

**ĐẠI HỌC QUỐC GIA TP.HCM  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH**



**BÁO CÁO  
ĐỒ ÁN TỐT NGHIỆP**

**PHÁT TRIỂN MOTOR DRIVER  
TRÊN NỀN TẢNG ROS  
SỬ DỤNG MẠCH NHÚNG JETSON**

Ngành: Kỹ thuật máy tính

**HỘI ĐỒNG: HỘI ĐỒNG 5 KỸ THUẬT MÁY TÍNH**

**GVHD: TS. LÊ TRỌNG NHÂN**

**GVPB: ThS. HUỲNH HOÀNG KHA**

---o0o---

**SVTH 1: Huỳnh Hoàng Ly (2013728)**

**SVTH 2: Hà Trung Quyền (2014314)**

**SVTH 3: Hoàng Minh Triết (2012262)**

## TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA: KH &amp; KT Máy tính

BỘ MÔN: Kỹ thuật Máy tính

## NHIỆM VỤ LUẬN ÁN TỐT NGHIỆP

Chú ý: Sinh viên phải dán tờ này vào trang nhất của bản thuyết trình

HỌ VÀ TÊN: HUỲNH HOÀNG LY

MSSV: 2013728

HỌ VÀ TÊN: HÀ TRUNG QUYỀN

MSSV: 2014314

HỌ VÀ TÊN: HOÀNG MINH TRIẾT

MSSV: 2012262

NGÀNH: KỸ THUẬT MÁY TÍNH

LỚP: MT20KTTN

## 1. Đầu đề luận án:

**PHÁT TRIỂN MOTOR DRIVER TRÊN NỀN TẢNG ROS SỬ DỤNG MẠCH NHÚNG JETSON**

## 2. Nhiệm vụ (yêu cầu về nội dung và số liệu ban đầu):

- Nghiên cứu và phát triển device driver điều khiển cánh tay robot 6 trực tự do (6DoF) và tích hợp ROS vào đó.
- Huấn luyện, phát triển mô hình trí tuệ nhân tạo nhận diện vật thể và tích hợp vào cánh tay robot.
- Tích hợp mô hình trí tuệ nhân tạo lên cánh tay Robot cho một ứng dụng minh họa

3. Ngày giao nhiệm vụ luận án: 22/01/2024

4. Ngày hoàn thành nhiệm vụ: 20/05/2024

## 5. Họ tên giảng viên hướng dẫn:

1) TS. LÊ TRỌNG NHÂN

## Phần hướng dẫn:

Toàn bộ

Nội dung và yêu cầu LVTN đã được thông qua Bộ môn.

Ngày ..... tháng ..... năm .....

CHỦ NHIỆM BỘ MÔN

(Ký và ghi rõ họ tên)

GIẢNG VIÊN HƯỚNG DẪN CHÍNH

(Ký và ghi rõ họ tên)

TS. Lê Trọng Nhân

PHẦN DÀNH CHO KHOA, BỘ MÔN:

Người duyệt (chấm sơ bộ):

Đơn vị:

Ngày bảo vệ:

Điểm tổng kết:

Nơi lưu trữ luận án:

---

## LỜI CAM ĐOAN

Chúng tôi cam kết rằng Đồ án này dựa trên ý tưởng và kiến thức của những người giám sát của chúng tôi. Tất cả các nghiên cứu và dữ liệu chưa được công bố. Các tài liệu tham khảo, các số liệu và thông kê là đáng tin cậy và trung thực. Nhóm đã hoàn thành các yêu cầu của Đồ án tốt nghiệp do khoa Khoa học và Kỹ thuật Máy tính đề ra.

Chữ ký GVHD

Trân trọng,

**Huỳnh Hoàng Ly  
Hà Trung Quyền  
Hoàng Minh Triết**

---

## LỜI CẢM ƠN

Trong thời gian thực hiện Đồ án tốt nghiệp, chúng em đã nhận được nhiều sự giúp đỡ, đóng góp ý kiến và hướng dẫn nhiệt tình của quý thầy cô và bạn bè.

Đặc biệt, chúng em xin bày tỏ sự kính trọng và lòng biết ơn sâu sắc nhất đến giảng viên hướng dẫn Đồ án của chúng em - TS. Lê Trọng Nhân, thầy là người đã tận tình chỉ bảo, góp ý về kiến thức cũng như những kỹ năng khác để chúng em có thể hoàn thành được đồ án này. Thầy đã luôn đồng hành và sẵn sàng giúp đỡ chúng em mỗi khi cần, thầy cho chúng em động lực, nguồn cảm hứng và những lời khuyên quý báu để có thể vượt qua những giai đoạn khó khăn trong quá trình thực hiện đồ án.

Ngoài ra, chúng em cũng muốn gửi lời cảm ơn chân thành đến các anh chị nhóm luận văn học kỳ 223 đã chia sẻ những kinh nghiệm hữu ích cũng như nhiệt tình tư vấn cho chúng em trong công tác chuẩn bị cho đồ án.

Chúng em cũng xin chân thành cảm ơn quý thầy cô giáo trường *Dai hoc Bach Khoa - Dai hoc Quoc gia thanh pho Ho Chi Minh* nói chung, quý thầy cô khoa *Khoa hoc va Ky thuat May tinh* nói riêng đã cung cấp cho chúng em kiến thức về các môn đại cương và chuyên ngành, giúp chúng em có cơ sở lý thuyết vững vàng trong quá trình học tập và hiện thực đồ án.

Với điều kiện thời gian không quá dài cũng như kinh nghiệm và kiến thức còn hạn chế, đồ án của chúng em sẽ không tránh được những thiếu sót cũng như mắc phải vẫn đề chưa giải quyết được. Do đó, chúng em rất mong nhận được sự chỉ dẫn, đóng góp ý kiến của các thầy cô để chúng em có điều kiện bổ sung, cải thiện, đồng thời nâng cao kiến thức của nhóm và có một nền tảng tốt hơn phục vụ cho công việc sau này.

Cuối cùng, chúng em xin kính chúc quý thầy, quý cô, quý nhà trường mạnh khỏe. Kính chúc cho sự nghiệp tròng người, đào tạo nhân tài phục vụ đất nước của trường nói chung, của khoa nói riêng luôn thành công và đạt được nhiều thành tựu rực rỡ.

Nhóm chúng em xin chân thành tri ân và gửi lời cảm ơn đến tất cả mọi người!

---

## TÓM TẮT

Cuộc cách mạng Công nghiệp 4.0 đang mở ra xu hướng mới cho các nhà máy công nghiệp thông minh. Với sự phát triển vượt bậc với sự hỗ trợ của robot, trao đổi dữ liệu cảm biến khổng lồ dựa trên Internet of Things và các tiến bộ công nghệ khác bao gồm Digital Twin, Augmented Reality (AR) và Virtual Reality (VR), các nhà máy có thể tăng cường hiệu suất và giảm chi phí trong quá trình sản xuất. Tiếp nối sự phát triển của ngành robot học, báo cáo này sẽ trình bày nghiên cứu và phát triển hệ thống điều khiển động cơ cho một Cánh tay Robot 6 bậc tự do (Robot Arm 6-DoF) dựa trên khung ROS (Robot Operating System), sử dụng bo mạch nhúng NVIDIA Jetson.

Để đáp ứng nhu cầu ngày càng tăng của các hệ thống robot tự động, công việc của nhóm tập trung vào việc tạo ra một mô-đun có trình điều khiển động cơ tương tác tốt với ROS, tận dụng khả năng xử lý song song của nền tảng Jetson. Báo cáo mô tả kiến trúc phần cứng và phần mềm, đặc biệt là tích hợp các nút ROS để kiểm soát và phản hồi động cơ. Thông qua các thử nghiệm mở rộng, chúng xác minh khả năng phản ứng và độ tin cậy thời gian thực của trình điều khiển động cơ trong các ứng dụng robot đa dạng, đóng góp vào hệ sinh thái ngày càng mở rộng của phần cứng tương thích với ROS. Ngoài các dự án điều khiển động cơ thông thường, công trình của chúng tôi còn giới thiệu một chiều hướng đổi mới bằng cách kết hợp mô hình Trí tuệ nhân tạo (AI) để phát hiện đối tượng theo thời gian thực.

Các từ khóa : Motor Driver, ROS, Jetson Nano, Robotics, Embedded Systems, Robot Arm 6DoF,...

---

## Mục lục

---

<b>Chương 1 GIỚI THIỆU TỔNG QUAN</b>	<b>2</b>
1.1 Robot trong đời sống hiện nay . . . . .	2
1.2 Lý do chọn đề tài . . . . .	4
1.3 Đối tượng nghiên cứu . . . . .	6
1.4 Phạm vi và mục tiêu của Đồ án . . . . .	8
1.4.1 Phạm vi . . . . .	8
1.4.2 Mục tiêu . . . . .	8
1.4.3 Giới hạn đề tài . . . . .	9
1.5 Cấu trúc của bài báo cáo Đồ án . . . . .	10
<b>Chương 2 KIẾN THỨC NỀN TẢNG</b>	<b>11</b>
2.1 Kiến thức cơ bản về Device Driver . . . . .	11
2.1.1 Sơ lược về Device Driver . . . . .	11
2.1.2 Quá trình hoạt động . . . . .	12
2.2 Kiến thức cơ bản về ROS . . . . .	14
2.2.1 Sơ lược về ROS . . . . .	14
2.2.2 Kiến trúc của ROS . . . . .	15
2.2.3 Ứng dụng trong thực tế . . . . .	17
2.2.4 Tiềm năng phát triển trong tương lai . . . . .	18
2.3 Tổng quan về 6DoF Robot Arm . . . . .	20
2.3.1 Khái niệm . . . . .	20
2.3.2 Ứng dụng . . . . .	21
2.4 Động học robot . . . . .	24
2.4.1 Quy tắc Denavit – Hartenberg (DH) . . . . .	25
2.4.2 Động học thuận . . . . .	28
2.4.3 Động học nghịch . . . . .	30
2.5 Tổng quan về Stepper motor . . . . .	32
2.5.1 Cấu tạo và nguyên lý hoạt động . . . . .	32
2.5.2 Nguyên lý điều khiển . . . . .	34
2.6 Tổng quan về vi điều khiển . . . . .	36
2.6.1 Jetson Nano . . . . .	36
2.6.2 Arduino Mega . . . . .	40

2.7	Giao tiếp giữa các thiết bị . . . . .	44
2.7.1	GPIO . . . . .	44
2.7.2	UART . . . . .	45
2.8	Những kỹ thuật trong xử lý ảnh và thị giác máy tính . . . . .	46
2.8.1	Một số đặc điểm về usb camera . . . . .	46
2.8.2	Kỹ thuật hiệu chỉnh camera (camera calibration) . . . . .	47
2.9	Nguyên lý nhận dạng vật thể . . . . .	50
2.9.1	Tiền xử lý . . . . .	50
2.9.2	Trích xuất đặc trưng . . . . .	50
2.9.3	Định vị và phân loại vật thể . . . . .	51
2.9.4	Hậu xử lý . . . . .	52
2.10	Một số công nghệ xử lý ảnh và thị giác máy tính phổ biến . . . . .	54
2.10.1	OpenCV . . . . .	54
2.10.2	YOLOv8 . . . . .	55
<b>Chương 3</b>	<b>THIẾT KẾ VÀ HIỆN THỰC HỆ THỐNG</b>	<b>57</b>
3.1	Thiết kế robot JetArm . . . . .	57
3.1.1	Cấu tạo cơ khí của robot JetArm . . . . .	58
3.1.2	Kết nối điện tử trong robot JetArm . . . . .	70
3.2	Đề xuất chức năng . . . . .	73
3.3	Xây dựng kiến trúc hệ thống . . . . .	74
3.3.1	Yêu cầu hệ thống . . . . .	74
3.3.2	Kiến trúc hệ thống . . . . .	77
3.4	Hiện thực hệ thống . . . . .	79
3.4.1	Chế độ Manual . . . . .	79
3.4.2	Chế độ Auto . . . . .	82
3.4.3	Huấn luyện mô hình nhận diện cờ tướng . . . . .	83
3.4.4	Hiệu chỉnh camera . . . . .	91
3.4.5	Tích hợp ROS . . . . .	93
3.4.6	Máy trạng thái . . . . .	94
3.5	Máy trạng thái tổng thể . . . . .	96
3.6	Mã nguồn toàn bộ chương trình . . . . .	97
3.7	Hình ảnh toàn bộ hệ thống . . . . .	97
<b>Chương 4</b>	<b>KẾT QUẢ ĐẠT ĐƯỢC &amp; ĐÁNH GIÁ KẾT QUẢ</b>	<b>98</b>
4.1	Mô hình nhận diện quân cờ . . . . .	98
4.2	Hiệu chỉnh (calibration) camera và việc gấp, thả quân cờ . . . . .	102
4.3	Đánh giá tổng quan . . . . .	103
4.4	So sánh với các đề tài khác . . . . .	110
<b>Chương 5</b>	<b>TỔNG KẾT VÀ HƯỚNG PHÁT TRIỂN</b>	<b>112</b>
5.1	Tổng kết . . . . .	112

5.2 Hướng phát triển trong tương lai . . . . .	113
<b>Tài liệu tham khảo</b>	<b>115</b>

---

## Danh sách hình ảnh

---

1.1	Robot JetArm . . . . .	6
1.2	Jetson Nano . . . . .	7
1.3	Stepper Motor Driver TB6600 . . . . .	7
2.1	Vai trò của Device Driver trong hệ thống máy tính . . . . .	12
2.2	Nguyên lý hoạt động của Device Driver . . . . .	13
2.3	Kiến trúc của ROS2 [11] . . . . .	16
2.4	Ứng dụng của ROS [11] . . . . .	17
2.5	Minh họa cấu trúc của cánh tay robot 6DoF . . . . .	21
2.6	6DoF robot arm part picking and handling . . . . .	22
2.7	Painting bằng 6DoF robot arm . . . . .	22
2.8	6DoF robot arm khoan trên bề mặt sản phẩm . . . . .	23
2.9	Cấu hình khung tọa độ DH cho robot JetArm . . . . .	26
2.10	Minh họa cách tìm các thông số cho bảng DH [14] . . . . .	27
2.11	Xác định cấu hình vị trí khớp của robot sử dụng động học robot . . . . .	29
2.12	Mối tương quan giữa động học thuận và động học nghịch . . . . .	30
2.13	Sơ đồ nguyên lý hoạt động của động cơ bước . . . . .	32
2.14	Hình ảnh thực tế bên trong động cơ bước . . . . .	34
2.15	Tín hiệu điều khiển động cơ bước khác nhau . . . . .	34
2.16	Hiệu suất của Jetson Nano . . . . .	37
2.17	Ứng dụng của Jetson Nano trong xe tự hành . . . . .	39
2.18	Ứng dụng của Jetson Nano trong việc nhận diện hành vi con người thông qua camera . . . . .	39
2.19	Ứng dụng của Jetson Nano trong Robotics . . . . .	40
2.20	Cấu tạo của Arduino Mega . . . . .	40
2.21	Arduino Mega và Ultrasonic . . . . .	42
2.22	Arduino Mega và LCD . . . . .	42
2.23	Ramp Shield . . . . .	43
2.24	Image Sensor[19] . . . . .	46
2.25	Formation of an image in a camera[20] . . . . .	47
2.26	Transformation from image to real object [21] . . . . .	48

2.27 Transformation from image to real object [21] . . . . .	49
2.28 Convolution Neutral Network . . . . .	52
2.29 Decision Tree . . . . .	52
2.30 YOLO version comparation[24] . . . . .	56
3.1 Cấu tạo của robot JetArm . . . . .	58
3.2 Cấu tạo trục tự do J6 . . . . .	59
3.3 Cấu tạo trục tự do J6 . . . . .	59
3.4 Cấu tạo trục tự do J6 . . . . .	59
3.5 Cấu tạo trục tự do J6 . . . . .	60
3.6 Cấu tạo trục tự do J6 . . . . .	60
3.7 Trục tự do J6 trong robot JetArm . . . . .	60
3.8 Cấu tạo trục tự do J5 . . . . .	61
3.9 Cấu tạo trục tự do J5 . . . . .	61
3.10 Cấu tạo trục tự do J5 . . . . .	61
3.11 Trục tự do J5 trong robot JetArm . . . . .	61
3.12 Cấu tạo trục tự do J4 . . . . .	62
3.13 Cấu tạo trục tự do J4 . . . . .	62
3.14 Cấu tạo trục tự do J4 . . . . .	62
3.15 Cấu tạo trục tự do J4 . . . . .	62
3.16 Cấu tạo trục tự do J4 . . . . .	63
3.17 Cấu tạo trục tự do J4 . . . . .	63
3.18 Trục tự do J4 trong robot JetArm . . . . .	63
3.19 Cấu tạo trục tự do J3 . . . . .	64
3.20 Cấu tạo trục tự do J3 . . . . .	64
3.21 Cấu tạo trục tự do J3 . . . . .	64
3.22 Cấu tạo trục tự do J3 . . . . .	65
3.23 Cấu tạo trục tự do J3 . . . . .	65
3.24 Trục tự do J3 trong robot JetArm . . . . .	65
3.25 Cấu tạo trục tự do J2 . . . . .	66
3.26 Cấu tạo trục tự do J2 . . . . .	66
3.27 Cấu tạo trục tự do J2 . . . . .	66
3.28 Cấu tạo trục tự do J2 . . . . .	67
3.29 Cấu tạo trục tự do J2 . . . . .	67
3.30 Trục tự do J2 trong robot JetArm . . . . .	67
3.31 Cấu tạo trục tự do J1 . . . . .	68
3.32 Cấu tạo trục tự do J1 . . . . .	68
3.33 Cấu tạo trục tự do J1 . . . . .	68
3.34 Cấu tạo trục tự do J1 . . . . .	69

3.35	Cấu tạo trục tự do J1 . . . . .	69
3.36	Trục tự do J1 trong robot JetArm . . . . .	69
3.37	Sơ đồ đi dây giữa các trục tự do . . . . .	70
3.38	Hình ảnh thực tế connector D-sub 25 . . . . .	70
3.39	Sơ đồ kết nối driver và động cơ của J4, J5 và J6 . . . . .	71
3.40	Sơ đồ kết nối driver và động cơ của J1, J2 và J3 . . . . .	72
3.41	Sơ đồ kiến trúc hệ thống . . . . .	78
3.42	Sơ đồ các nút của game pad . . . . .	79
3.43	Manual State Machine . . . . .	81
3.44	Camera USB . . . . .	82
3.45	So sánh 5 loại mô hình YOLOv8 . . . . .	87
3.46	Huấn luyện các quân cờ riêng biệt với nhau . . . . .	87
3.47	Sắp xếp các quân cờ riêng biệt . . . . .	88
3.48	Sắp xếp ngẫu nhiên các quân cờ . . . . .	88
3.49	Hòa trộn quân đen và đỏ với nhau . . . . .	88
3.50	Một số ảnh dùng cho việc hiệu chỉnh camera . . . . .	92
3.51	Phát hiện góc cạnh bàn cờ mẫu . . . . .	92
3.52	Máy trạng thái của chế độ Auto . . . . .	94
3.53	Máy trạng thái cho việc chuyển đổi mode . . . . .	96
3.54	Máy trạng thái cho việc xử lý tín hiệu UART . . . . .	96
3.55	Toàn bộ hệ thống điều khiển cánh tay . . . . .	97
3.56	Hệ thống điều khiển cánh tay và trung tâm xử lý Jetson Nano . . . . .	97
4.3	Kết quả nhận diện thực tế 1 . . . . .	101
4.4	Kết quả nhận diện thực tế 2 . . . . .	101
4.5	Kết quả nhận diện và gấp quân cờ . . . . .	102
4.6	Robot ở vị trí home lúc khởi động . . . . .	103
4.7	Robot nghiên một góc để nhìn phía bên kia bàn cờ . . . . .	104
4.8	Robot gấp quân tượng đen . . . . .	105
4.9	Robot đánh cờ vào vị trí mới . . . . .	106
4.10	Robot quay về vị trí home sau khi đã đi nước cờ thành công . . . . .	106
4.11	Tỉ lệ nhận diện các quân cờ trên 100 lần thực nghiệm . . . . .	108
4.12	Tỉ lệ gấp trúng và thả các quân cờ trên thực nghiệm 100 lần . . . . .	109
4.13	Kiến trúc hệ thống của bài báo [28] . . . . .	110
4.14	Kiến trúc hệ thống của bài báo [29] . . . . .	111
4.15	Hiện thực phần cứng của bài báo [30] . . . . .	111

---

## Danh sách bảng

---

2.1	Bảng DH của robot JetArm . . . . .	27
2.2	Liên hệ giữa vi bước và mô men . . . . .	35
3.1	Các nút điều khiển Gamepad . . . . .	80
4.1	Tỉ lệ nhận diện các quân cờ trên 90 lần thực nghiệm ở các vị trí khác nhau . . . . .	107
4.2	Tỉ lệ gấp trúng và thả các quân cờ trên thực nghiệm 100 lần . . . . .	108

# CHƯƠNG 1

## GIỚI THIỆU TỔNG QUAN

### 1.1 Robot trong đời sống hiện nay

Trong thế kỷ 21 đầy biến động, sự tiến bộ nhanh chóng trong lĩnh vực công nghệ, khoa học - kỹ thuật đã mở ra nhiều cánh cửa mới cho sự phát triển của các lĩnh vực đầy tiềm năng. Trong số đó, robot hứa hẹn trở thành một trong những yếu tố quan trọng định hình tương lai của chúng ta [1]. Từ những con robot công nghiệp có khả năng thực hiện công việc lặp đi lặp lại đến những trợ lý ảo thông minh, sự hiện diện của robot ngày càng trở nên đa dạng và quan trọng trong đời sống hàng ngày.

Trong đời sống hàng ngày, chúng ta có thể thấy sự hiện diện của robot trong nhiều lĩnh vực khác nhau. Tại các nhà máy và xưởng sản xuất, robot công nghiệp giúp tăng cường năng suất và giảm nguy cơ cho lao động con người trong các nhiệm vụ nguy hiểm và lặp đi lặp lại. Trong lĩnh vực y tế, robot phẫu thuật và trợ lý y tế đã mang lại những tiện ích lớn trong quá trình chăm sóc sức khỏe. Ngoài ra, robot dần trở thành một phần quan trọng trong cuộc sống hàng ngày thông qua các ứng dụng gia đình thông minh. Từ robot hút bụi tự động đến trợ lý ảo giọng nói, chúng thực sự làm thay đổi cách chúng ta tương tác với môi trường xung quanh [2].

Trong bối cảnh trên, hệ điều hành Robot (Robot Operating System - ROS) đã nổi lên như một nhân tố quan trọng thúc đẩy sự phát triển và triển khai các ứng dụng Robot hiện đại. ROS là một framework phần mềm mã nguồn mở, đang đóng vai trò quan trọng trong việc cung cấp một nền tảng linh hoạt cho việc xây dựng và phát triển các hệ thống Robot với độ phức tạp cao. Nền tảng này không chỉ cung cấp các công cụ và thư viện chuẩn để quản lý cấu trúc, giao tiếp và điều khiển Robot mà còn giúp giảm bớt thời gian và nỗ lực cần thiết cho quá trình phát triển các ứng dụng Robot [3].

Trong thời điểm hiện tại, ROS đang trải qua sự phát triển mạnh mẽ và được liên

tục cập nhật để đáp ứng với các yêu cầu ngày càng cao về độ thông minh và tính phức tạp của công việc. Các phiên bản mới như ROS2 đã được giới thiệu với những cải tiến về hiệu suất, độ tin cậy và tính mở rộng, tạo ra khả năng tích hợp, tương tác và phát triển hệ thống Robot phức tạp một cách mạnh mẽ hơn. Điều này đặt nền tảng cho việc áp dụng rộng rãi trong các lĩnh vực như y tế, công nghiệp, nông nghiệp và dịch vụ [3].

Với sự tiến triển đáng kể của ROS và sự lựa chọn ngày càng rộng rãi trong lĩnh vực Robot, chúng ta có thể kỳ vọng rằng ROS sẽ tiếp tục định hình sự phát triển của Robot thông minh và đa nhiệm hơn trong tương lai. Chúng không chỉ sẽ thực hiện các nhiệm vụ lặp đi lặp lại một cách hiệu quả, mà còn có khả năng học hỏi và thích ứng để đáp ứng các yêu cầu đa dạng của cuộc sống. Sự phát triển của Robot hứa hẹn mở ra những cánh cửa mới cho sự tiến bộ của công nghệ và mang lại nhiều lợi ích cho toàn xã hội.

## 1.2 Lý do chọn đề tài

Như đã đề cập ở phần trước, hiện nay sự hiện diện của robot ngày càng nhiều và đóng vai trò quan trọng trong việc tự động hóa các quy trình sản xuất công nghiệp, nông nghiệp, dịch vụ và gia đình. Vì thế, việc lựa chọn đề tài về việc "**Phát triển motor driver trên nền tảng ROS sử dụng mạch nhúng Jetson**" xuất phát từ sự nhận thức về xu hướng công nghệ trong tương lai hứa hẹn số lượng robot sẽ gia tăng nhanh chóng, đóng vai trò quan trọng trong đời sống hằng ngày. Đồng thời, nhờ việc hiểu về Robot, nhóm có thể tiếp cận dễ dàng hơn đến các chủ đề khác như là Digital Twin, Smart Cities, Robotic Automation,...[4] Việc này đồng nghĩa với việc nghiên cứu và phát triển một hệ thống điều khiển động cơ với sự tích hợp của ROS và mạch nhúng Jetson, mở ra một loạt các khả năng mới và tiềm năng đổi mới trong việc ứng dụng robot di động và các hệ thống tự động.

Đối với nhóm, lựa chọn này không chỉ là một cơ hội để khám phá sâu rộng về robot, điều khiển và tích hợp hệ thống mà còn là một bước tiến quan trọng trong việc hiểu rõ hơn về cách ROS và Jetson có thể tương tác để tạo ra các ứng dụng robot thông minh. Sự kết hợp giữa công nghệ điều khiển động cơ và khả năng tích hợp mạnh mẽ của ROS có thể mở ra những cánh cửa mới cho ứng dụng trong nhiều lĩnh vực, từ dịch vụ tự động đến y tế và công nghiệp. Đồng thời, việc phát triển motor driver trên nền tảng Jetson cũng đáp ứng một nhu cầu ngày càng tăng về hệ thống nhúng mạnh mẽ và hiệu quả năng lượng trong các ứng dụng di động. Điều này có thể mang lại những đóng góp quan trọng cho sự phát triển của robot di động và các thiết bị tự động thông minh.

Song hành với sự phát triển của robot, việc phát triển "bộ não" mạnh mẽ là bước quan trọng để họ có thể tự học và tương tác với môi trường. Mạch nhúng Jetson, với sức mạnh tính toán và khả năng xử lý đồng thời, trở thành cột mốc quan trọng trong việc xây dựng "trí óc" cho robot. Bộ não của robot không chỉ là nơi lưu trữ mã lệnh, mà là trí óc chúng ta muốn chúng sở hữu. Làm thế nào để giúp robot hiểu biểu đồ môi trường, nhận diện vật thể và dự đoán hành vi của con người? Đâu là câu trả lời? Nó ẩn chứa trong cách kết hợp khả năng học máy và xử lý dữ liệu lớn trên nền tảng Jetson. Quá trình này không chỉ đặt ra những thách thức kỹ thuật trong triển khai thuật toán trí tuệ nhân tạo mà còn mở ra những cánh cửa cho sự đổi mới trong các ứng dụng thực tế.

Robot không chỉ là công cụ thực hiện công việc cụ thể, mà còn là đối tác linh hoạt có khả năng học hỏi từ môi trường và tương tác tự nhiên với người sử dụng. Kết hợp giữa motor driver và "bộ não" mạnh mẽ không chỉ là về việc kiểm soát chuyển động,

mà còn là sự mở rộng của khả năng nhận thức và quyết định của robot. ROS không chỉ quản lý và tương tác giữa các phần của robot mà còn tạo điều kiện thuận lợi cho việc thử nghiệm, mô phỏng và triển khai ứng dụng trên nền tảng Jetson một cách linh hoạt và hiệu quả.

Cuối cùng, hành trình phát triển motor driver và "bộ não" cho robot không chỉ là việc nghiên cứu và phát triển kỹ thuật mà còn là sự đóng góp vào sự tiến bộ của cộng đồng robot học và trí tuệ nhân tạo. Đây là hành trình của sự đổi mới và khám phá, nơi ý tưởng trở thành hiện thực và robot trở thành người bạn đồng hành thông minh, đồng cảm trong cuộc sống hàng ngày.

## 1.3 Đối tượng nghiên cứu

Đối tượng nghiên cứu và làm việc chính của đề tài là cánh tay robot JetArm với 6 bậc chuyển động tự do (Six Degrees of Freedom - 6DoF). Với khả năng chuyển động theo sáu bậc tự do, cánh tay robot này không chỉ có khả năng di chuyển dọc theo ba trục (x, y, z) mà còn xoay quanh ba trục góc (roll, pitch, yaw), tạo ra một phạm vi linh hoạt và đa dạng của các cử động [5]. Cánh tay robot này được thiết kế dựa trên hình dáng và các cử động của cánh tay người, do đó có thể dễ dàng mô phỏng lại các chuyển động của cánh tay người, từ đó đáp ứng đa dạng các ứng dụng thực tiễn khác nhau.



Hình 1.1: Robot JetArm

Robot JetArm là một công cụ mạnh mẽ được thiết kế để thực hiện nhiều loại công việc khác nhau trong các ngành công nghiệp như sản xuất, đóng gói, và thậm chí trong nghiên cứu và phát triển. Khả năng điều khiển linh hoạt mở ra nhiều cơ hội cho việc thí nghiệm và ứng dụng, từ việc xử lý vật liệu trong môi trường công nghiệp đến việc thực hiện các thao tác phức tạp trong y học [6], hoặc nghiên cứu khoa học và cả những công việc nguy hiểm như tàu lặn thăm dò, các hoạt động có sự hiện diện của phóng xạ [7], chất nổ,

...

Với sự kết hợp của độ chính xác cao và khả năng linh hoạt, robot JetArm không chỉ là một công cụ hữu ích cho sản xuất công nghiệp mà còn là một lựa chọn lý tưởng cho những nhu cầu chuyên sâu đòi hỏi sự linh hoạt và đa dạng trong các ứng dụng robot đương đại. Được thiết kế để làm tăng năng suất và chính xác, robot JetArm đưa chúng ta một bước gần hơn đến tương lai của tự động hóa thông minh và linh hoạt.

Để có thể điều khiển robot một cách hiệu quả và đáp ứng được các yêu cầu xử lý phức tạp đòi hỏi một bộ não xử lý đủ mạnh nhưng cũng cần đáp ứng được yêu cầu gọn nhẹ để có thể dễ dàng tích hợp. Do vậy, cánh tay robot được trang bị một board mạch nhúng Jetson , một sản phẩm máy tính nhỏ của Nvidia. Được xây dựng trên nền tảng này, robot JetArm có khả năng tích hợp các công nghệ trí tuệ nhân tạo tiên tiến như học sâu và thị giác máy tính. Sự hiệu quả của Jetson Nano cung cấp đủ sức mạnh tính toán để thực hiện các tác vụ AI đương đại, bao gồm nhận diện đối tượng, phân loại tự động, nhận diện khuôn mặt, cũng như nhận diện cử chỉ và nhiều chức năng khác [8]. Nhờ vào Jetson Nano, cánh tay robot không chỉ đơn giản là một sản phẩm cơ khí thông thường mà còn có thể xem như là một giải pháp thông minh và linh hoạt trong việc áp dụng trí tuệ nhân tạo vào các ứng dụng thực tế.



Hình 1.2: Jetson Nano



Hình 1.3: Stepper Motor Driver TB6600

Cuối cùng, thiết bị đóng vai trò quan trọng và trực tiếp đến việc thực hiện các thao tác cử động của robot là các step motor - động cơ bước. Để có thể điều khiển robot một cách mượt mà và linh hoạt, cần phải có những thiết bị trung gian chuyên dụng đảm nhiệm việc điều khiển các motor này - các motor driver. Stepper motor driver, hay còn được gọi là bộ điều khiển động cơ bước, đóng vai trò quan trọng trong hệ thống điều khiển động, mang lại sự chính xác và kiểm soát linh hoạt cho các ứng dụng khác nhau. Một trong những đặc điểm nổi bật của stepper motor driver là khả năng kiểm soát động cơ một cách chính xác theo từng bước nhất định, giúp đạt được độ chính xác cao trong các ứng dụng yêu cầu điều khiển chính xác và đa dạng [9]. Điều này rất quan trọng trong các lĩnh vực như máy in 3D, máy CNC, và các hệ thống tự động hóa công nghiệp.

Với sự tiện lợi và linh hoạt, stepper motor driver chính là "não bộ" của các hệ thống điều khiển động, mang lại hiệu suất và độ tin cậy cao trong quá trình vận hành. Sự kết hợp giữa độ chính xác và khả năng điều khiển linh hoạt đã làm cho stepper motor driver trở thành một thành phần quan trọng và không thể thiếu trong nhiều ứng dụng công nghiệp và dân dụng.

## 1.4 Phạm vi và mục tiêu của Đồ án

### 1.4.1 Phạm vi

Trong đồ án này, chúng tôi tập trung trước hết vào việc xây dựng hệ thống cơ điện hoàn chỉnh cho cánh tay robot. Tiếp theo là xây dựng device driver dựa trên nền tảng ROS sử dụng board nhúng Jetson Nano để có thể điều khiển được các khớp xoay của cánh tay robot. Thông qua việc sử dụng một thiết bị cụ thể là tay cầm gamepad để điều khiển robot, chúng tôi có thể chứng minh device driver có tính bao đóng và có thể cung cấp các API để các ứng dụng khác có thể dễ dàng điều khiển cánh tay theo các cử động khác nhau một cách linh hoạt. Bên cạnh đó, với những công việc đòi hỏi độ chính xác cao, nhóm sẽ tập trung hiện thực chức năng chơi cờ tướng tự động để có thể chứng minh device driver hoàn toàn có thể điều khiển cánh tay với độ chính xác cao.

### 1.4.2 Mục tiêu

Trong giai đoạn Đồ án môn học Kỹ thuật Máy tính, nhóm đã nghiên cứu và đề xuất những mục tiêu sau:

- ① Nghiên cứu, hoàn thiện hệ thống cơ điện cho cánh tay Robot
- ② Tìm hiểu, nghiên cứu và ứng dụng lý thuyết về động học thuận và động học nghịch vào việc điều khiển robot JetArm.
- ③ Lập trình sử dụng stepper motor driver điều khiển robot JetArm.
- ④ Tích hợp chức năng điều khiển bằng Gamepad cho robot JetArm.
- ⑤ Tích hợp ROS vào cánh tay Robot
- ⑥ Hiệu chuẩn (Calibration) độ chính xác của camera
- ⑦ Phát triển được model nhận diện quân cờ tướng và tích hợp vào cánh tay Robot
- ⑧ Xây dựng thuật toán đánh cờ tướng và tích hợp vào cánh tay Robot
- ⑨ Đánh giá kết quả hiện thực trong Đồ án tốt nghiệp và định hướng phát triển trong tương lai.

### 1.4.3 Giới hạn đề tài

#### 1. Nghiên cứu và phát triển robot JetArm:

- Nghiên cứu và trình bày về cấu trúc, đặc điểm và nguyên tắc hoạt động của robot JetArm dựa trên phần cơ khí được cung cấp bởi AC Lab. Dánh giá các khó khăn và hạn chế hiện tại của việc điều khiển cánh tay robot.
- Nghiên cứu và ứng dụng các nguyên tắc của động học trong việc điều khiển cánh tay robot.
- Lập trình các chức năng bằng ngôn ngữ lập trình Python và driver bằng ngôn ngữ C++ trên nền tảng tích hợp thêm ROS (Robot Operating System).
- Robot JetArm sẽ được lắp đặt thêm camera và đầu hút để thực hiện chức năng tự động gấp thả vật thể.

#### 2. Nghiên cứu và ứng dụng xử lý ảnh và thị giác máy tính vào robot JetArm:

- Tìm hiểu các phương pháp và thuật toán trong lĩnh vực Trí tuệ nhân tạo, bao gồm học máy, học sâu và học tăng cường, cùng với các ứng dụng trong điều khiển robot phù hợp với robot JetArm.
- Tìm hiểu về thị giác máy và các kỹ thuật nhận dạng vật thể, nhận dạng chuyển động, v.v., và cách áp dụng chúng vào hệ thống điều khiển robot.
- Mô hình học máy được sử dụng phải hoạt động được trên Jetson Nano.
- Nghiên cứu và áp dụng giải thuật đánh cờ vào robot.

#### 3. Kiểm tra và đánh giá hiệu năng của hệ thống:

- Thời gian phản hồi của các chức năng được dựa trên giới hạn của các thành phần phần cứng tương ứng.
- Dựa theo giới hạn từ cơ cấu phần cứng của robot JetArm, những quân cờ được chơi sẽ phải ở trong phạm vi của cánh tay robot.
- Kiểm tra và đánh giá model nhận diện quân cờ.
- Kiểm tra và đánh giá độ chính xác trong việc gấp thả quân cờ.
- Tiến hành thử nghiệm và đánh giá hiệu suất của hệ thống trong các tình huống thực tế.
- So sánh và phân tích kết quả dựa vào số liệu từ các quá trình thử nghiệm.

## 1.5 Cấu trúc của bài báo cáo Đồ án

Cấu trúc của bài báo cáo Đồ án tốt nghiệp bao gồm các chương sau:

- **Chương 1. Giới thiệu tổng quan:**

Chương này giới thiệu đề tài và mục tiêu của đồ án môn học. Nội dung của chương trình bày sự quan trọng của đề tài và lý do lựa chọn nghiên cứu đồ án này. Ngoài ra, phần này cũng cung cấp cái nhìn tổng quan về tình hình nghiên cứu liên quan và các công trình liên quan đã được thực hiện trước đây.

- **Chương 2. Kiến thức nền tảng:**

Chương này trình bày các kiến thức cơ bản liên quan đến đồ án môn học. Nội dung bao gồm cơ sở lý thuyết, phương pháp và nguyên lý hoạt động liên quan đến đề tài. Ngoài ra, phần này cũng trình bày về các công nghệ, công cụ và ngôn ngữ lập trình được sử dụng trong quá trình thực hiện đồ án.

- **Chương 3. Thiết kế và Hiện thực hệ thống:**

Chương này trình bày chi tiết về quá trình phát triển một chức năng hoàn thiện. Nội dung bao gồm việc thiết kế hệ thống, nghiên cứu và sử dụng các thiết bị thích hợp. Bên cạnh đó, chương này trình bày chi tiết về quá trình hiện thực từng tính năng trong hệ thống. Ngoài ra cũng cung cấp các hướng dẫn về từng giai đoạn nhằm mục đích phục vụ cho việc học tập, nghiên cứu và phát triển của các khóa sinh viên sau.

