phút
- Tính kinh tế: Ưu tiên sử dụng các vật tư sẵn có, chi phí thấp, phù hợp với mô hình robot giáo dục hoặc nghiên cứu cơ bản.

3.1.2. Sơ đồ tổng quát mô hình

Ta sẽ đi vào thiết kế hệ thống có sơ đồ khái quát như sau:



Hình 3.1:Sơ đồ khái quát hệ thống

Chức năng của các thành phần trong hệ thống như sau:

- Khối Camera: đóng vai trò là thiết bị thu nhận hình ảnh sản phẩm thực tế trên băng tải. Camera được lắp cố định phía trên vùng làm việc để chụp ảnh và gửi dữ liệu thô về máy tính xử lý.
- Khối PC: đóng vai trò là trung tâm xử lý dữ liệu cao cấp, chạy các thuật toán học sâu để nhận diện và phân loại sản phẩm. Sau khi tính toán được tọa độ (x,y,z) vật thể, PC sẽ gửi kết quả này xuống vi điều khiển qua cổng truyền thông UART.
- Khối vi điều khiển: là bộ điều khiển trung tâm chịu trách nhiệm tiếp nhận lệnh từ PC và điều phối hoạt động của các thiết bị ngoại vi. Vi điều khiển sẽ xuất tín hiệu để điều khiển Robot Delta, băng tải, đèn báo và tiếp nhận phản hồi từ các nút nhấn để quản lý toàn bộ hệ thống.
- Khối nút nhấn: là thiết bị để người vận hành tương tác trực tiếp với mô hình. Khi nhấn nút, tín hiệu sẽ được gửi đến vi điều khiển để thực hiện các chức năng như khởi động hoặc dừng hệ thống theo ý muốn.
- Khối đèn báo: có chức năng hiển thị trạng thái làm việc của hệ thống thông qua tín hiệu đèn. Đèn báo giúp người dùng dễ dàng nhận biết máy đang hoạt động bình thường hay đang gặp sự cố.
- Khối Robot delta: là cơ cấu chấp hành chính thực hiện việc phân loại sản phẩm. Dựa vào tọa độ được vi điều khiển cung cấp, robot sẽ di chuyển để gấp, xoay vật theo đúng hướng và đặt vào vị trí đã chỉ định.
- Băng tải: có nhiệm vụ vận chuyển sản phẩm đi qua khu vực thu nhận ảnh của camera và vùng gấp của robot. Tốc độ của băng tải được vi điều khiển điều chỉnh sao cho khớp với nhịp làm việc của hệ thống xử lý ảnh và cánh tay robot.

3.1.3. Lựa chọn vi điều khiển và giao thức truyền thông

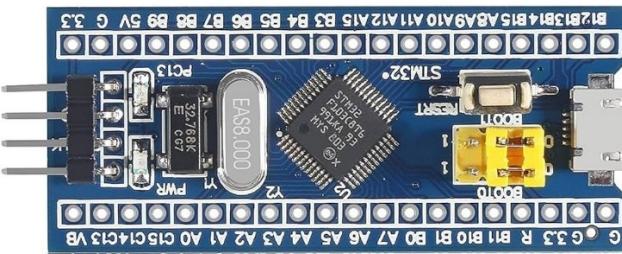
Trong thiết kế điều khiển của hệ thống, nhóm phân chia nhiệm vụ rất rõ ràng để tận dụng tối đa thế mạnh của từng thiết bị: Máy tính (PC) đóng vai trò là "trung tâm trí tuệ" thực hiện các tác vụ nặng về xử lý ảnh và tính toán hình học, còn vi điều khiển đóng vai trò là "cơ cấu thực thi" để điều khiển trực tiếp các động cơ.

3.1.3.1. Lựa chọn vi điều khiển STM32F103C8T6 [8]

Dòng vi điều khiển **STM32F103C8T6** được nhóm tin dùng để làm bộ xử lý tại chỗ cho robot. Dù phần tính toán động học nghịch phức tạp đã được máy tính xử lý xong và gửi xuống, STM32 vẫn đóng vai trò cực kỳ quan trọng nhờ các lý do sau:

- **Tốc độ thực thi lệnh nhanh chóng:** Với xung nhịp 72MHz, STM32 đảm bảo việc tiếp nhận dữ liệu từ máy tính, giải mã các chuỗi lệnh Serial và xuất tín hiệu điều khiển các trục motor diễn ra tức thời, không gây ra độ trễ giữa "lệnh" và "thực thi".

- Phát xung điều khiển motor ổn định:** Robot Delta đòi hỏi sự đồng bộ rất cao giữa 3 cánh tay. STM32 sở hữu các bộ Timer mạnh mẽ giúp tạo ra các chuỗi xung điều khiển cho Driver TB6600 một cách đều đặn và mịn màng, giúp robot di chuyển êm ái, chính xác đến vị trí vật thể mà không bị rung lắc.
- Khả năng quản lý thiết bị ngoại vi:** Vì điều khiển này có đủ số lượng chân I/O để nhóm kết nối đồng thời: 3 trục động cơ bước, hệ thống công tắc hành trình để lấy gốc (Homing), điều khiển van hút chân không và các nút nhấn điều khiển trực tiếp trên mô hình.
- Giá thành và độ phổ biến:** STM32 có chi phí rất rẻ, kích thước nhỏ gọn nhưng hiệu năng làm việc rất bền bỉ trong môi trường thực nghiệm, dễ dàng lắp đặt vào tủ điện điều khiển của robot.



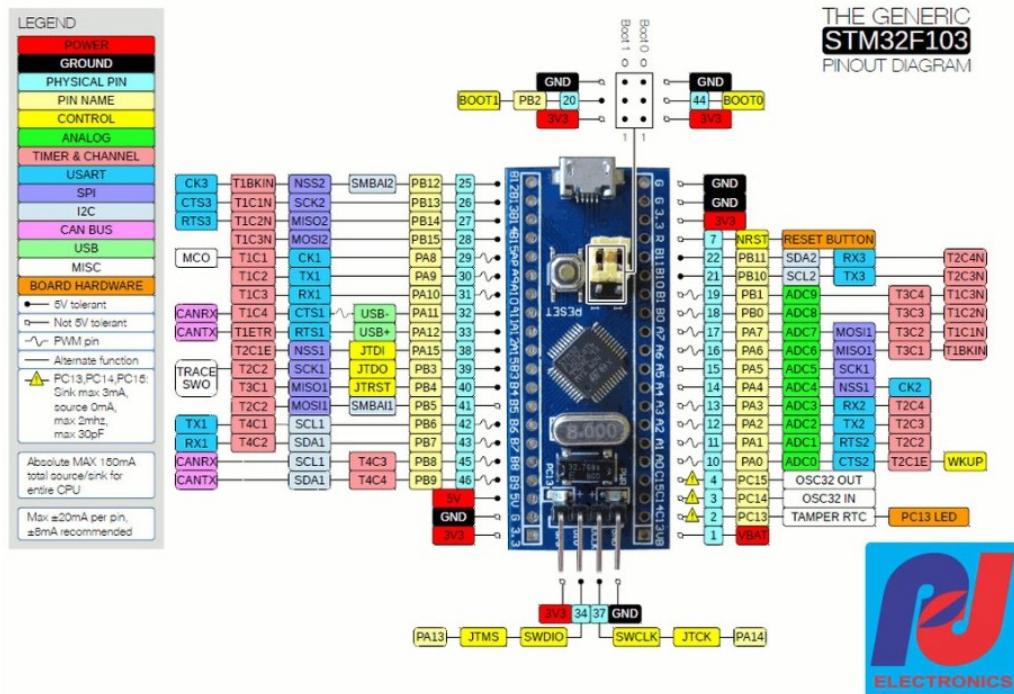
Hình 3.2: STM32F103C8T6

- Thông số kỹ thuật:

Bảng 3-1: Thông số STM32F103C8T6

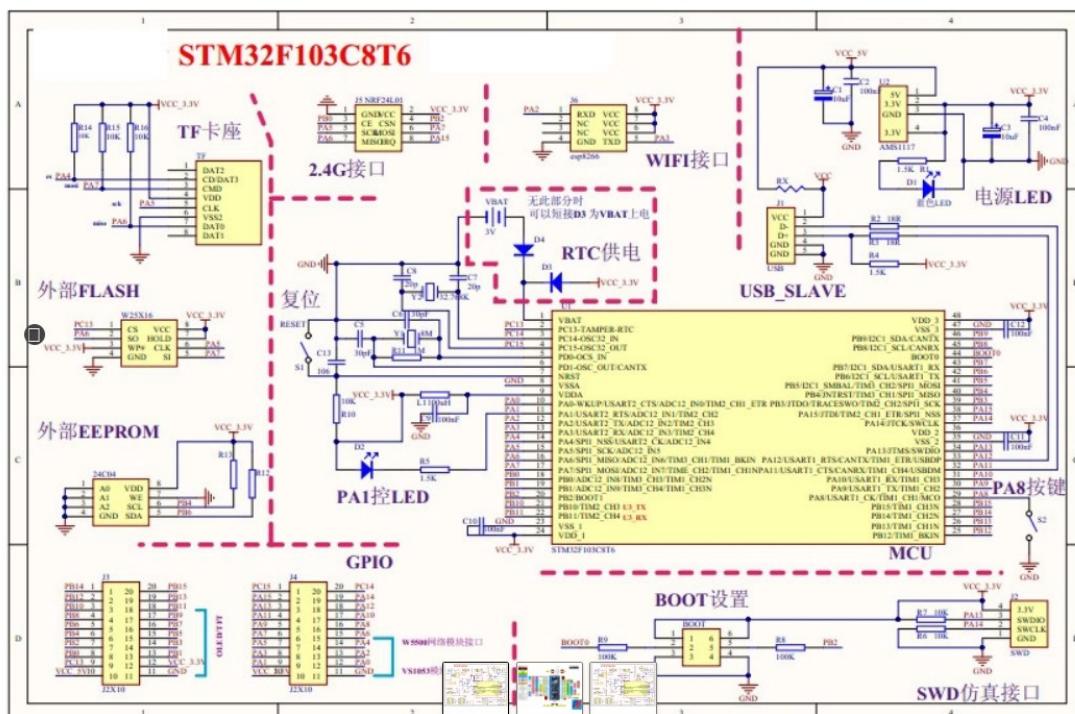
STT	Thuộc tính	Thông số
1	Nguồn	2.7V -5.5V
2	Input/Output	32
3	Số chân	40
4	Core	ARM 32Cortex-M3 CPU
5	Dòng hoạt động	1.1mA
6	Dòng output	20mA
7	Các chuẩn giao tiếp	I2C,SPI,USART/USART,USB...
8	Bộ nhớ ROM	128KByte
9	Bộ nhớ SRAM	20KByte
10	Nhiệt độ hoạt động	-40°C ~ 85°C
11	Cổng ADC	12Bit - Tự thiết lập
12	Timer	8Bit hoặc 16Bit
13	PWM	2 Kênh

- Chức năng các chân:



Hình 3.3: Chức năng các chân STM32F103C8T6

- Sơ đồ nguyên lý:



3.4:Sơ đồ mạch nguyên lý

3.1.3.2. Lựa chọn giao thức truyền thông

Trong hệ thống này, máy tính (PC) đảm nhận toàn bộ phần tính toán nặng từ xử lý ảnh đến giải động học nghịch, sau đó gửi tọa độ trực tiếp xuống cho robot thực thi. Để

thực hiện việc này qua cổng USB, nhóm đã tiến hành so sánh giữa hai phương thức truyền thông phổ biến trên dòng STM32 Blue Pill là UART (qua mạch chuyển đổi) và USB CDC (Communication Device Class).

a) So sánh UART và USB CDC

Bảng 3-2:So sánh giao thức UART và USB CDC

Đặc điểm	UART	USB CDC
Kết nối	Cần thêm mạch chuyển đổi USB-to-UART	Kết nối trực tiếp từ cổng USB máy tính vào cổng Micro-USB có sẵn trên board Blue Pill.
Tốc độ	Bị giới hạn bởi các mức Baudrate cố định(như 9600, 115200 bps).	Tốc độ truyền dữ liệu rất cao nhờ tận dụng băng thông của chuẩn USB Full Speed (12 Mbps).
Độ phức tạp	Rất đơn giản, dễ lập trình và gỡ lỗi thủ công.	Phải cấu hình thư viện USB phức tạp trong STM32CubeIDE.
Tính ổn định	Rất cao, ít bị xung đột driver trên máy tính.	Đôi khi gặp lỗi driver hoặc mất kết nối khi khởi động lại.

b) Lựa chọn giao thức USB CDC

Sau khi so sánh, nhóm quyết định lựa chọn giao thức **USB CDC** để truyền thông vì các lý do thực tế sau:

- Tiết kiệm linh kiện: Nhóm không cần phải lắp thêm mạch chuyển đổi USB-to-UART rời, giúp tủ điện điều khiển robot Delta trông chuyên nghiệp và gọn gàng hơn rất nhiều.
- Tốc độ nhận lệnh tức thời: Vì toàn bộ việc tính toán động học nghịch đã được thực hiện trên PC, STM32 chỉ cần nhận kết quả cuối cùng. Tốc độ cao của USB CDC giúp robot nhận lệnh ngay lập tức khi máy tính xử lý xong ảnh, đảm bảo không bị trễ nhịp gấp sán pheam.
- Cơ chế phản hồi nhịp nhàng: Nhóm sử dụng cơ chế xác nhận lệnh (Handshake). Khi STM32 điều khiển robot gấp xong và gửi tín hiệu phản hồi DONE về PC, đường truyền USB CDC đảm bảo tín hiệu này được máy tính nhận diện ngay để chuẩn bị cho vật thể tiếp theo trên băng tải.

Việc dùng cổng USB trực tiếp trên STM32 Blue Pill là lựa chọn tối ưu nhất. Nó vừa giúp giảm bớt dây nhợ rắc rối, vừa đảm bảo tốc độ truyền tọa độ cực nhanh để robot Delta đạt được năng suất gấp 20 vật/phút như yêu cầu đề tài.

3.2. Thiết kế cơ khí Robot Delta

3.2.1. Thông số yêu cầu làm việc

Dựa trên yêu cầu ứng dụng của robot trong công việc gấp sản phẩm, robot Delta cần đạt được các yêu cầu sau:

Hình dạng vùng làm việc:

- Hình trụ đường kính vùng làm việc: $D = 300 \text{ mm}$
- Chiều cao vùng làm việc: $H = 100 \text{ mm}$
- Tải trọng tối đa: $P = 0.1 \text{ kg}$
- Tốc độ di chuyển tối đa: $v_{\max} = 3 \text{ m/s}$
- Gia tốc tối đa: $a_{\max} = 10 \text{ m/s}^2$

3.2.2. Lựa chọn cấu hình Robot Delta

Lựa chọn cấu hình Robot Delta song song 3-DOF cổ điển. Cấu hình này tối ưu về tỉ lệ công suất/trọng lượng vì tất cả các động cơ đều được đặt trên bệ tĩnh. Điều này giúp giảm thiểu quán tính của các bộ phận chuyển động, từ đó cho phép mô hình đạt được tốc độ và gia tốc cao hơn trong phạm vi tải trọng thấp.

3.2.3. Ràng buộc thiết kế

Các ràng buộc kỹ thuật cần đảm bảo trong quá trình thiết kế:

- Góc quay khớp động cơ: $\theta \in [-45^\circ, 70^\circ]$
- Tránh vùng singularity: $\det(J) \neq 0$
- Hệ số an toàn: $SF = 1.5$

3.2.4. Lựa chọn tỉ lệ

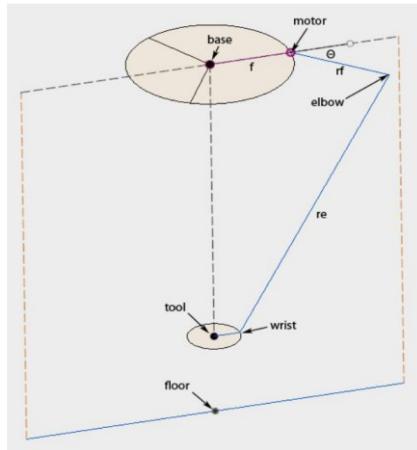
Một robot Delta điển hình có 3 chuỗi động học giống hệt nhau, mỗi chuỗi bao gồm:

1. **Khâu chủ động (Active Arm/Upper Arm):** Có chiều dài L_1 , quay quanh khớp tại đế.
2. **Khâu bị động (Forearm/Lower Arm):** Có chiều dài L_2 , thường là cơ cấu hình bình hành, nối khâu chủ động với bàn di động (End-Effector).

Các thông số thiết kế chính:

- f : Bán kính của đế cố định (Base Platform).
- e : Bán kính của bàn di động (Moving Platform).
- rf : Chiều dài khâu chủ động.
- re : Chiều dài khâu bị động.
- $\theta_{\min}, \theta_{\max}$: Giới hạn góc quay của khớp chủ động.

Để đơn giản hóa bài toán, chúng ta thường chọn trước một số thông số dựa trên kinh nghiệm và ràng buộc thực tế.



Hình 3.5: Ký hiệu tham số tính toán

Tỉ lệ giữa rf và re chọn trong khoảng 1.5-2.5:

- Nếu $rf / re < 1.5$: Vùng làm việc bị giới hạn, góc θ thay đổi quá lớn
- Nếu $rf / re > 2.5$: Độ cứng vững giảm, độ chính xác kém
- Chọn $rf / re = 2.0$: Cân bằng tối ưu

Tỷ lệ này đảm bảo cân bằng giữa độ cứng vững và vùng làm việc. Giá trị $\lambda_1 = 2.0$ cho phép robot có độ linh hoạt cao trong khi vẫn duy trì độ chính xác tốt.

Tỉ lệ giữa f và w chọn trong khoảng 1.5-3.0:

- Nếu $f / e < 1.5$: Platform quá lớn, dễ xảy ra can thiệp cơ học
- Nếu $f / e > 3.0$: Vùng làm việc bị thu hẹp ở tâm
- Chọn $f / e = 2.0$: Tối ưu vùng làm việc và tránh can thiệp

Tỷ lệ này giúp tối ưu hóa không gian làm việc và giảm thiểu vùng điểm kì dị. Ngoài ra đây là mô hình thu nhỏ, nếu tỉ lệ này quá lớn sẽ khiến cho bệ di chuyển quá nhanh gây ra khó khăn khi thiết kế cơ cấu xoay và hút.

3.2.5. Tính toán thông số cơ bản

3.2.5.1. Xác định bán kính bàn di động (e) và đế cố định (f)

Để xác định bán kính bàn di động w , ta xuất phát từ yêu cầu đường kính vùng làm việc $D = 300$ mm. Bán kính bàn di động cần được chọn sao cho các điểm biên ngoài cùng của vùng làm việc vẫn nằm trong tầm với của robot.

Từ kinh nghiệm thiết kế các robot Delta công nghiệp, bán kính bàn di động thường được chọn theo công thức:

$$e = \frac{D}{k}$$

Trong đó k là hệ số điều chỉnh, thường chọn trong khoảng $4 \div 8$. Với yêu cầu vùng làm việc không quá lớn và tải trọng nhẹ, ta chọn:

$$w = \frac{300}{7} = 42.8 \text{ mm}$$

Chọn: w = 40 mm

Từ tỷ lệ thiết kế đã chọn $f/e = 1.5$, ta có:

$$f = 1.5 \times e = 1.5 \times 40 = 60 \text{ mm}$$

Chọn: f = 60 mm

Tỷ lệ này đảm bảo:

- Khoảng cách $f - w = 20 \text{ mm}$ đủ lớn để tránh can thiệp giữa các cánh tay
- Bàn di động $w = 40 \text{ mm}$ đủ lớn để thiết kế cơ cấu hút chân không
- Để cố định $f = 60 \text{ mm}$ phù hợp với kích thước động cơ step nhỏ gọn

3.2.5.2. Tính toán độ dài cánh tay chủ động rf và bị động re

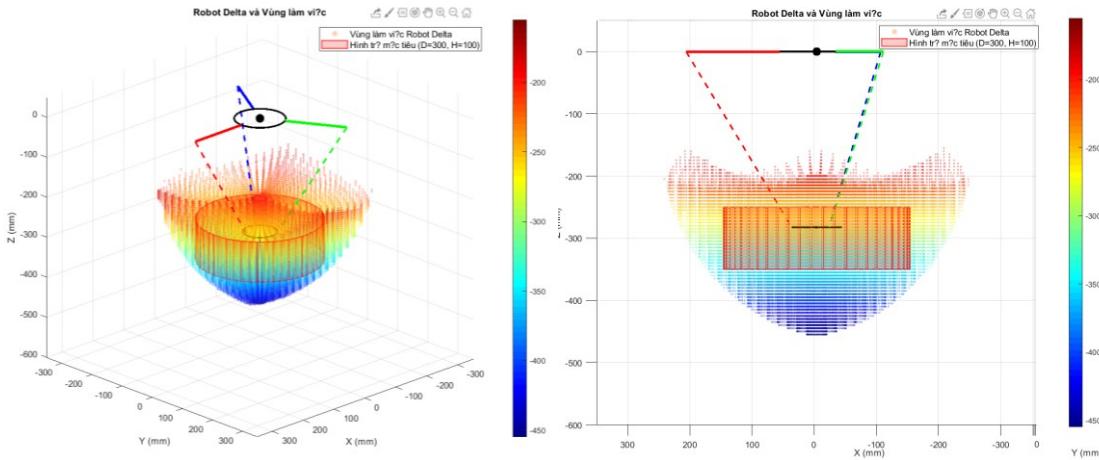
Lựa chọn $rf = 150 \text{ mm}$, với tỉ lệ $re / rf = 2.0$, chọn $re = 330 \text{ mm}$

Bảng 3-3: Tóm tắt thông số

Thông số	Ký hiệu	Giá trị	Đơn vị
Bán kính đế cố định	f	60	mm
Bán kính bàn di động	w	40	mm
Chiều dài khâu chủ động	rf	150	mm
Chiều dài khâu bị động	re	330	mm
Phạm vi góc quay cánh tay	θ	[-45, 70]	°
Bán kính vùng làm việc	R	150	mm
Chiều cao vùng làm việc	H	100	mm
Tải trọng tối đa	P	0.1	kg

3.2.5.3. Kiểm tra vùng làm việc với Matlab

Với kích thước đã tính được tiến hành mô phỏng trong Matlab và có được kết quả như sau:



Hình 3.6: Kiểm tra vùng làm việc Robot Delta

Khi đặt trung tâm vùng làm việc $D = 300$, $H = 100$ ở độ cao $Z = -300$ so với bệ cố định thì robot delta có thể bao phủ hoàn toàn vùng làm việc này, vậy ta lựa chọn thông số robot Delta như đã tính toán.

