BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC PHENIKAA

BÁO CÁO TỔNG KẾT

TÊN ĐỀ TÀI:

Nghiên cứu thiết kế mô phỏng bàn tay người cho robot

Lĩnh vực: Cơ khí – Cơ điện tử

Chuyên ngành: Cơ điện tử

Nhóm sinh viên thực hiện	Giới tính	Mã sinh viên	Lớp
Lê Mạnh Trung	Nam	21013314	K15-KTCĐT1
Phạm Ngọc Việt	Nam	21012363	K15-KTCĐT2
Trần Trọng Tú	Nam	21011453	K15-KTCĐT1

Người hướng dẫn chính: PGS.TS. Vũ Lê Huy

Hà Nội, tháng 5 năm 2023

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC PHENIKAA

BÁO CÁO TỔNG KẾT

TÊN ĐỀ TÀI:

Nghiên cứu thiết kế mô phỏng bàn tay người cho robot

Lĩnh vực: Cơ khí – Cơ điện tử

Chuyên ngành: Cơ điện tử

Nhóm sinh viên thực hiện	Giới tính	Mã sinh viên	Lớp
Lê Mạnh Trung	Nam	21013314	K15-KTCĐT1
Phạm Ngọc Việt	Nam	21012363	K15-KTCĐT2
Trần Trọng Tú	Nam	21011453	K15-KTCĐT1

Người hướng dẫn chính: PGS.TS. Vũ Lê Huy

Hà Nội, tháng 5 năm 2023

Mục lục

Danh mục hình ảnh

Danh mục bảng biểu

Mở đầu

Hiện nay, việc robot hình dạng người xuất hiện đã thay đổi rất nhiều thứ trong cuộc sống của con người nói chung và các ngành công nghiệp nói riêng. Hầu hết mọi người đều chỉ nghĩ robot dạng hình người là những robot tích hợp trí thông minh nhân tạo, có thể giao tiếp, có ngoại hình và khả năng cử động tương tự con người, mà không biết rằng các nghiên cứu về lĩnh vực robot hình dạng người còn hướng đến việc mô phỏng hoặc chế tạo lại các bộ phận trên cơ thể (cánh tay, bàn tay, chân…) và các hoạt động (cầm, nắm) của con người.

Bàn tay robot 5 ngón là một trong những robot phổ biến nhất được sử dụng trong lĩnh vực y tế hoặc các thao tác cần mô phỏng chuyển động của bàn tay người sử dụng từ khoảng cách xa. Bàn tay robot sẽ được lập trình để có thể ghi nhận và thực hiện lại các cử chỉ từ bàn tay người sử dụng.

Tuy nhiên việc lập trình lệnh điều khiển bàn tay robot 5 ngón là điều không đơn giản, nó đòi hỏi sự chính xác và ổn định. Vì vậy việc lập sẵn 1 đoạn chương trình có sẵn sẽ giúp việc điều khiển một cách dễ dàng và tích kiệm thời gian, hơn thế nữa có thể áp dụng chương trình này đến tất cả cánh tay robot khác. Điều này giúp tiệt kiệm được thời gian và công sức. Việc học tập hay chế tạo một bàn tay robot có khả năng mô phỏng chính xác các cử chỉ của người dùng ở thời điểm hiện tại còn nhiều khó khăn về thời gian, tiền bạc, công nghệ. Để đơn giản hóa việc nghiên cứu cấu tạo, nguyên lý hoạt động, giải các bài toán động học điều khiển bàn tay robot hay phục vụ cho việc thử nghiệm chế tạo thử trước khi sản xuất thì việc sử dụng các phần mềm mô phỏng là rất cần thiết. Tuy nhiên các phần mềm để phục vụ chuyên biệt để mô phỏng bàn tay robot trong nghiên cứu, đào tạo thì còn rất hạn chế, đó cũng là lí do ta chọn đề tài mô phỏng bàn tay người cho robot.

Chương 1. Tổng quan

1.1. Đặt vấn đề

Từ thời cổ xưa, con người đã mong muốn tạo ra những vật giống mình để bắt chúng phụ vụ cho bản thân mình. Ví dụ như trong kho thần thoại Hy Lạp có chuyện người khổng lồ Promethe đúc ra người từ đất sét và truyền cho họ sự sống. Cho đến những năm 40 nhà văn viên tưởng người Nga, Issac Asimow, mô tả robot là một chiếc máy tự động, mang diện mạo của con người, được điều khiển bằng một hệ thần kinh khả trình Positron, do chính con người lập trình. Asimov cũng đặt tên cho ngành khoa học nghiên cứu về robot là Robotics, trong đó có 3 nguyên tắc cơ bản:

- Robot không được xúc phạm con người và không gây tổn hại cho con người.
- Hoạt động của robot phải tuân theo các quy tắc do con người đặt ra. Các quy tắc này không được vi phạm nguyên tắc thứ nhất.
- Một robot phải bảo vệ sự sống của mình, nhưng không được vi phạm 2 nguyên tắc trước.

Các nguyên tắc trên sau này trở thành nền tảng cho việc thiết kế robot [1].

Từ sự hư cấu của khoa học viễn tưởng, robot dần dần được giới kĩ thuật hình dung như một chiếc máy đặc biệt, được con người phỏng tác theo cấu tạo và hoạt động của chính mình, dùng để thay thế mình trong một số công việc nhất định. Để hoàn thành nhiệm vụ đó, robot cần có khả năng cảm nhận các thông số trạng thái của môi trường và tiến hành các loạt hoạt động tương tự con người. Khả năng hoạt động của robot được đảm bảo bởi hệ thống cơ khí, gồm cơ cấu vận động để đi lại và cơ cấu hành động để có thể làm việc. Việc thiết kế và chế tạo hệ thống này thuộc lĩnh vực khoa học về cơ cấu truyền động, chấp hành và vật liệu cơ khí.

Chức năng cảm nhận, gồm thu nhận tín hiệu về trạng thái môi trường và trạng thái của bản thân hệ thống, do các cảm biến (sensor) và các thiết bị liên quan thực hiện. Hệ thống này được gọi là hệ thống thu nhận và xử lý tín hiệu, hay đơn giản là hệ thống cảm biến.

Muốn phối hợp hoạt động của hai hệ thống trên, đảm bảo cho robot có thể tự mình tự điều chỉnh "hành vi" của mình và hoạt động theo đúng chức năng quy định trong điều kiện môi trường thay đổi, trong đó robot phải có hệ thống điều khiển. Xây dựng các hệ thống điều khiển thuộc phạm vi điện tử, kỹ thuật điều khiển và công nghệ thông tin.

Một cách đơn giản, Robotics được hiểu là một ngành khoa học, có nhiệm vụ nghiên cứu về thiết kế, chế tạo các robot và ứng dụng chúng trong các lĩnh vực hoạt động khác nhau của xã hội loài người, như nghiên cứu khoa học-kỹ thuật, kinh tế, quốc phòng và dân sinh. Từ hiểu biết sơ bộ về chức năng và kết cấu của robot, chúng ta hiểu Robotics là một khoa học liên ngành, gồm cơ khí, điện tử, kỹ thuật điều khiển và công nghệ tin học. Theo thuật ngữ hiện nay, robot là sản phẩm của ngành cơ điện tử (Mechatronics).

Theo khía cạnh nhân văn và khía cạnh khoa học – kỹ thuật của việc chế tạo robot thống nhất ở một điểm: thực hiện hoài bão của con người, là tạo ra thiết bị thay thế mình trong những hoạt động không thích hợp với mình, như:

- Các công việc lặp đi lặp lại, nhàm chán, nặng nhọc: vận chuyển nguyên vật liệu, lắp ráp, lau dọn nhà, …

- Trong môi trường khắc nhiệt hoặc nguy hiểm: như ngoài không gian vũ trụ, trên chiến trường, dưới nước sâu, trong lòng đất, nơi có phóng xạ, nhiệt độ cao, …

- Những việc đòi hỏi độ chính xác cao như thông tắc mạch máu hoặc các ống dẫn trong cơ thể, lắp ráp các cấu tử trong vi mạch, …

Lĩnh vực ứng dụng của robot rất rộng và ngày càng được mở rộng thêm. Ngày nay, khái niệm về robot đã mở rộng hơn khái niệm nguyên thủy rất nhiều. Sự phỏng tác về kết cấu, chức năng, dáng vẻ của con người là cần thiết nhưng không còn ngự trị trong kỹ thuật robot nữa. Kết cấy của cơ thể người và chúng cũng có thể thực hiện được những việc vượt xa khả năng của con người.

Hiện nay, với việc phát triển vượt bậc của khoa học công nghệ đã giúp chúng ta rất nhiều trong việc xây dựng và sản suất, đặc biệt là trong lĩnh vực công nghiệp [1]. Nhờ sự phát triển của những robot điều khiển đã gia tăng năng suất làm việc và phát

triển, đặc biệt là các tay máy robot trong công nghiệp. Robot công nghiệp thường có hai loại cấu trúc: dạng chuỗi hoặc song song. Robot cấu trúc dạng chuỗi có không gian làm việc rộng nhưng độ chính xác và khả năng chịu tải kém hơn. Do vậy, việc sử dụng cấu trúc loại nào thường phụ thuộc vào ứng dụng cụ thể. Ví dụ như ứng dụng gắp thả vật, phun sơn, kỹ thuật hàn và lắp ráp thường dùng robot chuỗi, trong khi đó ứng dụng gia công cơ khí hay thiết bị mô phỏng buồng lái thường dùng robot song song.

Trong thực tế công nghiệp, thì robot chuỗi vẫn được phổ biến hơn và việc phân tích, thiết kế cho chúng cũng có phần đơn giản hơn robot song song. Sự linh hoạt và không gian làm việc của robot chuỗi phụ thuộc vào số bậc tự do. Vì vậy mà nhiều loại robot công nghiệp 5 hoặc 6 bậc tự do được các công ty chú trọng phát triển.

Tiện lợi là như vậy nhưng việc điều khiển các tay máy robot không phải là điều đơn giản, chúng ta phải nhập các lệnh chương trình làm việc để có thể điều khiển những tay máy robot này. Thế nhưng việc lập trình cách đoạn dữ liệu để có thể điều khiển cánh tay robot là rất mất thời gian hơn thế nữa trong việc lập trình ta có thể xảy ra những lỗi liên quan đến các mã lệnh vì vậy việc lập trình để điều khiển cánh tay robot là rất mất thời gian và dễ có sai sót.

Hiện nay có khá nhiều phần mềm của các hãng lớn trên thế giới hỗ trợ thực hiện mô phỏng hoạt động robot. Chẳng hạn như phần mềm easy-rob là phần mềm phục vụ cho việc lập kế hoạch và mô phỏng sản xuất khi sử dụng các tế bào robot trong dây chuyền. Tất cả các chuỗi xử lí khi sử dụng robot ví dụ như: cầm nắm, lắp ráp, sơn phủ, hàn đều được lập chương trình cụ thể bằng phần mềm này và các tính toán đó ngay lập tức được cụ thể hóa bằng mô hình 3D ngay trong phần mềm. Các hoạt động của robot được mô phỏng có thể gồm chỉ 1 robot hoặc cùng một lúc nhiều robot với các phiên bản cao cấp hơn của phần mềm. Phần mềm này cho phép:

- Kiểm tra được tính năng và hoạt động của một cấu hình robot đã có sẵn, hỗ trợ cho việc sử dụng, sắp xếp một trạm robot hiệu quả hơn cũng như giúp cho việc quyết định đầu tư vào một loại robot nào đó được hợp lí và chính xác hơn, từ đó tiết kiệm được rất nhiều chi phí và thời gian.

- Thiết kế mới một loại robot nào đó, có thể sử dụng phần mềm này kết hợp với hệ thống 3D CAD kiểm tra khả năng làm việc của cấu hình robot. Khả năng nay hỗ trợ rất tốt cho việc thiết kế mới.
- Khả năng hỗ trợ nghiên cứu học tập về robot. Nhờ khả năng mô phỏng chính xác và linh hoạt các cấu hình robot mà phần mềm Easy-rob có thể xây dựng các giáo cụ ảo trực quan cho phép người học và người nghiên cứu có thể quan sát và tính toán cụ thể các cấu hình robot cũng như hiểu hơn về quá trình điều khiển robot bằng phương pháp dạy học mà không cần có robot thực tế.

Hoặc phần mềm FD on Desk là phần mềm mô phỏng robot nachi và cho phép phần mềm của bộ điều khiển FD/CFD hoạt động qua điều khiển của máy tính trên bàn làm việc hoặc bất kỳ máy tính nào có cài đặt phần mềm FD on Desk. Phần mềm FD on Desk có các tính năng đặc biệt sau :

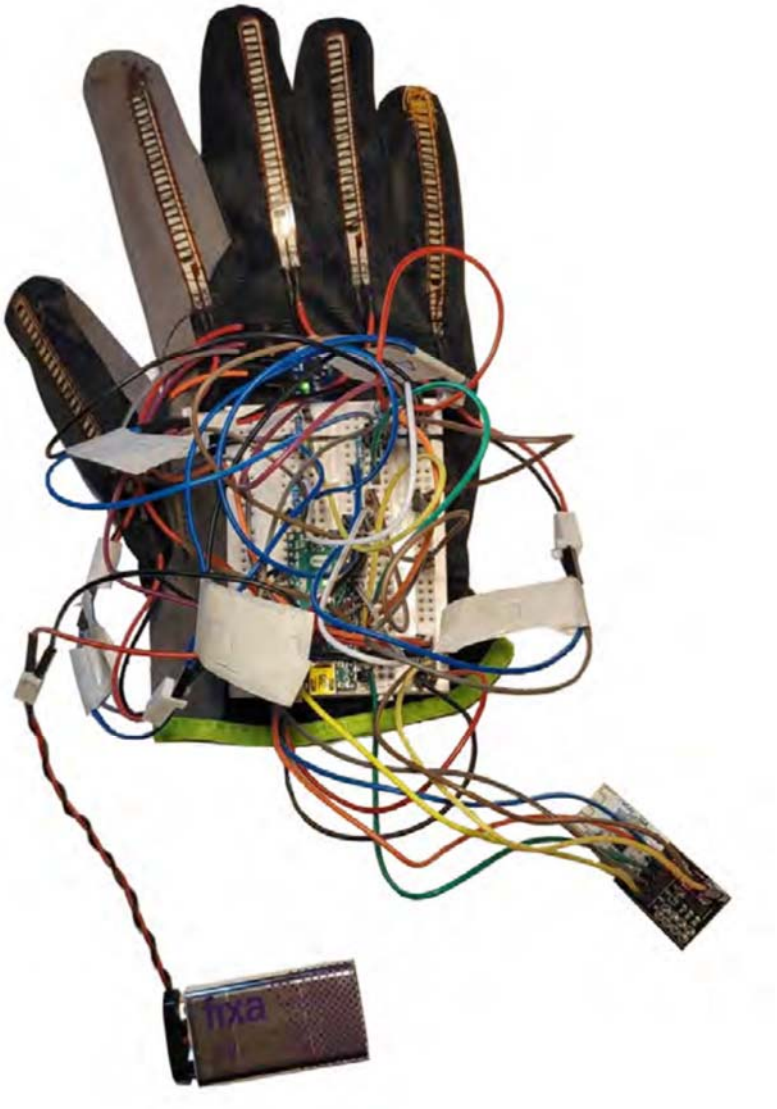
- Có thể được dùng ở bất cứ đâu, với bất cứ hệ điều hành nào. Không yêu cầu phần cứng đặc biệt
- Lập trình robot, kiểm tra chương trình trước khi đưa robot vào vận hành, kiểm tra trình tự các bước vận hành,…
- Các tập tin CAD (như IGES, STEP v…v…) có thể được insert vào trong việc mô phỏng robot nachi . Vì vậy, có thể thực hiện lập trình ngoại tuyến và kiểm tra sự giao thoa giữa các thiết bị ngoại vi v..v…
- Cho phép thực hiện mô phỏng theo thời gian chu trình với độ chính xác cao
- Bằng cách kết nối máy tính với bộ điều khiển FD/CFD, có thể thực hiện các thiết lập của FD/CFD mà không cần điều khiển robot trực tiếp. Có các chức năng giám sát từ xa.
- Có thể lập trình ngoại tuyến (“dạy”) một chương trình làm việc khi đang xác nhận tư thế robot hay tín hiệu I/O.
- Có thể thiết đặt chương trình PLC, các điều kiện hàn, thiết kế IFP v…v…
- Có thể thiết lập các thông số khác nhau cho các chương trình PLC, điều kiện hàn và thiết kế giao diện bảng điều khiển cũng như các chương trình làm việc.

- Tất cả các tập tin hoàn toàn tương thích với bộ điều khiển FD/CFD, cho phép dễ dàng phát lại các trạng thái hoạt động của các đơn vị thực tế trên bàn làm việc.
- FD on Desk giúp chúng ta có thể đào tạo mà chưa cần có robot nachi thực tế.

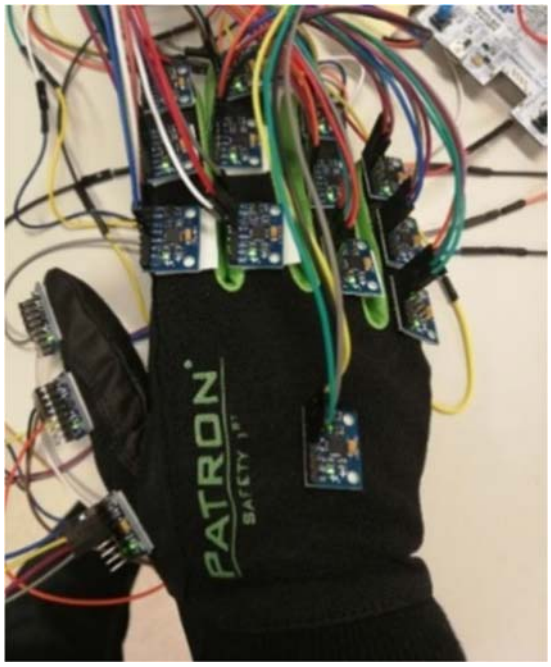
Ngoài ra còn có các phần mềm RobotStudio mô phỏng robot của hãng ABB, hoặc của hãng Universal Robots. Tuy nhiên đây đều là những phần mềm đi kèm với các robot của các hãng, do đó việc sử dụng phần mềm với một mô hình robot bất kỳ khác sẽ gặp khó khăn và vấn đề bản quyền.

Gần đây, bàn tay robot mô phỏng bàn tay người đã nghiên cứu và ứng dụng khá nhiều trong thực tế từ y học, công nghiệp đến cuộc sống. Chẳng hạn như các bàn tay robot hỗ trợ bác sĩ trong các ca mổ từ xa, bàn tay robot trị liệu phục hồi chức năng bàn tay người (Hình 1.1), … [2-4], hay sử dụng trong đào tạo thể thao [5], thao tác trong không gian ảo khi kết hợp với kính thực tại ảo VR [6], các ứng dụng điều khiển thiết bị (tay máy robot, se tự hành, …) từ xa [7], …. Tuy nhiên để các bàn tay đó hoạt động được giống với tay người thật thì cần có dữ liệu chuyển động của bàn tay thực. Việc thu thập và lấy dữ liệu đó thường sử dụng găng tay có gắn các cảm biến [8] để xác định chuyển động của các ngón tay và thông thường cần qua các bước thử nghiệm, mô phỏng lại hoạt động đó trước khi điều khiển các cơ cấu chấp hành. Một số loại găng tay sử dụng cảm biến điện trở (Hình 1.2) gắn dọc theo chiều dài ngón tay [6-12]. Khi ngón tay co hay duỗi sẽ làm các biến trở thay đổi giá trị điện trở, thay đổi tín hiệu điện và qua đó thu được mức độ co duỗi ngón tay. Phương pháp này giúp găng tay có thiết kế gọn gàng nhưng không xác định được chính xác góc xoay của từng đốt ngón tay. Gần đây, một số nghiên cứu tập trung phát triển loại găng tay sử dụng cảm biến gia tốc MPU5060 (Hình 1.3) [5,7,13-16]. Các cảm biến này sẽ được gắn vào các đốt ngón tay riêng biệt, từ đó tính toán được chuyển động giữa các đốt nhờ góc xoay thu được từ các cảm biến. Sử dụng các cảm biến MPU5060 khiến cho găng tay có hình dáng kích thước cồng kềnh hơn khi sử dụng cảm biến điện trở nêu trên. Tuy nhiên, phương pháp này cho độ chính xác chuyển động của bàn tay cao hơn và do đó có nhiều ứng dụng hữu ích hơn. Các nghiên cứu đã tập trung chủ yếu vào phát triển thiết kế găng tay và chương trình điều khiển thu nhận dữ liệu mà chưa có nghiên cứu nào phát triển tổng thể hệ thống từ thiết kế găng tay, phần mềm mô phỏng theo theo gian thực cũng như thu thập dữ liệu mô phỏng và tính toán mô phỏng cùng với phần cứng bàn tay robot dạng bàn tay người.

Hình 1.1: Găng tay hỗ trợ phục hồi chức năng bàn tay



Hình 1.2: Găng tay thu nhận chuyển động của bàn tay sử dụng cảm biến điện trở



Hình 1.3: Găng tay thu nhận chuyển động của bàn tay sử dụng cảm biến gia tốc MPU5060

Việc mô phỏng chuyển động cánh tay robot có thể giúp hiểu được nguyên lý cấu tạo và hoạt động của bàn tay robot dạng người. Tạo ra được lệnh điều khiển robot có thể áp dụng đến những bàn tay robot dạng người khác, giúp tiết kiệm thời gian lập trình chương trình chuyển động và tăng cao độ chính xác. Từ đó có thể dễ dàng phát triển mở rộng ra nhiều ứng dụng hữu ích trong cuộc sống.

1.2. Lý do lựa chọn đề tài

Hiện nay việc chế tạo và sản xuất bàn tay robot 5 ngón phục vụ người khuyết tật về tay có tiềm năng phát triển vô cùng lớn. Để thực hiện mô phỏng và điều khiển bàn tay robot theo những cử chỉ mà người dùng đang thực hiện hoặc muốn thực hiện, cùng với mục tiêu xây dựng một phần mềm có thể mô phỏng được hoạt động của bàn tay robot và tiến tới điều khiển trực tiếp bàn tay robot đó, đề tài ngiên cứu và mô phỏng chuyển động bàn tay robot đã được lựa chọn. Sản phẩm của đề tài sẽ giúp học sinh và sinh viên có thể hiểu rõ hơn về cấu tạo và hoạt động của bàn tay robot mô phỏng. Đề tài cũng giúp ta hiểu hơn về ứng dụng các phần mềm thiết kế và lập trình ứng dụng với Visual Studio bằng C++ để xây dựng chương trình mô phỏng 3D động học của robot phục vụ trong đào tạo.

1.3. Mục tiêu

Nghiên cứu tìm hiểu ứng dụng phần mềm SolidWorks và lập trình ứng dụng với Visual Studio bằng C++ để xây dựng chương trình mô phỏng 3D động học của bàn tay robot phục vụ trong đào tạo. Chương trình được tạo ra có thể thực hiện mô phỏng động học của các tay máy robot khác nhau nhờ vào việc vẽ mô hình trên SolidWorks hoặc phần mềm vẽ 3D bất kỳ và thiết lập tệp tin cấu hình mô tả robot đó.

Chương 2. Xây dựng mô hình 3D bàn tay robot

2.1. Tìm hiểu phần mềm SolidWorks

2.1.1. Lịch sử phần mềm SolidWorks

SolidWorks là phần mềm thiết kế 3D chạy trên hệ điều hành Windows và có mặt từ năm 1997, được tạo bởi công ty Dassault Systèmes SolidWorks Corp., là một nhánh của Dassault Systèmes, S. A. (Vélizy, Pháp) [17]. SolidWorks hiện tại được dùng bởi hơn 2 triệu kỹ sư và nhà thiết kế với hơn 165,000 công ty trên toàn thế giới.

Hình 2.1: Logo phần mềm SolidWorks

Công ty SolidWorks được thành lập vào tháng 12 năm 1993 bởi Hirschtick, tốt nghiệp trường MIT nổi tiếng- Massachusetts Institute of Technology; Hirschtick sử dụng1 triệu $ mà anh ta gây dựng được khi là thành viênMIT Blackjack Team để thành lập công ty. Trụ sở ban đầu ở Waltham, Massachusetts, USA, Hirschtick tuyển dụng một nhóm kỹ sư nhằm tạo một phần mềm 3D CAD dễ sử dụng, giá cả phải chăng, và có thể tùy biến trên Windows desktop. Sau này đổi địa chỉ là Concord, Massachusetts, SolidWorks đã phát hành phiên bản đầu tiên SolidWorks 95, năm 1995. Năm 1997 Dassault, Công ty nổi tiếng nhất với phần mềm CATIA, đã mua lại SolidWorks với 310 triệu đô la cổ phiếu. SolidWorks hiện tại có một số phiên bản như SolidWorks CAD, eDrawings một công cụ hỗ trợ, và DraftSight, một sản phẩm 2D CAD. SolidWorks được điều hành bởi John McEleney từ 2001 tới July 2007 và Jeff Ray từ 2007 tới tháng 1-2011. CEO hiện tại là Bertrand Sicot.

2.1.2. Các tính năng chính của phần mềm SolidWorks

a) Thiết kế mô hình 3D

Trong phần mềm SolidWorks thì đây được coi là tính năng nổi bật với việc thiết kế các các biên dạng 2D bạn sẽ dựng được các khối 3D theo yêu cầu.

b) Lắp ráp các chi tiết

Các chi tiết 3D sau khi được thiết kế xong bởi tính năng thiết kế có thể lắp ráp lại với nhau tạo thành một bộ phận máy hoặc một máy hoàn chỉnh. Tính năng này giúp bạn dễ dàng chỉnh sửa, thỏa sức sáng tạo và nghiên cứu dễ dàng cho những sản phẩm mới.

c) Xuất bản vẽ dễ dàng

Phần mềm SolidWorks cho phép ta tạo các hình chiếu vuông góc các chi tiết hoặc các bản lắp với tỉ lệ và vị trí do người sử dụng quy định mà không ảnh hưởng đến kích thước.