- **Chương 4. Đánh giá kết quả:**

Chương này trình bày về kết quả sau quá trình hiện thực hệ thống. Sau đó, hệ thống được thử nghiệm trong tình huống thực tế. Từ các số liệu từ thực nghiệm, nhóm sẽ tiến hành so sánh, phân tích và đánh giá những chức năng trong hệ thống về hiệu suất, tính hoàn thiện, tính khả thi,...

- **Chương 5. Tổng kết và hướng phát triển:**

Chương này tổng hợp tất cả những thành tựu đạt được và những vấn đề khó khăn trong suốt quá trình hiện thực hệ thống. Từ những đánh giá và những vấn đề còn chưa giải quyết được, nhóm sẽ đề xuất các hướng phát triển và cải thiện trong tương lai cho hệ thống.

# CHƯƠNG 2

## KIẾN THỨC NỀN TẢNG

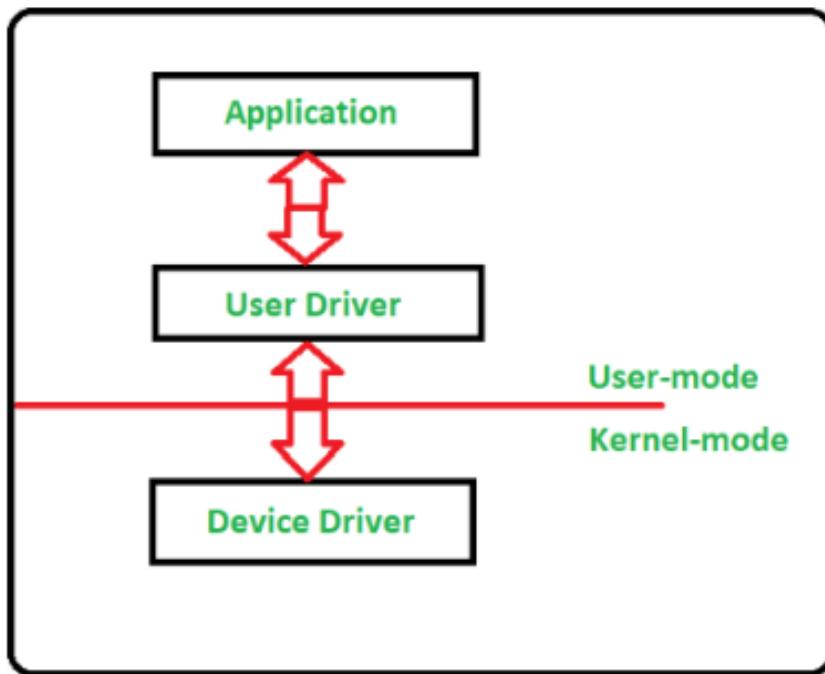
Trong chương này, nhóm sẽ đề cập đến những kiến thức liên quan và cần thiết về việc phát triển motor driver trên nền tảng ROS sử dụng mạch nhúng Jetson.

### 2.1 Kiến thức cơ bản về Device Driver

#### 2.1.1 Sơ lược về Device Driver

Device Driver trong lĩnh vực máy tính là một phần mềm chuyên dụng được thiết kế để điều khiển một phần cứng cụ thể, giúp tương tác giữa các thiết bị phần cứng khác nhau và hệ điều hành máy tính. Trình điều khiển này thường tương tác với phần cứng thông qua các hệ thống con hoặc bus được kết nối với thiết bị. Nhiệm vụ chính của trình điều khiển là đảm bảo rằng hệ điều hành có thể hiểu và tương tác đúng đắn với phần cứng cụ thể mà nó điều khiển.

Mỗi thiết bị phần cứng đều có kết nối và các quy tắc kết nối riêng, và device driver được xây dựng để chuyển đổi yêu cầu và lệnh từ hệ điều hành thành các hành động cụ thể mà phần cứng hiểu được. Điều này giúp đảm bảo rằng mọi phần của máy tính có thể làm việc cùng nhau một cách hiệu quả. Ví dụ, nếu bạn kết nối một máy in vào máy tính của mình, device driver của máy in sẽ đảm nhận vai trò chuyển đổi yêu cầu in từ hệ điều hành thành các tín hiệu và lệnh cụ thể mà máy in cần để thực hiện quá trình in. Điều này giúp đảm bảo rằng máy in hoạt động đúng cách và tương tác một cách hiệu quả với máy tính.



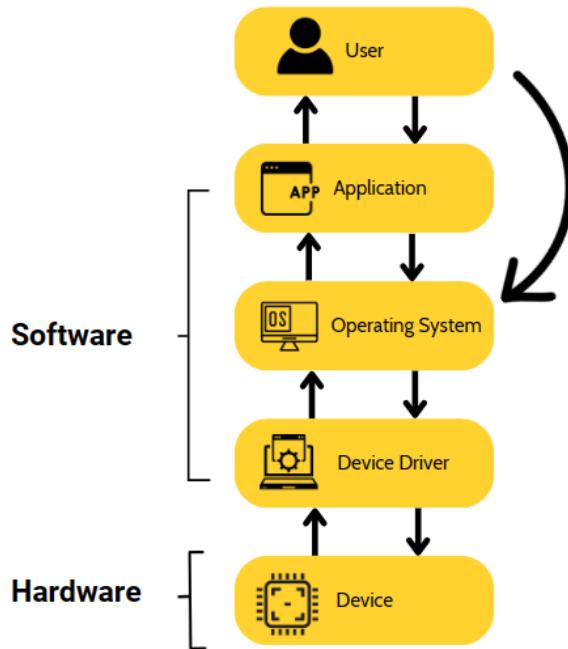
Hình 2.1: Vai trò của Device Driver trong hệ thống máy tính

Device Driver đóng vai trò rất quan trọng để hệ thống máy tính hoạt động bình thường. Không có chúng, phần cứng nhất định sẽ không hoặc rất khó để thực hiện được chức năng mong muốn, cản trở hoạt động chung của hệ thống. Thuật ngữ "Driver" và "Hardware Driver" thường được sử dụng thay thế cho nhau để chỉ thành phần thiết yếu này [10].

Tóm lại, Device Driver là một phần quan trọng của hệ thống máy tính, giúp cải thiện sự tương tác và tương thích giữa phần mềm và phần cứng, đồng thời đảm bảo rằng mọi thành phần của máy tính hoạt động một cách mượt mà và hiệu quả.

### 2.1.2 Quá trình hoạt động

Device Driver phụ thuộc vào các lệnh từ hệ điều hành để truy cập và thực hiện các hành động cụ thể đối với thiết bị. Khi nhận được lệnh từ hệ điều hành, Device Driver sẽ thực hiện các hành động xác định đối với phần cứng tương ứng. Đồng thời, chúng cũng truyền đầu ra hoặc trạng thái/thông báo từ thiết bị phần cứng về hệ điều hành để thông báo về kết quả của các hành động đó.



Hình 2.2: Nguyên lý hoạt động của Device Driver

Để hiểu rõ hơn, hãy tưởng tượng khi bạn kết nối một ổ đĩa USB với máy tính của mình và sau đó bạn muốn truy cập dữ liệu trên ổ đĩa USB, hệ điều hành sẽ sử dụng Device Driver của ổ đĩa USB để thực hiện các thao tác đọc và ghi dữ liệu. Driver này sẽ sử dụng các hướng dẫn của hệ điều hành để giao tiếp với ổ đĩa USB và đảm bảo rằng dữ liệu được truyền đúng cách. Sau khi hoàn thành các thao tác đọc và ghi, Driver sẽ gửi kết quả thông điệp hoặc trạng thái từ ổ đĩa USB đến hệ điều hành. Nếu mọi thứ diễn ra suôn sẻ, trình điều khiển có thể báo hiệu hệ điều hành rằng thao tác đã thành công và dữ liệu có thể được sử dụng. Ngược lại, nếu có lỗi nào đó, trình điều khiển có thể thông báo về sự cố đó để hệ điều hành có thể xử lý tương ứng [10].

Tóm lại, Device Driver không chỉ làm nhiệm vụ thực hiện các hành động cụ thể trên thiết bị phần cứng mà còn thực hiện giao tiếp với hệ điều hành và truyền đạt thông tin về kết quả của các hành động đó. Điều này làm cho mọi quá trình tương tác giữa phần mềm và phần cứng trở nên mượt mà và hiệu quả.

## 2.2 Kiến thức cơ bản về ROS

### 2.2.1 Sơ lược về ROS

ROS (Robot Operating System) là một hệ thống phần mềm mở nguồn linh hoạt, được chế tạo để đơn giản hóa quá trình phát triển ứng dụng Robot. Nó không chỉ cung cấp một bộ công cụ và thư viện đa dạng mà còn tạo điều kiện thuận lợi cho việc chia sẻ và tái sử dụng thành phần giữa các ứng dụng, giúp tối ưu hóa tốc độ phát triển và giảm chi phí.

Tính năng nổi bật của ROS là khả năng hỗ trợ nhiều ngôn ngữ lập trình, bao gồm C++, Python, Java và Matlab, mang lại sự linh hoạt cho những nhà phát triển. Điều này cho phép họ sử dụng ngôn ngữ mà họ thoải mái nhất để sáng tạo các ứng dụng Robot. ROS cũng nổi bật với khả năng tương thích với nhiều nền tảng phần cứng khác nhau, từ Robot di động đến Robot công nghiệp, máy bay không người lái và nhiều thiết bị khác.

Một đặc điểm quan trọng khác của ROS là khả năng phân tán ứng dụng, giúp các nút ROS có thể chạy trên nhiều máy tính khác nhau và tương tác thông qua mạng. Điều này mở ra khả năng tạo ra các ứng dụng Robot phức tạp và mở rộng một cách thuận tiện và mạnh mẽ.

Để bắt đầu với việc sử dụng ROS, chúng ta cần có kiến thức sâu rộng về lập trình và hiểu biết vững về các khái niệm trong lĩnh vực Robot học. Dưới đây là một số kiến thức kỹ thuật quan trọng để bắt đầu làm việc với ROS:

- ROS hỗ trợ một loạt các ngôn ngữ lập trình khác nhau, bao gồm C++, Python, Java, Lua, Matlab, Octave, và nhiều ngôn ngữ khác. Tuy nhiên, trong cộng đồng ROS, C++ và Python được xem là hai ngôn ngữ phổ biến nhất và thường được ưu tiên sử dụng. Cả hai đều mang lại sự linh hoạt và mạnh mẽ, giúp nhà phát triển tận dụng tối đa các tính năng của ROS một cách hiệu quả.
- Để làm việc hiệu quả với ROS, người sử dụng cần có kiến thức cơ bản về các khái niệm trong lĩnh vực Robot học, bao gồm hệ tọa độ, khung tọa độ, quy hoạch động, và lập trình điều khiển chuyển động,...
- Kiến trúc của ROS được xây dựng theo mô hình phân tán, trong đó mỗi "nút" (node) đại diện cho một thành phần cụ thể trong hệ thống Robot. Các nút có khả năng tương tác với nhau thông qua các "chủ đề" (topic) hoặc "dịch vụ" (service), tạo nên một môi trường linh hoạt và có khả năng mở rộng.

- Các công cụ hỗ trợ trong ROS, bao gồm ROS Command Line Tools, RViz, rosbag, và nhiều ứng dụng khác, chính là những trợ lực quan trọng cho nhà phát triển trong quá trình xây dựng, thử nghiệm, và kiểm thử các ứng dụng Robot. Đây là những công cụ không thể thiếu giúp đảm bảo tính đồng bộ và hiệu suất ổn định của ứng dụng Robot.
- ROS cung cấp nhiều gói phần mềm quan trọng để hỗ trợ nhà phát triển trong quá trình xây dựng các ứng dụng Robot. Một số gói phần mềm đáng chú ý bao gồm move\_base, navigation, gmapping, Robot\_localization, và nhiều gói khác. Các gói này cung cấp các chức năng và thuật toán chuyên sâu trong lĩnh vực điều hướng, tạo bản đồ, và xác định vị trí, giúp nhà phát triển triển khai và tối ưu hóa chức năng của robot một cách thuận tiện.
- ROS có thể linh hoạt sử dụng trên các hệ thống nhúng như Raspberry Pi, BeagleBone Black, giúp xây dựng các Robot nhỏ gọn và di động. Việc này mở ra khả năng triển khai ROS trên các thiết bị có tài nguyên hạn chế, thích hợp cho các ứng dụng Robot tương đối nhỏ và nhẹ. Điều này giúp nhà phát triển tận dụng các ưu điểm của các hệ thống nhúng nhỏ gọn để tạo ra các Robot di động với khả năng tích hợp linh hoạt và chi phí thấp.

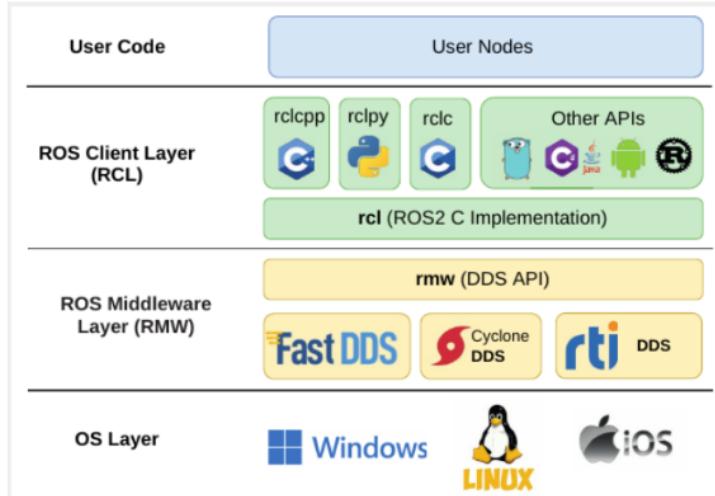
Tóm lại, để làm việc hiệu quả với ROS, người ta cần có kiến thức vững về lập trình và các khái niệm trong Robot học. Đồng thời, việc hiểu rõ về kiến trúc của ROS, sử dụng các công cụ quan trọng cùng các gói phần mềm, là chìa khóa để phát triển thành công các ứng dụng Robot đa dạng và linh hoạt.

## 2.2.2 Kiến trúc của ROS

ROS là một hệ thống phân tán được cấu thành từ nhiều phần mềm được gọi là "gói" (packages), và những gói này được phát triển bởi cộng đồng người dùng ROS. Kiến trúc chính của ROS bao gồm ba phần quan trọng:

- **ROS Graph:** Quản lý kết nối giữa các "nút" trong ROS, bao gồm ROS Master, ROS Nodes, ROS Topics, ROS Services, và ROS Parameters. Mỗi node là một đơn vị tính toán độc lập có thể gửi và nhận tin nhắn qua ROS Topics hoặc ROS Services. ROS Master duy trì danh sách các nút, topics, và services trong hệ thống, quản lý thông tin kết nối giữa chúng.
- **ROS Communication:** Quản lý truyền thông giữa các nút trong ROS sử dụng giao thức truyền thông được gọi là TCPROS (Transmission Control Protocol ROS). Giao thức này cho phép truyền các tin nhắn giữa các nút trực tiếp hoặc thông qua ROS Master.

- **ROS Packages:** Cung cấp các tính năng và chức năng cần thiết cho các ứng dụng trong ROS. Các gói ROS có thể bao gồm thư viện, phần mềm, tài liệu, và các file cấu hình. ROS cung cấp nhiều gói cho nhiều nhiệm vụ khác nhau như điều khiển Robot, xử lý hình ảnh, điều khiển động cơ, và nhiều ứng dụng khác.



Hình 2.3: Kiến trúc của ROS2 [11]

Khác với ROS, ROS2 được thiết kế với các lớp (layer) khác nhau như hình trên:

- **OS Layer:** ROS 2 chạy trên các hệ điều hành khác nhau (Linux, Windows, Mac).
- **ROS Middleware Layer (RMW):** ROS 2 được xây dựng trên Data Distribution Service (DDS), một tiêu chuẩn mở cho giao tiếp được triển khai qua UDP. Vì có thể sử dụng các nhà cung cấp DDS khác nhau, một giao diện DDS API được cung cấp, gọi là ROS Middleware interface (rmw). Việc sử dụng DDS cho phép trao đổi thông tin giữa các quy trình với các đặc tính thời gian thực, khả năng bảo mật và chất lượng dịch vụ tùy chỉnh của mỗi kết nối.
- **ROS client library (RCL):** Các chức năng cơ bản của tất cả các thành phần ROS 2 được triển khai trong một thư viện C đơn gọi là rcl; sau đó, các thư viện rclcpp và rclpy được sử dụng để điều chỉnh chức năng này theo đặc điểm cụ thể của mỗi ngôn ngữ, tương ứng là C++ và Python. Các API cho các ngôn ngữ khác cũng có thể được triển khai. Điều này cho phép các chức năng mới sẽ sớm có sẵn trong bất kỳ ngôn ngữ nào.
- **User Code:** ROS 2 cung cấp một quy ước về cách viết các node bằng cách buộc các node được triển khai phải kế thừa từ một lớp Node đã có tất cả các chức năng cần thiết. Điều này giúp tiết kiệm rất nhiều thời gian, tạo ra một cấu trúc modular tốt và giúp giảm bớt sự phức tạp trong việc hợp tác giữa các dự án.

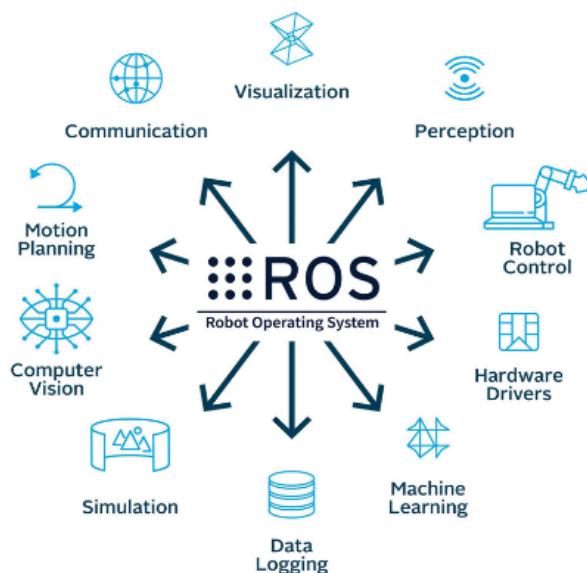
Tổng thể, kiến trúc của ROS tạo điều kiện cho việc tương tác giữa các nút và gói ROS một cách dễ dàng và hiệu quả. Việc phân tán các nút và tính năng của ROS cũng mang lại khả năng mở rộng linh hoạt, cho phép người dùng nhanh chóng mở rộng ứng dụng của họ bằng cách thêm các nút hoặc gói mới.

### 2.2.3 Ứng dụng trong thực tế

Các ứng dụng của ROS không ngừng mở rộng, đa dạng và lan rộng sang nhiều lĩnh vực quan trọng trong cuộc sống hiện đại. Trong lĩnh vực lắp ráp và tự động, ROS không chỉ giúp kiểm soát Robot trong quá trình lắp ráp sản phẩm mà còn đóng vai trò quan trọng trong việc tối ưu hóa quá trình sản xuất. Khả năng tự động hóa và tối ưu hóa được ROS mang lại có thể giúp tăng cường độ chính xác và hiệu suất, đồng thời giảm thiểu sai số và lỗi trong quá trình sản xuất.

ROS chứng minh sức mạnh của mình trong việc phát triển và triển khai Robot di động và máy bay không người lái. Tính linh hoạt và khả năng tương tác với môi trường tự nhiên giúp ROS tạo ra những giải pháp hiệu quả cho giám sát môi trường, thám hiểm, và thậm chí tham gia vào các nhiệm vụ cứu hộ và khẩn cấp.

Trong lĩnh vực y tế và chăm sóc sức khỏe, sự ứng dụng của ROS trở nên rất đặc biệt. Việc sử dụng Robot để hỗ trợ bệnh nhân trong quá trình điều trị và chăm sóc sức khỏe mở ra những khả năng mới. Các Robot có thể thực hiện nhiệm vụ như hỗ trợ di chuyển cho bệnh nhân, giúp thực hiện các bài tập phục hồi và thậm chí cung cấp hỗ trợ tinh thần thông qua tương tác nhân loại.



Hình 2.4: Ứng dụng của ROS [11]

Cuối cùng, trong lĩnh vực giáo dục và nghiên cứu Robot học, ROS không chỉ là một công cụ mạnh mẽ cho việc phát triển ứng dụng Robot mà còn là một nguồn tài nguyên lý tưởng. Với sự linh hoạt và tính mở rộng của mình, ROS cung cấp môi trường thích hợp cho việc thử nghiệm và nghiên cứu, giúp định hình tương lai của Robot học và ứng dụng của chúng trong nhiều lĩnh vực đời sống.

### 2.2.4 Tiềm năng phát triển trong tương lai

Trong lĩnh vực Robot học, ROS không chỉ là một hệ thống phần mềm mã nguồn mở mà còn là công cụ hỗ trợ quan trọng cho nghiên cứu và phát triển Robot. Với tính linh hoạt và tính năng ưu việt, ROS đã trở thành nguồn động lực cho sự tiến bộ nhanh chóng trong cộng đồng Robot học toàn cầu. Tuy nhiên, để đáp ứng với sự phức tạp ngày càng tăng, ROS cần khai thác tiềm năng phát triển và liên tục nâng cao hiệu suất cũng như tích hợp công nghệ mới. Sự đầu tư vào nghiên cứu sẽ giúp ROS duy trì vị thế là một công cụ tiên tiến và động lực trong cộng đồng Robot học.

Các tiềm năng phát triển của ROS:

- **Tăng cường tính khả thi giải:** Hiện tại, ROS cung cấp các công cụ giúp giải thích hoạt động của Robot và hiển thị dữ liệu một cách trực quan. Tuy nhiên, để nâng cao sự hiểu biết của người sử dụng về hoạt động của Robot, ROS có thể được phát triển thêm để cung cấp giải thích kết quả và quá trình làm việc của Robot một cách cụ thể và dễ hiểu hơn. Điều này giúp tăng cường tính khả thi giải, giảm sự phức tạp và hỗ trợ người sử dụng trong việc tương tác và quản lý Robot một cách hiệu quả.
- **Tích hợp các hệ thống học máy:** Mặc dù ROS đã tích hợp tốt với các hệ thống phân tích hình ảnh và nhận dạng giọng nói, nhưng để nâng cao khả năng phân tích dữ liệu, ROS có thể được cải thiện bằng cách tích hợp các hệ thống học máy trong tương lai. Việc này sẽ giúp Robot có khả năng phân tích dữ liệu một cách nhanh chóng và chính xác hơn, đồng thời tăng cường hiệu suất và tính năng của Robot.
- **Hỗ trợ tốt hơn cho Robot hợp tác:** Sự tương tác giữa nhiều Robot với nhau đang trở thành một xu hướng quan trọng trong lĩnh vực Robot học. Để hỗ trợ tốt hơn cho việc này, ROS có thể được tối ưu hóa trong tương lai. Điều này có thể bao gồm các cải tiến nhằm giúp các Robot hợp tác có khả năng phối hợp với nhau một cách nhanh chóng và hiệu quả hơn, tạo ra một môi trường làm việc tích cực và hiệu quả cho các dự án Robot cùng làm việc với nhau.

- **Tăng cường tính linh hoạt:** Mặc dù ROS hiện đang có khả năng tích hợp và điều khiển nhiều loại Robot khác nhau, nhưng để đáp ứng nhu cầu ngày càng đa dạng của người dùng, ROS có thể được tăng cường tính linh hoạt. Cụ thể, có thể cải tiến ROS để hỗ trợ điều khiển và tích hợp với nhiều loại Robot khác nhau như drone, Robot hút bụi, Robot y tế và Robot công nghiệp. Điều này sẽ mở rộng khả năng ứng dụng của ROS và tạo ra một hệ sinh thái Robot linh hoạt và đa dạng.

Trên đây là những tiềm năng phát triển của ROS trong tương lai. Những cải tiến này sẽ giúp ROS trở thành một hệ thống phần mềm Robot học linh hoạt và tiên tiến hơn, đáp ứng được nhu cầu ngày càng đa dạng của người sử dụng. Việc nghiên cứu và khai thác các tiềm năng phát triển của ROS không chỉ quan trọng cho sự tiến bộ trong lĩnh vực Robot học mà còn đóng góp vào sự phát triển mạnh mẽ của ngành công nghiệp Robot hóa.

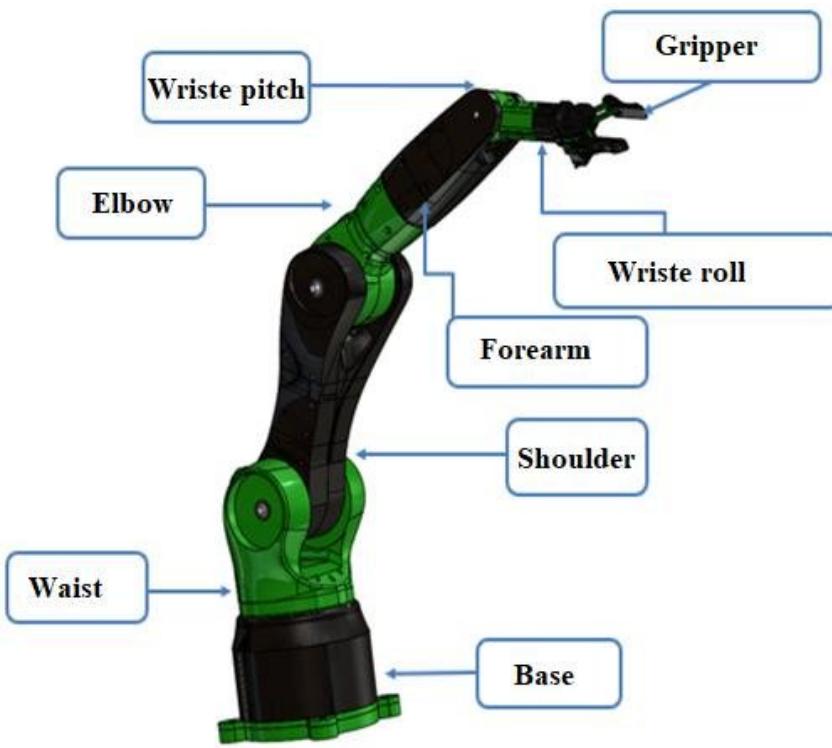
## 2.3 Tổng quan về 6DoF Robot Arm

### 2.3.1 Khái niệm

Cánh tay robot 6 bậc tự do (6-DOF robotic arm) là một loại cơ cấu robot có khả năng di chuyển và xoay theo sáu độ tự do riêng biệt. Điều này cho phép nó có khả năng định vị và điều khiển công cụ cuối cùng (end effector) trong không gian ba chiều một cách linh hoạt và đa dạng.

Cánh tay robot 6 bậc tự do thường được thiết kế với sáu khớp, mỗi khớp đại diện cho một độ tự do. Các khớp này bao gồm:

1. **Khớp cơ sở (Base joint)**: Đây là khớp đầu tiên của cánh tay robot, nằm ở gốc của cơ cấu robot và cho phép cánh tay xoay quanh trục nằm ngang.
2. **Khớp vai (Shoulder joint)**: Khớp này cho phép cánh tay robot di chuyển lên xuống theo trục thẳng đứng. Nó cho phép cánh tay robot nâng hoặc hạ công cụ cuối cùng.
3. **Khớp khuỷu tay (Elbow joint)**: Khớp này giữa vai và cổ tay cho phép cánh tay robot cong hoặc duỗi.
4. **Khớp cổ tay (Wrist joint)**: Khớp này cho phép cánh tay robot nghiêng và quay xung quanh trục dọc. Nó cung cấp khả năng xoay và nghiêng công cụ cuối cùng.
5. **Khớp quay cổ tay (Wrist rotation joint)**: Đây là khớp cho phép cánh tay robot xoay công cụ cuối cùng xung quanh trục dọc, cung cấp khả năng xoay và định hướng công cụ.
6. **Khớp công cụ cuối cùng (End effector joint)**: Đây là khớp cuối cùng của cánh tay robot và nằm trong công cụ cuối cùng. Nó cho phép công cụ cuối cùng xoay và di chuyển để đáp ứng các yêu cầu ứng dụng cụ thể.



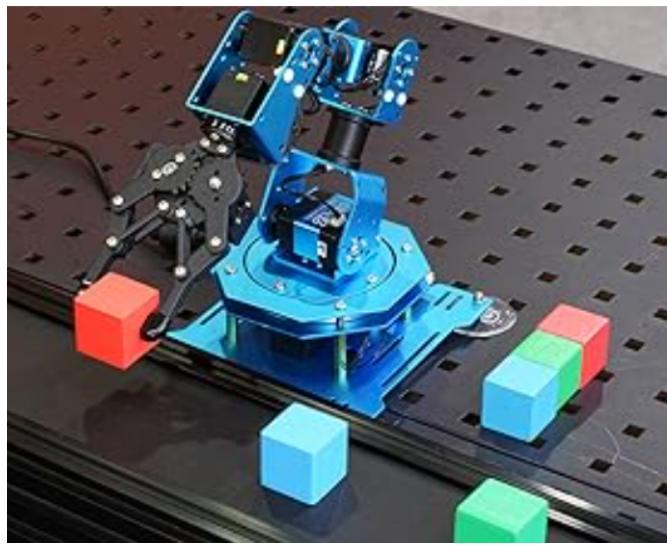
Hình 2.5: Minh họa cấu trúc của cánh tay robot 6DoF

### 2.3.2 Ứng dụng

Cánh tay robot 6 bậc tự do thường được sử dụng trong nhiều ứng dụng công nghiệp và nghiên cứu. Với sự linh hoạt của nó, nó có thể thực hiện nhiều tác vụ phức tạp như định vị chính xác, lắp ráp, hàn, vận chuyển và nhiều nhiệm vụ khác trong môi trường công nghiệp. Một số ứng dụng tiềm năng của cánh tay robot 6DoF:

#### ① Part Picking and Handling

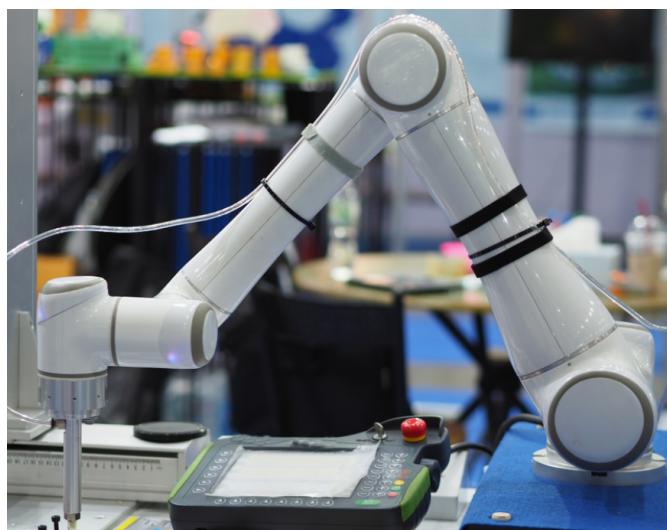
Hầu hết các loại robot có khả năng thực hiện các tác vụ gấp và đặt đơn giản, nhưng khi đến những ứng dụng đòi hỏi mức độ khéo léo cao, 6DoF Robot Arm đôi khi trở thành lựa chọn duy nhất. Một ví dụ cụ thể có thể là ứng dụng chăm sóc máy tự động, nơi mà robot cần thực hiện các chuyển động phức tạp như mở cửa, lấy bộ phận và đặt nó vào vị trí cụ thể. Các loại robot khác như SCARA, delta, và cartesian thường không đủ khả năng để thực hiện những chuyển động phức tạp này, đặc biệt là trong môi trường làm việc chật chội. Việc sử dụng một loại robot khác để thực hiện các chuyển động đòi hỏi mức độ khéo léo cao như vậy có thể dẫn đến các vấn đề không mong muốn.



Hình 2.6: 6DoF robot arm part picking and handling

## ② Painting

6DoF robot arm là sự lựa chọn hoàn hảo cho việc xử lý phun sơn sản phẩm có độ phức tạp cao. Những robot này có khả năng tích hợp với nhiều bộ phận tác động cuối khác nhau để thực hiện các nhiệm vụ đa dạng. Các ví dụ bao gồm việc sử dụng súng phun, bơm bánh răng, bộ điều chỉnh áp suất và ống góp thay đổi màu sắc. Với độ chính xác và khả năng khéo léo, chúng có thể thực hiện các chuyển động cực kỳ chính xác, cần thiết để hoàn thiện các chi tiết sản phẩm. Đặc biệt, trong các ứng dụng yêu cầu phạm vi tiếp cận rộng, 6DoF robot arm trở thành lựa chọn phổ biến, trong khi một số loại robot khác như SCARA và delta thường gặp hạn chế về kích thước trong các ứng dụng như vậy.



Hình 2.7: Painting bằng 6DoF robot arm

### ③ Material Removal

Ứng dụng loại bỏ vật liệu là một thách thức đặc biệt phức tạp, đòi hỏi sức mạnh và độ chính xác cao trong môi trường nguy hiểm để tạo ra sản phẩm cuối cùng. Robot sáu trục xuất sắc với sự kết hợp linh hoạt của sức mạnh và độ chính xác, làm cho chúng trở thành lựa chọn lý tưởng để tích hợp với các công cụ phù hợp để thực hiện nhiệm vụ. Các ví dụ điển hình bao gồm các công việc như khoan, mài hoặc cắt. Trong khi hầu hết các loại robot khác thường gặp khó khăn trong các ứng dụng như vậy vì nhiều lý do. Chúng có thể không đủ mạnh mẽ để đáp ứng các yêu cầu công việc nặng nề, và phạm vi tiếp cận có thể là một hạn chế đối với chúng. Trong khi đó, SCARA và delta, theo truyền thống, thường có kích thước nhỏ hơn so với robot sáu trục, điều này có thể tạo ra những hạn chế về khả năng tiếp cận trong các ứng dụng như vậy. Việc lựa chọn sai loại robot có thể có ảnh hưởng tiêu cực đến sự thành công của dự án tự động hóa của bạn.



Hình 2.8: 6DoF robot arm khoan trên bề mặt sản phẩm

## 2.4 Động học robot

Động học robot là một lĩnh vực trong robot học tập và điều khiển robot nghiên cứu về chuyển động và vị trí của robot. Nó tập trung vào việc nghiên cứu và mô hình hóa cách mà robot di chuyển và tương tác với môi trường xung quanh.

Các khái niệm cơ bản trong động học robot bao gồm:

1. **Hệ tọa độ robot:** Động học robot sử dụng hệ tọa độ để mô tả vị trí và hướng di chuyển của robot trong không gian. Hệ tọa độ thường được định nghĩa bằng các thông số như tọa độ XYZ và góc quay.
2. **Biểu diễn chuyển động:** Động học robot nghiên cứu cách biểu diễn chuyển động của robot. Điều này bao gồm mô hình hóa các khớp và cơ cấu của robot để hiểu cách chúng tương tác và di chuyển.
3. **Điều khiển chuyển động:** Động học robot cung cấp các phương pháp và thuật toán để điều khiển chuyển động của robot. Những phương pháp này bao gồm điều khiển vị trí, điều khiển tốc độ và điều khiển lực.
4. **Tính toán động học ngược:** Tính toán động học ngược là quá trình tính toán các thông số khớp cần thiết để đạt được một vị trí hay chuyển động mong muốn của robot. Nó đảo ngược quá trình tính toán động học thuận để tìm ra các thông số khớp.
5. **Lập kế hoạch chuyển động:** Động học robot cung cấp các phương pháp để lập kế hoạch các chuyển động của robot. Điều này bao gồm tìm đường đi tối ưu, tránh vật cản và lập lịch di chuyển.

Động học robot đóng vai trò quan trọng trong việc phát triển các hệ thống robot thông minh, từ robot công nghiệp đến robot dịch vụ và robot tự hành. Nó cung cấp cơ sở lý thuyết và công cụ để nghiên cứu và điều khiển chuyển động của robot một cách hiệu quả và chính xác.

Đây là một lĩnh vực rộng và tương đối phức tạp. Trong đồ án này, chúng tôi cố gắng áp dụng những kiến thức cơ bản nhất của động học robot cho cánh tay để có thể đáp ứng được yêu cầu tối thiểu nhất là di chuyển được cánh tay đến một tọa độ nhất định theo hệ trục tọa độ được quy định trước. Những yếu tố liên quan phức tạp hơn (độ chính xác, vận tốc, đường đi tối ưu,...) trong động học sẽ được nâng cấp dần và hoàn thiện trong giai đoạn Đồ án tốt nghiệp.

## 2.4.1 Quy tắc Denavit – Hartenberg (DH)

Quy tắc Denavit-Hartenberg (D-H) [12] là một phương pháp phổ biến trong động học robot để mô hình hóa và mô tả các khớp và liên kết của một robot. Quy tắc này được đặt tên theo hai nhà toán học người Mỹ, Jacques Denavit và Richard Hartenberg, người đã phát triển nó vào những năm 1950.

Quy tắc Denavit-Hartenberg sử dụng hệ tọa độ để mô hình hóa các khớp và liên kết trong robot. Các bước chính của quy tắc này bao gồm:

1. **Đánh số các khớp:** Mỗi khớp trong robot được đánh số thứ tự từ 1 đến n, với n là tổng số khớp trong robot.
2. **Xác định các trục chung:** Xác định các trục chung cho mỗi khớp. Trục chung là trục xoay hoặc trục di chuyển mà chúng ta quan tâm khi di chuyển từ khớp trước đến khớp sau.
3. **Xác định hệ tọa độ:** Xác định hệ tọa độ cục bộ cho mỗi khớp. Hệ tọa độ cục bộ được đặt tại trục chung của khớp đó và được xác định bằng cách sử dụng các thông số như góc quay và độ dài.
4. **Xác định các thông số DH:** Xác định các thông số Denavit-Hartenberg (DH) cho mỗi khớp. Các thông số này bao gồm  $\alpha$ ,  $a$ ,  $d$  và  $\theta$ , tương ứng với góc quay quanh trục cục bộ, độ dài theo trục chung trước, khoảng cách dọc theo trục chung và góc quay quanh trục chung.
5. **Xác định ma trận biến đổi:** Sử dụng các thông số DH, xây dựng ma trận biến đổi (hay còn gọi là ma trận DH) cho mỗi khớp. Ma trận này mô tả quan hệ tương đối giữa các hệ tọa độ cục bộ của các khớp liên kết với nhau.
6. **Tích chất ma trận:** Tính toán tích chất ma trận của các ma trận DH để xây dựng ma trận biến đổi từ hệ tọa độ cơ sở đến hệ tọa độ công việc của robot.

Quy tắc Denavit-Hartenberg giúp mô hình hóa và tính toán các chuyển động và vị trí của robot một cách dễ dàng và hợp lý. Nó được sử dụng rộng rãi trong việc phân tích và điều khiển chuyển động của các robot công nghiệp và robot manipulator.