3.2.6. Tính toán lựa chọn động cơ

3.2.6.1. Thông số tính toán đầu vào

- Chiều dài cánh tay chủ động (L_1): 0,15 (m)
- Chiều dài cánh tay bị động (L_2): 0,33 (m)
- Khối lượng bệ di động và đầu gấp (m_{ee}): 0,2 (kg)
- Khối lượng vật gấp tối đa (m_p): 0,1 (kg)
- Tổng khối lượng quy đổi về đầu công tác: $m = m_{ee} + m_p = 0,3$ (kg)
- Gia tốc trọng trường: $g = 9,81(m/s^2)$
- Gia tốc yêu cầu của hệ thống: $a_{max} = 10 (m/s^2)$

3.2.6.2. Xác định momen xoắn yêu cầu tại trực cánh tay

Để đảm bảo robot vận hành ổn định ở tốc độ cao mà không bị mất bước, momen xoắn tại trực khớp chủ động được xác định dựa trên tổng momen tĩnh và momen động

- Momen tĩnh (M_{tinh}): Mô-men lớn nhất cần thiết để giữ tải ở trạng thái tĩnh (xuất hiện khi cánh tay chủ động nằm ngang):

$$M_{tinh} = m \cdot g \cdot L_1 = 0,3 \cdot 9,81 \cdot 0,15 = 0,44 Nm$$

- Momen động (M_a): Mô-men cần thiết để thắng lực quán tính khi robot gia tốc đột ngột:

$$M_{dong} = m \cdot a_{max} \cdot L_1 = 0,3 \cdot 10 \cdot 0,15 = 0,45 Nm$$

- Tổng momen yêu cầu (M_{req}): Áp dụng hệ số an toàn $k = 1,5$ để bù đắp ma sát tại các khớp cầu và tổn hao cơ khí:

$$M_{req} = (M_{tính} + M_{động}).k$$

$$\Rightarrow M_{req} = (0,44 + 0,45).1,5 = 1,34 \text{ Nm}$$

3.2.6.3. Lựa chọn phương án động cơ bước và hộp số



Electrical Characteristics	
1. Number of Phase	2
2. Step Angle	1.8°
3. Rated Voltage	2.6V DC
4. Rated Current	2.0A
5. Holding Torque	480mN.m Min (额定扭矩)
6. Phase Resistance	1.3Ω ± 15% (20°C)
7. Phase Inductance	1.9mH ± 20% (1kHz 1V rms)
8. Motor Inertia	82g.cm²
9. Motor Weight	360g Ref.
10. Insulation Resistance	100MΩ Min. (DC 500V)
11. Insulation Class	B(130°C)

Hình 3.7: Thông số động cơ NENA 17

Dựa trên khung máy, động cơ bước NENA 17, thông số momem giữ của động cơ là $M_{motor} \approx 0,5 \text{ Nm}$ không đủ đáp ứng yêu cầu 1,34 (Nm). Nhóm quyết định sử dụng hộp số giảm tốc có tỉ lệ $i = 13,7$ để tăng momen.

- a) Kiểm tra momen đầu ra: Momen thực tế sau hộp số ($M_{thực}$) với hiệu suất truyền động $\eta = 0,8$:

$$M_{thực} = M_{motor} \cdot i \cdot \eta$$

$$\Rightarrow M_{thực} = 0,5 \cdot 13,7 \cdot 0,8 = 5,48 \text{ Nm}$$

Nhận xét $M_{thực}(5,48 \text{ Nm}) > M_{req}(1,34 \text{ Nm})$. Hệ thống dư công suất để hoạt động ở tốc độ cao mà không bị trượt bước.

- b) Kiểm tra độ phân giải: động cơ bước có góc 1,8°, cài đặt vi bước 1/4. Độ phân giải góc quay tại trực cánh tay chủ động là:

$$\Delta\theta = \frac{1,8^\circ}{4 \cdot i} = \frac{1,8^\circ}{4 \cdot 13,7} = 0,0328^\circ/xung$$

3.2.6.4. Đánh giá phương án

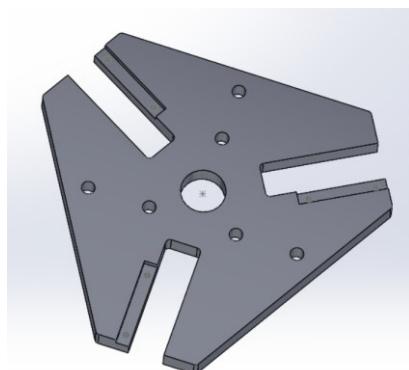
Việc sử dụng hộp số 1:13,7 giúp giải quyết triệt để vấn đề yếu tải của động cơ bước NEMA 17 khi điều khiển trực tiếp. Mặc dù tốc độ quay của động cơ cần cao hơn, nhưng nhờ hiệu năng từ nguồn 24V và Driver TB6600, hệ thống vẫn duy trì được ổn định.

3.2.7. Thiết kế cơ khí chi tiết Robot Delta

3.2.7.1. Tấm đế cố định

Tấm đế cố định được thiết kế với độ cứng vững cao nhằm chịu tải trọng của toàn bộ hệ thống robot Delta. Do kích thước hình học lớn, tấm đế được chia thành 3 module riêng biệt để tối ưu hóa quá trình gia công CNC và tiết kiệm chi phí vật liệu mà vẫn đảm bảo các tiêu chuẩn kỹ thuật đề ra.

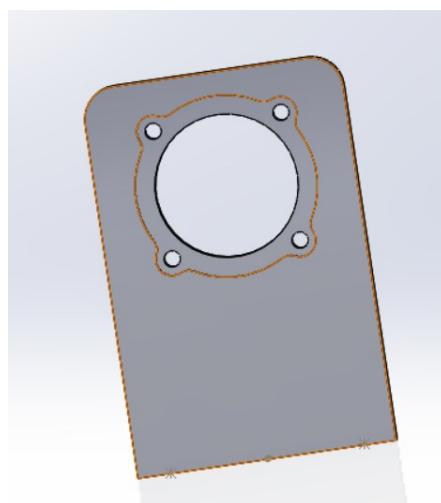
- Vật liệu: Nhôm nguyên khối dạng tấm.
- Kích thước: $f = 60mm$



Hình 3.8: Tấm đế cố định

3.2.7.2. Giá đỡ động cơ và hộp số

Giá đỡ động cơ được sử dụng để cố định động cơ và hộp số, đảm bảo cho động cơ và hộp số không bị xê dịch trong quá trình hoạt động

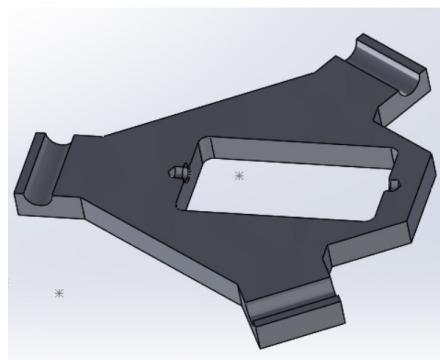


Hình 3.9: Giá đỡ động cơ và hộp số

3.2.7.3. Tấm đế di động

Tấm đế di động được tối ưu hóa về mặt thiết kế nhằm giảm thiểu khối lượng mà vẫn duy trì được độ cứng vững tối đa trong suốt quá trình vận hành. Trên bề mặt tấm đế, các lỗ kín thuật được công chính xác để lắp đặt cơ cấu chấp hành cuối (end-effector).

- Vật liệu: Nhôm nguyên khối dạng tấm.
- Kích thước: $w = 40mm$



Hình 3.10: Tấm đế di động

3.2.7.4. Cánh tay chủ động

Cánh tay chủ động là bộ phận trực tiếp chịu momen xoắn từ động cơ, do đó thiết kế cần đặc biệt chú trọng đến khả năng chống biến dạng và phá hủy khi hoạt động ở tốc độ cao.



Hình 3.11: Cánh tay chủ động

3.3. Lựa chọn thiết bị cho mô hình

3.3.1. Module Driver TB6600

Module Driver TB6600 là một bộ điều khiển động cơ bước (stepper motor driver) phổ biến, mạnh mẽ và giá cả phải chăng, thường được sử dụng rộng rãi trong các ứng dụng máy CNC, máy in 3D và các hệ thống tự động hóa sử dụng vi điều khiển như

STM32. Nó cho phép bạn điều khiển các động cơ bước 2 pha, 4 pha, 4 dây, 6 dây hoặc 8 dây.



Hình 3.12: Module Driver TB6600

- Thông số kỹ thuật:

Bảng 3-4: Thông số Module Driver

STT	Thuộc tính	Thông số
1	Điện áp cấp	9 – 42VDC
2	Dòng điều khiển	0.5 đến 4A
3	Điều khiển vi bước (Microstep)	1, 1/2, 1/4, 1/8, 1/16, 1/32
4	Tín hiệu điều khiển	PUL(xung), DIR (chiều), ENA (kích hoạt)
5	Tần số xung tối đa	~ 200kHz

- Nguyên lý hoạt động:

- Driver TB6600 nhận tín hiệu xung (PUL) nhận tín hiệu từ vi điều khiển để tạo từng bước quay cho động cơ
- Mỗi xung = động cơ quay 1 microstep
- Chân DIR xác định hướng quay (thuận/ngược)
- Chân ENA cho phép driver hoạt động hoặc dừng

- Cách cài đặt DIP Switch:

Chọn vi bước

Bảng 3-5: Thông số cài đặt vi bước

Micro	Pulse/rev	SW1	SW2	SW3
NC	NC	0	0	0
1	200	0	0	1
2A	400	0	1	0

2B	400	0	1	1
4	800	1	0	0
8	1600	1	0	1
16	3200	1	1	0
32	6400	1	1	1

Chọn dòng:

Bảng 3-6: Thông số cài đặt dòng

Current(A)	PK Current	SW4	SW5	SW6
0.5	0.7	0	0	0
1.0	1.2	0	1	0
1.5	1.7	0	0	1
2.0	2.2	0	1	1
2.5	2.7	1	0	0
2.8	2.9	1	1	1
3.0	3.2	1	1	0
3.5	4.0	1	1	1

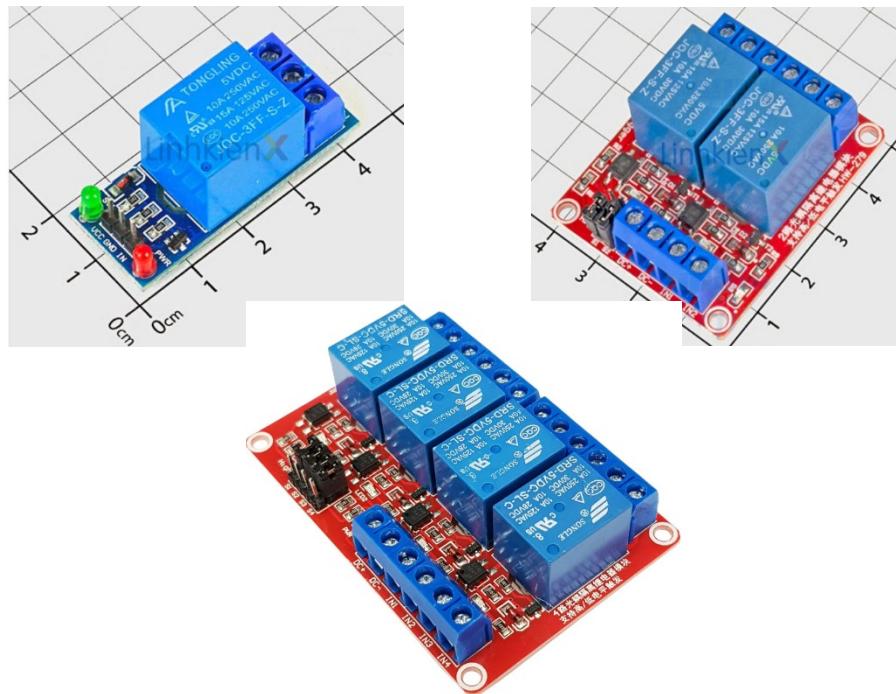
3.3.2. Module Relay

Module Relay là một thiết bị điện tử giúp chuyển mạch bằng cách sử dụng tín hiệu điều khiển điện áp thấp để điều khiển các thiết bị điện công suất cao. Relay đóng vai trò như một công tắc điều khiển từ xa, cho phép bật/tắt các thiết bị điện mà không cần tiếp xúc trực tiếp. Nhờ đó, relay được sử dụng rộng rãi trong các hệ thống tự động hóa, nhà thông minh và IoT.

Các loại Module Relay phổ biến

- Module Relay 1 kênh: Điều khiển một thiết bị duy nhất, thường được sử dụng trong các ứng dụng đơn giản như bật/tắt đèn, quạt hoặc bơm nước.
- Module Relay 2 kênh: Có khả năng điều khiển đồng thời hai thiết bị, thích hợp cho các hệ thống điều khiển đa chức năng như quản lý nhiều khu vực chiếu sáng.

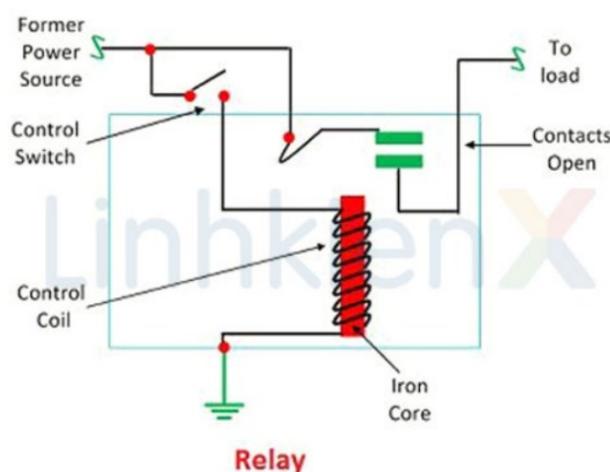
Module Relay 4 kênh: Cho phép điều khiển bốn thiết bị cùng lúc, thường được ứng dụng trong các hệ thống tự động hóa công nghiệp hoặc nhà thông minh.



Hình 3.13: Hình ảnh các Module Relay

- Nguyên lý hoạt động:

Relay hoạt động dựa trên nguyên tắc của điện từ trường. Khi dòng điện chạy qua cuộn dây, nó tạo ra một từ trường hút hoặc nhả tiếp điểm bên trong relay. Nhờ đó, relay có thể đóng hoặc mở mạch điện đầu ra, giúp điều khiển các thiết bị điện lớn bằng tín hiệu điều khiển điện áp thấp.



Hình 3.14:Sơ đồ nguyên lý Module Relay

Cấu tạo chi tiết của Module Relay:

- Cuộn dây (Coil): Là bộ phận tạo từ trường khi có dòng điện chạy qua, quyết định khả năng đóng/mở của relay.
- Tiếp điểm (Contacts): Thành phần quan trọng giúp relay đóng/mở mạch điện, điều khiển trực tiếp thiết bị.
- Diode bảo vệ: Giúp ngăn chặn dòng ngược từ cuộn dây, tránh làm hỏng vi điều khiển.
- Mạch cách ly (Optocoupler, Transistor): Đảm bảo sự an toàn và ổn định của tín hiệu điều khiển, giảm thiểu nhiễu điện.
- LED báo trạng thái: Hiển thị trạng thái hoạt động của relay, giúp người dùng dễ dàng kiểm tra.

3.3.3. Đèn báo

Đèn báo pha (hay còn gọi là đèn báo tín hiệu, đèn chỉ thị) là một thiết bị điện quan trọng, được lắp đặt trên mặt cánh của tủ điện, bảng điều khiển hoặc các thiết bị máy móc.

Chức năng cơ bản của đèn báo pha là cung cấp tín hiệu thị giác (visual signal) nhanh chóng và rõ ràng, giúp người vận hành và kỹ thuật viên giám sát trạng thái hoạt động của hệ thống điện và thiết bị.



Hình 3.15: Đèn báo

- Thông số kỹ thuật:

Bảng 3-7: Thông số kỹ thuật đèn báo

STT	Thuộc tính	Thông số
1	Điện áp định mức	24VDC
2	Dòng định mức	5A
3	Số lượng tiếp điểm	1NO + 1NC + 1COM
4	Nhiệt độ môi trường hoạt động	-10°C ~ 70°C
5	Chất liệu	Kim loại hoặc nhựa chịu lực

Ứng dụng:

- Giám sát trạng thái nguồn điện tổng: báo điện có pha, mất pha,...

- Hiện thị trạng hoạt động các thiết bị
- Cảnh báo lỗi và sự cố

3.3.4. Công tắc hành trình

Công tắc hành trình là thiết bị được sử dụng để giới hạn hành trình hoạt động của các bộ phận chuyển động khác, có chức năng đóng mở, được đặt tại một vị trí nhất định trên đường hoạt động của một dòng điện hay động cơ nào đó mà đến vị trí của công tắc sẽ có sự thay đổi xảy ra



Hình 3.16: Công tắc hành trình

Thông số kỹ thuật:

Bảng 3-8: Thông số công tắc hành trình

STT	Thuộc tính	Thông số
1	Xuất sứ	Đài Loan
2	Kích thước lỗ	22mm -23mm
3	Dài	51mm
4	Điện áp hoạt động	220VDC
5	Dòng tiêu thụ	> 20mA
6	Tuổi thọ	Trên 100000 giờ sáng liên tục

3.3.5. Bơm hút chân không

Bơm hút chân không mini nổi bật với thiết kế kích thước nhỏ gọn lý tưởng cho thiết bị di động, khả năng tự mồi mà không cần chuẩn bị, và cơ chế hoạt động khô không yêu cầu chất bôi trơn hay chất lỏng làm mát.



Hình 3.17:Bơm hút chân không

Thông số kỹ thuật:

Bảng 3-9: Thông số kỹ thuật Pump

STT	Thuộc tính	Thông số
1	Loại bơm	Bơm màng
2	Điện áp hoạt động	24VDC
3	Động cơ	Động cơ chổi than DC
4	Dòng điện	0.2 ~ 0.6A
5	Áp suất chân không tối đa	-85kPa ~ -60kPa
6	Lưu lượng khí	5L/min ~ 15L/min
7	Môi chất bơm	Không khí, khí tro
8	Độ ồn	Thường ≤ 65dB
9	Tuổi thọ	3000 ~5000 giờ

3.3.6. Van điện từ

Van điện từ 2V025-08 là một thành phần quan trọng trong hệ thống tự động hóa, đóng vai trò như một chiếc "khóa điện" để kiểm soát dòng lưu chất (khí nén, nước, dầu hoặc gas) một cách chính xác. Với thiết kế nhỏ gọn và độ tin cậy cao, đây là loại van phổ biến nhất trong các ứng dụng điều khiển ON/OFF hiện nay.

Trong mô hình này van điện từ đóng vai trò phá chân không, khi động cơ ngừng hút, nó không có cơ chế cho dòng khí đi ngược lại làm giữ chân không nên vật không rơi khỏi đầu hút.



Hình 3.18: Van điện 2V025-08

3.3.7. Nút nhấn

Nút nhấn là loại nút bấm dùng nguồn điện một chiều, khi nhấn thì tiếp điểm thay đổi trạng thái (Mở/Đóng) và tự động trở về trạng thái ban đầu khi nhả tay, không có đèn báo hiệu, thường dùng trong tủ điện, điều khiển máy móc, thiết bị công nghiệp (như nút khởi động, dừng khẩn cấp, chuông) với cấu tạo có các chân tiếp điểm thường mở (NO) và thường đóng (NC).



Hình 3.19: Nút nhấn

- Thông số kỹ thuật:

Bảng 3-10: Thông số nút nhấn

STT	Thuộc tính	Thông số
1	Điện áp tối đa	660VDC
2	Dòng tối đa	10A
3	Màu sắc	Đỏ, vàng, xanh
4	Kích thước	66x 36x 29mm
5	Khối lượng	50g

3.3.8. Camera Sony IMX258

Sony IMX258 là cảm biến hình ảnh CMOS 13MP hiệu suất cao thuộc dòng Exmor RSTM, được thiết kế với cấu trúc xếp chồng (stacked) và công nghệ chiếu sáng mặt sau (BSI). Cảm biến này mang lại sự cân bằng tối ưu giữa độ phân giải cao, kích thước nhỏ gọn và mức tiêu thụ điện năng thấp, lý tưởng cho các ứng dụng quét tài liệu, thị giác máy tính và thiết bị nhúng.



