BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TPHCM

KHOA CÔNG NGHỆ CƠ KHÍ

BÀI BÁO CÁO KẾT THÚC MÔN

MÔN : THỰC HÀNH TỰ ĐỘNG HÓA ROBOT

Giảng viên hướng dẫn : Lương Quốc Việt

Sinh viên thực hiên: Nhóm 1

1. Ngô Giang Nam	**2025230134**
2. Hồ Ngọc Thành	**2025230213**
3. Lê Minh Tân	**2025230188**
4. Nguyễn Thành Nam	**2025230129**

Tp. Hồ Chí Minh tháng 8 năm 2025

LỜI MỞ ĐẦU

Trong bối cảnh cuộc Cách mạng công nghiệp 4.0, công nghệ robot và tự động hóa đóng vai trò ngày càng quan trọng trong sản xuất và đời sống. Robot công nghiệp không chỉ giúp nâng cao năng suất, giảm chi phí, mà còn đảm bảo chất lượng sản phẩm và an toàn lao động.

Môn học **Thực hành Tự động hóa Robot** được xây dựng nhằm giúp sinh viên làm quen với các thiết bị, phần mềm và kỹ thuật điều khiển robot hiện đại. Thông qua các bài thực hành, sinh viên được tiếp cận quy trình vận hành, lập trình và tối ưu hóa chuyển động của robot, đồng thời rèn luyện kỹ năng phân tích, xử lý sự cố và phối hợp nhóm.

Việc nắm vững kiến thức và kỹ năng trong môn học này sẽ là nền tảng quan trọng để sinh viên có thể áp dụng vào thực tiễn sản xuất, đáp ứng nhu cầu nhân lực chất lượng cao trong lĩnh vực cơ điện tử, tự động hóa và công nghiệp thông minh.

CHƯƠNG 1: TỔNG QUAN

1. Mục tiêu buổi thực hành

1.1 Nắm vững kiến thức cơ bản về cấu tạo và nguyên lý hoạt động của robot công nghiệp

- Tìm hiểu các bộ phận chính của robot: cơ cấu chấp hành, động cơ truyền động, hệ thống cảm biến, và bộ điều khiển.
- Hiểu vai trò của từng khớp (Joint) và các bậc tự do (DOF) trong quá trình chuyển động của robot.

1.2 Làm quen và sử dụng thành thạo phần mềm điều khiển robot

- Khởi tạo dự án mới, lựa chọn mô hình robot phù hợp và thiết lập môi trường làm việc.
- Thực hiện các thao tác điều khiển cơ bản: di chuyển bằng tay (Jog), đặt điểm (Point), và chạy chương trình thử nghiệm.

1.3 Thực hành lập trình và tối ưu hóa quỹ đạo chuyển động

- Lập trình các thao tác cơ bản như gắp – thả vật, di chuyển theo trục tọa độ, và chạy theo quỹ đạo định sẵn.
- Điều chỉnh tốc độ, thời gian dừng (Delay) và thứ tự thao tác để đảm bảo độ chính xác và hiệu quả.
- Rèn luyện kỹ năng xử lý sự cố khi robot vận hành, đồng thời đánh giá và cải thiện chương trình điều khiển.

1. Cơ sở lý thuyết – Robot 6 bậc tự do

- Robot 6 bậc tự do (6 DoF – Degrees of Freedom) là loại cánh tay robot có khả năng thực hiện chuyển động trong không gian 3D với đầy đủ các phép tịnh tiến và quay. Sáu bậc tự do này bao gồm:
- **Xoay quanh trục đứng** (Base rotation – Joint 1)
- **Nâng/hạ cánh tay** (Shoulder pitch – Joint 2)
- **Gập/duỗi cánh tay** (Elbow pitch – Joint 3)
- **Xoay cổ tay** (Wrist roll – Joint 4)
- **Gập/duỗi cổ tay** (Wrist pitch – Joint 5)
- **Xoay cổ tay quanh trục** (Wrist yaw – Joint 6)
- Nhờ cấu hình này, robot có thể đặt công cụ hoặc gắp vật tại **bất kỳ vị trí và hướng mong muốn** trong vùng làm việc của nó.
- **Các thành phần chính:**
- **Khung và khớp cơ khí:** Được gia công từ nhôm hoặc thép nhẹ, tạo sự cứng vững.

