

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



**BÁO CÁO
ĐỒ ÁN MÔN HỌC KỸ THUẬT MÁY TÍNH**

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

Ngành: Kỹ thuật máy tính

HỘI ĐỒNG: HỘI ĐỒNG 3 KỸ THUẬT MÁY TÍNH

GVHD: TS. LÊ TRỌNG NHÂN

TKHĐ: TS. NGUYỄN THIÊN ÂN

---o0o---

SVTH 1: Huỳnh Hoàng Ly (2013728)

SVTH 2: Hà Trung Quyền (2014314)

SVTH 3: Hoàng Minh Triết (2012262)

LỜI CAM KẾT

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 môn học Kỹ thuật Máy tính 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 môn học Kỹ thuật Máy tính, chúng em đã nhận được nhiều sự giúp đỡ, đóng góp ý kiến và hướng dẫn nhiệt tình của 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 các thầy cô giáo trong trường *Dai hoc Bach Khoa - Dai hoc Quoc gia thanh pho Ho Chi Minh* nói chung, các thầy cô trong khoa *Khoa học và Kỹ thuật Máy tính* 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 còn hạn chế, đồ án của chúng em sẽ không tránh được những thiếu sót cũng như mắc phải vấn 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 cho giai đoạn Đồ án Tốt nghiệp, 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 trồng 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 LƯỢC

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

| | |
|--|-----------|
| Chương 1 GIỚI THIỆU TỔNG QUAN | 2 |
| 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 |
| 1.5 Cấu trúc của bài báo cáo Đồ án | 9 |
| Chương 2 KIẾN THỨC NỀN TẢNG | 10 |
| 2.1 Kiến thức cơ bản về Device Driver | 10 |
| 2.1.1 Sơ lược về Device Driver | 10 |
| 2.1.2 Quá trình hoạt động | 11 |
| 2.2 Kiến thức cơ bản về ROS | 13 |
| 2.2.1 Sơ lược về ROS | 13 |
| 2.2.2 Kiến trúc của ROS | 14 |
| 2.2.3 Ứng dụng trong thực tế | 16 |
| 2.2.4 Tiềm năng phát triển trong tương lai | 17 |
| 2.3 Tổng quan về Robot Arm 6DoF | 19 |
| 2.3.1 Khái niệm | 19 |
| 2.3.2 Ứng dụng | 20 |
| 2.4 Động học robot | 23 |
| 2.4.1 Quy tắc Denavit – Hartenberg (DH) | 24 |
| 2.4.2 Động học thuận | 27 |
| 2.4.3 Động học nghịch | 29 |
| 2.5 Tổng quan về Stepper motor | 31 |
| 2.5.1 Cấu tạo và nguyên lý hoạt động | 31 |
| 2.5.2 Nguyên lý điều khiển | 33 |
| 2.6 Tổng quan về vi điều khiển | 35 |
| 2.6.1 Jetson Nano | 35 |
| 2.6.2 Arduino Mega | 39 |
| 2.7 Giao tiếp giữa các thiết bị | 43 |

| | | |
|---------------------------|---|-----------|
| 2.7.1 | Các chuẩn giao tiếp được hỗ trợ | 43 |
| Chương 3 | THIẾT KẾ VÀ HIỆN THỰC HỆ THỐNG | 45 |
| 3.1 | Thiết kế Robot Arm 6DoF | 45 |
| 3.1.1 | Cấu tạo cơ khí của Robot Arm 6DoF | 46 |
| 3.1.2 | Kết nối điện tử trong Robot Arm 6DoF | 58 |
| 3.2 | Xây dựng kiến trúc hệ thống | 61 |
| 3.2.1 | Yêu cầu hệ thống | 61 |
| 3.2.2 | Kiến trúc hệ thống | 62 |
| 3.3 | Hiện thực hệ thống | 64 |
| 3.3.1 | Chế độ Manual | 64 |
| 3.3.2 | Chế độ Auto | 65 |
| 3.4 | Máy trạng thái | 67 |
| 3.5 | Hình ảnh toàn bộ hệ thống | 70 |
| Chương 4 | ĐÁNH GIÁ KẾT QUẢ | 71 |
| 4.1 | Kết quả đạt được | 71 |
| 4.2 | Những hạn chế | 72 |
| 4.3 | So sánh với các Robot trên thị trường | 72 |
| Chương 5 | TỔNG KẾT VÀ HƯỚNG PHÁT TRIỂN | 74 |
| 5.1 | Tổng kết | 74 |
| 5.2 | Hướng phát triển trong tương lai | 75 |
| Tài liệu tham khảo | | 76 |

Danh sách hình ảnh

| | | |
|------|--|----|
| 1.1 | 6DoF Robot Arm | 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 | 11 |
| 2.2 | Nguyên lý hoạt động của Device Driver | 12 |
| 2.3 | Kiến trúc của ROS2 [11] | 15 |
| 2.4 | Ứng dụng của ROS | 17 |
| 2.5 | Minh họa cấu trúc của cánh tay robot 6DoF | 20 |
| 2.6 | Robot Arm 6DoF Part Picking and Handling | 21 |
| 2.7 | Painting bằng Robot Arm 6DoF | 21 |
| 2.8 | Robot Arm 6DoF khoan trên bề mặt sản phẩm | 22 |
| 2.9 | Cấu hình khung tọa độ DH cho cánh tay 6DoF | 25 |
| 2.10 | Minh họa cách tìm các thông số cho bảng DH [14] | 26 |
| 2.11 | Xác định cấu hình vị trí khớp của robot sử dụng động học robot | 28 |
| 2.12 | Mối tương quan giữa động học thuận và động học nghịch | 29 |
| 2.13 | Sơ đồ nguyên lý hoạt động của động cơ bước | 31 |
| 2.14 | Hình ảnh thực tế bên trong động cơ bước | 33 |
| 2.15 | Tín hiệu điều khiển động cơ bước khác nhau | 33 |
| 2.16 | Hiệu suất của Jetson Nano | 36 |
| 2.17 | Ứng dụng của Jetson Nano trong xe tự hành | 38 |
| 2.18 | Ứng dụng của Jetson Nano trong việc nhận diện hành vi con người thông qua camera | 38 |
| 2.19 | Ứng dụng của Jetson Nano trong Robotics | 39 |
| 2.20 | Cấu tạo của Arduino Mega | 39 |
| 2.21 | Arduino Mega và Ultrasonic | 41 |
| 2.22 | Arduino Mega và LCD | 41 |
| 2.23 | Ramp Shield | 42 |
| 3.1 | Cấu tạo của Robot Arm 6DoF | 46 |
| 3.2 | Cấu tạo trực tự do J6 | 47 |

| | | |
|------|--|----|
| 3.3 | Cấu tạo trục tự do J6 | 47 |
| 3.4 | Cấu tạo trục tự do J6 | 47 |
| 3.5 | Cấu tạo trục tự do J6 | 48 |
| 3.6 | Cấu tạo trục tự do J6 | 48 |
| 3.7 | Trục tự do J6 trong Robot Arm 6DoF | 48 |
| 3.8 | Cấu tạo trục tự do J5 | 49 |
| 3.9 | Cấu tạo trục tự do J5 | 49 |
| 3.10 | Cấu tạo trục tự do J5 | 49 |
| 3.11 | Trục tự do J5 trong Robot Arm 6DoF | 49 |
| 3.12 | Cấu tạo trục tự do J4 | 50 |
| 3.13 | Cấu tạo trục tự do J4 | 50 |
| 3.14 | Cấu tạo trục tự do J4 | 50 |
| 3.15 | Cấu tạo trục tự do J4 | 50 |
| 3.16 | Cấu tạo trục tự do J4 | 51 |
| 3.17 | Cấu tạo trục tự do J4 | 51 |
| 3.18 | Trục tự do J4 trong Robot Arm 6DoF | 51 |
| 3.19 | Cấu tạo trục tự do J3 | 52 |
| 3.20 | Cấu tạo trục tự do J3 | 52 |
| 3.21 | Cấu tạo trục tự do J3 | 52 |
| 3.22 | Cấu tạo trục tự do J3 | 53 |
| 3.23 | Cấu tạo trục tự do J3 | 53 |
| 3.24 | Trục tự do J3 trong Robot Arm 6DoF | 53 |
| 3.25 | Cấu tạo trục tự do J2 | 54 |
| 3.26 | Cấu tạo trục tự do J2 | 54 |
| 3.27 | Cấu tạo trục tự do J2 | 54 |
| 3.28 | Cấu tạo trục tự do J2 | 55 |
| 3.29 | Cấu tạo trục tự do J2 | 55 |
| 3.30 | Trục tự do J2 trong Robot Arm 6DoF | 55 |
| 3.31 | Cấu tạo trục tự do J1 | 56 |
| 3.32 | Cấu tạo trục tự do J1 | 56 |
| 3.33 | Cấu tạo trục tự do J1 | 56 |
| 3.34 | Cấu tạo trục tự do J1 | 57 |
| 3.35 | Cấu tạo trục tự do J1 | 57 |
| 3.36 | Trục tự do J1 trong Robot Arm 6DoF | 57 |
| 3.37 | Sơ đồ đi dây giữa các trục tự do | 58 |
| 3.38 | Hình ảnh thực tế connector D-sub 25 | 58 |
| 3.39 | Sơ đồ kết nối driver và động cơ của J4, J5 và J6 | 59 |
| 3.40 | Sơ đồ kết nối driver và động cơ của J1, J2 và J3 | 60 |
| 3.41 | Sơ đồ kiến trúc hệ thống | 63 |

| | |
|--|----|
| 3.42 Sơ đồ các nút của game pad | 64 |
| 3.43 Camera Ohstem AI | 65 |
| 3.44 Nhận diện vật thể dùng YOLOv8 | 66 |
| 3.45 Máy trạng thái cho chế độ Manual và Auto | 67 |
| 3.46 Máy trạng thái cho các mode | 69 |
| 3.47 Toàn bộ hệ thống điều khiển cánh tay | 70 |
| 3.48 Hệ thống điều khiển cánh tay và trung tâm xử lý Jetson Nano | 70 |
| 4.1 Hình ảnh robot 6DOF trên thị trường | 73 |

Danh sách bảng

| | | |
|-----|--|----|
| 2.1 | Bảng DH của Robot Arm 6Dof | 26 |
| 2.2 | Liên hệ giữa vi bước và mô men | 34 |
| 3.1 | Các nút điều khiển Gamepad | 65 |

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 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 với 6 bậc chuyển động tự do (Six Degrees of Freedom - 6DoF). Với khả năng chuyển động theo sáu bậc tự do, chiếc 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: 6DoF Robot Arm

6DoF Robot Arm 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, 6DoF Robot Arm 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, 6DoF Robot Arm 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, 6DoF Robot Arm 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 tử hoàn chỉnh cho cánh tay robot. Tiếp nối 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, chúng tôi còn cung cấp thị giác cho Robot thông qua camera tích hợp để có thể đề xuất các ứng dụng trong giai đoạn Đồ án tốt nghiệp của đề tài.

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 tử 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 Arm 6DoF.
- ③ Lập trình sử dụng stepper motor driver điều khiển Robot Arm 6DoF.
- ④ Tích hợp chức năng điều khiển bằng joystick cho Robot Arm 6DoF.
- ⑤ Đánh giá kết quả hiện thực trong Đồ án môn học và định hướng phát triển cho Đồ án Tốt nghiệp.