Hình 3.20: Sony IMX258

Thông số kỹ thuật:

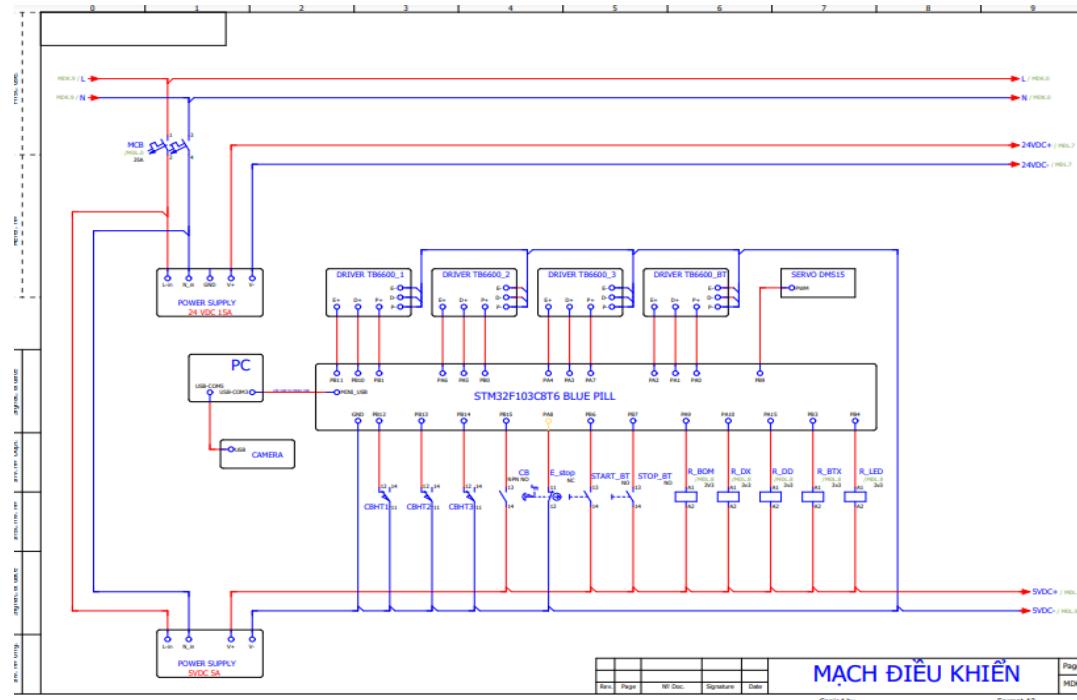
Bảng 3-11: Thông số IMX258

STT	Thuộc tính	Thông số
1	Độ phân giải tối đa	13.1 Megapixel (4224 x 3192px)
2	Kích thước cảm biến	1/3.06 inch
3	Kích thước điểm ảnh	1.12 um x 1.12 um

4	Tốc độ khung hình (USB 2.0)	10 fps ở 13MP; 30 fps ở 1080p
5	Dải động	68DB

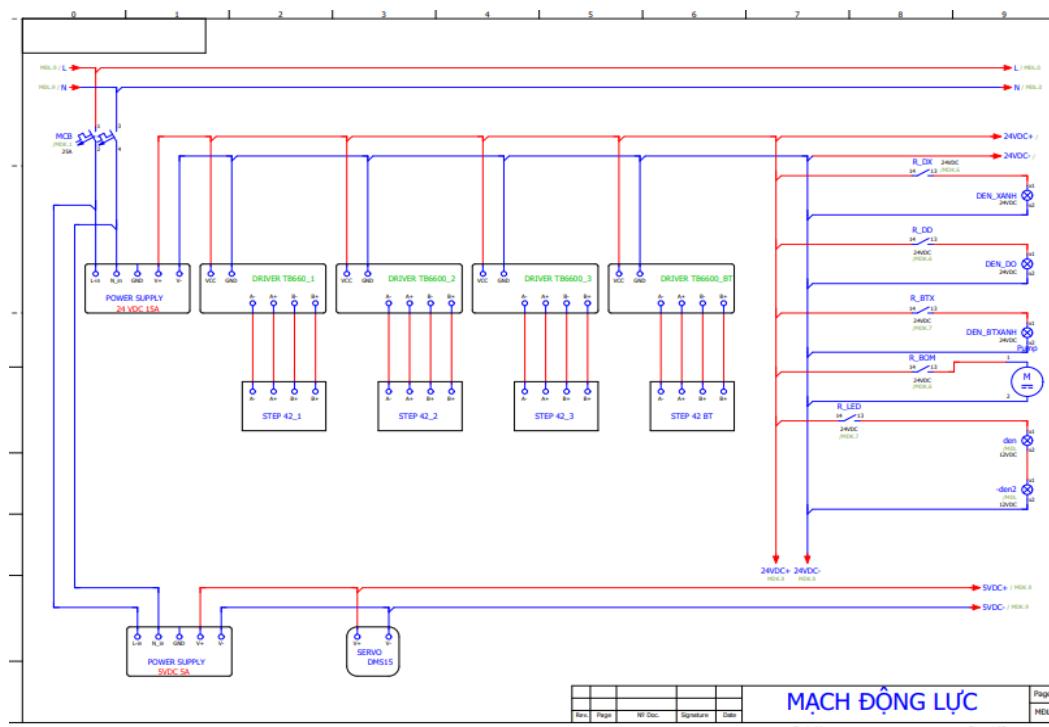
3.4. Sơ đồ đi dây

3.4.1. Sơ đồ mạch điều khiển



Hình 3.18: Sơ đồ mạch điều khiển

3.4.2. Sơ đồ mạch động lực



Hình 3.19: Sơ đồ mạch động lực

3.5. Thiết kế tủ điện

3.5.1. Mặt trong tủ

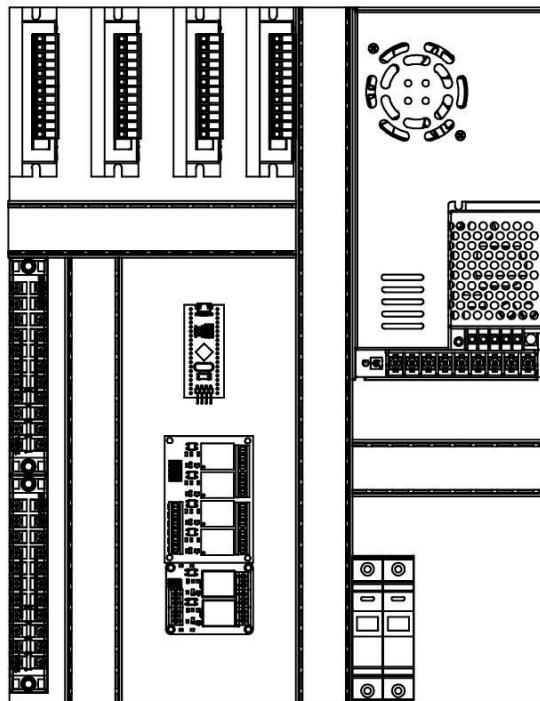
Tủ điện được xem như bộ não của hệ thống điều khiển tự động vì nó chứa bao gồm tất cả thiết bị dùng điều khiển các thiết bị tải.

Mục đích:

Bảo vệ các thiết bị điện: giúp các thiết bị bên trong khỏi bụi bẩn và nhiệt độ cao, nước ẩm và các yếu tố môi trường khác.

An toàn: Tủ điện là giúp hạn chế thiệt hại do sự cố cháy nổ hoả hoạn và đồng thời bảo mật hệ thống điện trước sự xâm nhập trái phép.

Phân phối điện: Tủ điện là nơi đấu nối và phân phối điện cho các công trình đảm bảo điện được phân phối đúng nơi và an toàn



Hình 3.20: Bản vẽ mặt trong của tủ

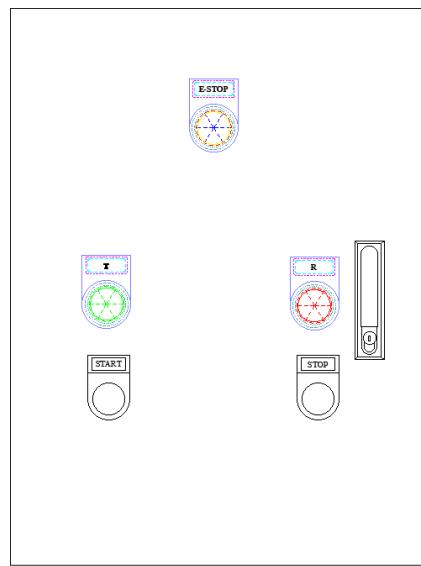
Mô tả

Tủ điện gồm những thiết bị:

- Hệ thống nguồn tổ ong: Sử dụng song song hai loại nguồn là 24V (cung cấp năng lượng cho 4 Driver TB6600 để chạy động cơ robot, động cơ băng tải và vận hành bơm hút chân không) và 5V (cấp nguồn nuôi mạch STM32 cùng động cơ Servo điều khiển xoay đầu gấp)

- Bộ điều khiển trung tâm: Một vi điều khiển STM32F103C8T6 đóng vai trò xử lý các chương trình và điều hành toàn bộ hoạt động của hệ thống.
- Các bộ truyền động: Gồm 4 Module Driver TB6600 dùng để điều khiển chính xác các bước quay của động cơ robot và băng tải.
- Hệ thống ro-le và bảo vệ: Sử dụng 6 Module Relay để đóng ngắt các thiết bị như đèn, động cơ. Ngoài ra còn có CB (Aptomat) để bảo vệ quá dòng, ngắt mạch khi gặp sự cố và một công tắc tơ.

3.5.2. Mặt ngoài tủ



Hình 3.21: Bản vẽ mặt ngoài tủ

Mô tả:

Mặt ngoài được bố trí các nút nhấn và đèn báo dễ dàng điều khiển và giám sát trạng thái hoạt động của hệ thống gồm 3 nút nhấn và 2 đèn báo

Start: Nút nhấn khởi động hệ thống ở chế độ Auto

Stop: Nút nhấn dừng hoạt động của hệ thống một cách bình thường.

Estop: Nút nhấn dừng khẩn cấp dùng trong trường hợp có sự cố nguy hiểm. Khi nhấn hệ thống lập tức ngừng hoạt động

Đèn báo xanh: Hệ thống robot đang hoạt động

Đèn báo đỏ: Báo hiệu hệ thống đang gặp lỗi

3.6. Thi công mô hình

3.6.1. Khâu lắp ráp khung và đế Robot

- Sử dụng các thanh nhôm định hình để lắp ghép phần khung xương chắc chắn cho toàn bộ hệ thống.

- Cố định tấm bệ nhôm hình tam giác lên khung để làm giá đỡ cho 3 động cơ bước.
- Tinh chỉnh độ phẳng và độ vuông góc của khung nhôm để triệt tiêu rung động, giữ cho mô hình đứng vững ngay cả khi động cơ hoạt động ở công suất tối đa.



Hình 3.22: Gia cố lắp đặt động cơ

3.6.2. Khâu lắp ráp cánh tay Robot

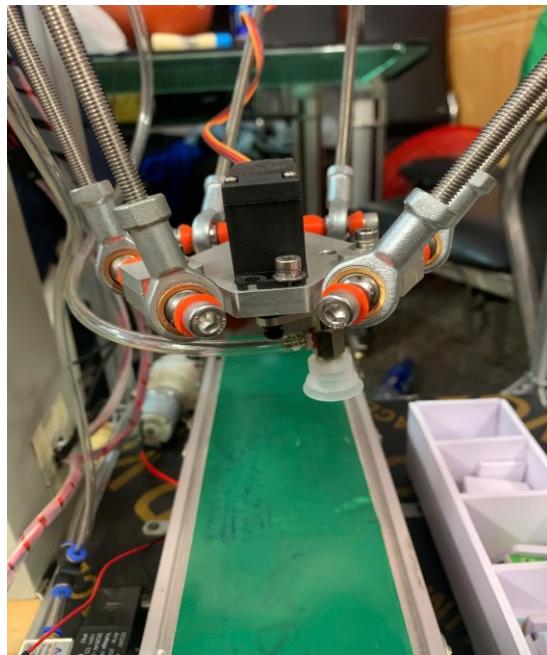
- Gắn trực tiếp 3 cánh tay trên có chiều dài 149mm vào trực của các động cơ bước.
- Nối các cặp thanh cánh tay dưới dài 325mm với cánh tay trên thông qua các khớp cầu.
- Kiểm tra độ linh hoạt của các khớp để cánh tay có thể chuyển động mượt mà, không bị kẹt



Hình 3.23: Khớp nối của cánh tay robot

3.6.3. Khâu lắp đặt bệ di động và đầu gấp

- Lắp đặt đầu hút chân không vào chính giữa bệ di động tam giác cạnh 139mm.
- Gắn thêm động cơ Servo nhỏ để điều khiển việc xoay đầu gấp theo hướng của vật thể.
- Cố định ống dẫn khí từ đầu hút về máy bơm chân không để đảm bảo lực hút ổn định.



Hình 3.24:Lắp đặt động servo và giác hút

3.6.4. Khâu lắp đặt hệ thống băng tải và cảm biến

- Đặt băng tải dọc theo khung máy, chạy xuyên qua vùng làm việc của cánh tay robot.
- Gắn các công tắc hành trình tại vị trí cao nhất của cánh tay trên để xác định gốc tọa độ (Homing).

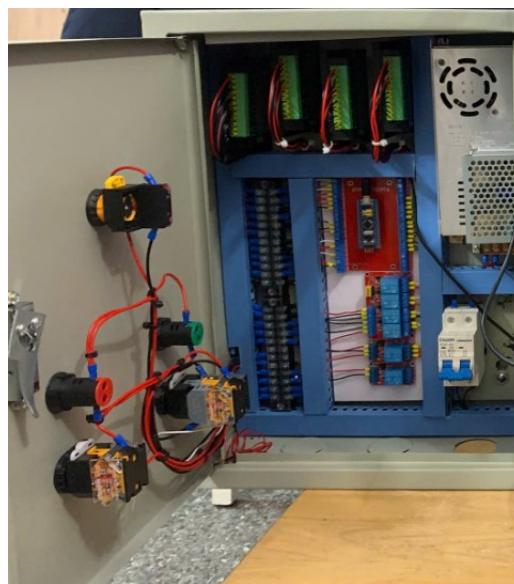
3.6.5. Khâu lắp đặt hệ thống camera

- Lắp đặt giá đỡ camera ở phía trên cao, ống kính hướng thẳng xuống mặt băng tải.
- Căn chỉnh vị trí camera sao cho nhìn thấy sản phẩm từ sớm để máy tính kịp thời xử lý.
- Đảm bảo khoảng cách giữa camera và robot đủ xa để robot có đủ thời gian nhận lệnh và thực hiện thao tác gấp.



Hình 3.25: Khâu lắp camera

3.6.6. Tủ điện



Hình 3.26: Tủ điện

- Mô tả:

Tủ điện có thiết bị sau:

- 1 STM32F103C8T6
- 1 nguồn tổ ong 24V và 5V
- 6 module relay
- 1 công tắc tơ
- 2 đèn báo

- 3 nút nhấn
- 4 driver TB6600

CHƯƠNG 4: LẬP TRÌNH ĐIỀU KHIỂN

4.1. Quy trình công nghệ

Để hệ thống vận hành chính xác và tránh các va chạm cơ khí không đáng có, quy trình công nghệ được thiết lập theo trình tự bắt buộc: thực hiện thiết lập ở chế độ Thủ công (Manual) trước, sau đó mới kích hoạt chế độ Tự động (Auto).

4.1.1. Chế độ thủ công và thiết lập ban đầu

Đây là bước đệm quan trọng, dùng để kiểm tra thiết bị và đưa robot về vị trí sẵn sàng làm việc.

- Kết nối hệ thống: Đầu tiên, cần kiểm tra và thiết lập kết nối giữa máy tính (PC) và vi điều khiển STM32 qua cổng USB. Khi phần mềm báo đã nhận diện được cổng COM thành công, dữ liệu tọa độ mới có thể truyền nhận thông suốt.
- Thực hiện lấy gốc (Homing): Vì khi mới cấp điện, robot chưa xác định được vị trí hiện tại trong không gian, nên ta phải nhấn lệnh Homing trên giao diện. Lúc này, STM32 điều khiển 3 cánh tay robot đi lên cho đến khi chạm vào các công tắc hành trình gắn ở đỉnh khung nhôm. Đây là bước để máy tự xác lập "tọa độ 0" chuẩn cho mọi hoạt động gấp thả sau đó.
- Kiểm tra thiết bị ngoại vi: Trong chế độ này, người dùng có thể bấm thử các nút chức năng trên màn hình để xem băng tải có quay không, đầu hút chân không có dính vật không và động cơ Servo xoay đầu hút có mượt không.
- Xác nhận sẵn sàng: Chỉ khi robot đã về vị trí Home an toàn và các thiết bị chạy ổn định, hệ thống mới cho phép chuyển sang chế độ Tự động.

4.1.2. Chế độ tự động (Auto)

Khi đã hoàn tất các bước setup ở chế độ Manual, ta nhấn nút **START** trên mặt tủ điện hoặc trên giao diện điều khiển để máy bắt đầu quy trình làm việc khép kín.

- Nhận diện và phân loại ảnh: Băng tải bắt đầu chạy để đưa sản phẩm đi tới. Khi vật phẩm đi qua tầm mắt của Camera, máy tính sẽ chụp ảnh và sử dụng mô hình học sâu để biết đây là vật nào nằm ở vị khay nào.
- Tính toán tọa độ: Máy tính lập tức tính toán sau đó gửi lệnh điều khiển xuống mạch STM32 qua cổng USB.
- Thực hiện gấp vật: Nhận được lệnh, STM32 phát xung cho 3 động cơ bước đưa cánh tay robot lao xuống đúng chỗ sản phẩm. Nếu sản phẩm nằm nghiêng, động cơ Servo sẽ xoay đầu gấp cho khớp hướng, đồng thời kích hoạt bơm màng để tạo lực hút giữ chặt vật.

- Phân loại sản phẩm: Robot nhắc vật lên và đưa về đúng vị trí đã quy định rồi ngắt lực hút để thả vật xuống.
- Lắp lại chu kỳ: Xong một lượt, robot quay về vị trí chờ sản phẩm tiếp theo. Đèn báo trên tủ điện sẽ sáng liên tục trong suốt quá trình này để thông báo máy vẫn đang vận hành bình thường.

4.2. Tổng quan kiến trúc điều khiển

Hệ thống được thiết kế theo mô hình **Master-Slave**, trong đó máy tính (PC) đóng vai trò Master chịu trách nhiệm xử lý tác vụ nặng (Computer Vision, Kinematics, Nội suy quỹ đạo), và vi điều khiển STM32 đóng vai trò Slave chịu trách nhiệm điều khiển thời gian thực (phát xung điều khiển động cơ bước)

4.2.1. Giao thức giao tiếp

Để đảm bảo việc truyền tải dữ liệu giữa PC và STM32 diễn ra chính xác, tin cậy và đáp ứng được yêu cầu về tốc độ, đồ án sử dụng giao thức giao tiếp qua cổng **USB CDC** (**Communication Device Class**) giả lập cổng Serial.

4.2.1.1. Cấu hình vật lý:

- **Baudrate giả lập:** 115200 bps (để tương thích phần mềm).
- **Tốc độ vật lý:** ~12 Mbps. **Kết nối:** Cáp USB Type-C/Micro-USB nối trực tiếp từ PC xuống cổng USB của STM32.

4.2.1.2. Định dạng gói tin.

Dữ liệu được truyền dưới dạng chuỗi ký tự ASCII để thuận tiện cho việc gỡ lỗi (debug), cấu trúc lệnh được thiết kế theo dạng KEYWORD:DATA.

- Lệnh từ PC gửi xuống (Request):
 - ADD_BLOCK:id:{params}: Lệnh di chuyển quan trọng nhất. Trong đó, params là chuỗi định dạng giống JSON chứa thông tin thời gian thực hiện (t), số bước động cơ (s), góc servo (a) và trạng thái bơm (b).
 - HOME: Yêu cầu robot về điểm gốc.
 - STATUS: Yêu cầu vi điều khiển báo cáo trạng thái hiện tại.
 - CONVEYOR:START/STOP: Điều khiển băng tải.
- Phản hồi từ STM32 gửi lên (Response):
 - DONE:id: Báo cáo đã thực hiện xong lệnh di chuyển có mã id. Tín hiệu này cực kỳ quan trọng để PC biết khi nào nên gửi tiếp lệnh mới (Flow Control).

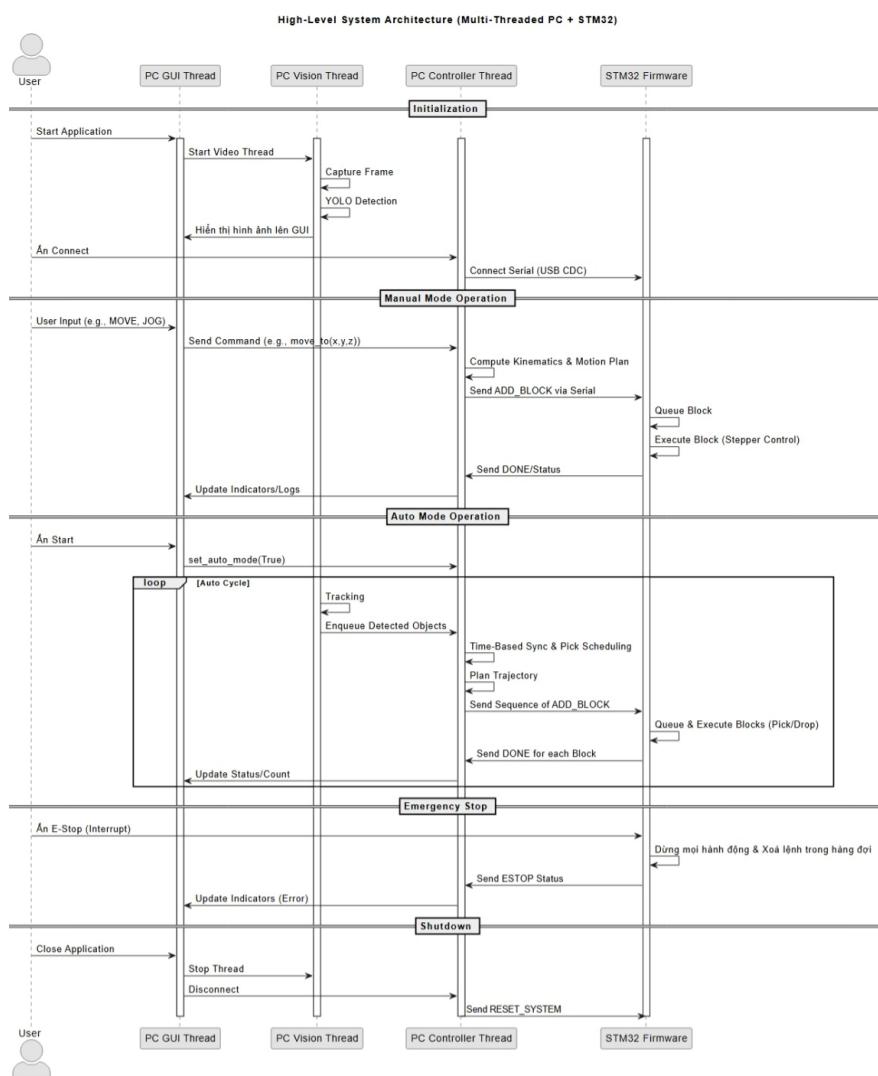
- o STATUS:state:homed:estop...: Gói tin chứa toàn bộ trạng thái sensor, nút nhấn, tọa độ hiện tại.
- o ERROR:msg: Báo lỗi hệ thống.

4.2.1.3. Cơ chế điều khiển luồng (Flow Control)

Do bộ nhớ RAM của STM32 có hạn, hệ thống không gửi toàn bộ quỹ đạo di chuyển một lúc. Thay vào đó, một cơ chế "**Sliding Window**" được áp dụng: PC duy trì một hàng đợi lệnh trên STM32 luôn đầy ở mức an toàn. Khi STM32 thực hiện xong một lệnh, PC sẽ lập tức gửi lệnh tiếp theo để điền vào chỗ trống.

4.2.2. Tổng quát về kiến trúc hệ thống

Dựa trên sơ đồ kiến trúc hệ thống, hoạt động của robot được chia thành các giai đoạn chính sau đây, tận dụng kiến trúc đa luồng trên máy tính để đảm bảo xử lý song song các tác vụ.



Hình 4.1: Kiến trúc của hệ thống robot Delta

a. Khởi tạo (Initialization)

- Khi người dùng khởi động ứng dụng, ba luồng chính được kích hoạt: Luồng giao diện (GUI), Luồng xử lý ảnh và Luồng điều khiển.
- **Vision Thread:** Ngay lập tức kết nối với Camera, bắt đầu chụp hình và nạp mô hình YOLOv8 để sẵn sàng nhận diện [7].
- **Controller Thread:** Người dùng ấn nút "Connect", PC thiết lập kết nối Serial với STM32. STM32 khởi tạo các ngoại vi (GPIO, Timer, USB) và đi vào trạng thái chờ lệnh.

b. Chế độ Thủ công

Đây là chế độ dùng để kiểm tra, tinh chỉnh hoặc xử lý sự cố:

- Khi người dùng thao tác trên GUI (ví dụ: nhấn nút JOG X+), lệnh move được gửi đến Controller Thread.
- PC tính toán động học nghịch để ra số bước, sau đó đóng gói thành lệnh ADD_BLOCK gửi xuống STM32.
- STM32 nhận lệnh, đưa vào hàng đợi và sử dụng ngắt Timer để điều khiển động cơ bước chạy đúng quỹ đạo. Sau khi hoàn thành, STM32 gửi DONE để PC cập nhật giao diện.

c. Chế độ Tự động

Đây là chế độ hoạt động chính của hệ thống, thể hiện tính liên tục và đồng bộ thời gian thực:

1. **Nhận diện:** Vision Thread liên tục xử lý hình ảnh từ camera. Khi phát hiện vật thể, nó gửi thông tin (tọa độ, loại vật, góc xoay) vào một hàng đợi an toàn (Thread-safe Queue).
2. **Lập lịch:** Controller Thread lấy dữ liệu từ hàng đợi. Dựa trên tốc độ băng tải và vị trí hiện tại của robot, thuật toán sẽ tính toán điểm đón (Intercept Point) sao cho robot và vật gặp nhau tại đúng một thời điểm trong tương lai.
3. **Quy hoạch & Gửi lệnh:** Quỹ đạo di chuyển được chia nhỏ thành nhiều đoạn ngắn và gửi liên tục xuống STM32 thông qua cơ chế Sliding Window đã mô tả ở trên.
4. **Thực thi:** STM32 thực thi các lệnh trong bộ đệm vòng (Ring Buffer), điều khiển đầu hút và thả vật mà không cần dừng lại chờ PC, giúp chuyển động mượt mà.

d. Xử lý An toàn (Emergency Stop)

- Nút E-Stop được đấu nối trực tiếp vào phần cứng STM32 và được xử lý bằng ngắt ngoài (External Interrupt).
- Khi nút E-Stop bị nhấn, STM32 lập tức ngắt xung cấp cho động cơ, xóa toàn bộ hàng đợi lệnh và gửi trạng thái ESTOP lên PC.
- PC nhận tín hiệu sẽ khóa giao diện điều khiển và hiển thị cảnh báo lỗi, ngăn chặn mọi hành động không kiểm soát thao tác cho đến khi xử lý được sự cố
 - e. Đóng hệ thống (Shutdown)
- Khi người dùng tắt ứng dụng, PC gửi lệnh ngắt kết nối và giải phóng tài nguyên Camera, Serial để tránh treo cổng COM cho lần sử dụng sau. STM32 nhận lệnh RESET_SYSTEM để đưa các cơ cấu chấp hành về trạng thái an toàn.