Công cụ tạo kích thước tự động và kích thước theo quy định của người sử dụng. Sau đó nhanh chóng tạo ra các chú thích cho các lỗ một cách nhanh chóng. Chức năng ghi độ nhám bề mặt, dung sai kích thước và hình học được sử dụng dễ dàng.

d) Tính năng Tab và Slot

Phần mềm SolidWorks 2022 cho phép người dùng tự động tạo ra các tính năng tab và slot được sử dụng để tự lắp ghép các bộ phận hàn. Các tính năng cải tiến kim loại khác bao gồm tính năng Normal Cut mới đảm bảo duy trì khoảng cách thích hợp cho sản xuất, và khả năng uốn mới cho phép người dùng tạo mới và trải phẳng góc uốn.

e) Cải tiến Quản lý dự án và quy trình

SolidWorks Manage cung cấp công cụ quản lý dữ liệu, dự án, và quản lý quy trình trong một gói phần mềm quen thuộc. Các khả năng quản lý các dự án, và quản lý quy trình được thêm vào SolidWorks PDM Professional.

f) Các tiện ích cải tiến

Online Licensing giúp cho việc sử dụng các license trên nhiều máy tính tiện lợi hơn trước rất nhiều. SolidWorks Login sẽ chuyển các nội dung và cài đặt các tùy chịn đến bất kỳ máy tính nào được cài SolidWorks, trong khi Admin Portal cho phép quản lý các sản phẩm và dịch vụ của SolidWorks dễ dàng hơn.

g) Tính năng gia công

Giải pháp gia công CAD CAM kết hợp, giải pháp có tên SolidWorks CAM, nó được tách ra để bán riêng. Giải pháp này khá đơn giản và dễ dùng. Các modul đơn giản thân thiện. Vậy nên để trải nghiệm bạn có thể đăng kí tại đây để dùng thử ngay giải pháp này cho gia công.

h) Phân tích động lực học

SolidWorks Simulation cung cấp các công cụ mô phỏng để kiểm tra và cải thiện chất lượng bản thiết kế của bạn. Các thuộc tính vật liệu, mối ghép, quan hệ hình học được định nghĩa trong suốt quá trình thiết kế được cập nhật đầy đủ trong mô phỏng.

2.1.3. Quy trình xây dựng mô hình trên SolidWorks

Phần mềm SolidWorks quản lý các tài liệu dạng PART, ASSEMBLY, DRAWING, để thực hiện vẽ thì đầu tiên ta cần vẽ dạng PART sau đó ta bắt đầu lắp ráp (ASSEMBLY) lại với nhau tạo thành một mô hình 3D hoàn chỉnh. Với dạng PART thì nguyên lí cơ bản là vẽ từ các dạng sketch sau đó ta bắt đầu dựng lên. Sau khi có PART ta bắt đầu tiến hành lắp ráp lại với nhau.

2.2. Tìm hiểu cấu tạo bàn tay robot 5 ngón

2.2.1. Về mặt giải phẫu bàn tay người

Cơ bản thì bàn tay gồm có 3 phần: lòng bàn tay, mu bàn tay, các ngón tay (ngón cái, ngón trỏ, ngón giữa, ngón áp út, ngón út). Đi sâu hơn về mặt giải phẫu thì bàn tay còn có các cấu trúc xương (nâng đỡ, định hình hình dáng bàn tay), cấu trúc cơ (tạo lực và chuyển động). Cấu trúc xương bàn tay có 27 xương (8 xương cổ tay, 5 xương bàn tay, 14 xương đốt ngón tay), để cấu trúc này gắn kết với nhau thì cần 29 khớp và ít nhất 123 dây chằng [18]. Cấu trúc cơ thì gồm 2 nhóm cơ chính là cơ gấp và cơ duỗi. Các cơ gấp được kết nối với mặt dưới của cánh tay và bổ trợ cho động tác gập của các ngón tay. Cơ duỗi được gắn với phía trên của cẳng tay và làm thẳng các ngón tay.

2.2.2. Bàn tay robot 5 ngón

Việc áp dụng hoàn toàn cấu tạo bàn tay người lên bàn tay robot 5 ngón có thể nói là bất khả thi, hay việc mô phỏng lại những cử động rất phức tạp của tay người thường không cần thiết. Nhằm tối giản hóa thiết kế và tập trung vào mô phỏng các cử chỉ cơ bản nhưng cần thiết trong hoạt động sinh hoạt, làm việc của người dùng, thì cấu tạo bàn tay robot 5 ngón có thay đổi đáng kể so với bàn tay người trên thực tế. Để nâng đỡ và định hình thì bàn tay robot có lòng bàn tay và 15 đốt ngón tay (tương ứng với cấu

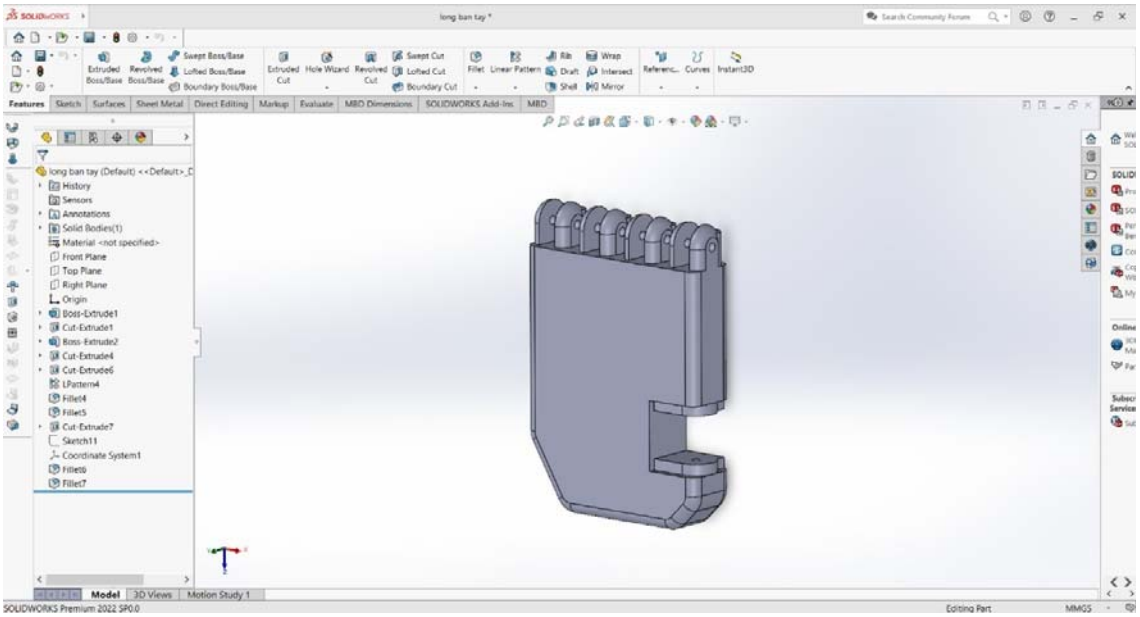
trúc 27 xương của bàn tay người) [19]. Việc tạo chuyển động cho các đốt ngón tay tương ứng với các hệ cơ của bản tay, có thể sử dụng các hệ thống khí nén hoặc các dây kéo. Hệ thống servo và dây cước có chức năng tạo chuyển động cho từng đốt ngón tay. Giống với hệ thống cơ bàn tay thì dây cước cũng sẽ bao gồm nhóm dây duỗi và dây gập giúp chuyền chuyển động của servo đến các đốt ngón tay

2.3. Xây dựng mô hình 3D và bài toán động học

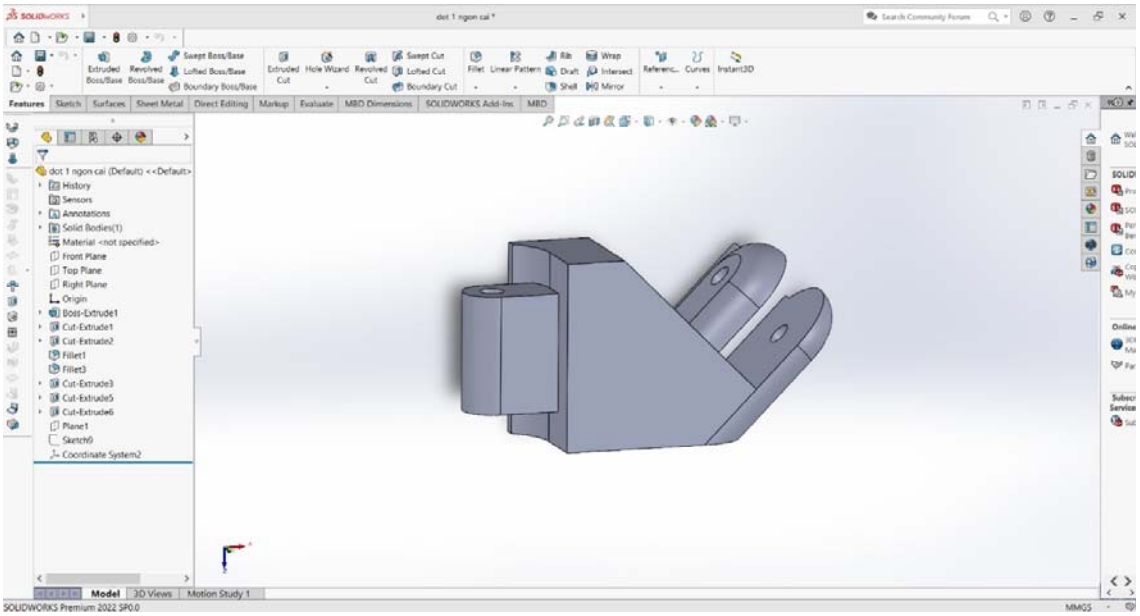
2.3.1. Xây dựng mô hình 3D bàn tay robot 5 ngón

Để có thể xây dựng mô hình bàn tay robot 5 ngón trên SolidWorks, trước tiên cần phải vẽ các PART. Để vẽ PART ta cần có những số liệu về kích thước độ dài, độ rộng, và độ cao của các bộ phận trong bàn tay robot. Đầu tiên vẽ khâu 0 (lòng bàn tay, Hình 2.5) của bàn tay máy, để vẽ ta sử dụng Sketch để vẽ ra hình dáng của vật, sau khi chọn hình dáng của vật cần vẽ ta dùng Smart Dimension để lựa chọn kích thước của vật(lưu ý phải chọn theo đúng kích thước để khi ta lắp ráp sẽ không xảy ra bất kì lỗi gì về kích thước của vật). Khi đã vẽ được hình dáng vật ta chọn vào phần Features sau đó chọn Extruded Boss để dựng vật lên thành hình khối, để có thể cắt gọt đi những phần không cần thiết trên hình khối ta chọn Extruded Cut để cắt. Áp dụng lại bước vẽ Sketch, Extruded Boss, Extruded Cut để tạo phần khớp nối giữa lòng bàn tay và các ngón tay. Phần khớp này được thiết kế để có thể giới hạn góc quay của các đốt sao cho gần giống với thực tế. Tiếp đến dùng Linear Pattern để tạo thêm 3 bản sao của các khớp nối dọc theo chiều ngang của lòng bàn tay. Cuối cùng dùng lệnh Fillet để bo cong các đoạn gấp khúc.

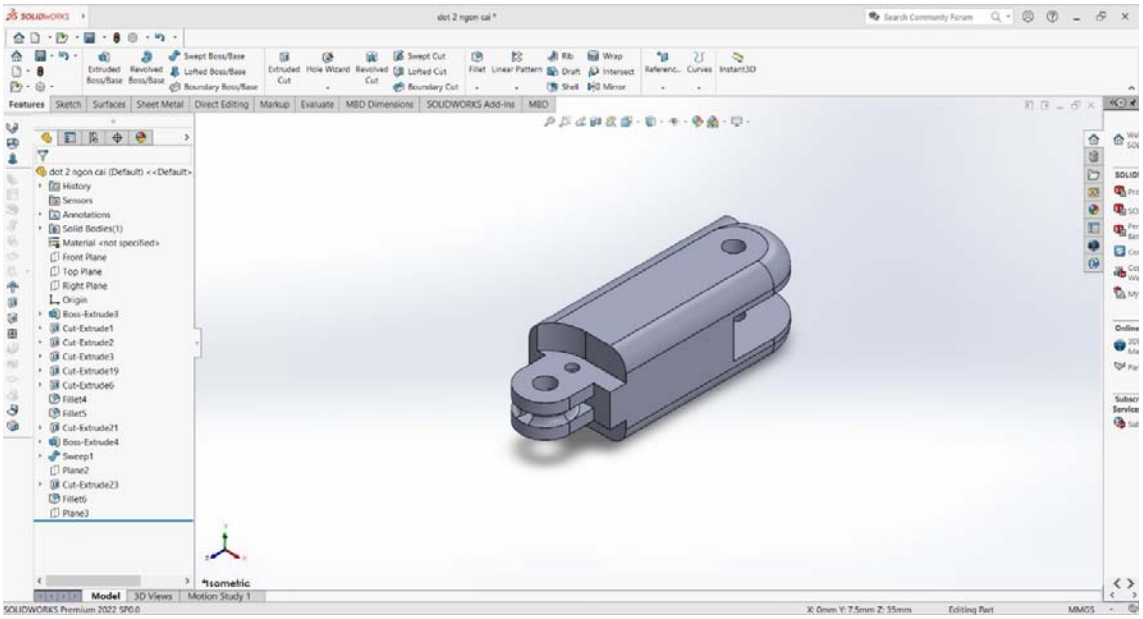
Tiếp theo là dựng các đốt ngón tay bằng các lệnh tương tự như việc dựng lòng bàn tay ở trên. Ngoài ra cũng cần dùng thêm lệnh Swept Boss/Base để tạo rãnh và dựng thêm mặt phẳng phụ qua lệnh Plane để khoét phần đi dây (Sketch, Extrude Cut). Các đốt hầu hết đều có cấu tạo tương tự nhau và được ràng buộc theo độ dài đốt. Như vậy chỉ cần xây dựng 3 loại đốt cơ bản là có thể hoàn thành toàn bộ 15 đốt ngón tay từ việc copy và thay đổi kích thước chiều dài đốt. Ba loại đốt bao gồm: đốt xiên (đốt 1 ngón cái), đốt 2 khớp nối thẳng (đốt 2 ngón cái và đốt 1-2 các ngón còn lại), đốt 1 khớp nối (đốt 3 của năm ngón).



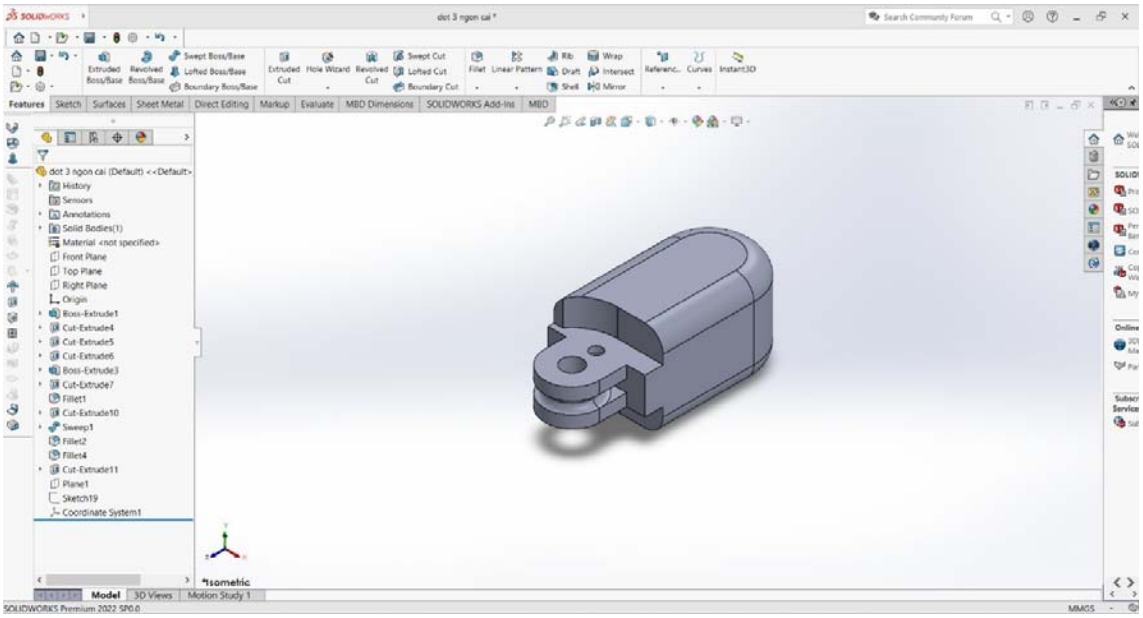
Hình 2.2: Lòng bàn tay robot 5 ngón



Hình 2.3: Đốt xiên (đốt 1 ngón cái)



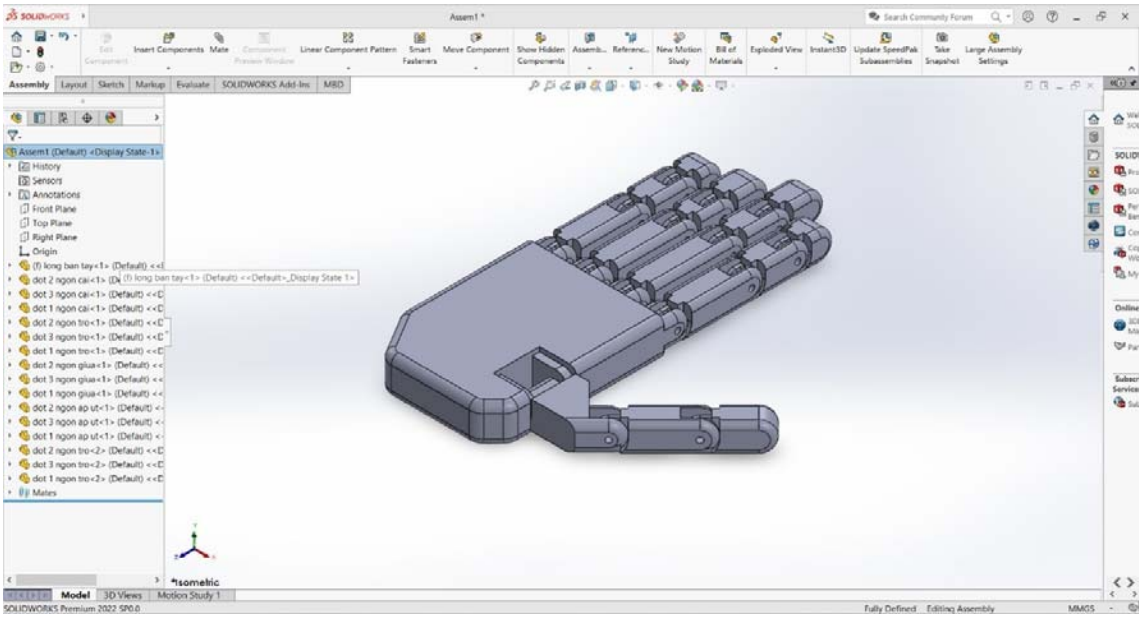
Hình 2.4: Đốt 2 khớp nối thẳng (đốt 2 ngón cái)



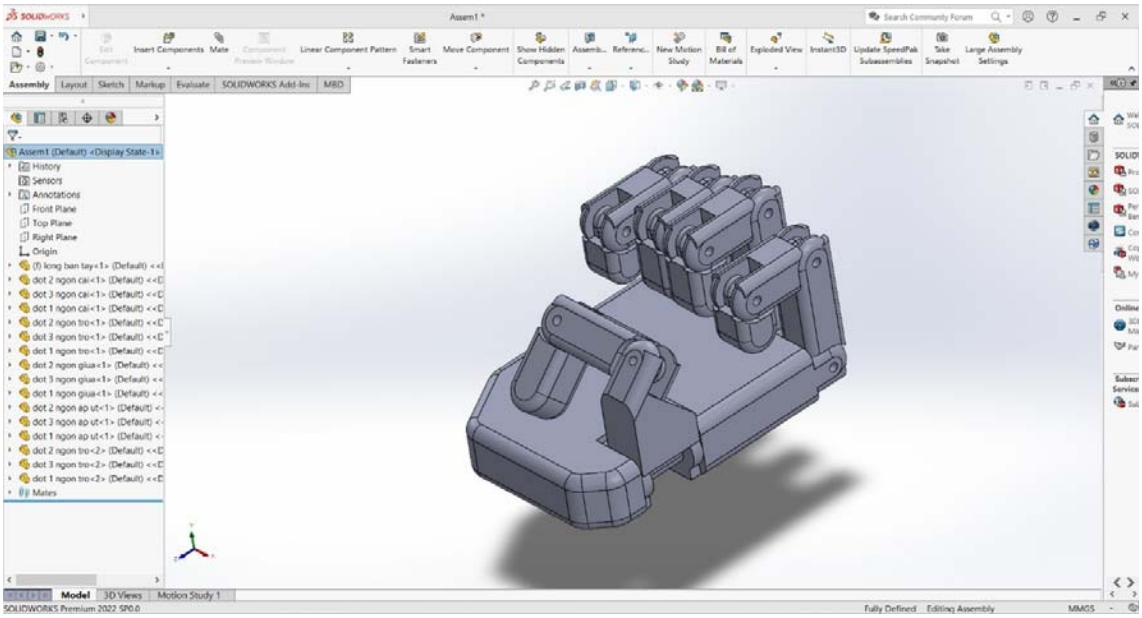
Hình 2.5: Đốt 1 khớp nối (đốt 3 ngón cái)

Sau khi xây dựng đầy đủ lòng bàn tay và các đốt ngón tay ở mỗi trường Part ta chuyển các file Part sang môi trường Assembly để lắp ghép thành bàn tay robot hoàn chỉnh. Tệp lòng bàn tay được thêm vào đầu tiên sẽ được mặc định là khâu cố định, các file đốt ngón tay được thêm vô sau và dùng Mate để tạo ràng buộc hạn chế một số bậc tự do của các đốt và lòng bàn tay với nhau. Kết quả lắp ghép mô hình bàn tay được

thể hiện trên Hình 2.6. Với mô hình lắp ghép đã được tạo ràng buộc đầy đủ (bằng lệnh Mate) sẽ cho chép dịch chuyển vị trí của các ngón tay như trên hình 2.7.



Hình 2.6: Mô hình bàn tay robot 5 ngón khi mở hoàn toàn

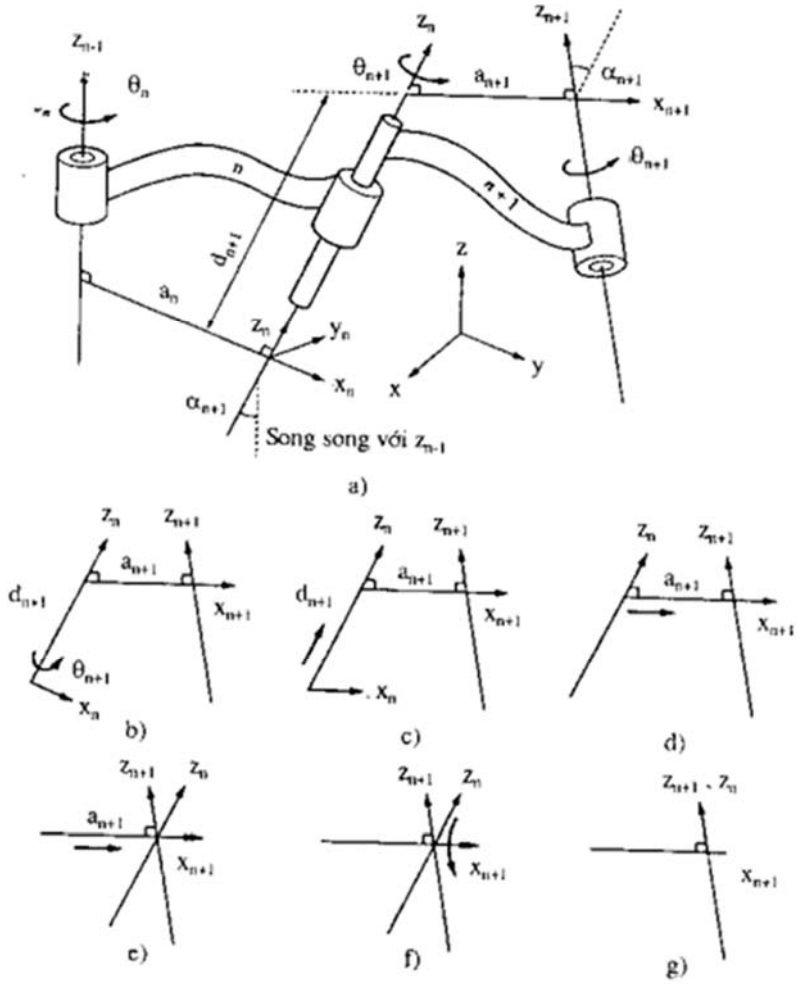


Hình 2.7: Mô hình bàn tay robot 5 ngón khi gập hoàn toàn

2.3.2. Xây dựng bài toán động học của bàn tay.

2.3.2.1. Phương pháp thiết kế hệ tọa độ - phép biểu diễn Denavit-Hartenberg

Để nghiên cứu mối quan hệ giữa các thanh nối, khớp và tay robot, ta đặt hệ tọa độ cho các thanh nối. Theo phương pháp biểu diễn Danevit-Hartenberg (D-H), hệ toạ độ thanh nối i được xây dựng thao nguyên tắc sau (Hình 2.8) [20].