### 2.4.1.1 Gán khung tọa độ DH cho cánh tay robot

Trong phần này, chúng tôi sẽ giới thiệu các nguyên tắc cơ bản về cách gán các khung tọa độ Denavit-Hartenberg (tức là các trục x, y và z) cho các loại cánh tay robot khác nhau [13].

Khung Denavit-Hartenberg (D-H) giúp chúng ta rút ra các phương trình cho phép chúng ta điều khiển một cánh tay robot.

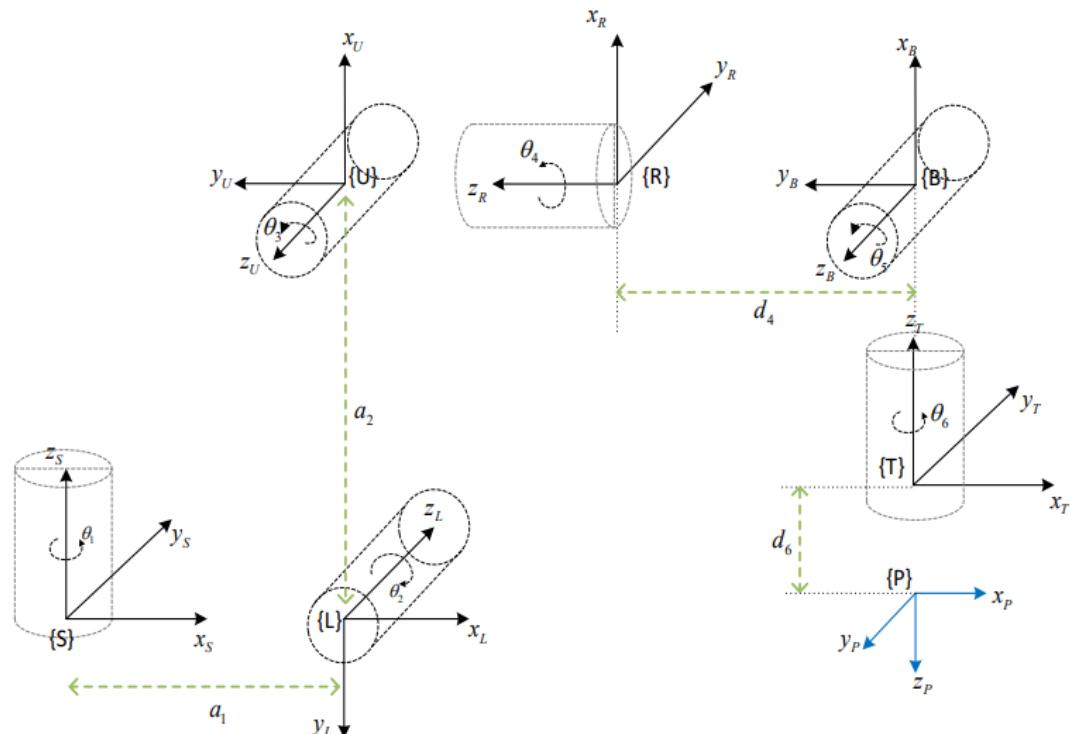
Các khung D-H của một cánh tay robot cụ thể có thể được phân loại như sau:

- **Khung tọa độ toàn cầu (global frame)**: Khung tọa độ này có thể có nhiều tên như world frame, base frame,...
- **Khung khớp**: Chúng ta cần một khung tọa độ cho mỗi khớp.
- **Khung tác động cuối**: Chúng ta cần một khung tọa độ cho bộ phận tác động cuối của robot (tức là bộ kẹp, bàn tay,...).

Dưới đây là bốn quy tắc hướng dẫn vẽ hệ tọa độ D-H:

1. Trục z là trục quay của khớp quay.
2. Trục x phải vuông góc với cả trục z hiện tại và trục z trước đó.
3. Trục y được xác định từ trục x và trục z bằng cách sử dụng quy tắc bàn tay phải.
4. Trục x phải cắt trục z trước đó (quy tắc không áp dụng cho khung 0).

Dựa vào quy tắc trên, ta có thể vẽ được hệ khung tọa độ cho cánh tay robot của chúng ta như sau:



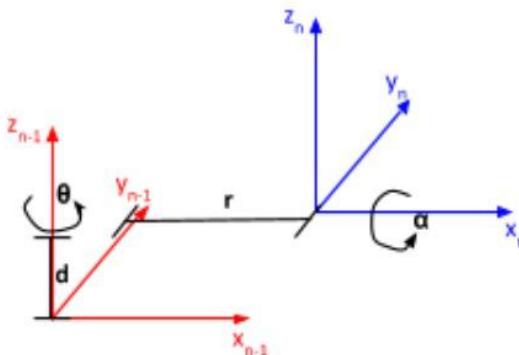
Hình 2.9: Cấu hình khung tọa độ DH cho robot JetArm

### 2.4.1.2 Bảng DH

Bảng Denavit-Hartenberg (DH) là một bảng dùng để mô tả các tham số của các khớp trong một cơ cấu robot dựa trên quy tắc Denavit-Hartenberg. Bảng DH thường được sử dụng để xác định các ma trận biến đổi D-H, từ đó mô hình hóa và tính toán vị trí và hướng của các khớp trong robot.

Bảng DH bao gồm các cột dữ liệu cho mỗi khớp trong robot. Thông thường, các cột này bao gồm các thông số sau:

- $a_i$ : Độ dài dọc theo trục  $x_i$  từ khớp i-1 đến khớp i.
- $\alpha_i$ : Góc quay xung quanh trục  $x_i$  từ khớp i-1 đến khớp i.
- $d_i$ : Độ dài dọc theo trục  $z_i$  từ khớp i-1 đến khớp i.
- $\theta_i$ : Góc quay xung quanh trục  $z_i$  từ khớp i-1 đến khớp i.



Hình 2.10: Minh họa cách tìm các thông số cho bảng DH [14]

Dựa vào cấu hình của 6DoF robot arm, ta có bảng DH như sau:

Khớp	$a_i$ (mm)	$\alpha_i$ (rad)	$d_i$ (mm)	$\theta_i$ (rad)
1	47	$-\pi/2$	133	0
2	110	$-\pi$	0	$-\pi/2$
3	26	$-\pi/2$	0	0
4	0	$\pi/2$	117.5	0
5	0	$-\pi/2$	0	$-\pi/2$
6	0	$\pi$	28	0

Bảng 2.1: Bảng DH của robot JetArm

## 2.4.2 Động học thuận

Động học thuận (forward kinematics) [15] là hệ phương trình mô tả vị trí và định hướng của đầu công tác robot tương ứng với từng giá trị các khớp. Động học thuận là câu trả lời cho vấn đề: Bộ phận tác động cuối cùng của robot (ví dụ: tay kẹp, tay, cốc hút chân không, v.v.) nằm ở đâu trong không gian khi chúng ta biết các góc xoay của động cơ?

Động học thuận thường được thực hiện bằng cách áp dụng các ma trận biến đổi D-H (Denavit-Hartenberg) cho các khớp trong robot và kết hợp chúng để tính toán ma trận biến đổi cuối cùng từ khớp gốc đến công cụ cuối cùng. Quá trình này thường bao gồm các bước sau:

1. Xây dựng bảng DH
2. Tính toán ma trận biến đổi: Sử dụng bảng DH, tính toán ma trận biến đổi D-H cho mỗi khớp trong robot.
3. Kết hợp ma trận biến đổi: Kết hợp các ma trận biến đổi D-H để tính toán ma trận biến đổi từ khớp gốc đến công cụ cuối cùng. Thông thường, quá trình này được thực hiện bằng cách nhân các ma trận biến đổi với nhau theo thứ tự các khớp.
4. Tính toán vị trí và hướng: Từ ma trận biến đổi cuối cùng, tính toán vị trí và hướng của công cụ cuối cùng hoặc các khớp khác trong robot.

Quá trình động học thuận cung cấp thông tin về vị trí và hướng của công cụ cuối cùng dựa trên các thông số và góc quay của các khớp trong robot. Nó là một bước quan trọng trong việc điều khiển và lập trình các cơ cấu robot để thực hiện nhiệm vụ cụ thể.

Cụ thể với cánh tay 6DoF của chúng ta, qua mỗi khớp, ta sẽ có mối liên hệ giữa hệ toạ độ (vị trí và định hướng) của khớp  $i$  và khớp  $i - 1$  như sau:

$$A_i^{i-1}(q_i) = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & -a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

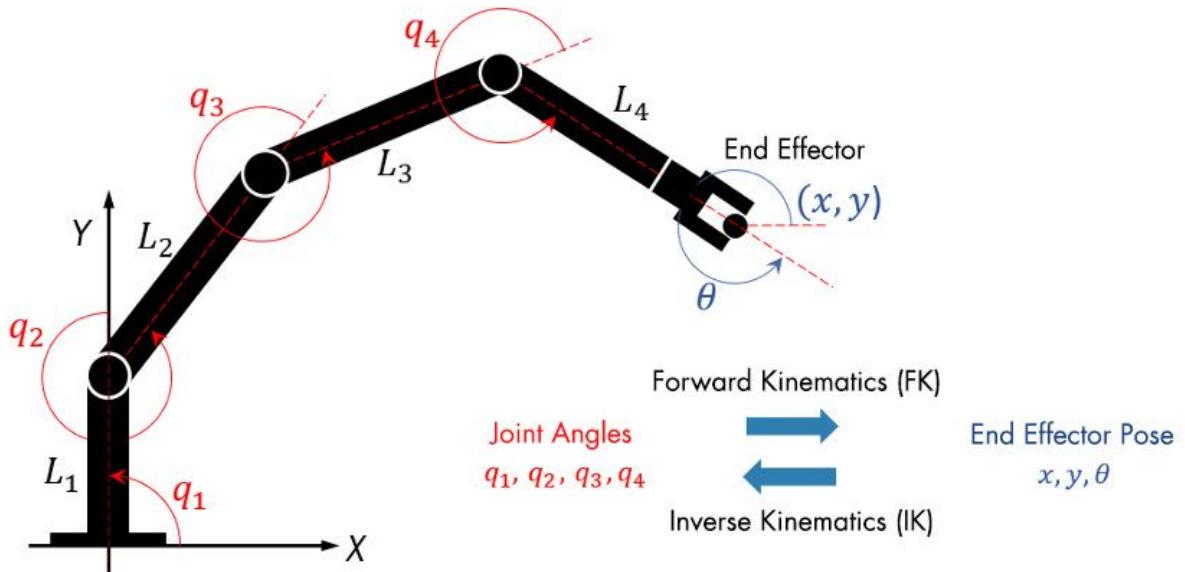
Trong đó,  $c\theta$  đại diện cho hàm cosin của góc  $\theta$ ,  $s\theta$  đại diện cho hàm sine của góc  $\theta$ . Các biến  $\theta, \alpha, d, a$  được lấy từ bảng DH đã lập trước đó.

Xem xét một cánh tay robot n bậc tự do tổng quát, Phương trình động học thuận có thể viết gọn lại bằng ma trận biến đổi sau đây:

$$T_n^0(q) = \begin{bmatrix} R(q) & p(q) \\ 0^T & 1 \end{bmatrix} = A_1^0(q_1)A_2^1(q_2)\dots A_n^{n-1}(q_n)$$

Trong phương trình trên, ma trận  $R$  thể hiện các góc xoay của đầu công tác, ma trận  $p$  thể hiện tọa độ  $x, y, z$  của đầu công tác.

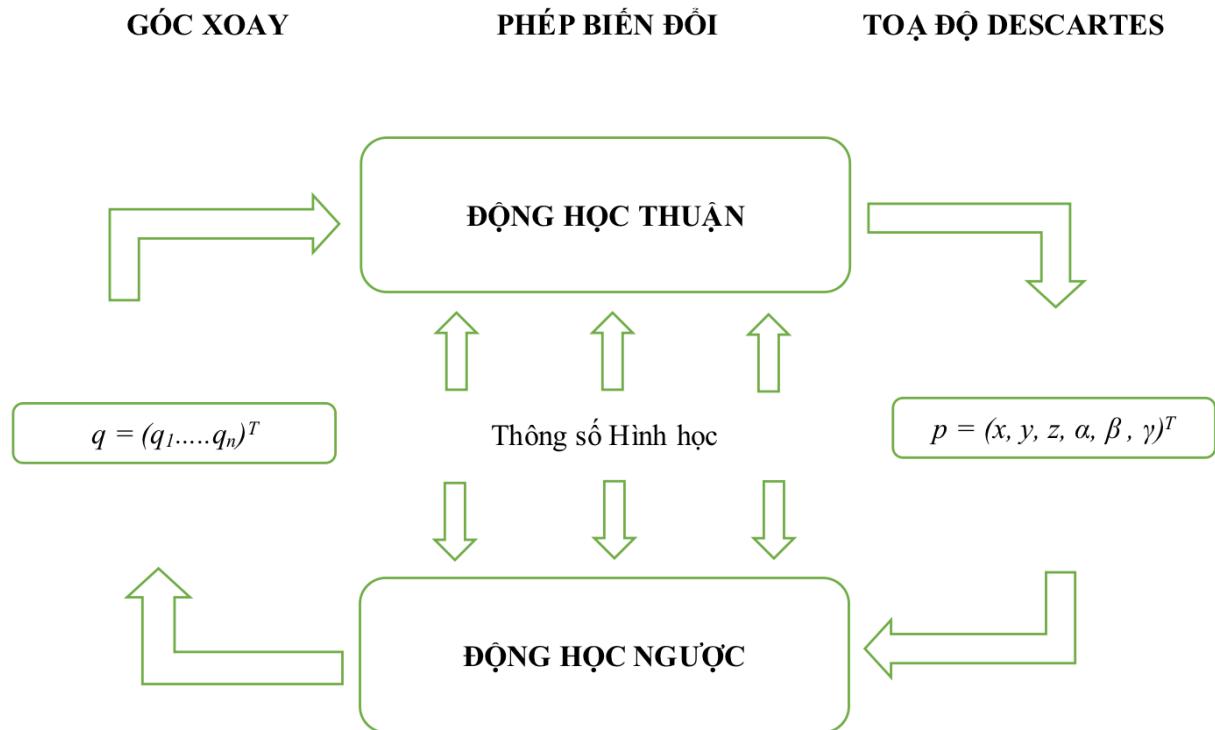
Vector  $q = [q_1 \dots q_n]^T$  là vector các giá trị biến khớp. Vị trí và góc xoay của đầu công tác phụ thuộc vào vector  $q$  này.



Hình 2.11: Xác định cấu hình vị trí khớp của robot sử dụng động học robot trong không gian 2D [16]. Với vector các giá trị biến khớp và sử dụng động học thuận, ta có thể xác định được tọa độ và góc xoay hiện tại của đầu công tác.

### 2.4.3 Động học nghịch

Động học nghịch (inverse kinematics) [17] là quá trình tính toán các giá trị góc quay của các khớp trong một cơ cấu robot dựa trên vị trí và hướng mong muốn của công cụ cuối cùng (end effector) hoặc các điểm trong không gian làm việc. Nói cách khác, động học nghịch chính là bài toán nghịch đảo của bài toán động học thuận.



Hình 2.12: Mối tương quan giữa động học thuận và động học nghịch

Động học nghịch là một phần quan trọng trong việc điều khiển và lập trình các robot để thực hiện các nhiệm vụ cụ thể như điều khiển đồng bộ, định vị, hoặc tránh vật cản. Nó cho phép robot tự động tính toán các giá trị góc quay để đáp ứng các yêu cầu vị trí và hướng tương ứng.

Quá trình động học nghịch thường bao gồm các bước sau:

1. Xác định vị trí và hướng mong muốn: Xác định vị trí và hướng mong muốn của công cụ cuối cùng hoặc các điểm trong không gian làm việc.
2. Xây dựng mô hình động học nghịch: Dựa trên cấu trúc và thông số của robot, xây dựng mô hình động học ngược để tính toán các giá trị góc quay của các khớp.

3. Giải quyết các phương trình nghịch: Sử dụng mô hình động học nghịch, giải quyết các phương trình phi tuyến để tìm các giá trị góc quay của các khớp.
4. Kiểm tra và điều chỉnh: Kiểm tra các giá trị góc quay tìm được và điều chỉnh nếu cần thiết để đảm bảo rằng vị trí và hướng của công cụ cuối cùng đạt được mong muốn.

Giải quyết các phương trình phi tuyến là một thách thức quan trọng trong động học nghịch. Trong nhiều trường hợp, các phương trình phi tuyến phải được giải bằng các phương pháp số hoặc các thuật toán tìm kiếm nghiệm gần đúng. Có nhiều phương pháp để giải quyết các phương trình phi tuyến trong động học ngược, phương pháp cụ thể phụ thuộc vào cấu trúc của robot và mục tiêu xác định. Dưới đây là một số phương pháp thường được sử dụng:

1. **Phương pháp Newton-Raphson:** Đây là một phương pháp lặp để tìm nghiệm gần đúng của các phương trình phi tuyến. Phương pháp này yêu cầu tính toán đạo hàm riêng và ma trận Jacobi của hệ phương trình, và sau đó sử dụng các bước lặp để tiến tới nghiệm gần đúng.
2. **Phương pháp tìm kiếm theo dòng (Gradient-based methods):** Các phương pháp tìm kiếm theo dòng như phương pháp gradient và phương pháp Quasi-Newton có gắng tìm kiếm nghiệm bằng cách di chuyển theo hướng giảm dần của hàm mục tiêu. Các phương pháp này phụ thuộc vào tính toán đạo hàm của hàm mục tiêu.
3. **Phương pháp tìm kiếm không gian trạng thái (State-space search methods):** Các phương pháp này xem xét không gian trạng thái của robot và tìm kiếm trong không gian này để tìm ra nghiệm. Các phương pháp như thuật toán Dijkstra, A\*, hoặc thuật toán tìm kiếm thông tin có thể được áp dụng trong trường hợp này.
4. **Phương pháp tối ưu hóa:** Một số bài toán động học ngược có thể được chuyển thành bài toán tối ưu hóa, trong đó ta tìm kiếm các giá trị góc quay tối ưu hóa hàm mục tiêu. Các phương pháp tối ưu hóa như Gradient Descent, Quasi-Newton, hoặc thuật toán di truyền có thể được áp dụng để tìm nghiệm gần đúng.

Cần lưu ý rằng giải quyết các phương trình phi tuyến trong động học nghịch có thể yêu cầu sự kết hợp của nhiều phương pháp và có thể đòi hỏi tính toán số phức tạp. Các thuật toán và phương pháp cụ thể phụ thuộc vào bài toán cụ thể và yêu cầu của hệ thống robot.

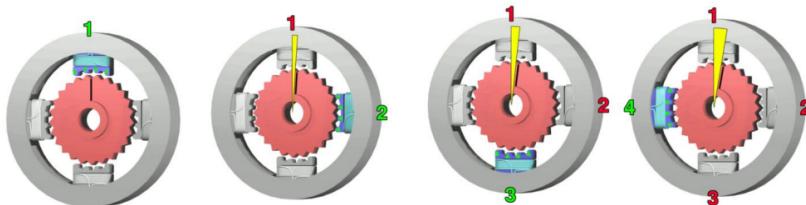
## 2.5 Tổng quan về Stepper motor

### 2.5.1 Cấu tạo và nguyên lý hoạt động

Stepper motor là một loại động cơ điện được sử dụng rộng rãi trong các ứng dụng cần kiểm soát chính xác vị trí và chuyển động. Nó được gọi là "stepper motor" (động cơ bước) vì nó hoạt động bằng cách di chuyển từng bước rời rạc, thay vì quay liên tục như các loại động cơ khác.

Stepper motor có cấu trúc đơn giản, gồm một rotor (rotating part) và một stator (stationary part). Rotor bao gồm một số cực nam châm và cực nam châm, còn stator bao gồm các cuộn dây được xếp thành các phần tử điện từ. Khi dòng điện được điều khiển thông qua các cuộn dây, sức hút từ các phần tử điện từ trong stator tác động lên rotor, tạo ra một lực làm quay rotor một góc nhất định.

Điều làm cho động cơ bước trở nên độc đáo là số lượng cực lớn mà nó có và cách chúng hoạt động. Không giống như động cơ DC thông thường, có thể tìm thấy trong quạt, đồ chơi, máy trộn và nhiều thiết bị điện tử tiêu dùng khác thường có 4-12 cực, động cơ bước thông thường có 200 cực mà động cơ có thể có.



Hình 2.13: Sơ đồ nguyên lý hoạt động của động cơ bước

Sơ đồ trên thể hiện những điều cơ bản về cách hoạt động của động cơ bước. Rôto, màu đỏ, có một bộ răng lớn và là bộ phận quay. Có bốn nam châm điện, trong đó nam châm có điện được đánh dấu màu xanh lam.

Bước đầu tiên, nam châm điện phía trên được bật lên, các răng của rôto thẳng hàng với các răng của nam châm. Để quay động cơ, nam châm điện thứ hai (phải) được cấp điện khi nam châm điện thứ nhất tắt. Rôto quay theo chiều kim đồng hồ cho đến khi các răng khớp lại, đây là một bước quay.

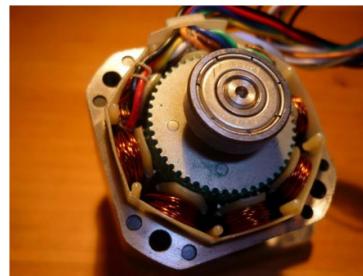
Để quay xa hơn, quá trình này được lặp lại đối với nam châm điện thứ ba và thứ tư, rôto tiếp tục quay theo chiều kim đồng hồ khi các răng tiếp tục khớp với nam

châm điện đang hoạt động. Khi nam châm điện phía trên được cấp điện trở lại, rôto sẽ quay một vị trí răng và thực hiện bốn bước để làm như vậy. Rôto trong ví dụ này có 25 răng và với 4 bước trên mỗi răng, rôto này sẽ cần thực hiện 100 bước để rôto quay hết một vòng. Như vậy, mỗi bước quay của motor tương ứng với một góc quay  $3.6^\circ$ .

Khi loại động cơ này được giữ trong phạm vi hoạt động của nó, nó không cần vòng phản hồi. Mỗi bước mà động cơ này thực hiện đều có một số vòng quay nhất định, khi có được độ quay trực mong muốn, nó chỉ cần thực hiện đúng số bước để đạt được mức này. Tuy nhiên, khi một bước bị bỏ lỡ, chẳng hạn do tải quá nặng, hệ thống này không có cách nào để nhận biết và không có cách nào để tự sửa chữa. Khi được giữ trong phạm vi hoạt động của nó, những động cơ này rất lý tưởng để điều khiển định vị chính xác. Đây là劣势 điểm lớn nhất của động cơ bước. Để khắc phục劣势 này, chúng ta có thể thêm vào các bộ encoder để thực hiện cơ chế phản hồi (feedback).

Bên cạnh đó, động cơ bước có những ưu điểm đáng chú ý phù hợp với cánh tay robot vốn dùng để nghiên cứu và học tập của chúng ta:

- 1. Kiểm soát vị trí chính xác:** Động cơ bước cho phép kiểm soát vị trí chính xác bằng cách di chuyển từng bước rời rạc. Mỗi bước di chuyển tương ứng với một xung điện số, giúp đạt được độ chính xác cao trong việc định vị và điều khiển vị trí.
- 2. Tính ổn định:** Động cơ bước có tính ổn định cao trong việc duy trì vị trí sau khi di chuyển. Do các bước di chuyển là rời rạc và không dễ bị ảnh hưởng bởi tải ngoại lực, động cơ bước thường không trượt hoặc bị mất bước trong quá trình vận hành.
- 3. Khả năng giữ vị trí khi không có nguồn điện:** Động cơ bước có khả năng giữ vị trí ngay cả khi không có nguồn điện. Khi đóng cửa nguồn điện, động cơ bước vẫn giữ vị trí hiện tại, điều này có thể rất hữu ích trong các ứng dụng an toàn hoặc khi cần tránh mất dữ liệu.
- 4. Độ tin cậy cao:** Vì động cơ bước có cấu trúc đơn giản và không có bộ phận tiếp xúc trực tiếp, nên nó thường có tuổi thọ cao và độ bền lâu dài. Điều này giúp giảm chi phí bảo trì và thay thế trong quá trình sử dụng.
- 5. Dễ dàng điều khiển:** Động cơ bước có thể được điều khiển dễ dàng bằng các tín hiệu xung điện số. Việc điều chỉnh tốc độ, hướng di chuyển và vị trí của động cơ bước có thể được thực hiện một cách linh hoạt và chính xác thông qua việc kiểm soát số lượng và tần số của các xung điện.

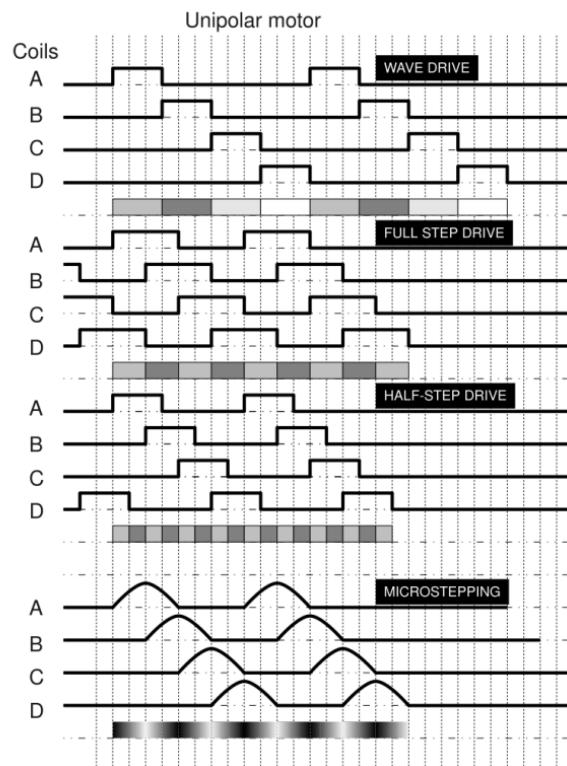


Hình 2.14: Hình ảnh thực tế bên trong động cơ bước

## 2.5.2 Nguyên lý điều khiển

Những động cơ bước này hoạt động dựa trên cơ chế điều khiển tín hiệu từng bước để cung cấp năng lượng cho các nam châm điện xung quanh rôto. Như đã nêu ở trên, rôto của động cơ bước có các răng bị hút bởi nam châm điện khi được cấp điện.

Những giải thích được đưa ra ở trên về cách hoạt động của động cơ bước giúp chúng ta hiểu được những điều cơ bản nhưng thường không được sử dụng trong thực tế bởi khi không có sự chồng chéo giữa các bước và động cơ phải di chuyển một tải, có thể rôto sẽ bị trượt tại thời điểm chuyển tiếp giữa hai nam châm điện. Sự so sánh giữa các loại tín hiệu bước khác nhau có thể được xem ở hình dưới.



Hình 2.15: Tín hiệu điều khiển động cơ bước khác nhau

Bước toàn pha (full stepping) có cùng số bước như chế độ điều khiển sóng (wave drive) nhưng có lợi thế luôn bật hai nam châm. Điều này đảm bảo không có khoảng thời gian không có công suất ở mỗi bước. Momen xoắn cực đại được đạt được ở kỹ thuật điều khiển này.

Bước nửa pha (half stepping) luân phiên giữa việc bật hai nam châm và chỉ bật một nam châm. Điều này làm tăng gấp đôi độ phân giải góc nhưng đi kèm với mất mô-men xoắn. Ở vị trí bước đầy đủ, khi chỉ có một điện từ bật, nó có khoảng 70% mô-men xoắn tối đa của nó.

Bước vi mô (Micro stepping) tăng độ phân giải góc nhiều nhất. Thường được gọi là 'sine cosin microstepping'. Ở đây, các bước tăng dần được giảm xuống thành phô quay trực tiếp liên tục về mặt lý thuyết. Bằng cách tăng số lượng vi bước sẽ tạo ra nhiều vị trí hơn giữa các bước hoàn chỉnh. Tuy nhiên, điều này không tạo ra sự tăng độ chính xác. Lượng mô-men xoắn được cung cấp bởi mỗi bước nhỏ sẽ giảm bằng cách tăng số lượng bước nhỏ trên mỗi bước đầy đủ, như có thể thấy trong hình bên dưới. Do sự giảm mô-men xoắn này nên không phải mọi bước vi mô đều dẫn đến chuyển động của trực. Nó có thể kết hợp mô-men xoắn của một số vi bước để khắc phục ma sát trong động cơ.

Giảm tiếng ồn cơ học và điện từ là lợi ích thực sự của vi bước. Việc truyền mô-men xoắn sẽ tổng quát hơn dẫn đến ít vần đè cộng hưởng hơn, tăng độ tin cậy của hệ thống vòng hở đồng bộ và ít mài mòn động cơ hơn.

Microsteps/Full step	% Holding Torgue/Microstep
1	100.00%
2	70.71%
4	38.27%
8	19.51%
16	9.80%
32	4.91%
64	2.45%
128	1.23%
256	0.61%

Bảng 2.2: Liên hệ giữa vi bước và mô men

## 2.6 Tổng quan về vi điều khiển

### 2.6.1 Jetson Nano

Jetson Nano là một bo mạch tích hợp (SBC) mạnh mẽ do NVIDIA phát triển, được thiết kế đặc biệt cho các ứng dụng trí tuệ nhân tạo (AI) và học máy. Với kích thước nhỏ gọn và hiệu suất ấn tượng, Jetson Nano trở thành một lựa chọn lý tưởng cho những người phát triển, sinh viên và đối tác trong lĩnh vực AI.

Được trang bị bộ xử lý GPU NVIDIA Maxwell với 128 lõi CUDA, Jetson Nano cung cấp khả năng xử lý đồng thời và hiệu suất tính toán đáng kể, làm cho nó trở thành một nền tảng mạnh mẽ để triển khai và thử nghiệm mô hình học máy và ứng dụng AI. Với khả năng kết nối với nhiều thiết bị ngoại vi, Jetson Nano cung cấp môi trường phát triển linh hoạt và đa dạng.

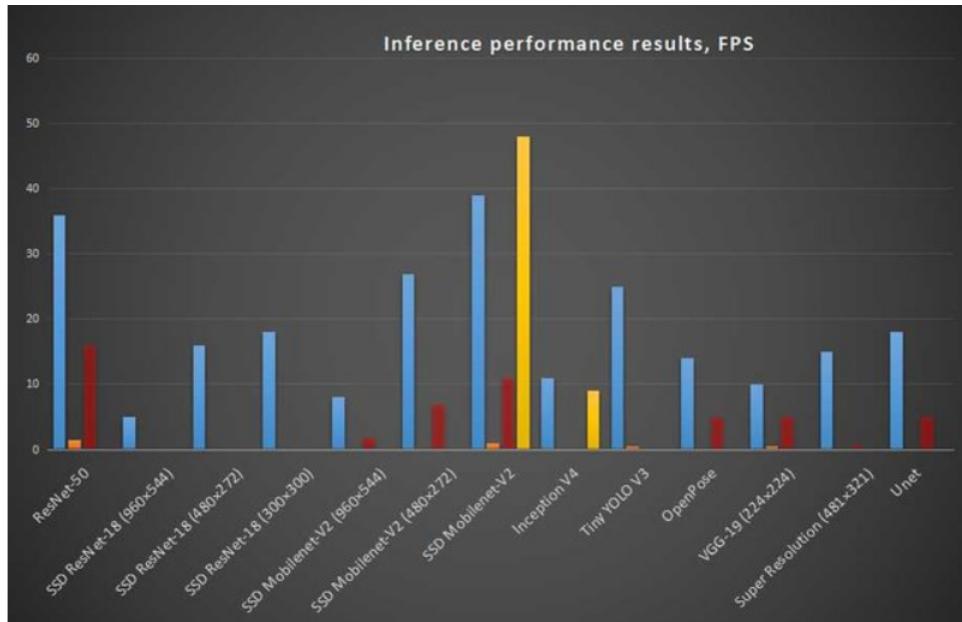
Jetson Nano không chỉ giúp đơn giản hóa quá trình phát triển và triển khai mô hình AI mà còn là cầu nối cho sự tích hợp linh hoạt vào các dự án IoT và nhúng. Bằng cách cung cấp một môi trường mạnh mẽ và thân thiện với người dùng, Jetson Nano đánh dấu một bước quan trọng trong việc đưa công nghệ AI đến gần hơn với cộng đồng phát triển và người sử dụng cuối cùng.

#### 2.6.1.1 Đặc điểm nổi bật của Jetson Nano

Nvidia Jetson Nano là một bộ phát triển gồm một Module hệ thống (SoM) và một bo mạch thử nghiệm tham chiếu. Nó chủ yếu được hướng đến việc tạo ra các hệ thống nhúng cần sức mạnh xử lý cao cho ứng dụng máy học, thị giác máy và xử lý video. Nvidia đang nỗ lực để đưa trí tuệ nhân tạo vào đám đông thông qua bo mạch này, tương tự như Raspberry Pi đã làm với máy tính nhúng[18].

Hệ thống được xây dựng xung quanh một GPU Maxwell với 128 lõi CUDA và CPU ARM A57 bốn lõi chạy ở tốc độ 1,43 GHz, kết hợp với 4GB bộ nhớ LPDDR4. Mặc dù CPU ARM A57 khá nhanh và 4GB bộ nhớ, nhưng điểm mấu chốt của khả năng tăng tốc máy học của bo mạch nằm trong GPU của nó. Với 128 lõi CUDA, GPU của Jetson Nano được Nvidia tuyên bố có công suất xử lý là 472 GFLOPS (Giga Floating Point Operations per Second). Con số này thường được sử dụng khi so sánh với các sản phẩm khác của Nvidia khi xác định tốc độ suy luận thực tế. Để đánh giá tốt hơn, việc so sánh trực tiếp chúng chạy các mô hình máy học phổ biến là một kiểm thử tốt [18].

Dưới đây là hình ảnh về kết quả so sánh về hiệu suất của jetson nano (màu xanh dương) khi so sánh với các board mạch khác là Raspberry Pi Model 3 (màu cam), Raspberry Pi 3 + Intel Neural Compute Stick 2 (màu đỏ) và bo mạch Google Edge TPU Coral (màu vàng) trong các phần mềm về học máy và thị giác máy tính[18]:



Hình 2.16: Hiệu suất của Jetson Nano

Như chúng ta có thể thấy, Jetson Nano rõ ràng vượt trội trước Raspberry Pi 3 + Intel Neural Compute Stick 2 trên tất cả các phần mềm được so sánh. Còn đối với Google Edge TPU Coral, hiện tại vẫn chưa có đầy đủ các so sánh như ảnh trên nên chưa thể đưa ra được kết luận cuối cùng.

Ngoài ra, Jetson Nano còn có một hệ thống phần cứng hỗ trợ khá đầy đủ bao gồm: 1 Micro SD Card Slot, 1 Expansion Header Pinout (J41), 1 Micro USB Card Slot, 1 Gigabit Ethernet, USB3.0 x 4, 1 HDMI Type A Slot, 1 Display Port và 1 5VDC Power Port.

Bên cạnh đó, Jetson Nano cũng hỗ trợ kết nối không dây thông qua WiFi và Bluetooth, tăng khả năng linh hoạt và di động của thiết bị. Những kết nối đa dạng này cung cấp sự linh hoạt cho Jetson Nano trong việc tích hợp và mở rộng chức năng, làm cho nó trở thành một lựa chọn mạnh mẽ cho nhiều ứng dụng trong lĩnh vực trí tuệ nhân tạo và học máy cũng như lập trình nhúng và Robot.

### 2.6.1.2 Môi trường cung cấp

Jetson Nano chạy trên hệ điều hành Ubuntu Linux (18.04), một hệ điều hành mã nguồn mở phổ biến và mạnh mẽ. Điều này mang lại cho người phát triển sự linh hoạt trong việc tận dụng các công cụ và thư viện mã nguồn mở để phát triển ứng dụng AI và ML. Hệ điều hành Linux cung cấp một môi trường làm việc đồng nhất và quen thuộc, giúp giảm bớt sự phức tạp trong quá trình phát triển và triển khai ứng dụng trí tuệ nhân tạo.

Ngôn ngữ lập trình chủ yếu là Python, tuy nhiên, cũng có sự hỗ trợ cho C/C++ và nhiều ngôn ngữ khác. Và do chạy trên nền tảng linux nên Jetson Nano luôn có sẵn môi trường để chúng ta làm việc hiệu quả đặc biệt với hai ngôn ngữ lập trình chính là Python và C/C++. Các IDE như Visual Studio Code thường được ưa chuộng để phát triển và debug ứng dụng. Chúng có thể được cài đặt trực tiếp lên Jetson Nano. Jetson Nano hỗ trợ nhiều thư viện và framework quan trọng như TensorFlow, PyTorch, OpenCV, CUDA và cuDNN, giúp đơn giản hóa quá trình phát triển mô hình học máy.

Kết nối với máy tính host được thực hiện thông qua cổng USB hoặc mạng Ethernet, cho phép truyền dữ liệu, triển khai ứng dụng và debug từ xa. Các cổng kết nối như HDMI, GPIO, cổng camera CSI, cổng USB mang lại tính linh hoạt trong việc kết nối với màn hình, camera, cảm biến và nhiều thiết bị ngoại vi khác. Ngoài ra chúng ta có thể sử dụng giao thức ssh để truy cập trực tiếp vào Jetson Nano thông qua địa chỉ IP của nó miễn là máy tính của chúng ta và Jetson Nano chung một DNS.

Công cụ phân tích hiệu năng như NVIDIA Nsight Systems và NVIDIA Nsight Compute hỗ trợ nhà phát triển trong quá trình phân tích hiệu năng và debug các ứng dụng GPU trên Jetson Nano.

### 2.6.1.3 Ứng dụng

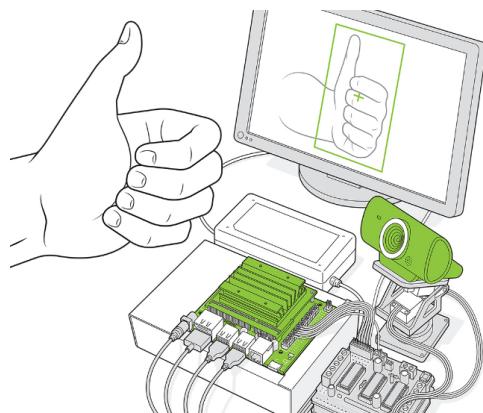
Jetson Nano, với sức mạnh tính toán và khả năng xử lý AI, có nhiều ứng dụng rộng rãi trong lĩnh vực trí tuệ nhân tạo và máy học. Một số ứng dụng tiêu biểu bao gồm:

- Nhận diện Đối tượng và Phân loại:** Jetson Nano có khả năng xử lý các mô hình học máy phức tạp, giúp ứng dụng nhận diện và phân loại đối tượng trong thời gian thực. Điều này có thể được áp dụng trong các lĩnh vực như giám sát an ninh, theo dõi đối tượng và ô tô tự lái.