4.3. Xây dựng chương trình điều khiển STM32

4.3.1. Cấu trúc tổ chức mã nguồn

Chương trình điều khiển nhúng được viết bằng ngôn ngữ C, sử dụng thư viện HAL (Hardware Abstraction Layer) của STM32 để đảm bảo tính ổn định và dễ dàng bảo trì. Mã nguồn được chia thành các module chức năng riêng biệt nhằm đảm bảo tính đóng gói (encapsulation).

Danh sách các tệp tin mã nguồn chính trong dự án:

2. Tầng Ứng dụng (Application Layer):

- main.c: Tệp chính chứa hàm main(), khởi tạo hệ thống và vòng lặp vô tận (while(1)). Đóng vai trò bộ điều phối trung tâm, gọi các hàm xử lý từ các module khác.

3. Tầng Logic Điều khiển (Control Logic Layer):

- robot_control.c: Chứa thuật toán cốt lõi điều khiển động cơ bước, quy trình về gốc (Homing), và xử lý ngắt Timer để tạo xung.
- command_queue.c: Cài đặt cấu trúc dữ liệu hàng đợi vòng (Ring Buffer) để lưu trữ các lệnh chuyển động, giúp tách biệt quá trình nhận lệnh và thực thi lệnh.
- command_parser.c: Phân tích chuỗi ký tự nhận được từ máy tính (JSON text) thành các tham số số học (thời gian, bước, trạng thái I/O).

4. Tầng Giao tiếp & Ngoại vi (Driver/Peripheral Layer):

- cdc_handler.c: Xử lý giao tiếp USB CDC ở cấp độ thấp, nhận dữ liệu thông tin từ bộ đệm phần cứng.

- button_handler.c: Xử lý ngắt nút nhấn, khử rung (debounce) và logic dừng khẩn cấp (E-Stop).
- conveyor.c: Các hàm điều khiển động cơ băng tải (Start/Stop/Set Speed).
- status_led.c: Điều khiển đèn báo trạng thái hệ thống.

4.3.2. Phân công đầu ra đầu vào và cấu hình STM32 trên Cube MX

4.3.2.1. Phân công đầu vào đầu ra

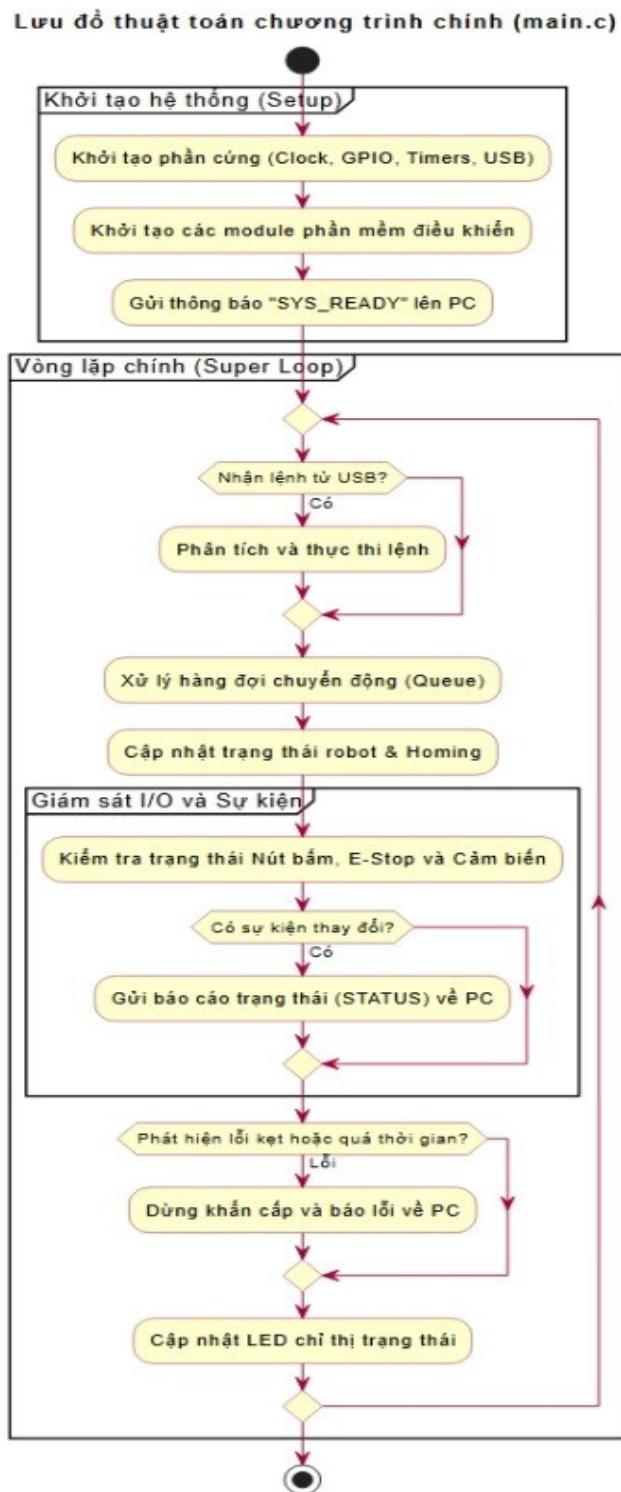
Bảng 4-1: Bảng phân công đầu vào đầu ra

	STT	Tên chức năng	Ký hiệu trong code	Cổng	Chân	Cấu hình nội trở (Đầu vào)
Đầu ra	1	PUL Motor1	M1_PUL	GPIOB	Pin 1	
	2	DIR Motor1	M1_DIR	GPIOB	Pin 10	
	3	ENA Motor1	M1_ENA	GPIOB	Pin 11	
	4	PUL Motor2	M2_PUL	GPIOB	Pin 0	
	5	DIR Motor2	M2_DIR	GPIOA	Pin 5	
	6	ENA Motor2	M2_ENA	GPIOA	Pin 6	
	7	PUL Motor3	M3_PUL	GPIOA	Pin 7	
	8	DIR Motor3	M3_DIR	GPIOA	Pin 3	
	9	ENA Motor3	M3_ENA	GPIOA	Pin 4	
	10	PUL MotorBT	CONV_PUL	GPIOA	Pin 0	
	11	DIR MotorBT	CONV_DIR	GPIOA	Pin 1	
	12	ENA MotorBT	CONV_ENA	GPIOA	Pin 2	
	13	Bơm hút chân không	PUMP	GPIOA	Pin 9	

	14	Đèn trợ sáng camera	CAMERA_LIGHT	GPIOB	Pin 4	
	15	Đèn báo xanh	GREEN_LED	GPIOA	Pin 10	
	16	Đèn báo đỏ	RED_LED	GPIOB	Pin 15	
	17	Đèn nút Start	BUTTON_LED	GPIOB	Pin 3	
Đầu vào	18	Cảm biến khay	CONV_SENSOR	GPIOB	Pin 15	Pull_Down
	19	CBHT1	LS1	GPIOB	Pin 12	Pull_up
	20	CBHT2	LS2	GPIOB	Pin13	Pull_up
	21	CBHT3	LS3	GPIOB	Pin14	Pull_up
	22	E_STOP	ESTOP	GPIOA	Pin 8	Pull_Up
	23	START	BTN_START	GPIOB	Pin 6	Pull_Down
	24	STOP	BTN_STOP	GPIOB	Pin 7	Pull_Down

4.3.3. Xây dựng chương trình chính

Chương trình chính trong tệp main.c là trung tâm điều khiển của hệ thống phần vững trên vi điều khiển STM32F103. Nó chịu trách nhiệm khởi tạo các thành phần phần cứng và phần mềm, đồng thời quản lý vòng lặp chính để xử lý các sự kiện thời gian thực. Chương trình được thiết kế theo mô hình không chặn, sử dụng các hàm cập nhật định kỳ để đảm bảo hệ thống phản ứng nhanh chóng với các đầu vào từ nút bấm, cảm biến, và giao tiếp USB, đồng thời duy trì tính an toàn và hiệu suất cao.



Hình 4.2: Lưu đồ thuật toán vòng lặp xử lý chính của STM32

Chương trình chính được chia làm hai giai đoạn: giai đoạn khởi tạo và vòng lặp chính.

a. Giai đoạn khởi tạo (Initialization)

Ngay khi cấp điện, hệ thống thực hiện các bước sau:

1. HAL_Init & SystemClock_Config: Cấu hình thư viện HAL và thiết lập bộ dao động thạch anh ngoại (HSE) để hệ thống chạy ở tốc độ 72MHz.

2. Khởi tạo Ngoại vi (Peripheral Init):

- MX_GPIO_Init: Cấu hình các chân I/O cho Step/Dir, bơm hút, cảm biến hành trình, nút bấm.
- MX_TIMx_Init: Cấu hình Timer 2 (PWM băng tải), Timer 4 (PWM Servo), và Timer 3 (Bộ đếm bước).
- MX_USB_DEVICE_Init: Kích hoạt giao tiếp USB CDC (Virtual COM Port).

3. Khởi tạo Module phần mềm: Gọi các hàm init của từng module con (robot_init, conveyor_init, queue_init) để thiết lập các biến trạng thái ban đầu.

4. Thông báo sẵn sàng: Sau khi ổn định (delay 100ms), gửi chuỗi "SYS_READY" lên máy tính để báo hiệu quá trình khởi động hoàn tất.

b. Vòng lặp chính (Main Loop)

Vòng lặp while(1) thực hiện các tác vụ tuần tự với tốc độ rất cao:

1. Xử lý giao tiếp (Communication):

- Hàm cdc_handler_get_command kiểm tra xem có dữ liệu mới từ bộ đệm USB không.
- Nếu có lệnh, parse_command sẽ phân tích chuỗi ký tự (ví dụ: ADD_BLOCK, CONVEYOR, HOME) và thực thi hành động tương ứng ngay lập tức.

2. Quản lý hàng đợi chuyển động (Queue Management):

- queue_process(): Kiểm tra xem robot có đang rảnh không và hàng đợi có lệnh chờ không. Nếu có, nó nạp lệnh tiếp theo vào bộ điều khiển robot.
- queue_handle_done_messages(): Xử lý các tín hiệu từ ngắt (Interrupt Service Routine - ISR) báo về khi một chuyển động hoàn tất, gửi phản hồi ACK hoặc DONE lên PC.

3. Giám sát đầu vào (Input Monitoring):

- Nút bấm & E-Stop: button_handler_update() đọc trạng thái nút nhấn và nút dừng khẩn cấp, có áp dụng thuật toán chống rung (debounce). Nếu E-Stop bị kích hoạt, trạng thái sẽ được gửi ngay lập tức về PC.
- Cảm biến khay (Tray Sensor): Giám sát chân PB15. Nếu trạng thái khay thay đổi (có/không có khay), hệ thống gửi gói tin STATUS cập nhật cho phần mềm điều khiển.

4. Logic robot & an toàn (Safety Logic):

- robot_update_homing_state(): Quản lý máy trạng thái của quy trình về Home (Homing sequence).
- Kiểm tra quá trình về home: Nếu quá trình về Home kéo dài quá lâu (do kẹt cơ khí hoặc hỏng cảm biến), hệ thống tự động hủy và báo lỗi.
- Công tắc hàng trình: Kiểm tra xem các công tắc hành trình có bị kẹt (stuck) hay không. Nếu phát hiện bất thường, robot sẽ dừng khẩn cấp (robot_abort) để bảo vệ phần cứng.

5. Chỉ thị trạng thái: status_led_update() điều khiển đèn LED trên mạch nhấp nháy theo các tần số khác nhau tùy thuộc vào trạng thái hiện tại (đang chạy, lỗi, hoặc chờ), giúp người vận hành dễ dàng nhận biết trực quan.

4.3.4. Xử lý lệnh từ PC

4.3.4.1. Mục đích

Trong quá trình hoạt động tự động, máy tính (PC) gửi xuống vi điều khiển hàng loạt các tọa độ chuyển động liên tiếp (Path Planning). Tốc độ gửi dữ liệu qua USB (12 Mbps) nhanh hơn rất nhiều so với tốc độ cơ học của robot (thường mất vài trăm mili-giây để thực hiện một chuyển động).

Nếu robot thực hiện xong một lệnh rồi mới chờ nhận lệnh tiếp theo, chuyển động sẽ bị giật cục (Stop-and-Go). Để giải quyết vấn đề này, hệ thống sử dụng cơ chế hàng đợi lệnh theo cấu trúc vào trước thì xử lý trước và bộ đệm vòng điền này cho phép vi điều khiển nhận trước các lệnh tương lai trong khi đang thực hiện lệnh hiện tại.

4.3.4.2. Cấu trúc dữ liệu

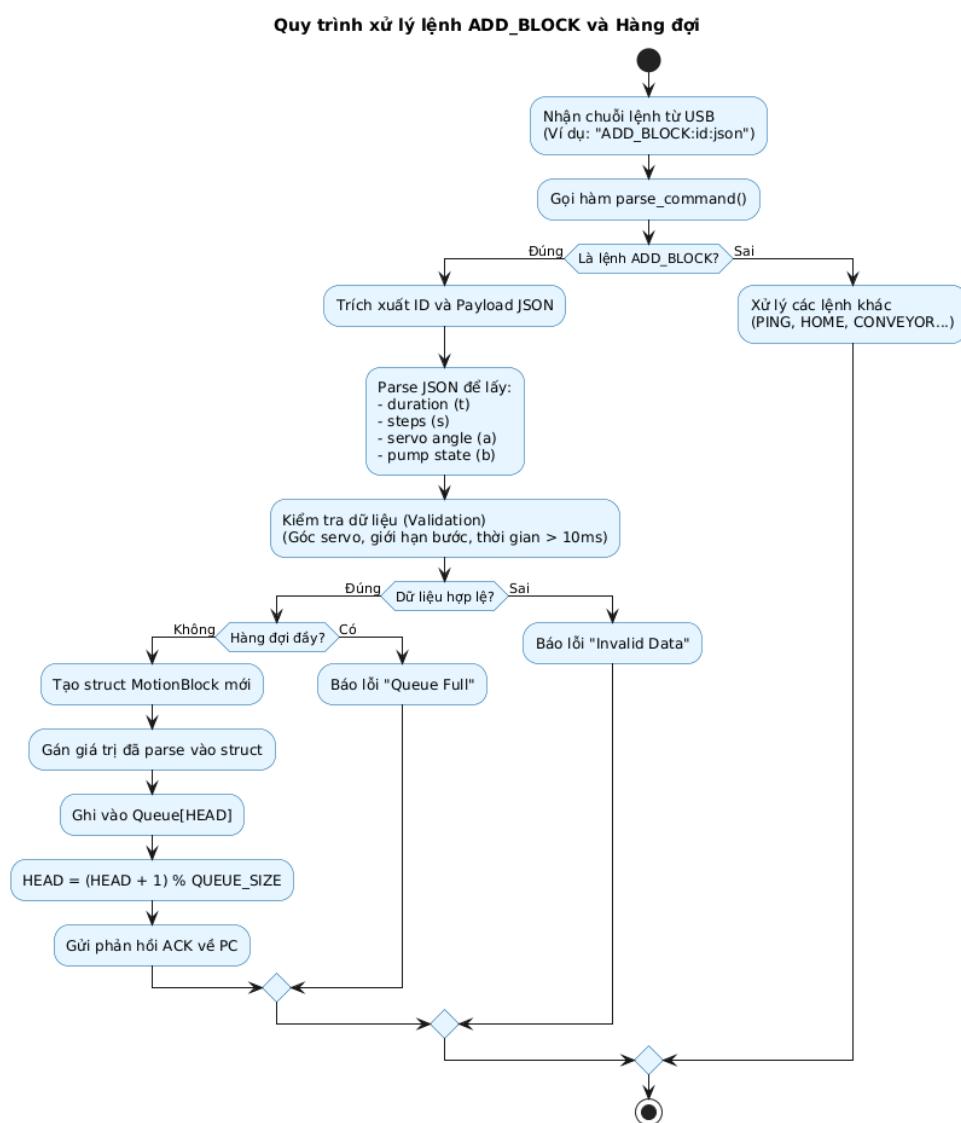
Mỗi lệnh chuyển động được đóng gói thành một cấu trúc dữ liệu MotionBlock. Cấu trúc này chứa đầy đủ thông tin cần thiết để bộ điều khiển robot thực thi mà không cần tính toán lại.

```
typedef struct {  
    uint32_t id; là ID của block (để đồng bộ với PC)  
    int32_t motor_steps[3]; là số bước cần di chuyển cho 3 trục (Delta Steps)  
    uint32_t duration; là thời gian thực hiện chuyển động (ms)  
    uint16_t servo_pulse; là góc quay Servo (độ rộng xung PWM)  
    bool pump_state; là trạng thái bơm hút (ON/OFF)
```

4.3.4.3. Quy trình xử lý

Quy trình từ lúc nhận chuỗi ký tự thô từ USB đến khi đưa vào hàng đợi diễn ra như sau:

1. Phân tích cú pháp (Parsing): Hệ thống nhận chuỗi lệnh ADD_BLOCK định dạng JSON (ví dụ: 0:{ "t":500, "s": [100,200,300]...}).
2. Trích xuất dữ liệu: Các trường thông tin như thời gian t, số bước s, góc servo a được tách ra và chuyển đổi thành số nguyên.
3. Kiểm tra tính hợp lệ: Đảm bảo các giá trị nằm trong giới hạn an toàn (ví dụ: góc servo không quá giới hạn cơ khí, thời gian không bằng 0).
4. Đẩy vào hàng đợi: Nếu hàng đợi chưa đầy, dữ liệu được ghi vào vị trí Head, sau đó tăng chỉ số Head.



Hình 4.3: Lưu đồ thuật toán xử lý lệnh từ PC

4.3.4.4. Nguyên lý bộ đệm vòng

Hàng đợi được quản lý bởi hai biến chỉ số:

- head: Vị trí sẽ ghi lệnh mới vào.
- tail: Vị trí lệnh đang được robot lấy ra để thực thi.

Quy tắc hoạt động:

1. Thêm lệnh (PC gửi xuống): Ghi vào buffer[head] và tăng head. Nếu head đuổi kịp tail, hàng đợi đầy (Full).
2. Thực thi lệnh (Robot chạy): Trong vòng lặp chính, kiểm tra nếu head != tail (hàng đợi có dữ liệu) và robot đang rảnh (Idle), hệ thống sẽ lấy lệnh tại buffer[tail], nạp vào bộ tạo xung và tăng tail.

4.3.4.5. Ý nghĩa trong hệ thống

Việc thiết kế tách biệt giữa nhận lệnh và thực thi lệnh thông qua hàng đợi giúp hệ thống đạt được các lợi ích:

- Đa nhiệm: Vì điều khiển có thể xử lý việc nhận dữ liệu USB ngay cả khi động cơ đang quay.
- Ôn định: Robot di chuyển liên tục theo quỹ đạo được quy hoạch trước mà không bị phụ thuộc vào độ trễ truyền thông của máy tính.
- An toàn: Dữ liệu được kiểm tra kỹ lưỡng trước khi vào hàng đợi, ngăn chặn các lệnh sai gây hỏng phần cứng.

4.3.5. Xử lý ngắt phát xung điều khiển động cơ

4.3.5.1. Giới thiệu

Đây là thành phần cốt lõi trong firmware của bộ điều khiển robot, chịu trách nhiệm chuyển đổi các thông số quỹ đạo thành các tín hiệu điện để điều khiển chính xác 3 động cơ bước theo thời gian thực. Hệ thống sử dụng Timer phần cứng TIM3 hoạt động ở chế độ ngắt chu kỳ với tần số 80kHz (tương đương chu kỳ ngắt $T = 12.5\mu s$).

4.3.5.2. Giải thuật phát xung DDS (Direct Digital Synthesis)

Để đảm bảo robot di chuyển mượt mà với vận tốc chính xác, hệ thống không sử dụng các vòng lặp delay truyền thống (vốn gây lãng phí CPU và khó đồng bộ nhiều trực). Thay vào đó, thuật toán DDS (hay còn gọi là thuật toán Bresenham mở rộng cho vận tốc) được áp dụng.

- Nguyên lý hoạt động:

Mỗi động cơ được quản lý bởi một biến tích lũy 32-bit (accumulator) và một giá trị gia tốc (speed_addend) đại diện cho vận tốc góc mong muốn.