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

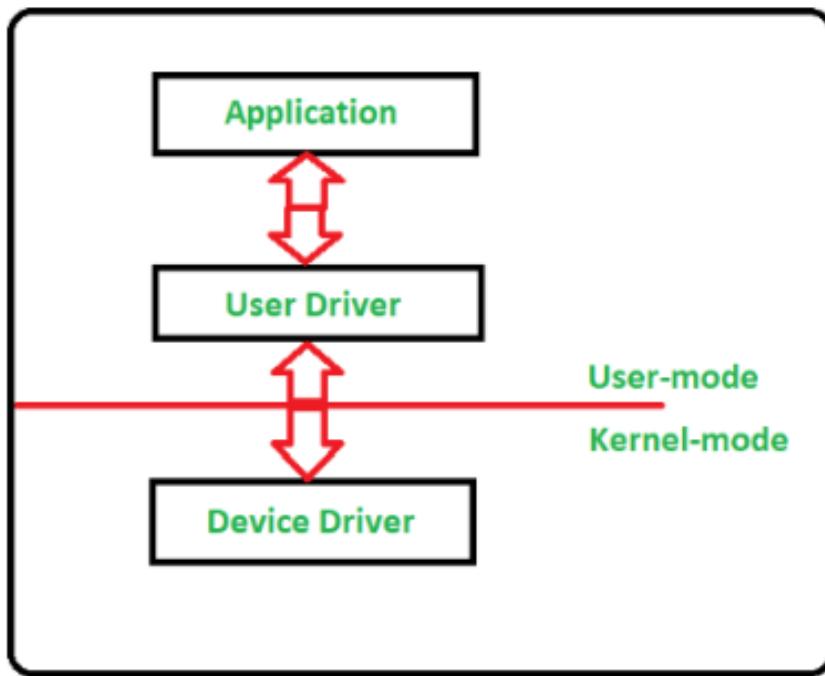
Trong chương này, nhóm sẽ đề cập đến những kiến thức liên quan và cần thiết về việc phát triển motor driver trên nền tảng ROS sử dụng mạch nhúng Jetson.

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.



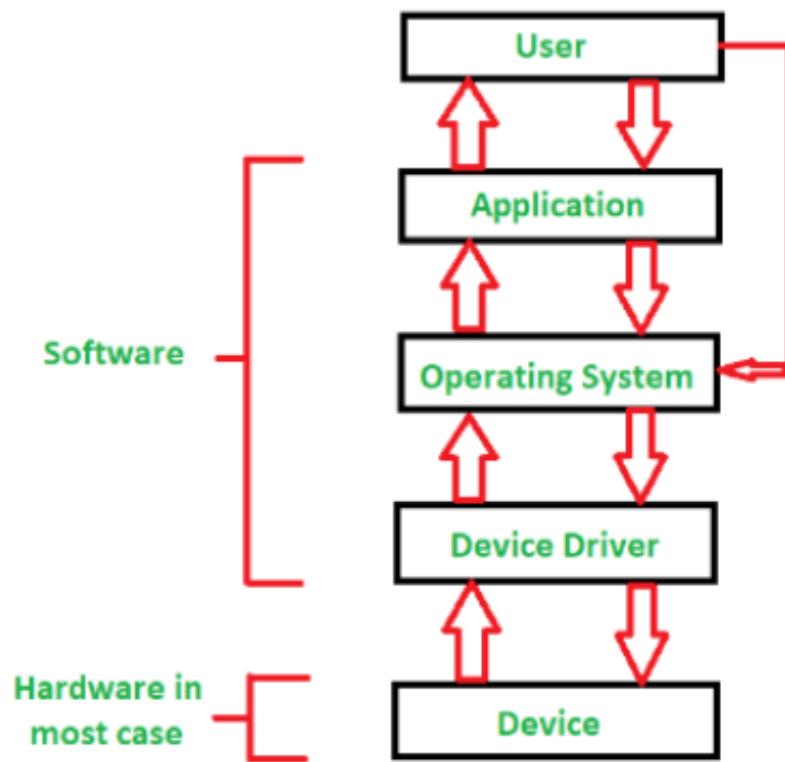
Hình 2.1: Vai trò của Device Driver trong hệ thống máy tính

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.

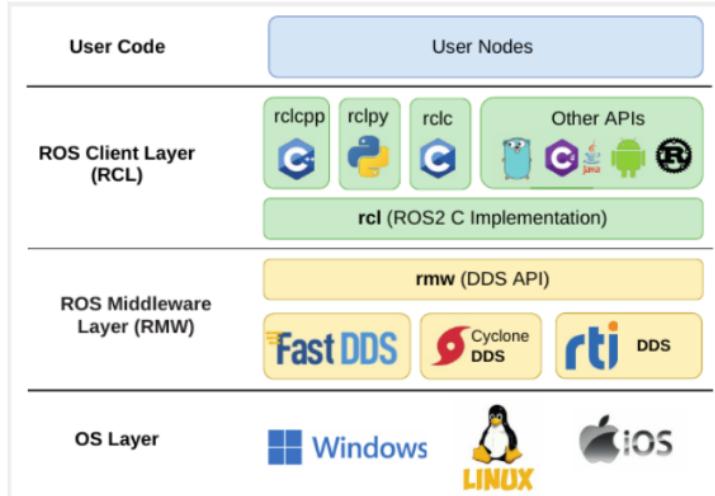
Tóm lại, để làm việc hiệu quả với ROS, người ta cần có kiến thức vững về lập trình và các khái niệm trong Robot học. Đồng thời, việc hiểu rõ về kiến trúc của ROS, sử dụng các công cụ quan trọng cùng các gói phần mềm, là chìa khóa để phát triển thành công các ứng dụng Robot đa dạng và linh hoạt.

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.

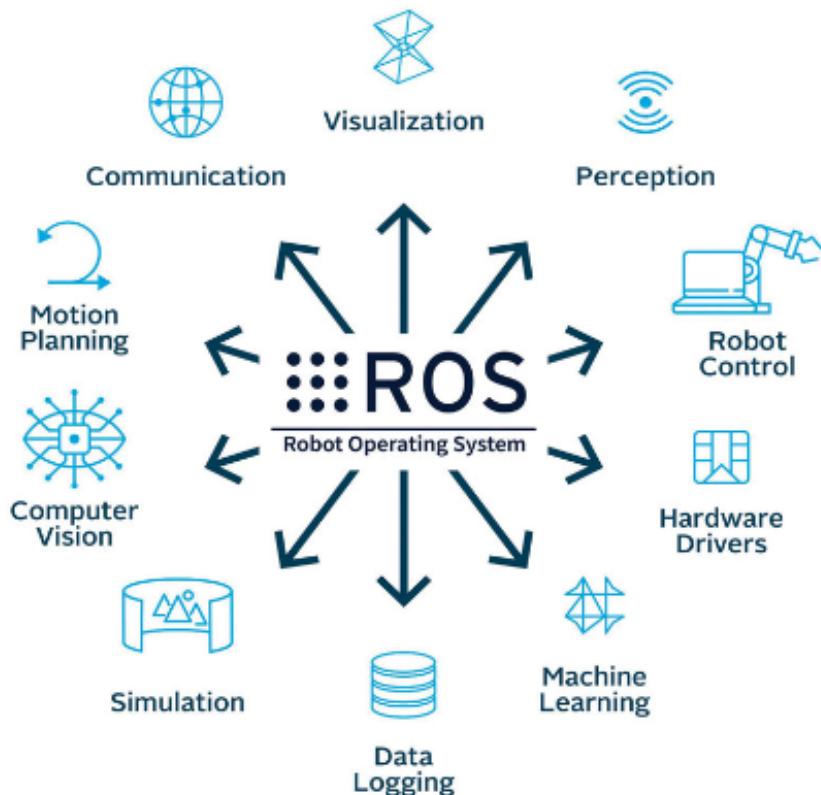
Tổng thể, kiến trúc của ROS tạo điều kiện cho việc tương tác giữa các nút và gói ROS một cách dễ dàng và hiệu quả. Việc phân tán các nút và tính năng của ROS cũng mang lại khả năng mở rộng linh hoạt, cho phép người dùng nhanh chóng mở rộng ứng dụng của họ bằng cách thêm các nút hoặc gói mới.

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

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.

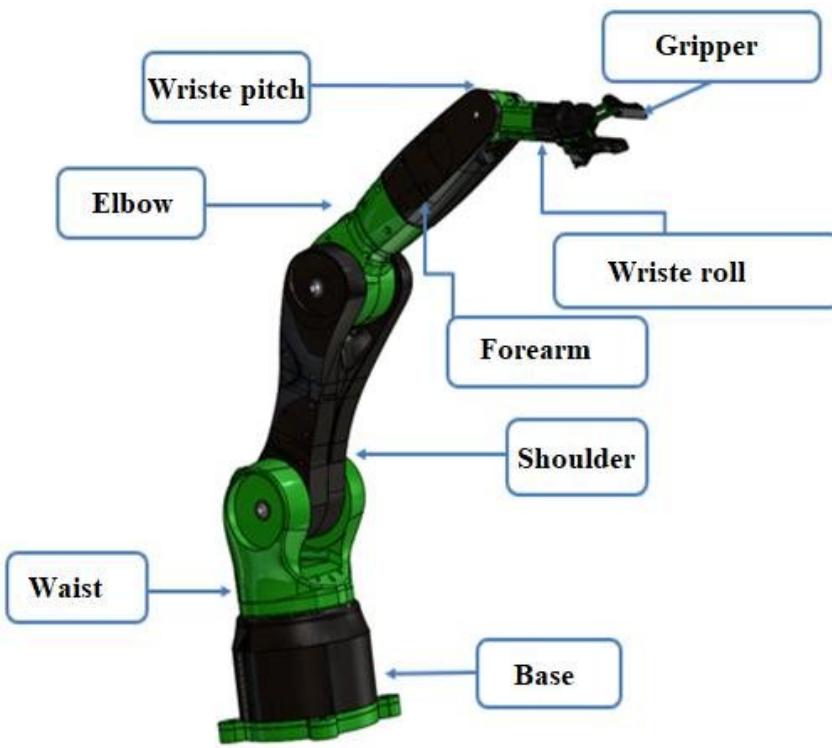
2.3 Tổng quan về Robot Arm 6DoF

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

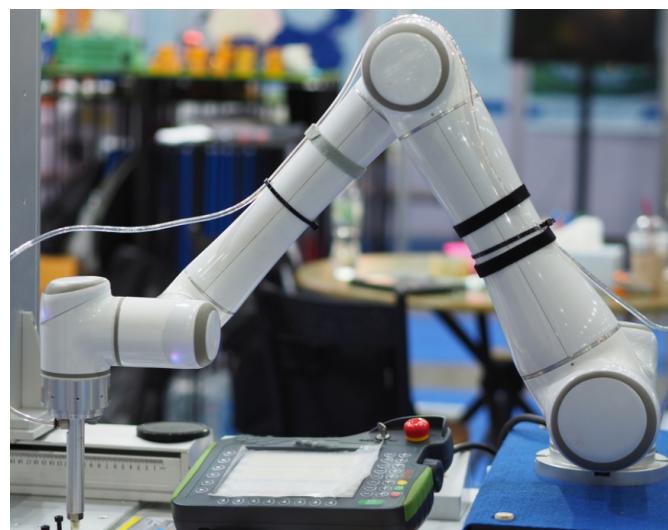
Hầu hết các loại robot có khả năng thực hiện các tác vụ gấp và đặt đơn giản, nhưng khi đến những ứng dụng đòi hỏi mức độ khéo léo cao, Robot Arm 6DoF đôi khi trở thành lựa chọn duy nhất. Một ví dụ cụ thể có thể là ứng dụng chăm sóc máy tự động, nơi mà robot cần thực hiện các chuyển động phức tạp như 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: Robot Arm 6DoF Part Picking and Handling

② Painting

Robot Arm 6DoF 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, Robot Arm 6DoF 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 Robot Arm 6DoF

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: Robot Arm 6DoF 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 α , a , d và θ , 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.