Hình 2.17: Ứng dụng của Jetson Nano trong xe tự hành

- **Xử lý Ảnh và Video:** Jetson Nano là một công cụ mạnh mẽ cho xử lý ảnh và video, từ việc tăng cường hình ảnh đến theo dõi chuyển động và phân tích hành vi. Ứng dụng trong y tế, giáo dục và giải trí là những ví dụ điển hình.



Hình 2.18: Ứng dụng của Jetson Nano trong việc nhận diện hành vi con người thông qua camera

- **Robotics và Điều khiển Tự động:** Với khả năng tích hợp các cảm biến và điều khiển thiết bị ngoại vi, Jetson Nano là sự lựa chọn lý tưởng cho phát triển robot và hệ thống điều khiển tự động. Nó có thể thực hiện các nhiệm vụ như điều khiển động cơ, xử lý dữ liệu cảm biến và đưa ra quyết định trong thời gian thực.



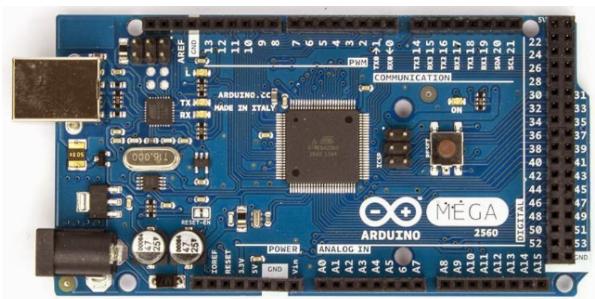
Hình 2.19: Ứng dụng của Jetson Nano trong Robotics

- Học Máy và Phát triển Model AI:** Jetson Nano cung cấp môi trường phát triển cho việc xây dựng và huấn luyện mô hình máy học. Người dùng có thể tận dụng GPU tích hợp để giảm thời gian huấn luyện và tối ưu hóa hiệu suất của mô hình AI.
- Giáo dục và Học tập:** Jetson Nano là một công cụ học tập lý tưởng để giới thiệu về trí tuệ nhân tạo và lập trình đối với sinh viên, học sinh và những người mới bắt đầu trong lĩnh vực công nghệ.

Với sự linh hoạt và khả năng tùy chỉnh cao, Jetson Nano đóng vai trò quan trọng trong việc đưa công nghệ trí tuệ nhân tạo đến cộng đồng phát triển và ứng dụng đa dạng.

## 2.6.2 Arduino Mega

### 2.6.2.1 Cấu tạo



Hình 2.20: Cấu tạo của Arduino Mega

Arduino Mega là một trong những board phổ biến trong hệ thống Arduino, được thiết kế để cung cấp nhiều chân I/O và tài nguyên tính toán hơn so với các phiên bản

Arduino khác. Đây là một board mạch mở và linh hoạt, thường được sử dụng trong các dự án đòi hỏi nhiều chân kết nối và xử lý dữ liệu. Dưới đây là một giới thiệu tổng quan về Arduino Mega:

- **Số chân IO:**

- Arduino Mega cung cấp 54 chân số (Digital I/O), trong đó có 15 chân có thể sử dụng để tạo PWM.
- 16 chân vào analog (ADC), giúp đọc giá trị từ cảm biến và thiết bị đo lường.
- Có 6 chân hỗ trợ interrupt

- **Vi xử lý và Bộ Nhớ:**

- Sử dụng vi điều khiển ATmega2560 với tần số hoạt động 16 MHz.
- Bộ nhớ Flash 256 KB cho chương trình (Code).
- Bộ nhớ SRAM 8 KB cho dữ liệu tạm thời.
- Bộ nhớ EEPROM 4 KB cho lưu trữ dữ liệu không thay đổi.

- **Các chuẩn giao tiếp:**

- Cổng USB để kết nối với máy tính và tải chương trình.
- UART (Serial) có 4 bộ UART, SPI có 1 bộ (chân 50 -> 53) dùng với thư viện SPI của Arduino, I2C có 1 bộ cho giao tiếp với các thiết bị khác.
- Cổng Ethernet và khe cắm thẻ nhớ SD cho mở rộng khả năng kết nối và lưu trữ.

- **Các tính Năng:**

- Hỗ trợ nhiều nguồn nguồn cấp: 5V trực tiếp, hoặc qua cổng USB, hoặc nguồn ngoại vi có thể từ 7V đến 12V.
- Hỗ trợ tính năng tự điều chỉnh nguồn (Auto Power Select).
- Nhiều LED chỉ trạng thái và hoạt động.

- **Phần mềm hỗ trợ và ngôn ngữ lập trình:**

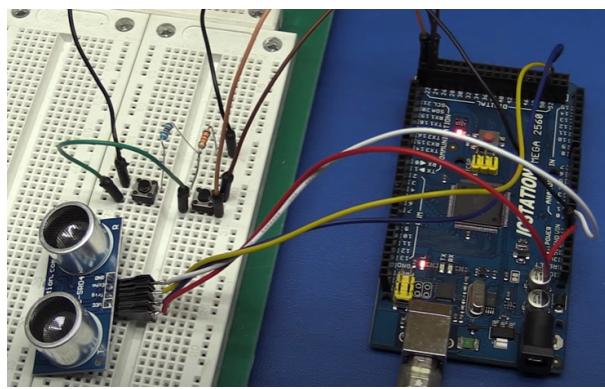
- Hỗ trợ Arduino IDE, là môi trường lập trình dễ sử dụng cho người mới bắt đầu. Ngoài ra có thể sử dụng IO PlatForm trong VScode để làm việc với Arduino Mega.
- Có thể sử dụng ngôn ngữ lập trình C/C++ thông thường.

Ngoài ra tất cả các Shield của Arduino Uno đều chạy được với Arduino Mega. Tuy nhiên, Arduino Mega không dùng được thư viện SoftwareSerial vì Mega đã có 4 bộ UART và không hỗ trợ thêm về SoftwareSerial.

### 2.6.2.2 Ứng dụng

Arduino Mega thường được sử dụng trong các dự án lớn hơn và phức tạp hơn, bao gồm:

- **Robotics:** Điều khiển nhiều động cơ và cảm biến.
- **IoT (Internet of Things):** Kết nối với Internet thông qua giao tiếp Ethernet hoặc thẻ nhớ SD.
- **Điều khiển thiết bị:** điều khiển các khối ngoại vi hoặc nhận dữ liệu từ các cảm biến và gửi về gateway thông qua các giao thức đơn giản.



Hình 2.21: Arduino Mega và Ultrasonic



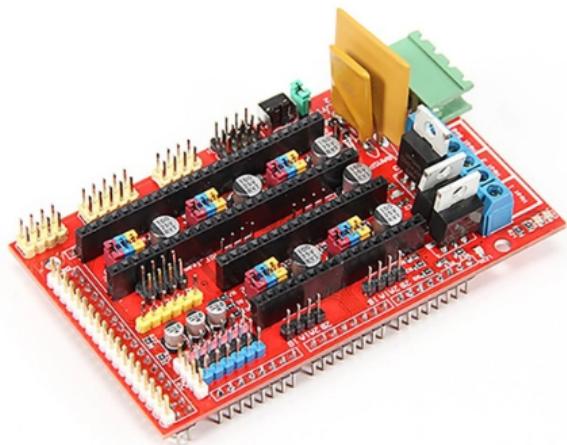
Hình 2.22: Arduino Mega và LCD

### 2.6.2.3 Ramp Shield

RAMPS 1.4 (Reprap Arduino Mega Pololu Shield) là board mở rộng cắm trên Arduino Mega 2560 và dùng để điều khiển các máy in 3D Reprap cũng như các ứng dụng khác.

RAMPS chủ yếu điều khiển các driver động cơ bước (stepper motor drivers) để di chuyển các trục của máy in 3D, và nó cũng bao gồm các chân kết nối cho cảm biến nhiệt độ, cảm biến đo mức, và các linh kiện khác.

RAMPS bao gồm nhiều chân kết nối cho các driver động cơ A4988 hoặc DRV8825, cổng USB để kết nối với máy tính để chạy mã điều khiển, cổng điều khiển LCD, cổng kết nối cảm biến nhiệt độ, và các cổng kết nối khác.



Hình 2.23: Ramp Shield

Các điểm đặc biệt của Ramp Shield:

- Dùng để điều khiển máy in 3D và các dạng robot 3 trục tịnh tiến
- Có thẻ mở rộng cho các phụ kiện điện tử khác
- 3 mạch công suất cho các đầu nung và quạt, các mạch xử lý tín hiệu nhiệt điện trở
- Điều khiển bàn nhiệt (có bảo vệ bằng cầu chì tự phục hồi 11A)
- Có 5 khay cắm mô đun điều khiển động cơ bước A4988
- Có thẻ tích hợp thẻ nhớ
- Hiển thị trạng thái hoạt động bằng Led
- Hỗ trợ tối 2 động cơ trục Z trên các máy in 3D Prusa Mendel

## 2.7 Giao tiếp giữa các thiết bị

### 2.7.1 GPIO

- **Định nghĩa**

GPIO (General-Purpose Input/Output) là một tính năng cơ bản trong các vi điều khiển, đóng vai trò quan trọng trong chức năng tổng thể của chúng. Nó cho phép vi điều khiển tương tác với các thiết bị ngoại vi và thực hiện các hoạt động như đọc tín hiệu (đầu vào) hoặc xuất tín hiệu điều khiển (đầu ra).

- **Nguyên lý hoạt động**

Các chân GPIO trên vi điều khiển có khả năng được thiết lập để thực hiện vai trò đầu vào hoặc đầu ra. Khi được cấu hình là đầu vào, những chân này có thể đọc giá trị từ các tín hiệu ngoại vi như cảm biến, công tắc hoặc nút nhấn, giúp vi điều khiển thu thập thông tin từ môi trường xung quanh. Ngược lại, khi được cấu hình là đầu ra, các chân GPIO có thể gửi tín hiệu điều khiển đến các thiết bị ngoại vi như đèn LED, động cơ hoặc relay. Điều này cho phép vi điều khiển tương tác và kiểm soát các thiết bị ngoại vi theo ý muốn.

- **Đặc tính nổi bật**

- Tính linh hoạt là một trong những ưu điểm quan trọng của GPIO, cho phép vi điều khiển thích ứng và tương tác với nhiều giao thức khác nhau như UART, I2C và SPI. Khả năng này mở rộng khả năng tích hợp với các thiết bị ngoại vi, tạo ra nhiều cơ hội và linh hoạt cho việc kết nối và tương tác của vi điều khiển.
- Các chân GPIO cũng có khả năng được cấu hình để hoạt động với ngắt (interrupt). Chức năng interrupt cho phép vi điều khiển phản ứng ngay lập tức với các sự kiện cụ thể, tăng cường khả năng phản hồi và hiệu suất của nó. Bằng cách sử dụng interrupt, vi điều khiển có thể ưu tiên các nhiệm vụ quan trọng và nhanh chóng xử lý các hoạt động đòi hỏi thời gian, giúp nâng cao hiệu quả toàn diện của hệ thống.

GPIO đóng vai trò là cầu nối kết nối giữa vi điều khiển và thế giới bên ngoài. Đặc tính linh hoạt và khả năng làm việc với interrupt làm cho GPIO trở thành một công cụ mạnh mẽ, đặc biệt hữu ích cho việc phát triển các hệ thống nhúng và thực hiện đa dạng ứng dụng trong lĩnh vực điện tử và tự động hóa.

## 2.7.2 UART

- **Định nghĩa**

UART (Universal Asynchronous Receiver/Transmitter) là một chuẩn giao thức truyền thông phổ biến, thường được sử dụng trong các hệ thống điện tử và vi điều khiển. Nó cho phép truyền dữ liệu giữa các thiết bị thông qua kết nối điểm-điểm, làm cho quá trình tín hiệu linh hoạt và hiệu quả.

- **Nguyên lý hoạt động**

- UART thực hiện theo cơ chế bất đồng bộ (asynchronous), tức là không yêu cầu sự đồng bộ thông qua một tín hiệu clock chung giữa các thiết bị. Thay vào đó, nó tận dụng các khung dữ liệu để định dạng và đồng bộ hóa quá trình truyền và nhận dữ liệu. Mỗi khung dữ liệu bao gồm các bit dữ liệu, parity bit để kiểm tra lỗi và các bit đồng bộ, bao gồm start bit để bắt đầu khung dữ liệu và stop bit để đánh dấu kết thúc khung dữ liệu. Điều này tạo ra một phương thức truyền thông hiệu quả và linh hoạt mà không cần tới sự đồng bộ hóa thông qua tín hiệu clock chung.
- UART có khả năng hoạt động ở nhiều tốc độ truyền khác nhau, được đo lường bằng tốc độ baud (baud rate). Tốc độ baud định rõ số lượng bit được truyền trong một giây, và đối với quá trình truyền thông tin chính xác, tốc độ này cần phải được đồng bộ giữa cả hai thiết bị truyền và nhận. Điều này đồng nghĩa với việc cấu hình chính xác tốc độ baud trên cả hai đầu của liên kết truyền thông để đảm bảo sự tương thích và chính xác trong quá trình truyền dữ liệu.

- **Đặc tính nổi bật**

9600 và 115200 là hai tốc độ baud phổ biến được lựa chọn trong nhiều ứng dụng.

- Tốc độ baud 9600 được ưa chuộng trong nhiều ứng dụng, ví dụ như truyền thông giữa vi điều khiển và các module, cảm biến, màn hình LCD, hoặc khi giao tiếp với các thiết bị ngoại vi khác. Tốc độ này đủ nhanh để xử lý việc truyền dữ liệu thông thường và đảm bảo độ tin cậy trong hầu hết các tình huống thông thường.
- Tốc độ baud 115200 là một lựa chọn phổ biến trong các ứng dụng đòi hỏi tốc độ truyền nhanh hơn, như truyền dữ liệu video, âm thanh, hoặc trong các tình huống cần xử lý dữ liệu nhanh. Với khả năng cung cấp tốc độ truyền đáng kể và độ tin cậy, nó là sự chọn lựa lý tưởng cho các ứng dụng đòi hỏi băng thông lớn và hiệu suất cao.

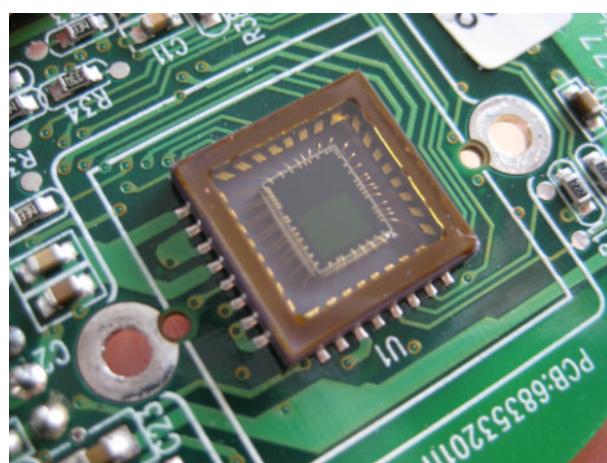
## 2.8 Những kỹ thuật trong xử lý ảnh và thị giác máy tính

### 2.8.1 Một số đặc điểm về usb camera

Webcam là một máy ảnh kỹ thuật số nhỏ gọn, có thể được kết nối với máy tính để phát hình ảnh video trong thời gian thực. Giống như một máy ảnh kỹ thuật số, nó thu nhận ánh sáng qua một ống kính nhỏ ở phía trước bằng cách sử dụng một mảng lưới các cảm biến ánh sáng siêu nhỏ được tích hợp vào một vi mạch cảm biến hình ảnh (CCD (charge-coupled device) hoặc CMOS chip). Cảm biến hình ảnh và mạch điện của nó chuyển đổi hình ảnh trước ống kính thành định dạng kỹ thuật số—một chuỗi nhị phân mà máy tính có thể xử lý được[19].

Khác với máy ảnh kỹ thuật số, webcam không có chip nhớ hoặc bộ nhớ flash vì nó không cần phải ghi nhớ hình ảnh vì chúng được thiết kế để chụp và truyền chúng ngay lập tức đến máy tính[19].

Image sensor (CCD hoặc là CMOS chip), là vật hình vuông ở giữa ống kính. Chỉ phần nhỏ màu xanh lá cây ở trung tâm là nhạy sáng: phần còn lại của chip liên quan đến việc kết nối cảm biến ánh sáng với mạch lớn hơn bao quanh nó[19].



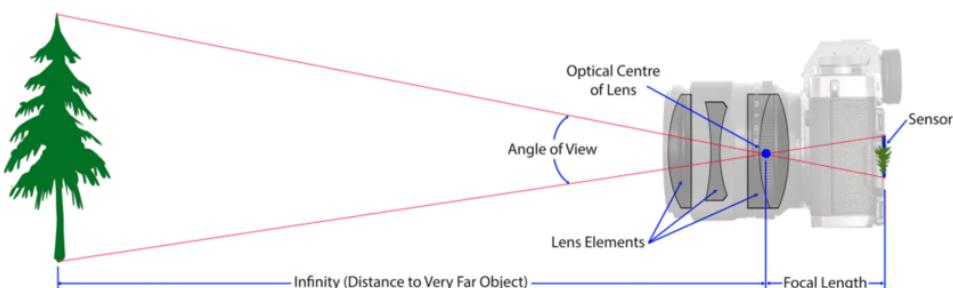
Hình 2.24: Image Sensor[19]

Image sensor được cấu tạo từ hàng triệu ô vuông nhỏ nhạy sáng sắp xếp theo mô hình lưới liền kề nhau. Những ô vuông này được gọi là điểm ảnh (pixel). Webcam cơ bản sử dụng cảm biến nhỏ với chỉ vài trăm nghìn điểm ảnh (thường là  $640 \times 480$ ). Với những camera có độ phân giải cao hơn thì những điểm ảnh này có thể lên tới hàng

triệu[19].

Hình ảnh vật thể trong không gian sẽ được chuyển đổi trực tiếp thành tín hiệu số nếu image sensor là CMOS chip. Trái lại, đối với image sensor là CCD hình ảnh sẽ được chuyển đổi thành tín hiệu analog trước khi đến bộ ADC.

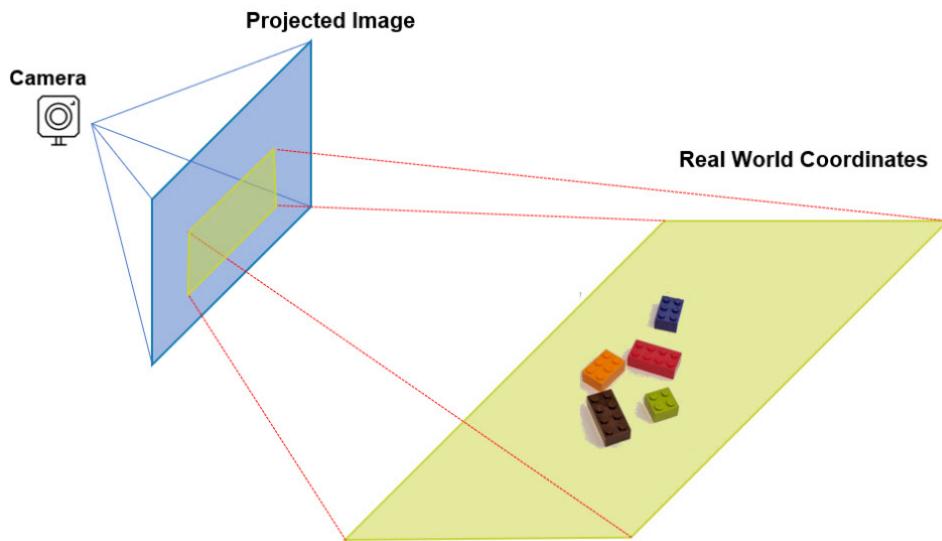
Sau khi đã chuyển đổi hình ảnh vật thể về tín hiệu số, các giá trị số từ tất cả các điểm ảnh được kết hợp lại để tạo thành một hình ảnh kỹ thuật số hai chiều. Mỗi điểm ảnh trong hình ảnh kỹ thuật số tương ứng với một điểm trên cảm biến hình ảnh. Trong quá trình này, dữ liệu thu thập từ các điểm ảnh cá nhân trên cảm biến được dịch thành một lưới cấu trúc của các điểm ảnh trong hình ảnh kỹ thuật số cuối cùng. Giá trị của mỗi điểm ảnh đại diện cho cường độ và thông tin màu sắc được ghi lại tại điểm cụ thể đó trên cảm biến. Sự kết hợp của các giá trị pixel tạo thành một biểu đồ đồng nhất của cảnh gốc, được biến đổi thành một hình ảnh phản phù hợp để hiển thị trên màn hình hoặc lưu trữ dưới định dạng kỹ thuật số. Quá trình chuyển đổi này đảm bảo rằng sắp xếp không gian và đặc điểm hình ảnh của cảnh được bảo tồn đúng mức trong bản diễn giải kỹ thuật số.



Hình 2.25: Formation of an image in a camera[20]

## 2.8.2 Kỹ thuật hiệu chỉnh camera (camera calibration)

Camera calibration là quá trình xác định các thông số và tham số của một hệ thống camera để tái tạo không gian 3D của một cảnh, vật thể nào đó được camera ghi lại được. Quá trình này bao gồm việc đo lường các thông số của ống kính và cảm biến hình ảnh, cũng như các sai lệch hình học và nghiêng của camera.



Hình 2.26: Transformation from image to real object [21]

Một số camera pinhole hiện nay gây ra nhiều biến dạng (distortion) cho hình ảnh. Hai biến dạng chính là biến dạng tâm và biến dạng tiếp xúc. Do biến dạng tâm, các đường thẳng sẽ xuất hiện cong. Hiệu ứng của nó càng lớn khi chúng ta di chuyển xa khỏi trung tâm của hình ảnh. Một biến dạng khác là biến dạng tiếp xúc xảy ra vì ống kính chụp hình không được căn chỉnh hoàn hảo song song với mặt phẳng hình ảnh. Vì vậy, một số khu vực trong hình ảnh có thể trông gần hơn so với mong đợi. Đại diện cho những điều này là hệ số biến dạng *Distortion coefficients*[22]

Một thông số quan trọng dùng để tái tạo không gian 3D là *camera matrix* bao gồm các thông số là tiêu cự ( $f_x, f_y$ ), trung tâm quang học ( $c_x, c_y$ ), ... và được biểu diễn thành ma trận 3x3:

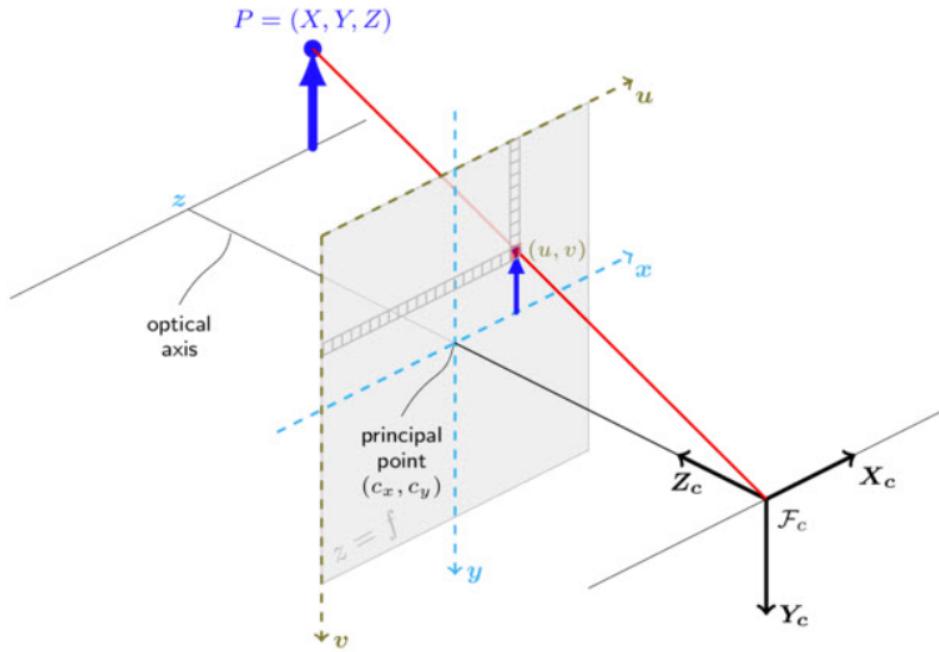
$$\text{camera\_matrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Các tham số còn lại tương ứng với các vector quay và dịch chuyển, dịch các tọa độ của một điểm 3D sang một hệ tọa độ. Từ đó ta có phương trình tổng quát như sau:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Chúng ta cần tìm là các thông số (X, Y, Z) là tọa 3D của vật thể trong không gian thật và là mục tiêu chúng ta cần tìm. Trong khi các thông số (u, v) là phép chiếu tọa

độ điểm trên pixel. Các thông số  $r$ ,  $t$  là các phép quay vector và dịch chuyển, dịch tọa độ. Những thông số này có thể được tìm thông qua các bước trong [21]



Hình 2.27: Transformation from image to real object [21]

## 2.9 Nguyên lý nhận dạng vật thể

### 2.9.1 Tiền xử lý

Tiền xử lý ảnh là quá trình xử lý ảnh được thực hiện trước khi áp dụng các phương pháp phân tích, nhận dạng hoặc xử lý chính. Mục tiêu của tiền xử lý là cải thiện chất lượng của hình ảnh, làm sạch dữ liệu, loại bỏ nhiễu, chuẩn hóa các đặc trưng, và chuẩn bị dữ liệu cho các bước xử lý tiếp theo.

Các công việc phổ biến trong tiền xử lý ảnh bao gồm:

- **Làm sạch dữ liệu:** Loại bỏ nhiễu và giảm độ sáng nền là các bước quan trọng để cải thiện chất lượng hình ảnh và video. Việc loại bỏ các đối tượng không mong muốn giúp duy trì thông tin chính xác và tránh làm sai lệch nội dung. Những yếu tố này có thể gây ra mất mát hoặc thay đổi thông tin so với thực tế, làm cho việc đánh giá nội dung trở nên không chính xác. Để khử nhiễu, người ta dùng các bộ lọc như: bộ lọc Gaussian, bộ lọc trung bình, bộ lọc trung vị, bộ lọc trung vị thích nghi,...
- **Cân bằng màu sắc và độ tương phản:** Điều chỉnh màu sắc và độ tương phản để làm nổi bật các đặc điểm quan trọng của hình ảnh. Khi tăng cường độ tương phản, sẽ gia tăng khả năng phân biệt các đối tượng trong ảnh.
- **Cắt ảnh và thay đổi kích thước:** Cắt hoặc thay đổi kích thước của hình ảnh để loại bỏ phần không cần thiết hoặc tạo ra độ phân giải mong muốn.
- **Chuẩn hóa:** Đảm bảo rằng dữ liệu đầu vào đồng nhất về độ sáng, độ tương phản, và tỉ lệ khung hình.
- **Phát hiện và loại bỏ nền:** Phát hiện và loại bỏ nền không mong muốn trong hình ảnh để tập trung vào các đối tượng quan trọng.
- **Biến đổi hình dạng và góc nhìn:** Điều chỉnh hình dạng và góc nhìn của đối tượng trong hình ảnh để tạo ra góc nhìn đồng nhất hoặc chuẩn hóa.
- **Làm mềm và làm sắc nét:** Làm mềm hoặc làm sắc nét hình ảnh để giảm nhiễu hoặc tăng độ chi tiết.

### 2.9.2 Trích xuất đặc trưng

Trích xuất đặc trưng là quá trình tìm ra và rút trích thông tin quan trọng từ dữ liệu đầu vào. Trong ngữ cảnh của xử lý ảnh, trích xuất đặc trưng liên quan đến việc xác

định và trích xuất các đặc điểm độc đáo hoặc quan trọng từ hình ảnh để làm nổi bật các đặc tính của đối tượng hoặc khu vực quan tâm. Các đặc trưng có thể là các thuộc tính như cạnh, điểm nổi bật, màu sắc, hình dạng, hoặc các đặc điểm khác của hình ảnh.

Quá trình trích xuất đặc trưng thường bao gồm các bước sau:

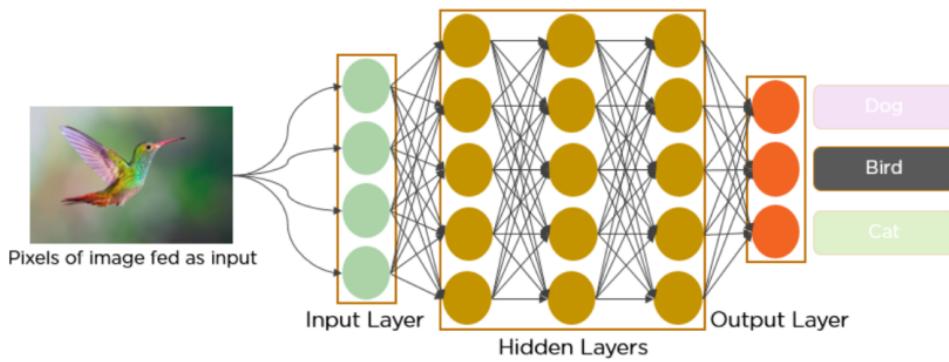
- **Chọn lọc đặc trưng:** Lựa chọn các phương pháp phù hợp để trích xuất các đặc trưng phù hợp với mục tiêu cụ thể của ứng dụng
- **Trích xuất thông tin:** Sử dụng các kỹ thuật phù hợp để trích xuất các đặc trưng từ hình ảnh, như bộ lọc, phát hiện biên, phát hiện điểm nổi bật, hoặc phân tích màu sắc
- **Biểu diễn đặc trưng:** Biểu diễn các đặc trưng được trích xuất dưới dạng vectơ hoặc các cấu trúc dữ liệu khác để làm cho chúng có thể sử dụng được cho các mục đích tiếp theo như nhận dạng hoặc phân loại
- **Kiểm tra và điều chỉnh:** Kiểm tra hiệu suất của các đặc trưng trích xuất và điều chỉnh các tham số hoặc phương pháp nếu cần thiết để cải thiện hiệu suất của hệ thống

### 2.9.3 Định vị và phân loại vật thể

Định vị vật thể là quá trình xác định chính xác vị trí của vật thể trong ảnh. Sau khi phát hiện vật thể, các thuật toán định vị sẽ tập trung vào việc xác định kích thước và vị trí cụ thể của nó trong khung hình. Các phương pháp phổ biến để định vị vật thể bao gồm sử dụng các mô hình Deep Learning như Faster R-CNN, RetinaNet,...

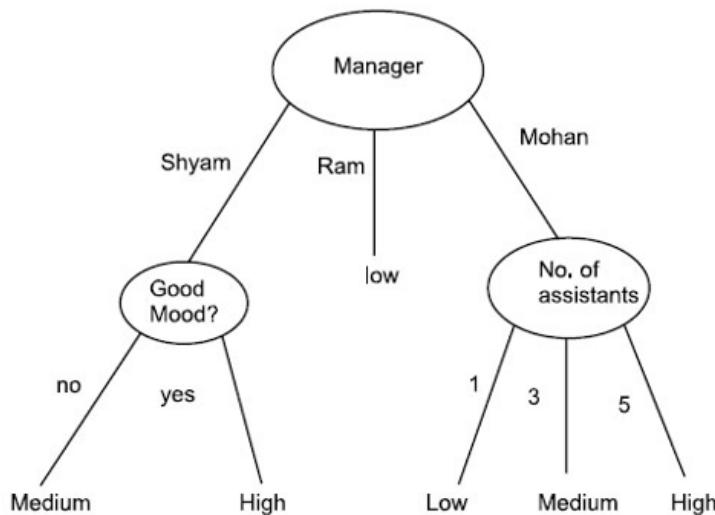
Sau khi định vị xong, bước tiếp theo là phân loại vật thể trong ảnh vào các danh mục hoặc nhóm tương đương dựa trên các đặc trưng đã trích xuất. Để phân loại vật thể, chúng ta cần sử dụng các mô hình học máy có khả năng học và tự động rút ra các quy luật và tính chất từ dữ liệu huấn luyện. Một số phương pháp phổ biến để phân loại vật thể là:

- **SVM (Support Vector Machines):** Sử dụng siêu phẳng để phân loại các điểm dữ liệu vào các nhóm khác nhau
- **K-Nearest Neighbors (KNN):** Dựa trên khoảng cách đến các điểm lân cận để phân loại
- **Neural Networks và Deep Learning:** Các mạng nơ-ron sâu, như CNN (Convolutional Neural Networks), có khả năng học và phân loại các đặc trưng phức tạp trong dữ liệu hình ảnh



Hình 2.28: Convolution Neutral Network

- **Random Forests:** Sử dụng nhiều cây quyết định để phân loại và đưa ra dự đoán chính xác



Hình 2.29: Decision Tree

Nhờ vào các kỹ thuật này, quá trình định vị và phân loại vật thể trong ảnh trở nên chính xác và hiệu quả hơn, mở ra nhiều ứng dụng trong thực tế như giám sát an ninh, nhận dạng khuôn mặt, và tự động hóa quy trình sản xuất.

## 2.9.4 Hậu xử lý

Hậu xử lý ảnh là giai đoạn xử lý dữ liệu sau khi ảnh đã được chụp và các bước tiền xử lý cơ bản đã được thực hiện. Mục tiêu của hậu xử lý là cải thiện chất lượng ảnh, làm nổi bật các đặc điểm quan trọng, và chuẩn bị ảnh cho các ứng dụng hoặc phân tích tiếp theo. Các kỹ thuật hậu xử lý thường bao gồm:

- **Làm mịn và làm sắc nét ảnh:** Sử dụng các bộ lọc để giảm nhiễu hoặc tăng độ chi tiết của ảnh
- **Chỉnh sửa màu sắc và độ tương phản:** Điều chỉnh màu sắc, độ tương phản, và độ sáng để hình ảnh trông tự nhiên hơn hoặc làm nổi bật các chi tiết cần thiết
- **Hiệu chỉnh biến dạng:** Sửa chữa các biến dạng quang học do ống kính gây ra, như biến dạng tâm (radial distortion) và biến dạng tiếp xúc (tangential distortion)
- **Tách nền và đối tượng:** Sử dụng các kỹ thuật để tách đối tượng chính ra khỏi nền, giúp dễ dàng nhận diện và phân tích hơn
- **Ghép ảnh và tạo ảnh toàn cảnh:** Kết hợp nhiều hình ảnh lại với nhau để tạo thành một hình ảnh lớn hơn hoặc toàn cảnh
- **Phát hiện và đánh dấu đối tượng:** Xác định và đánh dấu các đối tượng trong ảnh, chẳng hạn như nhận diện khuôn mặt hoặc biển số xe
- **Chuyển đổi định dạng và nén:** Chuyển đổi hình ảnh sang các định dạng khác nhau hoặc nén để tiết kiệm dung lượng lưu trữ

## 2.10 Một số công nghệ xử lý ảnh và thị giác máy tính phổ biến

### 2.10.1 OpenCV

#### 2.10.1.1 Sơ lược về OpenCV

OpenCV (Open Source Computer Vision Library) là một thư viện mã nguồn mở chứa hàng trăm thuật toán về thị giác máy tính, được sử dụng rộng rãi trong xử lý ảnh và video. Thư viện này cung cấp các công cụ và thuật toán cho nhiều ứng dụng, bao gồm phát hiện đối tượng, theo dõi vật thể, phân tích chuyển động, phân loại hình ảnh, ... API của OpenCV 2.x được viết bằng ngôn ngữ C++, trong khi phiên bản API OpenCV 1.x dựa trên ngôn ngữ C. Phiên bản hiện tại của OpenCV là 5.0.0.

OpenCV (Open Source Computer Vision Library) có cấu trúc module và bao gồm một số thư viện tĩnh (static libraries) và thư viện động (shared libraries). Thư viện này cung cấp một số module chính như:

- **Chức năng cốt lõi (Core functionality - core):** Đây là module xác định các cấu trúc dữ liệu cơ bản, bao gồm mảng đa chiều và các chức năng cơ bản được sử dụng bởi tất cả các module khác
- **Xử lý ảnh (Image Processing - imgproc):** Module này dùng để xử lý ảnh, bao gồm các bộ lọc ảnh tuyến tính và phi tuyến tính, biến đổi hình học (thay đổi kích thước, xoay, lật,...), chuyển đổi không gian màu, histogram và nhiều chức năng khác
- **Phân tích video (Video analysis):** Hỗ trợ xử lý video thông qua các hàm như xác định chuyển động, theo dõi vật thể, trích xuất khung hình, ghép nối các khung hình, và nén video
- **Phát hiện đối tượng và Học máy (Object Detection - objdetect):** Module này giúp phát hiện đối tượng và các trường hợp của các lớp được định nghĩa trước (ví dụ: khuôn mặt, mắt, người, ô tô, v.v.), từ đó tạo nên các ứng dụng như phân loại hình ảnh, nhận dạng khuôn mặt, nhận dạng vật thể, và phân tích cảm xúc

OpenCV đã trở thành một công cụ cực kỳ phổ biến trong lĩnh vực thị giác máy tính và xử lý ảnh. Nó hỗ trợ nhiều ngôn ngữ lập trình như C++, Python, và Java,

với các liên kết API cho các ngôn ngữ này. OpenCV cũng có sẵn trên nhiều nền tảng khác nhau, bao gồm Windows, Linux, MacOS, iOS và Android.

Các ứng dụng của OpenCV rất đa dạng, bao gồm cả lĩnh vực công nghiệp và khoa học như robotic, xe tự hành, an ninh, sản xuất, ... Với sự đa dạng và hiệu quả của mình, OpenCV được coi là một trong những thư viện xử lý hình ảnh và video phổ biến nhất trên thế giới.