Tại mỗi chu kỳ ngắt Timer (12.5μs), hệ thống thực hiện phép tính:

$$Accumulator_{new} = Accumulator_{old} + Speed_{Addend}$$

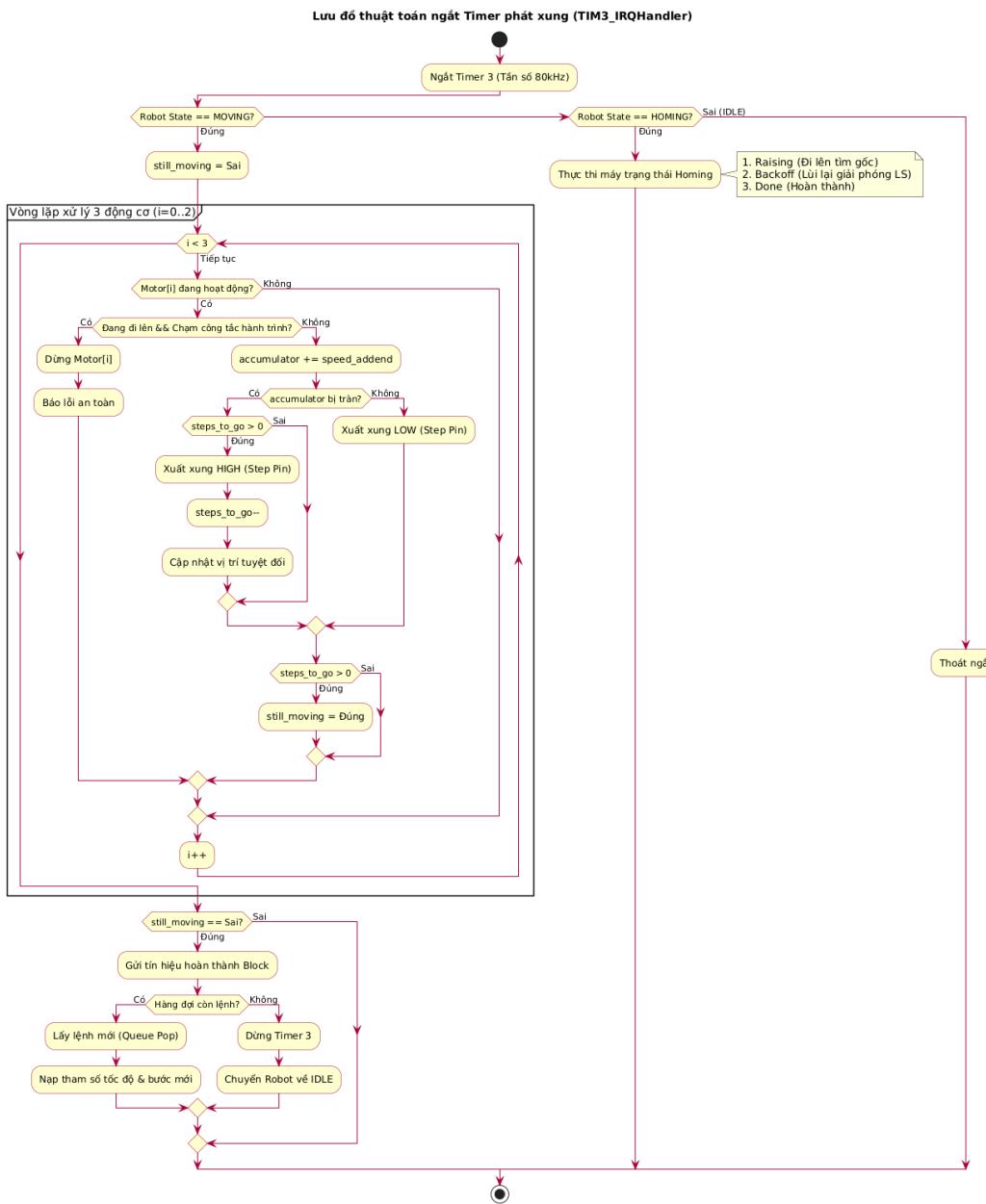
- Nếu biến accumulator bị tràn (tức giá trị quay vòng từ lớn nhất về nhỏ):
- Một xung điều khiển mức cao được gửi ra chân STEP.
- Số bước cần đi (steps_to_go) giảm đi 1.
- Nếu không tràn: Chân STEP được giữ hoặc kéo về mức thấp.

Giải thuật này cho phép tạo ra xung điều khiển với độ phân giải thời gian rất cao, giúp khử triệt để hiện tượng rung động (jitter) khi động cơ chạy ở tốc độ thấp.

4.3.5.3. Cấu trúc bộ điều khiển trong ngắt

Hàm xử lý ngắt TIM3_IRQHandler thực hiện các tác vụ điều khiển tuần tự:

1. Kiểm tra trạng thái hệ thống: Ngắt chỉ thực thi logic điều khiển khi robot ở trạng thái MOVING hoặc HOMING.
2. Vòng lặp điều khiển đa trực: Quét lần lượt qua 3 động cơ để xử lý độc lập.
3. Giám sát an toàn: Trước khi phát xung, hệ thống đọc trạng thái các công tắc hành trình. Nếu phát hiện robot đang di chuyển lên mà chạm công tắc, động cơ tương ứng sẽ bị dừng khẩn cấp.
4. Phát xung: Thực thi thuật toán DDS để quyết định việc đảo trạng thái chân GPIO.
5. Quản lý chuyển tiếp lệnh: Khi tất cả động cơ hoàn thành số bước của lệnh hiện tại, hệ thống tự động lấy lệnh tiếp theo từ hàng đợi. Việc nạp lệnh mới diễn ra ngay trong ngắt, đảm bảo robot chuyển động liên tục mà không bị dừng giữa các đoạn đường.



Hình 4.4: Lưu đồ thuật toán phát xung điều khiển động cơ

4.3.6. Quy trình xác định điểm gốc hệ tọa độ

Quy trình xác định điểm gốc là bước bắt buộc ngay sau khi khởi động hệ thống hoặc sau khi xảy ra sự cố dừng khẩn cấp. Mục đích của quy trình này là đồng bộ hóa vị trí vật lý của ba cánh tay robot với hệ tọa độ tính toán trong phần mềm.

Lưu đồ quy trình xác định điểm gốc



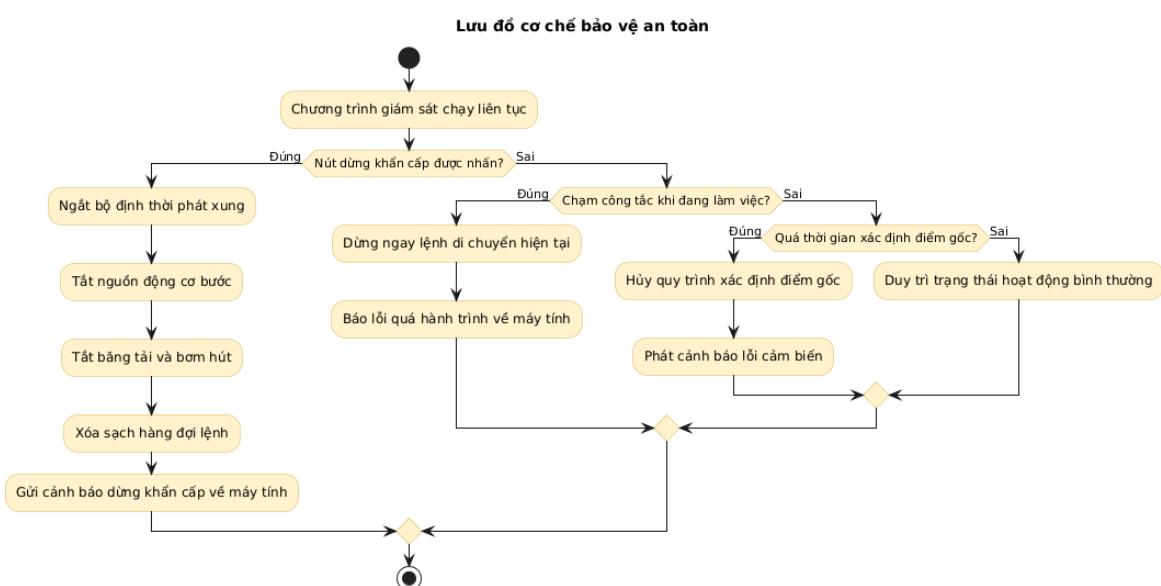
Hình 4.5: Quy trình về home

Quy trình bắt đầu bằng việc kích hoạt các động cơ quay theo chiều nâng trực lên phía trên. Hệ thống liên tục kiểm tra tín hiệu từ ba công tắc hành trình. Khi tất cả các trục đã tiếp xúc với công tắc, robot sẽ dừng lại và chuyển sang giai đoạn lùi trực. Vì để giảm thời gian về HOME, tốc độ nâng lên khác nhanh, nên khi chạm vào công tắc hành trình bị quá và không chính xác, vì vậy việc lùi trực với tốc độ chậm đảm bảo robot ở đún vị trí home khi công tắc vừa nhả ra. Khi công tắc đã trở về trạng thái hở, bộ định thời sẽ bị ngắt và vị trí này chính thức được coi là mốc tọa độ không của hệ thống.

4.3.7. Các cơ chế bảo vệ và an toàn hệ thống

Để đảm bảo robot hoạt động ổn định và tránh hỏng hóc cơ khí, phần mềm điều khiển được tích hợp các lớp bảo vệ đa tầng.

Lưu đồ cơ chế dừng khẩn cấp và bảo vệ:

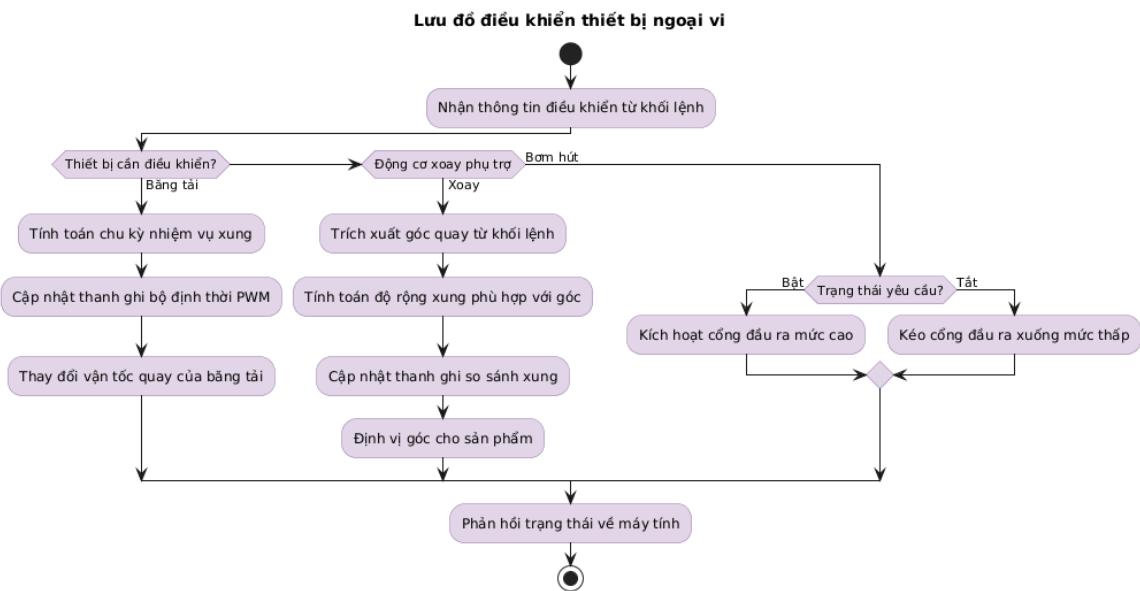


Hình 4.6: Lưu đồ cơ chế dừng an toàn của hệ thống

Hệ thống giám sát an toàn hoạt động dựa trên thứ tự ưu tiên. Mức ưu tiên cao nhất thuộc về nút dừng khẩn cấp vật lý; khi được kích hoạt, toàn bộ cơ cấu chấp hành và bộ định thời sẽ bị ngắt điện ngay lập tức để đảm bảo an toàn cho người và máy. Mức ưu tiên thứ hai là bảo vệ quá hành trình, ngăn chặn cánh tay robot va đập vào khung cơ khí khi có sai số tính toán. Cuối cùng là bộ giám sát thời gian, giúp phát hiện các sự cố tiềm ẩn như đứt dây cảm biến hoặc kẹt động cơ trong quá trình tìm điểm gốc.

4.3.8. Điều khiển các thiết bị ngoại vi

Vì điều khiển quản lý đồng bộ các thiết bị hỗ trợ để hoàn thành chu trình phân loại sản phẩm một cách tự động.



Hình 4.7: Lưu đồ thuật toán điều khiển ngoại vi

Việc điều khiển thiết bị ngoại vi được phân chia theo đặc tính kỹ thuật của từng loại:

- Đối với băng tải và động cơ xoay sản phẩm, hệ thống sử dụng phương pháp điều chế độ rộng xung thông qua các bộ định thời phần cứng độc lập.
- Riêng với động cơ xoay, phần mềm thực hiện thêm bước tính toán để chuyển đổi giá trị góc quay từ lệnh máy tính sang độ rộng xung tương ứng trong dải micro giây.
- Bơm hút sản phẩm có cấu tạo đơn giản hơn nên được điều khiển trực tiếp ON/OFF qua chân đầu ra của STM32

Toàn bộ các thao tác này được nạp sẵn và thực thi đồng bộ cùng với các khôi lệnh di chuyển chính của robot.

4.4. Xây dựng chương trình điều khiển trên máy tính

Phần mềm điều khiển trên máy tính đóng vai trò là trung tâm xử lý trí tuệ của toàn bộ hệ thống. Nhiệm vụ chính của phần mềm là thu nhận hình ảnh từ camera, thực hiện các thuật toán nhận diện vật thể sử dụng trí tuệ nhân tạo, tính toán động học nghịch và gửi lệnh điều khiển xuống vi điều khiển để vận hành cánh tay robot.

4.4.1. Môi trường phát triển và công cụ

Để đáp ứng yêu cầu về tốc độ xử lý ảnh và khả năng tính toán phức tạp, nhóm thực hiện đã cân nhắc kỹ lưỡng và quyết định lựa chọn các công cụ phát triển phù hợp.

Về ngôn ngữ lập trình, Python phiên bản 3.11 trở lên được lựa chọn làm ngôn ngữ chủ đạo. Lý do chính cho quyết định này nằm ở sự phổ biến vượt trội của Python trong lĩnh vực Trí tuệ nhân tạo và Thị giác máy tính. Python sở hữu một hệ sinh thái thư viện khổng lồ, cộng đồng hỗ trợ lớn và khả năng phát triển ứng dụng nhanh chóng. Mặc dù là ngôn ngữ thông dịch, nhưng với sự hỗ trợ của các công cụ tăng tốc hiện đại, Python hoàn toàn có thể đáp ứng tốt các bài toán thời gian thực trong phạm vi đồ án.

Môi trường soạn thảo mã nguồn được sử dụng là Visual Studio Code, một công cụ nhẹ, linh hoạt và hỗ trợ tốt việc quản lý dự án cũng như gỡ lỗi.

Các thư viện nòng cốt được sử dụng trong dự án bao gồm:

- PyQt6 là thư viện dùng để xây dựng giao diện người dùng đồ họa hiện đại. Thư viện này hỗ trợ cơ chế xử lý tín hiệu và khe (Signal & Slot), giúp tách biệt luồng giao diện và luồng xử lý, đảm bảo ứng dụng hoạt động mượt mà không bị treo khi thực hiện các tác vụ nặng.
- OpenCV và Ultralytics là hai thư viện quan trọng nhất cho nhiệm vụ thị giác máy tính. OpenCV đảm nhiệm việc đọc, tiền xử lý hình ảnh và các phép biến đổi hình học. Trong khi đó, Ultralytics cung cấp nền tảng tối ưu để triển khai mô hình học sâu YOLOv8, giúp việc nhận diện vật thể đạt độ chính xác cao và tốc độ nhanh.
- NumPy và Numba đóng vai trò nền tảng cho các tính toán toán học. NumPy hỗ trợ các phép toán ma trận hiệu năng cao cần thiết cho xử lý ảnh. Đặc biệt, thư viện Numba được sử dụng để biên dịch các hàm tính toán động học nghịch sang mã máy ngay tại thời điểm chạy. Điều này giúp tốc độ xử lý các phép toán lượng giác phức tạp tăng lên đáng kể, đạt mức hiệu năng tương đương với ngôn ngữ C/C++.
- PySerial là thư viện quản lý giao tiếp nối tiếp, đóng vai trò cầu nối truyền nhận dữ liệu lệnh và phản hồi giữa máy tính và vi điều khiển STM32 thông qua cổng USB.

4.4.2. Cấu trúc tổng thể chương trình

Chương trình được thiết kế theo tư duy hướng đối tượng và kiến trúc mô-đun hóa. Việc chia tách mã nguồn thành các tệp chức năng riêng biệt giúp mã nguồn dễ dàng quản lý, bảo trì và mở rộng các tính năng mới sau này. Cấu trúc mã nguồn trong thư mục dự án được tổ chức thành ba tầng xử lý chính.

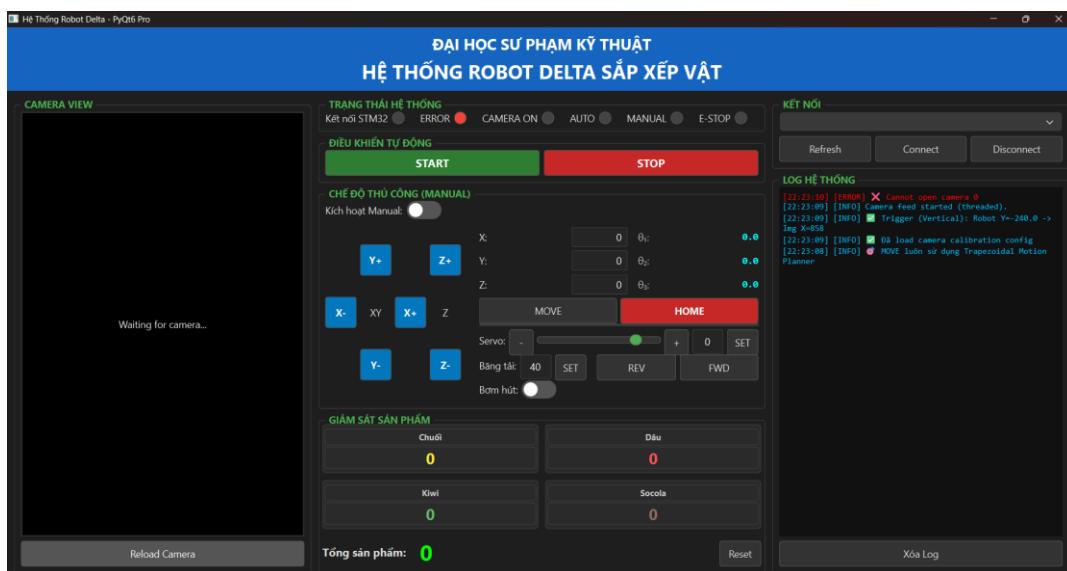
- Tầng ứng dụng và giao diện bao gồm tệp khởi chạy main.py và tệp giao diện pyqt_delta_gui.py. Tệp main.py chịu trách nhiệm khởi tạo toàn bộ hệ thống, thiết lập các luồng xử lý song song và kích hoạt vòng lặp sự kiện. Tệp

pyqt_delta_gui.py chứa mã nguồn thiết kế các thành phần trực quan như nút bấm, khung hiển thị camera, ô nhập liệu thông số và bảng hiển thị trạng thái, giúp người dùng tương tác dễ dàng với hệ thống.

- Tầng xử lý và tính toán bao gồm các mô-đun vision_system.py, deep_learning_processor.py và kinematics.py. Các mô-đun xử lý ảnh quản lý kết nối với camera công nghiệp, chạy luồng thu nhận hình ảnh riêng biệt, thực thi mô hình nhận diện YOLOv8 và thực hiện các phép biến đổi tọa độ từ ảnh sang không gian thực. Mô-đun kinematics.py chứa các hàm toán học thuần túy để giải bài toán động học nghịch, áp dụng kỹ thuật tối ưu hóa để đảm bảo các phép tính được thực hiện trong thời gian ngắn nhất (dưới 1 mili-giây).
- Tầng điều khiển giao tiếp bao gồm robot_controller.py, auto_mode_controller.py và connection_manager.py. Trong đó, robot_controller.py đóng vai trò là bộ não trung tâm, điều phối luồng dữ liệu từ hệ thống thị giác sang bộ phận điều khiển động cơ. Mô-đun auto_mode_controller.py chứa logic điều khiển riêng cho chế độ tự động, quản lý hàng đợi các vật thể cần gấp và tính toán thời điểm đồng bộ hóa. Cuối cùng, connection_manager.py là lớp vật lý xử lý việc mở, đóng cổng kết nối và truyền nhận các gói tin dữ liệu xuống vi điều khiển STM32 một cách tin cậy.

4.4.3. Thiết kế giao diện người dùng

Giao diện người dùng được xây dựng dựa trên thư viện PyQt6, áp dụng kiến trúc đa luồng để đảm bảo hiệu năng. Luồng chính của ứng dụng chỉ tập trung vào việc cập nhật hình ảnh và phản hồi thao tác của người dùng, trong khi các tác vụ nặng như xử lý ảnh hay giao tiếp dữ liệu được đẩy xuống các luồng phụ chạy nền.



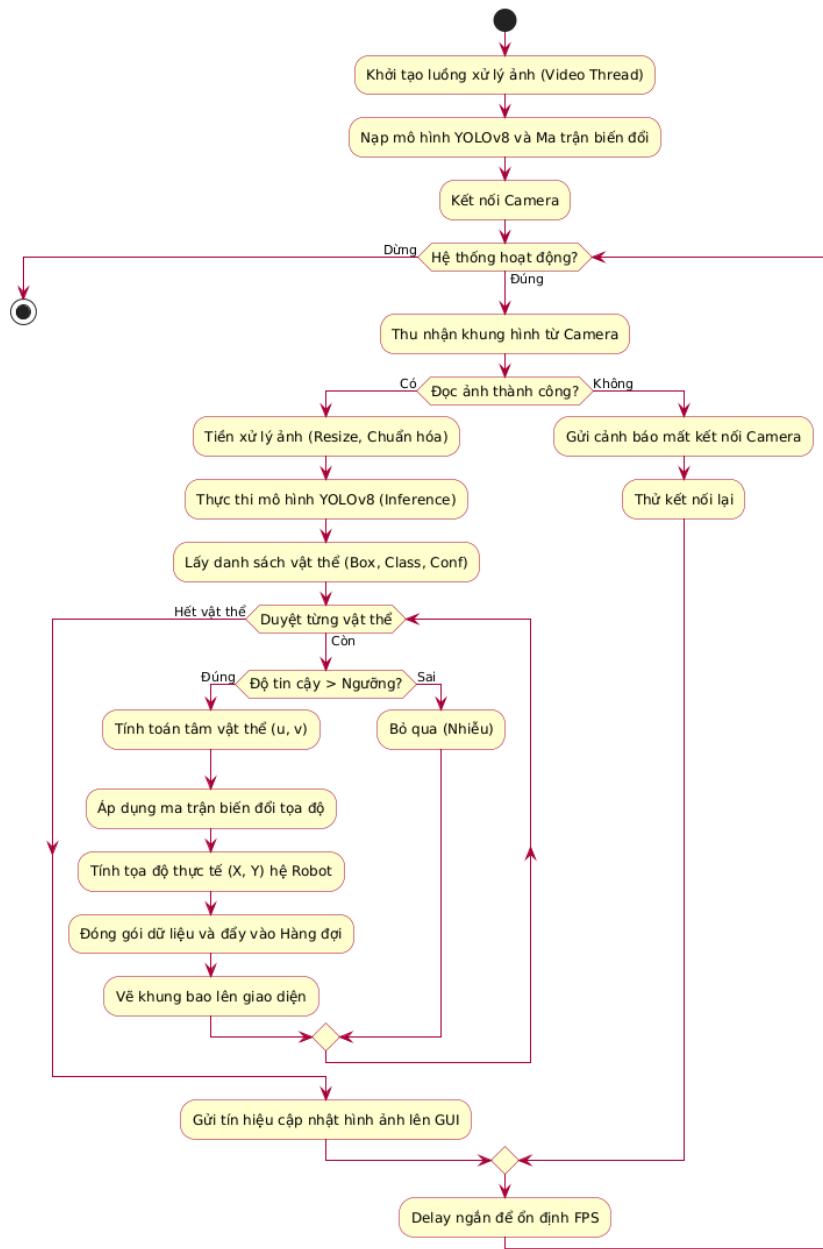
Hình 4.8: Giao diện điều khiển trên máy tính

Giao diện phần mềm cung cấp đầy đủ các công cụ giám sát và điều khiển cần thiết cho quá trình vận hành.