- **Động cơ servo:** Cung cấp mô-men xoắn và khả năng định vị góc chính xác.
- **Bộ điều khiển trung tâm:** Mạch điều khiển nhận lệnh từ máy tính hoặc bộ lập trình, tính toán quỹ đạo và điều khiển động cơ.
- **Nguồn cấp:** Cung cấp điện cho toàn bộ hệ thống.
- **Ứng dụng:**
- **Đào tạo & nghiên cứu** về robot công nghiệp.
- **Thí nghiệm lập trình quỹ đạo** và điều khiển động học – động lực học.
- **Mô phỏng các thao tác công nghiệp** như hàn, lắp ráp, bốc xếp.

3. Dụng cụ và phần mềm sử dụng

3.1. Dụng cụ

1. Khung và giá đỡ

- Cấu tạo bằng kim loại (thường là hợp kim nhôm) để đảm bảo độ cứng và nhẹ.
- Các khớp nối được gia công sẵn, liên kết bằng ốc vít.

2. Các động cơ servo

- Loại servo kỹ thuật số (digital servo) có mô-men xoắn cao, điều khiển từng bậc tự do.
- Bạn có thể thấy rõ một số động cơ gắn ở các khớp, dây tín hiệu màu cam–đỏ–nâu.

3. Bo mạch điều khiển

- Nhìn giống board điều khiển servo hoặc vi điều khiển (ví dụ: Arduino, hoặc board chuyên dụng như SSC-32U).
- Có các chân cắm servo được bố trí thành hàng, kèm cổng kết nối USB mini/micro để lập trình.

4. Cáp tín hiệu và dây nguồn

- Dây servo được bọc ống gen bảo vệ màu đen.
- Cáp USB dùng để kết nối máy tính với bo mạch.

5. Nguồn điện

- Hộp màu đen phía dưới có thể là pin Li-Po hoặc bộ nguồn DC để cung cấp điện cho servo.

6. Các khớp nối và trục

- Có trục quay, ổ trục và tay đòn liên kết giữa các bậc.

3.2. Phần mềm sử dụng

Visual Studio Code là một trình soạn thảo mã nguồn được phát triển bởi Microsoft dành cho Windows, Linux và macOS. Nó hỗ trợ chức năng debug, đi kèm với Git, có chức năng nổi bật cú pháp (syntax highlighting), tự hoàn thành mã thông minh, snippets, và cải tiến mã nguồn. Nó cũng cho phép tùy chỉnh, do đó, người dùng có thể thay đổi theme, phím tắt, và các tùy chọn khác. Nó miễn phí và là phần mềm mã nguồn mở theo giấy phép MIT, mặc dù

bản phát hành của Microsoft là theo giấy phép phần mềm miễn phí. Visual Studio Code được dựa trên Electron, một nền tảng được sử dụng để triển khai các ứng dụng Node.js máy tính cá nhân chạy trên động cơ bố trí Blink. Mặc dù nó sử dụng nền tảng Electron nhưng phần mềm này không phải là một bản khác của Atom, nó thực ra được dựa trên trình biên tập của Visual Studio Online (tên mã là "Monaco"). Visual Studio Code là một trình biên tập mã. Nó hỗ trợ nhiều ngôn ngữ và chức năng tùy vào ngôn ngữ sử dụng theo như trong bảng sau. Nhiều chức năng của Visual Studio Code không hiển thị ra trong các menu tùy chọn hay giao diện người dùng. Thay vào đó, chúng được gọi thông qua khung nhập lệnh hoặc qua một tập tin .json (ví dụ như tập tin tùy chỉnh của người dùng). Khung nhập lệnh là một giao diện theo dòng lệnh. Tuy nhiên, nó biến mất khi người dùng nhấp bất cứ nơi nào khác, hoặc nhấn tổ hợp phím để tương tác với một cái gì đó ở bên ngoài đó. Tương tự như vậy với những dòng lệnh tốn nhiều thời gian để xử lý. Khi thực hiện những điều trên thì quá trình xử lý dòng lệnh đó sẽ bị hủy.