### 2.10.1.2 Khả năng tương thích ngôn ngữ

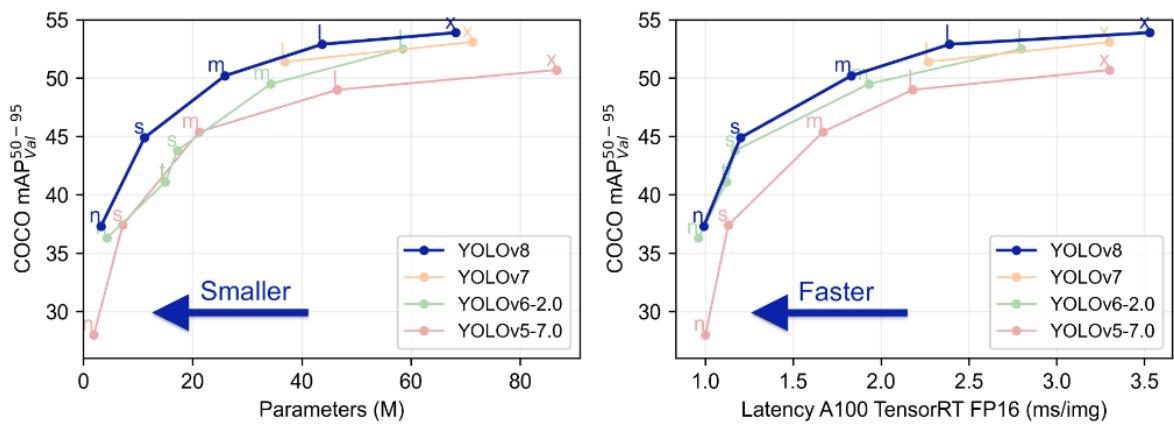
OpenCV hỗ trợ, cung cấp các API trên nhiều ngôn ngữ khác nhau giúp người dùng dễ dàng hơn trong việc phát triển ứng dụng:

- **C++:** Ngôn ngữ này cung cấp nhiều tùy chọn và được sử dụng phổ biến nhất. Khi sử dụng Visual Studio làm IDE, C++ là lựa chọn tốt nhất. Mặc dù ban đầu có thể phức tạp, nhưng các thiết lập của C++ mang lại nhiều lợi ích khi phát triển sản phẩm trong tương lai
- **Python:** OpenCV-Python có ưu điểm là không cần thiết lập nhiều và mã ngắn gọn. Vì vậy, Python thường được sử dụng để thử nghiệm OpenCV. Ngoài ra, Python còn cho phép lập trình trên nhiều hệ điều hành khác nhau
- **Java:** Ngôn ngữ này khá giống với C++, với ưu điểm là tốc độ nhanh và tính đa nền tảng
- **C#:** Ưu điểm của C# là được hỗ trợ bởi thư viện đa nền tảng EmguCV, cho phép viết mã nhanh chóng và tiện lợi

### 2.10.2 YOLOv8

YOLOv8 là phiên bản mới nhất của YOLO do Ultralytics phát triển. Là một mô hình tiên tiến và hiện đại, YOLOv8 kế thừa sự thành công của các phiên bản trước, giới thiệu các tính năng và cải tiến mới nhằm nâng cao hiệu suất, tính linh hoạt và hiệu quả. YOLOv8 hỗ trợ đầy đủ các nhiệm vụ trí tuệ nhân tạo và thị giác máy tính, bao gồm phát hiện đối tượng, phân đoạn, ước lượng tư thế, theo dõi và phân loại. Sự đa dạng này cho phép người dùng tận dụng khả năng của YOLOv8 trong nhiều ứng dụng và lĩnh vực khác nhau[23].

So với các phiên bản trước YOLOv8 đã có những cải thiện đáng kể:



Hình 2.30: YOLO version comparation[24]

Khi nhìn về 2 biểu đồ so sánh ở trên, ta có thể thấy rằng, về cơ bản với số lượng tham số ít hơn hẳn so với các phiên bản trước đó nhưng độ chính xác của model nhận diện lại tăng lên. Điều này sẽ giúp giảm khối lượng biến cần nhớ cho bộ xử lý và tăng tốc độ nhận diện hay giảm độ trễ trong việc nhận diện hay xử lý ảnh. Tuy nhiên với các model loại l và x, YOLOv8 gấp vấn đề với sự trễ hơn trong việc nhận diện so với loại model loại l và x của các version trước.

Ngoài việc tối ưu về các giải thuật deep learning để tăng độ chính xác của model thì ở phiên bản YOLOv8 này, một tính năng mới xuất hiện là chúng ta có thể sử dụng YOLOv8 thông qua CLI hoặc là python code giống như các phiên bản trước.

## CHƯƠNG 3

### THIẾT KẾ VÀ HIỆN THỰC HỆ THỐNG

Trong chương này, chúng tôi sẽ trình bày chi tiết về cấu tạo, thiết kế của cánh tay robot - đối tượng nghiên cứu chính của đề tài. Sau khi đã hoàn thiện cơ khí và điện tử cho cánh tay robot, chúng tôi tiến hành thiết kế và hiện thực driver cho robot, đồng thời đề xuất một số ứng dụng để có thể kiểm chứng được tính đúng đắn và độ chính xác của driver robot.

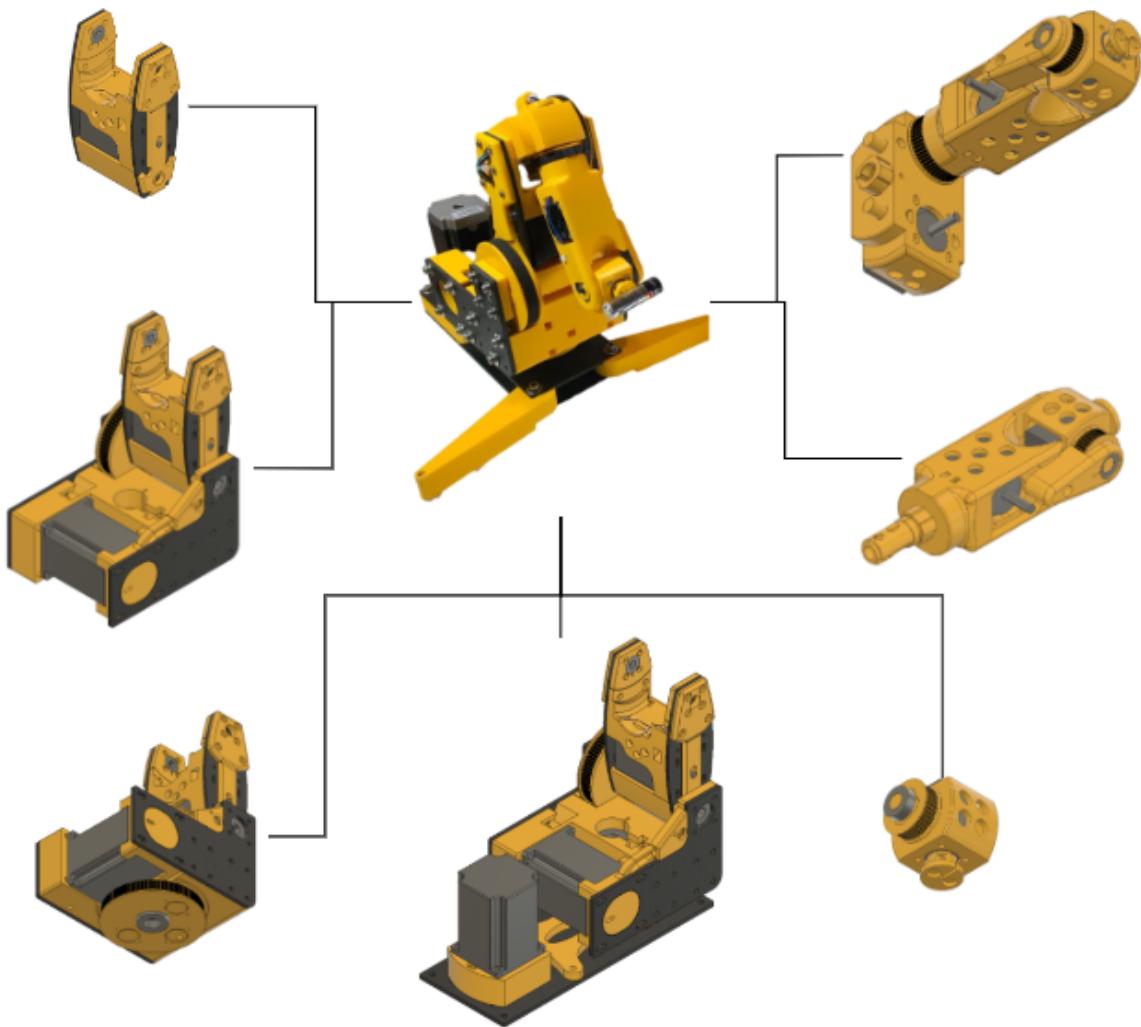
#### 3.1 Thiết kế robot JetArm

Ở phần này nhóm sẽ tập trung giới thiệu chi tiết về cấu tạo cơ khí và điện tử của robot JetArm. Robot này là một sản phẩm của công nghệ in 3D, thiết kế được tham khảo và lấy ý tưởng từ dự án mã nguồn mở trên Github [5].

Qua việc trực tiếp lắp ráp và hoàn thiện yếu tố cơ điện tử của cánh tay robot, chúng tôi đã nắm rõ được cấu tạo của robot, từ đó mới có thể thiết kế và triển khai driver phù hợp cho cánh tay robot.

Đây cũng là một điều hết sức may mắn của nhóm khi được thầy Lê Trọng Nhân và anh Cao Hùng tận tình chỉ dẫn, tạo điều kiện để nhóm có cơ hội được tiếp xúc với các thành phần cơ khí và điện tử - điều rất quan trọng đối với một kỹ sư khi có thể nắm rõ được hệ thống của mình đang làm việc. Đây cũng là cơ hội để nâng cao kiến thức về cơ khí và củng cố các kiến thức về điện tử đã được học trước đây.

### 3.1.1 Cấu tạo cơ khí của robot JetArm

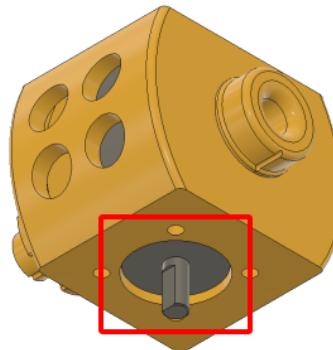


Hình 3.1: Cấu tạo của robot JetArm

Hình trên là một bản lắp ráp cơ bản cấu tạo nên robot JetArm, các thành phần cấu tạo nên mô hình trên gồm 6 trực tự do được điều khiển bởi các Nema Motors, dưới đây là phân tích chi tiết về cấu tạo của robot JetArm.

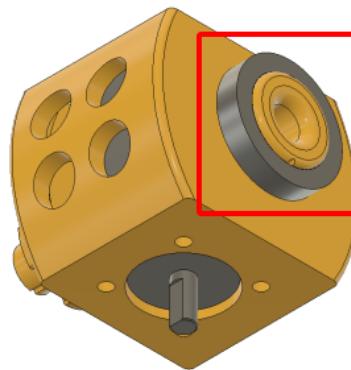
- Trục tự do J6 (Joint 6):

- Động cơ được trục tự do J6 sử dụng là Nema 8 và được cố định bằng 4 vít M2x6mm



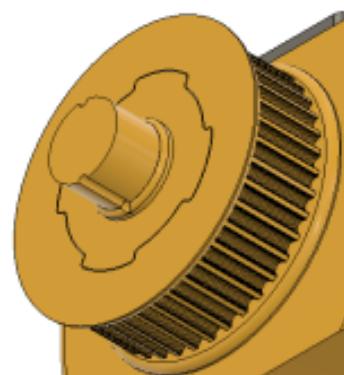
Hình 3.2: Cấu tạo trục tự do J6

- Vòng bi (21x15x4mm) được lắp ở một bên



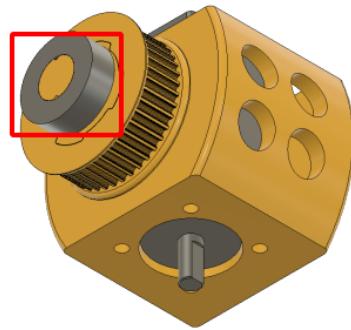
Hình 3.3: Cấu tạo trục tự do J6

- Mặt còn lại được lắp Pulley42 in 3D



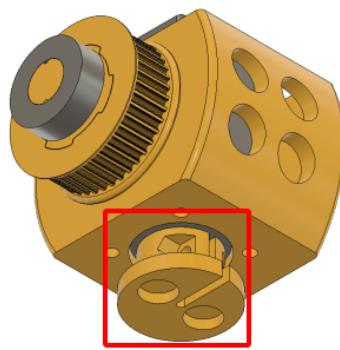
Hình 3.4: Cấu tạo trục tự do J6

- Vòng bi (16x8x5mm) được lắp cùng một phía

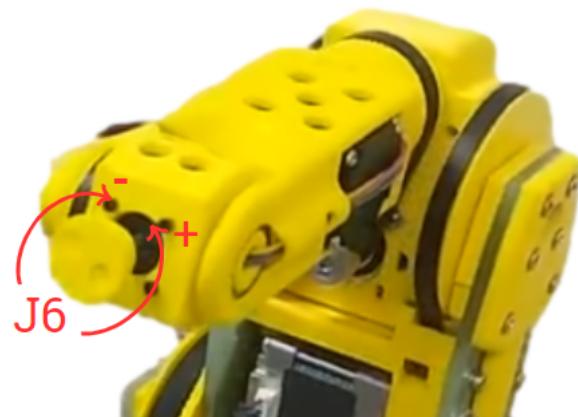


Hình 3.5: Cấu tạo trực tự do J6

- Hai nam châm nhỏ có thể được dán vào hai lỗ trên đầu của J6

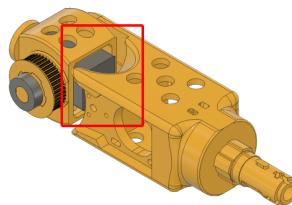


Hình 3.6: Cấu tạo trực tự do J6



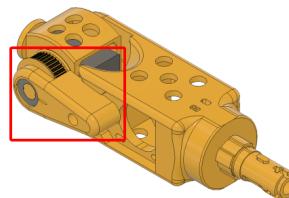
Hình 3.7: Trục tự do J6 trong robot JetArm

- Trục tự do J5 (Joint 5):
  - J6 được lắp đặt trên phần của J5



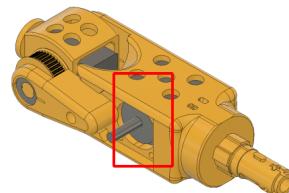
Hình 3.8: Cấu tạo trục tự do J5

- J5Bracket được cố định bằng ba chốt, vít M3x25mm và đai ốc M3

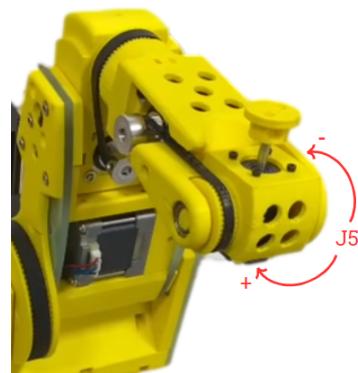


Hình 3.9: Cấu tạo trục tự do J5

- Động cơ Nema 11 với ròng rọc GT2 20 răng và đai răng GT2 87 được cố định bằng bốn vít M2.5x8mm và vòng đệm M2.5

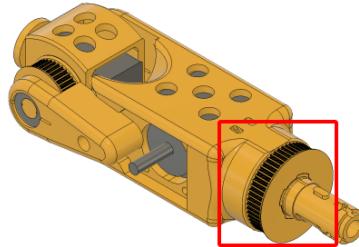


Hình 3.10: Cấu tạo trục tự do J5



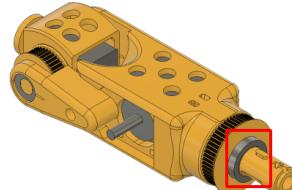
Hình 3.11: Trục tự do J5 trong robot JetArm

- Trục tự do J4 (Joint 4):
  - Được lắp vào Pulley56



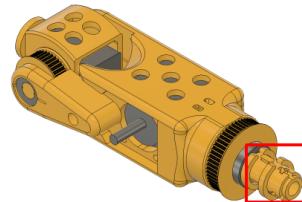
Hình 3.12: Cấu tạo trục tự do J4

- Được lắp vòng bi (21x15x4mm)



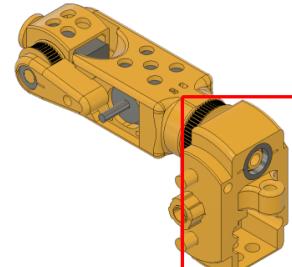
Hình 3.13: Cấu tạo trục tự do J4

- Axis4limit được lắp đặt sau ố trục



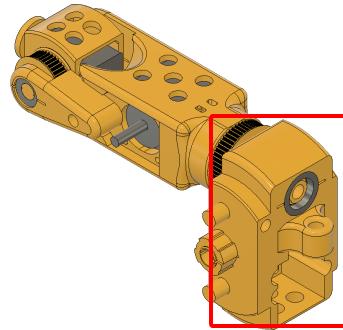
Hình 3.14: Cấu tạo trục tự do J4

- Axis5screwHolder và vòng bi (21x15x4mm) được lắp đặt sau đó



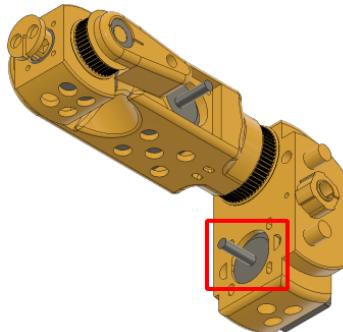
Hình 3.15: Cấu tạo trục tự do J4

- Axis5screwHolder và vòng bi (21x15x4mm) được lắp đặt sau đó



Hình 3.16: Cấu tạo trục tự do J4

- Động cơ Nema 11 với puli răng GT2 20 và đai răng GT2 87 được cố định bằng bốn vít M2.5x8mm và vòng đệm M2.5 sau cùng



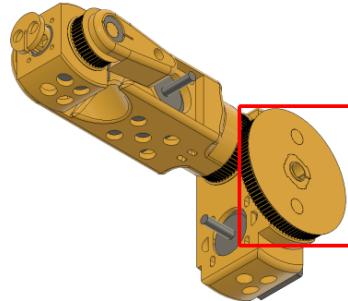
Hình 3.17: Cấu tạo trục tự do J4



Hình 3.18: Trục tự do J4 trong robot JetArm

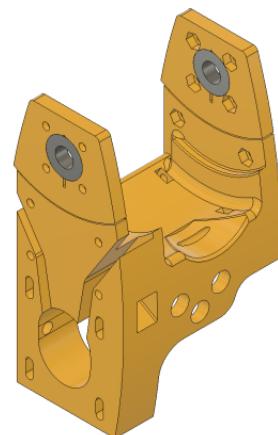
- Trục tự do J3 (Joint 3):

– Dùng Pulley100 lắp vào phần J4 phía trên



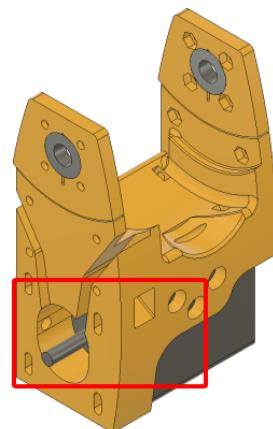
Hình 3.19: Cấu tạo trục tự do J3

– Axis3part2 với tám đai ốc M3 được lắp đặt được sử dụng



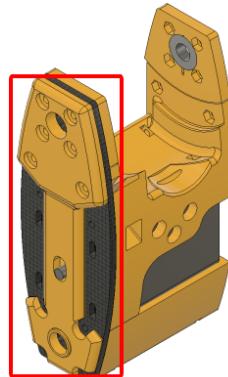
Hình 3.20: Cấu tạo trục tự do J3

– Lắp động cơ Nema 17 với ròng rọc GT2 20 răng và đai răng GT2 139 vào Axis3part2



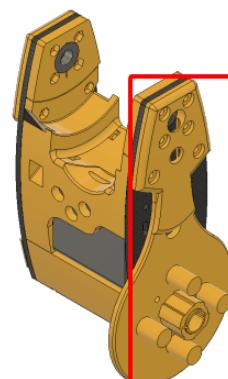
Hình 3.21: Cấu tạo trục tự do J3

- Bao quanh Axis3part2 bằng Axis3part3 với tám đai ốc M3 được lắp đặt  
được cố định bằng bốn vít M3x20mm và đai ốc M3

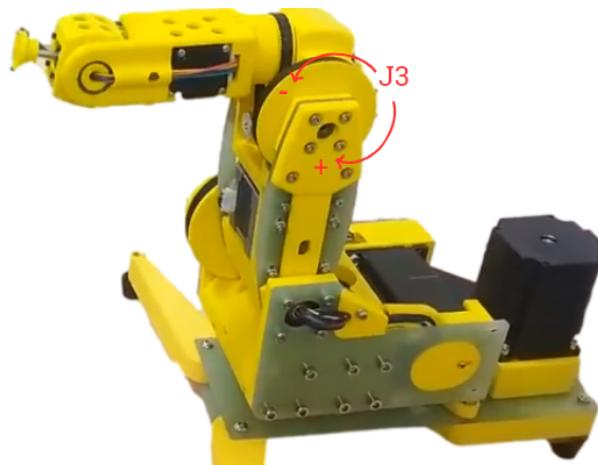


Hình 3.22: Cấu tạo trục tự do J3

- Bên phía còn lại được lắp đặt và cố định bằng sáu M3x12mm



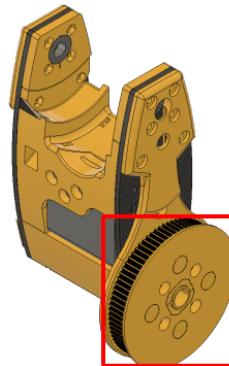
Hình 3.23: Cấu tạo trục tự do J3



Hình 3.24: Trục tự do J3 trong robot JetArm

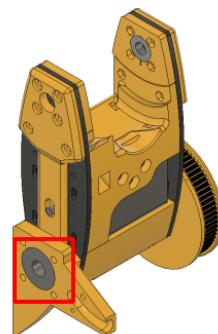
- Trục tự do J2 (Joint 2):

- Axis2Pulley được lắp đặt và cố định bằng bốn vít M3x30mm và vòng đệm M3 trên trục tự do J3



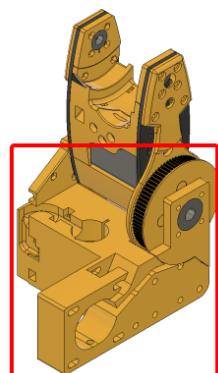
Hình 3.25: Cấu tạo trục tự do J2

- Axis2bearHolder với bốn đai ốc M4 được lắp đặt được cố định trên một trong các ô trục



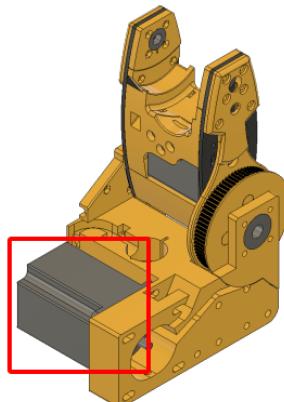
Hình 3.26: Cấu tạo trục tự do J2

- Axis2part2 được lắp đặt ở một bên của J2



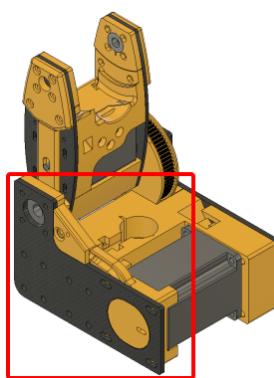
Hình 3.27: Cấu tạo trục tự do J2

- Động cơ Nema 23 với ròng rọc GT2 3 mm 20 răng và dây đai GT2 3 mm 128 răng được lắp đặt sau đó



Hình 3.28: Cấu tạo trục tự do J2

- Axis2MotorOp được lắp đặt ở phía cùn lại và cố định bằng vít M3x12mm, vòng đệm M3 và dai ốc M3



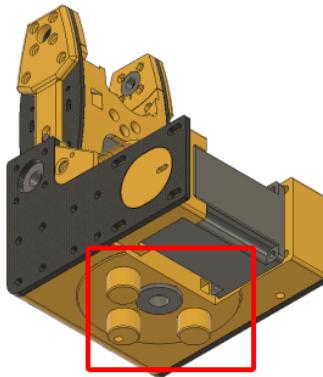
Hình 3.29: Cấu tạo trục tự do J2



Hình 3.30: Trục tự do J2 trong robot JetArm

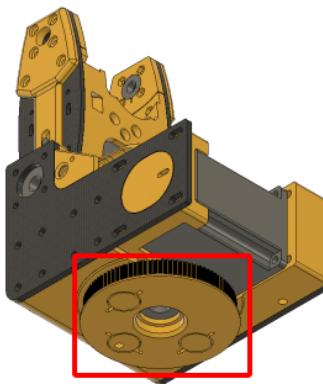
- Trục tự do J1 (Joint 1):

– Ở bi sẽ được lắp đặt ở phía đáy và phía trên bệ mặt (OD 12mm) của J2



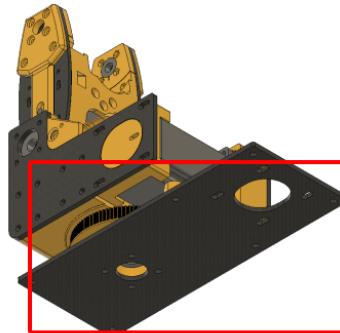
Hình 3.31: Cấu tạo trục tự do J1

– Axis1Pulley được lắp trên ổ bi dưới đáy



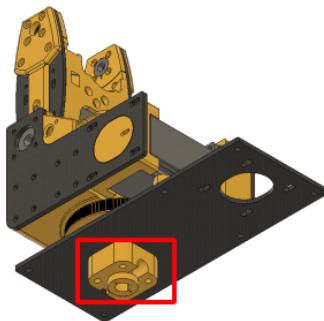
Hình 3.32: Cấu tạo trục tự do J1

– Axis1washer được lắp đặt bốn đai ốc M5 nằm dưới ròng rọc và lắp thêm một tấm nhựa



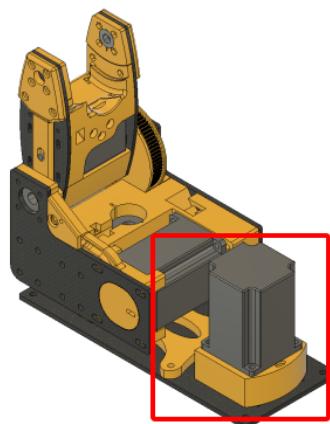
Hình 3.33: Cấu tạo trục tự do J1

- Axis1Holder được lắp vào ống và cố định bằng hai vít M3x16mm, vòng đệm M3 và đai ốc M3



Hình 3.34: Cấu tạo trục tự do J1

- Axis1MotorHolder nằm trên Axis1part2, nơi mà lắp đặt động cơ Nema 23 với ròng rọc GT2 3 mm 20 răng và dây đai GT2 3 mm 152 răng tiếp tục đĩnh của Axis1MotorHolder



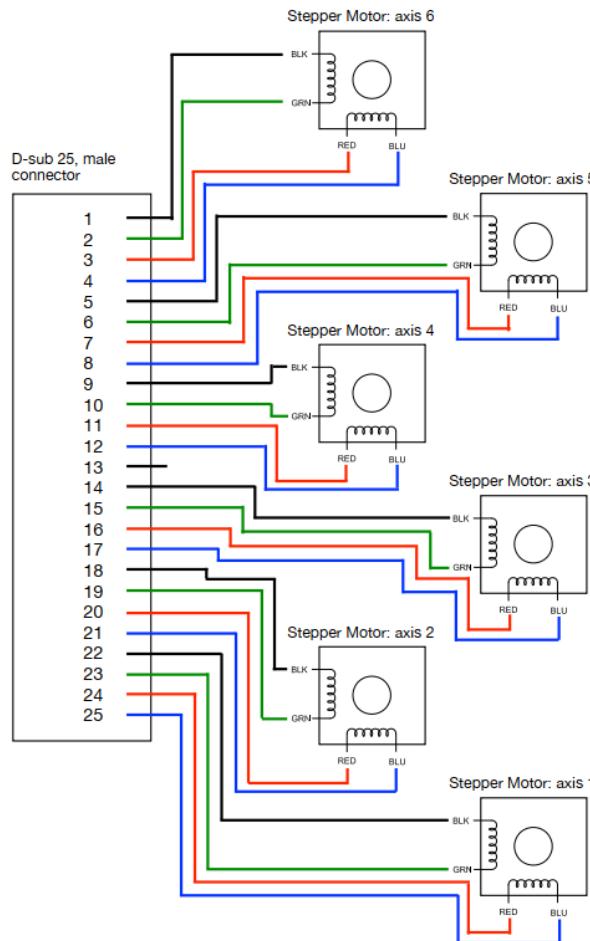
Hình 3.35: Cấu tạo trục tự do J1



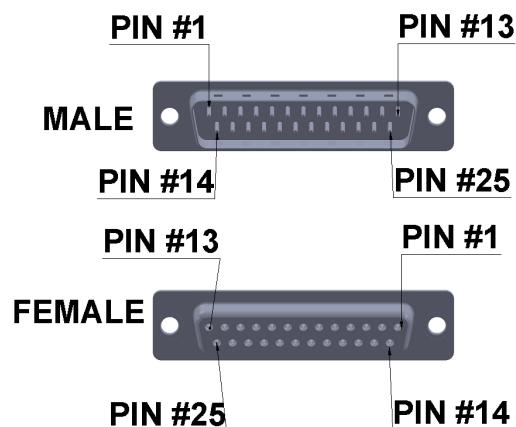
Hình 3.36: Trục tự do J1 trong robot JetArm

### 3.1.2 Kết nối điện tử trong robot JetArm

Hình ảnh dưới đây mô tả chi tiết kết nối của các động cơ bước trong Robot 6DoF vào connector D-sub 25 [5]:



Hình 3.37: Sơ đồ đi dây giữa các trục tự do

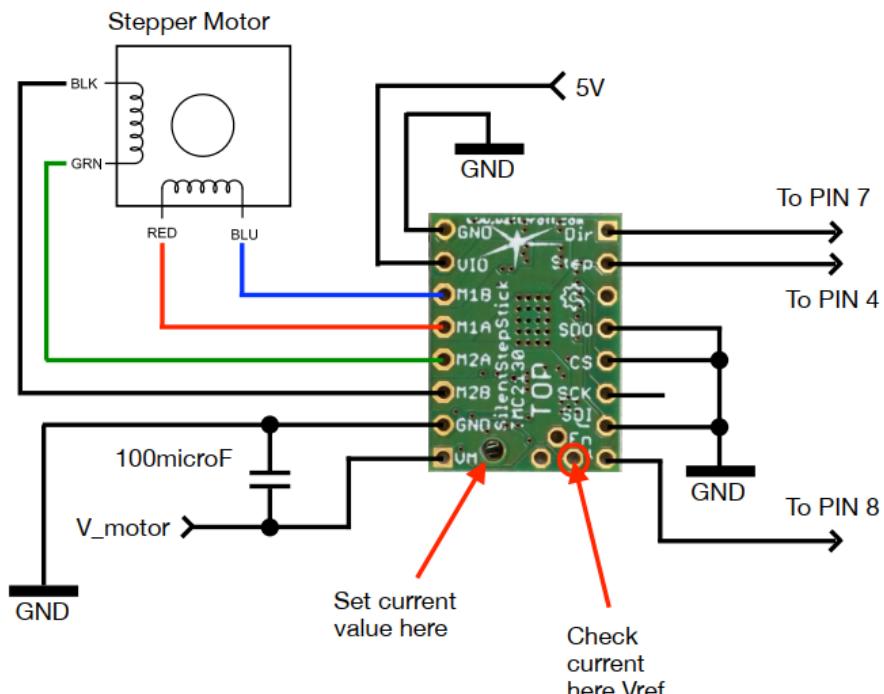


Hình 3.38: Hình ảnh thực tế connector D-sub 25

Để đảm bảo rằng động cơ bước di chuyển đúng và theo đúng hướng, việc kết nối dây của động cơ bước là rất quan trọng. Cụ thể, việc xác định cuộn dây nào được nối với cuộn dây nào và xác định cực tính của cuộn dây đó đều đóng một vai trò quan trọng. Mặc dù không có tiêu chuẩn chung về mã màu cho dây của động cơ bước, nhiều nhà sản xuất vẫn duy trì sự thống nhất bằng cách giữ cho dây bước của họ tương ứng với các cặp cuộn dây trong động cơ. Điều này là quan trọng đặc biệt khi làm việc với động cơ bước và được thường được biết đến với thuật ngữ "mã màu động cơ bước" để chỉ dẫn về cách kết nối dây điện một cách chính xác [25].

Như hình minh họa, mỗi trục tự do của hệ thống sẽ có tổng cộng 4 đầu dây được đặt tên là BLK, GRN, RED và BLUE, tương ứng với các màu đen, xanh lá cây, đỏ và xanh. Cặp dây BLK và GRN cũng như cặp dây RED và BLUE mô tả hai cuộn dây động cơ hoạt động đôi một trong từng Stepper Motor, tổng cộng là 6 động cơ bước trên hình vẽ. Các đầu dây này, tức là BLK, GRN, RED, và BLUE, sẽ được kết nối vào Connector D-sub 25 một cách theo cặp để điều khiển từng Stepper Motor. Bởi vì chỉ có 6 Stepper Motor, mỗi cái điều khiển một trục tự do cụ thể, nên tổng số dây là 24. Do đó, theo sơ đồ kết nối, đầu dây thứ 13 của Connector sẽ được loại bỏ để đảm bảo rằng hệ thống hoạt động chính xác và hiệu quả.

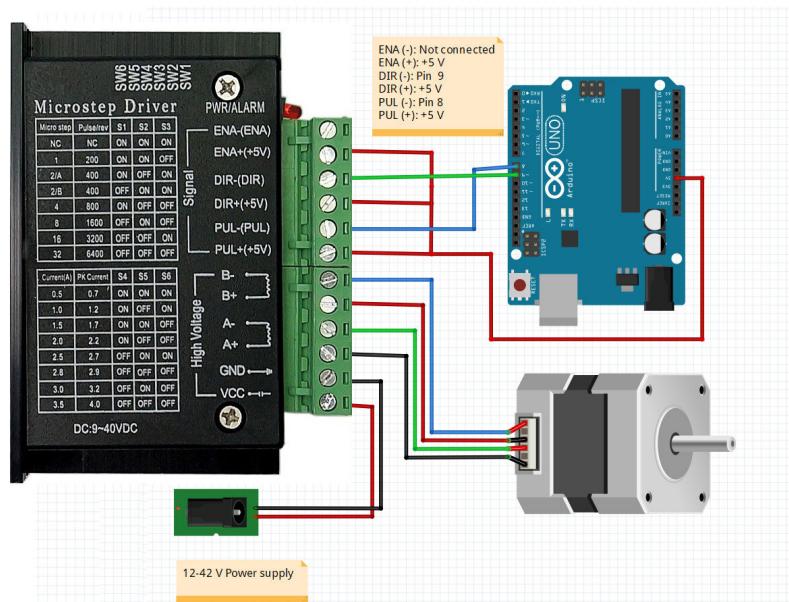
Đối với các trục J4, J5, J6 sử dụng các motor Nema nhỏ, chúng ta sử dụng driver nhỏ gọn như TMC2130 hoặc a4988 để điều khiển. Sơ đồ kết nối như sau:



Hình 3.39: Sơ đồ kết nối driver và động cơ của J4, J5 và J6

- J4, J5 và J6 kết nối trình điều khiển và động cơ với các chân khác nhau của Arduino Mega:
  - J6 sử dụng các chân Dir, Step lần lượt là PIN7 và PIN4
  - J5 sử dụng các chân Dir, Step lần lượt là PIN5 và PIN2
  - J4 sử dụng các chân Dir, Step lần lượt là PIN6 và PIN3
- Dòng điện nên đặt ở mức thấp hơn 0,67A, riêng với J6 thì thấp hơn 0.6A

Các trục J1, J2, J3 sử dụng động cơ Nema lớn, do đó cần phải sử dụng driver TB6600 để có thể đáp ứng. Sơ đồ kết nối minh họa như sau:



Hình 3.40: Sơ đồ kết nối driver và động cơ của J1, J2 và J3

- J1, J2 và J2 kết nối trình điều khiển và động cơ với các chân khác nhau của Arduino Mega
  - J3 sử dụng các chân Step/Dir lần lượt là PIN51 và PIN49
  - Dòng điện của J3 nên đặt ở mức thấp hơn 0,85A
  - J2 sử dụng các chân Step/Dir lần lượt là PIN45 và PIN43
  - J1 sử dụng các chân Step/Dir lần lượt là PIN39 và PIN37
  - Dòng điện của J1 và J2 nên đặt ở mức thấp hơn 2.8A
  - Các chân ENA-, DIR- và PUL- được nối đất chung với nhau

## 3.2 Đề xuất chức năng

Trong suốt quá trình tìm hiểu và nghiên cứu trong quá trình làm Đồ án môn học Kỹ thuật Máy tính, nếu chỉ dừng lại ở việc hiện thực driver cho cánh tay robot thì chúng ta không thể kiểm chứng được cánh tay có hoạt động đúng không, có linh hoạt trong chuyển động không. Vì thế để có thể chứng minh được tính đúng đắn và hoàn thiện trong việc lập trình điều khiển driver cho robot JetArm của nhóm, nhóm sẽ đưa ra một số đề xuất dựa trên những đặc điểm về robot như sau:

- Robot JetArm có khả năng kết hợp với các thiết bị ngoại vi khác để thực hiện một số hành động nhất định, có thể chuyển động linh hoạt trong không gian.
- Lắp thêm camera cho Robot Arm để hiện thực một số ứng dụng liên quan đến Trí tuệ nhân tạo.
- Bộ não của Robot là Jetson Nano - một vi xử lý mạnh mẽ có khả năng thực hiện các ứng dụng tính toán phức tạp cũng như là Trí tuệ nhân tạo.
- Jetson Nano cũng hỗ trợ khá nhiều chuẩn giao tiếp, giúp kết nối được nhiều thiết bị ngoại vi để tạo ra những tính năng mới.
- Với số trục tự do lên đến 6 bậc, robot JetArm có thể thực hiện các góc quay phức tạp và linh hoạt.

Dựa trên những đặc điểm này, nhóm đề xuất hiện thực hai chức năng chính như sau:

**Hiện thực chức năng sử dụng Gamepad để điều khiển cử động của robot JetArm xoay quanh các trục tự do và phối hợp xoay giữa các trục với nhau**

**Hiện thực ứng dụng sử dụng Trí tuệ nhân tạo và Thị giác máy tính vào robot JetArm để nhận diện và xác định vị trí các quân cờ trong cờ tướng, sau đó cánh tay sẽ tự chơi cờ dựa trên thuật toán**

## 3.3 Xây dựng kiến trúc hệ thống

### 3.3.1 Yêu cầu hệ thống

Đối với một hệ thống cánh tay Robot ứng dụng ROS 6DOF (Degrees of Freedom), chúng ta cần thiết phải xem xét và đáp ứng các yêu cầu sau đây:

#### 3.3.1.1 Yêu cầu về tính năng

##### 1. Chuyển động của cánh tay robot:

- Phải điều khiển chính xác các cấu trúc khớp và động cơ của cánh tay robot để đạt các bước, các vị trí và hướng di chuyển mong muốn.
- Cánh tay robot cần có khả năng di chuyển và xoay trong không gian 3 chiều để đạt được nhiều vị trí và góc mong muốn, từ đó có thể thực hiện các tác vụ khác nhau.
- Các trục tự do phải xoay độc lập và cùng lúc với nhau.