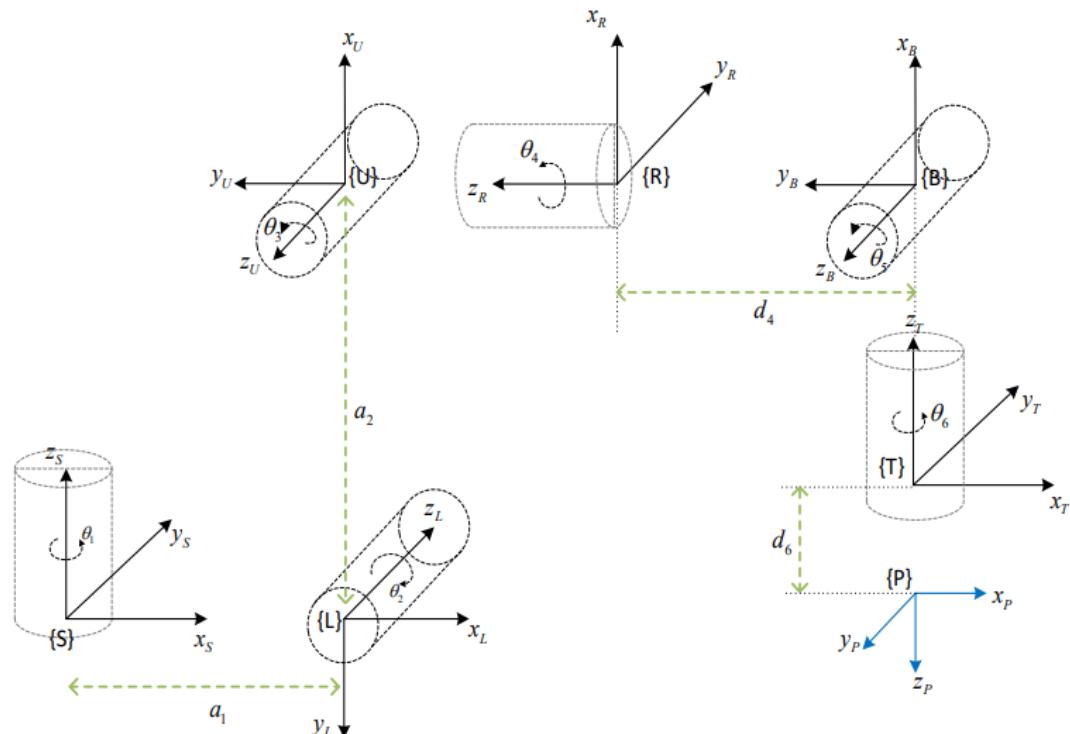
Các khung D-H của một cánh tay robot cụ thể có thể được phân loại như sau:

- **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:



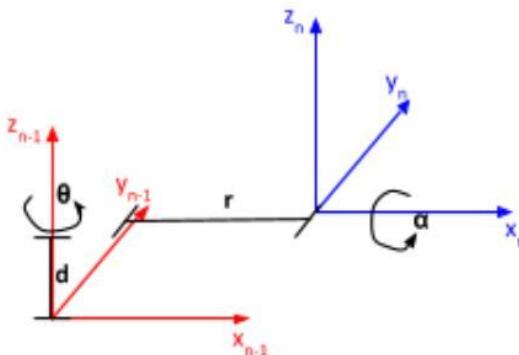
Hình 2.9: Cấu hình khung tọa độ DH cho cánh tay 6DoF

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.
- α_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.
- θ_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 Robot Arm 6DoF, ta có bảng DH như sau:

| Khớp | a_i (mm) | α_i (rad) | d_i (mm) | θ_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 | π | 28 | 0 |

Bảng 2.1: Bảng DH của Robot Arm 6Dof

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:

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

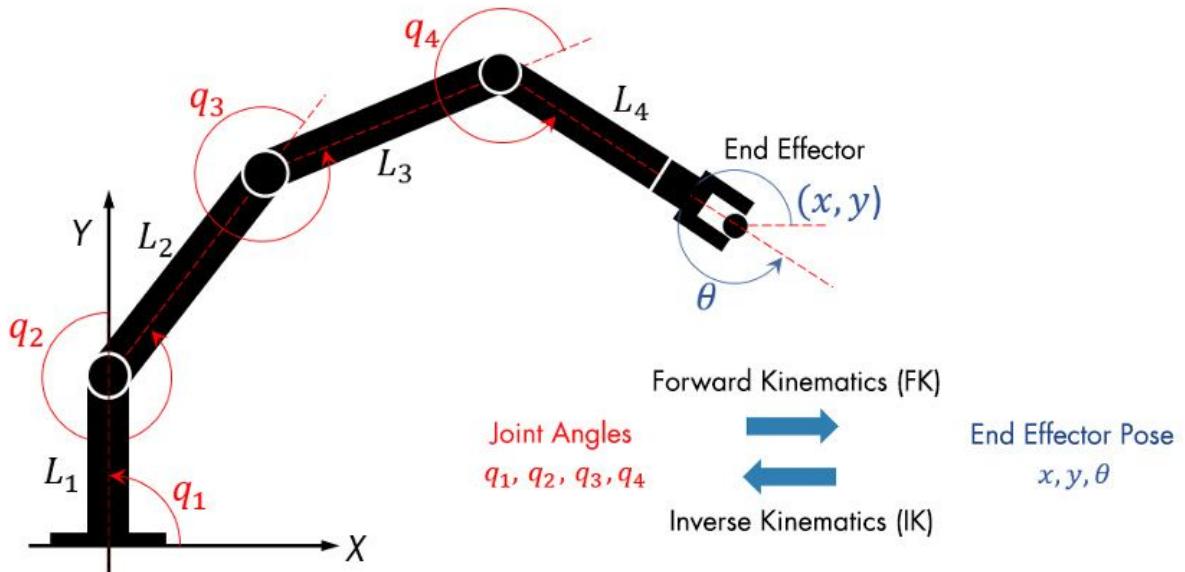
Trong đó, $c\theta$ đại diện cho hàm cosin của góc θ , $s\theta$ đại diện cho hàm sine của góc θ . Các biến θ, α, d, a được lấy từ bảng DH đã lập trước đó.

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.

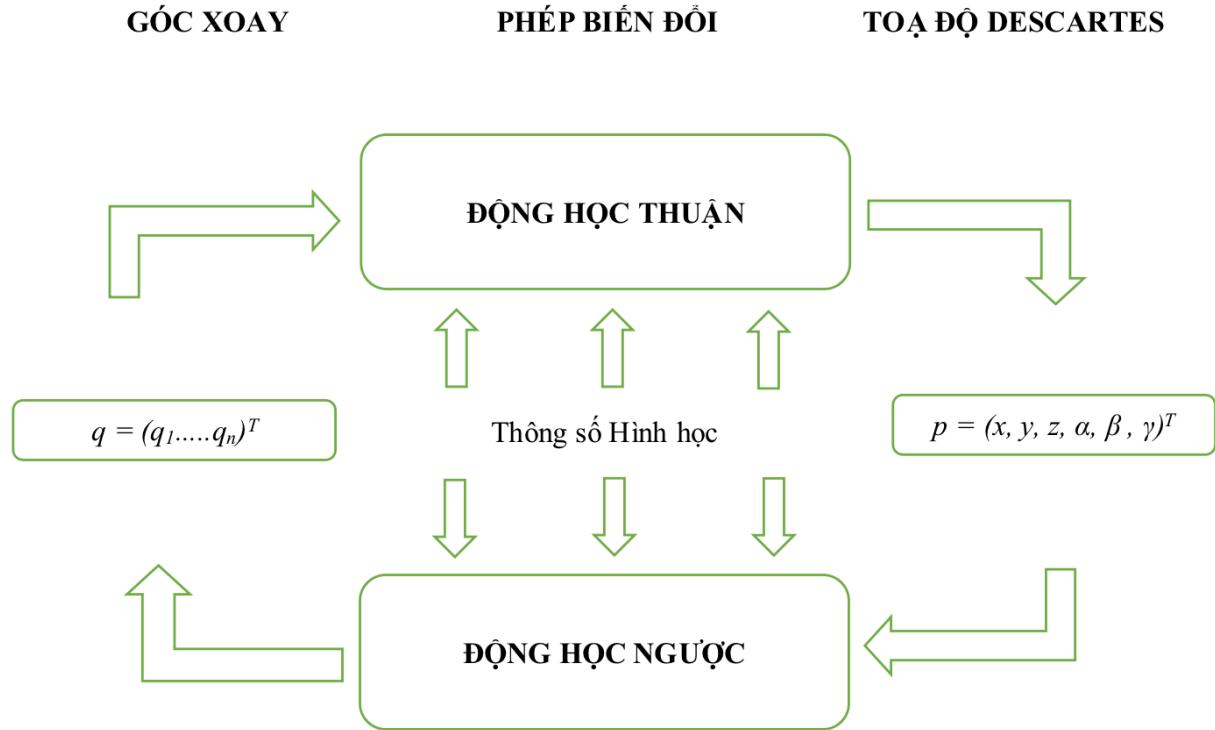
Vector $q = [q_1 \dots q_n]^T$ là vector các giá trị biến khớp. Vị trí và góc xoay của đầu công tác phụ thuộc vào vector q này.



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.

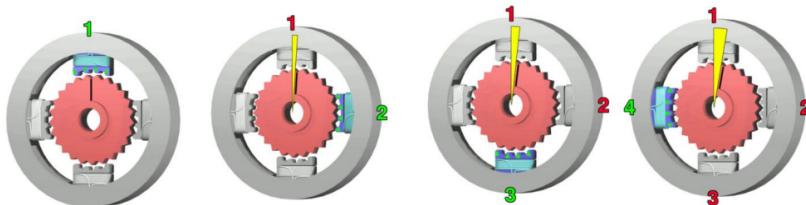
2.5 Tổng quan về Stepper motor

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° .

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.

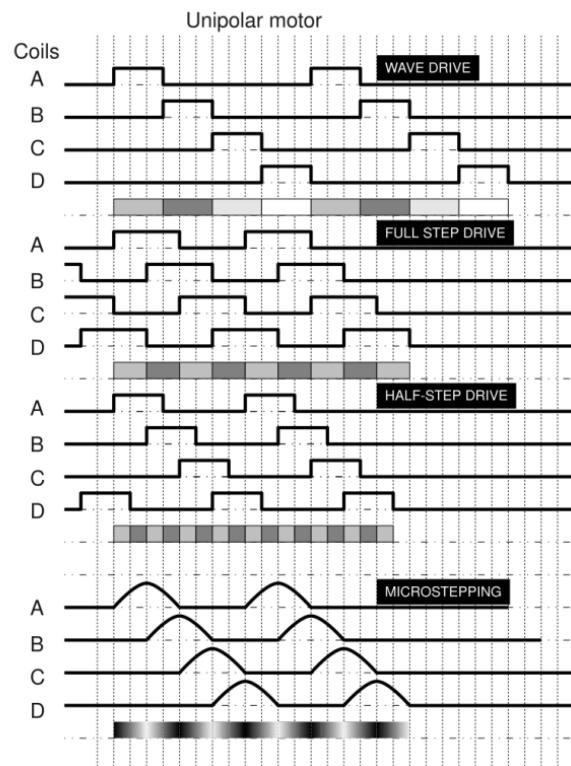


Hình 2.14: Hình ảnh thực tế bên trong động cơ bước

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.