Hình 2.8: Thiết kế khung tọa độ thanh nối

+ Gốc hệ tọa độ thanh i đặt trùng với chân pháp tuyến chung của trục i và i+1 và nằm trên trục khớp thứ i+1

+ Trục z_i đặt theo phương trục khớp i+1.

+ Trục x_i đặt theo phương pháp tuyến chung của trục i và i+1 theo hướng đi từ trục i đến i+1.

Một số trường hợp đặc biệt:

+ Khi hai trục cắt nhau: sẽ không có pháp tuyến chung giữa hai khớp. Khi đó điểm gốc của hệ tọa độ là giao điểm của hai trục và trục x được đặt dọc theo đường vuông góc với mặt phẳng chứa hai trục z đó.

+ Hai trục song song, sẽ có nhiều pháp tuyến chung. Khi đó sẽ chọn được pháp tuyến chung trùng với pháp tuyến chung của khớp trước. Hệ tọa độ chọn sao cho d_i là nhỏ nhất.

+ Đối với khớp tịnh tiến: khoảng cách d_i là biến khớp. Hướng của trục khớp trùng với hướng di chuyển của khớp. Hướng của trục được xác định, nhưng vị trí trong không gian không được xác định. Khi đó chiều dài a_i không có ý nghĩa nên đặt $a_i = 0$. Gốc tọa độ đặt trùng với thanh nối tiếp theo.

Theo nguyên tắc đặt hệ tọa độ như trên, bắt đầu gắn hệ tọa độ từ bệ (thân) robot là hệ tọa độ 0: trục z_0 trùng với khớp 1. Gốc hệ tọa độ thanh 6 trùng với hệ tọa độ thanh nối 5.

2.3.2.2. Quan hệ giữa hai hệ tọa độ i và i-1

Một cách tổng quát, quan hệ giữa hai hệ tọa độ i và i-1 được xác định bằng các phép biến đổi theo thứ tự sau:

+ Quay quanh trục zi-1 một góc θi sao cho trục xi-1 trùng với phương của trục xi.

+ Tịnh tiến dọc trục zi-1 một đoạn di để gốc khung tọa độ mới trùng với chân pháp tuyến chung trục i và i-1.

+Tiện tĩnh dọc trục xi-1 (phương pháp tuyến chung) một đoạn ai.

+ Quay xung quanh trục xi-1 một góc αi sao cho trục zi-1 trùng với trục zi.

Các phép biến đổi trên được thực hiện so với tọa độ hiện tại. Do đó phép biến đổi tổng hợp được xác định như sau:

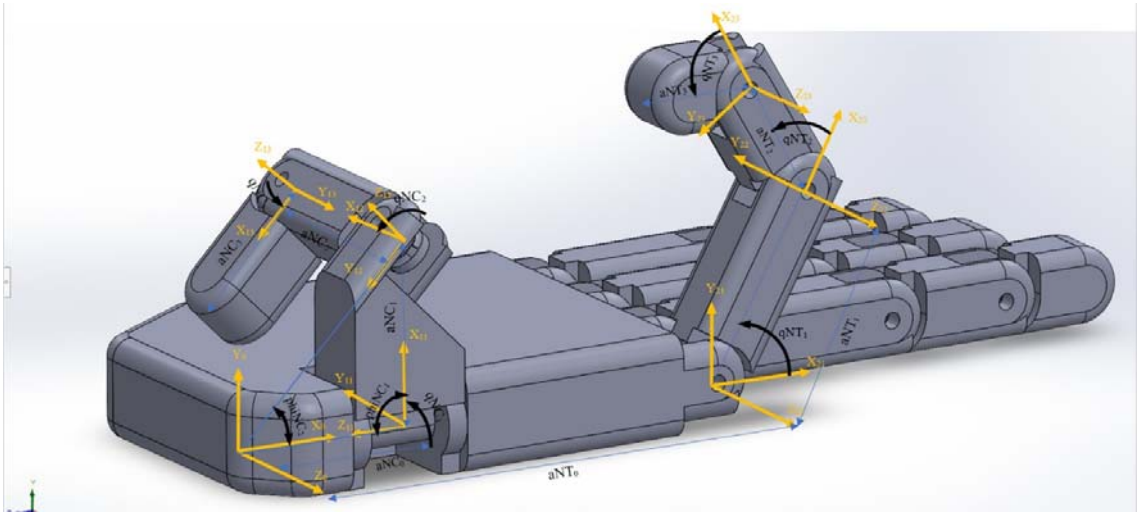
$$A_i = Rot(z,\theta_i)Trans(0,0,d_i)\,\mathrm{Trans}(a_i,0,0)Rot(x,\alpha_i)$$

Thay các ma trận của phép biến đổi đơn vào ma trận biểu diễn hệ tọa độ, sau một số biến đổi, nhận được ma trận biểu diễn quan hệ giữa hai hệ tọa độ i và i-1 như sau:

$$^{i-1}A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.3.2.3. Thiết lập hệ trục tọa độ của bàn tay robot theo quy tắc Denavit Hartenberg

Mỗi ngón của bàn tay robot sẽ bao gồm 3 khâu, các ngón như ngón trỏ, ngón giữa, ngón áp út, ngón út có cách chọn hệ tọa độ tương tự nhau. Áp dụng quy tắc Denavit Hartenberg ta được các hệ trục tọa độ cho từng đốt và các tham số được biểu diễn như sau (Hình 2.9)



Hình 2.9: Sơ đồ gắn các hệ trục tọa độ cho cả bàn tay

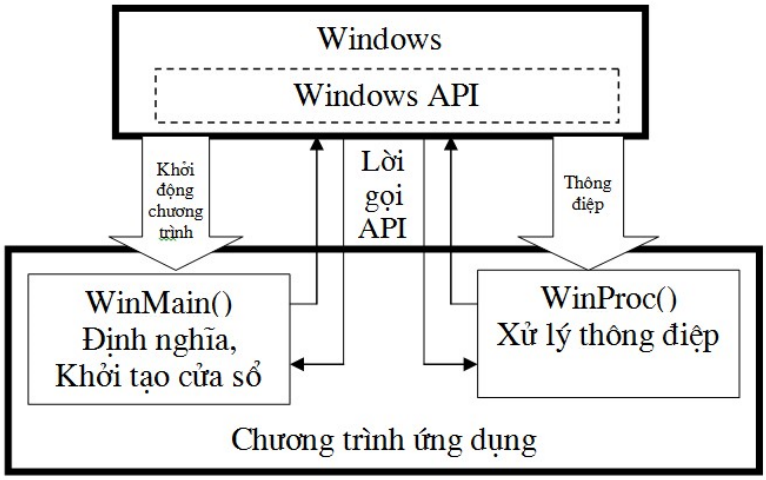
Bảng 2.1: Bảng thông số D-H của 5 ngón tay

Ngón	Khâu	d_i	θ_i	a_i	α_i
Ngón cái	1	aNC0	qNC1	0	phiNC1
	2	aNC1	qNC2	0	phiNC2
	3	aNC2	qNC3	0	0
Ngón trỏ	1	aNT0	qNT1	0	0
	2	aNT1	qNT2	0	0
	3	aNT2	qNT3	0	0
Ngón giữa	1	aNG0	qNG1	0	0
	2	aNG1	qNG2	0	0
	3	aNG2	qNG3	0	0
Ngón áp út	1	aNAPUT0	qNAPUT1	0	0
	2	aNAPUT1	qNAPUT2	0	0
	3	aNAPUT2	qNAPUT3	0	0
Ngón út	1	aNUT0	qNUT1	0	0
	2	aNUT1	qNUT2	0	0
	3	aNUT2	qNUT3	0	0

Chương 3. Xây dựng phần mềm mô phỏng bàn tay robot

3.1. Lập trình ứng dụng Windows bằng Visual Studio 2022

Nguyên lý cơ bản của lập trình Window là cơ sở xử lý thông điệp như mô tả trên Hình 3.1 [21]. Tuy nhiên để lập chương trình bằng các hàm API là vô cùng phức tạp và rất khó quản lý, do đó với lợi thế của ngôn ngữ lập trình hướng đối tượng, Microsoft đã tập hợp các hàm API đó lại và đóng gọi lại thành các lớp nền tảng của Microsoft và được gọi là MFC (Microsoft Foundation Class). Do vậy, chương trình mô phỏng ở đây sẽ được xây dựng dựa trên các lớp MFC.

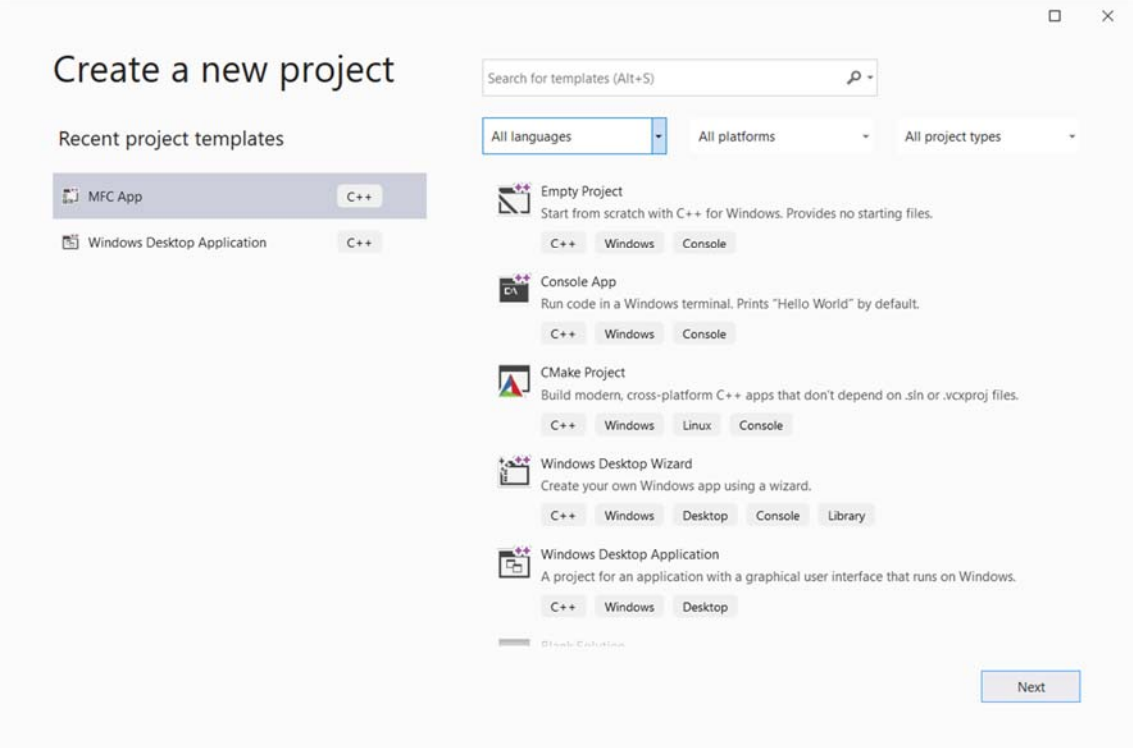


Hình 3.1: Mô tả cơ chế lập trình xử lý thông điệp trên Windows

3.1.1. Các bước tạo đề án MFC

Để tạo đề án (project) khung cho chương trình mô phỏng, dùng tiến trình tạo đề án của bộ Microsoft Visual Studio với phiên bản Visual Studio 2022. Quá trình này gồm các bước như sau:

Từ trình thực đơn (menu) của Microsoft Visual Studio C++, chọn "File\New ...", sẽ có hộp thoại "New" xuất hiện như trên Hình 3.2. Trên hộp thoại này, cần chọn trang "Project" rồi chọn "MFC AppWizard (exe)" rồi ấn nút OK để bắt đầu tiến trình tạo khung ứng dụng với nền tảng MFC, tiến trình này sẽ trải qua các bước được mô tả bên dưới.



Hình 3.2: Giao diện tạo đề án mới của Visual Studio C++

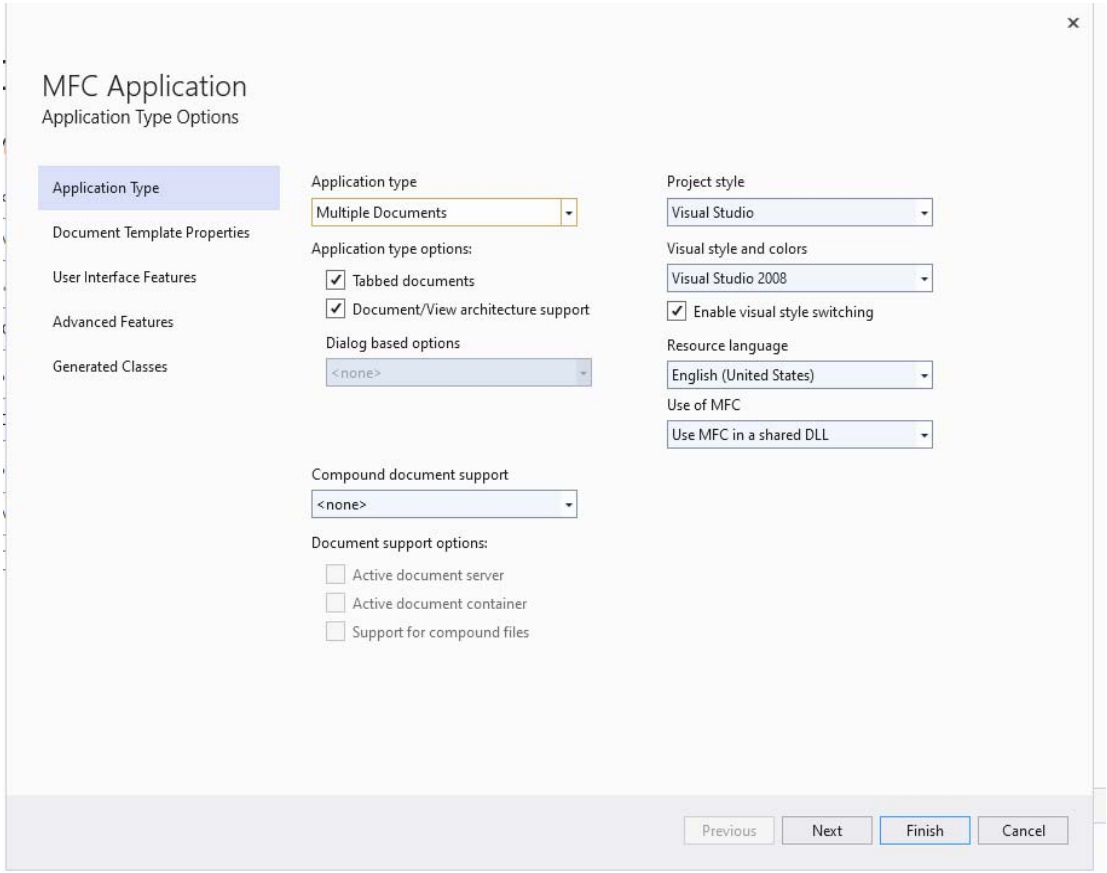
Bước 1 (Step 1): là bước đầu tiên của tiến trình, cho phép người dùng lựa chọn dạng khung giao diện ứng dụng muốn tạo (Hình 3.3).

Single Document (SDI: Single Document Interface): cho phép tạo ứng dụng đơn tài liệu, tại mỗi thời điểm chương trình chỉ có thể mở một tài liệu.

Multi Documents (MDI: Multi Documents Interface): cho phép tạo ứng dụng đa tài liệu, tại mỗi thời điểm chương trình có thể mở đồng thời nhiều tài liệu, mỗi tài liệu được quản lý bởi một cửa sổ riêng nằm gọn trong cửa sổ chính của chương trình.

Dialog Based: tạo ứng dụng là một hộp thoại. Đây là loại ứng dụng đơn giản nhất.

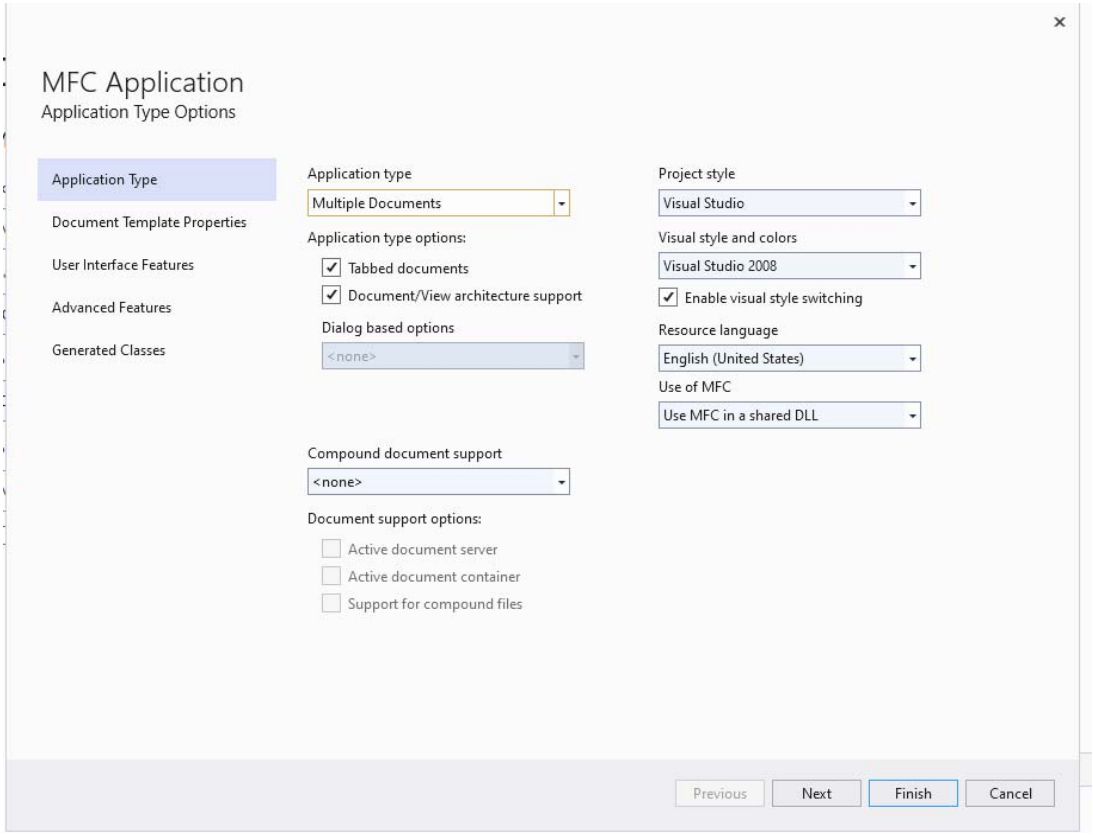
Với mục đích của chương trình mô phỏng ở đề tài này, chúng tôi lựa chọn dạng khung ứng dụng đơn tài liệu (Single Document).



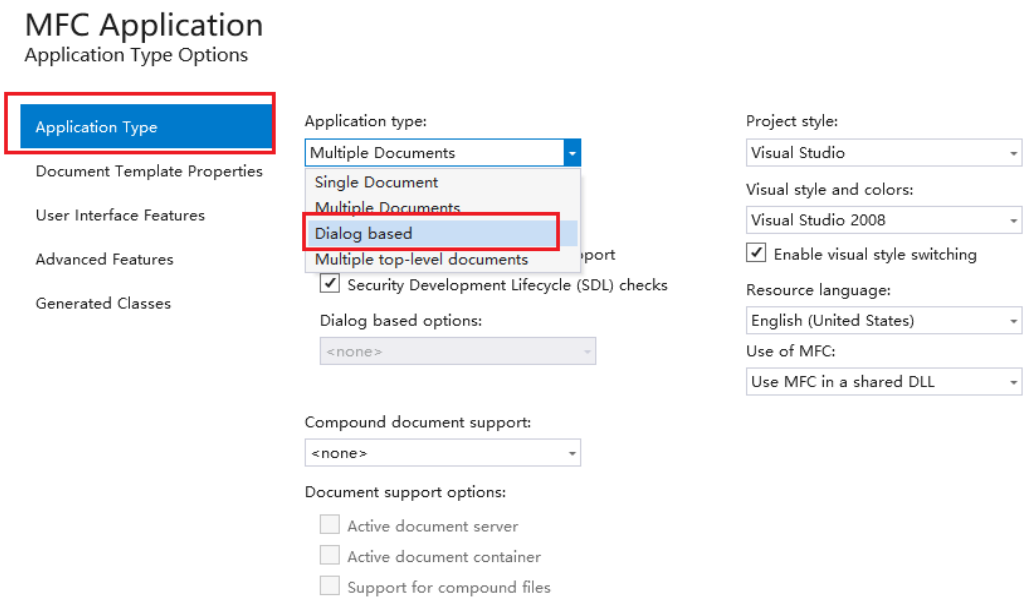
Hình 3.3: Giao diện lựa chọn dạng giao diện ứng dụng muốn tạo

Bước 2 và 3 có giao diện như trên Hình 3.4. Các bước này cho phép người lập trình thiết đặt một số tùy chọn cho khung ứng muốn khởi tạo. Ở đây, các lựa chọn đã phù hợp với mục đích của chương trình mô phỏng nên không cần thay đổi gì. Chỉ cấn nhấn nút “Next” để chuyển qua bước tiếp theo.

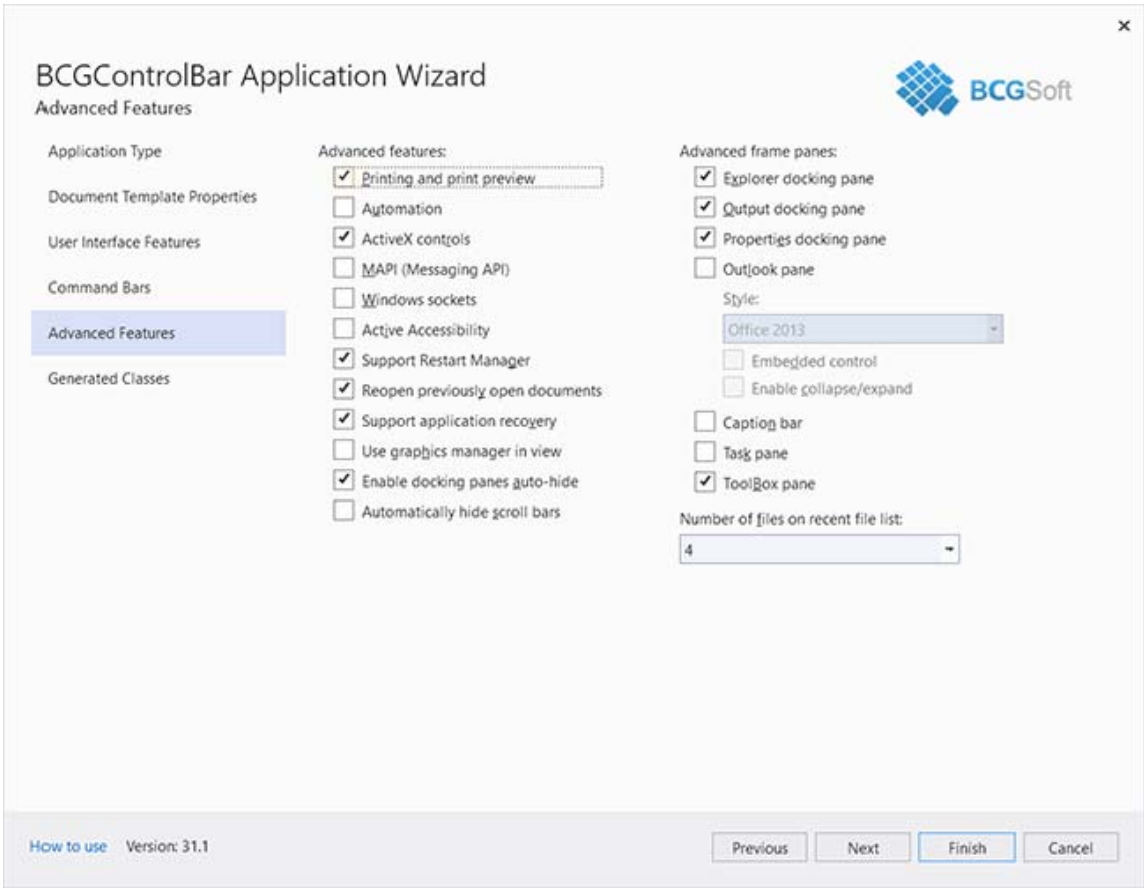
Bước 4: cho phép tùy chọn một số thể hiện trên giao diện của ứng dụng sẽ được tạo lập (Hình 3.5). Chẳng hạn như có tạo ra các thanh công cụ có khả năng thả nổi và bắt dính vào khung ứng dụng hay không, có thanh trạng thái hay không, có chức năng in và quan sát trước khi in hay không, …



Hình 3.4: Giao diện của bước 2 và 3 trong tiến trình tạo đề án



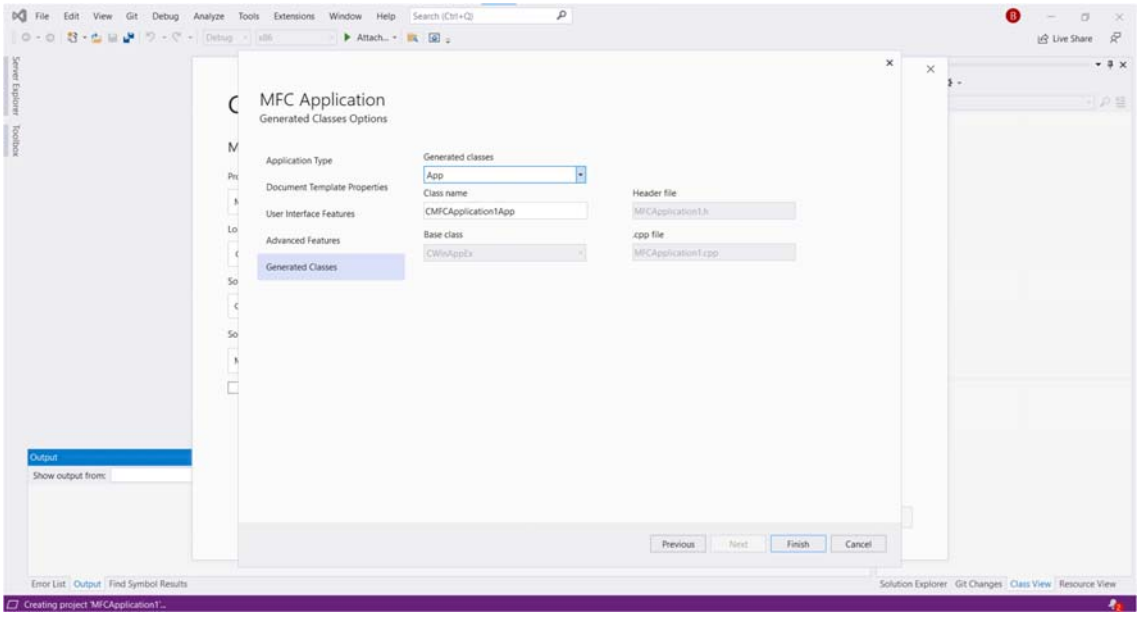
Hình 3.5: Giao diện của bước 4 trong tiến trình tạo đề án



Hình 3.6: Giao diện của chức năng Advance Options

Trong bước này cần chú ý đến nút "Advance". Nếu nhấn vào nút này sẽ cho ra hộp thoại "Advance Options" như trên Hình 3.6. Trên đó có hai trang với các yêu cầu lựa chọn khác nhau. Ở trang thứ nhất là "Document Template Strings" cho phép đặt tên và phần mở rộng của loại tài liệu có thể mở được bằng hộp thoại mở tệp (Open) của chương trình. Còn ở trang thứ hai là "Window Styles" cho phép lựa chọn khung ứng dụng khi được khởi tạo có các nút phóng to, thu nhỏ cửa sổ, thực đơn hệ thống hay khi xuất hiện có phóng to hết khung màn hình máy tính hay không.