##### 2. Điều khiển bằng gamepad:

- Sử dụng Gamepad điều khiển cánh tay robot di chuyển được theo các trục tự do riêng biệt và cùng lúc
- Cánh tay Robot phải di chuyển mượt mà liên tục khi nhấn giữ các nút trên Gamepad
- Các nút trên Gamepad phải tương ứng với từng trục tự do của robot JetArm, một số nút có chức năng gohome và gofold (trạng thái trước khi khởi động của robot)
- Khi về trạng thái gofold, nếu người dùng nhấn bất kỳ nút nào khác trên gamepad không thể hủy chức năng gofold và đợi đến khi robot về vị trí gofold thì chỉ cho phép nhấn nút gohome để về vị trí home

##### 3. Xử lý ảnh và thị giác máy tính:

- Model học sâu phải có khả năng phát hiện và nhận diện chính xác quân cờ có trong tập dữ liệu
- Camera cần được hiệu chỉnh (calibration) với độ chính xác cao để có thể tìm được tọa độ thực của quân cờ trên bàn cờ để cánh tay có thể đến được chính xác vị trí đó

- Các vị trí trên bàn cờ phải được xác định chính xác dựa vào kỹ thuật thị giác máy tính

#### 4. Giải thuật đánh cờ:

- Giải thuật đánh cờ phải tính được các nước cờ dựa vào các quân cờ có trên bàn cờ
- Giải thuật đánh cờ phải chọn ra nước cờ tốt nhất và giúp cho robot đi đúng vị trí đó

#### 5. Đầu hút vật thể:

- Đầu hút này phải được điều khiển hút và thả đúng lúc các quân cờ để đưa được các quân cờ đến vị trí mong muốn

#### 6. Arduino Mega:

- Được lập trình và xây dựng cấu trúc logic hợp lý dựa trên ngôn ngữ cấp cao là C++
- Giao tiếp và tiếp nhận đầy đủ các mã lệnh được gửi từ máy tính nhúng
- Gửi chính xác các mã lệnh giao tiếp sau khi tiếp nhận lệnh từ máy tính nhúng
- Phân tích và tính toán các thông số cần thiết để gửi đến bộ điều khiển động cơ (device driver)
- Tính toán các thông số góc quay để giới hạn góc quay của cánh tay do phần cứng
- Điều khiển chính xác máy bơm dựa vào lệnh từ máy tính nhúng

#### 7. Jetson Nano:

- Giao tiếp với arduino mega bằng giao thức UART và gửi đúng các cú pháp giao tiếp
- Giao tiếp với camera để thu thập hình ảnh
- Sử dụng những model học sâu để phát hiện và nhận dạng các quân cờ trong tập dữ liệu đã huấn luyện
- Chạy và tính toán được giải thuật đánh cờ
- Tính toán được chính xác vị trí bàn cờ và quân cờ dựa vào kỹ thuật về xử lý ảnh
- Tích hợp được ROS để phân luồng xử lý các chức năng riêng biệt
- Các ứng dụng được lập trình bằng các ngôn ngữ cấp cao

### 3.3.1.2 Yêu cầu phi tính năng

#### 1. Độ chính xác:

- Robot có thể tự điều khiển cánh tay đến vị trí quân cờ và nhắc lên với sai số không quá 3mm
- Các quân cờ phải được nhận diện chính xác với độ tự tin không dưới 50

2. **Tốc độ:** Tốc độ phản hồi giữa Jetson Nano và Arduino Mega phải đủ nhanh (dưới 1s) và chính xác.

3. **Độ tin cậy:** Hệ thống cần đảm bảo độ tin cậy cao, có khả năng di chuyển các khớp chính xác theo mong muốn, tránh sự cố và đảm bảo an toàn trong quá trình hoạt động.

4. **Hiệu suất:** Hệ thống cần đáp ứng yêu cầu về hiệu suất, độ trễ điều khiển thấp, cánh tay di chuyển chính xác để thực hiện các tác vụ một cách nhanh chóng và hiệu quả.

5. **Dễ sử dụng và bảo trì:** Hệ thống cần phải dễ sử dụng và dễ bảo trì. Điều này bao gồm giao diện người dùng thân thiện, tài liệu hướng dẫn, và khả năng xác định và khắc phục sự cố một cách dễ dàng.

6. **Khả năng mở rộng:** Hệ thống cần có khả năng mở rộng để có thể nâng cấp và mở rộng chức năng trong tương lai nếu cần thiết.

### 3.3.2 Kiến trúc hệ thống

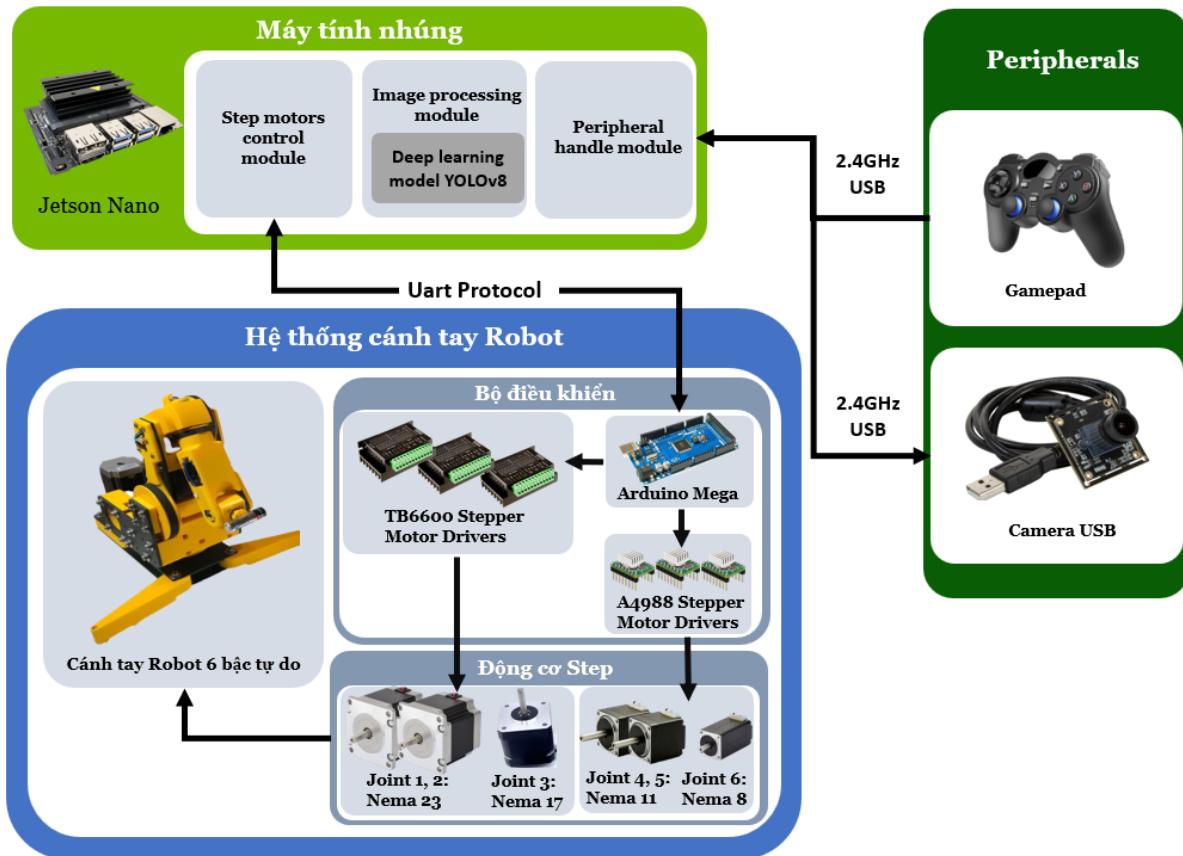
Kiến trúc hệ thống là tổng quan về cách mà các thành phần riêng lẻ của một hệ thống tương tác với nhau và hoạt động cùng nhau để thực hiện các tác vụ và chức năng. Xây dựng một kiến trúc hệ thống phù hợp sẽ mang lại các lợi ích cho hệ thống như:

1. **Hiệu suất cao:** Giúp định rõ cách các thành phần của hệ thống tương tác và làm việc cùng nhau. Bằng cách tối ưu hóa việc trao đổi dữ liệu, tài nguyên và thông tin giữa các thành phần, kiến trúc hệ thống giúp đạt được hiệu suất cao hơn và tăng khả năng xử lý của hệ thống.
2. **Dễ dàng mở rộng:** Khi cần thêm các thành phần mới hoặc mở rộng quy mô của hệ thống, kiến trúc hệ thống giúp đảm bảo rằng sự thay đổi này có thể được thực hiện một cách linh hoạt và không gây ảnh hưởng đến toàn bộ hệ thống.
3. **Sự linh hoạt và tái sử dụng:** Bằng cách phân chia hệ thống thành các thành phần độc lập, mỗi thành phần có thể được phát triển và duy trì riêng biệt. Điều này giúp giảm thiểu sự phụ thuộc giữa các thành phần và tạo điều kiện thuận lợi cho việc sử dụng lại các thành phần trong các dự án khác nhau.
4. **Dễ dàng quản lý và bảo trì:** Điều này giúp đơn giản hóa việc quản lý và bảo trì hệ thống, cũng như giảm thiểu sự cố và hạn chế tác động của lỗi từ một thành phần đến các thành phần khác.
5. **Khả năng tích hợp và tương thích:** Tạo điều kiện thuận lợi cho việc tương thích và giao tiếp giữa các thành phần, đồng thời giảm thiểu sự xung đột và xung đột giữa các công nghệ và quy trình khác nhau.

Có thể thấy, kiến trúc hệ thống là một yếu tố cần thiết để xây dựng và quản lý các hệ thống máy tính hiệu quả. Một kiến trúc tốt sẽ giúp tối ưu hóa hiệu suất, mở rộng quy mô, tăng tính linh hoạt và khả năng tái sử dụng, dễ dàng quản lý và bảo trì, cũng như tạo điều kiện thuận lợi cho tích hợp và tương thích giữa các thành phần và hệ thống khác nhau.

Từ các yêu cầu chức năng và phi chức năng cần phải thỏa mãn. Nhóm thiết kế hệ thống điều khiển cho cánh tay Robot là một hệ thống kết hợp giữa máy tính và các yếu tố cơ khí, máy tính sẽ điều khiển cơ khí của cánh tay thông qua Driver.

Kiến trúc hệ thống như sau:



Hình 3.41: Sơ đồ kiến trúc hệ thống

Trong đó:

- Trung tâm xử lý bộ não của hệ thống được đảm nhận bởi Nvidia Jetson Nano. Tại đây, các module chức năng sẽ được chia nhỏ thành các node ROS và được public cùng với subscribe. Cam Processing module chịu trách nhiệm nhận ảnh liên tục từ camera, sau đó tiến hành xử lý ảnh và gửi cho Moving module. Moving module khi nhận tín hiệu từ Cam Processing module sẽ tiến hành sử dụng model AI để nhận diện vật thể và đưa ra tọa độ của vật thể đó, sau đó sẽ điều khiển cánh tay robot đến chính xác vị trí của vật thể được phát hiện bởi model AI.
- Dữ liệu từ Camera truyền về dựa trên kết nối USB sẽ là đầu vào cho model AI để nhận diện vật thể.
- Để điều khiển cánh tay robot, Jetson Nano sẽ gửi tín hiệu điều khiển xuống Arduino Mega theo giao thức UART. Tại Arduino mega, cánh tay Robot sẽ được điều khiển theo máy trạng thái. 3 khớp lớn 1, 2, 3 sẽ được điều khiển sử dụng motor driver TB6600, và 3 khớp nhỏ sẽ được điều khiển thông qua driver A4988.

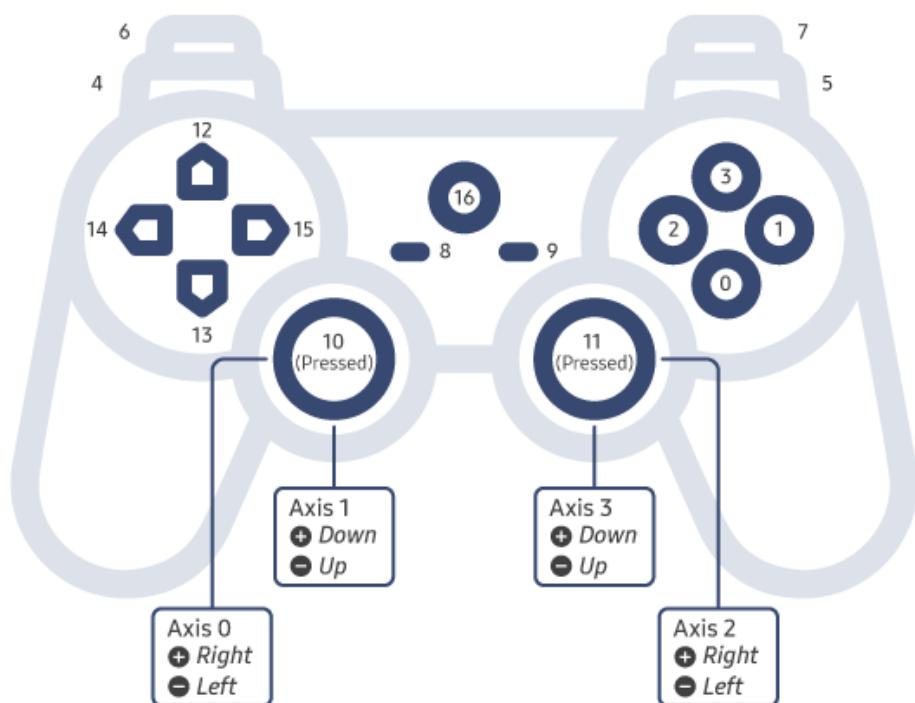
## 3.4 Hiện thực hệ thống

Từ kiến trúc hệ thống trên, cánh tay của nhóm sẽ có hai chế độ. Ở chế độ tự động cánh tay sẽ tự nhận diện và xác định vật thể thông camera và model xử lý ảnh. Vị trí thực tế của vật thể sẽ được tính toán và sau đó cánh tay sẽ đến vị trí của vật thể. Ngược lại đối với chế độ điều khiển bằng tay thì cánh tay sẽ được điều khiển bằng Gamepad.

### 3.4.1 Chế độ Manual

#### 3.4.1.1 Cơ sở lý thuyết và hiện thực

Sơ đồ của Gamepad sẽ bao gồm các nút như sau:



Hình 3.42: Sơ đồ các nút của game pad

Ở chế độ này, Gamepad sẽ được điều khiển chính bởi Jetson Nano, thông qua kết nối USB 2.4GHz, trong khi Jetson Nano sẽ giao tiếp với arduino mega - bộ phận điều khiển của driver thông qua UART. Để điều khiển cánh tay, ta cần phải điều khiển 6 bậc tự do sử dụng các nút điều khiển của Gamepad như hình trên. Nhóm sẽ quy định các nút, axis trên gamepad điều khiển các Joint khác nhau và một số chức năng khác như bảng dưới.

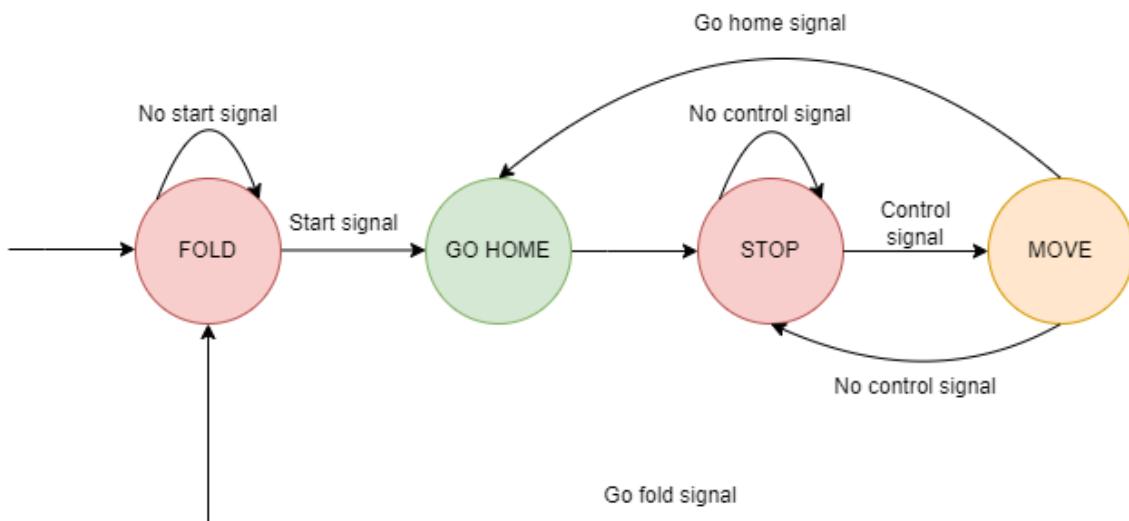
Ý tưởng cơ bản để điều khiển Robot dựa vào Gamepad nằm ở chỗ các Joint của robot chỉ có hai cách chuyển động là ngược chiều kim đồng hồ và cùng chiều kim đồng hồ. Dựa trên ý tưởng trên nhóm sẽ tạo ra một chuỗi điều khiển gồm 6 số theo cú pháp `#xxxxxx!`. Ở đây x sẽ có 3 giá trị là 0, 1 và 2. Ý nghĩa của ứng với từng con số lần lượt là lệnh Stop còn 1 và 2 sẽ là lệnh Move với 1 là di chuyển cùng chiều kim - đi xuống và 2 là di chuyển ngược chiều kim - đi lên.

Axis 0	Joint 1
Axis 1	Joint 2
Axis 2	Joint 3
Axis 3	Joint 4
Button 0 and 3	Joint 5
Button 1 and 2	Joint 6
Button 4	Go home (only in manual)
Button 6	Go fold (robot sẽ gấp lại như trạng thái lúc chưa khởi động)

Bảng 3.1: Các nút điều khiển Gamepad

Với số lượng các nút ở trên, một điều đặt biệt mà nhóm đã hiện thực là người dùng có thể nhấn giữ hoặc nhấn đồng thời nhiều nút với nhau. Khi nhấn giữ một nút, hoặc axis thì Joint được nút hay axis đó điều khiển sẽ di chuyển liên tục về một phía. Với các axis 0, 1, 2, 3 khi nhấn giữ bên phải thì Joint sẽ di chuyển về bên phải và tương tự với bên trái. Đối với Joint 5 và 6, khi nhấn nút 0, 1 thì hai trực này sẽ di chuyển theo chiều kim đồng hồ và ngược lại với 2 nút còn lại.

### 3.4.1.2 Máy trạng thái



Hình 3.43: Manual State Machine

Trong chế độ Manual, mặc định khi khởi động, robot sẽ ở trạng thái Fold (gấp). Chỉ khi nhận được tín hiệu hoạt động từ Jetson Nano, robot mới tiến hành về vị trí Home - vị trí tọa độ gốc và bắt đầu phiên làm việc của mình:

- Mỗi khi nhận tín hiệu điều khiển hợp lệ, robot sẽ chuyển sang trạng thái MOVE và di chuyển cho đến khi nhận được tín hiệu dừng hoặc không có tín hiệu nào hợp lệ được gửi đến. Vì vậy, chỉ cần giữ nút điều khiển Gamepad thì cánh tay sẽ di chuyển đến khi người dùng thả nút.
- Khi nhận tín hiệu về Home, robot trở về tọa độ gốc và đợi tín hiệu điều khiển tiếp theo.
- Khi nhận tín hiệu Fold, robot sẽ trở về vị trí gấp - vị trí ban đầu khi chưa có tín hiệu khởi động từ Jetson và chờ cho đến khi có tín hiệu khởi động để bắt đầu lại phiên làm việc.

### 3.4.2 Chế độ Auto

Trong chế độ Auto, Robot sẽ tự động thực hiện hành động đối với vật thể dựa trên những đặc điểm được nhận diện từ camera. Camera sẽ kết nối với Jetson Nano, bộ não của Robot thông qua kết nối USB 2.4GHz, nhờ đó, hình ảnh sẽ được gửi về Jetson Nano. Arduino Mega vẫn tiếp tục là xương sống của Robot điều khiển toàn bộ hoạt động của robot.



Hình 3.44: Camera USB

Tuy nhiên, đối với trạng thái Auto này nguyên tắc của chuỗi điều khiển sẽ khác so với chế độ Manual và có dạng `!x:y:z:a:b:c#`. Với x, y, z lần lượt là tọa độ theo trục x, trục y và trục z của cánh tay. Các thông số còn lại a, b, c đại diện cho góc quay của các trục cánh tay theo biến đổi euler.

Về cơ bản, khi nói ý tưởng để tạo ra cú pháp như trên thì ta có thể nhận thấy rằng: Dựa trên những kiến thức ở **Phần 2: Kiến thức nền tảng**, khi chúng ta áp dụng những kiến thức về xử lý ảnh, và hiệu chỉnh camera cùng với đó là mô hình nhận diện vật thể thì hoàn toàn có thể tìm được tọa độ thực của vật đó. Một cách chi tiết hơn, khi camera thu lại hình ảnh về vật thể và gửi đến Jetson Nano, ở ngay đây, các mô hình học sâu và trí tuệ nhân tạo sẽ được sử dụng để nhận diện chính xác đâu là vật thể cần tìm. Sau đó, với kỹ thuật xử lý ảnh và hiệu chỉnh camera, tọa độ thực của vật thể sẽ được tìm ra và khi đó, robot có thể di chuyển đến đó mà thực thi hành động.

Dựa trên cơ sở lý thuyết như trên nhóm sẽ tiến hành lần lượt các bước với vật thể cần nhận diện sẽ là những quân cờ tướng.

### 3.4.3 Huấn luyện mô hình nhận diện cờ tướng

#### 3.4.3.1 Lựa chọn mô hình huấn luyện

Hiện nay có khá nhiều mô hình huấn luyện nhận diện được sử dụng như:

- **YOLO (You Only Look Once)**: chỉ sử dụng một mạng Neural Network duy nhất để dự đoán trực tiếp Bounding Box từ toàn bộ bức ảnh bằng một lần đánh giá duy nhất, nổi bật với tốc độ và độ chính xác cao, dễ sử dụng cho các tác vụ phát hiện và theo dõi đối tượng, phân đoạn đối tượng, phân loại hình ảnh và ước lượng tư thế
- **Faster R-CNN (Faster Region-based Convolutional Neural Network)**: mô hình này sử dụng một mạng nơ-ron tích chập (CNN) để tạo ra các vùng đề xuất đối tượng, sau đó phân loại các vùng này và xác định vị trí chính xác của đối tượng trong ảnh. Faster R-CNN được biết đến với độ chính xác cao nhưng yêu cầu thời gian tính toán lớn
- **RetinaNet**: RetinaNet kết hợp hai mạng là ResNet và Feature Pyramid Network (FPN) để phát hiện đối tượng. Mô hình này nổi bật với khả năng xử lý tốt các đối tượng nhỏ và đạt hiệu suất cao trong các tác vụ phát hiện đối tượng nhờ vào việc sử dụng hàm focal loss
- **SSD (Single Shot MultiBox Detector)**: SSD là một mô hình phát hiện đối tượng nhanh và hiệu quả, sử dụng một mạng CNN duy nhất để dự đoán bounding box và phân loại đối tượng trực tiếp từ các đặc trưng của ảnh. SSD có tốc độ nhanh hơn so với Faster R-CNN nhưng có thể kém chính xác hơn
- **Mask R-CNN**: Đây là sự mở rộng của Faster R-CNN, thêm một nhánh để phân đoạn đối tượng. Ngoài việc phát hiện và xác định vị trí đối tượng, Mask R-CNN còn tạo ra mặt nạ phân đoạn cho từng đối tượng, rất hữu ích trong các ứng dụng yêu cầu phân đoạn chi tiết
- **EfficientDet**: EfficientDet là một mô hình phát hiện đối tượng hiệu quả, sử dụng EfficientNet làm mạng nền tảng và kết hợp với một cấu trúc FPN cải tiến. Mô hình này nổi bật với khả năng đạt được sự cân bằng tốt giữa hiệu suất và hiệu quả tính toán
- **CenterNet**: CenterNet sử dụng một cách tiếp cận mới, xác định các đối tượng bằng cách phát hiện trung tâm của chúng và xác định kích thước cũng như offset của bounding box. Mô hình này cung cấp sự cân bằng tốt giữa tốc độ và độ chính xác

Trong các mô hình trên, YOLO dường như nổi bật hơn bởi những đặc điểm sau:

- **Tốc độ cao:** YOLO (You Only Look Once) nổi bật với tốc độ xử lý nhanh vượt trội so với nhiều mô hình nhận diện vật thể khác. YOLO thực hiện việc phát hiện đối tượng trong một lần chạy duy nhất qua mạng nơ-ron, giúp giảm thiểu thời gian xử lý và tăng hiệu suất
- **Độ chính xác cao:** Các phiên bản mới nhất như YOLOv5 và YOLOv8 đã cải thiện đáng kể độ chính xác trong việc nhận diện và phân loại đối tượng. Những cải tiến trong kiến trúc mạng và thuật toán học giúp YOLO nhận diện được nhiều loại đối tượng khác nhau với độ chính xác cao
- **Dễ sử dụng và tích hợp:** YOLO được thiết kế để dễ sử dụng, với hỗ trợ đầy đủ cho các ngôn ngữ lập trình phổ biến như Python. Gói YOLO-Python và giao diện dòng lệnh (CLI) giúp việc triển khai và thử nghiệm mô hình trở nên dễ dàng và nhanh chóng
- **Tính linh hoạt:** YOLO hỗ trợ nhiều nhiệm vụ thị giác máy tính khác nhau, bao gồm phát hiện đối tượng, theo dõi đối tượng, phân đoạn đối tượng, phân loại hình ảnh và ước lượng tư thế. Sự linh hoạt này cho phép sử dụng YOLO trong nhiều ứng dụng và lĩnh vực khác nhau, từ an ninh, giám sát đến xe tự lái và robot học

Dựa trên những đặc điểm nổi bật trên nhóm quyết định chọn YOLO làm mô hình huấn luyện chính. Trong suốt quá trình phát triển, tính đến thời điểm hiện tại, có 8 loại mô hình bao gồm:

- **YOLOv1:** đặc điểm nổi bật nhất của YOLOv1 là khả năng thực hiện nhận diện và định vị các đối tượng trong ảnh chỉ trong một lần chạy qua mạng nơ-ron tích chập, thay vì sử dụng các phương pháp hai bước như các mô hình trước đó. Điều này giúp YOLOv1 đạt được tốc độ xử lý rất nhanh, lý tưởng cho các ứng dụng yêu cầu thời gian thực. YOLOv1 chia ảnh đầu vào thành một lưới SxS và mỗi ô lưới sẽ dự đoán bounding boxes và các xác suất thuộc về các lớp đối tượng. Một trong những điểm đặc biệt của YOLOv1 là việc dự đoán trực tiếp tọa độ bounding box và xác suất lớp, từ đó giảm thiểu sự trùng lặp và tối ưu hóa quá trình phát hiện đối tượng
- **YOLOv2:** là phiên bản nâng cấp của YOLOv1, YOLOv2 cải thiện độ chính xác và tốc độ bằng cách sử dụng các kỹ thuật như Batch Normalization, Anchor Boxes và đa tỷ lệ (multi-scale training). Mô hình này cũng hỗ trợ nhận diện hơn 9000 loại đối tượng khác nhau. Những cải tiến này giúp YOLOv2 trở nên mạnh mẽ và linh hoạt hơn trong việc phát hiện và định vị đối tượng, đáp ứng tốt hơn nhu cầu của các ứng dụng thời gian thực.

- **YOLOv3:** tiếp tục cải tiến từ YOLOv2 bằng cách sử dụng một kiến trúc mạng sâu hơn và mạnh mẽ hơn, Darknet-53. Nó sử dụng các liên kết dư thừa (residual connections) và cải thiện khả năng phát hiện các đối tượng nhỏ nhờ vào việc dự đoán ở ba quy mô khác nhau. YOLOv3 cũng tăng cường độ chính xác mà không làm giảm tốc độ, giữ vững danh tiếng là một trong những mô hình phát hiện đối tượng hàng đầu
- **YOLOv4:** mang đến nhiều cải tiến đáng kể trong cả độ chính xác và tốc độ. Nó tích hợp các kỹ thuật tiên tiến như Cross Stage Partial connections (CSP), Mish activation, và Mosaic data augmentation. YOLOv4 tối ưu hóa quá trình huấn luyện và suy luận, giúp cải thiện hiệu suất và độ chính xác, trở thành lựa chọn hàng đầu cho nhiều ứng dụng thị giác máy tính thời gian thực
- **YOLOv5:** nổi bật với sự đơn giản và hiệu quả trong việc triển khai. Nó được viết hoàn toàn bằng PyTorch, dễ dàng tích hợp và mở rộng. YOLOv5 cung cấp các mô hình kích thước từ nhỏ đến lớn (small, medium, large, extra-large), cho phép người dùng lựa chọn mô hình phù hợp với nhu cầu cụ thể. Hiệu suất cao và tính tiện lợi của YOLOv5 đã giúp nó nhanh chóng trở thành một trong những phiên bản phổ biến nhất
- **YOLOv6:** tập trung vào việc cải thiện hiệu suất trên các thiết bị nhúng và di động. Nó tối ưu hóa kiến trúc mạng để giảm thiểu độ trễ và tiêu thụ tài nguyên, đồng thời duy trì độ chính xác cao trong việc phát hiện đối tượng. YOLOv6 đặc biệt hữu ích cho các ứng dụng yêu cầu sự di động và khả năng tính toán thấp
- **YOLOv7:** tiếp tục cải thiện độ chính xác và tốc độ bằng cách tích hợp nhiều kỹ thuật mới như các module phân đoạn (segmentation modules) và các lớp dự đoán đa tỷ lệ. Nó tăng cường khả năng phát hiện và phân loại đối tượng trong các tình huống phức tạp, làm cho YOLOv7 trở nên mạnh mẽ và linh hoạt hơn trong các ứng dụng thị giác máy tính hiện đại
- **YOLOv8:** tiếp tục xây dựng trên thành công của các phiên bản trước. Nó giới thiệu các tính năng và cải tiến mới để nâng cao hiệu suất, tính linh hoạt và hiệu quả. YOLOv8 hỗ trợ đầy đủ các nhiệm vụ trí tuệ nhân tạo thị giác như phát hiện đối tượng, phân đoạn, ước lượng tư thế, theo dõi và phân loại hình ảnh. Với tốc độ nhanh, độ chính xác cao và dễ sử dụng, YOLOv8 là lựa chọn lý tưởng cho nhiều ứng dụng và lĩnh vực khác nhau
- **YOLOv9:** nổi bật với cách tiếp cận sáng tạo để vượt qua những thách thức mất thông tin vốn có trong các mạng nơ-ron sâu. Bằng cách tích hợp PGI và kiến trúc GELAN linh hoạt, YOLOv9 không chỉ nâng cao năng lực học tập của mô

hình mà còn đảm bảo lưu giữ thông tin quan trọng trong suốt quá trình phát hiện, do đó đạt được độ chính xác và hiệu suất vượt trội [26]

Trong suốt quá trình tìm hiểu nhóm quyết định chọn mô hình YOLOv8 là bởi: Tuy rằng đây không phải là phiên bản mới nhất của YOLO nhưng khi nói về YOLOv9 thì hiệu suất và khả năng vẫn chưa có nhiều những kiểm chứng cụ thể. Trái lại YOLOv8 đã thể hiện nhiều nổi bật đáng kể trong lĩnh vực nhận diện khi so với các phiên bản trước đó.

Cụ thể về YOLOv8, mô hình này sẽ được phân loại thành 5 mô hình chính:

- **YOLOv8n:** là phiên bản nhỏ nhất trong tất cả các mô hình, có ít tầng mạng nơ-ron và ít tham số hơn các phiên bản khác, tuy nhiên lại có tốc độ dự đoán nhanh nhất và phù hợp cho thiết bị có ít tài nguyên
- **YOLOv8m:** là phiên bản lớn hơn YOLOv8n, có nhiều tầng mạng nơ-ron và nhiều tham số hơn. Cho ra độ chính xác tốt hơn YOLOv8n. Nhưng tốc độ cũng sẽ giảm đi
- **YOLOv8s:** là phiên bản tương đối cân bằng về tốc độ và độ chính xác khá cao, tốc độ cũng không giảm đi quá nhiều so với YOLOv8m nhưng độ chính xác đã tăng lên rất nhiều là sấp sỉ với YOLOv8l và YOLOv8x
- **YOLOv8l:** là phiên bản cho ra độ chính xác cao, đòi hỏi thời gian dự đoán lâu hơn và tiêu tốn nhiều tài nguyên trên thiết bị hơn, tuy các tham số đã tăng lên gần gấp đôi nhưng độ chính xác vẫn k cải thiện nhiều so với YOLOv8s và tốc độ bị giảm đi khá nhiều
- **YOLOv8x:** là phiên bản cho ra độ chính xác cao nhất trong tất cả 5 mô hình, tiêu tốn nhiều tài nguyên nhất và thời gian dự đoán cũng lâu nhất, được khuyến cáo sử dụng nếu thiết bị có tài nguyên mạnh mẽ

Model	size (pixels)	mAP <sup>val</sup> 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
<a href="#">YOLOv8n</a>	640	37.3	80.4	0.99	3.2	8.7
<a href="#">YOLOv8s</a>	640	44.9	128.4	1.20	11.2	28.6
<a href="#">YOLOv8m</a>	640	50.2	234.7	1.83	25.9	78.9
<a href="#">YOLOv8l</a>	640	52.9	375.2	2.39	43.7	165.2
<a href="#">YOLOv8x</a>	640	53.9	479.1	3.53	68.2	257.8

Hình 3.45: So sánh 5 loại mô hình YOLOv8

Nhìn chung với tài nguyên có sẵn của Jetson Nano không thật sự quá vượt trội để có thể sử dụng nhiều loại model nên nhóm quyết định chọn mô hình YOLOv8n với tốc độ nhận diện dưới 1s. Nhưng bù lại, nhóm phải huấn luyện mô hình với nhiều lượt huấn luyện hơn để có thể bù đắp độ chính xác của mô hình.

### 3.4.3.2 Chuẩn bị tập dữ liệu

Trước khi bắt đầu huấn luyện một mô hình nhận dạng những quân cờ, chúng ta cần có một tập dữ liệu chứa các hình ảnh về các quân cờ đó nhiều nhất có thể.

Các tập dữ liệu của quân cờ tướng sẽ được nhóm thu thập từ thực tế bằng việc sử dụng camera của robot để chụp lại hình ảnh của các quân cờ. Để nói về chiến lược xây dựng tập dữ liệu thì nhóm đã xây dựng tập dữ liệu dựa trên những nguyên tắc sau:

- Huấn luyện tuần tự từng quân cờ riêng biệt từ nhiều góc độ khác nhau



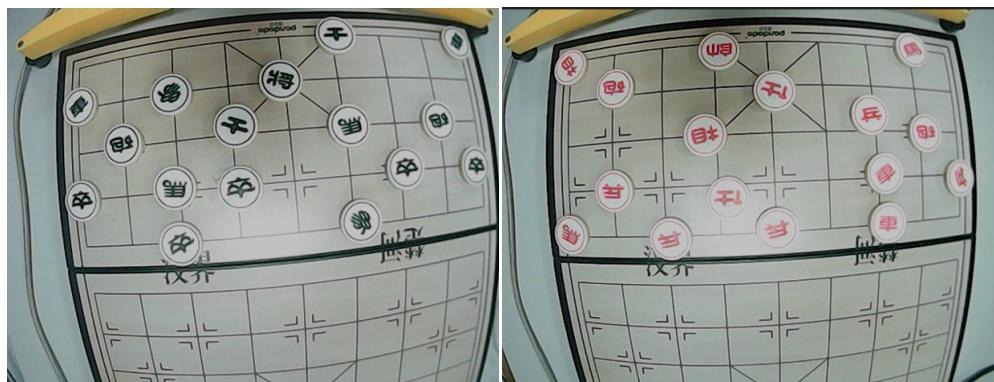
Hình 3.46: Huấn luyện các quân cờ riêng biệt với nhau

- Sắp xếp các quân cờ đúng theo thứ tự trên bàn cờ, phân biệt giữa các quân đỏ và đen



Hình 3.47: Sắp xếp các quân cờ riêng biệt

- Sắp xếp ngẫu nhiên các quân cờ trên toàn bộ bàn cờ



Hình 3.48: Sắp xếp ngẫu nhiên các quân cờ

- Hòa trộn quân đen và đỏ với nhau



Hình 3.49: Hòa trộn quân đen và đỏ với nhau

Tập dữ liệu được nhóm thu thập sẽ được tăng lên bằng cách sử dụng các kỹ thuật xử lý ảnh như:

- Preprocessing: tiền xử lý dữ liệu, giảm thời gian huấn luyện và tăng hiệu suất bằng cách áp dụng các biến đổi hình ảnh cho tất cả các hình ảnh trong tập dữ liệu
  - Auto-Orient: Loại bỏ các phép quay EXIF và chuẩn hóa thứ tự pixel
  - Resize: Giảm kích thước hình ảnh thành 640x640 để có kích thước tệp nhỏ hơn và huấn luyện nhanh hơn
  - Auto-Adjust Contrast: Tăng độ tương phản dựa trên biểu đồ của hình ảnh để cải thiện khả năng chuẩn hóa và phát hiện đường thẳng trong các điều kiện ánh sáng khác nhau
- Augmentation: Tạo các tập huấn luyện mới cho mô hình bằng cách tạo ra các phiên bản tăng cường của mỗi hình ảnh trong tập huấn luyện sẵn có
  - Flip: Thêm các thao tác lật ngang hoặc dọc để giúp model không bị ảnh hưởng bởi hướng của vật thể
  - 90° Rotate: Thêm góc xoay 90 độ để giúp model không bị ảnh hưởng nhiều bởi hướng camera
  - Rotation: Thêm sự biến đổi vào các phép xoay để giúp model trở nên linh hoạt hơn với việc xoay camera
  - Shear: Thêm sự biến đổi vào phôi cảnh để giúp model trở nên linh hoạt hơn với việc thay đổi góc nghiêng và xoay của camera và đối tượng
  - Saturation: Điều chỉnh ngẫu nhiên độ rực rỡ của màu sắc trong các hình ảnh
  - Brightness: Thêm sự biến đổi vào độ sáng của hình ảnh để giúp model linh hoạt hơn với các thay đổi về ánh sáng và cài đặt camera

Tập dữ liệu được làm phong phú sẽ được chia làm 3 tập con là train, valid và test theo tỉ lệ 7:2:1.