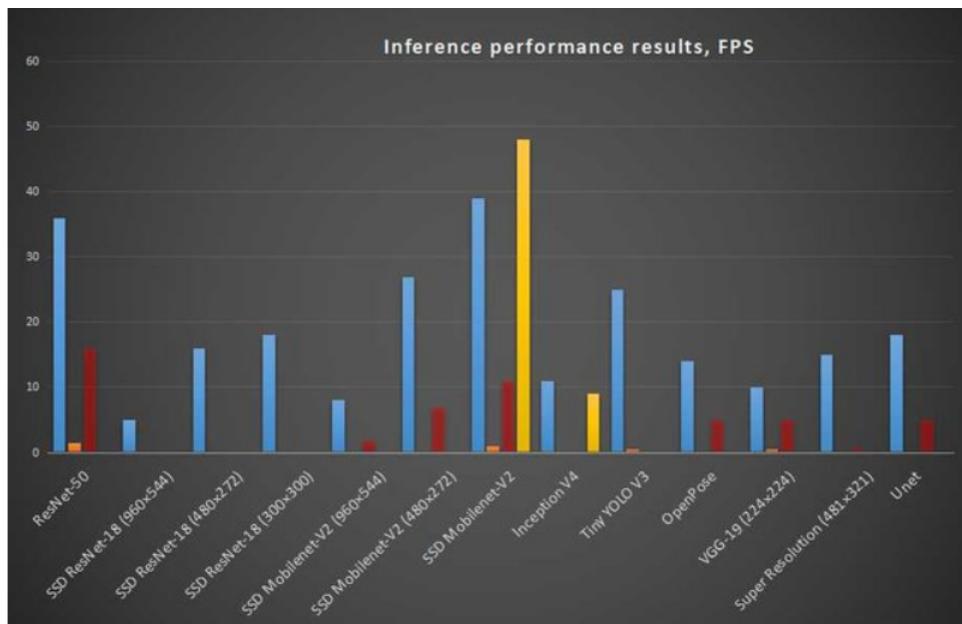
2.6.1.1 Đặc điểm nổi bật của Jetson Nano

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). Đây là một con số thú vị, nhưng thường hữu ích hơn khi so sánh với các sản phẩm khác của Nvidia hơn là so sánh với các đối thủ trực tiếp của nó, vì khi xác định tốc độ suy luận thực tế, có nhiều yếu tố khác được tính đến. 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

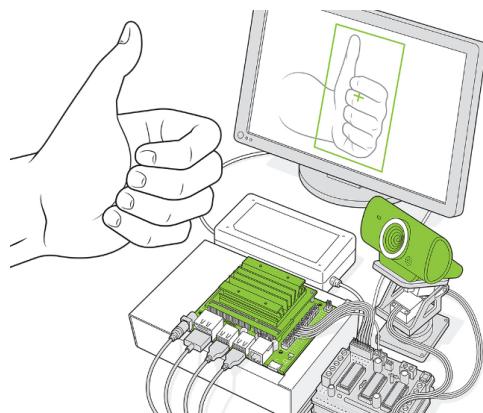
Jetson Nano, với sức mạnh tính toán và khả năng xử lý AI, có nhiều ứng dụng rộng rãi trong lĩnh vực trí tuệ nhân tạo và máy học. Một số ứng dụng tiêu biểu bao gồm:

- 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.



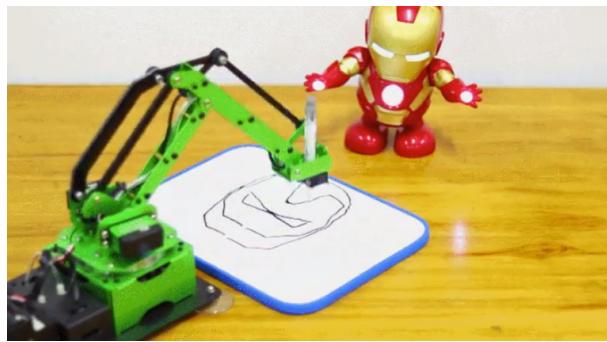
Hình 2.17: Ứng dụng của Jetson Nano trong xe tự hành

- **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:**

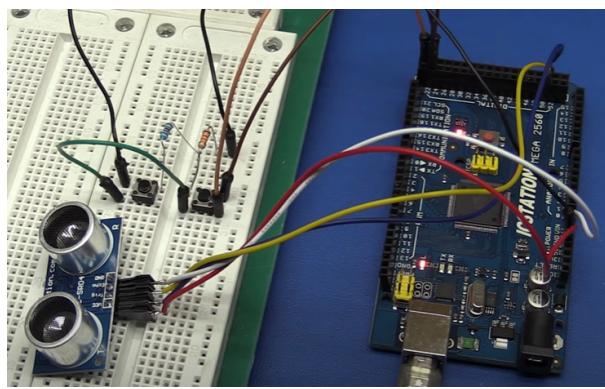
- Hỗ trợ Arduino IDE, là môi trường lập trình dễ sử dụng cho người mới bắt đầu. Ngoài ra có thể sử dụng IO PlatForm trong VScode để làm việc với Arduino Mega.
- Có thể sử dụng ngôn ngữ lập trình C/C++ thông thường.

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



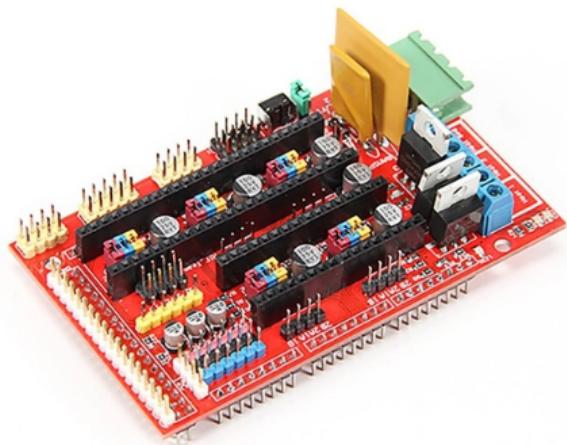
Hình 2.22: Arduino Mega và LCD

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 Các chuẩn giao tiếp được hỗ trợ

2.7.1.1 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.

2.7.1.2 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 truyền thông 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.

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 sử dụng tay cầm gamepad để kiểm thử chức năng của driver.

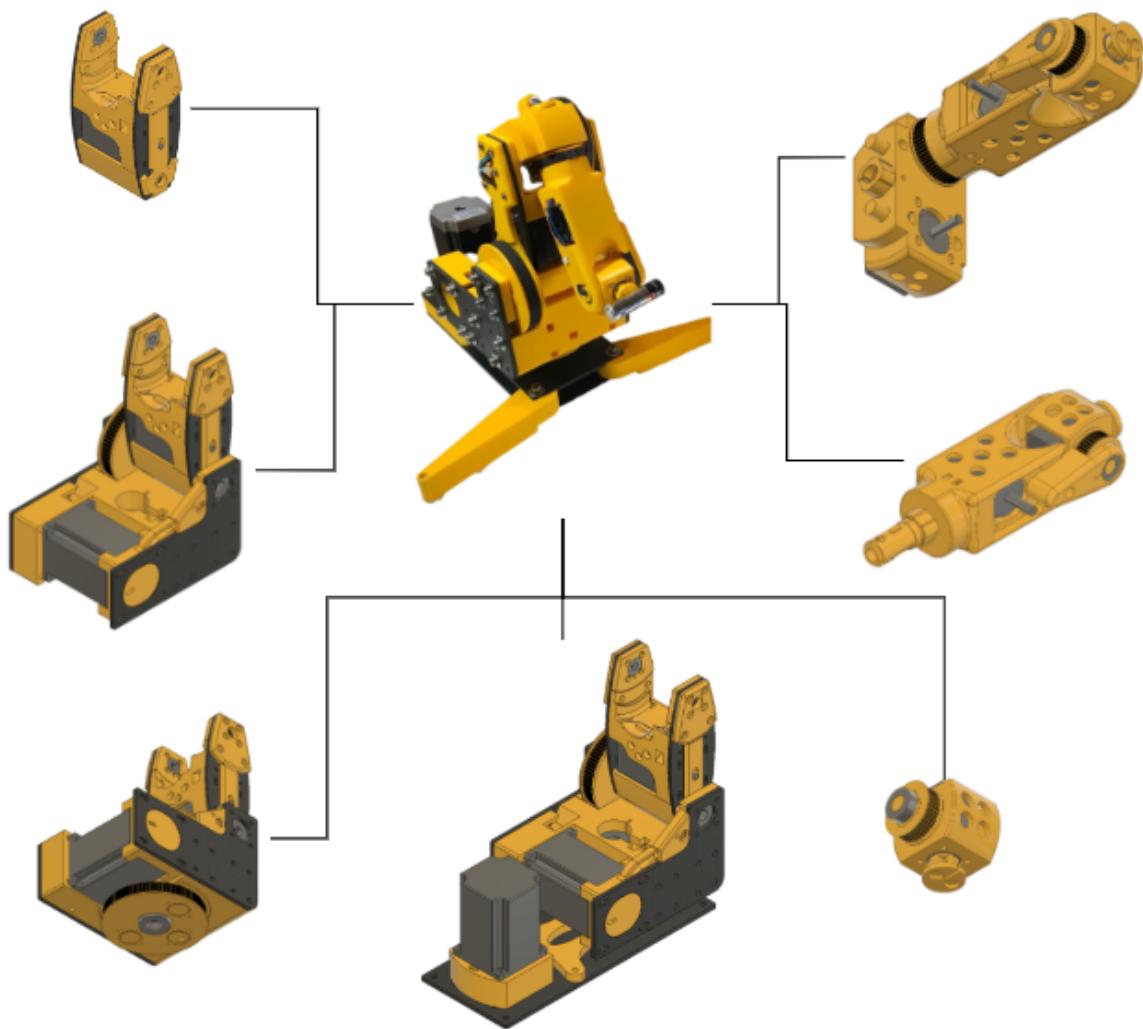
3.1 Thiết kế Robot Arm 6DoF

Ở phần này nhóm sẽ tập trung giới thiệu chi tiết về cấu tạo cơ khí và điện tử của Robot Arm 6DoF. 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 Arm 6DoF

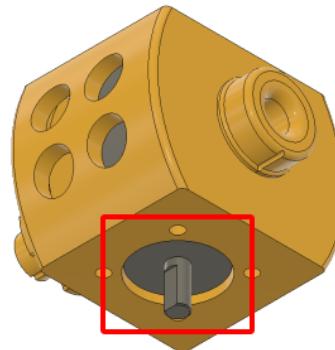


Hình 3.1: Cấu tạo của Robot Arm 6DoF

Hình trên là một bản lấp ráp cơ bản cấu tạo nên Robot Arm 6DoF, 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 Arm 6DoF.

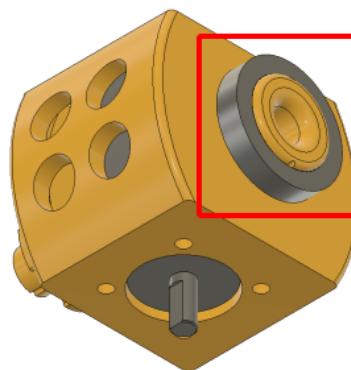
- Trục tự do J6 (Joint 6):