Bước thứ 5 có giao diện như ở Hình 3.7, cho phép lựa chọn giao diện ứng dụng theo kiểu chuẩn của MFC hay theo phong cách của ứng dụng Windows Explorer, có thêm các ghi chú vào tệp tin mã nguồn hay không, hay có sử dụng thư viện MFC theo cách chia sẻ hay liên kết tĩnh vào ứng dụng.



Hình 3.7: Giao diện của bước 5 trong tiến trình tạo đề án

Bước thứ 6 là bước cuối cùng của quá trình này và có giao diện như trên Hình 3.8. Trong bước này sẽ tổng hợp lại các lớp đối tượng sẽ được tạo ra để quản lý khung ứng dụng chính. Ở đây, người dùng có thể lựa chọn lớp đối tượng quản lý khung quan sát trong ứng dụng là Cview, CeditView, CformView, … tùy thuộc vào mục đích của ứng dụng. Trong đề tài này, chúng tôi sử dụng lớp Cview để quản lý khung quan sát, đó sẽ là vùng được đặt môi trường đồ họa OpenGL để thể hiện các mô hình mô phỏng.

Danh sách các lớp sẽ được tạo ra:

CxxxView được kế thừa từ lớp quan sát, có thể là CView (mặc định), CEditView, CFormView, ... đảm nhận công việc thể hiện tài liệu trong chương trình. Được đặt trong file “xxxView.h" và "xxxView.cpp".

CxxxApp được kế thừa từ lớp CWndApp, đảm nhận công việc khởi tạo và quản lý ứng dụng. Được đặt trong file "xxx.h" và "xxx.cpp".

CMainFrame được kế thừa từ lớp CFrameWnd nếu chương trình dạng Single Document, còn nếu là Multi Document thì kế thừa từ lớp CMDIFrameWnd, là lớp quản lý cửa sổ chính của chương trình. Được đặt trong file "MainFrm.h" và "MainFrm.cpp".

CxxxDoc được kế thừa từ lớp CDocument, quản lý tài liệu của chương trình. Được đặt trong file "xxxDoc.h" và "xxxDoc.cpp".

Nếu chương trình là dạng MultiDocument thì sẽ có thêm lớp CChildFrame kế thừa từ lớp CMDIChildWnd, là lớp quản lý cửa sổ con ứng với mỗi tài liệu đang có được đặt trong file "ChildFrm.h" và "ChildFrm.cpp".

3.1.2. Làm việc với Menu, ToolBar, Status

5.1.2.1. Sửa đổi Menu và ToolBar có sẵn

Mặc định ban đầu của MFC sẽ tạo ra cho bạn một menu và một thanh công cụ với ID là IDR_MAINFRAME, đây sẽ là menu và thanh công cụ mặc định của chương trình và gắn với cửa sổ ứng dụng chính và được khởi tạo bởi CMainFrame. Menu và thanh công cụ này mới chỉ có một số chức năng cơ bản có ở đa số các ứng dụng.

Để bổ xung thêm chức năng mới của menu và thanh công cụ trên bạn nên dùng trình soạn thảo Resource của Microsoft Visual Studio.Để tạo công việc xử lý sự kiện bạn cần tạo ra một hàm xử lý sự kiện và tạo ra ánh xạ thông điệp để quy chiếu sự kiện tới hàm đó, công việc này có thể được thực hiện nhanh chóng nhờ ClassWizard hoặc có thể kích chuột phải lên tên lớp (ở vùngWorkSpace) rồi chọn "Add Windows Message Handler …". Trong file tiêu đề (.h) được bổ xung thêm dòng khai báo hàm thành viên có dạng :

afx_msg ReVal memberFxn (ListParam);

ReVallà giá trị trả về của hàm

ListParamlà danh sách tham số, afx_msgchỉ rằng hàm này là một hàm xử lý thông điệp.

Trong file nguồn (.cpp), bổ xung thêm ánh xạ thông điệp vào giữa khu vực :

BEGIN_MESSAGE_MAP()

……

……

END_MESSAGE_MAP()

Bổ xung triển khai của hàm thành viên đã được thêm khai báo trong file tiêu đề (thường được đặt vào cuối cùng của file cpp).

5.1.2.2. Khai báo một thanh công cụ mới :

CToolBar m_wndToolBar;

Trong hàm int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct) thêm các dòng lệnh sau để khởi tạo và gán biến trên với một thanh công cụ đã tạo trong tài nguyên thông qua ID của nó:

```
if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD |
WS_VISIBLE | CBRS_TOP | CBRS_GRIPPER | CBRS_TOOLTIPS |
CBRS_FLYBY | CBRS_SIZE_DYNAMIC) ||
!m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
{
TRACE0("Failed to create toolbar\n");
return -1;      // fail to create
}
```

ID của thanh công cụ là IDR_MAINFRAME, có thể thay ID khác.

3.2. Tìm hiểu môi trường đồ họa 3D với OpenGL

3.2.1. Khái niệm OpenGL

OpenGL là viết tắt của Open Graphic Library, nghĩa là thư viện đồ họa mở, được phát triển đầu tiên bởi Silicon Graphic, Inc., là một giao diện phần phần mềm hướng thủ tục theo chuẩn công nghiệp hỗ trợ đồ họa 3 chiều [22]. Thư viện cung cấp khoảng 120 tác vụ để vẽ các primitive trong nhiều mode khác nhau. Đây là một giao diện phần mềm độc lập với phần cứng (hardware – independent software interface) hỗ trợ cho lập trình đồ họa với ngôn ngữ lập trình và thiết bị phần cứng khác nhau (Hình 3.8).

Hình 3.8: OpenGL

Để sử dụng các nội dung trên của OpenGL trong đề án chương trình, thêm các dòng sau vào header file (nên đưa vào stdafx.h):

```
#include <gl\gl.h>
#include <gl\glu.h>
#pragma comment(lib, "opengl32.lib")
#pragma comment(lib, "glu32.lib")
```

3.2.2. Các kiểu dữ liệu

OpenGL định nghĩa nhiều kiểu dữ liệu có thể sử dụng trong chương trình OpenGL:

Kiểu dữ liệu	**Kiểu tương đương**
GLbyte	signed char
GLshort	short
GLint	int
GLsizei	long
GLfloat	float
GLclampf	float
GLdouble	double
GLclampd	double
GLubyte	unsigned char
GLboolean	unsigned char
GLushort	unsigned short
GLuint	unsigned long
GLenum	unsigned long
GLbitfield	unsigned long
GLvoid	void
HGLRC	HGDIOBJ

3.2.3. Ngữ cảnh diễn tả

Có 5 hàm "WGL" được cung cấp trong triển khai OpenGL trong WindowNT chịu trách nhiệm quản lý các ngữ cảnh diễn tả được liệt kê trong Bảng 3.1.

Bảng 3.1: Các hàm quản lý ngữ cảnh diễn tả

Tên hàm	Sử dụng
wglCreateContext()	tạo lập một ngữ cảnh diễn tả mới.
wglDeleteContext()	xoá bỏ một ngữ cảnh diễn tả.
wglGetCurrentContext()	trả về một điều khiển tới ngữ cảnh diễn tả hiện thời.
wglGetCurrnetDC()	lấy điều khiển tới DC cộng tác với ngữ cảnh diễn tả hiện thời.
wglMakeCurrent()	đưa một ngữ cảnh diễn tả thành hiện hành.

3.2.4. Các định dạng điểm ảnh

Trước khi chương trình của bạn có thể tạo lập một ngữ cảnh diễn tả, nó phải được đặt định dạng điểm ảnh của thiết bị mà nó chứa các thuộc tính cho bề mặt vẽ của thiết bị bằng các hàm ở Bảng 3.2. Các thuộc tính này được gộp vào trong bề mặt vẽ sử dụng chế độ mầu RGBA hoặc chỉ số, hoặc vùng đệm điểm ảnh sử dụng vùng đệm đơn hay kép, số bit mầu, số bit được sử dụng trong các vùng đệm sâu và khung tô, và các thông tin đồ hoạ khác của OpenGL.

Bảng 3.2: Các hàm Win32 quản lý các định dạng điểm ảnh

Tên hàm	Sử dụng
ChoosePixelFormat()	trả lại định dạng điểm ảnh bị đóng cuối cùng khi đặt một định dạng mới.
DescribePixelFormat()	thu được thông tin về định dạng điểm ảnh.
GetPixelFormat()	Lấy định dạng điểm ảnh của ngữ cảnh thiết bị.
SetPixelFormat()	Đặt một định dạng điểm ảnh của ngữ cảnh thiết bị.

3.2.5. Thiết đặt một định dạng điểm ảnh

Một khi bạn có cấu trúc PIXELFORMATDESCRIPTOR đã khởi tạo, bạn có thể thiết đặt một định dạng điểm ảnh. Đoạn mã sau trích từ một chương trình MFC thể hiện cách làm việc như thế nào : CClientDC clientDC(this);

int pixelFormat = ChoosePixelFormat(clientDC.m_hDC,&pfd);

BOOL success = SetPixelFormat(clientDC.m_hDC,pixelFormat,&pfd);

Trong dòng đầu tiên, chương trình lấy một DC cho vùng làm việc của cửa sổ ứng dụng.

Dòng thứ 2 gọi tới ChoosePixelFormat() rồi yêu cầu một chỉ số định dạng điểm ảnh cho một định dạng điểm ảnh gần nhất làm thành định dạng yêu cầu. Hai đối mục của hàm này là một điều khiển tới DC cho lựa chọn định dạng điểm ảnh và địa chỉ của cấu trúc PIXELFORMATDESCRIPTOR nắm giữ các thuộc tính của định dạng điểm ảnh yêu cầu. Nếu hàm gọi sai, ChoosePixelFormat() trả lại 0, ngược lại nó trả lại chỉ số định dạng điểm ảnh.

Dòng thứ 3 trong đoạn mã đơn giản gọi SetPixelFormat() để đặt định dạng điểm ảnh. Ba đối mục của hàm này là điều khiển tới DC, chỉ số định dạng điểm ảnh và địa chỉ của cấu trúc PIXELFORMATDESCRIPTOR. Hàm trả lại giá trị TRUE nếu thành công, ngược lại nó trả lại FALSE.

3.2.6. Tạo lập ngữ cảnh diễn tả

Trong phương pháp thứ hai của việc quản lý ngữ cảnh diễn tả, chương trình tạo lập và gỡ bỏ DC của Window mỗi lần chương trình phải vẽ lên cửa sổ, do đó chương trình không cần phải giữ DC cho toàn chương trình trong lúc chạy. Mỗi lần chương trình tạo lập DC, dù sao, nó cũng phải nắm ngữ cảnh diễn tả thành hiện hành.

Một cách triển khai của phương pháp thứ hai của việc quản lý ngữ cảnh diễn tả trong chương trình MFC có thể giống như sau :

```
int CxxxView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
        if (CView::OnCreate(lpCreateStruct) == -1) return -1;
        // TODO: Add your specialized creation code here
        PIXELFORMATDESCRIPTOR pfd=
```

```
        {
                sizeof(PIXELFORMATDESCRIPTOR),
                1,
                PFD_DRAW_TO_WINDOW |
                PFD_SUPPORT_OPENGL|PFD_TYPE_RGBA,
                24,
                0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,
                32,
                0,0,
                PFD_MAIN_PLANE,
                0,
                0,0,0
        };
        CClientDC clientDC(this);
        int pixelFormat=ChoosePixelFormat(clientDC.m_hDC,&pfd);
        BOOL success
=SetPixelFormat(clientDC.m_hDC,pixelFormat,&pfd);
        m_hRC=wglCreateContext(clientDC.m_hDC);          return 0;
        }
        //-----------------------------------------------------------------
        void CxxxView::OnDraw(CDC* pDC)
        {
                CChapter4Doc* pDoc = GetDocument();
                ASSERT_VALID(pDoc);
                // TODO: add draw code for native data here
        wglMakeCurrent(pDC->m_hDC,m_hRC);       DrawWithOpenGL();
        wglMakeCurrent(pDC->m_hDC,NULL);
        }
        //----------------------------------------------------------------- void
        CxxxView::OnDestroy()
        {
                CView::OnDestroy();
                // TODO: Add your message handler code here
        wglDeleteContext(m_hRC);
        }
```

Hàm OnCreate() đáp lại thông điệp WM_CREATE với việc đầu tiên lấy DC của vùng làm việc của cửa sổ, rồi gọi ChoosePixelFormat() để tìm lại chỉ số tới định dạng điểm ảnh cho DC và gọi SetPixelFormat() để đặt định dạng điểm ảnh.

Tiếp theo, OnCreate() gọi wglCreatContext() để tạo lập ngữ cảnh diễn tả. Đối mục đơn của hàm này là điều khiển của DC. Nếu thành công, nó trả về điều khiển tới ngữ cảnh diễn tả, ngược lại nó trả lại 0.

Trong đoạn chương trình này, OnCreate() tạo lập chỉ một DC tạm thời, bởi phạm vi địa phương của nó, nó tự động được xoá bỏ khi hàm kết thúc. Vì DC bị xoá bỏ, nó không thể làm ngữ cảnh diễn tả thành hiện hành ở điểm này trong chương trình.

Để cập nhật nội dung của cửa sổ ứng dụng, MFC gọi hàm OnDraw() của lớp quan sát. Bây giờ, nó gọi hàm OnDraw() để làm ngữ cảnh diễn tả thành hiện hành, nó thực hiện bằng lời gọi hàm wglMakeCurrent() sử dụng điều khiển của đối tượng DC chuyển tới hàm. Hàm này có 2 đối mục là điều khiển tới DC và điều khiển của ngữ cảnh diễn tả. Nếu thành công, wglMakeCurrent() trả lại TRUE, ngược lại nó trả về FALSE.

Sau khi làm ngữ cảnh diễn tả thành hiện hành, OmPaint() gọi hàm DrawWithOpenGL() để vẽ, sau đó gọi tới wglMakeCurrent() lần thứ hai với đối mục NULL là tham số thứ 2, làm ngữ cảnh diễn tả không còn là hiện hành. (Bạn phải tự viết lấy hàm DrawWithOpenGL() để biểu diễn những gì bạn muốn thể hiện).

Cuối cùng, khi ứng dụng đóng lại, MFC gọi hàm OnDestroy() của lớp quan sát. Trong OnDestroy(), chương trình chỉ gọi wglDeleteContext() để xoá bỏ ngữ cảnh diễn tả. Hàm này có đối mục đơn là điều khiển của ngữ cảnh diễn tả. Nếu thành công, hàm này trả về TRUE, ngược lại trả về FALSE. Bởi vì DC đã được tạo lập và bị phá phá huỷ trong thông điệp WM_PAINT, không có DC để xoá ở cuối chương trình..

3.2.7. Tạo lập thư viện hỗ trợ OpenGL với Visual C++

Trong lĩnh vực cơ khí có rất nhiều phần mềm cho phép người dùng có thể tiến hành xây dựng các mô hình và thực hiện mô phỏng với các mô hình 3 chiều (3D) và có thể tiến hành giải các bài toán phần tử hữu hạn trên các mô hình đó. Khi tiến hành xây dựng các mô hình đòi hỏi độ chính xác và khả năng thay đổi mô hình theo tham số đầu vào của một bài toán thì vẫn chưa thực hiện được. Đó là lý do tại sao với công việc nghiên cứu khi xây dựng các mô hình đòi hỏi thay đổi theo các thông số đầu vào, người nghiên cứu vẫn phải tiến hành tự lập lấy một chương trình riêng đáp ứng được yêu cầu công việc. Nhưng để xây dựng một chương trình mô phỏng 3D lại gặp nhiều khó khăn, hiện nay đa số mọi người sử dụng các thư viện đồ hoạ DirectX hoặc OpenGL. Với thư viện đồ hoạ OpenGL và dựa trên thư viện đó để xây dựng công cụ giúp quá trình lập

một chương trình mô phỏng 3D trở nên dễ dàng dùng cho các chương trình được lập bằng Visual C++.

5.3.7.1. Một số vấn đề khi sử dụng thư viện OpenGL

OpenGL là một thư viện đồ hoạ mở, bao gồm các hàm, các kiểu dữ liệu và các hằng số được định nghĩa sẵn. Thư viện này giúp người lập trình có được khả năng tiến hành xây dựng các mô hình 3D và thực hiện mô phỏng. Khi lập một đề án (project) cho một chương trình mới phải tiến hành khởi tạo môi trường cho OpenGL [22,23], đây là một công việc khá khó khăn và mất thời gian. Việc sử dụng và quản lý một chương trình với nhiều giao diện OpenGL lại càng phức tạp. Để sử dụng thư viện OpenGL trong một đề án cần thực hiện các bước :

1. Gộp các tệp tin tiêu đề (*.h) và tệp thư viện (*.lib) cần sử dụng của thư viện OpenGL vào đề án :

- Gộp các tệp tin tiêu đề vào đề án :

```
#include <gl\gl.h>
#include <gl\glu.h>
```

Các tệp tin thư viện :

```
opengl32.lib
glu32.lib
```

2. Khởi tạo môi trường của OpenGL ứng với một ngữ cảnh :

-Khởi tạo cấu trúc (struct) điểm ảnh PIXELFORMATDESCRIPTOR, cấu trúc này có 26 trường thông tin. Sau đó lựa chọn định dạng đó cho ngữ cảnh muốn khởi tạo bằng hàm ChoosePixelFormat().

-Truyền địa chỉ của cấu trúc vừa khởi tạo cho hàm SetPixelFormat().

-Tạo lập ngữ cảnh diễn tả cho OpenGL với hàm wglCreateContext().

3. Thực hiện các thao tác vẽ hoặc dựng mô hình :

Mỗi lần muốn thực hiện vẽ với OpenGL, cần phải thực hiện gọi ngữ cảnh của OpenGL thành hiện hành và sau khi vẽ xong thì cần đặt trả nó thành không hiện hành với hàm wglMakeCurrent().

4. Khi kết thúc chương trình hoặc giao diện OpenGL cần phải xoá bỏ ngữ cảnh để giải phóng bộ nhớ với hàm wglDeleteContext().

5.3.7.2. Xây dựng lớp điều khiển COpenGLCtrl

Khi tạo một đề án mới phải thực hiện lặp lại những công việc trên sẽ rất mất thời gian và khởi tạo môi trường sao cho hợp lý là không đơn giản. Trên cơ sở đó xây dựng lớp điều khiển COpenGLCtrl kế thừa từ lớp CWnd của MFC (MFC : thư viện lớp nền tảng của Microsoft), đã đóng gói toàn bộ các bước trên, do đó người lập trình không phải quan tâm tới các công việc đó nữa [23]. Lớp COpenGLCtrl cho phép người lập trình sử dụng thư viện OpenGL đơn giản, nhanh chóng và hiệu quả với một số chức năng thuận tiện trong quá trình thao tác trên giao diện như : di chuyển (pan), quay (rotate), phóng to thu nhỏ (zoom), đặt vị trị quan sát theo các hướng trên (top), dưới (bottom), trái (left), phải (right), phía trước (front) và phía sau (back). Lớp này còn cho phép khởi tạo điều khiển COpenGLCtrl theo nhiều cách khác nhau :

- Khởi tạo trực tiếp : dùng một trong hai phương thức sau BOOL COpenGLCtrl::Create(const RECT& rect, CWnd* parent,

UINT nID, DWORD dwStyle = WS_CHILD |WS_TABSTOP | WS_VISIBLE);

```
BOOL COpenGLCtrl::CreateEx(
        const RECT& rect,
        CWnd* pParentWnd, UINT nID,
        DWORD dwStyle = WS_CHILD  |WS_TABSTOP | WS_VISIBLE,
        DWORD dwStyleEx= WS_EX_CLIENTEDGE);
```

- Khởi tạo từ một điều khiển tĩnh (Static Control) có trên giao diện :

BOOL CreateFromStatic(UINT nID, CWnd* pParent); trong đó nID là chỉ danh của điều khiển Static đã có trên giao diện (phải khác IDC_STATIC).

- Khởi tạo từ điều khiển khách (Custom Control) có trên giao diện :

Tạo một điều khiển Custom đặt trên giao diện và đặt các thuộc tính :

ID : chỉ danh của điều khiển, tuỳ đặt sao cho dễ nhớ và dễ hiểu.

Class : bắt buộc phải điền vào "MFCOpenGLCtrl".

Trong mã nguồn của lớp quản lý giao diện trên bổ xung các dòng :

COpenGLCtrl m_OpenGLCtrl;

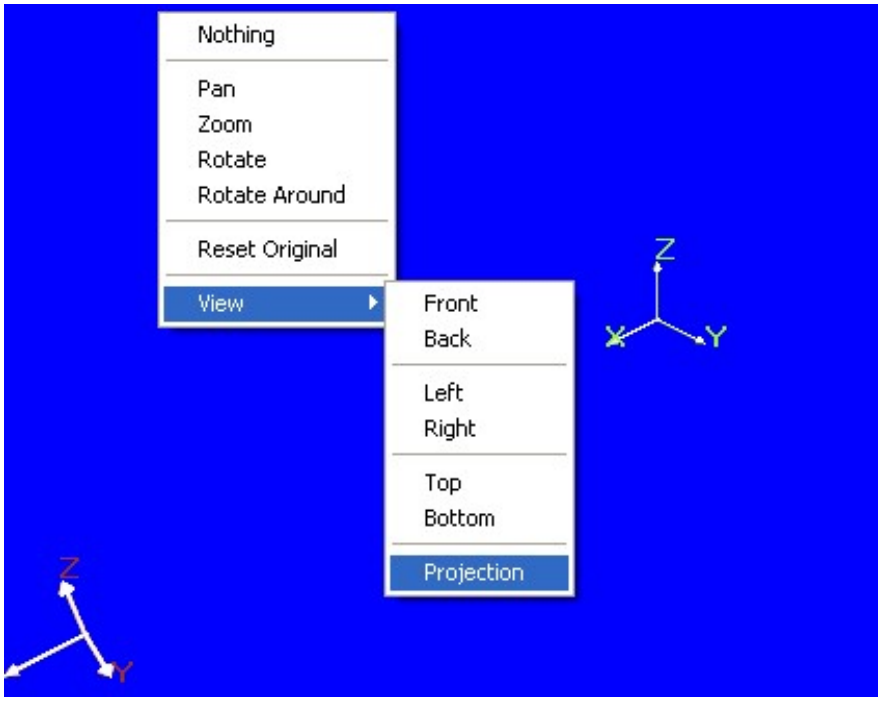
đặt trong khai báo của lớp quản lý giao diện.

DDX_Control(pDX, IDC_CUSTOM_PREVIEW, m_OpenGLCtrl); đặt trong triển khai hàm

DoDataExchange (CDataExchange* pDX)

của lớp quản lý giao diện, với IDC_CUSTOM_PREVIEW là chỉ danh của điều khiển Custom vừa tạo, m_OpenGLCtrl là biến thuộc lớp quản lý giao diện có kiểu COpenGLCtrl.

Ngay sau khi khởi tạo xong, khi thực thi chương trình sẽ cho giao diện của COpenGLCtrl như trên Hình 3.9.



Hình 3.9: Giao diện ban đầu của COpenGLCtrl

Để thực hiện vẽ mô hình lên điều khiển COpenGLCtrl tiến hành theo một trong hai cách:

- Cách 1 : Quá trình vẽ được đặt trong một hàm độc lập, có cú pháp void FunctionDrawOpenGLCtrl(COpenGLCtrl *);

Với FunctionDrawOpenGLCtrl là tên hàm tuỳ đặt. Sau đó truyền con trỏ hàm này tới COpenGLCtrl, COpenGLCtrl sẽ tự động gọi tới hàm đó khi cần thể hiện mô hình. Truyền con trỏ hàm thực hiện quá trình vẽ cho COpenGLCtrl sử dụng phương thức:

void COpenGLCtrl::SetFunctionDrawOpenGL(FunctionDrawOpenGLCtrl f);

trong đó FunctionDrawOpenGLCtrl là kiểu con trỏ cho các hàm có định dạng hàm nêu trên.