- Tập train: Đây là tập dữ liệu được sử dụng để huấn luyện mô hình. Mô hình sẽ học từ các mẫu trong tập dữ liệu này thông qua việc điều chỉnh các tham số của nó. Tập dữ liệu huấn luyện cần phải đại diện cho đa dạng của dữ liệu thực tế nhưng cũng cần đảm bảo rằng mô hình không bị overfitting
- Tập valid: Tập dữ liệu này được sử dụng để đánh giá hiệu suất của mô hình trong quá trình huấn luyện. Sau mỗi lượt huấn luyện, mô hình sẽ được đánh giá trên tập dữ liệu validation để kiểm tra xem liệu nó có đang học tốt hay không,

và để điều chỉnh các siêu tham số (hyperparameters) nếu cần thiết. Tập dữ liệu validation cần phải được lựa chọn sao cho đại diện cho dữ liệu mà mô hình có thể gặp phải trong thực tế, nhưng không được sử dụng trong quá trình huấn luyện để tránh overfitting

- Tập test: Sau khi huấn luyện hoàn tất và mô hình đã được điều chỉnh dựa trên tập dữ liệu validation, tập dữ liệu test được sử dụng để đánh giá hiệu suất cuối cùng của mô hình. Mô hình sẽ không được huấn luyện hoặc điều chỉnh dựa trên tập dữ liệu này. Tập dữ liệu test cần phải được lựa chọn một cách độc lập và đại diện cho dữ liệu mới mà mô hình có thể gặp phải trong thực tế

Sau khi đã thu thập dữ liệu, nhóm sẽ tiến hành đến bước gán nhãn dữ liệu. Với tập dữ liệu là các quân cờ tướng thì nhóm sẽ chia dữ liệu thành 14 lớp bao gồm: '**b\_ma**', '**b\_phao**', '**b\_si**', '**b\_tot**', '**b\_tuong**', '**b\_voi**', '**b\_xe**', '**r\_binh**', '**r\_ma**', '**r\_phao**', '**r\_si**', '**r\_tuong**', '**r\_voi**', '**r\_xe**'. Với r đại diện cho quân cờ đỏ và b đại diện cho quân cờ đen.

Ngoài ba tập dữ liệu ở trên thì file **data.yaml** là file chứa toàn bộ thông tin về tập dữ liệu bao gồm đường dẫn đến các folder train, valid và test; thông tin các lớp đã được gán nhãn.

### 3.4.3.3 Huấn luyện mô hình

Sau khi đã chuẩn bị tập dữ liệu, nhóm đã tiến hành việc huấn luyện mô hình nhận dạng các quân cờ. Một điểm đáng chú ý là đối với một model nhận diện vật thể sử dụng YOLO thì sau khi train xong sẽ trả về một file có đuôi là '**pt**'. Thông thường đối với những máy tính thông thường thì để chạy những file này không quá khó khăn nhưng đối với một board mạch nhúng như Jetson Nano, dù tài nguyên đã được nâng cấp đáng kể, nhưng vẫn gặp khó khăn trong việc chạy liên tục những model như thế. Do đó việc chuyển model dưới định dạng '**pt**' sang dạng TensorRT nên được thực hiện. Nguyên nhân chủ yếu của việc chuyển đổi này là vì TensorRT được dùng nhiều cho inference và sử dụng được hiệu năng vượt trội từ GPU

Khi đó, một vấn đề mới xuất hiện, đối với model dạng '**onnx**' nó chỉ được chạy chỉ trên chính kiến trúc mà nó được tạo ra hay nói cách khác là nó không có tính **cross-platform**. Vì thế để có thể tạo ra file này chúng ta có hai cách là tạo nó ra trên chính Jetson Nano hoặc chúng ta sẽ thực hiện việc build định dạng file trung gian là **onnx** trên chính máy của chúng ta và chuyển nó sang dạng TensorRT trên Jetson Nano. Với cách đầu tiên, thời gian tiêu tốn có thể lên đến vài ngày vì tài nguyên hạn chế, nhưng đối với cách hai, thời gian sẽ được rút ngắn hơn nhiều nếu chúng ta

đủ mạnh. Và nhóm đã chọn cách hai.

Việc tiến hành huấn luyện mô hình sẽ diễn ra thành các bước:

- Tải mô hình nền yolov8n.pt bằng cách tải từ <https://github.com/ultralytics/assets/releases/>
- Chuẩn bị đầy đủ tập dữ liệu
- Chạy các lệnh sau

```
1 model = YOLO('yolov8n.pt')
2 model.train(data = '/path/to/data.yaml', epochs = 200, batch = 16,
              name = export_name, close_mosaic = 0, imgsz = 640)
3
```

- Sau khi model được huấn luyện xong thì sẽ xuất ra file best.pt. Sau đó ta tiến hành chuyển đổi sang định dạng **onnx**

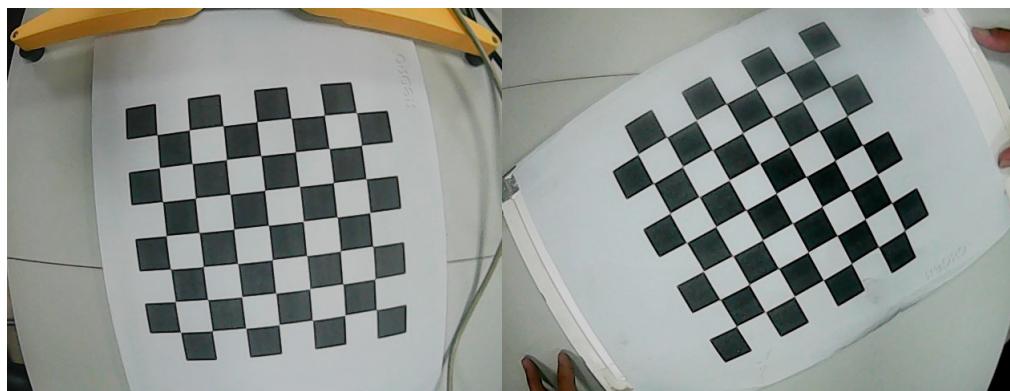
```
1 model = YOLO('/path/to/best.pt')
2 model.export(format = 'onnx', int8 = True)
3
```

- Tiếp đến, chúng ta tiến hành chuyển đổi dạng '**onnx**' sang '**trt**' trên Jetson Nano dưới dạng '**.engine**'
- Và cuối cùng là đưa mô hình vào hệ thống

```
1 model = YOLO('/path/to/best.engine')
2 output = model.track(frame, imgsz=640, conf=0.5, device = 0, save =
                         False)
3
```

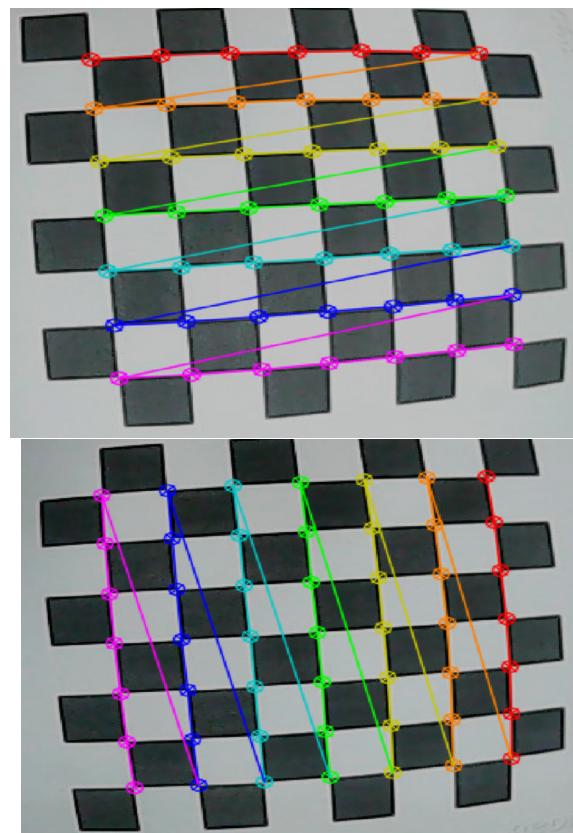
### 3.4.4 Hiệu chỉnh camera

Việc hiệu chỉnh (calibration) camera sẽ được thực hiện dựa trên việc sử dụng một bàn cờ mẫu đặt trước camera, sau đó chụp một số góc cạnh của bàn cờ mẫu sao cho bức ảnh chứa được trọn vẹn cả bàn cờ. Dưới đây là một vài bức ảnh dùng cho việc hiệu chỉnh camera:



Hình 3.50: Một số ảnh dùng cho việc hiệu chỉnh camera

Sau khi đã thu thập một số bức ảnh như trên, bước tiếp theo là tính toán các ma trận và thông số của camera. Nguyên lý cơ bản của việc này là sử dụng thư viện của OpenCV để tính toán và phát hiện góc cạnh của các bức hình trên:



Hình 3.51: Phát hiện góc cạnh bàn cờ mẫu

Nhờ việc phân tích các góc cạnh của các bức ảnh được chụp bởi chính camera được lắp đặt trên robot mà ta có thể tính toán được ma trận camera, hệ số distortion, các vector  $r$ ,  $t$ . Dưới đây là ma trận extrinsic thu được:

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 520.5878920236871 & 0.0 & 308.63922611555165 \\ 0.0 & 519.8497596687193 & 220.3444341942921 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$$

Hệ số distortion là:

$$DistortionCoeff = \begin{bmatrix} -0.43619 & 0.20562 & 0.00019 & -0.00076 & -0.03729 \end{bmatrix}$$

Vector r và t là:

$$r = \begin{bmatrix} 0.01384395861879 \\ 0.05499399196909 \\ 1.49040257466169 \end{bmatrix}$$

$$t = \begin{bmatrix} -1.24503929845461 \\ -27.69694603068828 \\ 225.9966125202619 \end{bmatrix}$$

Những ma trận này sẽ được tích hợp vào quá trình nhận diện vật thể để tính toán ra tọa độ thực của vật thể và từ đó giúp robot có thể di chuyển đến vị trí đó.

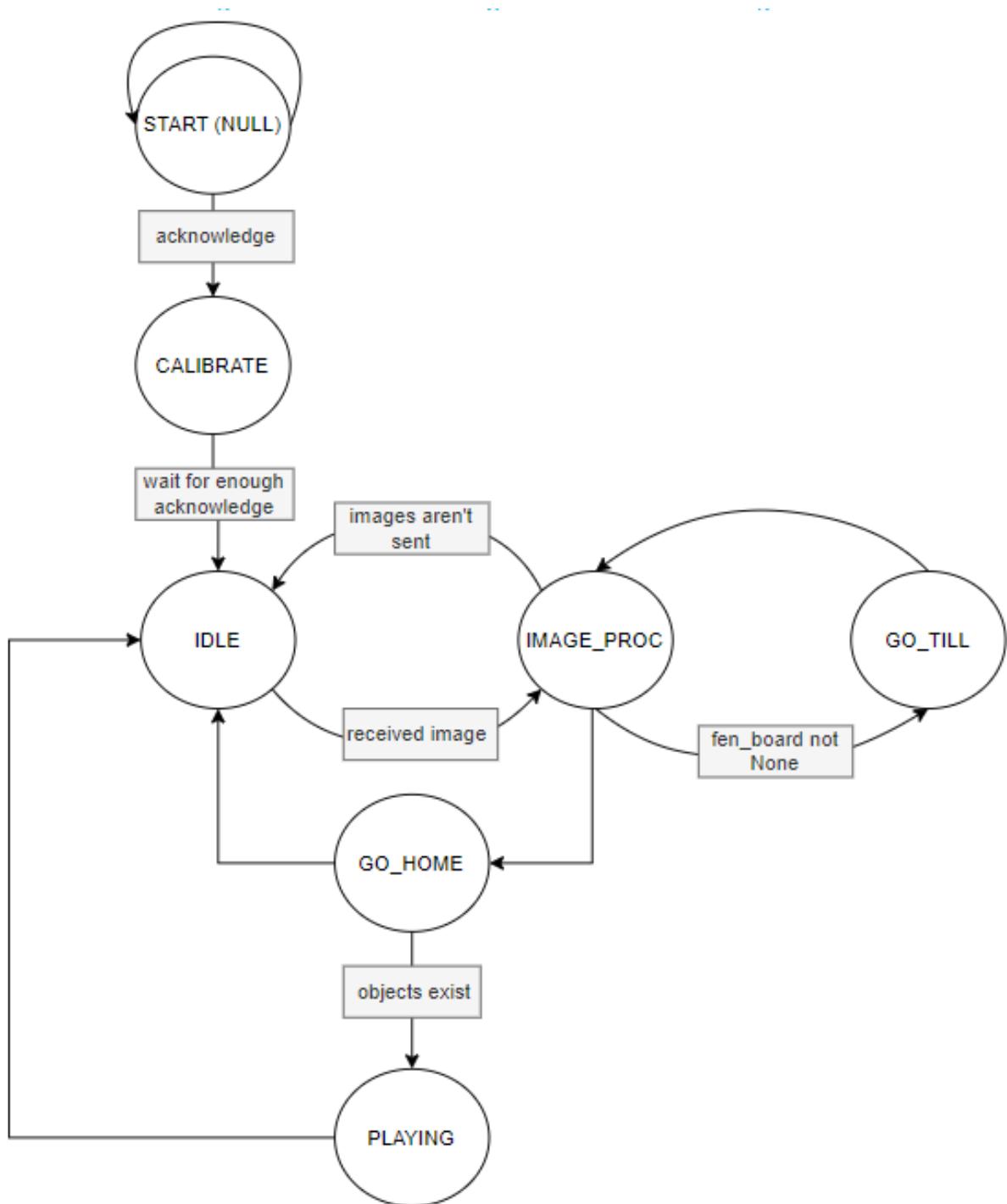
### 3.4.5 Tích hợp ROS

Với chức năng auto mode, hiện tại với việc phân tích chức năng của robot là nhận diện và di chuyển các quân cờ tướng thì nhóm sẽ chia hệ thống thành 2 node "image\_process" và "control\_moving" với 2 chức năng chính của mỗi node là thu nhận ảnh liên tục và node còn lại có nhiệm vụ xử lý ảnh và điều khiển cánh tay.

Với sự hỗ trợ của ROS, việc chia luồng trở nên dễ dàng hơn để thực hiện các tác vụ song song. Với việc nhận ảnh liên tục bởi một node đảm bảo cho Jetson Nano luôn nhận được hình ảnh mới nhất trên bàn cờ và từ đó node còn lại sẽ dùng hình ảnh mới nhất đó để tiến hành xử lý, nhận diện các quân cờ. Thông qua quá trình đó, tọa độ của các quân cờ sẽ được tính toán.

Sau đó, thuật toán đánh cờ sẽ tính toán các quân cờ đang có mặt trên bàn cờ và sắp xếp chúng vào hàng đợi. Từ đó cánh tay robot sẽ đến vị trí của quân cờ đầu tiên trong hàng đợi mà thực hiện nước đi, khi xong quá trình đó, ảnh mới sẽ tiếp tục được gửi từ node "image\_process" sang node "control\_moving" và lại tiếp tục quá trình trên.

### 3.4.6 Máy trạng thái



Hình 3.52: Máy trạng thái của chế độ Auto

Trong chế độ Auto, robot sẽ ở vị trí Home, camera hướng thẳng vuông góc và nhìn xuống bàn cờ, khi đó hệ thống sẽ ở trạng thái Start. Khi nhận được tín hiệu camera đã được bật thì ngay lập tức hệ thống sẽ chuyển qua trạng thái Calibrate. Ở trạng thái này, robot sẽ đợi cho đến khi đủ số lần ack được gửi và sẽ chuyển qua trạng thái

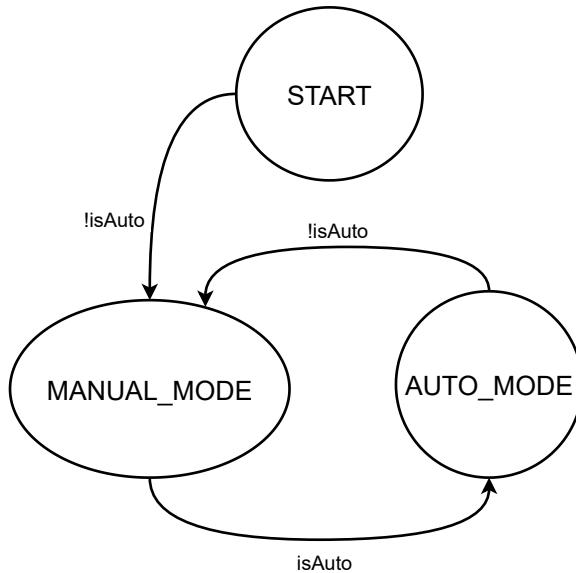
Idle. Đối với trạng thái Idle, khi camera gửi ảnh đến Jetson Nano, hệ thống sẽ chuyển đổi sang Image\_Process tiến hành nhận diện các quân cờ có trong ảnh được gửi và cùng với đó là trả về tọa độ của các quân cờ.

Tiếp đến dựa trên hình ảnh đã nhận diện được nếu trong hình không phát hiện được mép bàn cờ thì cánh tay robot sẽ chuyển qua trạng thái Go\_Till. Ở trạng thái này, cánh tay robot sẽ điều khiển Joint 5 và quay một góc hướng lên trên để bức ảnh chụp được có thể ghi nhận được mép bên kia của bàn cờ. Sau đó, hệ thống sẽ tự động chuyển về trạng thái Image\_Process và tiếp tục nhận diện, tính toán tọa độ của các quân cờ phía còn lại của bàn cờ.

Khi toàn bộ thông tin về bàn cờ đã được ghi nhận hệ thống sẽ chuyển về trạng thái Go\_home và bắt đầu kiểm tra tập nhận diện. Nếu tập nhận diện không có bất cứ quân cờ nào được phát hiện hệ thống sẽ tự động chuyển về trạng thái Idle và tiếp tục nhận ảnh. Ngược lại, nếu tập nhận diện có tồn tại thì robot sẽ bắt đầu chuyển qua trạng thái Playing. Trạng thái này bao gồm nhiều quá trình xảy ra liên tiếp nhau. Đầu tiên, giải thuật đánh cờ tướng sẽ xem xét các quân cờ có trong tập nhận diện và tiến hành tính toán các nước cờ khả thi và chọn nước cờ tốt nhất để tiến hành chơi cờ. Robot sẽ tiến hành di chuyển đến vị trí của quân cờ được chọn để đi, và máy bơm sẽ bật để hút quân cờ lên. Tiếp đến, cánh tay robot sẽ di chuyển về vị trí home, trong suốt quá trình đó, máy bơm luôn được bật. Sau đó, cánh tay sẽ di chuyển lập tức đến vị trí đã được tính toán và máy bơm sẽ được tắt, quân cờ sẽ được thả ra. Sau khi đã đi nước cờ xong thì hệ thống lại trở về trạng thái Idle và tiếp tục vòng lặp như vậy đến khi không có nước cờ nào được tính toán nữa.

## 3.5 Máy trạng thái tổng thể

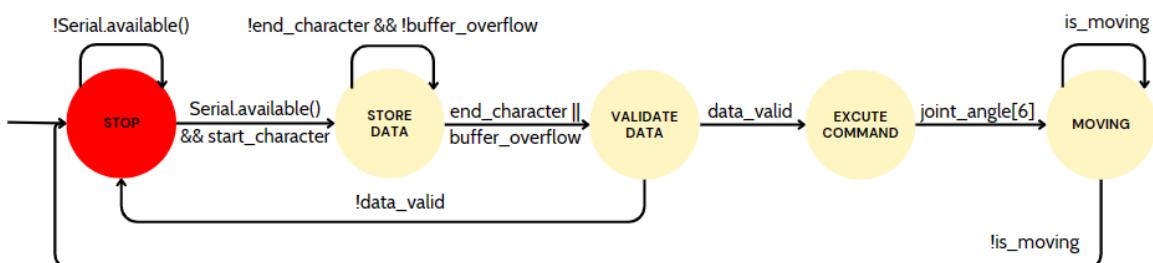
Cuối cùng, ta sẽ có một máy trạng thái nữa để kiểm soát các chế độ Auto và Manual. Khi được khởi động trạng thái đầu tiên sẽ là Manual, khi nhận lệnh isAuto, máy sẽ chuyển sang trạng thái AUTO\_MODE và ngược lại.



Hình 3.53: Máy trạng thái cho việc chuyển đổi mode

Nhằm đảm bảo các lệnh điều khiển chắc chắn nhận được, mỗi lần gửi lệnh xuống Arduino, sẽ có một lệnh phản hồi là "ACK" được gửi từ Arduino lên Jetson. Chỉ khi nhận được lệnh ACK, Jetson mới gửi lệnh tiếp theo. Việc này sẽ giúp cho việc thông tin gửi đi và nhận về được kiểm soát chặt chẽ hơn, tránh thất lạc gây ra lỗi trong khi điều khiển.

Ngoài ra, các câu lệnh điều khiển cũng cần phải được kiểm tra tính đúng đắn trước khi được thư thi. Do đó, một máy trạng thái cho việc xử lý giao tiếp UART trên Arduino là điều cần thiết:



Hình 3.54: Máy trạng thái cho việc xử lý tín hiệu UART

## 3.6 Mã nguồn toàn bộ chương trình

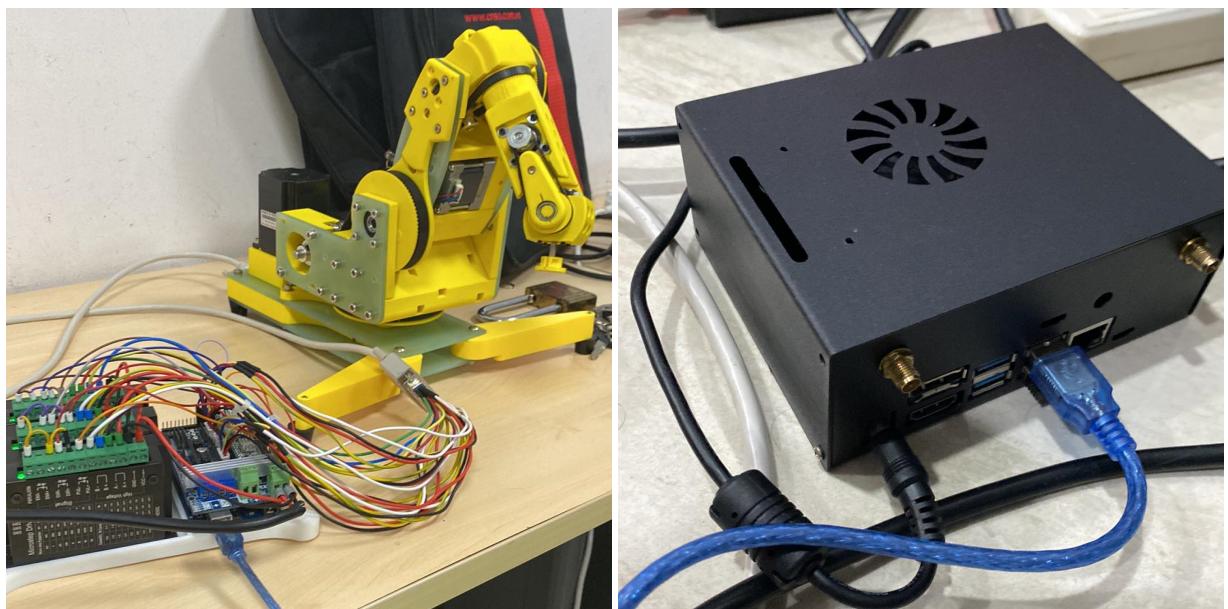
Link mã nguồn toàn bộ chương trình của nhóm:

[https://github.com/HMTCT/BKU\\_Thesis/tree/dev\\_oop](https://github.com/HMTCT/BKU_Thesis/tree/dev_oop)

## 3.7 Hình ảnh toàn bộ hệ thống



Hình 3.55: Toàn bộ hệ thống điều khiển cánh tay



(a) Cánh tay Robot và mạch driver Arduino      (b) Arduino được điều khiển bởi Jetson thông qua giao thức UART

Hình 3.56: Hệ thống điều khiển cánh tay và trung tâm xử lý Jetson Nano

## CHƯƠNG 4

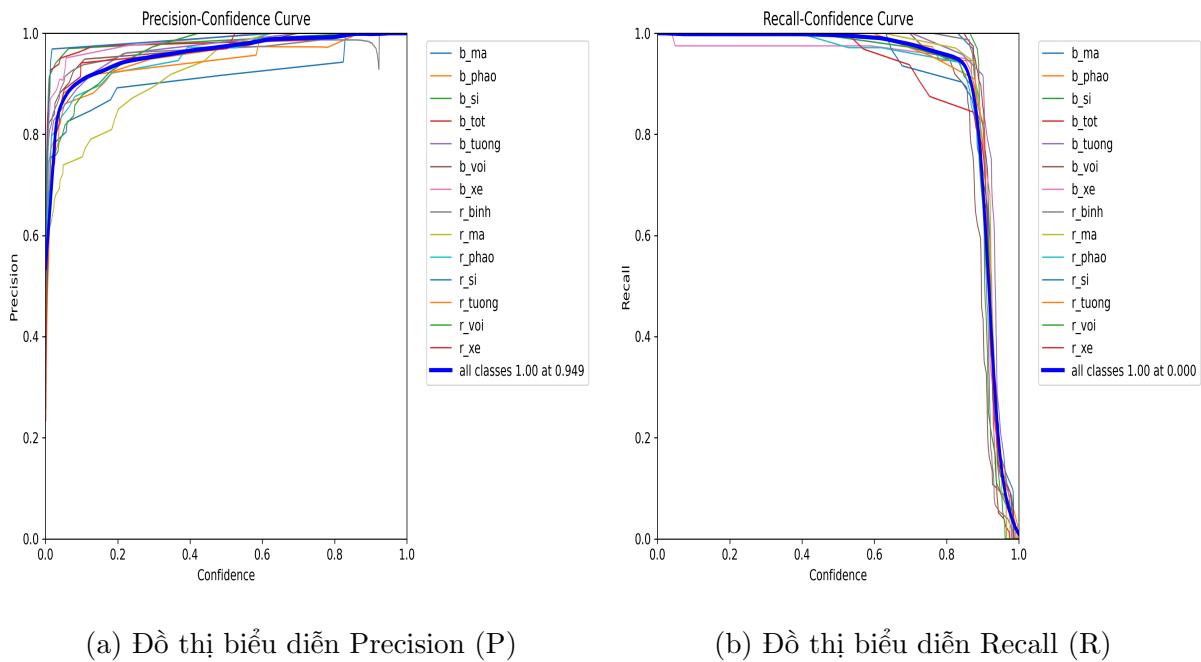
### KẾT QUẢ ĐẠT ĐƯỢC & ĐÁNH GIÁ KẾT QUẢ

#### 4.1 Mô hình nhận diện quân cờ

Với bài toán mà nhóm đặt ra là việc nhận diện quân cờ phải thật chính xác để thuật toán đánh cờ có thể phát huy một cách tốt nhất và duy trì liên tục được khả năng nhận diện cùng với đó là việc phát hiện mọi góc xoay của quân cờ. Vì thế các thông số cần thiết để đánh giá xem mô hình nhận diện của nhóm có hiệu quả hay không sẽ thông qua các thông số dưới đây:

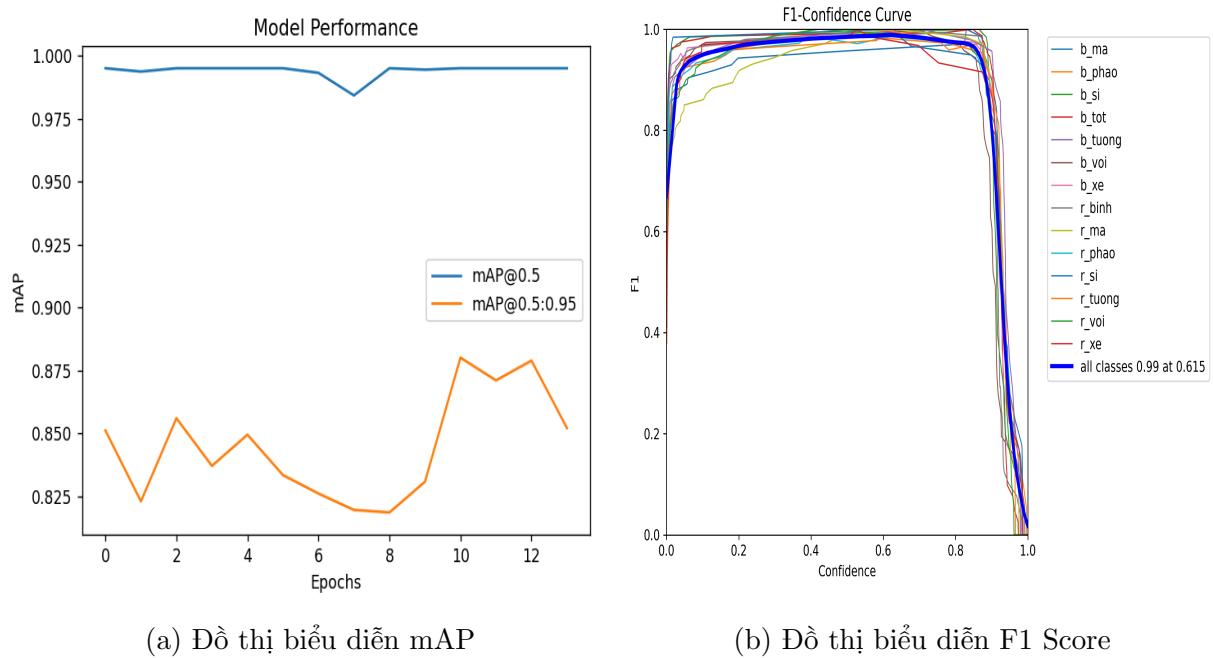
- **Mean Average Precision (mAP):** Phù hợp để đánh giá về hiệu suất của mô hình được train có ổn định hay không
- **Precision (P):** Đây là thông số dùng để đánh giá yếu tố quan trọng khi giảm thiểu phát hiện sai, độ chính xác là ưu tiên hàng đầu
- **Recall (R):** Được sử dụng khi chúng ta xem độ quan trọng khi cần phát hiện mọi trường hợp của một đối tượng là ưu tiên
- **F1 Score:** Đây là thông số cân bằng giữa độ chính xác (P) và độ nhạy (R) của việc nhận diện

Dựa trên những thông số trên nhóm đã tiến hành đánh giá và thu được các kết quả sau đây:



Nhìn chung, khi nhìn vào hai đồ thị biểu diễn Precision và Recall ta có thể nhận thấy rằng:

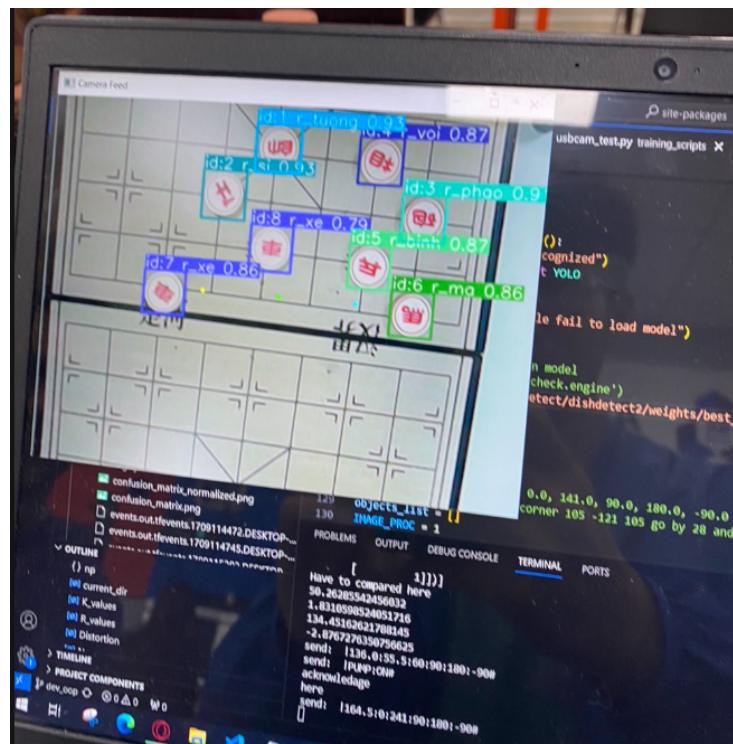
- Trong đồ thị Precision (P) hiển thị các giá trị độ chính xác tại các ngưỡng confidence khác nhau. Đường cong này giúp hiểu được cách độ chính xác thay đổi khi ngưỡng confidence thay đổi. Ví dụ, khi chúng ta ràng buộc điều kiện độ tự tin trong việc nhận diện từ 0.6 trở lên thì độ chính xác trong việc nhận diện của model có độ chính xác trên 0.8
- Trong khi đó đồ thị (Recall) minh họa cách các giá trị độ nhạy thay đổi qua các ngưỡng confidence khác nhau. Với giá trị của confidence từ 0.8 trở lên thì độ nhạy (thay đổi nhận diện) giảm đi rõ rệt và dần chạm tới 0



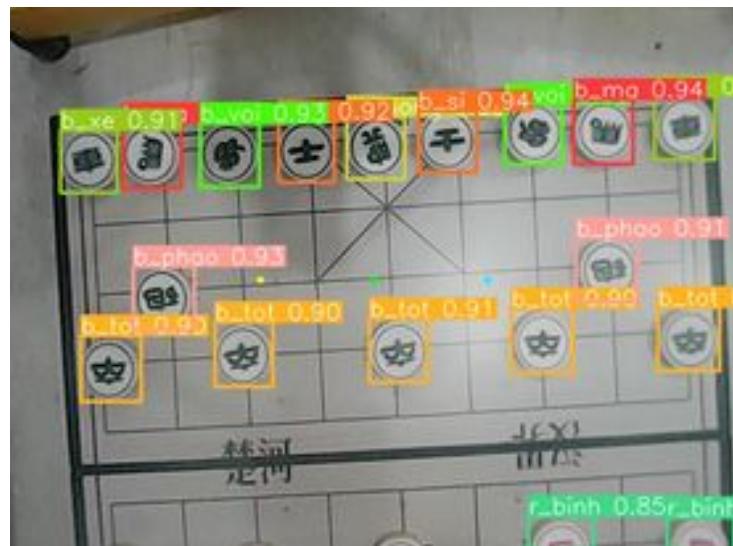
Từ 2 đồ thị về mAP và F1 Score, ta có thể nhận thấy một số điểm sau đây:

- Đồ thị F1 Score là sự kết hợp giữa Precision và Recall nên nó tổng hợp cả hai đặc điểm của hia đồ thị trên.
- Đối với đồ thị biểu diễn mAP, đường màu xanh là độ chính xác của dự đoán với điều kiện là giá trị ngưỡng IoU của dự đoán đó là 0.5 (tức là những dự đoán có kết quả trên 50% được coi là chính xác). Bên cạnh đó, đường màu cam là độ chính xác của dự đoán, khác với mAP@0.5, thì thông số này sẽ dự đoán với nhiều ngưỡng IoU khác nhau (0.5, 0.55, 0.6, 0.65,...,0.9,0.95). Sau đó, nó sẽ tính trung bình các giá trị này. Do đó, giá trị của mAP@0.5:0.95 sẽ nhỏ hơn mAP@0.5, nhưng sẽ có ý nghĩa hơn. Nhìn chung cả hai giá trị này của nhóm đều có mAP trên 0.8 thể hiện sự ổn định trong nhận diện của mô hình

Dưới đây là một số kết quả nhận diện thực tế của model:



Hình 4.3: Kết quả nhận diện thực tế 1



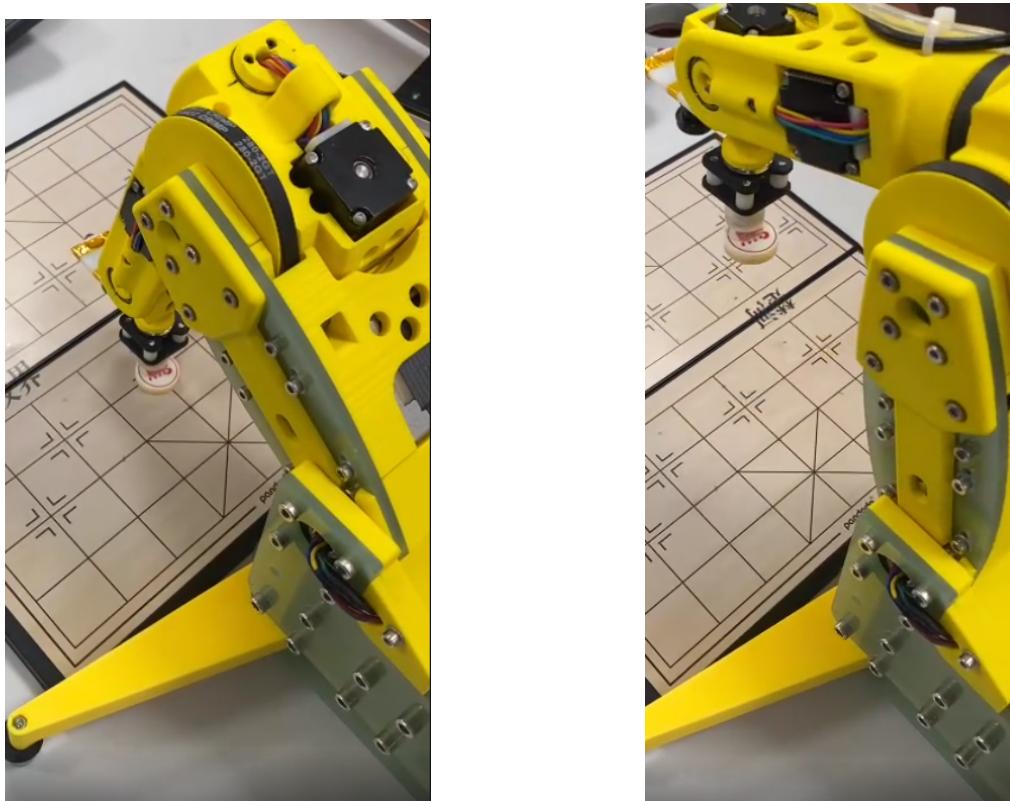
Hình 4.4: Kết quả nhận diện thực tế 2

Nhìn chung, độ chính xác của mô hình nhận diện khá cao, hầu hết độ tự tin của việc nhận diện các quân cờ đều trên 0.8 trong điều kiện ánh sáng đủ đáp ứng. Tuy nhiên nếu trong môi trường ánh sáng quá ít hoặc quá cao thì kết quả nhận diện của mô hình có thể bị sai lệch. Đối với những quân cờ nhân quân xe thì độ chính xác nhận diện đôi khi sẽ bị sai vì vị trí của những quân này trên bàn cờ là ở rìa, việc camera bắt lách góc sẽ dẫn đến góc nhận diện bị sai.

## 4.2 Hiệu chỉnh (calibration) camera và việc gấp, thả quân cờ

Việc hiệu chỉnh camera cũng đạt được những thành tựu đáng kể. Đối với việc gấp các quân cờ sẽ gặp khó khăn rất lớn nếu như không thể có độ chính xác cao từ camera vì những quân cờ này khá nhỏ với bán kính chỉ tầm 11mm, trong khi bán kính của đầu hút chỉ tầm 8mm. Nên độ chính xác đòi hỏi phải rất cao để đầu hút có thể hút được quân cờ.