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



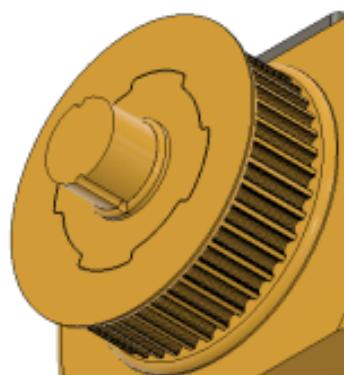
Hình 3.2: Cấu tạo trục tự do J6

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



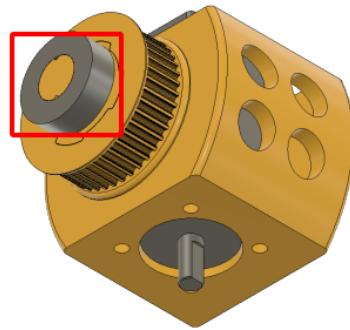
Hình 3.3: Cấu tạo trục tự do J6

- Mặt còn lại được lắp Pulley42 in 3D



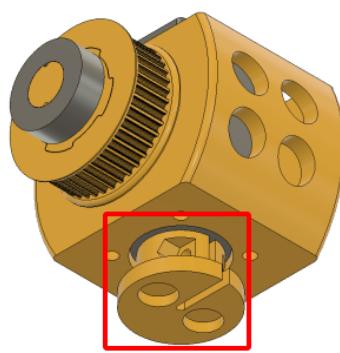
Hình 3.4: Cấu tạo trục tự do J6

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

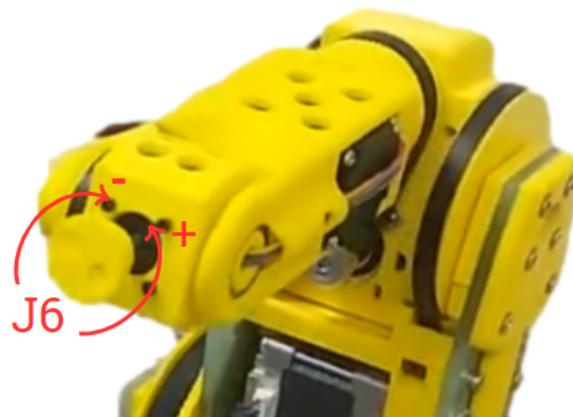


Hình 3.5: Cấu tạo trực tự do J6

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

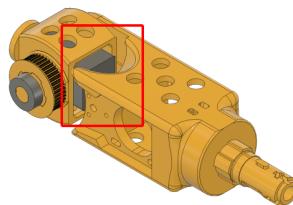


Hình 3.6: Cấu tạo trực tự do J6



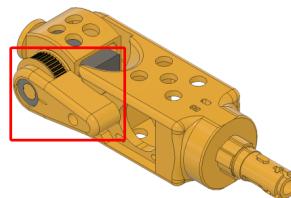
Hình 3.7: Trục tự do J6 trong Robot Arm 6DoF

- 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

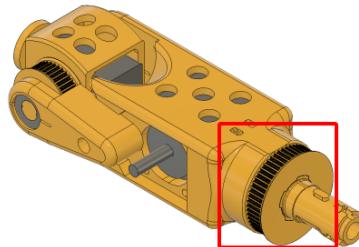
- Động cơ Nema 11 với ròng rọc GT2 20 răng và đai răng GT2 87 được cố định bằng bốn vít M2.5x8mm và vòng đệm M2.5

Hình 3.10: Cấu tạo trục tự do J5

Hình 3.11: Trục tự do J5 trong Robot Arm 6DoF

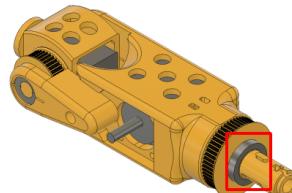
49

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



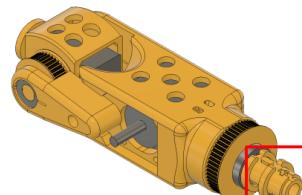
Hình 3.12: Cấu tạo trục tự do J4

- Được lắp vòng bi (21x15x4mm)



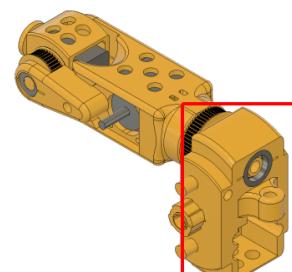
Hình 3.13: Cấu tạo trục tự do J4

- Axis4limit được lắp đặt sau ố trục



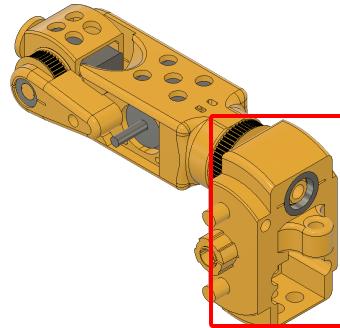
Hình 3.14: Cấu tạo trục tự do J4

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



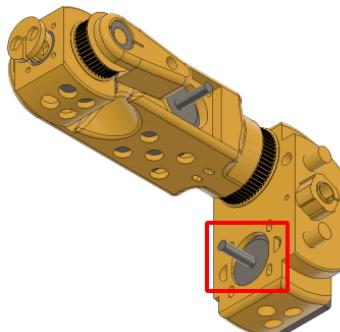
Hình 3.15: Cấu tạo trục tự do J4

- 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

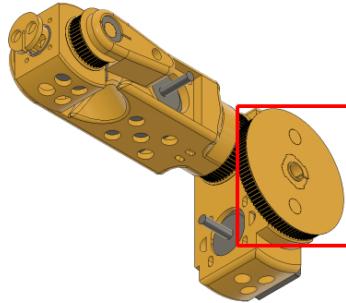


Hình 3.17: Cấu tạo trục tự do J4



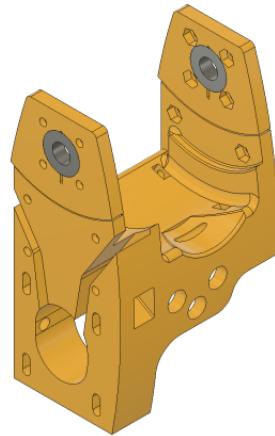
Hình 3.18: Trục tự do J4 trong Robot Arm 6DoF

- Trục tự do J3 (Joint 3):
 - Dùng Pulley100 lắp vào phần J4 phía trên



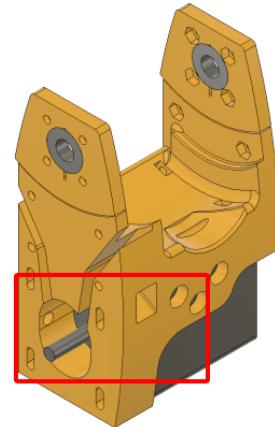
Hình 3.19: Cấu tạo trục tự do J3

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



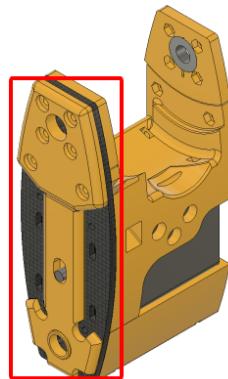
Hình 3.20: Cấu tạo trục tự do J3

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



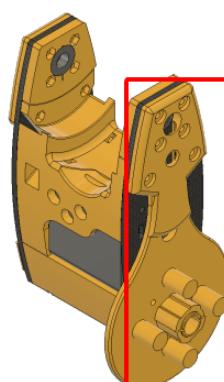
Hình 3.21: Cấu tạo trục tự do J3

- Bao quanh Axis3part2 bằng Axis3part3 với tám đai ốc M3 được lắp đặt và 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



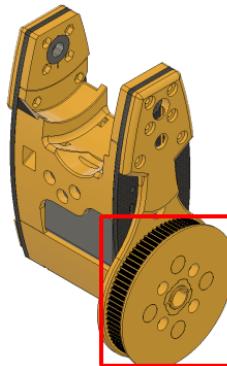
Hình 3.23: Cấu tạo trục tự do J3



Hình 3.24: Trục tự do J3 trong Robot Arm 6DoF

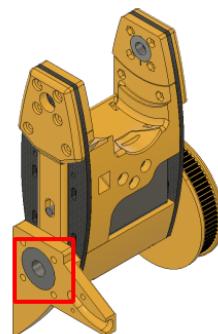
- Trục tự do J2 (Joint 2):

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



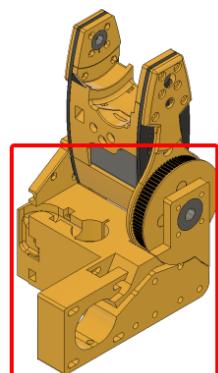
Hình 3.25: Cấu tạo trục tự do J2

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



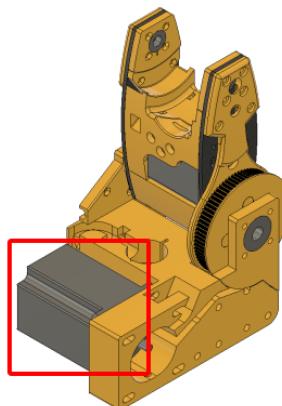
Hình 3.26: Cấu tạo trục tự do J2

- Axis2part2 được lắp đặt ở một bên của J2



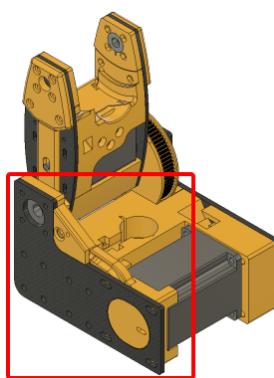
Hình 3.27: Cấu tạo trục tự do J2

- 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 còn lại và cố định bằng vít M3x12mm, vòng đệm M3 và dai ốc M3



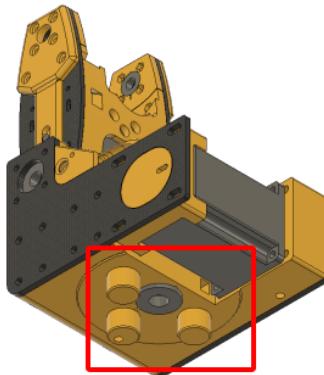
Hình 3.29: Cấu tạo trục tự do J2



Hình 3.30: Trục tự do J2 trong Robot Arm 6DoF

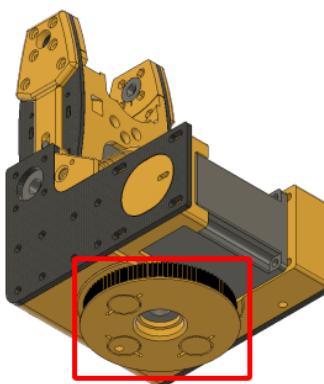
- Trục tự do J1 (Joint 1):

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



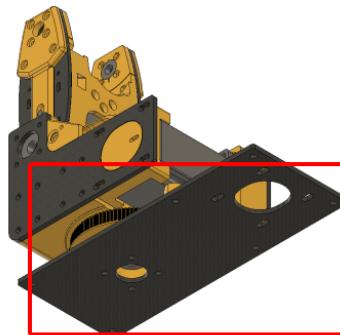
Hình 3.31: Cấu tạo trục tự do J1

– Axis1Pulley được lắp trên ổ bi dưới đáy



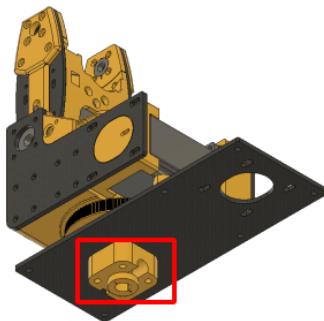
Hình 3.32: Cấu tạo trục tự do J1

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



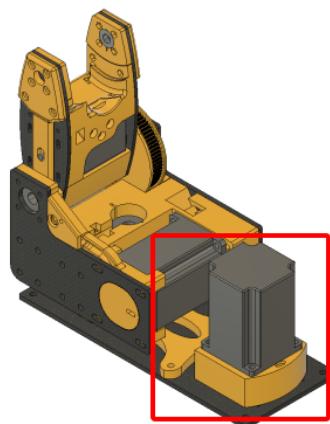
Hình 3.33: Cấu tạo trục tự do J1

- 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



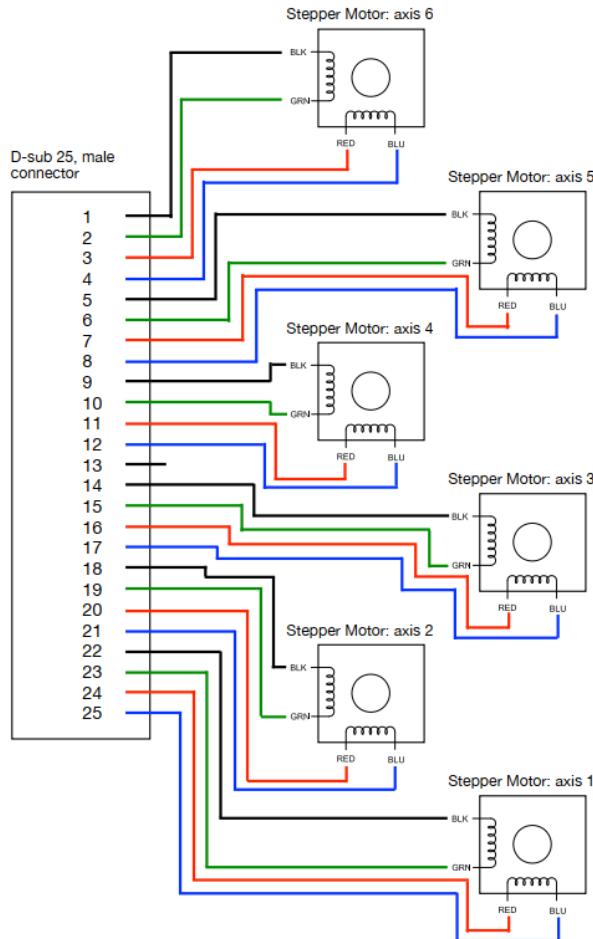
Hình 3.35: Cấu tạo trục tự do J1



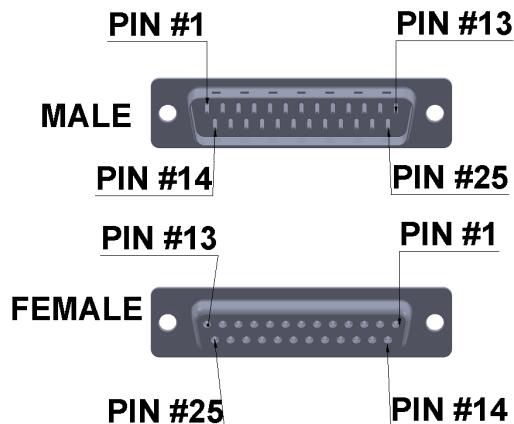
Hình 3.36: Trục tự do J1 trong Robot Arm 6DoF