- Cách 2 : Tạo một lớp điều khiển kế thừa từ lớp COpenGLCtrl, quá trình vẽ được đặt trong hàm ảo, có cú pháp :

virtual void COpenGLCtrl::DrawInOpenGL();

5.3.7.3. Xây dựng một số lớp và hàm tiện ích

Để xây dựng chương trình với mô hình động, tức là có hình dạng và kích thước thay đổi theo các thông số người dùng đưa vào thì người lập trình phải tự dựng mô hình trực tiếp trong chương trình, đây là một công việc rất phức tạp. Nhằm tạo thuận tiện cho quá trình xây dựng mô hình, trong thư viện cung cấp một số lớp và hàm tiện ích, dưới đây là một số lớp, cấu trúc và hàm chính :

GL_LIGHT : cấu trúc lưu trữ các biến thành phần khởi tạo nguồn sáng.

GL_MATERIAL : cấu trúc lưu trữ các biến thành phần khởi tạo vật liệu.

CPoint2d, CPline : lớp lưu trữ điểm trong mặt phẳng và lớp lưu trữ kiểu đường Polyline.

CRegionGL : lớp tạo và vẽ một miền phẳng, giúp thực hiện vẽ một miền phẳng bất kỳ, có thể khoét các lỗ trên đó.

DrawTriangles, DrawQuads : các hàm vẽ các hình tam giác hoặc tứ giác, đó là các hình cơ bản để ghép lại thành một mô hình dạng bề mặt.

ExtrudePline, ExtrudeRegion : các hàm thực hiện dựng lên một hình trụ từ một đường đa tuyến (polyline) hoặc miền (region).

RevolvePline, RevolveRegion : các hàm thực hiện tạo một hình tròn xoay từ một đường đa tuyến (polyline) hoặc miền (region).

Ngoài ra còn có nhiều lớp và hàm tiện ích khác.

5.3.7.4. Đóng gói công cụ

Các công cụ trên đã được tiến hành đóng gói thành các thư viện liên kết động (Dynamic Linked Library : DLL). Thư viện được cung cấp gồm các tệp tin :

*.h : các tệp tin tiêu đề, dùng để gộp (include) các khai báo vào đề án.

OpenGL.lib, CreateModel.lib : tệp tin thư viện, chứa các địa chỉ liên kết của thư viện. Tệp tin này được đưa vào đề án để trình biên dịch C++ tiến hành liên kết chương trình với tệp tin thư viện liên kết động (DLL).

OpenGL.dll, CreateModel.dll : tệp tin thư viện liên kết động, các tệp tin này được đặt cùng thư mục của chương trình thực thi (*.exe) và được dùng tới khi chạy chương trình thực thi đó.

Nhờ thực hiện việc đóng gói này, khi người lập trình tạo một chương trình mới chỉ cần sao chép các tệp tin trên tới đề án mới và tiến hành khởi tạo như đã nêu ở trên. Điều đó giúp người lập trình xây dựng chương trình mới dễ dàng và nhanh chóng.

3.2.8. Sử dụng thư viện OpenGLSetting

5.3.8.1. Nhiệm vụ của thư viện này

Đặt vật liệu và chiếu sáng với nhiều tham số khá rắc rối, đặc biệt là khi phải thiết đặt chúng một cách thủ công.

Thư viện OpenGLSetting cho phép thiết đặt tham số vật liệu và ánh sáng thông qua giao diện hộp thoại một cách tường minh.

Trong thư viện này định nghĩa sẵn 2 hộp thoại:

CDlgSettingLight - Hộp thoại thiết đặt tham số ánh sáng

CDlgSettingMaterial - Hộp thoại thiết đặt tham số vật liệu Kỹ thuật lập trình mô phỏng Rô bốt và các hệ cơ điện tử

5.3.8.2. Đưa thư viện vào đề án

Copy thư mục "OpenGLSetting" vào thư mục của project đã tạo.

Đưa file"OpenGLSetting\bin\OpenGLSetting.lib"hoặc"OpenGLSettingd .lib" (tùy theo chế độ biên dịch của đề án) vào đề án bằng chức năng "Add Files to Project …"

Trong file "pch.h" thêm dòng sau

#include "OpenGLSetting/Inc/OpenGLSetting.h"

Khai báo sử dụng hộp thoại thiết đặt ánh sáng:

CDlgSettingLight dlg;

dlg.SetData(m_glLight, &m_openGL);

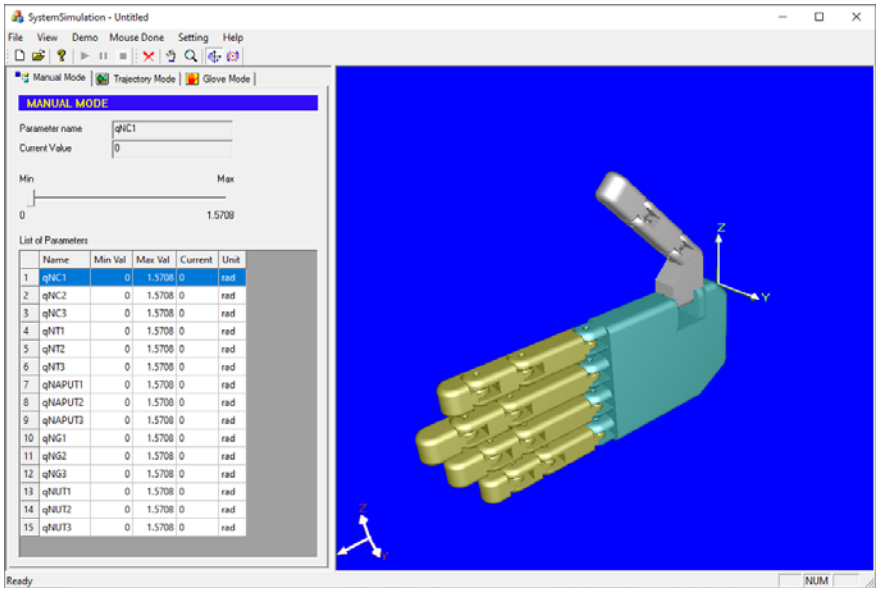
dlg.DoModal();

Sử dụng hộp thoại đặt vật liệu CDlgSettingMaterial tương tự.

3.3. Xây dựng phần mềm mô phỏng 3D tay máy robot

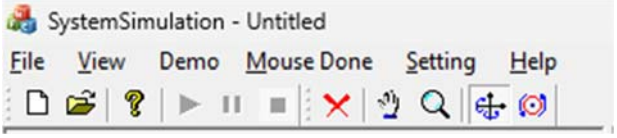
Chương trình "System Simulation" được lập trình bằng ngôn ngữ C++ trên môi trường Microsoft Visual Studio.

Để thực thi chương trình, người dùng mở tệp "System Simulation", giao diện ban đầu của chương trình như Hình 3.10.



Hình 3.10: Giao diện chương trình

3.3.1. Giới thiệu về các công cụ chính của chương trình



Để có thể mở file mới ta có thể nhấm vào biểu tưởng hoặc có thể nhấn phím tắt Ctrl+N để cho nhanh.

: mở một tệp tin tài liệu được tạo ra và được lưu trên các thiết bị lưu trữ

Để có thể thay đổi công dụng của con chuột ta chọn "Mouse Done". Khi nhấn vào đây ta có 5 công cụ khác nhau:

- : dùng để duy chuyển mô hình xung quanh môi trường chương trình.

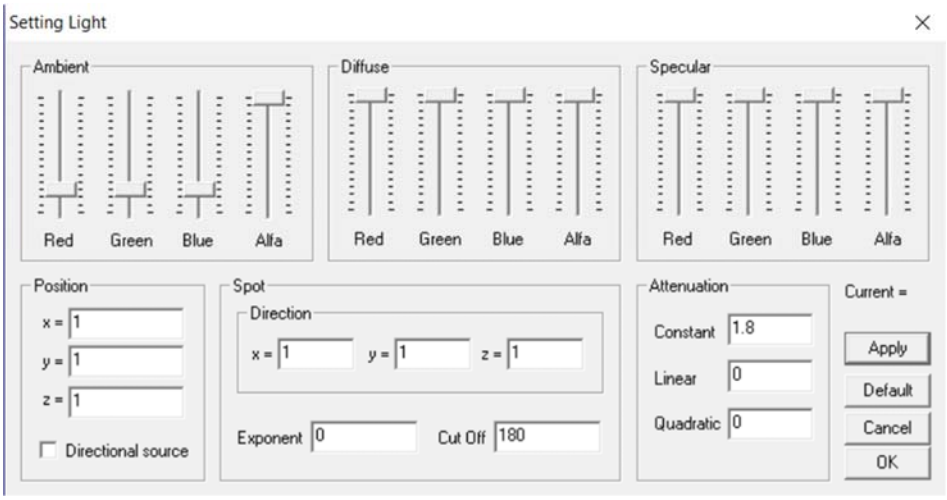
- : dùng để phóng to hoặc thu nhỏ mô hình 3D .

- : dùng để xoay chuyển mô hình theo ý muốn của người dùng.

- : dùng để xoay vật theo hướng vòng tròn.

- : dùng để xóa hết các lệnh mà người dùng sử dụng lúc trước.

Để có thể lựa chọn màu sắc cho mô hình ta nhấn vào phần "Setting" và chọn phần "Light" còn nếu muốn lựa chọn nguyên vật liệu cho mô hình ta chọn phần "Material" (Hình 3.11 và 3.12).



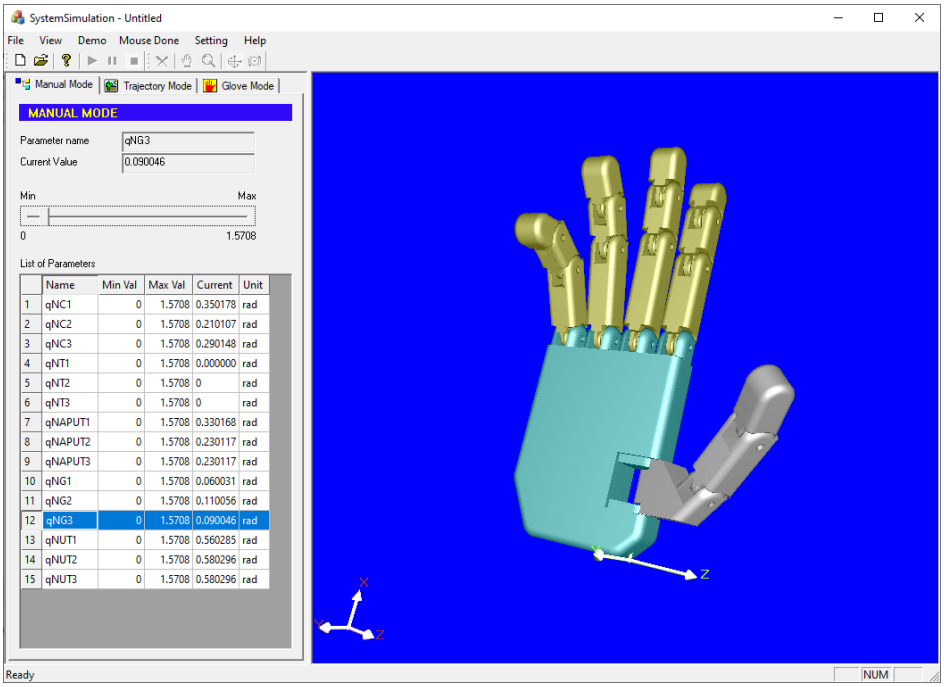
Hình 3.11: Giao diện hộp thoại cài đặt thông số ánh sáng Setting Light

Hình 3.12: Giao diện hộp thoại lựa chọn đối tượng đặt thông số vật liệu

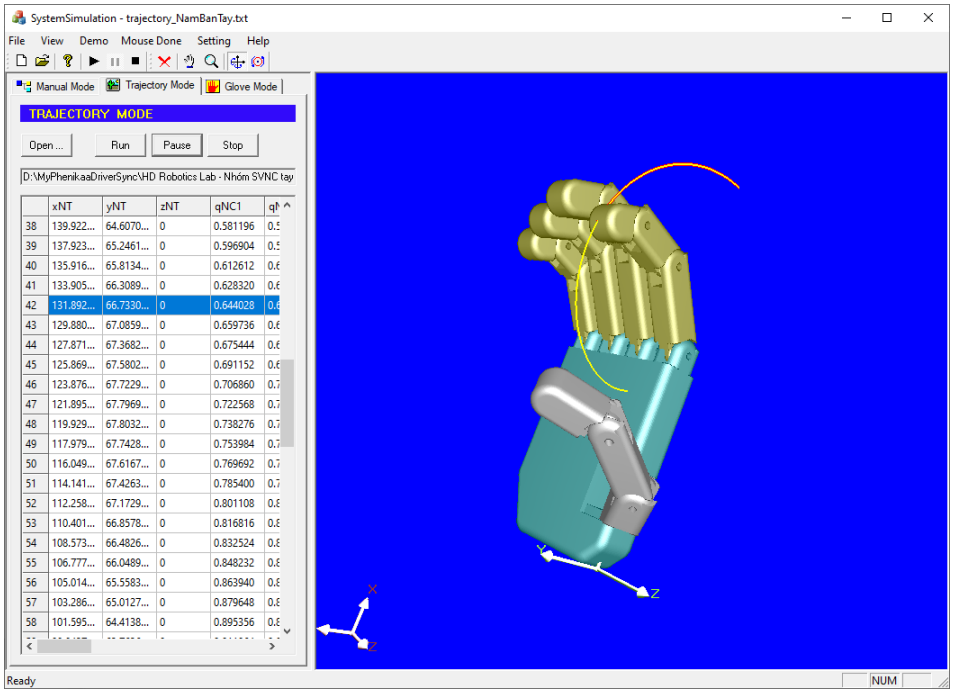
Trong chương trình gồm 3 chế độ đó là chế độ thủ công(Manual Mode) và chế độ quỹ đạo(Trajectory Mode).

Manual Mode (Hình 3.13) cho phép xem được quỹ đạo chuyển động của từng đốt ngón tay của mô hình bàn tay. Khi chọn vào từng phần của bảng số liệu, người dùng có thể điều khiển được từng đốt ngón tay chuyển động bằng cách kéo thanh trượt, giúp biết được giới hạn di chuyển và cách di chuyển của từng đốt.

Trajectory Mode (Hình 3.14) cho phép xem quỹ đạo chuyển động của bàn tay robot bằng cách nạp vào một đoạn chương trình với các giá trị biến khớp đã được tính toán sẵn để kích hoạt chuyển động của bàn tay. Phụ lục 2 là ví dụ của một tệp tin của thao tác nắm bàn tay có thể được mở bằng cách kéo thả hoặc mở bằng nút "Open".

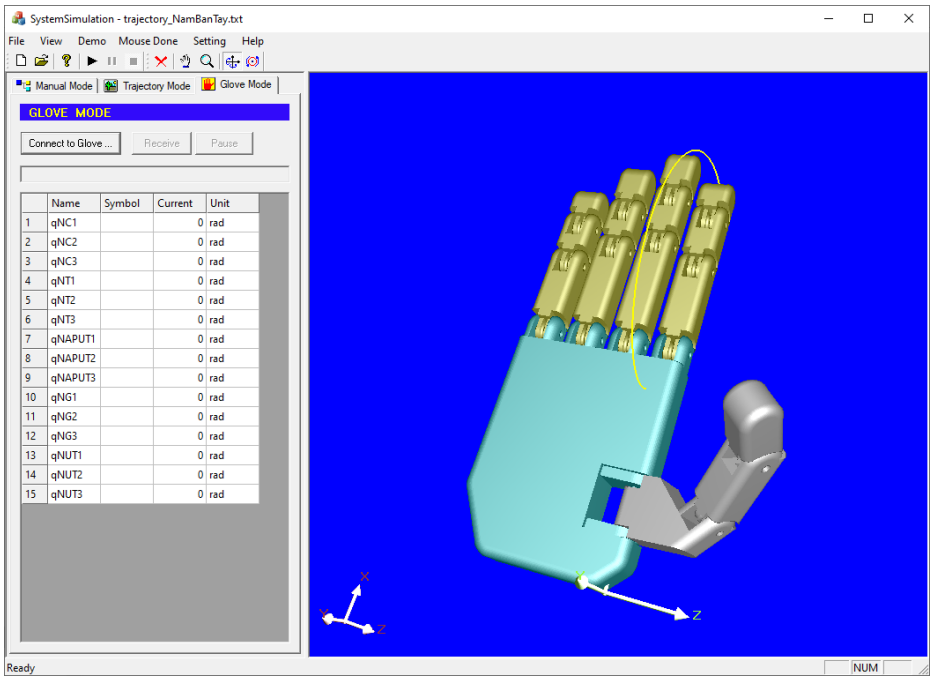


Hình 3.13: Giao diện phần mềm ở chế độ Manual Mode



Hình 3.14: Giao diện phần mềm ở chế độ Trajectory Mode

Glove Mode (Hình 3.15) cho phép kết nối với găng tay điện tử (sẽ được giới thiệu ở chương sau) để điều khiển mô phỏng lại chuyển động thực tế của bàn tay người điều khiển. Việc kết nối được thực hiện bằng nút "Connect to Glove ...", khi đó sẽ hiển thị hộp thoại để lựa chọn cổng kết nối và tốc độ truyền dữ liệu.



Hình 3.15: Giao diện phần mềm ở chế độ Glove Mode

3.3.2. Tệp tin cấu hình quản lý mô hình tay máy robot.

Mô hình bàn tay robot khi được xuất ra tập tin STL gồm rất nhiều bộ phận và chi tiết khác nhau. Tất cả các bộ phận cần sự chuyển động độc lập phải được xuất ra các tập tin STL riêng. Hơn nữa các bộ phận dù không chuyển động độc lập mà muốn thể hiện sự khác biệt về mầu sắc và vật liệu cũng cần được xuất ra các tập tin STL khác nhau. Do vậy, trong chuyên đề tác giả đã tạo ra một tệp tin để quản lý mô hình có cấu trúc định dạng tuân thủ theo cấu trúc được giới thiệu trong bài báo [24,25]. Việc đọc định dạng tệp tin này được thực hiện dễ dàng nhờ các hàm đã được xây dựng và giới thiệu trong bái báo đó. Theo định dạng, tệp tin quản lý mô hình có cấu trúc như thể hiện ở Phụ lục 1. Trong đó, các ký tự ở sau dấu chấm phẩy ";" trên mỗi một dòng là phần chú thích, không chứa dữ liệu. Các từ được đặt trong các cặp dấu [] là các từ khóa để tra cứu theo vùng dữ liệu. Từ khóa "workspace" xác định vùng dữ liệu quy định việc hiển thị không gian làm việc thực hiện việc hiển thị và mô phỏng mô hình. Từ khóa "elements" chỉ ra vùng dữ liệu cung cấp các từ khóa thực hiện tra cứu thông tin dữ liệu các đốt ngón tay. Từ phần dữ liệu này, ta sẽ có được các ma trận chuyển của các đốt ngón tay là MBT0, MNC1…, còn dữ liệu về các tệp tin chứa các chi tiết được quy định bởi các từ khóa BT0, NC1..., Trong đó phần dữ liệu với từ khóa BT0 và NC1 sẽ chứa thông tin của bàn tay và đốt ngón tay. Các ma trận chuyển đặt trong vùng dữ liệu T được cung cấp ở dạng công thức tham số, phần mềm mô phỏng sẽ đọc và tính toán các ma trận chuyển này dựa theo các giá trị của biến khớp và hằng số được cung cấp. Các ma trận T có thể được xây dựng dựa theo ma trận Denavit Hartenberg [20]. Từ khóa "parameters" cung cấp vùng dữ liệu của các biến khớp trong mô hình. Mỗi một dòng trong vùng này sẽ là thông tin của một biến khớp. Từ khóa "constants" chỉ vùng dữ liệu chứa các thông số dạng hằng số trong mô hình. Dựa vào thông tin của các tham số biến khớp và hằng số trong mô hình, các ma trận chuyển của các khâu có thể được tính toán phục vụ việc hiển thị các chi tiết đúng vị trí. Hệ thống các trục tọa độ, tham số và biến khớp được mô tả trên Hình 2.9.

3.3.3. Kết quả chương trình mô phỏng bàn tay robot .

Kết quả cuối cùng của chương trình mô phỏng bàn tay robot có giao diện như trên Hình 3.13 và 3.15. Khi chạy chương trình mô phỏng, người dùng sẽ có giao diện

điều khiển mô hình theo các biến khớp (Hình 3.13) hoặc chạy mô phỏng theo quỹ đạo yêu cầu cho trước (Hình 3.14). Trong đó, quỹ đạo được cho trước là một đường hình cung đi theo quỹ đạo của ngón trỏ giống như gập tay người, từ đó các biến đốt ngón tay đã được giải bằng excel và xuất kết quả ra tệp tin với định dạng như ở Phụ lục 2. Trong đó ba cột đầu thể hiện tọa độ của Ngón trỏ, mười lăm cột sau thể hiện giá trị của các biến đốt ngón tay tại mỗi thời điểm ứng với vị trí gập của các đốt ngón tay . Trong chế độ Trajectory Mode, sau khi người dùng mở tệp tin quỹ đạo trên, có thể chọn các nút Run, Pause hoặc Stop để thực hiện chạy mô phỏng hoạt động của bàn tay robot.

Chương 4. Lập trình phần mềm điều khiển

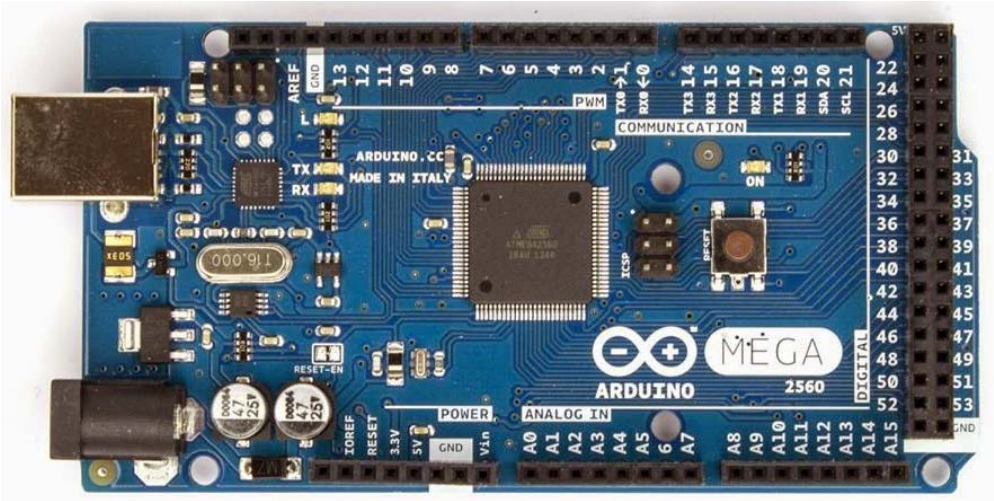
4.1. Arduino mega 2560.

Arduino là một bo mạch vi điều khiển do một nhóm giáo sư và sinh viên nước Ý thiết kế và đưa ra đầu tiên vào năm 2005. Mạch Arduino được sử dụng để cảm nhận và điều khiển nhiều đối tượng khác nhau. Nó có thể thực hiện nhiều nhiệm vụ lấy tín hiệu từ cảm biến đến điều khiển đèn, động cơ, và nhiều đối tượng khác. Ngoài ra mạch còn có khả năng liên kết với nhiều module khác nhau như module đọc thẻ từ, ethernet shield, sim900A, ….để tăng khả ứng dụng của mạch [26].

Trong đó, Arduino mega 2560 nổi bật lên với kích thước , dung lượng RAM lớn cũng như nhiều chân xuất tín hiệu I/O, thích hợp cho những dự án yêu cầu khối lượng chương trình lớn cũng như như yêu cầu nhiều chân tín hiệu.

Bảng 4.1: Bảng thông số kỹ thuật Arduino mega 2560 R3

Arduino Mega	**Tính năng, đặc điểm**
Vi điều khiển	AVR ATmega 2560 (8bit)
Nguồn cung cấp	7-12V (Bộ điều chỉnh sẵn có cho bộ điều khiển)
Số chân I/O số	54
Số chân I/O tương tự	16
Xung clock	16 MHz (nhà sản xuất cài đặt là 1MHz)
Bộ nhớ flash	128 KB
SRAM	8 KB
Giao tiếp	USB (Lập trình với ATmega 8), ICSP (lập trình), SPI, I2C và USART
Bộ Timer	2 (8bit) + 4 (16bit) = 6 Timer
PWM	12 (2-16 bit)
ADC	16 (10 bit)
USART	4
Ngắt thay đổi chân	24



Hình 4.1: Arduino mega 2560 R3

4.2. Cảm biến góc MPU6050.

Cảm biến góc GY-521 MPU6050 là một cảm biến sáu trục, có chứa một gia tốc 3 trục con quay hồi chuyển 3 trục. Hoạt động với 3.3V và giao tiếp I2C với tốc độ tối đa 400kHz.

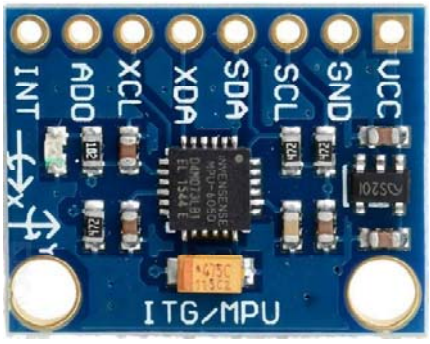
Các cảm biến bên trong MPU-6050 sử dụng bộ chuyển đổi tương tự – số (Anolog to Digital Converter – ADC) 16-bit cho ra kết quả chi tiết về góc quay, tọa độ…

Tùy thuộc vào yêu cầu của bạn, cảm biến MPU-6050 có thể hoạt động ở chế độ tốc độ xử lý cao hoặc chế độ đo góc quay chính xác (chậm hơn). MPU-6050 có khả năng đo ở phạm vi:

+ con quay hồi chuyển: ± 250 500 1000 2000 dps

+ gia tốc: ± 2 ± 4 ± 8 ± 16g

Hơn nữa, MPU-6050 có sẵn bộ đệm dữ liệu 1024 byte cho phép vi điều khiển phát lệnh cho cảm biến, và nhận về dữ liệu sau khi MPU-6050 tính toán xong.