Arduino là nền tảng vi mạch thiết kế mở phần cứng (Open-source hardware) và phần mềm (Open-source software). Phần

cứng Arduino là những bộ vi điều khiển bo mạch đơn (Single-board microcontroller) được tạo ra tại thị trấn Ivrea ở Ý, nhằm xây dựng các ứng dụng tương tác với nhau hoặc với môi trường được thuận lợi hơn. Phần cứng bao gồm một board mạch nguồn mở được thiết kế trên nền tảng vi xử lý AVR Atmel 8bit, hoặc ARM Atmel 32-bit. Những model hiện tại được trang bị gồm 1 cổng giao tiếp USB, 6 chân đầu vào analog, 14 chân I/O kỹ thuật số tương thích với nhiều board mở rộng khác nhau. Được giới thiệu vào năm 2005, những nhà thiết kế của Arduino cố gắng mang đến một phương thức dễ dàng, không tốn kém cho những người yêu thích, sinh viên và giới chuyên nghiệp để tạo ra những thiết bị có khả năng tương tác với môi trường thông qua các cảm biến và các cơ cấu chấp hành. Những ví dụ phổ biến cho những người yêu thích mới bắt đầu bao gồm các robot đơn giản, điều khiển nhiệt độ và phát hiện chuyển động. Đi cùng với nó là một môi trường phát triển tích hợp (IDE) chạy trên các máy tính cá nhân thông thường và cho phép người dùng viết các chương trình cho mạch bằng ngôn ngữ Arduino, một ngôn ngữ riêng được phát triển dựa trên C/C++.

Git (/gɪt/, đọc là "Ghít") là phần mềm quản lý mã nguồn phân tán được phát triển bởi Linus Torvalds vào năm 2005, ban đầu dành cho việc phát triển nhân Linux. Hiện nay, Git trở thành một trong các phần mềm quản lý mã nguồn phổ biến nhất. Git là phần mềm mã nguồn mở được phân phối theo giấy phép công cộng GPL2.

Khái niệm cơ bản trong Git

- **Repository**

- Repository (nhà kho) hay được gọi tắt là Repo đơn giản là nơi chứa/cơ sở dữ liệu (database) tất cả những thông tin cần thiết để duy trì và quản lý các sửa đổi và lịch sử của dự án.
- Trong Repo có 2 cấu trúc dữ liệu chính là Object Store và Index. Tất cả dữ liệu của Repo đều được chứa trong thư mục bạn đang làm việc dưới dạng folder ẩn có tên là.git (không có phần tên trước dấu chấm).

- **Object store**

- Object store là trái tim của Git, nó chứa dữ liệu nguyên gốc (original data files), các file log ghi chép quá trình sửa đổi, tên người tạo file, ngày tháng và các thông tin khác. Git có bốn loại object là: Blobs, Trees, Commits, Tags

- **Blobs**: là file nhị phân có thể chứa được mọi loại dữ liệu bất kể là dữ liệu của chương trình gì.

- **Trees**: lớp đại diện cho thông tin thư mục như thông tin định danh của blob, đường dẫn, chứa một ít metadata chứa thông tin cấu trúc và các thư mục nhỏ có trong thư mục đó.

- **Commits**: Chứa metadata có thông tin về mọi thứ như tên tác giả, người tải lên (commiter), ngày tải lên, thông tin log...
- **Tags**: đánh dấu cho dễ đọc. Thay vì một cái tên dài như là 9da581d910c9c4ac93557ca4859e767f5caf5169, chúng ta có thể tên tag là Ver-1.0- Alpha. Dễ nhớ và dễ sử dụng hơn.
- **Index**

Index là file nhị phân động và tạm thời miêu tả cấu trúc thư mục của toàn bộ Repo và trạng thái của dự án được thể hiện thông qua commit và tree tại một thời điểm nào đó trong lịch sử phát triển. Git là một hệ thống truy tìm nội dung (content tracking system).

Index không chứa nội dung file mà chỉ dùng để truy tìm (track) những thứ mà bạn muốn commit.

GitHub là một dịch vụ cung cấp kho lưu trữ mã nguồn Git dựa trên nền web cho các dự án phát triển phần mềm. GitHub cung cấp cả phiên bản trả tiền lẫn miễn phí cho các tài khoản. Các dự án mã nguồn mở sẽ được cung cấp kho lưu trữ miễn phí. Tính đến tháng 4 năm 2016, GitHub có hơn 14 triệu người sử dụng với hơn

35 triệu kho mã nguồn, làm cho nó trở thành máy chủ chứa mã nguồn lớn trên thế giới.