- Khu vực hiển thị trực quan nằm ở trung tâm, hiển thị video thời gian thực từ camera với các khung bao quanh vật thể đã được nhận diện, kèm theo thông tin về loại vật thể và độ tin cậy của thuật toán.
- Bảng điều khiển vận hành bao gồm các nút chức năng cơ bản như Khởi động, Dừng, Về gốc (Homing) và các nút điều khiển tinh chỉnh vị trí robot theo từng trực (Jogging), giúp người dùng dễ dàng thao tác kiểm tra máy.
- Khu vực cài đặt thông số cho phép người dùng cấu hình các tham số quan trọng như cổng kết nối COM, tọa độ làm việc giới hạn và vận tốc băng tải thực tế.
- Bảng giám sát trạng thái hiển thị liên tục tọa độ hiện tại của đầu công tác robot (X, Y, Z), trạng thái kết nối của các thiết bị phần cứng và nhật ký hoạt động chi tiết của hệ thống để phục vụ việc theo dõi và gỡ lỗi.

4.4.4. Mô-đun thị giác máy tính và nhận diện

Mô-đun thị giác máy tính được xây dựng dựa trên tệp mã nguồn vision_system.py, đóng vai trò là đôi mắt của hệ thống robot. Để đảm bảo khả năng xử lý thời gian thực và độ chính xác cao, quy trình xử lý ảnh được thiết kế theo một lưu đồ thuật toán chặt chẽ, hoạt động trên một luồng riêng biệt song song với giao diện chính.



Hình 4.9: Lưu đồ xử lý thị giác của hệ thống

Quy trình thuật toán được mô tả chi tiết qua các bước sau:

- Bước khởi tạo và kết nối: Khi hệ thống bắt đầu, chương trình sẽ khởi tạo một luồng xử lý video riêng biệt. Tại đây, mô hình trí tuệ nhân tạo YOLOv8 và các ma trận biến đổi tọa độ (được lưu từ quá trình hiệu chuẩn trước đó) sẽ được nạp vào bộ nhớ RAM. Sau đó, hệ thống tiến hành kết nối với camera công nghiệp thông qua giao thức tối ưu để đảm bảo độ trễ thấp nhất.
- Vòng lặp thu nhận và xử lý: Hệ thống đi vào một vòng lặp vô hạn để liên tục thu nhận các khung hình từ camera. Mỗi bức ảnh thu được sẽ trải qua bước tiền xử lý để điều chỉnh kích thước và chuẩn hóa màu sắc phù hợp với đầu vào của mô hình học sâu.

- Giai đoạn nhận diện: Bức ảnh sau khi xử lý được đưa qua mạng nơ-ron tích chập YOLOv8. Kết quả đầu ra là danh sách các vật thể phát hiện được, bao gồm tọa độ khung bao hình chữ nhật, loại vật thể và độ tin cậy của dự đoán. Để loại bỏ nhiễu và các nhận diện sai, thuật toán sẽ lọc bỏ tất cả các kết quả có độ tin cậy thấp hơn một ngưỡng cài đặt trước (ví dụ 70%).

```
def process_frame(self, frame, draw=True):
    # Cắt ảnh (ROI) để tối ưu tốc độ xử lý
    crop_size = 640
    x_start = max(0, frame.shape[1] - crop_size)
    crop_frame = frame[:, x_start:]

    # Thực thi mô hình (Inference)
    results = self.model.predict(crop_frame, verbose=False, conf=0.7)
    result = results[0]

    detections = []
    if result.obb is not None: # Xử lý Oriented Bounding Box
        for obb in result.obb:
            # Lấy thông tin tọa độ, góc xoay, loại vật thể
            r = obb.xywhr[0].cpu().numpy()
            cx, cy, w, h, rot_rad = r

            # Chuyển đổi góc radian sang độ
            angle_deg = np.degrees(rot_rad)

            detection = {
                'bbox': [int(cx), int(cy), int(w), int(h)],
                'angle': angle_deg,
                'class_name': result.names[int(obb.cls)],
                'confidence': float(obb.conf)
            }
            detections.append(detection)

    return frame, detections
```

- Giai đoạn chuyển đổi tọa độ: Đây là bước quan trọng nhất để robot có thể tương tác với vật thể. Tọa độ tâm của vật thể trên ảnh (đơn vị điểm ảnh) sẽ được nhân với ma trận biến đổi đồng nhất (Homography Matrix). Phép toán đại số tuyến tính này sẽ ánh xạ vị trí từ không gian ảnh 2D sang không gian làm việc thực tế của robot (đơn vị mm), giúp robot xác định chính xác vị trí gấp trên băng tải.

```
def pixel_to_mm(self, pixel_x, pixel_y):
    if self._pixel_to_mm_matrix is None:
        return None

    # Biểu diễn điểm dưới dạng vector đồng nhất [x, y, 1]
```

```
point = np.array([pixel_x, pixel_y, 1.0], dtype=np.float64)

# Nhân với ma trận biến đổi
result = self._pixel_to_mm_matrix @ point

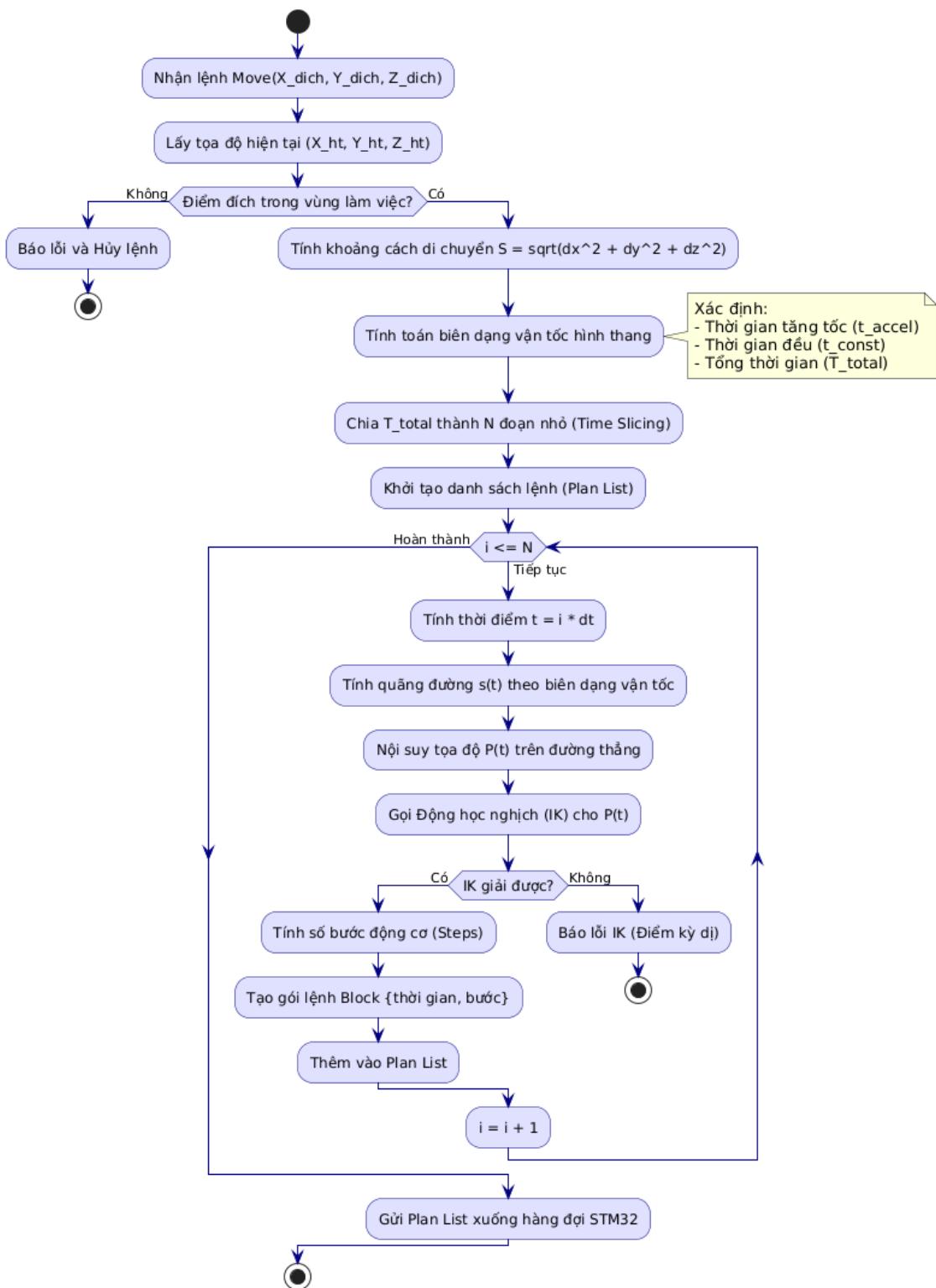
# Chuẩn hóa vector kết quả
robot_x = result[0] / result[2]
robot_y = result[1] / result[2]

return (robot_x, robot_y)
```

- Đóng gói và hiển thị: Cuối cùng, dữ liệu tọa độ thực tế cùng với loại vật thể sẽ được đóng gói và đẩy vào một hàng đợi an toàn để chuyển sang mô-đun điều khiển robot. Đồng thời, hệ thống sẽ vẽ các khung bao và thông tin lên hình ảnh gốc rồi gửi tín hiệu cập nhật lên giao diện người dùng, giúp người vận hành giám sát trực quan quá trình hoạt động.

4.4.5. Mô-đun tính toán động học và quy hoạch quỹ đạo

Mô-đun động học và quy hoạch quỹ đạo đóng vai trò quyết định đến độ chính xác và độ mượt mà trong chuyển động của robot. Để giải quyết bài toán gấp và đặt, robot Delta di chuyển qua các điểm trong không gian, hệ thống không gửi trực tiếp tọa độ đích mà thực hiện một quy trình tính toán phức tạp gồm nhiều bước.



Hình 4.10: Lưu đồ thuật toán quy hoạch quỹ đạo di chuyển

Quy trình xử lý chi tiết:

Đầu tiên là bước Tiếp nhận và Kiểm tra hợp lệ. Khi nhận được yêu cầu di chuyển đến tọa độ đích từ bộ điều khiển trung tâm, hệ thống sẽ kiểm tra xem tọa độ này có nằm trong vùng làm việc an toàn của robot hay không. Nếu điểm đích nằm ngoài tầm với hoặc có nguy cơ va chạm, lệnh sẽ bị hủy ngay lập tức để bảo vệ phần cứng.

Tiếp theo là Tính toán biên dạng vận tốc. Hệ thống tính toán khoảng cách Euclidean giữa điểm hiện tại và điểm đích. Dựa trên các thông số gia tốc và vận tốc tối đa đã cài đặt, thuật toán sẽ xây dựng một biên dạng vận tốc hình thang. Biên dạng này xác định rõ thời gian cần thiết cho giai đoạn tăng tốc, giai đoạn di chuyển đều và giai đoạn giảm tốc, đảm bảo tổng thời gian di chuyển là tối ưu nhất.

Bước quan trọng nhất là Chia nhỏ quỹ đạo theo thời gian (Time Slicing). Tổng thời gian di chuyển sẽ được chia thành hàng trăm lát cắt nhỏ, mỗi lát cắt tương ứng với một khoảng thời gian cố định (ví dụ 15 mili-giây).

Tại mỗi lát cắt thời gian, hệ thống thực hiện vòng lặp tính toán nội suy. Dựa vào biên dạng vận tốc, chương trình xác định quãng đường $s(t)$ mà robot đã đi được tại thời điểm t . Từ đó, tọa độ không gian tức thời $P(t)$ được nội suy tuyến tính nằm trên đường thẳng nối điểm đầu và điểm cuối.

Ngay sau đó, hàm Động học nghịch (được tối ưu hóa bằng Numba) sẽ được gọi để chuyển đổi tọa độ $P(t)$ thành góc quay của ba khớp động cơ. Kết quả là số bước xung cụ thể cần phát cho từng động cơ trong khoảng thời gian 15ms đó. Các dữ liệu này được đóng gói thành một cấu trúc Block lệnh.

Cuối cùng, toàn bộ danh sách các Block lệnh sẽ được đẩy xuống hàng đợi truyền thông để gửi dàn xuống vi điều khiển STM32, giúp robot thực thi chuyển động một cách liên tục và mượt mà.

```
# Trích đoạn logic từ lớp MotionPlannerTrapezoidal
def plan_cartesian_move_time_sliced(self, start_pos, end_pos, ...):
    # 1. Tính biên dạng vận tốc hình thang
    params = self._compute_profile_parameters(distance, v_max, accel)

    # 2. Chia nhỏ quỹ đạo thành các lát cắt
    num_slices = int(params["total_time"] / self.SEGMENT_TIME)

    plan = []
    for i in range(1, num_slices + 1):
        t_current = i * self.SEGMENT_TIME

        # 3. Tính quãng đường đi được tại thời điểm t
        dist_at_t = self._calculate_position_at_t(t_current, params)

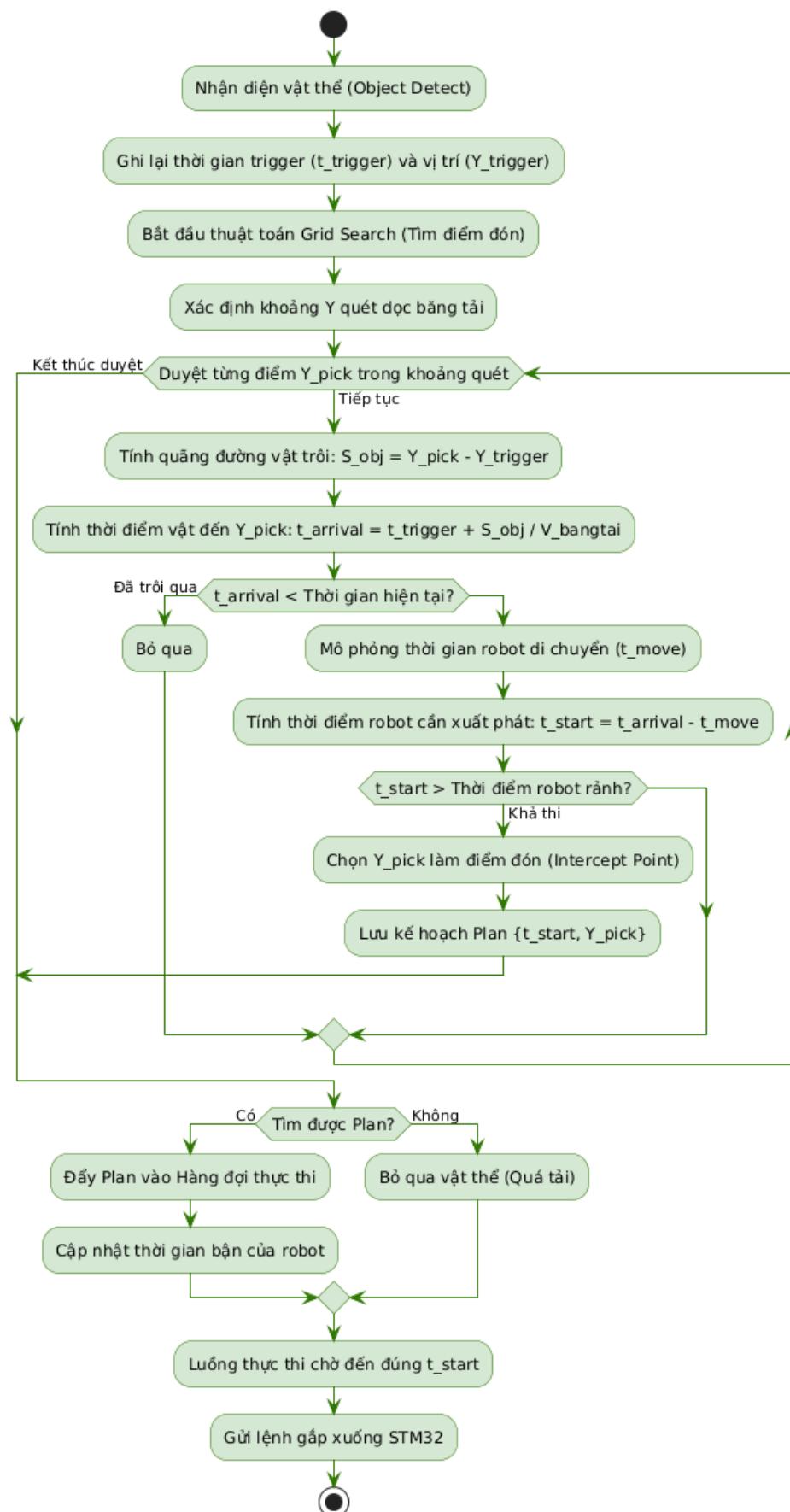
        # 4. Nội suy tọa độ (x, y, z) hiện tại trên đường thẳng
        ratio = dist_at_t / total_distance
        current_x = start_x + (end_x - start_x) * ratio

        # 5. Tính động học nghịch và tạo lệnh điều khiển
        angles = kinematics.inverse_kinematics(current_x, ... )
        plan.append({"t": segment_time, "s": angles_to_steps(angles)})
```

```
return plan
```

4.4.6. Chiến lược điều khiển và Logic tự động

Chiến lược điều khiển tự động là phần phức tạp nhất của hệ thống, được cài đặt trong tệp `auto_mode_controller.py`. Hệ thống sử dụng kỹ thuật đồng bộ hóa dựa trên thời gian để giải quyết bài toán bắt vật thể động trên băng tải mà không cần sử dụng encoder phản hồi vị trí băng tải.



Hình 4.11: Lưu đồ thuật toán lập lịch gấp và đồng bộ hoá

Công nghệ cốt lõi và Quy trình xử lý:

1. Định thời chính xác:

Thay vì xử lý tức thời ngay khi camera phát hiện vật, hệ thống ghi lại chính xác thời điểm xuất hiện (`t_trigger`) sử dụng đồng hồ đơn điệu `time.perf_counter()`. Vị trí của vật thể tại bất kỳ thời điểm tương lai nào đều được dự đoán dựa trên công thức:

$$Y(t) = Y_{\text{trigger}} + V_{\text{băng tải}} * (t - t_{\text{trigger}})$$

Điều này cho phép hệ thống biết trước vị trí của vật thể ngay cả khi vật đã trôi ra khỏi tầm nhìn của camera.

2. Thuật toán Tìm kiếm điểm đón:

Để robot bắt kịp vật thể, bộ điều khiển phải giải phương trình: *Thời điểm robot đến điểm P = Thời điểm vật thể đến điểm P*.

Hệ thống sử dụng thuật toán quét lưới dọc theo chiều dài băng tải (từ y = {-95:75} là vùng mà robot hoạt động ổn định nhất). Tại mỗi điểm ứng viên, chương trình mô phỏng ngược thời gian di chuyển của robot (sử dụng Motion Planner) để kiểm tra xem robot có kịp di chuyển từ vị trí hiện tại đến đó hay không. Điểm khả thi đầu tiên tìm được sẽ được chọn làm điểm đón

3. Lập lịch thông minh:

Hệ thống duy trì một biến trạng thái `robot_next_free_perf` để biết khi nào robot sẽ hoàn thành tác vụ hiện tại. Khi có vật thể mới, thuật toán sẽ tính toán thời điểm xuất phát `t_start` sao cho `t_start >= robot_next_free_perf`. Điều này đảm bảo robot hoạt động liên tục, gói đầu các tác vụ lên nhau mà không bị xung đột, tối ưu hóa năng suất gấp.

4. Bù trừ độ trễ:

Trong môi trường thực tế, luôn có độ trễ do xử lý ảnh, truyền tin USB và quán tính cơ khí. Hệ thống tích hợp một tham số bù trễ (khoảng 100-200ms) vào công thức tính toán thời gian xuất phát, đảm bảo đầu hút đến điểm gấp sớm hơn một chút để đón đầu vật thể chính xác.