Hình 4.2: Cảm biến góc MPU6050

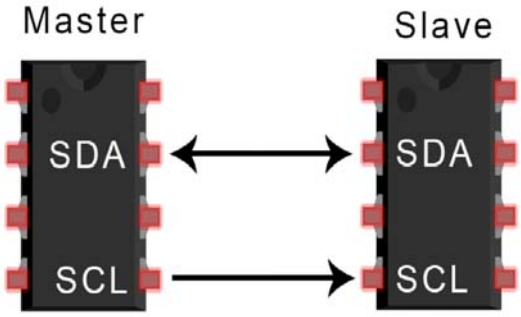
Bảng 4.2: Sơ đồ chân MPU6050

Ký hiệu	Ý nghĩa
VCC	5V/3V3
GND	Chân nối đất
SCL	Chân SCL trong giao tiếp I2C
SDA	Chân SDA trong giao tiếp I2C
XDA	Chân dữ liệu (kết nối với cảm biến khác)
XCL	Chân xung (kết nối với cảm biến khác)
AD0	Bit0 của địa chỉ I2C
INT	Chân ngắt

4.3. Giao tiếp I2C.

4.3.1. Tổng quan về giao tiếp I2C.

I2C là giao thức truyền thông nối tiếp đồng bộ phổ biến hiện nay, được sử dụng rộng rãi trong việc kết nối nhiều IC với nhau, hay kết nối giữa IC và các ngoại vi với tốc độ thấp. I2C kết hợp các tính năng tốt nhất của SPI và UART. Với I2C, bạn có thể kết nối nhiều slave với một master duy nhất (như SPI) và bạn có thể có nhiều master điều khiển một hoặc nhiều slave. Điều này thực sự hữu ích khi bạn muốn có nhiều hơn một vi điều khiển ghi dữ liệu vào một thẻ nhớ duy nhất hoặc hiển thị văn bản trên một màn hình LCD.



Hình 4.3: Truyền dữ liệu I2C

Giống như giao tiếp UART, I2C chỉ sử dụng hai dây để truyền dữ liệu giữa các thiết bị:

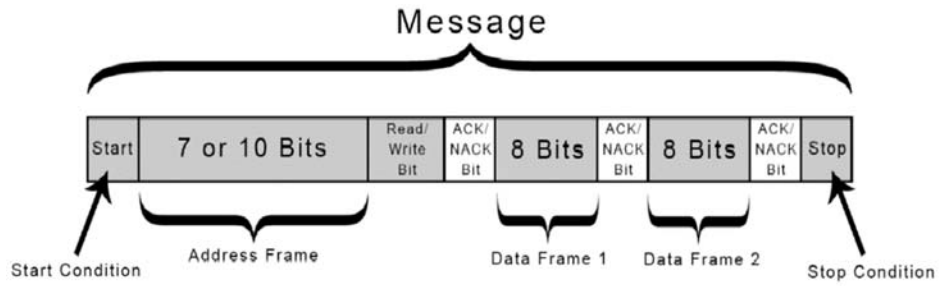
SDA (Serial Data) - đường truyền cho master và slave để gửi và nhận dữ liệu.

SCL (Serial Clock) - đường mang tín hiệu xung nhịp.

I2C là một giao thức truyền thông nối tiếp, vì vậy dữ liệu được truyền từng bit dọc theo một đường duy nhất (đường SDA).

4.3.2. Cách hoạt động của I2C.

Với I2C, dữ liệu được truyền trong các tin nhắn. Tin nhắn được chia thành các khung dữ liệu. Mỗi tin nhắn có một khung địa chỉ chứa địa chỉ nhị phân của địa chỉ slave và một hoặc nhiều khung dữ liệu chứa dữ liệu đang được truyền. Thông điệp cũng bao gồm điều kiện khởi động và điều kiện dừng, các bit đọc / ghi và các bit ACK / NACK giữa mỗi khung dữ liệu:



Hình 4.4: Thanh ghi

Điều kiện khởi động: Đường SDA chuyển từ mức điện áp cao xuống mức điện áp thấp trước khi đường SCL chuyển từ mức cao xuống mức thấp.

Điều kiện dừng: Đường SDA chuyển từ mức điện áp thấp sang mức điện áp cao sau khi đường SCL chuyển từ mức thấp lên mức cao.

Khung địa chỉ: Một chuỗi 7 hoặc 10 bit duy nhất cho mỗi slave để xác định slave khi master muốn giao tiếp với nó.

Bit Đọc / Ghi: Một bit duy nhất chỉ định master đang gửi dữ liệu đến slave (mức điện áp thấp) hay yêu cầu dữ liệu từ nó (mức điện áp cao).

Bit ACK / NACK: Mỗi khung trong một tin nhắn được theo sau bởi một bit xác nhận / không xác nhận. Nếu một khung địa chỉ hoặc khung dữ liệu được nhận thành công, một bit ACK sẽ được trả lại cho thiết bị gửi từ thiết bị nhận.

4.3.3. Các bước truyền dữ liệu

1. Master gửi điều kiện khởi động đến mọi slave được kết nối bằng cách chuyển

đường SDA từ mức điện áp cao sang mức điện áp thấp trước khi chuyển đường SCL từ mức cao xuống mức thấp.

2. Master gửi cho mỗi slave địa chỉ 7 hoặc 10 bit của slave mà nó muốn giao tiếp, cùng với bit đọc / ghi.
3. Mỗi slave sẽ so sánh địa chỉ được gửi từ master với địa chỉ của chính nó. Nếu địa chỉ trùng khớp, slave sẽ trả về một bit ACK bằng cách kéo dòng SDA xuống thấp cho một bit. Nếu địa chỉ từ master không khớp với địa chỉ của slave, slave rời khỏi đường SDA cao.
4. Master gửi hoặc nhận khung dữ liệu
5. Sau khi mỗi khung dữ liệu được chuyển, thiết bị nhận trả về một bit ACK khác cho thiết bị gửi để xác nhận đã nhận thành công khung.
6. Để dừng truyền dữ liệu, master gửi điều kiện dừng đến slave bằng cách chuyển đổi mức cao SCL trước khi chuyển mức cao SDA.

4.3.4. Ưu và nhược điểm của giao tiếp I2C.

Có rất nhiều điều ở I2C có thể khiến nó nghe có vẻ phức tạp so với các giao thức khác, nhưng có một số lý do chính đáng khiến chúng ta có thể muốn hoặc không muốn sử dụng I2C để kết nối với một thiết bị cụ thể:

Ưu điểm:

- Chỉ sử dụng hai dây
- Hỗ trợ nhiều master và nhiều slave
- Bit ACK / NACK xác nhận mỗi khung được chuyển thành công
- Phần cứng ít phức tạp hơn so với UART
- Giao thức nổi tiếng và được sử dụng rộng rãi

Nhược điểm:

- Tốc độ truyền dữ liệu chậm hơn SPI
- Kích thước của khung dữ liệu bị giới hạn ở 8 bit
- Cần phần cứng phức tạp hơn để triển khai so với SPI

4.4. Module mở rộng giao tiếp I2C TCA9548A

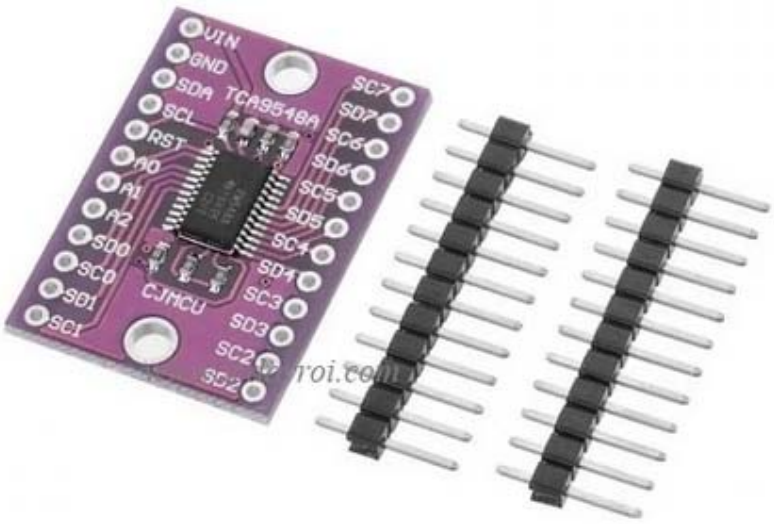
Nếu giao tiếp I2C dựa vào địa chỉ cố định của các cảm biến. Vậy nếu ta muốn điều khiển nhiều cảm biến có địa chỉ giống nhau bằng giao tiếp I2C thì điều đó là không thể nào. Vì vậy, Module TCA9548A được ra đời để giải quyết vấn đề này. Module mở rộng giao tiếp I2C 8 kênh TCA9548A Là module để mở rộng số thiết bị kết nối đến một kênh giao tiếp I2C của vi điều khiển. Mở rộng tối đa lên tới 8 kênh. Giúp bạn có thể kết nối đến tới 8 thiết bị I2C có cùng địa chỉ chỉ với 1 kênh I2C duy nhất. Nếu muốn nhiều hơn bạn có thể ghép nối Hieu module lại để kết nối lên tới 64 thiết bị cùng địa chỉ.

Sơ đồ chân TCA9548A được thể hiện chi tiết trong Bảng 4.3 và Hình 4.5.

Bảng 4.3: Sơ đồ chân TCA9548A

Ký hiệu	Miêu tả
VIN	Cấp nguồn cho bộ ghép kênh
GND	Kết nối với GND
SDA	Kết nối với chân SDA của vi điều khiển chính
SCL	Kết nối với chân SCL của vi điều khiển chính
RST	Chân RST thấp đang hoạt động—có thể được sử dụng để đặt lại bộ ghép kênh
A0	Chọn địa chỉ I2C của bộ ghép kênh—kết nối với GND hoặc VCC
A1	Chọn địa chỉ I2C của bộ ghép kênh—kết nối với GND hoặc VCC
A2	Chọn địa chỉ I2C của bộ ghép kênh—kết nối với GND hoặc VCC
SD0	SDA cho kênh 0
SC0	SCL cho kênh 0
SD1	SDA cho kênh 1
SC1	SCL cho kênh 1
SD2	SDA cho kênh 2
SC2	SCL cho kênh 2

SD3	SDA cho kênh 3
SC3	SCL cho kênh 3
SD4	SDA cho kênh 4
SC4	SCL cho kênh 4
SD5	SDA cho kênh 5
SC5	SCL cho kênh 5
SD6	SDA cho kênh 6
SC6	SCL cho kênh 6
SD7	SDA cho kênh 7
SC7	SCL cho kênh 7



Hình 4.5: Module mở rộng giao tiếp I2C TCA9548A

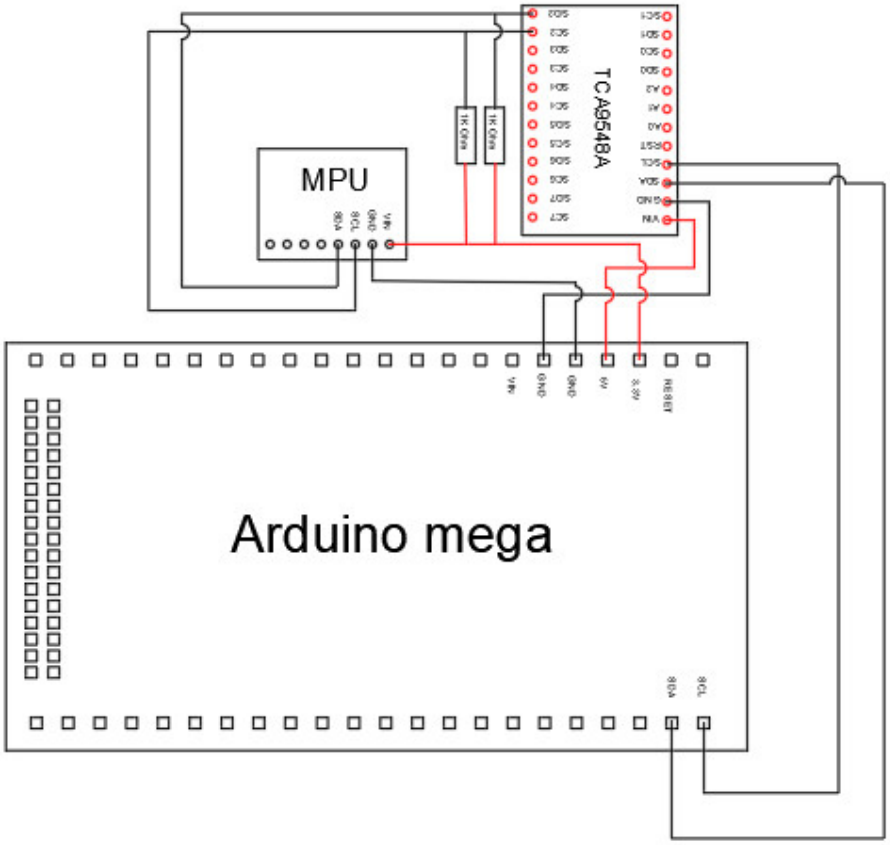
4.5. Găng tay

4.5.1. Sơ đồ kết nối mạch

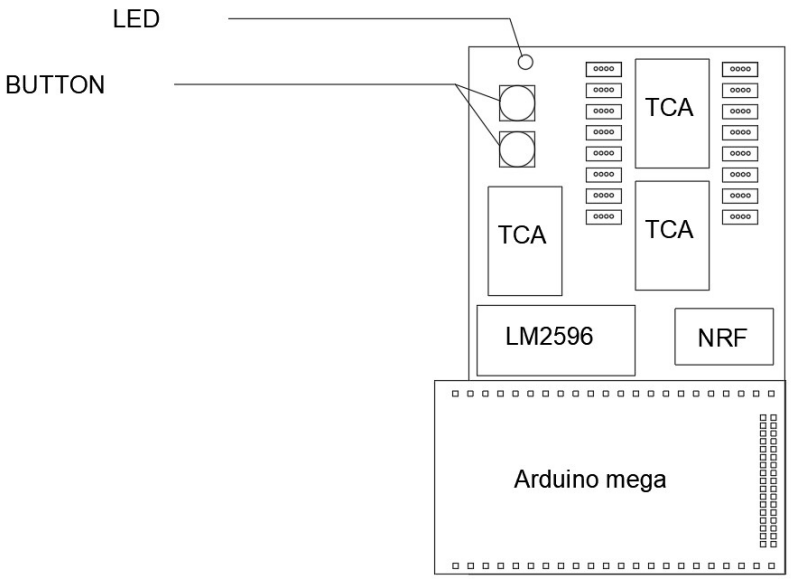
Với mục tiêu sử dụng ít dây nối nhất có thể, làm mạch trở nên dễ nhìn và vẫn đạt hiệu quả cao, tận dụng tối đa chân cắm hiện có của arduino cũng như cảm biến. Chúng ta có sơ đồ dây giữa các cảm biến và vi điều khiển như Hình 4.6.

Với số lượng cảm biến lớn cùng nhiều module chuyển đổi địa chỉ I2C, việc phân bố vị trí các module và tích hợp chúng sao cho mạch vừa dễ gọn mà lại có thể đi

cùng với găng tay, tích hợp đầy đủ chức năng điều khiển tắt, mở, nhận dữ liệu và ngừng nhận dữ liệu. Sơ đồ ở Hình 4.7 biểu diễn phân bố mạch điều khiển chính trên găng tay.



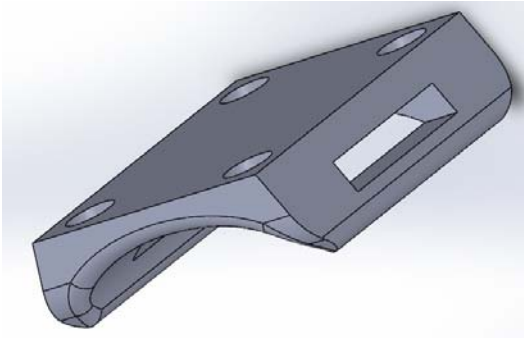
Hình 4.6: Phân bố tổng thể các thành phần điều khiển của găng tay



Hình 4.7: Phân bố mạch trên găng tay

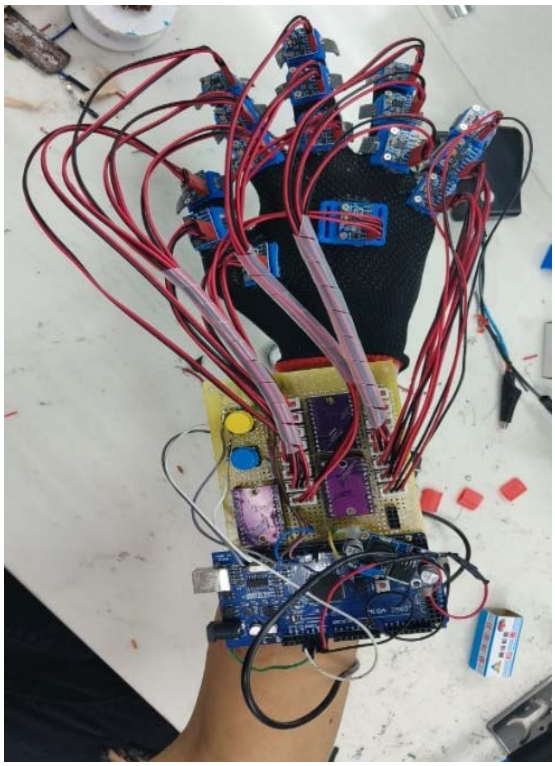
4.5.2. Ý tưởng thiết kế

Để gắn cảm biến vào găng tay cần phải thiết kế một giá đỡ cảm biến . Những giá đỡ cảm biến đã được tùy chỉnh theo kích cỡ của cảm biến và kích thước của từng đốt ngón tay. Những giá đỡ này sẽ giúp đỡ cho cảm biến hoạt động một cách tốt hơn. Cần phải đảm bảo chi phí sản suất của giá đỡ phải thấp . Chính vì vậy chúng tôi đã quyết định dùng phương pháp in 3D để đảm bảo được giải pháp tùy chỉnh , nhẹ và chi phí thấp. Giá đỡ cảm biến còn được thiết kế để tay người và cảm biến không tiếp xúc trực tiếp với nhau đảm bảo an toàn cho người sử dụng.



Hình 4.8: Giá đỡ cảm biến

Sau khi tất cả cả các hệ thống phụ cho găng tay được thiết kế ,găng tay được lắp ráp và tích hợp hệ thống, găng tay bao gồm bốn yếu tố chính của hệ thống cơ điện tử.



Hình 4.9: Găng tay thu nhận thông tin chuyển động của bàn tay người

4.5.3. Lập trình phần mềm điều khiển cho Arduino

Chương trình sử dụng điều khiển thu thập thông số bàn tay được lập trên công cụ Arduino IDE với tệp tin chương trình chính được thể hiện trong Phụ lục 3. Chương trình sẽ thu nhận thông tin về góc của các cảm biến MPU5060 để thực hiện tính các góc tương đối giữa các đốt ngón tay. Sau đó sẽ truyền dữ liệu góc về máy tính thông qua kết nối Serial. Mã nguồn chương trình được dịch và nạp vào vi điều khiển trên Arduino qua cổng USB.

Để đảm bảo dữ liệu góc thu nhận được ổn định, chương trình sử dụng bộ lọc Kalman. Khi nhận dữ liệu đầu vào liên tục từ cảm biến sẽ gây ra sai số lớn và nhiễu như cảm biến góc nghiêng, độ ẩm, nhịp tim,…. Ta có thể khắc phục điều đó bằng bộ lọc Kalman, sử dụng các kiến thức về sác xuất thống kê làm tiền đề, bộ lọc sẽ giúp dữ liệu đầu ra của chúng ta có sai số nhỏ và tránh việc bị nhiễu.

Chương trình được nạp vào Arduino gắn với găng tay như trên Hình 4.9 đã truyền gửi dữ liệu ổn định về máy tính. Kết quả chương trình mô phỏng (Hình 3.15) đã nhận dữ liệu và hiển thị lại đúng trạng thái bàn tay của người đeo găng tay.

Kết luận

Kết quả đạt được:

Đề tài này đã ứng dụng thành công phần mềm Solidworks để dựng mô hình 3D bàn tay robot 5 ngón để phục vụ mô phỏng.

Tìm hiểu và ứng dụng được lập trình C++ trên môi trường Visual Studio và sử dụng thư viện đồ họa mở OpenGL để thực hiện mô phỏng 3D hoạt động của bàn tay robot mô phỏng bàn tay người.

Tìm hiểu và ứng dụng Arduino, cảm biến gia tốc MPU5060 để thiết kế chế tạo găng tay thu nhận dữ liệu chuyển động của bàn tay người thực và truyền lên máy tính để thực hiện mô phỏng lại hoạt động đó.

Đề tài đã xây dựng được phần mềm mô phỏng 3D hoạt động của bàn tay robot với các chức năng thao tác điều khiển chuyển động linh hoạt và kết nối thành công với găng tay thu nhận thông tin chuyển động của bàn tay người thực để thực hiện mô phỏng lại chuyển động đó.

Hạn chế:

Thiết kế bàn tay chưa có khả năng xòe các ngón.

Thiết kế găng tay còn chưa được chau chuốt về hình thức.

Định hướng phát triển:

Tiếp tục hoàn thiện các hạn chế đã nêu.

Chế tạo bàn tay robot được điều khiển từ phần mềm mô phỏng cũng như điều khiển từ găng tay thu nhận chuyển động của bàn tay thực.

Phát triển phần mềm để có thể thực hiện được các chức năng đa dạng hơn như: chuyển trạng thái bàn tay thành ngôn ngữ phục vụ giao tiếp của người khiếm thị và khiếm thính, kết nối điều khiển bàn tay robot cơ khí, tay máy robot, cơ cấu máy, …

Mở rộng khả năng thực hiện với chuyển động của toàn bộ cánh tay hoặc cả cơ thể người.

Tài liệu tham khảo

[1] Đào Văn Hiệp, "Kỹ Thuật Robot", NXB khoa học và kỹ thuật, 2004.

[2] A. Hussein, C. Tarry, C. Lambert, S. Barreca, and O. B. Allen. "Results of Clinicians Using a Therapeutic Robotic System in an Inpatient Stroke." Journal of NeuroEngineering (BioMed Central) 8, no. 50 (2011).

[3] Y. Fu, Q. Zhang, F. Zhang and Z. Gan, "Design and development of a hand rehabilitation robot for patient-cooperative therapy following stroke," 2011 IEEE International Conference on Mechatronics and Automation, Beijing, China, 2011, pp. 112-117, doi: 10.1109/ICMA.2011.5985641.

[4] P. Polygerinos, Z. Wang, K. C. Galloway, R. J. Wood, C. J. Walsh, "Soft robotic glove for combined assistance and at-home rehabilitation", Robotics and Autonomous Systems, Vol. 73, (2015) 135-143.

[5] A. Nair, P.L. Nair, G. Mohan, S. Nair J and J. Baby, "E-gloves: Wearable assistance in sports training", International Journal of Physiology, Nutrition and Physical Education, SP2: 01-06 (2019)

[6] A. L. Angulo, "Design and Development of a Glove with Remote controller function for video games", Final Degree Work Bachelor's Degree in Video Game Design and Development Universitat Jaume I (2020).

[7] A..A. Deshpande, G.V. Keerthi, N. Ramya, N.S. Suchitra, "Gesture Controlled Robotic Arm for Testing and Diagnosis of Novel CORONA-VIRUS", International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE), Vol. 8 (5), (2021) pp. 44-50.

[8] Z.R. Saeed, Z. B. Zainol, B. B. Zaidan, A. H. Alamoodi, "A Systematic Review on Systems-Based Sensory Gloves for Sign Language Pattern Recognition: An Update From 2017 to 2022", IEEE Access, vol. 10, (2022) pp. 123358-123377.

[9] K. Haratiannejadi, "Smart glove and hand gesture-based control interface for multi-rotor aerial vehicles", a thesis in the department of electrical and computer engineering, Concordia university (2020).

[10] M. Kazi, M. Bill, "Robotic Hand Controlled by Glove Using Wireless Communication", Degree project in Mechanical Engineering, School of

Industrial Engineering and Management , KTH Royal Institute of Technology (2020).

[11] H. Wang, Z. Feng, J. Tian, and X. Fan, “MFA: A Smart Glove with Multimodal Intent Sensing Capability”,Computational Intelligence and Neuroscience, Vol. 2022, (2022) Article ID 3545850.

[12] Chong Khye Wen, Lau Chee Yong, C. Nataraj, “Gesture Controlled Robotic Arm For Hazardous Chemical Control”, Journal of Applied Technology and Innovation (e -ISSN: 2600-7304) vol. 6, no. 3, (2022) 47-54.

[13] N. Katusin, “Glove for Augmented and Virtual Reality”, International Journal of Innovative Technology and Interdisciplinary Sciences, Vol. 2(1) (2019) pp. 147-159.

[14] V.T. Minh, N. Katushin, and J. Pumwa, “Motion tracking glove for augmented reality and virtual reality”, Paladyn, J. Behav. Robot. (2019) 10:160–166.