3.1.2 Kết nối điện tử trong Robot Arm 6DoF

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

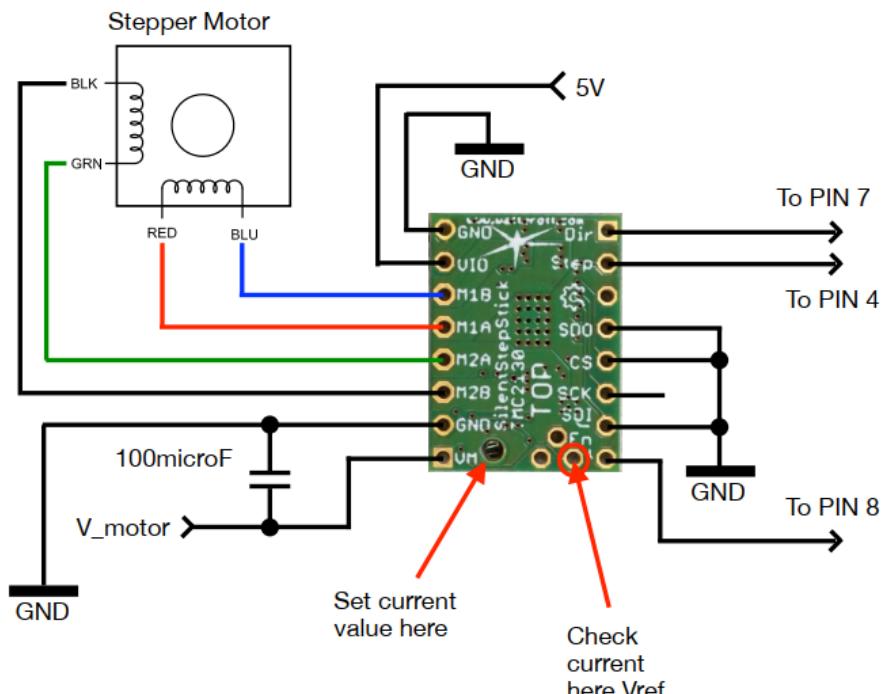


Hình 3.38: Hình ảnh thực tế connector D-sub 25

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 [19].

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.

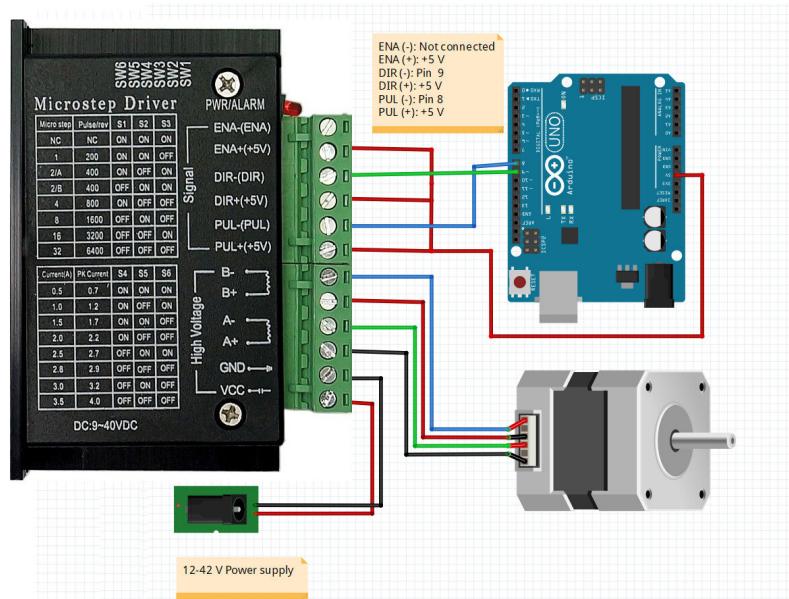
Đối với các trục J4, J5, J6 sử dụng các motor Nema nhỏ, chúng ta sử dụng driver nhỏ gọn như TMC2130 hoặc a4988 để điều khiển. Sơ đồ kết nối như sau:



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 Xây dựng kiến trúc hệ thống

3.2.1 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:

Yêu cầu chức năng:

1. **Điều khiển chính xác:** Hệ thống cần có khả năng 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.
2. **Khả năng di chuyển linh hoạt trong không gian:** 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.
3. **Tương thích với giao thức và ngôn ngữ lập trình phổ biến:** Hệ thống cần hỗ trợ các giao thức và ngôn ngữ lập trình phổ biến để dễ dàng tương tác và điều khiển từ các thiết bị và hệ thống khác.

Yêu cầu phi chức năng:

1. **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.
2. **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.
3. **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.
4. **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.

3.2.2 Kiến trúc hệ thống

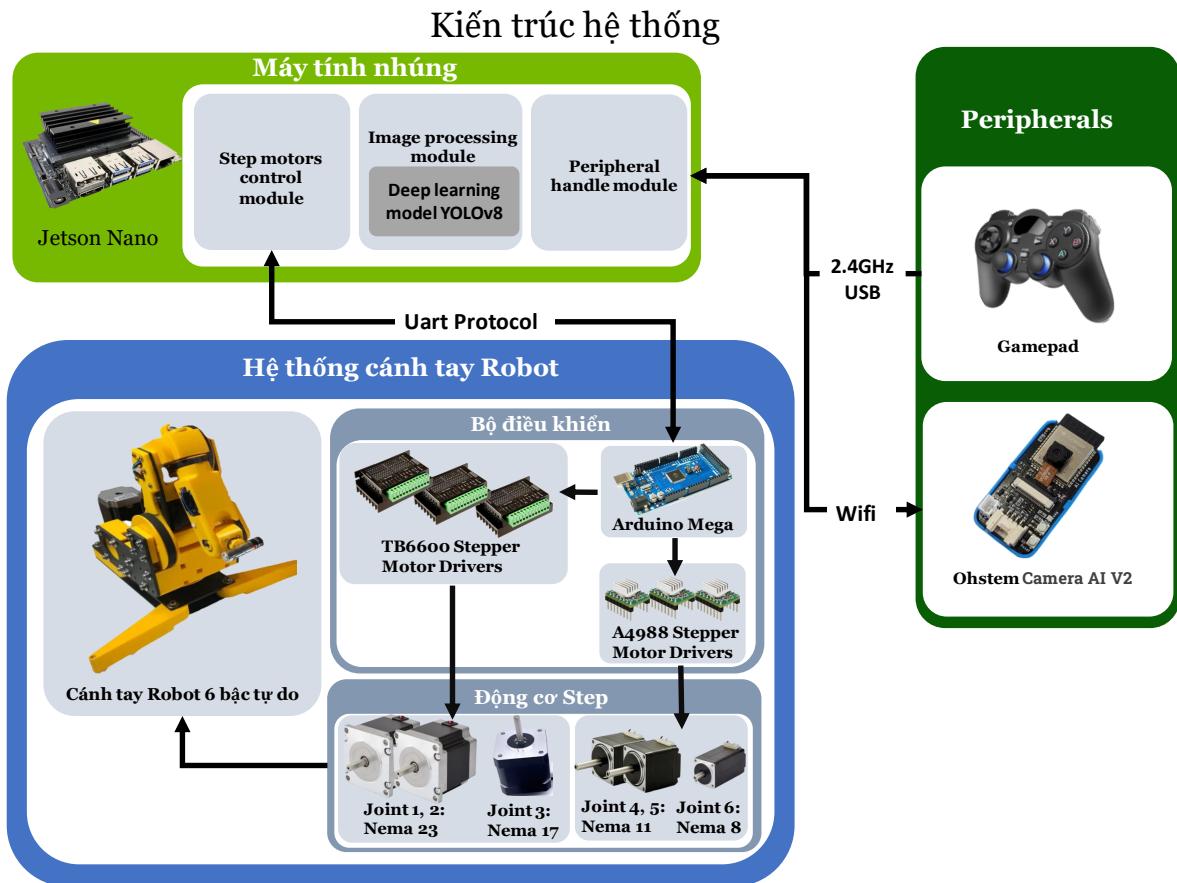
Kiến trúc hệ thống là tổng quan về cách mà các thành phần riêng lẻ của một hệ thống tương tác với nhau và hoạt động cùng nhau để thực hiện các tác vụ và chức năng. Xây dựng một kiến trúc hệ thống phù hợp sẽ mang lại các lợi ích cho hệ thống như:

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:

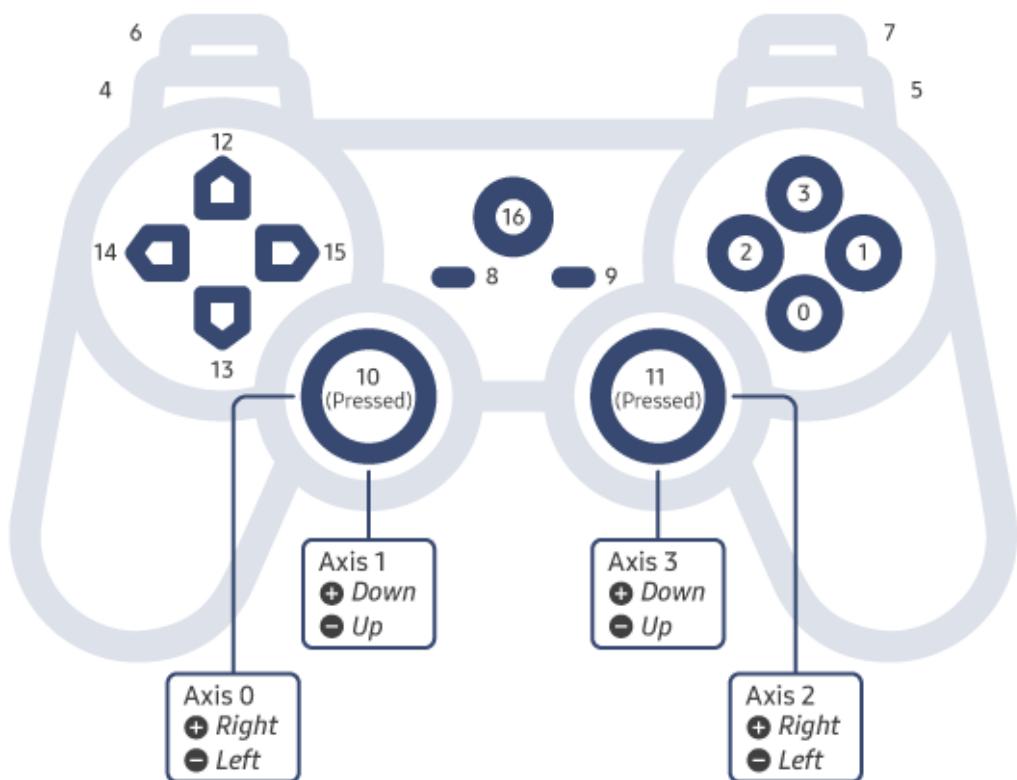
- Trung tâm xử lý được đảm nhận bởi Nvidia Jetson Nano. Tại đây, các module sẽ được pub và sub sử dụng ROS. Image processing module chịu trách nhiệm xử lý ảnh, nhận diện vật thông qua model AI. Peripheral handle module chịu trách nhiệm handle các event của các thiết bị ngoại vi như Camera và Gamepad. Từ các input trên Jetson sẽ đưa ra điều khiển phù hợp và điều khiển motor sử dụng Step motors control module.
- Dữ liệu từ Camera truyền về theo wifi sẽ là input cho model AI để nhận diện vật thể.
- Để điều khiển, máy tính 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.

