BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC PHENIKAA

BÁO CÁO TỔNG KẾT

TÊN ĐỀ TÀI:

Xây dựng phần mềm mô phỏng cánh tay Robto mini phục vụ đào tạo

Lĩnh vực: Cơ khí – Cơ điện tử
Chuyên ngành: Cơ điện tử

Sinh viên thực hiện chính: Bùi Thái Dương Nam, Nữ: Nam
Người hướng dẫn chính: PGS.TS. Vũ Lê Huy

Hà Nội, tháng 5 năm 2021

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC PHENIKAA

BÁO CÁO TỔNG KẾT

TÊN ĐỀ TÀI:

Xây dựng phần mềm mô phỏng cánh tay Robto mini phục vụ đào tạo

Lĩnh vực: Cơ khí – Cơ điện tử

Chuyên ngành: Cơ điện tử

Nhóm sinh viên thực hiện: Bùi Thái Dương

Lớp, khoa: k14 Năm thứ: nhất /Số năm đào tạo: 4,5 năm

Ngành học: Kỹ thuật cơ điện tử

Người hướng dẫn: PGS.TS. Vũ Lê Huy

Hà Nội, tháng 5 năm 2021

MỤC LỤC

Danh mục hình ảnh

Danh mục bảng biểu

Mở đầu

Hiện nay, việc robot 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 có những hiểu biết về robot chỉ dừng lại ở các hình ảnh như: những chú robot với trí thông minh nhân tạo, có thể giao tiếp, giúp đỡ việc nhà, bán hàng trong siêu thị, chăm sóc các cụ già hay trông trẻ… Và hầu hết chúng ta đều không để ý đến vai trò đặc biệt quan trọng của cánh tay robot trong những ngành công nghiệp, chế tạo, lắp ráp và sản xuất hiện đại ở khắp các nhà máy, công xưởng trên thế giới.

Cánh tay robot là một trong những robot phổ biến nhất được sử dụng trong các quy trình sản xuất. Cánh tay robot đều được lập trình và sử dụng trong hầu hết các trường hợp để thực hiện các nhiệm vụ cụ thể, phổ biến nhất cho sản xuất, chế tạo và các ứng dụng công nghiệp.

Tuy nhiên việc lập trình lệnh điều khiển cánh tay robot 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 và thực hành ngay trên các tay máy robot công nghiệp khi chưa thành thạo và hiểu rõ chuyển động của robot là hết sức nguy hiểm. Do đó, sử dụng phần mềm để thực hiện mô phỏng hoạt động của robot là rất quan trọng. Trong đào tạo thì nhu cầu thiết kế và mô phỏng hoạt động của các tay máy được mong muốn sử dụng rất nhiều giúp người học nắm chắc về cấu tạo, nguyên lý hoạt động, giải các bài toán động học, cũng như điều khiển robot. Tuy nhiên các phần mềm để phục vụ chuyên biệt để mô phỏng robot trong đào thì còn rất hạn chế, đó cũng là lí do ta chọn đề tài mô phỏng cánh tay robot cỡ nhỏ phục vụ đào tạo.

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.

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, đây à một 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.

Bên cạnh đó còn cócác 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.

Việc mô phỏng chuyển động cánh tay robot có thể giúp ta hiểu được nguyên lý cấu tạo và hoạt động của cánh tay robot. Tạo ra được lệnh điều khiển robot có thể áp dụng đến những cánh tay robot khác, giúp ta tiết kiệm thời gian lập trình chương trình chuyển động và tăng cao độ chính xác. Giúp ta hiểu hơn về phần mềm SolidWorks 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.

1.2. Lý do lựa chọn đề tài

Hiện nay tại Khoa Cơ khí – Cơ điện tử đã có mô hình tay máy robot cỡ nhoe được chế tạo phục vụ quảng bá tuyển sinh, nhưng tay máy này hoạt động dựa trên một quỹ đạo được xác định trước trong bộ điều khiển. Để thực hiện mô phỏng và điều khiển theo những quỹ đạo khác chưa được thực hiện. Từ mong muốn xây dựng một phần mềm có thể mô phỏng được hoạt động của robot và tiến tới điều khiển trực tiếp tay máy robot đó, đề tài ngiên cứu và mô phỏng chuyển động tay máy robot mini 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 cánh tay robot.