[15] M. Kumar, S. Shelji, P. Varghese, P. S Shilpa, D. Thomas, A. Paul, “Gyro Glove: Stabilizing the Lives of Those with the Hand Tremor”, International Journal of Engineering Science and Computing, Vo.. 10, No.7 (2020) pp. 26839-26843.

[16] T. Bright, S. Adali, G. Bright, “Low-Cost Sensory Glove for Human–Robot Collaboration in Advanced Manufacturing Systems”, Robotics, 11, 56 (2022).

[17] Phạm Quang Huy, “Giáo trình thiết kế cơ khí với SolidWorks”, NXB thiếu niên,2019.

[18] Nguyễn Văn Huy, Vũ Bá Anh. “Các xương và khớp của chi trên’’. Giải phẫu người. Nhà xuất bản Y học (2006). Trang 413-425.

[19] Nguyễn Văn Huy. “Cơ các phần cẳng tay và bàn tay’’. Giải phẫu người. Nhà xuất bản Y học (2006). Trang 73-80.

[20] Đinh GiaTường, Tạ Khánh Lâm. Nguyên lý máy, Tập 1 và 2. Nhà xuất bản Giáo dục, 2006.

[21] Lập trình Windows bằng Visual C++ - Đặng Văn Đức, Lê Quốc Hưng - Nhà xuất bản Giáo dục, 1999.

[22] Clayton Walnum, 3D graphics programing with OpenGL, Que Pub, 1995

[23] Vũ Lê Huy. “Ứng dụng thư viện đồ hoạ OpenGL, xây dựng một bộ công cụ trong lập trình nghiên cứu và mô phỏng 3 chiều các bộ truyền cơ khí”. Tuyển tập các bài báo khoa học Hội nghị khoa học lần thứ 20 - Đại học Bách Khoa Hà Nội, phân ban Cơ khí (2006) Trang 28-33.

[24] Vũ Lê Huy. “Thuật giải và định dạng mới cho các mô hình 3D trong lập trình mô phỏng”. Tuyển tập các công trình khoa học Hội nghị cơ học toàn quốc lần thứ tám. - Tập IV : Cơ học máy (2007) Trang: 280-289..

[25] Lê Văn Uyển, Trịnh Đồng Tính, Đỗ Đức Nam, Vũ Lê Huy. “Một phương pháp xây dựng và xử lý cơ sở dữ liệu trong bộ phần mềm "Tính toán thiết kế hệ dẫn động cơ khí"”. Tuyển tập công trình Hội nghị khoa học toàn quốc Cơ học vật rắn biến dạng lần thứ 7, Tập 2 (2004) Trang 967-977.

[26] Trần Vĩnh Thường. Giáo trình “Chuyên đề arduino và truyền thông’’. Trường Cao đẳng Kinh tế Kỹ thuật (2020)

Phụ lục 1

Nội dung tệp tin cấu hình bàn tay robot:

```
[workspace]
200     ;Length of Axes
0       ;type of model file - 0: stl in ASCII ; 1: stl in binary ; 2: mdf
;num Elements
;Bantay0        Ngoncai1        Ngoncai2        Ngoncai3        Ngontro1        Ngontro2
        Ngontro3
;M... ma tran chuyen
;qNC1...ngon cai
[elements]
BT0   NC1   NC2   NC3   NT1   NT2   NT3  NAPUT1   NAPUT2   NAPUT3   NG1
NG2  NG3  NUT1  NUT2  NUT3
MBT0 MNC1 MNC2 MNC3 MNT1 MNT2 MNT3 MNAPUT1   MNAPUT2   MNAPUT3
MNG1  MNG2 MNG3 MNUT1 MNUT2 MNUT3
;
;name    min     max         init  unit
[parameters]
qNC1    0       1.5708 0        "rad"
qNC2    0       1.5708 0        "rad"
qNC3    0       1.5708 0        "rad"
qNT1    0       1.5708 0        "rad"
qNT2    0       1.5708 0        "rad"
qNT3    0       1.5708 0        "rad"
qNAPUT1 0   1.5708 0        "rad"
qNAPUT2 0   1.5708 0        "rad"
qNAPUT3 0   1.5708 0        "rad"
qNG1    0       1.5708 0        "rad"
qNG2    0       1.5708 0        "rad"
qNG3    0       1.5708 0        "rad"
qNUT1           0       1.5708 0        "rad"
qNUT2           0       1.5708 0        "rad"
qNUT3           0       1.5708 0        "rad"
;
;
[constants]
phiNC1          1.57079632679         "rad"
phiNC2          -0.78539816339        "rad"
aNC0   35       "mm"
aNC1   35       "mm"
aNC2   35       "mm"
dNC    35       "mm"
aNT0   100      "mm"
aNT1   40       "mm"
aNT2   25       "mm"
dNT    0        "mm"
aNAPUT0 100 "mm"
aNAPUT1 45 "mm"
```

```
aNAPUT2 20 "mm"
dNAPUT  40  "mm"
aNG0  100     "mm"
aNG1  50      "mm"
aNG2  25      "mm"
dNG   20      "mm"
aNUT0100      "mm"
aNUT135       "mm"
aNUT220       "mm"
dNUT  60      "mm"
;
;Transform matrixes
[MBT0]
1       0       0       0
0       1       0       0
0       0       1       0
0       0       0       1
[MNC1]
cos(-phiNC1)*cos(qNC1)     -cos(-phiNC1)*sin(qNC1)     sin(phiNC1)   aNC0
    sin(qNC1)           cos(qNC1)         0         0
-sin(-phiNC1)*cos(qNC1)    sin(-phiNC1)*sin(qNC1)      cos(-phiNC1)   0
    0                       0         0         1
[MNC2]
-cos(qNC2)*(sin(phiNC1)*sin(phiNC2)-cos(phiNC1)*cos(phiNC2)*cos(qNC1))-
cos(phiNC1)*sin(qNC1)*sin(qNC2)                    sin(qNC2)*(sin(phiNC1)*sin(phiNC2)-
cos(phiNC1)*cos(phiNC2)*cos(qNC1))-cos(phiNC1)*cos(qNC2)*sin(qNC1)
        cos(phiNC2)*sin(phiNC1)+cos(phiNC1)*cos(qNC1)*sin(phiNC2)
        aNC0+aNC1*cos(phiNC1)*cos(qNC1)
cos(qNC1)*sin(qNC2)+cos(phiNC2)*cos(qNC2)*sin(qNC1)
                cos(qNC1)*cos(qNC2)-cos(phiNC2)*sin(qNC1)*sin(qNC2)
                                                    sin(phiNC2)*sin(qNC1)
                                        aNC1*sin(qNC1)
-cos(qNC2)*(cos(phiNC1)*sin(phiNC2)-cos(phiNC2)*cos(qNC1)*sin(phiNC1))-
sin(phiNC1)*sin(qNC1)*sin(qNC2)                    sin(qNC2)*(cos(phiNC1)*sin(phiNC2)-
cos(phiNC2)*cos(qNC1)*sin(phiNC1))-cos(qNC2)*sin(phiNC1)*sin(qNC1)
        cos(phiNC1)*cos(phiNC2)+cos(qNC1)*sin(phiNC1)*sin(phiNC2)
        aNC1*cos(qNC1)*sin(phiNC1)
0                                                                                   0
0                                       1
[MNC3]
sin(qNC3)*(sin(qNC2)*(sin(phiNC1)*sin(phiNC2)-cos(phiNC1)*cos(phiNC2)*cos(qNC1))-
cos(phiNC1)*cos(qNC2)*sin(qNC1))-cos(qNC3)*(cos(qNC2)*(sin(phiNC1)*sin(phiNC2)-
cos(phiNC1)*cos(phiNC2)*cos(qNC1))+cos(phiNC1)*sin(qNC1)*sin(qNC2))
                cos(qNC3)*(sin(qNC2)*(sin(phiNC1)*sin(phiNC2)-
cos(phiNC1)*cos(phiNC2)*cos(qNC1))-
cos(phiNC1)*cos(qNC2)*sin(qNC1))+sin(qNC3)*(cos(qNC2)*(sin(phiNC1)*sin(phiNC2)-
cos(phiNC1)*cos(phiNC2)*cos(qNC1))+cos(phiNC1)*sin(qNC1)*sin(qNC2))
        cos(phiNC2)*sin(phiNC1)+cos(phiNC1)*cos(qNC1)*sin(phiNC2)
        aNC0-aNC2*(cos(qNC2)*(sin(phiNC1)*sin(phiNC2)-
```

```
cos(phiNC1)*cos(phiNC2)*cos(qNC1))+cos(phiNC1)*sin(qNC1)*sin(qNC2))+aNC1*cos(p
hiNC1)*cos(qNC1)

cos(qNC3)*(cos(qNC1)*sin(qNC2)+cos(phiNC2)*cos(qNC2)*sin(qNC1))+sin(qNC3)*(cos(
qNC1)*cos(qNC2)-cos(phiNC2)*sin(qNC1)*sin(qNC2))
                                cos(qNC3)*(cos(qNC1)*cos(qNC2)-
cos(phiNC2)*sin(qNC1)*sin(qNC2))-
sin(qNC3)*(cos(qNC1)*sin(qNC2)+cos(phiNC2)*cos(qNC2)*sin(qNC1))

                    sin(phiNC2)*sin(qNC1)
aNC2*(cos(qNC1)*sin(qNC2)+cos(phiNC2)*cos(qNC2)*sin(qNC1))+aNC1*sin(qNC1)
sin(qNC3)*(sin(qNC2)*(cos(phiNC1)*sin(phiNC2)-cos(phiNC2)*cos(qNC1)*sin(phiNC1))-
cos(qNC2)*sin(phiNC1)*sin(qNC1))-cos(qNC3)*(cos(qNC2)*(cos(phiNC1)*sin(phiNC2)-
cos(phiNC2)*cos(qNC1)*sin(phiNC1))+sin(phiNC1)*sin(qNC1)*sin(qNC2))
                    cos(qNC3)*(sin(qNC2)*(cos(phiNC1)*sin(phiNC2)-
cos(phiNC2)*cos(qNC1)*sin(phiNC1))-
cos(qNC2)*sin(phiNC1)*sin(qNC1))+sin(qNC3)*(cos(qNC2)*(cos(phiNC1)*sin(phiNC2)-
cos(phiNC2)*cos(qNC1)*sin(phiNC1))+sin(phiNC1)*sin(qNC1)*sin(qNC2))
0
0
0                                                   1
[MNT1]
cos(qNT1)     -sin(qNT1)     0        aNT0
sin(qNT1)     cos(qNT1)      0        0
 0            0        1      0
 0            0        0      1
[MNT2]
cos(qNT1+qNT2)     -sin(qNT1+qNT2)     0        aNT1*cos(qNT1)+aNT0
sin(qNT1+qNT2)     cos(qNT1+qNT2)      0        aNT1*sin(qNT1)
0                  0     1    0
0                  0     0    1
[MNT3]
cos(qNT1+qNT2+qNT3)      -sin(qNT1+qNT2+qNT3)      0
        aNT2*cos(qNT1+qNT2)+aNT1*cos(qNT1)+aNT0
sin(qNT1+qNT2+qNT3)      cos(qNT1+qNT2+qNT3)       0
        aNT2*sin(qNT1+qNT2)+aNT1*sin(qNT1)
0       0       1       0
0       0       0       1
[MNAPUT1]
cos(qNAPUT1)        -sin(qNAPUT1)        0       aNAPUT0
sin(qNAPUT1)        cos(qNAPUT1)         0       0
0       0       1       -dNAPUT
0       0       0       1
[MNAPUT2]
cos(qNAPUT1+qNAPUT2)   -sin(qNAPUT1+qNAPUT2)  0
        aNAPUT1*cos(qNAPUT1)+aNAPUT0
sin(qNAPUT1+qNAPUT2)   cos(qNAPUT1+qNAPUT2)   0
        aNAPUT1*sin(qNAPUT1)
0       0       1       -dNAPUT
0       0       0       1
```

```
[MNAPUT3]
cos(qNAPUT1+qNAPUT2+qNAPUT3)      -sin(qNAPUT1+qNAPUT2+qNAPUT3)      0
        aNAPUT2*cos(qNAPUT1+qNAPUT2)+aNAPUT1*cos(qNAPUT1)+aNAPUT0
sin(qNAPUT1+qNAPUT2+qNAPUT3)      cos(qNAPUT1+qNAPUT2+qNAPUT3)      0
        aNAPUT2*sin(qNAPUT1+qNAPUT2)+aNAPUT1*sin(qNAPUT1)
0       0       1       -dNAPUT
0       0       0       1
[MNG1]
cos(qNG1)    -sin(qNG1)    0       aNG0
sin(qNG1)    cos(qNG1)     0       0
0       0       1       -dNG
0       0       0       1
[MNG2]
cos(qNG1+qNG2)    -sin(qNG1+qNG2)    0       aNG1*cos(qNG1)+aNG0
sin(qNG1+qNG2)    cos(qNG1+qNG2)     0       aNG1*sin(qNG1)
0       0       1       -dNG
0       0       0       1
[MNG3]
cos(qNG1+qNG2+qNG3)     -sin(qNG1+qNG2+qNG3)     0
        aNG2*cos(qNG1+qNG2)+aNG1*cos(qNG1)+aNG0
sin(qNG1+qNG2+qNG3)     cos(qNG1+qNG2+qNG3)     0
        aNG2*sin(qNG1+qNG2)+aNG1*sin(qNG1)
0       0       1       -dNG
0       0       0       1
[MNUT1]
cos(qNUT1)   -sin(qNUT1)   0       aNUT0
sin(qNUT1)   cos(qNUT1)    0       0
0       0       1       -dNUT
0       0       0       1
[MNUT2]
cos(qNUT1+qNUT2) -sin(qNUT1+qNUT2) 0       aNUT1*cos(qNUT1)+aNUT0
sin(qNUT1+qNUT2)  cos(qNUT1+qNUT2) 0       aNUT1*sin(qNUT1)
0       0       1       -dNUT
0       0       0       1
[MNUT3]
cos(qNUT1+qNUT2+qNUT3)         -sin(qNUT1+qNUT2+qNUT3)         0
        aNUT2*cos(qNUT1+qNUT2)+aNUT1*cos(qNUT1)+aNUT0
sin(qNUT1+qNUT2+qNUT3)         cos(qNUT1+qNUT2+qNUT3)         0
        aNUT2*sin(qNUT1+qNUT2)+aNUT1*sin(qNUT1)
0       0       1       -dNUT
0       0       0       1
;
;*X0    Y0      Z0  -  translation of an element - not used at moment
;filename                                                        R       G       B       X0
        Y0      Z0      phi     theta   -  X0, Y0, Z0, phi, theta are not used at moment
[BT0]
*0      0       0
"Ban tay robot_STL\Long ban tay\long ban tay.STL"       0       1       1       0       0
        0       0       0
[NC1]
```

```
*0      0       0
"Ban tay robot_STL\Ngon cai\dot 1 ngon cai.STL"          1       1       1       0       0
        0       0       0
[NC2]
*0      0       0
"Ban tay robot_STL\Ngon cai\dot 2 ngon cai.STL"          1       1       1       0       0
        0       0       0
[NC3]
*0      0       0
"Ban tay robot_STL\Ngon cai\dot 3 ngon cai.STL"          1       1       1       0       0
        0       0       0
[NT1]
*0      0       0
"Ban tay robot_STL\Ngon tro\dot 1 ngon tro.STL"          1       1       0       0       0
        0       0       0
[NT2]
*0      0       0
"Ban tay robot_STL\Ngon tro\dot 2 ngon tro.STL"          1       1       0       0       0
        0       0       0
[NT3]
*0      0       0
"Ban tay robot_STL\Ngon tro\dot 3 ngon tro.STL"          1       1       0       0       0
        0       0       0
[NAPUT1]
*0      0       0
"Ban tay robot_STL\Ngon ap ut\dot 1 ngon ap ut.STL"  1   1       0       0       0       0
        0       0
[NAPUT2]
*0      0       0
"Ban tay robot_STL\Ngon ap ut\dot 2 ngon ap ut.STL"  1   1       0       0       0       0
        0       0
 [NAPUT3]
*0      0       0
"Ban tay robot_STL\Ngon ap ut\dot 3 ngon ap ut.STL"  1   1       0       0       0       0
        0       0
 [NG1]
*0      0       0
"Ban tay robot_STL\Ngon giua\dot 1 ngon giua.STL"    1   1       0       0       0       0
        0       0
 [NG2]
*0      0       0
"Ban tay robot_STL\Ngon giua\dot 2 ngon giua.STL"    1   1       0       0       0       0
        0       0
 [NG3]
*0      0       0
"Ban tay robot_STL\Ngon giua\dot 3 ngon giua.STL"    1   1       0       0       0       0
        0       0
[NUT1]
*0      0       0
```

"Ban tay robot_STL\Ngon ut\dot 1 ngon ut.STL" 1 1 0 0 0 0
0 0
[NUT2]
*0 0 0
"Ban tay robot_STL\Ngon ut\dot 2 ngon ut.STL" 1 1 0 0 0 0
0 0
[NUT3]
*0 0 0
"Ban tay robot_STL\Ngon ut\dot 3 ngon ut.STL" 1 1 0 0 0 0
0 0
;

Phụ lục 2

Nội dung tệp dữ liệu quỹ đạo chuyển động của các ngón trên bàn tay robot:

xNT	yNT	zNT	qNC1	qNC2	qNC3	qNT1	qNT2	qNT3	qNG1	qNG2	qNG3
qNAPUT1			qNAPUT2			qNAPUT3			qNUT1		
qNUT2			qNUT3								
185	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0							
184.9605266	2.35569619	0	0.015708	0.015708	0.015708	0.015708					
0.015708	0.015708	0.015708	0.015708	0.015708	0.015708						
0.015708	0.015708	0.015708	0.015708	0.015708							
184.8421692	4.708370609	0	0.031416	0.031416	0.031416	0.031416					
0.031416	0.031416	0.031416	0.031416	0.031416	0.031416						
0.031416	0.031416	0.031416	0.031416	0.031416							
184.6451156	7.055006934	0	0.047124	0.047124	0.047124	0.047124					
0.047124	0.047124	0.047124	0.047124	0.047124	0.047124						
0.047124	0.047124	0.047124	0.047124	0.047124							
184.3696787	9.392599726	0	0.062832	0.062832	0.062832	0.062832					
0.062832	0.062832	0.062832	0.062832	0.062832	0.062832						
0.062832	0.062832	0.062832	0.062832	0.062832							
184.0162957	11.71815984	0	0.078540	0.078540	0.078540	0.078540					
0.078540	0.078540	0.078540	0.078540	0.078540	0.078540						
0.078540	0.078540	0.078540	0.078540	0.078540							
183.585527	14.02871982	0	0.094248	0.094248	0.094248	0.094248					
0.094248	0.094248	0.094248	0.094248	0.094248	0.094248						
0.094248	0.094248	0.094248	0.094248	0.094248							
183.0780555	16.32133921	0	0.109956	0.109956	0.109956	0.109956					
0.109956	0.109956	0.109956	0.109956	0.109956	0.109956						
0.109956	0.109956	0.109956	0.109956	0.109956							
182.4946852	18.59310986	0	0.125664	0.125664	0.125664	0.125664					
0.125664	0.125664	0.125664	0.125664	0.125664	0.125664						
0.125664	0.125664	0.125664	0.125664	0.125664							

181.8363393 20.84116114 0 0.141372 0.141372 0.141372 0.141372
0.141372 0.141372 0.141372 0.141372 0.141372 0.141372
0.141372 0.141372 0.141372 0.141372 0.141372

181.104059 23.06266507 0 0.157080 0.157080 0.157080 0.157080
0.157080 0.157080 0.157080 0.157080 0.157080 0.157080
0.157080 0.157080 0.157080 0.157080 0.157080

180.2990009 25.25484137 0 0.172788 0.172788 0.172788 0.172788
0.172788 0.172788 0.172788 0.172788 0.172788 0.172788
0.172788 0.172788 0.172788 0.172788 0.172788

179.4224351 27.41496244 0 0.188496 0.188496 0.188496 0.188496
0.188496 0.188496 0.188496 0.188496 0.188496 0.188496
0.188496 0.188496 0.188496 0.188496 0.188496

178.4757426 29.54035818 0 0.204204 0.204204 0.204204 0.204204
0.204204 0.204204 0.204204 0.204204 0.204204 0.204204
0.204204 0.204204 0.204204 0.204204 0.204204

177.4604127 31.62842074 0 0.219912 0.219912 0.219912 0.219912
0.219912 0.219912 0.219912 0.219912 0.219912 0.219912
0.219912 0.219912 0.219912 0.219912 0.219912

176.3780401 33.67660913 0 0.235620 0.235620 0.235620 0.235620
0.235620 0.235620 0.235620 0.235620 0.235620 0.235620
0.235620 0.235620 0.235620 0.235620 0.235620

175.2303219 35.68245368 0 0.251328 0.251328 0.251328 0.251328
0.251328 0.251328 0.251328 0.251328 0.251328 0.251328
0.251328 0.251328 0.251328 0.251328 0.251328

174.019054 37.64356039 0 0.267036 0.267036 0.267036 0.267036
0.267036 0.267036 0.267036 0.267036 0.267036 0.267036
0.267036 0.267036 0.267036 0.267036 0.267036

172.746128 39.55761505 0 0.282744 0.282744 0.282744 0.282744
0.282744 0.282744 0.282744 0.282744 0.282744 0.282744
0.282744 0.282744 0.282744 0.282744 0.282744

171.4135276 41.42238733 0 0.298452 0.298452 0.298452 0.298452
0.298452 0.298452 0.298452 0.298452 0.298452 0.298452
0.298452 0.298452 0.298452 0.298452 0.298452

170.0233242 43.23573454 0 0.314160 0.314160 0.314160 0.314160
0.314160 0.314160 0.314160 0.314160 0.314160 0.314160
0.314160 0.314160 0.314160 0.314160 0.314160

168.5776737 44.99560537 0 0.329868 0.329868 0.329868 0.329868
0.329868 0.329868 0.329868 0.329868 0.329868 0.329868
0.329868 0.329868 0.329868 0.329868 0.329868

167.0788117 46.70004332 0 0.345576 0.345576 0.345576 0.345576
0.345576 0.345576 0.345576 0.345576 0.345576 0.345576
0.345576 0.345576 0.345576 0.345576 0.345576

165.5290496 48.34719 0 0.361284 0.361284 0.361284 0.361284
0.361284 0.361284 0.361284 0.361284 0.361284 0.361284
0.361284 0.361284 0.361284 0.361284 0.361284

163.9307699 49.93528824 0 0.376992 0.376992 0.376992 0.376992
0.376992 0.376992 0.376992 0.376992 0.376992 0.376992
0.376992 0.376992 0.376992 0.376992 0.376992

162.286422 51.46268496 0 0.392700 0.392700 0.392700 0.392700
0.392700 0.392700 0.392700 0.392700 0.392700 0.392700
0.392700 0.392700 0.392700 0.392700 0.392700

160.5985172 52.92783385 0 0.408408 0.408408 0.408408 0.408408
0.408408 0.408408 0.408408 0.408408 0.408408 0.408408
0.408408 0.408408 0.408408 0.408408 0.408408

158.8696238 54.32929783 0 0.424116 0.424116 0.424116 0.424116
0.424116 0.424116 0.424116 0.424116 0.424116 0.424116
0.424116 0.424116 0.424116 0.424116 0.424116

157.1023627 55.6657513 0 0.439824 0.439824 0.439824 0.439824
0.439824 0.439824 0.439824 0.439824 0.439824 0.439824
0.439824 0.439824 0.439824 0.439824 0.439824

155.2994019 56.93598219 0 0.455532 0.455532 0.455532 0.455532
0.455532 0.455532 0.455532 0.455532 0.455532 0.455532
0.455532 0.455532 0.455532 0.455532 0.455532

153.4634517 58.13889366 0 0.471240 0.471240 0.471240 0.471240
0.471240 0.471240 0.471240 0.471240 0.471240 0.471240
0.471240 0.471240 0.471240 0.471240 0.471240

151.5972596 59.27350579 0 0.486948 0.486948 0.486948 0.486948
0.486948 0.486948 0.486948 0.486948 0.486948 0.486948
0.486948 0.486948 0.486948 0.486948 0.486948

149.7036048 60.33895679 0 0.502656 0.502656 0.502656 0.502656
0.502656 0.502656 0.502656 0.502656 0.502656 0.502656
0.502656 0.502656 0.502656 0.502656 0.502656

147.7852934 61.33450417 0 0.518364 0.518364 0.518364 0.518364
0.518364 0.518364 0.518364 0.518364 0.518364 0.518364
0.518364 0.518364 0.518364 0.518364 0.518364

145.8451527 62.25952557 0 0.534072 0.534072 0.534072 0.534072
0.534072 0.534072 0.534072 0.534072 0.534072 0.534072
0.534072 0.534072 0.534072 0.534072 0.534072

143.8860261 63.11351935 0 0.549780 0.549780 0.549780 0.549780
0.549780 0.549780 0.549780 0.549780 0.549780 0.549780
0.549780 0.549780 0.549780 0.549780 0.549780

141.9107678 63.896105 0 0.565488 0.565488 0.565488 0.565488
0.565488 0.565488 0.565488 0.565488 0.565488 0.565488
0.565488 0.565488 0.565488 0.565488 0.565488

139.9222372 64.60702329 0 0.581196 0.581196 0.581196 0.581196
0.581196 0.581196 0.581196 0.581196 0.581196 0.581196
0.581196 0.581196 0.581196 0.581196 0.581196

137.923294 65.2461361 0 0.596904 0.596904 0.596904 0.596904
0.596904 0.596904 0.596904 0.596904 0.596904 0.596904
0.596904 0.596904 0.596904 0.596904 0.596904

135.9167925 65.81342614 0 0.612612 0.612612 0.612612 0.612612
0.612612 0.612612 0.612612 0.612612 0.612612 0.612612
0.612612 0.612612 0.612612 0.612612 0.612612