3.3 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 vật thông camera và model xử lý ảnh. Vị trí của vật sau đó sẽ được tính toán và cánh tay sẽ tự gấp vật. Ngược lại đối với chế độ điều khiển bằng tay thì cánh tay sẽ được điều khiển bằng Gamepad.

3.3.1 Chế độ Manual

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



Hình 3.42: Sơ đồ các nút của game pad

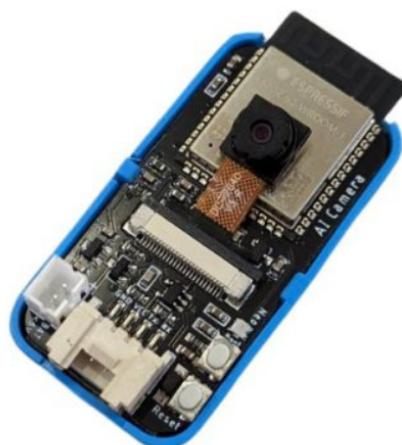
Để điều khiển cánh tay, ta cần phải điều khiển 6 bậc tự do sử dụng các nút điều khiển của Gamepad. Chuỗi điều khiển gồm 6 số là #000000. Trong đó 0 là lệnh Stop còn 1 và 2 sẽ là lệnh Move với 1 là di chuyển cùng chiều kim/đi xuống và 2 là di chuyển ngược chiều kim/đi lên.

| | |
|----------------|--------------------------|
| 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 6 | Go home (only in manual) |

Bảng 3.1: Các nút điều khiển Gamepad

3.3.2 Chế độ Auto

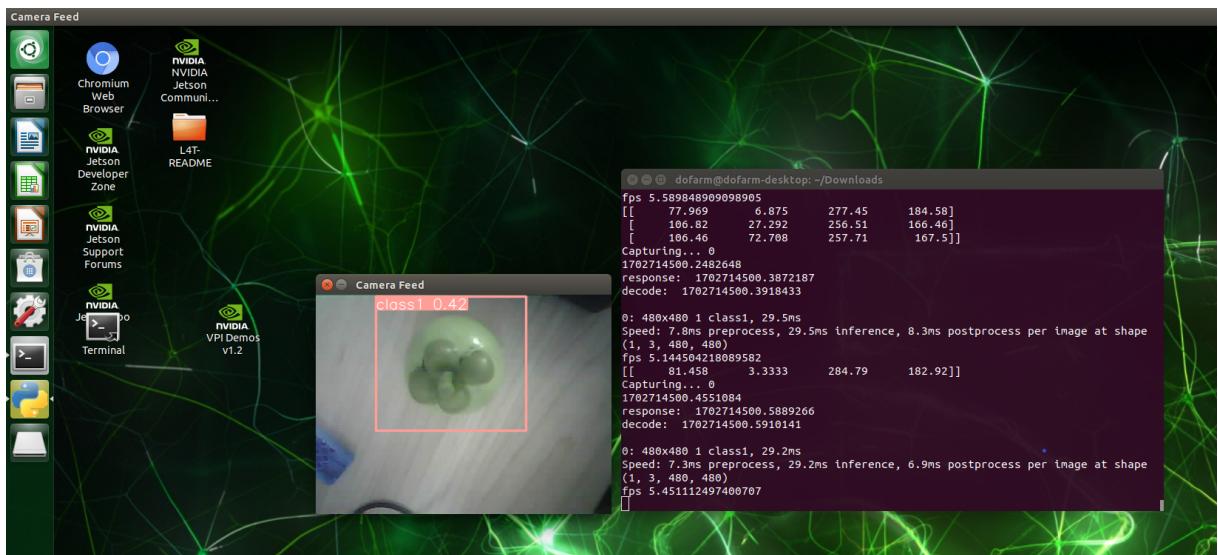
Với chế độ Auto, vật thể sẽ được tự động nhận diện thông qua camera. Hình ảnh sẽ được trả về từ Camera Ohstem AI v2 theo giao thức Wifi. Camera đã có sẵn API nên ta chỉ cần gọi API để có thể lấy ảnh.



Hình 3.43: Camera Ohstem AI

Ảnh trả về sẽ được đưa qua model học sâu YOLOv8n để có thể nhận diện vật thể. Jetson Nano sau đó sẽ tính toán tọa độ của vật từ ảnh sang tọa độ thực tế và gửi tọa độ điều khiển xuống Arduino Mega để điều khiển cánh tay.

Các vật thể trong ảnh sau khi qua xử lý sẽ được vẽ một khung bao với tọa độ góc trái trên và phải dưới (x_1, y_1), (x_2, y_2) để xác định tọa độ.

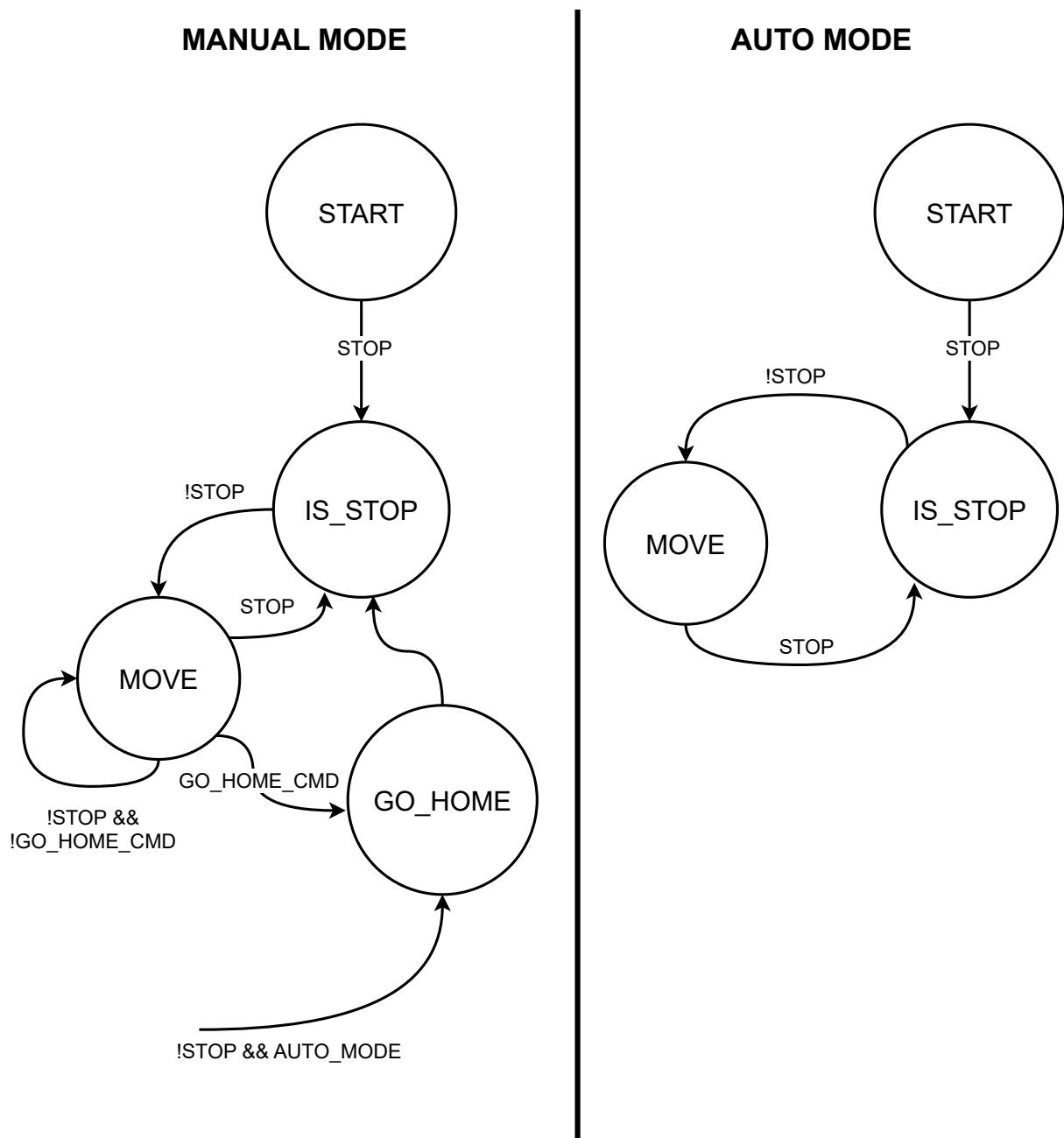


Hình 3.44: Nhận diện vật thể dùng YOLOv8

Từ đầu vào là tọa độ trên ảnh Jetson sẽ tính ra tọa độ thực tế và gửi tín hiệu điều khiển xuống cánh tay.

3.4 Máy trạng thái

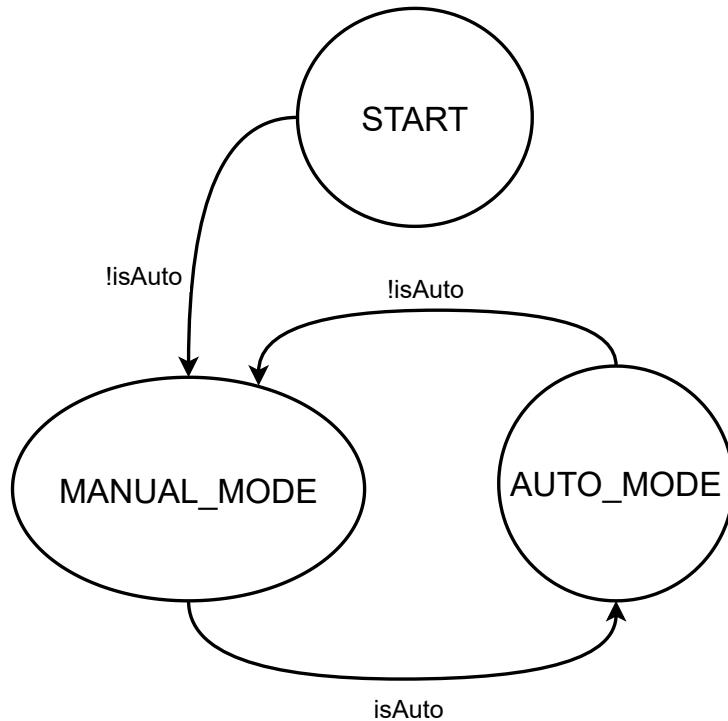
Các chế độ và hành vi của từng chế độ sẽ được kiểm soát bởi các máy trạng thái tại Arduino Mega. Với thiết kế của nhóm, cả hai trạng thái Manual và Auto sẽ có cùng cách trạng thái. Nhưng hành vi của từng Mode sẽ khác nhau.