Bằng việc tham khảo từ tài liệu internet và cũng như là sự gợi ý của quý thầy cô, nhóm đã có thể hiệu chỉnh được độ chính xác của camera với sai số khá nhỏ dưới 3mm. Nhờ đó việc tính toán tọa độ của các quân cờ không bị lệch quá nhiều giúp cho cánh tay robot có thể đến được và gấp chính xác được các quân cờ.



Hình 4.5: Kết quả nhận diện và gấp quân cờ

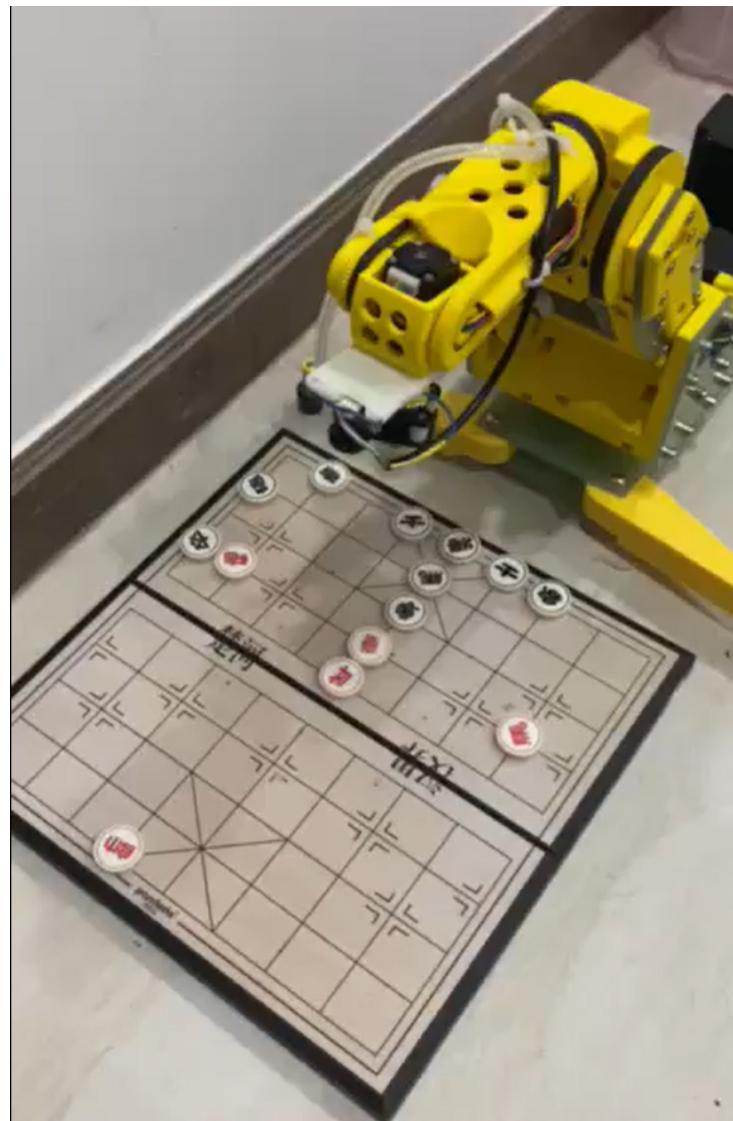
## 4.3 Đánh giá tổng quan

Toàn bộ hệ thống sẽ chạy theo như những mô tả đã nêu ở phần đề xuất chức năng. Sau khi khởi động robot, robot sẽ ở vị trí home, tức là cánh tay đòn giữa Joint 2 và Joint 3; giữa Joint 5 và Joint 6 (vị trí gắn camera) sẽ vuông góc với bàn cờ. Các cánh tay đòn còn lại sẽ song song với bàn cờ.



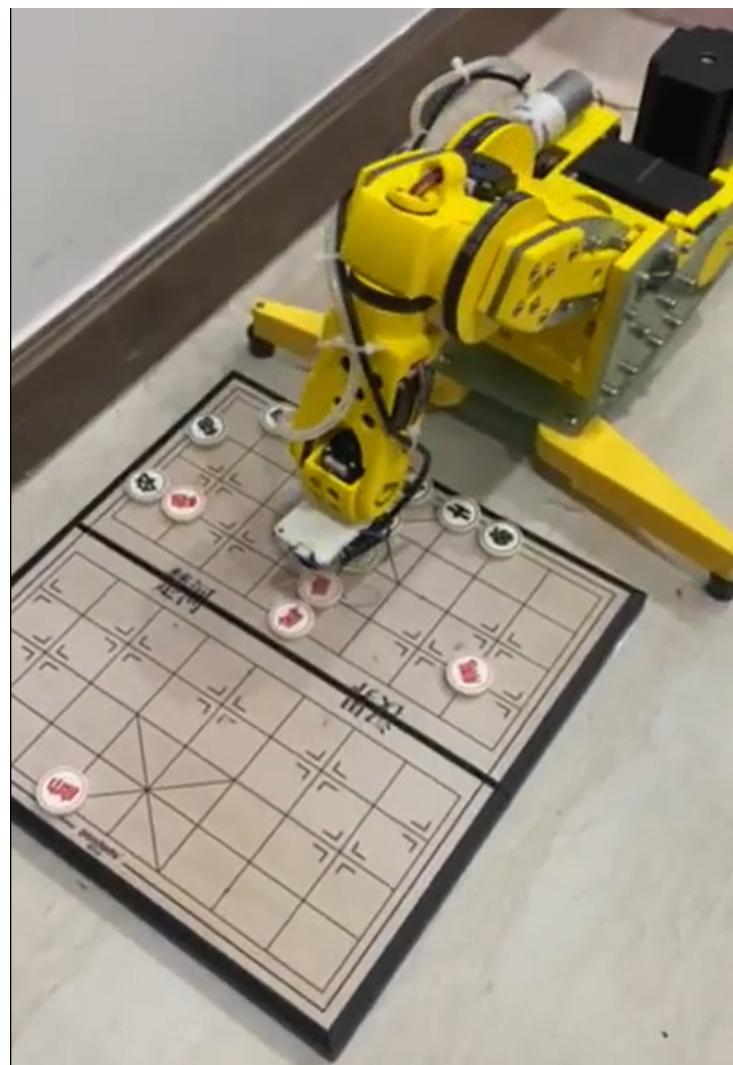
Hình 4.6: Robot ở vị trí home lúc khởi động

Lúc này camera sẽ bắt đầu gửi ảnh chụp được đến Jetson Nano. Sau đó Jetson Nano sẽ bắt đầu quá trình xử lý ảnh và kiểm tra biên. Nếu chưa có phát hiện biên thì camera sẽ quay lên một góc nhất định để thấy được biên phía bên kia của bàn cờ như hình dưới:



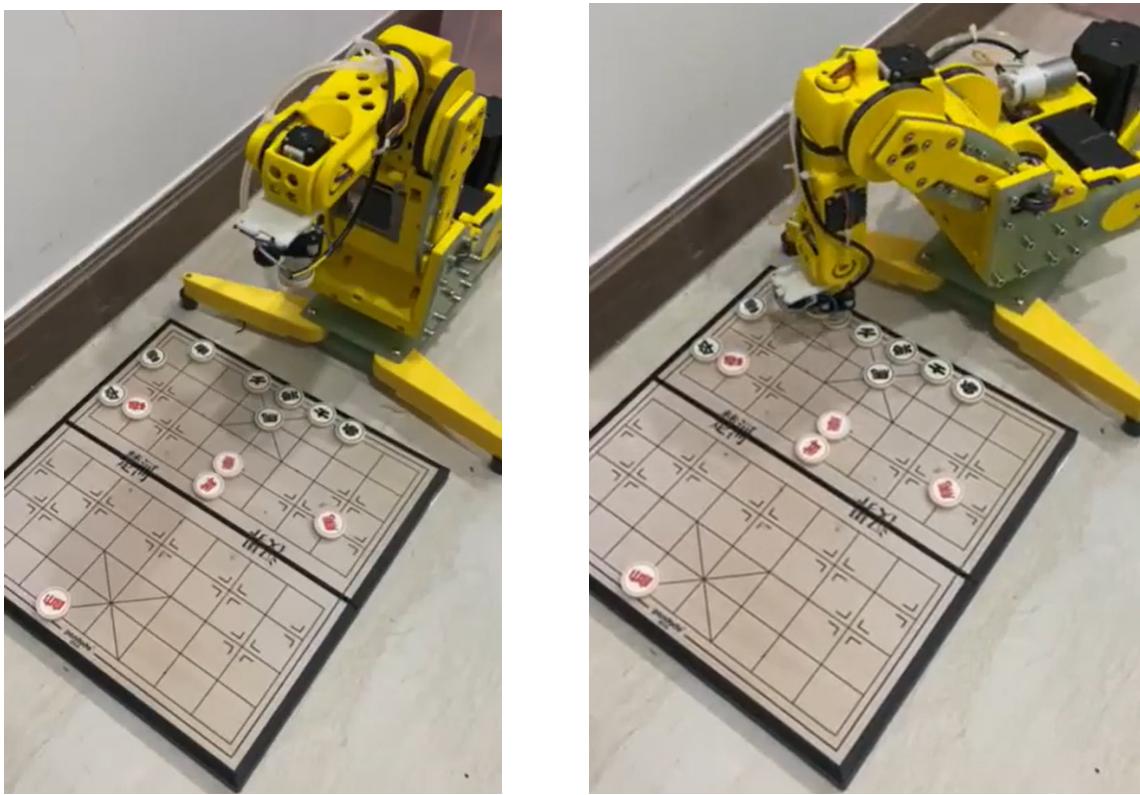
Hình 4.7: Robot nghiên một góc để nhìn phía bên kia bàn cờ

Tiếp đến robot sẽ quay lại về vị trí home như lúc khởi động, song song với đó là tiến hành xử lý ảnh để nhận diện toàn bộ quân cờ có trên bàn cờ. Dựa trên các quân cờ được nhận diện Jetson sẽ tìm ra nước cờ tốt nhất để thực hiện, phía bên dưới là hình ảnh robot di chuyển đến vị trí của quân tượng đen để thực hiện nước cờ

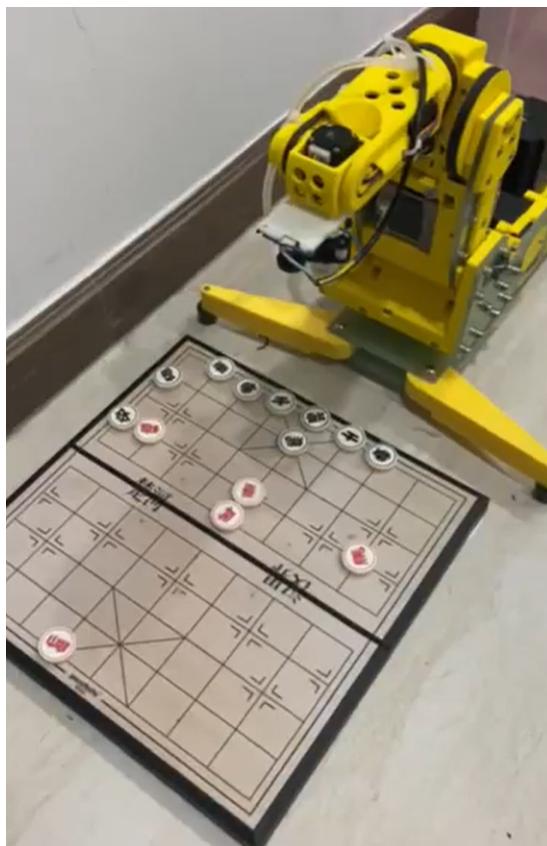


Hình 4.8: Robot gấp quân tượng đen

Khi gấp thành công quân cờ, robot sẽ di chuyển lập tức về vị trí home và sau đó di chuyển ngay về vị trí đánh cờ đã được Jetson Nano tính toán, phía dưới là quân tượng đen được di chuyển từ vị trí ban đầu đến vị trí giữa quân sĩ đen và xe đen. Và cuối cùng cánh tay robot lại di chuyển về vị trí home và lại tiếp tục vòng lặp như phía trên:



Hình 4.9: Robot đánh cờ vào vị trí mới

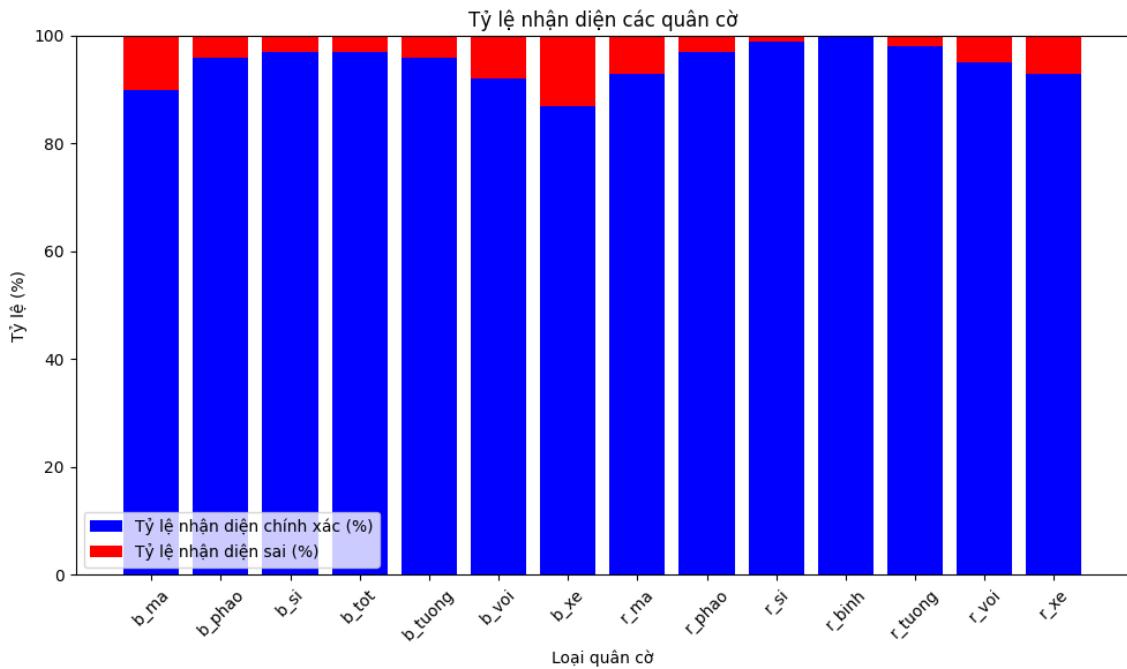


Hình 4.10: Robot quay về vị trí home sau khi đã đi nước cờ thành công

Sau khi đã hoàn thiện hệ thống, chúng ta sẽ tiến hành kiểm thử và đánh giá hiệu năng của hệ thống. Nhóm sẽ thực hiện đánh giá hệ thống 100 lần với việc sắp xếp tất cả các quân cờ lên bàn cờ ngẫu nhiên và tiến hành kiểm tra độ chính xác của việc nhận diện trong điều kiện ánh sáng đủ tốt và nhóm đã thu được kết quả sau đây:

Loại quân cờ	Tỉ lệ nhận diện chính xác	Tỉ lệ nhận diện sai
b_ma	90%	10%
b_phao	96%	4%
b_si	97%	3%
b_tot	97%	3%
b_tuong	96%	4%
b_voi	92%	8%
b_xe	87%	13%
r_ma	93%	7%
r_phao	97%	3%
r_si	99%	1%
r_binh	100%	0%
r_tuong	98%	2%
r_voi	95%	5%
r_xe	93%	7%

Bảng 4.1: Tỉ lệ nhận diện các quân cờ trên 90 lần thực nghiệm ở các vị trí khác nhau



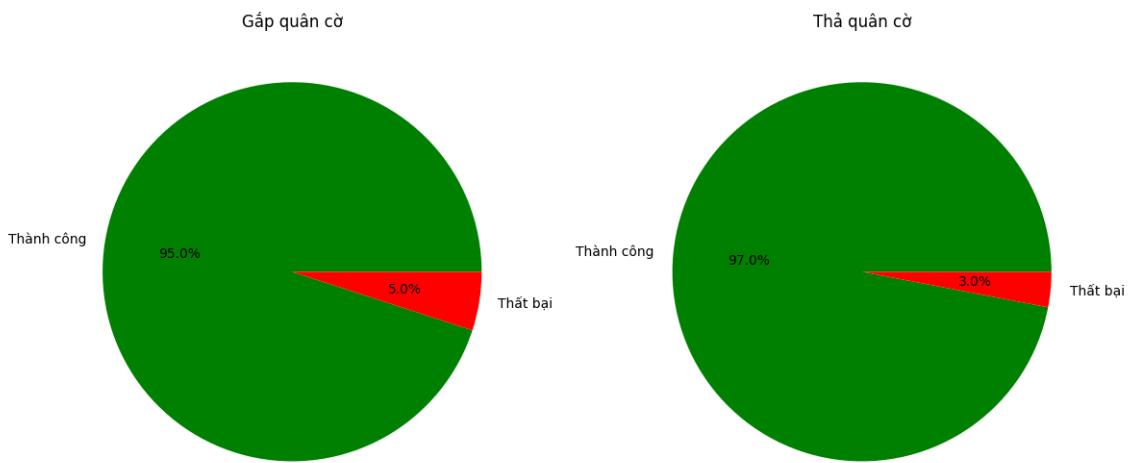
Hình 4.11: Tỉ lệ nhận diện các quân cờ trên 100 lần thực nghiệm

Nhìn chung về tỉ lệ nhận diện các quân cờ khá chính xác và ổn định trong môi trường ánh sáng đầy đủ. Tuy nhiên các quân cờ càng gần mép bàn cờ thì độ chính xác càng giảm do xa góc chụp camera nên độ rõ nét chưa được cao. Các quân màu đỏ sẽ có độ chính xác cao hơn các quân màu đen. Nếu có thêm thời gian để hiện thực, nhóm có thể huấn luyện lại mô hình với tập dữ liệu phong phú hơn và số lượt huấn luyện (epochs) sẽ lớn hơn giúp độ chính xác được cải thiện hơn nữa.

Sau đó, để đánh giá về tỉ lệ chính xác trong việc gấp thả vật thì nhóm sẽ tiến hành thực nghiệm 100 lần gấp quân cờ và thả vào vị trí khác. Sau thực nghiệm nhóm thu được kết quả sau đây:

Hành động	Tỉ lệ thành công	Tỉ lệ thất bại
Gấp quân cờ	95%	5%
Thả quân cờ	97%	3%

Bảng 4.2: Tỉ lệ gấp trúng và thả các quân cờ trên thực nghiệm 100 lần

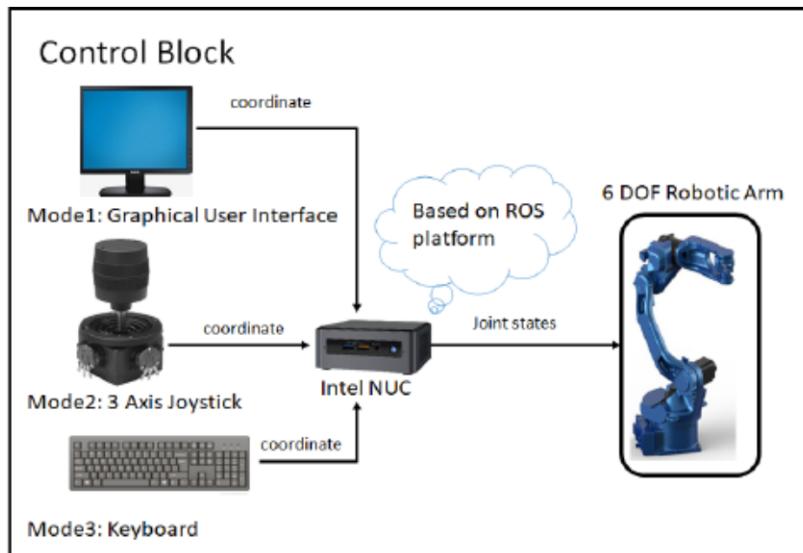


Hình 4.12: Tỉ lệ gấp trúng và thả các quân cờ trên thực nghiệm 100 lần

Việc gấp và thả vật thể chưa thẻ đạt được sự hoàn hảo do việc hiệu chỉnh camera chưa thật sự hoàn hảo (Đã đề cập ở phần **Hiệu chỉnh (calibration) camera và việc gấp, thả quân cờ**) dẫn đến tọa độ thực của các quân cờ được tính toán có thể bị chênh lệch dẫn đến một số lần bị thất bại trong việc gấp và thả quân cờ.

## 4.4 So sánh với các đề tài khác

Trong số các bài báo mà nhóm chúng tôi đã nghiên cứu về cánh tay robot, phần lớn chỉ tập trung vào việc thực hiện một cánh tay robot 6 bậc tự do (6DoF) và thiếu các trường hợp cho một ứng dụng cụ thể. Ví dụ, một số bài báo bao gồm [27] và [28] ... Những bài báo này đa phần đều tập trung vào hiện thực phân cờ khí, áp dụng các quy tắc về động học để điều khiển cánh tay robot 6 trực tự do di chuyển và tích hợp ROS vào đó. Đây cũng là điểm tương đồng với đồ án của nhóm. Với bài báo [28], kiến trúc hệ thống như hình bên dưới:

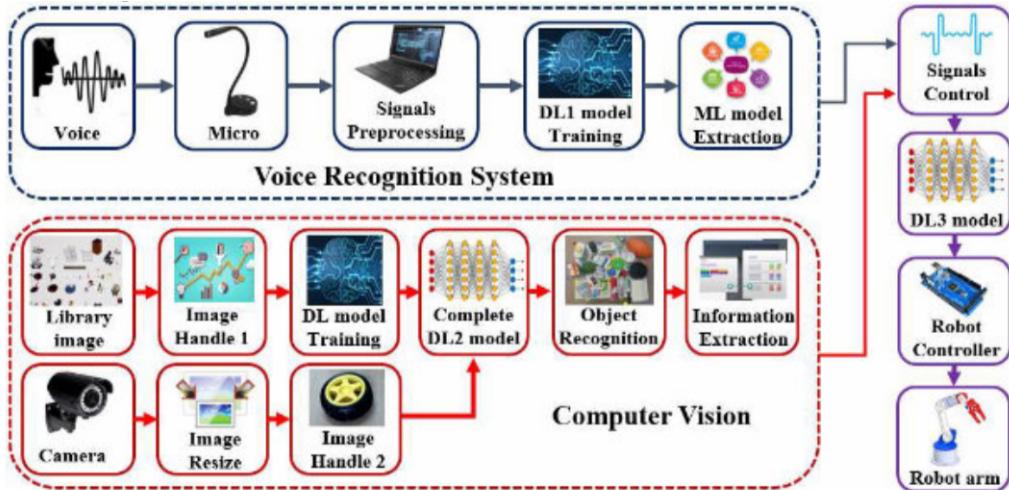


Hình 4.13: Kiến trúc hệ thống của bài báo [28]

Khi so sánh với kiến trúc hệ thống trên với hệ thống của nhóm thì có các điểm tương đồng như dùng một bộ não có bộ xử lý mạnh mẽ để điều khiển cánh tay robot, với đề tài này thì sử dụng Intel NUC và được tích hợp ROS vào đó. Tuy nhiên, với đề tài của nhóm, việc điều khiển được đảm nhận bởi cả jetson nano và arduino mega, với jetson nano là bộ não quyết định hành động của robot và xương sống là arduino mega để điều khiển hành vi của robot.

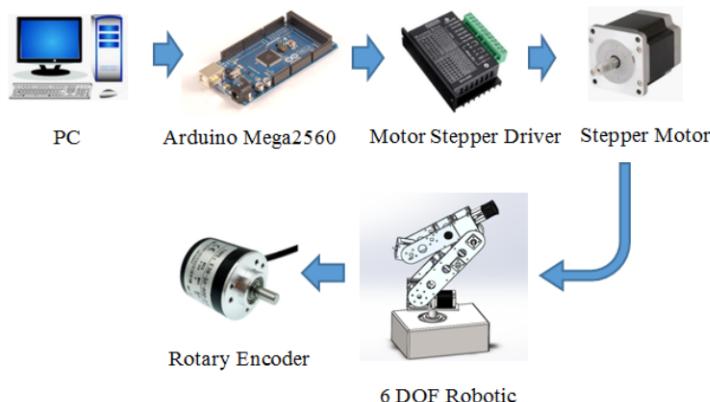
Ngoài ra, vẫn còn một số bài báo đã áp dụng các kỹ thuật tiên tiến về nhận diện và xử lý ảnh cho cánh tay robot 6 bậc tự do (6DoF). Chẳng hạn, một số bài báo như [29] áp dụng học sâu để điều khiển cánh tay robot 6DOF trong việc nắm bắt đối tượng. Bài báo này hiện thực một ứng dụng điều khiển bằng giọng nói các hành động của cánh tay robot, tích hợp vào đó là model xử lý ảnh được huấn luyện bởi YOLO. Đây là điểm tương đồng với mô hình nhận diện của nhóm. Ngoài ra, khi nhận được lệnh bằng giọng nói thì cánh tay robot trong bài báo sẽ thực hiện hành động và tiên hành

tìm kiếm vật thể được yêu cầu. Các cử động của robot trong bài báo này được điều khiển bởi arduino nano trong khi của nhóm là arduino mega.



Hình 4.14: Kiến trúc hệ thống của bài báo [29]

Một ví dụ khác là bài báo [30] áp dụng lập kế hoạch quỹ đạo cho các chuyển động hàn phẳng thẳng trên cánh tay robot 6-DOF. Đối với cánh tay robot này, có một điểm tương đồng trong hiện thực phần cứng của nhóm là đều sử dụng stepper driver, stepper motor và arduino mega để điều khiển:



Hình 4.15: Hiện thực phần cứng của bài báo [30]

Có thể nói rằng xu hướng áp dụng học máy và xử lý hình ảnh vào cánh tay robot đang gia tăng. Điều này dẫn đến sự phát triển của các robot thông minh có khả năng vận hành tự động mà không cần sự giám sát hoặc điều khiển của con người.

## CHƯƠNG 5

### TỔNG KẾT VÀ HƯỚNG PHÁT TRIỂN

#### 5.1 Tổng kết

Tổng kết lại quá trình tìm hiểu và hiện thực đề tài này, nhóm đã đạt được một số kết quả tiêu biểu như sau:

- Về cánh tay Robot: Cánh tay hoạt động mượt mà, cả 6 trục đều có thể di chuyển linh hoạt. Đầu trục tự do thứ 6 đã được tích hợp thêm đầu hút có khả năng hút các quân cờ.
- Về điều khiển: Nhóm đã thành công trong việc sử dụng tay cầm gamepad để điều khiển cử động xoay của robot, bao gồm cử động xoay từng trực đơn và cử động xoay đồng thời nhiều trực.
- Về chức năng liên quan đến nhận diện vật thể: Nhóm đã thành công trong việc huấn luyện mô hình dựa trên YOLOv8 phục vụ cho việc nhận diện các quân cờ với độ chính xác khá cao và có thể tích hợp vào hệ thống phục vụ cho chức năng auto.
- Về khả năng xử lý ảnh: Việc hiệu chỉnh (calibration) camera của nhóm đã đạt được những thành tựu đáng kể, nhờ việc này mà camera có thể nhận diện và tính toán được tọa độ thực tế của vật thể.
- Về kết nối giữa Arduino và Jetson Nano, nhóm đã kết nối hai thiết bị thông qua chuẩn giao tiếp UART rồi từ mạch Arduino kết nối với RAMPS để điều khiển động cơ bước. Nhóm cũng đã thành công trong việc truyền dữ liệu từ Jetson Nano xuống Arduino cũng như nhận tín hiệu từ Arduino đến Jetson Nano.

Tuy nhiên, do hạn chế về thiết bị, thời gian nên hệ thống dù hoạt động tốt nhưng vẫn còn tồn tại một số điểm hạn chế như sau:

- Về cơ khí, cánh tay của robot thiết kế với biên độ cử động bị hạn chế trong việc có thể chơi được hết cả bàn cờ.
- Về mô hình nhận diện vật thể, kết quả dự đoán đang còn chịu nhiều ảnh hưởng từ ánh sáng của môi trường dẫn đến việc nhận dạng còn sai sót nhưng không nhiều.
- Về xử lý ảnh và thị giác máy tính, việc hiệu chuẩn camera vẫn còn sai số tương đối dẫn đến đôi khi tọa độ của vật được trả về có thể sai sót.
- Phần đi dây linh kiện của cánh tay vẫn còn khá rối và chưa được gọn gàng, dễ dẫn đến hỏng hóc.

Nhóm đã hiểu được cách điều khiển các động cơ stepper, đồng thời áp dụng được lý thuyết của động học robot vào việc điều khiển robot. Trong tương lai, nhóm sẽ tiếp tục phát triển và hoàn thiện driver cho cánh tay, đồng thời xây dựng một ứng dụng cụ thể để có thể chứng minh tính thực tiễn của đề tài.

## 5.2 Hướng phát triển trong tương lai

Hiện tại, hệ thống đã hoàn thiện và có thể hoạt động đúng với chức năng đặt ra. Tuy nhiên, hệ thống vẫn cần phải được cải thiện một số đặc điểm để khắc phục các hạn chế về mặt cơ khí, xử lý ảnh và thị giác máy tính,... Và dựa trên thành quả hiện tại, nhóm có đề xuất một số hướng phát triển và mở rộng hệ thống trong tương lai như sau:

- Về việc huấn luyện mô hình nhận diện vật thể, cần huấn luyện một mô hình có độ đa dạng cao hơn về các quân cờ và cải thiện độ chính xác cho mô hình.
- Về xử lý ảnh và thị giác máy tính, nghiên cứu và tìm hiểu thêm về những kỹ thuật hiệu chỉnh camera để có thể đạt được độ chính xác cao hơn trong việc nhận diện vật thể. Có thể xem xét việc thay camera với độ phân giải cao hơn.
- Xem xét nâng cấp bộ não Jetson Nano mạnh mẽ hơn để có thể thực thi những tác vụ mạnh và duy trì sự ổn định cao và liên tục.
- Về phần cơ khí, cải thiện hệ thống dây cho gọn gàng hơn, nâng cấp cánh tay có biên độ hoạt động rộng hơn để thực tiễn hơn với thực tế.
- Cải thiện phần động học để gia tăng độ chính xác và tốc độ cho cánh tay.

- Mở rộng các tính năng phụ để làm đa dạng các hành vi của cánh tay cũng như tạo nên một hệ thống đa dạng tính năng hơn.
- Kết hợp hệ thống với băng tải để có thể đạt được hiệu suất công việc cao hơn và thuận lợi hơn trong việc vận chuyển vật thể và các thùng chứa.

---

## Tài liệu tham khảo

---

- [1] U. Robotics. “Robots in everyday life.” (), [Online]. Available: <https://www.unlimited-robotics.com/post/robots-in-everyday-life>.
- [2] geeksforgeeks. “Top 10 applications of robotics in 2020.” (), [Online]. Available: <https://www.geeksforgeeks.org/top-10-applications-of-robotics-in-2020/>.
- [3] JFrog. “What is ros and why it’s needed.” (), [Online]. Available: <https://jfrog.com/connect/post/what-is-ros-and-why-its-needed/>.
- [4] M. Thomas. “The future of robots and robotics.” (), [Online]. Available: <https://builtin.com/robotics/future-robots-robotics>.
- [5] t. f. e. Wikipedia. “Six degrees of freedom.” (), [Online]. Available: [https://www.wikiwand.com/en/Six\\_degrees\\_of\\_freedom](https://www.wikiwand.com/en/Six_degrees_of_freedom).
- [6] R. Taylor, J. Funda, B. Eldridge, *et al.*, “A telerobotic assistant for laparoscopic surgery,” *IEEE Eng. Med. Biol. Mag*, vol. 14, pp. 279–288, 1995.
- [7] G. Clement, J. Vertut, R. Fournier, B. Espiau, and G. Andre, “An overview of cat control in nuclear services,” *The 1985 IEEE International Conference on Robotics and Automation, St. Louis, MO, USA*, 1985.
- [8] G. Sims. “Jetson nano review: Is it ai for the masses?” (), [Online]. Available: <https://www.androidauthority.com/jetson-nano-review-969318/>.
- [9] watelectronics. “Tb6600 stepper motor driver : Pin configuration, interface with arduino, working & its applications.” (), [Online]. Available: <https://www.watelectronics.com/tb6600-stepper-motor-driver-module/>.
- [10] geeksforgeeks. “Device driver and it’s purpose.” (), [Online]. Available: <https://www.geeksforgeeks.org/device-driver-and-its-purpose/>.
- [11] sir.upc.edu. “Appendix: Ros 2.” (), [Online]. Available: <https://sir.upc.edu/projects/rostutorials/appendixRos2/index.html>.

- [12] Wikipedia. “Denavit–hartenberg parameters.” (), [Online]. Available: [https://en.wikipedia.org/wiki/Denavit–Hartenberg%5C\\_parameters](https://en.wikipedia.org/wiki/Denavit–Hartenberg%5C_parameters).
- [13] A. Addison. “How to assign denavit-hartenberg frames to robotic arms.” (), [Online]. Available: <https://automaticaddison.com/how-to-assign-denavit-hartenberg-frames-to-robotic-arms/>.
- [14] A. Addison. “How to find denavit-hartenberg parameter tables.” (), [Online]. Available: <https://automaticaddison.com/how-to-find-denavit-hartenberg-parameter-tables/>.
- [15] Wikipedia. “Forward kinematics.” (), [Online]. Available: [https://en.wikipedia.org/wiki/Forward%5C\\_kinematics](https://en.wikipedia.org/wiki/Forward%5C_kinematics).
- [16] MathWorks. “What is inverse kinematics?” (), [Online]. Available: <https://www.mathworks.com/discovery/inverse-kinematics.html>.
- [17] Wikipedia. “Inverse kinematics.” (), [Online]. Available: [https://en.wikipedia.org/wiki/Inverse%5C\\_kinematics](https://en.wikipedia.org/wiki/Inverse%5C_kinematics).
- [18] E. Wu. “Nvidia jetson nano developer kit detailed review.” (), [Online]. Available: <https://www.seeedstudio.com/blog/2019/04/03/nvidia-jetson-nano-developer-kit-detailed-review/>.
- [19] C. Woodford. “Webcams.” (), [Online]. Available: <https://www.explainthatstuff.com/webcams.html>.
- [20] C. M. Tanis. “Operational monitoring of snow cover using digital imagery.” (), [Online]. Available: [https://www.researchgate.net/figure/Formation-of-an-image-in-a-digital-camera\\_fig1\\_337099854](https://www.researchgate.net/figure/Formation-of-an-image-in-a-digital-camera_fig1_337099854).
- [21] fdxlabs. “Calculate x, y, z real world coordinates from image coordinates using opencv.” (), [Online]. Available: <https://www.fdxlabs.com/calculate-x-y-z-real-world-coordinates-from-a-single-camera-using-opencv>.
- [22] openCV. “Camera calibration.” (), [Online]. Available: [https://docs.opencv.org/3.3.0/dc/dbb/tutorial\\_py\\_calibration.html](https://docs.opencv.org/3.3.0/dc/dbb/tutorial_py_calibration.html).
- [23] G. Jocher and A. Exel. “Ultralytics yolov8 docs.” (), [Online]. Available: <https://docs.ultralytics.com/>.
- [24] Ultralytics. “Yolov8.” (), [Online]. Available: <https://github.com/ultralytics/ultralytics>.

- [25] “Stepper motor wire color and coil pairs.” (), [Online]. Available: <https://3ddistributed.com>.
- [26] Ultralytics. “Yolov9.” (), [Online]. Available: <https://docs.ultralytics.com/vi/models/yolov9>.
- [27] D. T. Tran Thanh Tung Nguyen Van Tinh. “Development of a prototype 6 degree of freedom robot arm.” (), [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2590123023001767>.
- [28] R. N. Rajesh Kannan Megalingam Raviteja Geesala. “Multimode control and simulation of 6-dof robotic arm in ros.” (), [Online]. Available: [https://www.researchgate.net/publication/341758667\\_Multimode\\_Control\\_and\\_Simulation\\_of\\_6-DOF\\_Robotic\\_Arm\\_in\\_ROS](https://www.researchgate.net/publication/341758667_Multimode_Control_and_Simulation_of_6-DOF_Robotic_Arm_in_ROS).
- [29] T. V. B. K. V. P. P. Bien Xuan Duong Anh Ngoc Mai. “Deep learning based 6dof robot arm control with computer vision.” (), [Online]. Available: [https://www.academia.edu/45499746/IJERT\\_Deep\\_Learning\\_based\\_6DOF\\_Robot\\_Arm\\_Control\\_with\\_Computer\\_Vision](https://www.academia.edu/45499746/IJERT_Deep_Learning_based_6DOF_Robot_Arm_Control_with_Computer_Vision).
- [30] P. M. A. Muhammad Arif Nur Huda Sugeng Hadi Susilo. “Implementation of inverse kinematic and trajectory planning on 6-dof robotic arm for straight-flat welding movement.” (), [Online]. Available: <https://ojs2.pnb.ac.id/index.php/LOGIC/article/view/385/220>.
- [31] Geeksforgeeks. “Universal asynchronous receiver transmitter (uart) protocol.” (), [Online]. Available: <https://www.geeksforgeeks.org/universal-asynchronous-receiver-transmitter-uart-protocol/>.
- [32] Wikipedia. “Universal asynchronous receiver-transmitter.” (), [Online]. Available: [https://en.wikipedia.org/wiki/Universal\\_asynchronous\\_receiver-transmitter](https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter).
- [33] nshop. “Mạch điều khiển động cơ bước tb6600 5a hy-div268n.” (), [Online]. Available: <https://nshopvn.com/product/mach-dieu-khien-dong-co-buoc-tb6600-5a-hy-div268n/>.
- [34] C. Basics. “Basics of uart communication.” (), [Online]. Available: <https://www.circuitbasics.com/basics-uart-communication/>.
- [35] ScienceDirect. “Forward kinematics.” (), [Online]. Available: <https://www.sciencedirect.com/topics/engineering/forward-kinematics>.

- [36] Hiwonder. “Jetmax ai vision robotic arm powered by jetson nano.” (), [Online]. Available: <https://www.hackster.io/hiwonder-technology/jetmax-ai-vision-robotic-arm-powered-by-jetson-nano-62b1f6>.

## THÔNG TIN SINH VIÊN

Danh sách tác giả Đồ Án:

1. **Huỳnh Hoàng Ly** - ID: 2013728
  - Số điện thoại: (+84) 943.437.312
  - Email: ly.huynhbkerk20@hcmut.edu.vn
2. **Hà Trung Quyền** - ID: 2014314
  - Số điện thoại: (+84) 942.145.198
  - Email: quyen.ha110602@hcmut.edu.vn
3. **Hoàng Minh Triết** - ID: 2012262
  - Số điện thoại: (+84) 919671337
  - Email: triet.hoang030719@hcmut.edu.vn