133.9055765 66.30899633 0 0.628320 0.628320 0.628320 0.628320
0.628320 0.628320 0.628320 0.628320 0.628320 0.628320
0.628320 0.628320 0.628320 0.628320 0.628320

131.8924741 66.73306901 0 0.644028 0.644028 0.644028 0.644028
0.644028 0.644028 0.644028 0.644028 0.644028 0.644028
0.644028 0.644028 0.644028 0.644028 0.644028

129.8802924 67.08598487 0 0.659736 0.659736 0.659736 0.659736
0.659736 0.659736 0.659736 0.659736 0.659736 0.659736
0.659736 0.659736 0.659736 0.659736 0.659736

127.8718123 67.36820166 0 0.675444 0.675444 0.675444 0.675444
0.675444 0.675444 0.675444 0.675444 0.675444 0.675444
0.675444 0.675444 0.675444 0.675444 0.675444

125.8697835 67.5802927 0 0.691152 0.691152 0.691152 0.691152
0.691152 0.691152 0.691152 0.691152 0.691152 0.691152
0.691152 0.691152 0.691152 0.691152 0.691152

123.8769198 67.72294512 0 0.706860 0.706860 0.706860 0.706860
0.706860 0.706860 0.706860 0.706860 0.706860 0.706860
0.706860 0.706860 0.706860 0.706860 0.706860

121.8958937 67.79695793 0 0.722568 0.722568 0.722568 0.722568
0.722568 0.722568 0.722568 0.722568 0.722568 0.722568
0.722568 0.722568 0.722568 0.722568 0.722568

119.9293319 67.80323983 0 0.738276 0.738276 0.738276 0.738276
0.738276 0.738276 0.738276 0.738276 0.738276 0.738276
0.738276 0.738276 0.738276 0.738276 0.738276

117.9798105 67.74280682 0 0.753984 0.753984 0.753984 0.753984
0.753984 0.753984 0.753984 0.753984 0.753984 0.753984
0.753984 0.753984 0.753984 0.753984 0.753984

116.0498507 67.61677962 0 0.769692 0.769692 0.769692 0.769692
0.769692 0.769692 0.769692 0.769692 0.769692 0.769692
0.769692 0.769692 0.769692 0.769692 0.769692

114.1419139 67.4263809 0 0.785400 0.785400 0.785400 0.785400
0.785400 0.785400 0.785400 0.785400 0.785400 0.785400
0.785400 0.785400 0.785400 0.785400 0.785400

112.2583979 67.17293226 0 0.801108 0.801108 0.801108 0.801108
0.801108 0.801108 0.801108 0.801108 0.801108 0.801108
0.801108 0.801108 0.801108 0.801108 0.801108

110.4016323 66.8578511 0 0.816816 0.816816 0.816816 0.816816
0.816816 0.816816 0.816816 0.816816 0.816816 0.816816
0.816816 0.816816 0.816816 0.816816 0.816816

108.5738749 66.48264726 0 0.832524 0.832524 0.832524 0.832524
0.832524 0.832524 0.832524 0.832524 0.832524 0.832524
0.832524 0.832524 0.832524 0.832524 0.832524

106.7773075 66.04891948 0 0.848232 0.848232 0.848232 0.848232
0.848232 0.848232 0.848232 0.848232 0.848232 0.848232
0.848232 0.848232 0.848232 0.848232 0.848232

105.0140325 65.55835176 0 0.863940 0.863940 0.863940 0.863940
0.863940 0.863940 0.863940 0.863940 0.863940 0.863940
0.863940 0.863940 0.863940 0.863940 0.863940

103.2860692 65.01270948 0 0.879648 0.879648 0.879648 0.879648
0.879648 0.879648 0.879648 0.879648 0.879648 0.879648
0.879648 0.879648 0.879648 0.879648 0.879648

101.5953509 64.41383545 0 0.895356 0.895356 0.895356 0.895356
0.895356 0.895356 0.895356 0.895356 0.895356 0.895356
0.895356 0.895356 0.895356 0.895356 0.895356

99.94372118 63.76364576 0 0.911064 0.911064 0.911064 0.911064
0.911064 0.911064 0.911064 0.911064 0.911064 0.911064
0.911064 0.911064 0.911064 0.911064 0.911064

98.3329314 63.06412554 0 0.926772 0.926772 0.926772 0.926772
0.926772 0.926772 0.926772 0.926772 0.926772 0.926772
0.926772 0.926772 0.926772 0.926772 0.926772

96.76463792 62.31732457 0 0.942480 0.942480 0.942480 0.942480
0.942480 0.942480 0.942480 0.942480 0.942480 0.942480
0.942480 0.942480 0.942480 0.942480 0.942480

95.24039949 61.52535283 0 0.958188 0.958188 0.958188 0.958188
0.958188 0.958188 0.958188 0.958188 0.958188 0.958188
0.958188 0.958188 0.958188 0.958188 0.958188

93.76167503 60.69037588 0 0.973896 0.973896 0.973896 0.973896
0.973896 0.973896 0.973896 0.973896 0.973896 0.973896
0.973896 0.973896 0.973896 0.973896 0.973896

92.3298215 59.81461019 0 0.989604 0.989604 0.989604 0.989604
0.989604 0.989604 0.989604 0.989604 0.989604 0.989604
0.989604 0.989604 0.989604 0.989604 0.989604

90.94609205 58.9003184 0 1.005312 1.005312 1.005312 1.005312
1.005312 1.005312 1.005312 1.005312 1.005312 1.005312
1.005312 1.005312 1.005312 1.005312 1.005312

89.61163438 57.94980448 0 1.021020 1.021020 1.021020 1.021020
1.021020 1.021020 1.021020 1.021020 1.021020 1.021020
1.021020 1.021020 1.021020 1.021020 1.021020

88.32748931 56.96540884 0 1.036728 1.036728 1.036728 1.036728
1.036728 1.036728 1.036728 1.036728 1.036728 1.036728
1.036728 1.036728 1.036728 1.036728 1.036728

87.09458959 55.9495034 0 1.052436 1.052436 1.052436 1.052436
1.052436 1.052436 1.052436 1.052436 1.052436 1.052436
1.052436 1.052436 1.052436 1.052436 1.052436

85.91375893 54.90448659 0 1.068144 1.068144 1.068144 1.068144
1.068144 1.068144 1.068144 1.068144 1.068144 1.068144
1.068144 1.068144 1.068144 1.068144 1.068144

84.78571129 53.83277839 0 1.083852 1.083852 1.083852 1.083852
1.083852 1.083852 1.083852 1.083852 1.083852 1.083852
1.083852 1.083852 1.083852 1.083852 1.083852

83.71105036 52.73681528 0 1.099560 1.099560 1.099560 1.099560
1.099560 1.099560 1.099560 1.099560 1.099560 1.099560
1.099560 1.099560 1.099560 1.099560 1.099560

82.69026926 51.61904518 0 1.115268 1.115268 1.115268 1.115268
1.115268 1.115268 1.115268 1.115268 1.115268 1.115268
1.115268 1.115268 1.115268 1.115268 1.115268

81.72375055 50.48192248 0 1.130976 1.130976 1.130976 1.130976
1.130976 1.130976 1.130976 1.130976 1.130976 1.130976
1.130976 1.130976 1.130976 1.130976 1.130976

80.8117664 49.327903 0 1.146684 1.146684 1.146684 1.146684
1.146684 1.146684 1.146684 1.146684 1.146684 1.146684
1.146684 1.146684 1.146684 1.146684 1.146684

79.95447899 48.159439 0 1.162392 1.162392 1.162392 1.162392
1.162392 1.162392 1.162392 1.162392 1.162392 1.162392
1.162392 1.162392 1.162392 1.162392 1.162392

79.15194116 46.97897424 0 1.178100 1.178100 1.178100 1.178100
1.178100 1.178100 1.178100 1.178100 1.178100 1.178100
1.178100 1.178100 1.178100 1.178100 1.178100

78.40409732 45.78893905 0 1.193808 1.193808 1.193808 1.193808
1.193808 1.193808 1.193808 1.193808 1.193808 1.193808
1.193808 1.193808 1.193808 1.193808 1.193808

77.71078447 44.59174551 0 1.209516 1.209516 1.209516 1.209516
1.209516 1.209516 1.209516 1.209516 1.209516 1.209516
1.209516 1.209516 1.209516 1.209516 1.209516

77.07173357 43.38978266 0 1.225224 1.225224 1.225224 1.225224
1.225224 1.225224 1.225224 1.225224 1.225224 1.225224
1.225224 1.225224 1.225224 1.225224 1.225224

76.48657104 42.18541177 0 1.240932 1.240932 1.240932 1.240932
1.240932 1.240932 1.240932 1.240932 1.240932 1.240932
1.240932 1.240932 1.240932 1.240932 1.240932

75.95482051 40.98096173 0 1.256640 1.256640 1.256640 1.256640
1.256640 1.256640 1.256640 1.256640 1.256640 1.256640
1.256640 1.256640 1.256640 1.256640 1.256640

75.47590475 39.77872454 0 1.272348 1.272348 1.272348 1.272348
1.272348 1.272348 1.272348 1.272348 1.272348 1.272348
1.272348 1.272348 1.272348 1.272348 1.272348

75.04914783 38.58095089 0 1.288056 1.288056 1.288056 1.288056
1.288056 1.288056 1.288056 1.288056 1.288056 1.288056
1.288056 1.288056 1.288056 1.288056 1.288056

74.67377744 37.38984581 0 1.303764 1.303764 1.303764 1.303764
1.303764 1.303764 1.303764 1.303764 1.303764 1.303764
1.303764 1.303764 1.303764 1.303764 1.303764

74.34892746 36.20756452 0 1.319472 1.319472 1.319472 1.319472
1.319472 1.319472 1.319472 1.319472 1.319472 1.319472
1.319472 1.319472 1.319472 1.319472 1.319472

74.07364062 35.03620839 0 1.335180 1.335180 1.335180 1.335180
1.335180 1.335180 1.335180 1.335180 1.335180 1.335180
1.335180 1.335180 1.335180 1.335180 1.335180

73.84687147 33.877821 0 1.350888 1.350888 1.350888 1.350888
1.350888 1.350888 1.350888 1.350888 1.350888 1.350888
1.350888 1.350888 1.350888 1.350888 1.350888

73.66748938 32.73438437 0 1.366596 1.366596 1.366596 1.366596
1.366596 1.366596 1.366596 1.366596 1.366596 1.366596
1.366596 1.366596 1.366596 1.366596 1.366596

73.53428179 31.60781537 0 1.382304 1.382304 1.382304 1.382304
1.382304 1.382304 1.382304 1.382304 1.382304 1.382304
1.382304 1.382304 1.382304 1.382304 1.382304

73.44595761 30.49996227 0 1.398012 1.398012 1.398012 1.398012
1.398012 1.398012 1.398012 1.398012 1.398012 1.398012
1.398012 1.398012 1.398012 1.398012 1.398012

73.40115075 29.41260143 0 1.413720 1.413720 1.413720 1.413720
1.413720 1.413720 1.413720 1.413720 1.413720 1.413720
1.413720 1.413720 1.413720 1.413720 1.413720

73.39842375 28.34743424 0 1.429428 1.429428 1.429428 1.429428
1.429428 1.429428 1.429428 1.429428 1.429428 1.429428
1.429428 1.429428 1.429428 1.429428 1.429428

73.43627166 27.30608416 0 1.445136 1.445136 1.445136 1.445136
1.445136 1.445136 1.445136 1.445136 1.445136 1.445136
1.445136 1.445136 1.445136 1.445136 1.445136

73.51312587 26.29009399 0 1.460844 1.460844 1.460844 1.460844
1.460844 1.460844 1.460844 1.460844 1.460844 1.460844
1.460844 1.460844 1.460844 1.460844 1.460844

73.62735824 25.30092335 0 1.476552 1.476552 1.476552 1.476552
1.476552 1.476552 1.476552 1.476552 1.476552 1.476552
1.476552 1.476552 1.476552 1.476552 1.476552

73.77728518 24.33994631 0 1.492260 1.492260 1.492260 1.492260
1.492260 1.492260 1.492260 1.492260 1.492260 1.492260
1.492260 1.492260 1.492260 1.492260 1.492260

73.96117192 23.40844925 0 1.507968 1.507968 1.507968 1.507968
1.507968 1.507968 1.507968 1.507968 1.507968 1.507968
1.507968 1.507968 1.507968 1.507968 1.507968

74.1772368	22.50762891	0	1.523676	1.523676	1.523676	1.523676	
1.523676	1.523676	1.523676	1.523676	1.523676	1.523676		
1.523676	1.523676	1.523676	1.523676	1.523676			
74.42365569	21.63859068	0	1.539384	1.539384	1.539384	1.539384	
1.539384	1.539384	1.539384	1.539384	1.539384	1.539384		
1.539384	1.539384	1.539384	1.539384	1.539384			
74.69856646	20.80234705	0	1.555092	1.555092	1.555092	1.555092	
1.555092	1.555092	1.555092	1.555092	1.555092	1.555092		
1.555092	1.555092	1.555092	1.555092	1.555092			
75.00007346	19.99981634	0	1.570800	1.570800	1.570800	1.570800	
1.570800	1.570800	1.570800	1.570800	1.570800	1.570800		
1.570800	1.570800	1.570800	1.570800	1.570800			

Phụ lục 3

Mã nguồn chương trình điều khiển trên Arduino IDE:

```
#include <MPU6050_tockn.h>
#include <Wire.h>
#include<math.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <SimpleKalmanFilter.h>

/*Lưu ý: Các kí tự A,B,C,D,E đại diện cho các ngón cái, trỏ, giữa, áp út và út !
  các số từ 1 đến 3 đại diện cho các đốt từ dưới lên trên */

// Định nghĩa biến cho địa chỉ TCA9548A
const byte ADD = 0x70;
const byte ADD_S = 0x71;
const byte ADD_S_S = 0x72;

// Khởi tạo các biến cho nút bấm cũng như led
int count = 0;
int macdinh = 0;
int led = 6;
int nut_on = 9;
int nut_off = 10;

// Gọi các thuộc tính cho NRF24
/*RF24 radio(7, 8);
const byte address[6] = "30403";
int receivedData[3];*/

//Gọi các thuộc tính cho cảm biến
MPU6050 mpu6050_0(Wire);

MPU6050 mpu6050A_1(Wire);
MPU6050 mpu6050A_2(Wire);
```

```
MPU6050 mpu6050A_3(Wire);

MPU6050 mpu6050B_1(Wire);
MPU6050 mpu6050B_2(Wire);
MPU6050 mpu6050B_3(Wire);

MPU6050 mpu6050C_1(Wire);
MPU6050 mpu6050C_2(Wire);
MPU6050 mpu6050C_3(Wire);

MPU6050 mpu6050D_1(Wire);
MPU6050 mpu6050D_2(Wire);
MPU6050 mpu6050D_3(Wire);

MPU6050 mpu6050E_1(Wire);
MPU6050 mpu6050E_2(Wire);
MPU6050 mpu6050E_3(Wire);

// Bộ lọc dữ liệu Kalman
SimpleKalmanFilter simpleKalmanFilter1(1, 1, 0.01);
SimpleKalmanFilter simpleKalmanFilter2(1, 1, 0.01);
SimpleKalmanFilter simpleKalmanFilter3(1, 1, 0.01);

void TCA9548A(uint8_t bus, byte address);
void setgoc(float gocbase, float gockhau1_0, float gockhau2_0, float gockhau3_0,char name);
void setngoncai(float gocbase,float gockhau1_0_x, float gockhau1_0_y, float gockhau2_0,
float gockhau3_0,char name);

void setup(){
  Serial.begin(115200);

  // Kết nối NRF24
  /*radio.begin();
  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_MIN);
```

```
radio.setChannel(14);
radio.setDataRate(RF24_2MBPS);*/

// khai báo bút bấm và đèn
pinMode(led,OUTPUT);  pinMode(nut_on,INPUT);  pinMode(nut_off,INPUT);

//Kết nối các cảm biến
Wire.begin();
TCA9548A(1,ADD);
mpu6050_0.begin();  mpu6050_0.calcGyroOffsets(true);

TCA9548A(0,ADD);  TCA9548A(6,ADD_S);
mpu6050A_1.begin();  mpu6050A_1.calcGyroOffsets(true);

TCA9548A(0,ADD);  TCA9548A(5,ADD_S);
mpu6050A_2.begin();  mpu6050A_2.calcGyroOffsets(true);

TCA9548A(0,ADD);  TCA9548A(4,ADD_S);
mpu6050A_3.begin();  mpu6050A_3.calcGyroOffsets(true);

 TCA9548A(0,ADD);    TCA9548A(3,ADD_S);
 mpu6050B_1.begin();    mpu6050B_1.calcGyroOffsets(true);

 TCA9548A(0,ADD);    TCA9548A(2,ADD_S);
 mpu6050B_2.begin();    mpu6050B_2.calcGyroOffsets(true);

 TCA9548A(0,ADD);    TCA9548A(7,ADD_S);    TCA9548A(1,ADD_S_S);
 mpu6050B_3.begin();    mpu6050B_3.calcGyroOffsets(true);

TCA9548A(0,ADD);  TCA9548A(7,ADD_S);  TCA9548A(0,ADD_S_S);
mpu6050C_1.begin();  mpu6050C_1.calcGyroOffsets(true);

TCA9548A(0,ADD);  TCA9548A(0,ADD_S);
mpu6050C_2.begin();  mpu6050C_2.calcGyroOffsets(true);
```

```
  TCA9548A(0,ADD);  TCA9548A(1,ADD_S);
  mpu6050C_3.begin();  mpu6050C_3.calcGyroOffsets(true);

  TCA9548A(0,ADD);  TCA9548A(7,ADD_S);  TCA9548A(4,ADD_S_S);
  mpu6050D_1.begin();  mpu6050D_1.calcGyroOffsets(true);

  TCA9548A(0,ADD);  TCA9548A(7,ADD_S);  TCA9548A(3,ADD_S_S);
  mpu6050D_2.begin();  mpu6050D_2.calcGyroOffsets(true);

  TCA9548A(0,ADD);  TCA9548A(7,ADD_S);  TCA9548A(2,ADD_S_S);
  mpu6050D_3.begin();  mpu6050D_3.calcGyroOffsets(true);

  TCA9548A(0,ADD);  TCA9548A(7,ADD_S);  TCA9548A(5,ADD_S_S);
  mpu6050E_1.begin();  mpu6050E_1.calcGyroOffsets(true);

  TCA9548A(0,ADD);  TCA9548A(7,ADD_S);  TCA9548A(6,ADD_S_S);
  mpu6050E_2.begin();  mpu6050E_2.calcGyroOffsets(true);

  TCA9548A(0,ADD);  TCA9548A(7,ADD_S);  TCA9548A(7,ADD_S_S);
  mpu6050E_3.begin();  mpu6050E_3.calcGyroOffsets(true);
}

void loop() {
  // Đọc giá trị nút
  int on_State =  digitalRead(nut_on);
  int off_State = digitalRead(nut_off);

  if(on_State != macdinh){
    if(on_State == HIGH)      count = 1;
  }macdinh = on_State;

  if(off_State != macdinh){
    if(off_State == HIGH)      count = 0;
  }macdinh = off_State;
```

```
if(count == 1){
digitalWrite(led,HIGH);

// Chuyển cổng và lấy giá trị base từ cảm biến !
TCA9548A(1,ADD);
mpu6050_0.update();
float gocbase_x = mpu6050_0.getAngleX();
float gocbase_y = mpu6050_0.getAngleY();

// ngón cái!
TCA9548A(0,ADD);  TCA9548A(6,ADD_S);
mpu6050A_1.update();
float gocA_1_x = mpu6050A_1.getAngleX();
float gocA_1_y = mpu6050A_1.getAngleY();

TCA9548A(0,ADD);  TCA9548A(5,ADD_S);
mpu6050A_2.update();
float gocA_2 = mpu6050A_2.getAngleX();

TCA9548A(0,ADD);  TCA9548A(4,ADD_S);
mpu6050A_3.update();
float gocA_3 = mpu6050A_3.getAngleX();

// ngón trỏ !
TCA9548A(0,ADD);  TCA9548A(3,ADD_S);
mpu6050B_1.update();
float gocB_1 = mpu6050B_1.getAngleX();

TCA9548A(0,ADD);  TCA9548A(2,ADD_S);
mpu6050B_2.update();
float gocB_2 = mpu6050B_2.getAngleX();

TCA9548A(0,ADD);  TCA9548A(7,ADD_S);  TCA9548A(1,ADD_S_S);
mpu6050B_3.update();
float gocB_3 = mpu6050B_3.getAngleX();
```

```
// ngón giữa
TCA9548A(0,ADD);  TCA9548A(7,ADD_S);  TCA9548A(0,ADD_S_S);
mpu6050C_1.update();
float gocC_1 = mpu6050C_1.getAngleX();

TCA9548A(0,ADD);  TCA9548A(0,ADD_S);
mpu6050C_2.update();
float gocC_2 = mpu6050C_2.getAngleX();

TCA9548A(0,ADD);  TCA9548A(1,ADD_S);
mpu6050C_3.update();
float gocC_3 = mpu6050C_3.getAngleX();

// ngón áp út
TCA9548A(0,ADD);  TCA9548A(7,ADD_S);  TCA9548A(4,ADD_S_S);
mpu6050D_1.update();
float gocD_1 = mpu6050D_1.getAngleX();

TCA9548A(0,ADD);  TCA9548A(7,ADD_S);  TCA9548A(3,ADD_S_S);
mpu6050D_2.update();
float gocD_2 = mpu6050D_2.getAngleX();

TCA9548A(0,ADD);  TCA9548A(7,ADD_S);  TCA9548A(2,ADD_S_S);
mpu6050D_3.update();
float gocD_3 = mpu6050D_3.getAngleX();

// Ngón út
TCA9548A(0,ADD);  TCA9548A(7,ADD_S);  TCA9548A(5,ADD_S_S);
mpu6050E_1.update();
float gocE_1 = mpu6050E_1.getAngleX();

TCA9548A(0,ADD);  TCA9548A(7,ADD_S);  TCA9548A(6,ADD_S_S);
mpu6050E_2.update();
float gocE_2 = mpu6050E_2.getAngleX();
```

```
  TCA9548A(0,ADD);  TCA9548A(7,ADD_S);  TCA9548A(7,ADD_S_S);
  mpu6050E_3.update();
  float gocE_3 = mpu6050E_3.getAngleX();

  // tính góc quay của từng đốt và gửi lên Serial dưới dạng chuỗi.
  setngoncai(gocbase_y,gocA_1_x,gocA_1_y,gocA_2,gocA_3,'A');
  setgoc(gocbase_x,gocB_1,gocB_2,gocB_3,'B');
  setgoc(gocbase_x,gocC_1,gocC_2,gocC_3,'C');
  setgoc(gocbase_x,gocD_1,gocD_2,gocD_3,'D');
  setgoc(gocbase_x,gocE_1,gocE_2,gocE_3,'E');
}else{
  digitalWrite(led,LOW);  delay(10);
}
}
void TCA9548A(uint8_t bus, byte address)
{
  Wire.beginTransmission(address);  // TCA9548A address is 0x70
  Wire.write(1 << bus);          // send byte to select bus
  Wire.endTransmission();
}

void  setgoc(float gocbase, float gockhau1_0,float gockhau2_0,float gockhau3_0,char name){
   // Tính góc các đốt
   float gockhau1_raw =  abs(gockhau1_0 - gocbase);
   float gockhau2_raw =  abs(gockhau2_0 - gockhau1_0);
   float gockhau3_raw =  abs(gockhau3_0 - gockhau2_0);

   float gockhau1 = abs(simpleKalmanFilter1.updateEstimate(gockhau1_raw));
   float gockhau2 = abs(simpleKalmanFilter2.updateEstimate(gockhau2_raw));
   float gockhau3 = abs(simpleKalmanFilter3.updateEstimate(gockhau3_raw));

  // lưu các giá trị vừa tính được và mảng và gửi đi qua NRF24
/*  receivedData[0] = gockhau1;
```

```
 receivedData[1] = gockhau2;
 receivedData[2] = gockhau3;
 radio.write(&receivedData, sizeof(receivedData));*/

 // In dữ liệu ra Serial
 Serial.print(name);  Serial.print("1:");  Serial.print(gockhau1);  Serial.print(";");
 Serial.print(name);  Serial.print("2:");  Serial.print(gockhau2);  Serial.print(";");
 Serial.print(name);  Serial.print("3: ");  Serial.print(gockhau3);  Serial.println(";");
}
void setngoncai(float gocbase,float gockhau1_0_x, float gockhau1_0_y, float gockhau2_0, float gockhau3_0,char name){
    // Tính góc các đốt
  float gockhau1_raw = abs(gockhau1_0_y - gocbase);
  float gockhau2_raw = abs(gockhau2_0 - gockhau1_0_x);
  float gockhau3_raw = abs(gockhau3_0 - gockhau2_0);

  float gockhau1 = abs(simpleKalmanFilter1.updateEstimate(gockhau1_raw));
  float gockhau2 = abs(simpleKalmanFilter2.updateEstimate(gockhau2_raw));
  float gockhau3 = abs(simpleKalmanFilter3.updateEstimate(gockhau3_raw));

 // lưu các giá trị vừa tính được và mảng và gửi đi qua NRF24
 /*receivedData[0] = gockhau1;
 receivedData[1] = gockhau2;
 receivedData[2] = gockhau3;
 radio.write(&receivedData, sizeof(receivedData));*/

 // In dữ liệu ra Serial
 Serial.print(name);  Serial.print("1:");  Serial.print(gockhau1);  Serial.print(";");
 Serial.print(name);  Serial.print("2:");  Serial.print(gockhau2);  Serial.print(";");
 Serial.print(name);  Serial.print("3: ");  Serial.print(gockhau3);   Serial.println(";");
}
```