Đoạn mã minh họa thuật toán lập lịch:

```
```python
Trích đoạn logic từ AutoModeController
def _schedule_pick_candidates(self):
 # Quét dọc theo trục Y để tìm điểm gấp khả thi
 y_range = range(self.y_pick_min, self.y_pick_max, self.y_pick_step)

 for pick_y in y_range:
```

```
1. Tính thời gian vật thể trôi đến điểm pick_y
dist_obj_travel = pick_y - trigger_y
time_travel = abs(dist_obj_travel) / self.conveyor_speed_mm_s
t_arrival = t_trigger + time_travel # Thời điểm Gặp Nhau

2. Mô phỏng thời gian robot di chuyển
t_robot_move = self._simulate_robot_move_time(current_pos, (robot_x, pick_y,
z_pick))

3. Tính thời điểm robot cần xuất phát (có bù trễ)
t_required_start = t_arrival - t_robot_move - LATENCY_COMPENSATION

4. Kiểm tra tính khả thi: Robot có kịp xuất phát không?
if t_required_start >= self.robot_next_free_perf:
 # Tìm thấy phương án khả thi!
 best_plan = {
 'pick_y': pick_y,
 't_start_action': t_required_start,
 't_arrival': t_arrival
 }
 # Cập nhật thời gian bận của robot cho vật tiếp theo
 self.robot_next_free_perf = t_arrival + estimate_drop_time()
 break
```

#### 4.4.7. Quản lý trạng thái và xử lý lỗi

Để đảm bảo hệ thống hoạt động ổn định và tin cậy trong môi trường công nghiệp, phần mềm tích hợp các cơ chế quản lý trạng thái và xử lý lỗi đa tầng.

##### Cơ chế điều khiển luồng cửa sổ trượt:

Do bộ nhớ đệm trên vi điều khiển stm32f103c8t6 có hạn, máy tính không thể gửi toàn bộ hàng trăm lệnh di chuyển của một quỹ đạo dài xuống cùng lúc. Để giải quyết vấn đề này, phần mềm sử dụng cơ chế cửa sổ trượt để đồng bộ hóa dữ liệu. Phần mềm trên máy tính duy trì một bộ đệm lệnh ảo. Khi bắt đầu di chuyển, máy tính sẽ gửi trước một gói lệnh gồm khoảng 16 đến 32 lệnh xuống vi điều khiển. Sau đó, mỗi khi vi điều khiển thực hiện xong một lệnh, nó sẽ gửi tín hiệu phản hồi ngược về máy tính. Nhận được tín hiệu này, phần mềm sẽ lập tức gửi bổ sung một lệnh tiếp theo từ hàng đợi xuống để lấp vào chỗ trống. Cơ chế này đảm bảo bộ đệm trên vi điều khiển luôn đầy ở mức an toàn, giúp robot di chuyển liên tục mà không bị ngắt quãng do chờ dữ liệu hoặc tràn bộ nhớ.

##### Giám sát kết nối và bộ định thời gian giám sát:

Hệ thống liên tục giám sát trạng thái kết nối giữa máy tính và các thiết bị ngoại vi. Nếu tín hiệu từ camera bị mất hoặc cổng usb bị ngắt kết nối đột ngột, phần mềm sẽ tự động kích hoạt quy trình xử lý lỗi gồm dừng băng tải, đưa robot về trạng thái an toàn và thử kết nối lại. Đối với các tác vụ quan trọng như về gốc tọa độ, hệ thống sử dụng một

bộ định thời giám sát. Nếu robot không hoàn thành việc về gốc trong thời gian quy định, phần mềm sẽ nhận định có sự cố cơ khí và lập tức dừng hệ thống để bảo vệ động cơ.

#### Các lớp bảo vệ an toàn:

Ngoài nút dừng khẩn cấp vật lý, phần mềm còn thiết lập các giới hạn mềm cho vùng làm việc. Trước khi gửi bất kỳ lệnh di chuyển nào, thuật toán sẽ kiểm tra xem tọa độ đích có nằm trong vùng làm việc an toàn hay không. Nếu phát hiện lệnh di chuyển ra ngoài giới hạn cho phép, phần mềm sẽ từ chối thực thi và cảnh báo người vận hành, ngăn ngừa nguy cơ va chạm cơ khí.

#### 4.4.8. Quy trình hiệu chuẩn hệ thống

Quy trình hiệu chuẩn là bước bắt buộc để đảm bảo sự đồng bộ giữa hệ thống thị giác và cơ cấu chấp hành của robot. Quy trình này bao gồm hai giai đoạn chính được thực hiện thông qua giao diện phần mềm.

Giai đoạn thứ nhất là hiệu chuẩn ống kính camera. Hình ảnh thu được từ camera thường bị biến dạng cong ở các góc do đặc tính của thấu kính. Phần mềm sử dụng các tham số hình học được tính toán trước để nắn chỉnh hình ảnh về dạng phẳng, đảm bảo tỉ lệ xích trên toàn bộ khung hình là đồng nhất.

Giai đoạn thứ hai là hiệu chuẩn không gian để thiết lập mối quan hệ giữa điểm ảnh và milimet thực tế. Người vận hành sẽ xác định bốn điểm chuẩn tương ứng với bốn góc của vùng làm việc trên băng tải. Tọa độ điểm ảnh của các điểm này được ghi nhận cùng với tọa độ thực tế mà robot có thể chạm tới. Dựa trên dữ liệu này, phần mềm sử dụng thuật toán ánh xạ đồng nhất để tạo ra ma trận biến đổi tọa độ. Ma trận này cho phép chuyển đổi tức thời vị trí tâm vật thể từ đơn vị điểm ảnh sang tọa độ không gian chính xác mà robot cần di chuyển tới để thực hiện thao tác gấp.

## CHƯƠNG 5: THỬ NGHIỆM ĐÁNH GIÁ KẾT QUẢ

Sau khi hoàn thành việc chế tạo mô hình và xây dựng chương trình điều khiển, nhóm thực hiện tiến hành các đợt thử nghiệm thực tế để đánh giá hiệu suất của hệ thống.

### 5.1. Thiết lập môi trường thực nghiệm

Quá trình thực nghiệm được thực hiện trên mô hình thực tế với các điều kiện cố định. Hệ thống camera được lắp ở độ cao 450 milimet so với mặt băng tải. Nguồn sáng được bố trí đồng đều để giảm thiểu bóng đổ gây nhiễu cho thuật toán nhận diện. Các sản phẩm thử nghiệm bao gồm bốn loại mô hình bánh lương khô với 4 vị là chuối, dâu, kiwi và socola. Băng tải được thiết lập vận hành ở dải tốc độ từ 30 đến 60 milimet trên giây để kiểm tra khả năng đáp ứng thời gian thực của hệ thống.

### 5.2. Đánh giá hệ thống nhận diện vật thể

Mô hình học sâu yolov8 được đánh giá dựa trên khả năng nhận diện chính xác loại vật thể và vị trí của chúng khi đang di chuyển trên băng tải.

Độ chính xác nhận diện:

Qua quá trình chạy thử với 100 mẫu vật khác nhau, mô hình đạt độ chính xác trung bình trên 90 phần trăm. Một số sai số nhỏ xảy ra khi các vật thể nằm quá sát nhau hoặc bị phản xạ ánh sáng mạnh từ bề mặt băng tải.

Tốc độ xử lý ảnh:

Nhờ việc tối ưu hóa mã nguồn và sử dụng luồng xử lý riêng, tốc độ nhận diện đạt mức 22 khung hình trên giây. Thời gian xử lý từ lúc camera thu nhận ảnh đến khi có tọa độ vật thể dao động trong khoảng 30 đến 50 mili-giây, hoàn toàn đáp ứng tốt yêu cầu điều khiển thời gian thực.

### 5.3. Đánh giá khả năng điều khiển và độ chính xác gấp th策

Khả năng điều khiển được đánh giá dựa trên sự phối hợp giữa quy hoạch quỹ đạo và đồng bộ hóa thời gian.

Độ chính xác vị trí:

Sai số vị trí tĩnh của đầu gấp sau khi về gốc tọa độ nằm trong khoảng dưới 3 milimet. Khi vận hành ở chế độ tự động, nhờ kỹ thuật nội suy biến dạng hình thang, đầu gấp di chuyển mượt mà và tiếp cận vật thể chính xác theo đường thẳng.

Mặc dù hộp số động cơ có độ rõ nhất định nhưng không ảnh hưởng đến độ chính xác robot

Hiệu quả đồng bộ hóa:

Thuật toán dự đoán điểm đón dựa trên thời gian thực tế hoạt động ổn định. Tại tốc độ băng tải 40 milimet trên giây, robot có thể xác định và gấp trúng vật thể với tỉ lệ thành công cao. Sai số xảy ra chủ yếu do quán tính của vật thể khi trượt trên băng tải hoặc sự thay đổi đột ngột của tốc độ động cơ băng tải.

#### **5.4. Đánh giá hiệu suất tổng thể hệ thống**

Hệ thống được vận hành liên tục để kiểm tra tính ổn định và năng suất phân loại.

##### **Năng suất hoạt động:**

Trong điều kiện vận hành tối ưu, hệ thống đạt năng suất gấp và phân loại khoảng 20 đến 25 sản phẩm trong một phút. Chu kỳ của một lần gấp thả bao gồm di chuyển đến điểm đón, hạ xuống gấp, nhấc lên, di chuyển đến khay chứa và quay về vị trí chờ mất khoảng 2,5 giây.

##### **Tỉ lệ phân loại đúng:**

Tỉ lệ gấp và thả đúng khay quy định đạt trên 85 phần trăm. Các trường hợp thất bại thường rơi vào tình huống vật thể nằm ngoài vùng làm việc an toàn của robot hoặc có quá nhiều vật thể xuất hiện cùng lúc trên băng tải dẫn đến quá tải bộ đệm lệnh.

Tổng kết lại, mô hình robot delta kết hợp xử lý ảnh học sâu đã hoạt động đúng theo mục tiêu đề ra, minh chứng được tính hiệu quả của các thuật toán quy hoạch quỹ đạo và đồng bộ hóa thời gian thực trong bài toán tự động hóa sản xuất.

## HƯỚNG PHÁT TRIỂN ĐỀ TÀI

Mặc dù hệ thống hiện tại đã đáp ứng được các mục tiêu cơ bản về phân loại sản phẩm sử dụng robot Delta kết hợp thị giác máy, nhưng để nâng cao hiệu suất và khả năng ứng dụng trong môi trường công nghiệp thực tế, nhóm thực hiện đề xuất các hướng phát triển trong tương lai như sau:

### 1. Nâng cấp về Thuật toán Điều khiển và Quy hoạch quỹ đạo

Tích hợp bộ điều khiển quỹ đạo S-Curve: Hiện tại hệ thống đang sử dụng biên dạng vận tốc hình thang. Trong tương lai, có thể áp dụng biên dạng S-Curve để kiểm soát độ giật, giúp robot di chuyển mượt mà hơn, giảm rung động cơ khí và tăng tuổi thọ thiết bị khi hoạt động ở tốc độ cao.

Tối ưu hóa bài toán lập lịch (Scheduling Optimization): Khi mật độ sản phẩm trên băng tải quá dày đặc, thuật toán hiện tại có thể bỏ sót vật thể. Cần nghiên cứu áp dụng các thuật toán tối ưu hóa (như giải thuật tham lam hoặc quy hoạch động) để robot chọn thứ tự gấp tối ưu nhất, giảm thiểu quãng đường di chuyển thừa.

### 2. Cải tiến hệ thống Thị giác máy và Trí tuệ nhân tạo

Tích hợp Camera 3D (Stereo Vision hoặc Depth Camera): Hệ thống 2D hiện tại gặp khó khăn khi xác định chiều cao vật thể hoặc khi các vật thể bị xếp chồng lên nhau. Việc sử dụng Camera 3D sẽ giúp robot xác định chính xác tọa độ (x, y, z) và hướng gấp trong không gian 3 chiều.

Kiểm tra chất lượng sản phẩm (QA/QC): Mở rộng mô hình học sâu (Deep Learning) không chỉ để phân loại bánh mà còn để phát hiện các lỗi ngoại quan như: bánh bị vỡ, cháy, méo mó hoặc thiếu tem nhãn, từ đó loại bỏ các sản phẩm không đạt chuẩn.

### 3. Tối ưu hóa Phần cứng và Hệ thống nhúng

Chuyển đổi sang hệ thống nhúng Edge AI: Thay vì phụ thuộc vào máy tính cá nhân (PC/Laptop), hệ thống có thể được tích hợp lên các bo mạch nhúng mạnh mẽ như NVIDIA Jetson Nano hoặc Jetson Orin. Điều này giúp thu gọn kích thước tủ điện, giảm chi phí và biến robot thành một thiết bị hoạt động độc lập.

Thiết kế đầu gấp linh hoạt (Soft Gripper): Nghiên cứu các loại tay gấp dạng mềm hoặc cơ cấu kẹp thích ứng để có thể gấp được đa dạng các loại sản phẩm có hình dáng phức tạp hoặc dễ vỡ mà không cần thay đầu hút.

### 4. Tích hợp IoT và Hệ thống giám sát (SCADA)

Xây dựng giao diện Web/App giám sát từ xa: Phát triển hệ thống IoT cho phép người quản lý theo dõi năng suất, trạng thái hoạt động và cảnh báo lỗi của robot thông qua thiết bị di động hoặc trình duyệt web.

Lưu trữ và phân tích dữ liệu: Tích hợp cơ sở dữ liệu (Database) để lưu lại lịch sử sản xuất, số lượng sản phẩm đạt/lỗi theo thời gian thực, phục vụ cho việc báo cáo và phân tích hiệu quả sản xuất.

## TÀI LIỆU THAM KHẢO

- [1] R. Clavel, "Device for the movement and positioning of an element in space," U.S. Patent 4 976 582, Dec. 11, 1990. (Đây là bằng sáng chế gốc của Robot Delta).
- [2] L. W. Tsai, *Robot Analysis: The Mechanics of Serial and Parallel Manipulators*. New York, NY, USA: Wiley, 1999
- [3] R. L. Williams II, "The Delta Parallel Robot: Kinematics Solutions," *Mechanical Engineering Faculty Publications*, vol. 18, 2016
- [4] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4th ed. New York, NY, USA: Pearson, 2018.
- [5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. (Sách gối đầu giường về Deep Learning/CNN).
- [6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, 2016, pp. 779–788.
- [8] STMicroelectronics, "STM32F103x8/B Datasheet - Medium-density performance line ARM-based 32-bit MCU," 2015. [Online]. Available: <https://www.st.com>.

## PHỤ LỤC 1: LẬP TRÌNH ĐỘNG HỌC

Chương trình trên file kinematics.py

```
import math
import numpy as np
from numba import jit
import constants as C

=====
NUMBA-OPTIMIZED CORE FUNCTIONS (20-100x faster)
=====
Numba không hỗ trợ methods với 'self', nên tách thành static functions
Các hằng số phải truyền vào như parameters

@jit(nopython=True, cache=True)
def _calc_angle_yz_numba(x0, y0, z0, tan30, f, e, rf, re, pi):
 """
 Numba-optimized: Tính góc theta cho 1 cánh tay.
 Tốc độ: ~50-100x nhanh hơn Python thuần.
 """

 # Hằng số f/2 * tan30
 y1 = -0.5 * tan30 * f

 # Trừ offset effector
 y0_mod = y0 - 0.5 * tan30 * e

 if abs(z0) < 1e-9:
 return np.nan # Tránh chia cho 0

 # z = a + b*y
 a = (x0**2 + y0_mod**2 + z0**2 + rf**2 - re**2 - y1**2) / (2.0 * z0)
 b = (y1 - y0_mod) / z0

 # discriminant
 d = -(a+b*y1)**2 + rf*(b*b*rf + rf)

 # Nếu d < 0, điểm không tồn tại
 if d < 1e-9:
 return np.nan

 # choosing outer point
 yj = (y1 - a * b - math.sqrt(d)) / (b**2 + 1.0)
 zj = a + b * yj

 # Ngăn chia cho 0 nếu yj == y1
 if abs(y1 - yj) < 1e-9:
 if zj < 0:
 theta_deg = -90.0
 else:
 theta_deg = 90.0
```

```
else:
 theta_rad = math.atan(-zj / (y1 - yj))
 theta_deg = 180.0 * theta_rad / pi

Xử lý góc phần tư
if yj > y1:
 theta_deg += 180.0

return theta_deg

@jit(nopython=True, cache=True)
def inverse_kinematics_numba(x0, y0, z0, tan30, f, e, rf, re, pi, sqrt3,
 cos120, sin120, arm_angle_min, arm_angle_max):
 """
 Numba-optimized: Tính động học nghịch cho TÂM BÊ.
 Trả về (theta1, theta2, theta3) hoặc (nan, nan, nan) nếu thất bại.
 """

 # Đảo dấu X để phù hợp với hệ tọa độ robot
 x0_corrected = x0
 y0_corrected = -y0

 # Cánh tay 1
 theta1 = _calc_angle_yz_numba(x0_corrected, y0_corrected, z0,
 tan30, f, e, rf, re, pi)

 # Cánh tay 2 (xoay -120 độ)
 x_rot2 = x0_corrected * cos120 + y0_corrected * sin120
 y_rot2 = y0_corrected * cos120 - x0_corrected * sin120
 theta2 = _calc_angle_yz_numba(x_rot2, y_rot2, z0,
 tan30, f, e, rf, re, pi)

 # Cánh tay 3 (xoay +120 độ)
 x_rot3 = x0_corrected * cos120 - y0_corrected * sin120
 y_rot3 = y0_corrected * cos120 + x0_corrected * sin120
 theta3 = _calc_angle_yz_numba(x_rot3, y_rot3, z0,
 tan30, f, e, rf, re, pi)

 # Kiểm tra thất bại
 if np.isnan(theta1) or np.isnan(theta2) or np.isnan(theta3):
 return np.nan, np.nan, np.nan

 # Kiểm tra giới hạn góc
 if not (arm_angle_min <= theta1 <= arm_angle_max):
 return np.nan, np.nan, np.nan
 if not (arm_angle_min <= theta2 <= arm_angle_max):
 return np.nan, np.nan, np.nan
 if not (arm_angle_min <= theta3 <= arm_angle_max):
 return np.nan, np.nan, np.nan
```

```
return theta1, theta2, theta3

@jit(nopython=True, cache=True)
def inverse_kinematics_tool_numba(xt, yt, zt, alpha_deg,
 tool_offset_radius, tool_offset_z,
 tan30, f, e, rf, re, pi, sqrt3,
 cos120, sin120, arm_angle_min,
 arm_angle_max):
 """
 Numba-optimized: Tính động học nghịch cho ĐẦU HÚT với tool offset.
 """
 alpha_rad = math.radians(alpha_deg)

 # Tính (x0, y0, z0) của tâm bệ Từ (xt, yt, zt) của đầu hút
 x0 = xt - tool_offset_radius * math.cos(alpha_rad)
 y0 = yt - tool_offset_radius * math.sin(alpha_rad)
 z0 = zt + tool_offset_z

 # Gọi hàm IK chuẩn
 return inverse_kinematics_numba(x0, y0, z0, tan30, f, e, rf, re, pi,
 sqrt3,
 cos120, sin120, arm_angle_min,
 arm_angle_max)

=====
DELTA KINEMATICS CLASS (Wrapper cho Numba functions)
=====

class DeltaKinematics:

 def __init__(self, e_side=None, f_side=None, re_lower=None, rf_upper=None,
 tool_offset_radius=15, tool_offset_z=53):
 """
 Khởi tạo với các thông số hình học của robot.
 Nếu không truyền tham số, sẽ lấy mặc định từ constants.py
 """
 # Lấy từ tham số hoặc constants
 self.e = e_side if e_side is not None else C.ROBOT_E_SIDE
 self.f = f_side if f_side is not None else C.ROBOT_F_SIDE
 self.re = re_lower if re_lower is not None else C.ROBOT_RE_LOWER
 self.rf = rf_upper if rf_upper is not None else C.ROBOT_RF_UPPER

 # Hằng số lượng giác
 self.sqrt3 = math.sqrt(3.0)
 self.pi = math.pi
 self.sin120 = self.sqrt3 / 2.0
 self.cos120 = -0.5
```

```
self.tan30 = 1.0 / self.sqrt3

Hằng số 't' từ C++ forward kinematics
self.t = (self.f - self.e) * self.tan30 / 2.0
self.dtr = self.pi / 180.0 # Độ sang Radian

Tool offset (QUAN TRỌNG)
self.tool_offset_radius = tool_offset_radius
self.tool_offset_z = tool_offset_z

Hằng số động cơ bước - Lấy từ constants.py để dễ calib
self.gear_ratio = C.MOTOR_GEAR_RATIO
self.steps_per_revolution = C.MOTOR_STEPS_PER_REV

Độ phân giải cuối cùng: số xung động cơ để quay cánh tay robot 1 độ.
self.steps_per_arm_degree = (self.steps_per_revolution *
self.gear_ratio) / 360.0

def warmup(self):
 """
 Kích hoạt JIT compilation (warm-up) cho các hàm Numba bằng cách gọi
 chúng với dữ liệu mẫu.
 Giúp tránh lag ở lần chạy đầu tiên.
 """
 # Dummy coordinates (gần vị trí Home)
 x, y, z = 0.0, 0.0, -380.0
 alpha = -90.0

 # Warmup inverse_kinematics_numba
 inverse_kinematics_numba(
 x, y, z,
 self.tan30, self.f, self.e, self.rf, self.re, self.pi, self.sqrt3,
 self.cos120, self.sin120, C.ARM_ANGLE_MIN, C.ARM_ANGLE_MAX
)

 # Warmup inverse_kinematics_tool_numba
 inverse_kinematics_tool_numba(
 x, y, z, alpha,
 self.tool_offset_radius, self.tool_offset_z,
 self.tan30, self.f, self.e, self.rf, self.re, self.pi, self.sqrt3,
 self.cos120, self.sin120, C.ARM_ANGLE_MIN, C.ARM_ANGLE_MAX
)

 # Warmup _calc_angle_yz_numba implicitly via above calls

 print("☑ DeltaKinematics: Numba functions warmed up!")

def forward_kinematics(self, theta1, theta2, theta3):
 """
 Tính toán tọa độ TÂM BỆ ĐỘNG (x0, y0, z0) từ các góc động cơ.
```

Logic dựa trên delta\_calcForward.

```
:return: (x0, y0, z0) hoặc None nếu không có nghiệm
"""
theta1_rad = theta1 * self.dtr
theta2_rad = theta2 * self.dtr
theta3_rad = theta3 * self.dtr

y1 = -(self.t + self.rf * math.cos(theta1_rad))
z1 = -self.rf * math.sin(theta1_rad)

y2 = (self.t + self.rf * math.cos(theta2_rad)) * 0.5 # 0.5 = sin30
x2 = y2 * self.sqrt3 # tan60
z2 = -self.rf * math.sin(theta2_rad)

y3 = (self.t + self.rf * math.cos(theta3_rad)) * 0.5
x3 = -y3 * self.sqrt3
z3 = -self.rf * math.sin(theta3_rad)

dnm = (y2 - y1) * x3 - (y3 - y1) * x2

if abs(dnm) < 1e-9:
 return None # Cấu hình kỳ dị

w1 = y1**2 + z1**2
w2 = x2**2 + y2**2 + z2**2
w3 = x3**2 + y3**2 + z3**2

x = (a1*z + b1)/dnm
a1 = (z2 - z1) * (y3 - y1) - (z3 - z1) * (y2 - y1)
b1 = -((w2 - w1) * (y3 - y1) - (w3 - w1) * (y2 - y1)) / 2.0

y = (a2*z + b2)/dnm;
a2 = -(z2 - z1) * x3 + (z3 - z1) * x2
b2 = ((w2 - w1) * x3 - (w3 - w1) * x2) / 2.0

a*z^2 + b*z + c = 0
a = a1**2 + a2**2 + dnm**2
b = 2.0 * (a1 * b1 + a2 * (b2 - y1 * dnm) - z1 * dnm * dnm)
c = (b2 - y1 * dnm)**2 + b1**2 + dnm**2 * (z1**2 - self.re**2)

discriminant
d = b**2 - 4.0 * a * c
if d < 0:
 return None # non-existing point

z0 = -0.5 * (b + math.sqrt(d)) / a
x0 = (a1 * z0 + b1) / dnm
y0 = (a2 * z0 + b2) / dnm
```

```

FIX: Đảo dấu X để phù hợp với hệ tọa độ robot
Update 28/11/2025: Xoay hệ tọa độ 180 độ (X -> -X, Y -> -Y)
User report: X+ bị ngược (ở bên trái) -> Đổi lại thành x0
return (x0, -y0, z0)

def forward_kinematics_tool(self, theta1, theta2, theta3, alpha_deg):
 """
 Tính toán tọa độ ĐẦU HÚT (xt, yt, zt) từ các góc động cơ
 VÀ góc servo. (HÀM MỚI)

 :param theta1, theta2, theta3: Góc 3 động cơ (độ)
 :param alpha_deg: Góc của servo (độ) dùng cho tool offset
 :return: (xt, yt, zt) của đầu hút, hoặc None nếu không có nghiệm
 """

1. Tính tọa độ TÂM BỆ DI ĐỘNG (effector)
effector_coords = self.forward_kinematics(theta1, theta2, theta3)

if effector_coords is None:
 # Không thể tính được vị trí tâm bệ
 return None

x0, y0, z0 = effector_coords

2. Từ tâm bệ, tính ra vị trí đầu hút (xt, yt, zt)
alpha_rad = math.radians(alpha_deg)

Tool offset theo alpha (góc quay của servo)
Quy ước servo: 0° → X+, 90° → Y+, 180° → X-, 270° → Y-
Update 02/12: Fix dấu Y - phải CỘNG cả X và Y
xt = x0 + self.tool_offset_radius * math.cos(alpha_rad)
yt = y0 + self.tool_offset_radius * math.sin(alpha_rad)

zt = z_effector - offset_z (vì tool thấp hơn)
zt = z0 - self.tool_offset_z

return (xt, yt, zt)

def _calc_angle_yz(self, x0, y0, z0):
 """
 Hàm trợ giúp tính góc theta cho 1 cánh tay.
 Tính cho TÂM BỆ (x0, y0, z0).
 """

 # Hằng số f/2 * tan30
 y1 = -0.5 * self.tan30 * self.f

 # Trừ offset effector
 y0_mod = y0 - 0.5 * self.tan30 * self.e

 if abs(z0) < 1e-9:

```

```

 return None # Tránh chia cho 0

 # z = a + b*y
 a = (x0**2 + y0_mod**2 + z0**2 + self.rf**2 - self.re**2 - y1**2) /
(2.0 * z0)
 b = (y1 - y0_mod) / z0

 # discriminant
 d = -(a+b*y1)**2 + self.rf*(b*b*self.rf + self.rf)

 # Nếu d < 0, điểm không tồn tại.
 # Nếu d = 0, điểm nằm trên biên của không gian làm việc (điểm kỳ dị).
 # Sử dụng một sai số nhỏ để tránh các vấn đề về số thực và coi các
 # điểm rất gần kỳ dị là không thể đạt tới.
 if d < 1e-9:
 return None # Điểm không tồn tại hoặc là điểm kỳ dị

 # choosing outer point
 yj = (y1 - a * b - math.sqrt(d)) / (b**2 + 1.0)
 zj = a + b * yj

 # Ngăn chia cho 0 nếu yj == y1
 if abs(y1 - yj) < 1e-9:
 if zj < 0:
 theta_deg = -90.0
 else:
 theta_deg = 90.0
 else:
 theta_rad = math.atan(-zj / (y1 - yj))
 theta_deg = 180.0 * theta_rad / self.pi

 # Xử lý góc phần tư
 if yj > y1:
 theta_deg += 180.0

 return theta_deg

def inverse_kinematics(self, x0, y0, z0):
 """
 Tính toán các góc động cơ (theta1, 2, 3) cho một tọa độ TÂM BÊ (x0,
 y0, z0).
 ↳ OPTIMIZED: Sử dụng Numba JIT compilation (20-100x nhanh hơn).