Github đã trở thành một yếu tố có sức ảnh hưởng trong cộng đồng phát triển mã nguồn mở. Thậm chí nhiều nhà phát triển đã bắt đầu xem nó là một sự thay thế cho sơ yếu lý lịch và một số nhà tuyển dụng yêu cầu các ứng viên cung cấp một liên kết đến tài khoản Github để đánh giá ứng viên.

Qt (cách phát âm chính thức tương tự như từ cute /ˈkjuːt/ trong tiếng Anh một kiểu phát âm phổ biến khác là Q.T. /ˌkjuːˈtiː/) là một khung ứng dụng đa nền tảng và bộ công cụ tiện ích để tạo giao diện người dùng đồ họa cổ điển và nhúng, và các ứng dụng chạy trên nhiều nền tảng phần mềm và phần cứng khác nhau hoặc ít thay đổi trong codebase cơ bản, trong khi vẫn là một ứng dụng gốc với khả năng và tốc độ cục bộ. Qt hiện đang được phát triển bởi cả The Qt Company, một công ty niêm yết công khai, và Qt Project dưới quản lý mã nguồn mở, liên quan đến các nhà phát triển cá nhân và các công ty làm việc để thúc đẩy Qt. Qt có sẵn theo cả giấy phép thương mại nguồn mở.và giấy phép GPL 2.0, GPL 3.0 và LGPL 3.0 nguồn mở.

Python (/ˈpaɪθɑːn/) là ngôn ngữ lập trình bậc cao đa năng. Triết lý thiết kế của nó nhấn mạnh khả năng đọc mã bằng cách sử dụng thụt lề đáng kể. Python có kiểu động và thu gom rác. Ngôn ngữ này hỗ trợ nhiều mô hình lập trình, bao gồm lập trình cấu trúc (đặc biệt là lập trình thủ tục), lập trình hướng đối tượng và lập trình chức năng. Nó thường được mô tả là ngôn ngữ "bao gồm pin" do có thư viện tiêu chuẩn toàn diện. Guido van Rossum bắt đầu nghiên cứu Python vào cuối những năm 1980 với tư cách là ngôn ngữ kế thừa cho ngôn ngữ lập trình ABC và phát hành nó lần đầu tiên vào năm 1991 với tên gọi Python 0.9.0. Python 2.0 được ra mắt vào năm 2000. Python 3.0 được ra mắt vào năm 2008, là bản sửa đổi lớn không hoàn toàn tương thích ngược với các phiên bản trước đó. Python 2.7.18, được phát hành vào năm 2020, là bản phát hành cuối cùng của Python 2. Python liên tục được xếp hạng là một trong những ngôn ngữ lập trình phổ biến nhất và được sử dụng rộng rãi trong cộng đồng học máy.

CH340

Trong các ứng dụng nhúng và truyền thông dữ liệu giữa máy tính với vi điều khiển, việc chuyển đổi tín hiệu giữa chuẩn USB và UART (hoặc các chuẩn nối tiếp khác) là rất phổ biến. IC **CH340** là một trong những giải pháp được sử dụng rộng rãi nhờ chi phí thấp, dễ tích hợp và hiệu năng ổn định. Đây là sản phẩm của công ty **WCH (Nanjing QinHeng Electronics)**, thường thấy trong các module nạp và giao tiếp như mạch Arduino clone, module USB–TTL, hoặc thiết bị đo lường dữ liệu.

CH340 là **USB-to-Serial Bridge IC** – một vi mạch chuyển đổi giao tiếp USB sang UART, hỗ trợ việc truyền và nhận dữ liệu nối tiếp giữa máy tính và các thiết bị nhúng. Ngoài UART, một số phiên bản CH340 còn hỗ trợ giao thức **I²C** và **SPI**.

Các phiên bản phổ biến

- **CH340G**: Phiên bản phổ biến nhất, cần thạch anh ngoài 12 MHz.
- **CH340C**: Tích hợp sẵn thạch anh bên trong, không cần linh kiện ngoài.
- **CH340T**: Dùng trong thiết bị thương mại, dạng SMD.

- **CH340N**: Kích thước nhỏ, phù hợp thiết bị gọn nhẹ.
- **CH340B**: Hỗ trợ nhiều giao thức ngoài UART.