Hình 3.45: Máy trạng thái cho chế độ Manual và Auto

- Khi hệ thống khởi động, tại Arduino, máy trạng thái sẽ ngay lập tức về trạng thái IS_STOP để đợi lệnh từ Jetson Nano. Đối với từng mode, lệnh gửi xuống sẽ có sự khác nhau.
- Manual mode:
 - Với mode Manual, Khi có một lệnh điều khiển được gửi xuống không phải là lệnh stop, mà là một chuỗi 6 số, máy trạng thái sẽ chuyển sang trạng thái MOVE. Ở trạng thái MOVE, cánh tay sẽ nhận lệnh điều khiển và di chuyển liên tục theo thông tin trong lệnh cho đến khi lệnh Stop tiếp theo được gửi đến. Lệnh stop sẽ được gửi mỗi khi ta thả nút tại Gamepad, chính vì vậy chỉ cần giữ nút, cánh tay sẽ tiếp tục di chuyển theo hướng được điều khiển.
 - Trong qua trình nhận lệnh di chuyển, nếu lệnh không phải là lệnh stop nhưng lại là go_home_cmd thì máy sẽ tiến hành di chuyển ngược lại các góc đã được lưu trong quá trình di chuyển để về lại vị trí home và về lại trạng thái IS_STOP.
 - Nếu có lệnh chuyển sang trạng thái Auto, cánh tay sẽ ngay lập tức chuyển sang trạng thái GO_HOME, về lại vị trí home và sau đó về trạng thái IS_STOP.
- Auto mode:
 - Trong chế độ Auto, Jetson sẽ tự điều khiển thông qua nhận diện vật thể.
 - Khởi tạo, trạng thái của chế độ auto vẫn là IS_STOP để đợi lệnh. Khi vật thể được nhận diện tại camera, dựa vào vị trí của vật trên khung hình Jetson sẽ tính toán và gọi hàm điều khiển cánh tay sẽ tự di chuyển đến vị trí cần thiết.
 - Khác với trạng thái Manual lệnh điều khiển của cánh tay trong trạng mode Auto sẽ là toạ độ mà cánh tay cần di chuyển đến. Khi nhận được lệnh mà không phải là Stop, cánh tay sẽ di chuyển đến toạ độ mong muốn trong lệnh và về lại trạng thái IS_STOP để đợi lệnh tiếp theo.
 - Khi chuyển từ trạng thái Auto sang Manual, cánh tay sẽ giữ nguyên vị trí và có thể được điều khiển tiếp bởi Manual.

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.



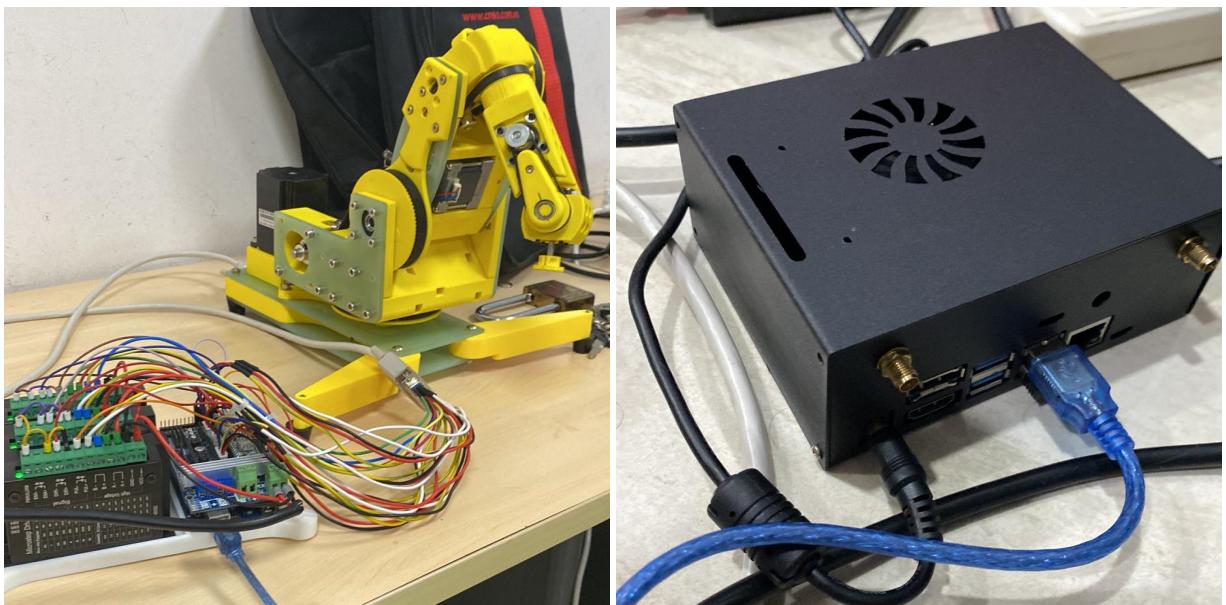
Hình 3.46: Máy trạng thái cho các mode

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.

3.5 Hình ảnh toàn bộ hệ thống



Hình 3.47: 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 3.48: Hệ thống điều khiển cánh tay và trung tâm xử lý Jetson Nano

CHƯƠNG 4

ĐÁNH GIÁ KẾT QUẢ

Nhìn chung, cánh tay robot hoạt động rất tốt và nhóm hài lòng với hiệu suất hiện tại của cánh tay robot. Tuy nhiên, bên cạnh đó vẫn còn một số vấn đề còn tồn đọng. Chương này sẽ làm rõ các vấn đề này và đưa ra các đề xuất để cải thiện cánh tay robot hơn nữa.

4.1 Kết quả đạt được

Dồ án đã hiện thực được mục tiêu đề ra ban đầu, cụ thể là hiện thực được **motor driver trên nền tảng ROS sử dụng mạch nhúng Jetson**:

1. Motor driver đã được thiết kế và triển khai thành công trên mạch nhúng Jetson và tích hợp vào hệ thống ROS.
2. Driver có thể điều khiển cả 6 trục động cơ của cánh tay, bao gồm điều khiển riêng lẻ từng động cơ và điều khiển đồng thời nhiều động cơ.
3. Driver đã có thể được sử dụng để áp dụng cho tay cầm gamepad để điều khiển robot.
4. Chức năng điều khiển bằng tay cầm tương đối chính xác, thông qua mắt thường có thể thấy Robot có thể bắt kịp được những tín hiệu điều khiển bằng tay cầm và di chuyển tương đối mượt mà.
5. Yếu tố động học robot đã được áp dụng thành công khi robot có thể tự tìm đường về vị trí home sau nhiều cử động liên tiếp.
6. Hệ thống ROS có thể đảm nhận được đồng thời việc điều khiển robot và thu thập hình ảnh từ camera cũng như xử lý hình ảnh với Model YOLO để nhận diện vật thể.

4.2 Những hạn chế

Bên cạnh những kết quả khả quan đạt được, đồ án vẫn còn tồn tại những hạn chế cần khắc phục, cụ thể:

1. Hiện tại tiến độ của nhóm chỉ mới xây dựng được chế độ Manual, dù module nhận diện vật thể đã có nhưng nhóm chưa kịp tích hợp được chế độ Auto - tự nhận diện và gấp vật thể cho cánh tay.
2. Chưa xây dựng được các bài test kiểm tra hiệu suất của Robot, bao gồm yếu tố tốc độ và sự chính xác, đặc biệt với cơ chế của stepper motor có thể xảy ra tình trạng trượt bước khi có áp dụng tải.
3. Chưa có cơ chế bảo vệ cánh tay khỏi các cử động vượt qua biên độ dao động, dẫn đến tình trạng kẹt khớp dễ gây hỏng hóc.
4. Chưa có bộ tản nhiệt, cánh tay có dấu hiệu bị nóng khi sử dụng trong thời gian dài.
5. Tốc độ cử động của robot đang được đặt cố định và còn khá chậm để đảm bảo độ an toàn cho cánh tay.
6. Chưa có cơ chế feedback để đảm bảo độ chính xác của robot.

4.3 So sánh với các Robot trên thị trường

Hiện nay trên thị trường có rất nhiều cánh tay robot khác nhau với nhiều loại giá thành khác nhau. Một trong số đó có thể kể đến Hiwonder với dòng robot JetMax [20] nổi tiếng.

Bên cạnh còn có một số dự án cũng liên quan trong thực tế như:

- **Thu hoạch trái cây trực tiếp**

(Nguồn: https://www.researchgate.net/figure/SSD-learning-parameters_tb13_336976022)

Robot nhận dạng và di chuyển tới vị trí quả chín trên cây để hái. Ý tưởng này nhằm thay thế cho con người trong việc thu gom trái cây. Tuy nhiên, số lượng trái cây được tiêu thụ trên thế giới là rất lớn, các trang trại trồng rau rất nhiều và đa dạng các loại cây với nhiều hình dạng và kích thước khác nhau. Việc ứng dụng ý tưởng này để hái trái cây chỉ có thể diễn ra trong một phạm vi nhỏ, khó có thể ứng dụng thực tiễn cao.

- **Đóng gói hàng hóa trên băng chuyền**

Đây thường là hệ thống tự động hóa toàn phần, có nghĩa là Robot sẽ tự động làm một công việc theo chu trình, băng chuyền tự động di chuyển và dừng lại theo chu trình, các thao tác luân phiên xen kẽ đều tự động. Điều này thường xảy ra khi sản phẩm đầu cuối đã được xác định trước các thao tác cần hiện thực của hệ thống, không có thay đổi trong quá trình vận hành.

So sánh với các sản phẩm thương mại và đề tài kể trên, robot của nhóm hoàn toàn có thể phát triển thêm để đáp ứng nhiều ứng dụng phức tạp hơn, đặc biệt khi chức năng điều khiển bằng gamepad đã được hiện thực tương đối hoàn chỉnh.



Hình 4.1: Hình ảnh robot 6DOF trên thị trường

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.
- 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 trục đơn và cử động xoay đồng thời nhiều trục.
- 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 phần động học vẫn chưa được tối ưu, do đó cử động của robot vẫn còn khá chậm.
- Về cơ khí, cánh tay chưa được trang bị các linh kiện có tính kiểm soát như cảm biến dòng nên đôi khi vẫn xảy ra tình trạng kẹt khớp.
- Phần 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, phần mềm,... 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 phần cơ khí, cải thiện hệ thống dây cho gọn gàng hơn và lắp đặt thêm cảm biến dòng để bảo vệ cánh tay khỏi tình trạng kẹt khớp.
- Cải thiện phần động học để gia tăng độ chính xác và tốc độ cho cánh tay.
- Thiết kế hoàn chỉnh mô hình kẹp gấp để có thể dễ dàng gấp được vật thể từ nhiều tư thế khác nhau.
- 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.
- Tích hợp thêm chế độ Auto để cánh tay có thể tự nhận diện và phân loại vật thể.

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.
- [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.
- [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.
- [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.
- [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] “Stepper motor wire color and coil pairs.” (), [Online]. Available: <https://3ddistributed.com>.
- [20] 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>.
- [21] Geeksforgeeks. “Universal asynchronous receiver transmitter (uart) protocol.” (), [Online]. Available: <https://www.geeksforgeeks.org/universal-asynchronous-receiver-transmitter-uart-protocol/>.
- [22] Wikipedia. “Universal asynchronous receiver-transmitter.” (), [Online]. Available: https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter.
- [23] 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/>.

- [24] C. Basics. “Basics of uart communication.” (), [Online]. Available: <https://www.circuitbasics.com/basics-uart-communication/>.
- [25] ScienceDirect. “Forward kinematics.” (), [Online]. Available: <https://www.sciencedirect.com/topics/engineering/forward-kinematics>.
- [26] G. Jocher and A. Exel. “Ultralytics yolov8 docs.” (), [Online]. Available: <https://docs.ultralytics.com/>.

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