 :return: (theta1, theta2, theta3) trong độ, hoặc None nếu không thể
 đạt
 """
 # Gọi Numba-optimized function
 theta1, theta2, theta3 = inverse_kinematics_numba(
 x0, y0, z0,
 self.tan30, self.f, self.e, self.rf, self.re, self.pi, self.sqrt3,

```

```
 self.cos120, self.sin120, C.ARM_ANGLE_MIN, C.ARM_ANGLE_MAX
)

Kiểm tra thất bại (Numba trả về nan)
if np.isnan(theta1):
 return None

return (theta1, theta2, theta3)

def inverse_kinematics_tool(self, xt, yt, zt, alpha_deg):
 """
 Tính động học nghịch cho ĐẦU HÚT (xt, yt, zt) có offset servo.
 ↪ OPTIMIZED: Sử dụng Numba JIT compilation (20-100x nhanh hơn).
 """
 # Gọi Numba-optimized function trực tiếp
 theta1, theta2, theta3 = inverse_kinematics_tool_numba(
 xt, yt, zt, alpha_deg,
 self.tool_offset_radius, self.tool_offset_z,
 self.tan30, self.f, self.e, self.rf, self.re, self.pi, self.sqrt3,
 self.cos120, self.sin120, C.ARM_ANGLE_MIN, C.ARM_ANGLE_MAX
)

 # Kiểm tra thất bại (Numba trả về nan)
 if np.isnan(theta1):
 return None

 return (theta1, theta2, theta3)

--- CÁC HÀM TIỆN ÍCH CHUYỂN ĐỔI SANG XUNG ---

def angles_to_steps(self, theta1, theta2, theta3):
 """
 Chuyển đổi góc khớp (độ) thành số xung động cơ.
 """
 steps1 = theta1 * self.steps_per_arm_degree
 steps2 = theta2 * self.steps_per_arm_degree
 steps3 = theta3 * self.steps_per_arm_degree

 return (round(steps1), round(steps2), round(steps3))

def steps_to_angles(self, steps1, steps2, steps3):
 """
 Chuyển đổi số xung động cơ thành góc khớp (độ).
 Đây là hàm ngược lại của angles_to_steps.
 """
 theta1 = steps1 / self.steps_per_arm_degree
 theta2 = steps2 / self.steps_per_arm_degree
 theta3 = steps3 / self.steps_per_arm_degree

 return (theta1, theta2, theta3)
```

```
def inverse_kinematics_to_steps(self, x0, y0, z0):
 """
 Tính số xung động cơ trực tiếp từ tọa độ TÂM BỆ (effector).
 (Hàm này chỉ dùng để test, controller nên dùng _tool_)
 """
 angles = self.inverse_kinematics(x0, y0, z0)
 if angles is None:
 return None
 return self.angles_to_steps(*angles)

def inverse_kinematics_tool_to_steps(self, xt, yt, zt, alpha_deg):
 """
 Tính số xung động cơ từ tọa độ ĐẦU HÚT.
 Đây là hàm mà robot_controller nên sử dụng.
 """
 angles = self.inverse_kinematics_tool(xt, yt, zt, alpha_deg)
 if angles is None:
 return None
 return self.angles_to_steps(*angles)
```

## PHỤ LỤC 2: CHƯƠNG TRÌNH CHÍNH TRÊN STM32

Chương trình file main.c

```
#include "main.h"
#include "usb_device.h"

/* Private includes -----
*/
/* USER CODE BEGIN Includes */
#include "cdc_handler.h"
#include "command_queue.h"
#include "robot_control.h"
#include "conveyor.h"
#include "status_led.h"
#include "command_parser.h"
#include "button_handler.h"
#include "stm32f1xx_hal.h"
#include <stdbool.h>
#include <stdint.h>
/* USER CODE END Includes */

/* Private typedef -----
*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define -----
*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -----
*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables -----
*/
TIM_HandleTypeDef htim2;
TIM_HandleTypeDef htim3;
TIM_HandleTypeDef htim4;

/* USER CODE BEGIN PV */

/* USER CODE END PV */
```

```
/* Private function prototypes -----
*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM2_Init(void);
static void MX_TIM3_Init(void);
static void MX_TIM4_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code -----
*/
/* USER CODE BEGIN 0 */
static char main_command_buffer[RX_BUFFER_SIZE]; // NOLINT
/* USER CODE END 0 */

/**
 * @brief The application entry point.
 * @retval int
 */
int main(void)
{

 /* USER CODE BEGIN 1 */

 /* USER CODE END 1 */

 /* MCU Configuration-----
 */

 /* Reset of all peripherals, Initializes the Flash interface and the
 Systick. */
 HAL_Init();

 /* USER CODE BEGIN Init */

 /* USER CODE END Init */

 /* Configure the system clock */
 SystemClock_Config();

 /* USER CODE BEGIN SysInit */

 /* USER CODE END SysInit */

 /* Initialize all configured peripherals */
 MX_GPIO_Init();
 MX_TIM2_Init();
 MX_TIM3_Init();
```

```
MX_TIM4_Init();
MX_USB_DEVICE_Init();
/* USER CODE BEGIN 2 */
// Khởi tạo tất cả các module phần mềm
status_led_init();
cdc_handler_init();
queue_init();
robot_init();
conveyor_init();
button_handler_init(); // Khởi tạo module xử lý nút bấm và debounce

// Đợi một chút để USB CDC ổn định
HAL_Delay(100);

// Gửi thông báo hệ thống đã sẵn sàng
cdc_handler_send_response("SYS_READY");
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
 /* USER CODE END WHILE */

 /* USER CODE BEGIN 3 */
 // Xử lý lệnh từ USB (NON-BLOCKING)
 // Kiểm tra xem cdc_handler có lệnh mới không
 if (cdc_handler_get_command(main_command_buffer, RX_BUFFER_SIZE))
 {
 // Nếu có, thực thi lệnh trong main loop (an toàn!)
 parse_command(main_command_buffer);
 }

 // Xử lý hàng đợi lệnh: kiểm tra và gửi lệnh cho robot nếu rảnh
 queue_process();

 // Xử lý các thông báo DONE từ hàng đợi (do ISR đẩy vào)
 queue_handle_done_messages();

 // Cập nhật trạng thái nút bấm và E-Stop (có debounce)
 button_handler_update();

 // Nếu E-Stop vừa thay đổi (được phát hiện bên trong
 button_handler_update),
 // ta nên gửi status ngay. Tuy nhiên button_handler_update gọi callback
 robot_estop_triggered.

 // Để đơn giản, ta kiểm tra cạnh ở đây.
 static bool last_estop_val = false;
 bool current_estop_val = estop_is_triggered();
 if (current_estop_val != last_estop_val) {
```

```
 last_estop_val = current_estop_val;
 send_status_report(); // Gửi status ngay khi E-Stop đổi trạng thái
}

// --- Xử lý sự kiện nút bấm ---
if (button_just_pressed(BUTTON_1) || button_just_pressed(BUTTON_2)) {
 send_status_report(); // Gửi trạng thái đầy đủ ngay khi nhấn nút
}

// --- Giám sát cảm biến khay (Tray Sensor) ---
static int last_tray_state = -1;
int current_tray_state = (HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_15) ==
GPIO_PIN_RESET) ? 1 : 0;

if (last_tray_state == -1) {
 last_tray_state = current_tray_state;
}
else if (current_tray_state != last_tray_state) {
 HAL_Delay(20);
 int confirm_state = (HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_15) ==
GPIO_PIN_RESET) ? 1 : 0;
 if (confirm_state == current_tray_state) {
 last_tray_state = current_tray_state;
 send_status_report(); // Gửi trạng thái đầy đủ khi cảm biến khay
thay đổi
 }
}

// Cập nhật trạng thái homing (gọi mỗi vòng lặp)
robot_update_homing_state();

// Kiểm tra Servo Idle để ngắt xung (Auto-Detach)
robot_poll_servo_idle();

// Clear flags để tránh memory leak (không log nhưng vẫn clear)
for (int i = 0; i < 3; i++) {
 robot_get_and_clear_flag_homing_ls_trig(i);
}

// Cập nhật LED trạng thái (gọi mỗi vòng lặp để nhấp nháy được smooth)
status_led_update();

// --- Xử lý Homing Watchdog và các cờ báo hiệu từ ISR ---

// Watchdog cho Homing (chỉ kiểm tra khi đang homing)
if (robot_get_state() == ROBOT_STATE_HOMING && robot_get_homing_state() ==
HOMING_STATE_RAISING)
{
 if ((HAL_GetTick() - robot_get_homing_start_tick()) >
HOMING_WATCHDOG_MS)
```

```

 {
 robot_abort(); // Dừng robot
 cdc_handler_send_response("ERROR:Homing timed out");
 }
}

// Các cờ khác
if (robot_get_and_clear_flag_homing_timeout_backing_off())
cdc_handler_send_response("ERROR:HOMING_TIMEOUT_BACKING_OFF");
if (robot_get_and_clear_flag_homing_done())
cdc_handler_send_response("HOME_DONE");

int misconfig_motor;
if (robot_get_and_clear_flag_homing_dir_misconfig(&misconfig_motor)) {
 cdc_handler_send_response("ERROR:HOMING_DIR_MISCONFIG on motor %d",
misconfig_motor);
}

// SAFETY: Kiểm tra limit switch bị kẹt/hởng
for (int i = 0; i < 3; i++) {
 if (robot_get_and_clear_flag_ls_stuck(i)) {
 cdc_handler_send_response("ERROR:LIMIT_SWITCH_STUCK:M%d (backoff
>2000 steps, sensor may be broken)", i + 1);
 robot_abort(); // Dừng khẩn cấp
 }
}

for (int i = 0; i < 3; i++) {
 if (robot_get_and_clear_flag_up_blocked(i))
cdc_handler_send_response("WARN:Motor %d upward movement blocked by limit
switch", i + 1);
}
/* USER CODE END 3 */
}

/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void)
{
 RCC_OscInitTypeDef RCC_OscInitStruct = {0};
 RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
 RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};

 /** Initializes the RCC Oscillators according to the specified parameters
 * in the RCC_OscInitTypeDef structure.
 */
 RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
}

```

```
RCC_OscInitStruct.HSEState = RCC_HSE_ON;
RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
RCC_OscInitStruct.HSISState = RCC_HSI_ON;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
 Error_Handler();
}

/** Initializes the CPU, AHB and APB buses clocks
 */
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
{
 Error_Handler();
}
PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USB;
PeriphClkInit.UsbClockSelection = RCC_USBCLKSOURCE_PLL_DIV1_5;
if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
{
 Error_Handler();
}
}

/**
 * @brief TIM2 Initialization Function
 * @param None
 * @retval None
 */
static void MX_TIM2_Init(void)
{
 /* USER CODE BEGIN TIM2_Init_0 */

 /* USER CODE END TIM2_Init_0 */

 TIM_ClockConfigTypeDef sClockSourceConfig = {0};
 TIM_MasterConfigTypeDef sMasterConfig = {0};

 /* USER CODE BEGIN TIM2_Init_1 */

 /* USER CODE END TIM2_Init_1 */
}
```

```
htim2.Instance = TIM2;
htim2.Init.Prescaler = 71;
htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
htim2.Init.Period = 99;
htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
{
 Error_Handler();
}
sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
{
 Error_Handler();
}
sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
{
 Error_Handler();
}
/* USER CODE BEGIN TIM2_Init 2 */
// FIX: Cấu hình GPIO cho TIM2_CH1 (PA0) để xuất PWM cho băng tải
HAL_TIM_MspPostInit(&htim2);
// Cấu hình channel PWM sẽ được thực hiện trong conveyor_init
// sMasterConfig và HAL_TIMEx_MasterConfigSynchronization đã được chuyển vào
conveyor_init
/* USER CODE END TIM2_Init 2 */

}

/**
 * @brief TIM3 Initialization Function
 * @param None
 * @retval None
 */
static void MX_TIM3_Init(void)
{

 /* USER CODE BEGIN TIM3_Init 0 */

 /* USER CODE END TIM3_Init 0 */

 TIM_ClockConfigTypeDef sClockSourceConfig = {0};
 TIM_MasterConfigTypeDef sMasterConfig = {0};

 /* USER CODE BEGIN TIM3_Init 1 */
 // Việc khởi tạo chi tiết timer này (prescaler, period, interrupt callback)
 // đã được chuyển vào hàm robot_init() trong file robot_control.c
 // để tránh bị CubeMX ghi đè.
}
```

```
// Hàm này được giữ lại ở đây để duy trì cấu trúc dự án của CubeMX.
/* USER CODE END TIM3_Init 1 */
htim3.Instance = TIM3;
htim3.Init.Prescaler = 71;
htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
htim3.Init.Period = 99;
htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
if (HAL_TIM_Base_Init(&htim3) != HAL_OK)
{
 Error_Handler();
}
sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
if (HAL_TIM_ConfigClockSource(&htim3, &sClockSourceConfig) != HAL_OK)
{
 Error_Handler();
}
sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig) != HAL_OK)
{
 Error_Handler();
}
/* USER CODE BEGIN TIM3_Init 2 */

/* USER CODE END TIM3_Init 2 */

}

/**
 * @brief TIM4 Initialization Function
 * @param None
 * @retval None
 */
static void MX_TIM4_Init(void)
{

/* USER CODE BEGIN TIM4_Init 0 */

/* USER CODE END TIM4_Init 0 */

TIM_ClockConfigTypeDef sClockSourceConfig = {0};
TIM_MasterConfigTypeDef sMasterConfig = {0};
TIM_OC_InitTypeDef sConfigOC = {0};

/* USER CODE BEGIN TIM4_Init 1 */
// Việc khởi tạo chi tiết timer này cho Servo (PWM)
// đã được chuyển vào hàm robot_init() trong file robot_control.c
// để tránh bị CubeMX ghi đè.
/* USER CODE END TIM4_Init 1 */
```

```
htim4.Instance = TIM4;
htim4.Init.Prescaler = 71;
htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
htim4.Init.Period = 19999;
htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim4.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
if (HAL_TIM_Base_Init(&htim4) != HAL_OK)
{
 Error_Handler();
}
sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
if (HAL_TIM_ConfigClockSource(&htim4, &sClockSourceConfig) != HAL_OK)
{
 Error_Handler();
}
if (HAL_TIM_PWM_Init(&htim4) != HAL_OK)
{
 Error_Handler();
}
sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
if (HAL_TIMEx_MasterConfigSynchronization(&htim4, &sMasterConfig) != HAL_OK)
{
 Error_Handler();
}
sConfigOC.OCMode = TIM_OCMODE_PWM1;
sConfigOC.Pulse = 1500;
sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
if (HAL_TIM_PWM_ConfigChannel(&htim4, &sConfigOC, TIM_CHANNEL_4) != HAL_OK)
{
 Error_Handler();
}
/* USER CODE BEGIN TIM4_Init_2 */

/* USER CODE END TIM4_Init_2 */
HAL_TIM_MspPostInit(&htim4);

}

/**
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
 */
static void MX_GPIO_Init(void)
{
 GPIO_InitTypeDef GPIO_InitStruct = {0};
 /* USER CODE BEGIN MX_GPIO_Init_1 */
```

```
/* USER CODE END MX_GPIO_Init_1 */

/* GPIO Ports Clock Enable */
__HAL_RCC_GPIOD_CLK_ENABLE();
__HAL_RCC_GPIOA_CLK_ENABLE();
__HAL_RCC_GPIOB_CLK_ENABLE();

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_3|GPIO_PIN_5
 |GPIO_PIN_7|GPIO_PIN_9, GPIO_PIN_RESET); // PA9
(PUMP) = 0V = TẮT

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2|GPIO_PIN_4|GPIO_PIN_6
 |GPIO_PIN_10|GPIO_PIN_15, GPIO_PIN_SET);

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_10, GPIO_PIN_RESET);

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_11|GPIO_PIN_3, GPIO_PIN_SET);

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(CAMERA_LIGHT_GPIO_Port, CAMERA_LIGHT_Pin,
 GPIO_PIN_RESET); // Đèn trợ sáng BẬT khi khởi động (Relay thường đóng: LOW = đóng relay = bật đèn)

/*Configure GPIO pins : PA0 PA1 PA2 PA3
 PA4 PA5 PA6 PA7
 PA9 PA10 PA15 */
GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3
 |GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7
 |GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_15;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/*Configure GPIO pins : PB0 PB1 PB10 PB11
 PB3 PB4 */
GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_10|GPIO_PIN_11
 |GPIO_PIN_3|GPIO_PIN_4;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

/*Configure GPIO pins : PB12 PB13 PB14 (Reserved - NC with PULLUP) */
GPIO_InitStruct.Pin = GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
```

```
GPIO_InitStruct.Pull = GPIO_PULLUP;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

/*Configure GPIO pins : PB15 (Conveyor Sensor - NPN NO with PULLUP) */
GPIO_InitStruct.Pin = GPIO_PIN_15;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_PULLUP;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

/*Configure GPIO pins : PB6, PB7 (START and STOP buttons - NO with VCC) */
/* Logic: Pressed = 1 (VCC), Released = 0 (Internal PULLDOWN) */
GPIO_InitStruct.Pin = GPIO_PIN_6|GPIO_PIN_7;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_PULLDOWN;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

/*Configure GPIO pin : PA8 (E-Stop - NC with PULLUP, interrupt on RISING
edge) */
GPIO_InitStruct.Pin = GPIO_PIN_8;
GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
GPIO_InitStruct.Pull = GPIO_PULLUP;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/* EXTI interrupt init*/
HAL_NVIC_SetPriority(EXTI9_5_IRQn, 2, 0);
HAL_NVIC_EnableIRQ(EXTI9_5_IRQn);

/* USER CODE BEGIN MX_GPIO_Init_2 */
// Các cấu hình GPIO khác như công tắc hành trình, nút nhấn...
// nên được thực hiện trong các module tương ứng (ví dụ: robot_init)
// để đảm bảo tính đóng gói và dễ quản lý.

/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */

/**
 * @brief GPIO EXTI Callback - Xử lý ngắt E-Stop
 * @param GPIO_Pin: Pin gây ra ngắt
 * @note: Đã chuyển sang xử lý debounce trong button_handler_update()
 * Ngắt này chỉ để phản ứng nhanh, logic thực sự xử lý ở polling
 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
 // Ngắt E-Stop - không xử lý gì ở đây
 // Debounce và xử lý logic sẽ được thực hiện trong button_handler_update()
 // Điều này tránh việc gọi hàm phức tạp từ ISR
 (void)GPIO_Pin; // Tránh warning unused parameter
}
```

```
/* USER CODE END 4 */

/**
 * @brief This function is executed in case of error occurrence.
 * @retval None
 */
void Error_Handler(void)
{
 /* USER CODE BEGIN Error_Handler_Debug */
 /* User can add his own implementation to report the HAL error return state
 */
 __disable_irq();
 while (1)
 {
 }
 /* USER CODE END Error_Handler_Debug */
}
```

## PHỤ LỤC 3: CHƯƠNG TRÌNH CHÍNH TRÊN MÁY TÍNH

Chương trình file main.py

```
import sys
import os
from PyQt6.QtWidgets import QApplication
from PyQt6.QtCore import QTimer
from pyqt_delta_gui import DeltaRobotGUI
from connection_manager import ConnectionManager
from robot_controller import RobotController
import constants as C

... [Các phần import khác và cấu hình môi trường] ...

def main():
 # Cấu hình môi trường Qt để hiển thị tốt trên màn hình High-DPI
 os.environ["QT_LOGGING_RULES"] = "*.debug=false;qt.qpa.*=false"

 global qt_app
 qt_app = QApplication(sys.argv)

 try:
 # 1. Khởi tạo Giao diện chính (GUI)
 app = DeltaRobotGUI()

 # 2. Khởi tạo Bộ điều khiển Robot & Quản lý kết nối
 app.robot_controller = RobotController(app, app.conn_manager)

 # 3. Liên kết các sự kiện nút bấm trên GUI với logic điều khiển
 app.set_button_commands({
 'run': app.on_run if hasattr(app, 'on_run') else lambda:
print("Run"),
 'stop': app.on_stop if hasattr(app, 'on_stop') else lambda:
print("Stop"),
 'connect_stm': lambda: on_connect_stm(app),
 # ... [Các lệnh bind nút bấm khác: Move, Home, Pump, Conveyor...]
 })

 # 4. Liên kết nút bấm cứng (Hardware Buttons) với Auto Mode
 app.robot_controller.hardware_start_pressed.connect(lambda:
app.robot_controller.set_auto_mode(True))
 app.robot_controller.hardware_stop_pressed.connect(lambda:
app.robot_controller.set_auto_mode(False))

 # 5. Đăng ký lắng nghe phản hồi từ STM32
 app.conn_manager.add_message_listener(app.robot_controller.handle_stm_
message)

 # 6. Hiển thị ứng dụng và bắt đầu xử lý Video
 app.show()

```

```
Khởi động luồng Camera sau 100ms (để GUI hiện lên trước)
QTimer.singleShot(100, lambda: start_video_feed(app))

Chạy vòng lặp sự kiện chính của Qt
sys.exit(qt_app.exec())

except KeyboardInterrupt:
 print("\nInterrupted.")
except Exception as e:
 print(f"Error: {e}")

if __name__ == "__main__":
 main()
```