Đặc điểm kỹ thuật

- **Điện áp hoạt động**: 3.3V – 5V (tùy phiên bản).
- **Tốc độ truyền dữ liệu UART**: lên tới 2 Mbps.
- **Chuẩn USB**: Hỗ trợ USB 2.0 Full-Speed.
- **Tích hợp**: Mạch quản lý dòng USB, bộ tạo xung clock (trong một số phiên bản).
- **Chân giao tiếp**:
- **USB**: D+ và D−.
- **UART**: TXD, RXD, RTS, CTS, DTR, DSR, DCD, RI.
- **Hỗ trợ driver**: Windows, macOS, Linux.

Ứng dụng thực tế

CH340 thường được sử dụng trong:

- **Module nạp Arduino clone**: Giao tiếp USB–UART để lập trình và giám sát.
- **Module USB–TTL**: Dùng để debug hoặc cấu hình thiết bị qua cổng nối tiếp.
- **Thiết bị đo lường dữ liệu**: Truyền dữ liệu cảm biến từ vi điều khiển lên PC.
- **Hệ thống IoT**: Cầu nối giữa máy tính và module vi điều khiển.

Ưu điểm và nhược điểm

Ưu điểm:

- Giá thành rẻ so với FT232RL hoặc CP2102.
- Dễ tìm mua và thay thế.
- Hỗ trợ nhiều hệ điều hành với driver có sẵn.
- Tiêu thụ điện năng thấp.

Nhược điểm:

- Cần cài driver thủ công trên một số máy tính (đặc biệt là Windows cũ).
- Không tích hợp các tính năng cao cấp như một số IC đắt tiền hơn.
- Tốc độ tối đa 2 Mbps thấp hơn một số chip chuyên dụng.

4. Cấu trúc giao diện sau chuyển đổi

4.1. Nhóm Position Control

Vùng điều khiển vị trí (Position Control) được bố trí ở phía trên giao diện, bao gồm:

Tiêu đề POSITION CONTROL (label_9) hiển thị bằng chữ in đậm, cỡ chữ lớn.

- Các nút điều khiển tọa độ:
- btn_pos_z_plus (↑ Z+)
- btn_pos_z_minus (↓ Z-)
- btn_pos_x_plus (X+ ↙)
- btn_pos_x_minus (X- ↗)

- btn_pos_home (O) ở trung tâm, dùng để đưa hệ trục về gốc.

Các nút được thiết kế màu nền xanh lá (#4CAF50) và chữ trắng để đảm bảo tính nhất quán.

4.2. Nhóm Joint Control

Phía dưới vùng Position Control là vùng điều khiển khớp (Joint Control), gồm:

Tiêu đề JOIN CONTROL (label_3).

- Sáu cụm điều khiển Joint 1 đến Joint 6, mỗi cụm gồm:
- Label tên khớp (Join 1 … Join 6).
- QLineEdit hiển thị giá trị góc hiện tại (chế độ read-only).
- Nút tăng góc (INC ↑) và giảm góc (DES ↓).

Đáng chú ý, trong file .ui có một số tên biến cần lưu ý khi ánh xạ với servo:

- line_j3_val_3 là giá trị Joint 5.
- line_j4_val_5 là giá trị Joint 6.

4.3. Nút chức năng chung

Bên dưới nhóm Joint Control là hai nút:

- btn_home – HOME: Đưa toàn bộ các Joint về vị trí mặc định.
- btn_setting – SETTING: Mở cửa sổ cấu hình.

4.4. Bảng Homogeneous Transformation Matrices

Ở cuối giao diện, phần label_2 hiển thị tiêu đề HOMOGENEOUS TRANSFORMATION MATRICES kèm theo bảng table_tmatrix

(4×4) để hiển thị ma trận biến đổi đồng nhất của robot. Bảng không hiển thị tiêu đề hàng và cột, chỉ hiển thị dữ liệu.

4. Ánh xạ từ code Python sang Qt Designer

Quá trình ánh xạ được thực hiện theo nguyên tắc:

- Giữ nguyên **tên biến điều khiển** trong .ui trùng hoặc gần trùng với biến trong code Python cũ, giúp tái sử dụng logic mà không cần thay đổi quá nhiều.
- Với các nút bấm, ánh xạ trực tiếp từ phương thức xử lý sự kiện cũ sang signal .clicked của QPushButton.
- Với QLineEdit hiển thị giá trị góc, ánh xạ vào biến trong code để cập nhật giá trị khi servo quay.
- Đảm bảo rằng các phần tử giao diện liên quan tới một joint đều nằm trong cùng một layout ngang (QHBoxLayout) để dễ quản lý.

5. Lợi ích đạt được

- Sau khi chuyển đổi sang Qt Designer:
- **Dễ bảo trì**: Chỉ cần mở file .ui để thay đổi bố cục hoặc thêm thành phần.
- **Tiết kiệm thời gian phát triển**: Không phải viết lại nhiều đoạn code tạo widget bằng tay.
- **Tăng tính trực quan**: Người thiết kế có thể nhìn thấy và điều chỉnh vị trí nút, label ngay trên màn hình thiết kế.
- **Tách biệt rõ ràng giữa giao diện và logic**: Logic điều khiển servo, xử lý ma trận… nằm riêng trong file .py.

CHƯƠNG 2: NGUYÊN LÝ HOẠT ĐỘNG

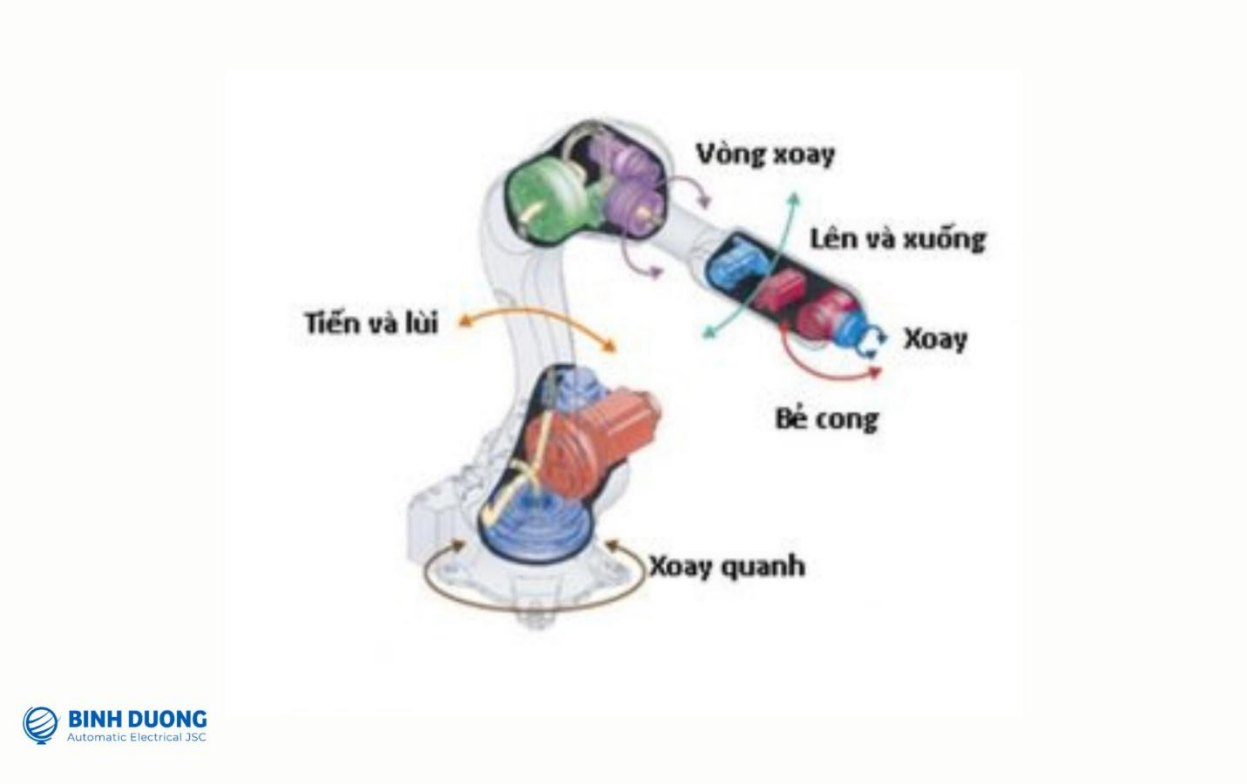
2. Quy trình thực hành

2.1. Nguyên lý chuyển động

Cánh tay robot hoạt động theo nguyên lý chuyển động tịnh tiến và xoay, mô phỏng chuyển động của tay người. Mỗi khớp của robot tương ứng với một bậc tự do. Số bậc tự do càng nhiều thì khả năng thao tác, tầm với và độ linh hoạt của robot càng cao.

Các chuyển động này bao gồm:

- Xoay tại gốc: Hỗ trợ cánh tay quay theo phương ngang quanh trục đứng.
- Gập duỗi: Cho phép robot vươn dài hoặc thu ngắn, tương tự như vai và khuỷu tay người.
- Xoay cổ tay: Tăng khả năng linh hoạt trong thao tác, đặc biệt quan trọng trong ứng dụng yêu cầu độ chính xác cao.



Nguyên lý chuyển động

2.2. Các bước chuẩn bị

Bước 1: Chuẩn bị

- **Kiểm tra thiết bị:**
- Đảm bảo robot, bộ điều khiển, nguồn điện, dây kết nối và máy tính/laptop ở tình trạng tốt.
- Kiểm tra các servo hoạt động bình thường, không bị kẹt hoặc hỏng bánh răng.
- **Bật nguồn hệ thống** và mở phần mềm điều khiển robot trên máy tính.
- **Cài đặt ban đầu:**
- Đưa robot về **vị trí Home** (các khớp ở góc 0° hoặc vị trí gốc).

- Cấu hình thông số kết nối (cổng COM, tốc độ Baud, chế độ điều khiển).

Bước 2: Lập trình và điều khiển cơ bản

- **Điều khiển từng khớp:**
- Sử dụng phần mềm để di chuyển từng khớp riêng lẻ, ghi nhận góc quay và tốc độ.
- **Điều khiển phối hợp nhiều khớp:**
- Thực hiện các chuyển động tuyến tính (linear motion) và quỹ đạo cong (curved motion).
- **Lưu chương trình thao tác:**
- Ghi lại các bước điều khiển thành chương trình để robot thực hiện tự động.

Bước 3: Thực hành bài tập

- **Bài 1 – Gắp và di chuyển vật:**
- Đưa tay gắp đến vị trí vật, kẹp vật, di chuyển và đặt vào vị trí mới.
- **Bài 2 – Vẽ quỹ đạo:**
- Lập trình robot di chuyển theo hình vuông, tam giác hoặc đường tròn.
- **Bài 3 – Tích hợp cảm biến (nếu có):**
    - Sử dụng cảm biến để phát hiện vật và tự động thực hiện thao tác gắp.

Bước 4: Kết thúc

Dừng robot và đưa về vị trí Home.

Tắt nguồn bộ điều khiển và ngắt kết nối.

Ghi chép kết quả:

Lưu chương trình, ghi lại thời gian hoàn thành, độ chính xác, các sự cố gặp phải.

Đánh giá:

So sánh kết quả thực tế với mục tiêu đề ra.

Đề xuất cải tiến hoặc rút kinh nghiệm.

CHƯƠNG 3: THIẾT KẾ GIAO DIỆN ĐIỀU KHIỂN

3.1.Định Nghĩa Giao Diện (UI) ở Dạng XML, Tạo Bởi QT DESIGNER,

a.ROBOTUI.UI:

File thiết kế giao diện gốc trong QtDesigner:

B1: Mở Qtdesigner, (giao diện) → tạo new Main Window.

B2: Tạo Label (hiển thị văn bản phân vùng)

- Position Control:
- Join Control (Join 1-6):
- Bảng ma trận (HOMOGENEOUS TRANSFORMATION MATRICES):

B3: Tạo Push Button →Đặt tên (Object Name):

- INC 1-6 tương ứng với 6 servo: khi kích hoạt sẽ tăng góc quay
- DES 1-6 tương ứng với 6 servo: khi kích hoạt sẽ giảm góc quay

btn_pos_z_plus:

btn_pos_z_minus:

btn_pos_x_plus:

btn_pos_x_minus:

➔ Điều chỉnh góc servo

- B4: Tạo Line Edit →Đặt tên (Object Name): Line_j1_val
- Line_j2_val
- Line_j3_val
- Line_j4_val
- Line_j3_val_3
- Line_j4_val_5

➔ Hiển thị góc của các servo

B5: Tạo Table Widget → Bảng ma trận

3.2.Chuyển Cấu Trúc UI Trong UI Thành Mã PYTHON

a.ROBOTUI.PY:

File Python tạo từ robotui.ui bằng lệnh Pyuic6, chứa class UI_MainWindow để vẽ giao diện

B1: Cài pyqt6, trong terminal/cmd của VSCode, nhập lệnh:

“pip install pyqt6”

“pip install pyqt6-tools”

“pip install pyuic6”

➔Cài các cài đặt thư viện (package) Python thông qua trình quản lý gói pip.

B2: Trong terminal/cmd của VSCode, nhập lệnh:

“pyuic 6 -x robotui.ui -o robotui.py”

➔Chuyển đổi file ui thành py để khởi chạy thử giao diện tĩnh trên VSCode

3.3.IMPORT UI_MAINWINDOW TỪ ROBOTUI.PY ĐỂ HIỂN THỊ GIAO DIỆN.

.MAIN.PY:

- **Chức năng:** Ứng dụng PyQt6 nhập từ robotui.py, giao tiếp UART với vi điều khiển để điều khiển tay máy 6 bậc tự do. Cho phép điều khiển từng khớp (joint space) và dịch chuyển đầu công tác (task space) theo trục X, Z, hiển thị ma trận biến đổi Denavit–Hartenberg và hỗ trợ điều chỉnh thông số vận hành.

1) Tổng quan:

- Giao diện người dùng từ Ui_MainWindow (robotui.py sinh ra từ robotui.ui).
- Điều khiển 6 servo (J1–J6) qua board servo 16 kênh.
- Giao tiếp Serial (mặc định COM4, baud 115200).
- Hai chế độ điều khiển:
- Joint Control: thay đổi góc từng khớp.

- Cartesian Control: dịch chuyển vị trí theo X/Z bằng thuật toán IK (Damped Least Squares – DLS).
- Nút HOME (cả btn_home và btn_pos_home) để về vị trí trung tâm (90° mỗi khớp).
- Hộp thoại Settings để chỉnh STEP-DIS, STEP-ROT, SPEED-LEVEL.

2) Cấu trúc chương trình:

ARMROBOTPROJECT2025/

├─ main.py # Chương trình chính điều khiển robot

├─ robotui.py # File Python sinh ra từ robotui.ui (UI code)

├─ robotui.ui # File thiết kế giao diện (Qt Designer)

└─ __pycache__/ # Thư mục cache Python

└─ robotui.cpython-*.pyc

3) Thành phần chính của main.py:

RobotArmController(QMainWindow)

├─ Biến trạng thái: servo_angles, step_rotation, step_cart, speed_level

├─ Điều khiển Joint: move_servo() + send_servo_command()

├─ Điều khiển Cartesian: move_cartesian()

├─ HOME: reset_all_servos()

├─ FK/DH: dh_transform(), forward_kinematics()

├─ IK/DLS: numeric_jacobian(), damped_least_squares()

├─ Hiển thị: show_matrix()

└─ Hộp thoại cài đặt: open_settings_dialog()

4) Giao diện & tên biến:

Joint Control

inc1/des1 → J1 (line_j1_val)

inc2/des2 → J2 (line_j2_val)

inc3/des3 → J3 (line_j3_val)

inc4/des4 → J4 (line_j4_val)

inc5/des5 → J5 (line_j3_val_3)

inc6/des6 → J6 (line_j4_val_5)

Cartesian Control

btn_pos_z_plus, btn_pos_z_minus: dịch trục Z.

btn_pos_x_plus, btn_pos_x_minus: dịch trục X.

HOME & Settings

btn_home hoặc btn_pos_home: reset tất cả khớp về 90°.

btn_setting: mở hộp thoại điều chỉnh.

5) Giao tiếp Serial:

Lệnh gửi: #<id>P<pulse>T200\r\n.

pulse = 500 + (2000 * angle / 180).

HOME gửi tuần tự từng khớp, trễ 50ms giữa các lệnh.

6) Động học thuận (FK):

Bảng tham số DH gồm 6 hàng.Khi góc = 90° toàn bộ, vị trí được đặt cố định (0.274, 0.0, 0.187).

7) Nghiệm ngược (IK):

Jacobian tính bằng sai phân số.

DLS giải $\Delta\theta = J^T(JJ^T + \lambda^2 I)^{-1}\Delta p$.

Lặp tối đa 5 lần để đạt mục tiêu.

8) An toàn & giới hạn:

Giới hạn góc: 0–180°.

T200 ms cố định, có thể chỉnh để mượt hơn.

9) Quy trình vận hành:

Chỉnh SERIAL_PORT.
Chạy chương trình.
HOME để về vị trí chuẩn.
Thử Joint Control và Cartesian Control.
Chỉnh thông số trong Settings.

10) Mở rộng:

Điều khiển trục Y, định hướng.
Giới hạn vùng làm việc.

Bù lệch servo.