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 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 của 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) [2]. 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. Tất cả những tính năng 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 2018 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. Giài 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 kết cấu tay máy robot

Về mặt cấu tạo robot được chế tạo rất khác nhau, nhưng chúng được xác định từ các thành phần cơ bản như: tay máy, nguồn cung cấp, bộ điều khiển [1].

Tay máy hay có thể là cánh tay cơ khí của robot công nghiệp thông thường là chuỗi hở được tạo thành từ nhiều khâu được liên kết với nhau nhờ các khớp động, khâu cuối của tay máy thường có dạng một tay gắp hoặc gắn công cụ thao tác. Mỗi khâu động trên tay máy có nguồn dẫn động riêng năng lượng và chuyển động truyền đến cho chúng được điều khiển trên cơ sở tín hiệu nhận được từ bộ phận phản hồi.

Thông thường các tay máy có trên một bậc tự do, số bậc tự do hay bậc chuyển động của tay máy là khả năng chuyển động độc lập của nó trong không gian hoạt động. Trong lĩnh vực robot học người ta gọi mỗi khả năng chuyển động (có thể là chuyển động tịnh tiến dọc theo trục song song với một trục khác hoặc chuyển động

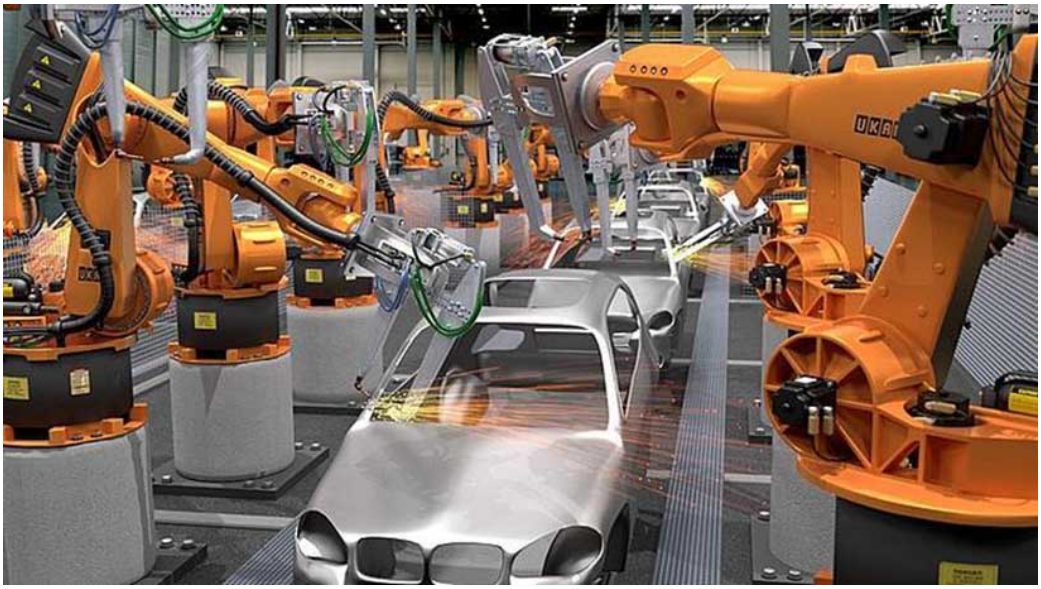
quay quanh trục là một trục tương ứng theo một trục là tọa độ suy rộng dùng để xác định vị trí của trục trong không gian hoạt động. Sáu bậc chuyển động sẽ được bố trí gồm:

Ba bậc chuyển động cơ bản hay chuyển động định vị.

Ba bậc chuyển động bổ sung hay chuyển động định hướng.

2.2.1. Tay máy robot công nghiệp

Cánh tay robot công nghiệp được hiểu là hệ thống máy mô phỏng các hoạt động của cả cánh tay với mục đích phục vụ việc hoàn thành dây chuyền sản xuất (Hình 2.2). Đây là loại máy móc bào gồm các bộ phận của cánh tay người. Hoạt động linh hoạt với những ưu điểm cũng như cơ chế vượt trội. Ngày nay, cánh tay robot công nghiệp đang được áp dụng vào nhiều lĩnh vực khác nhau, nhưng đều có một lí do đó là thay thế bàn tay của con người và làm việc với năng suất cao trong các hoạt động. Một tay máy robot thường có các thành phần chính sau:

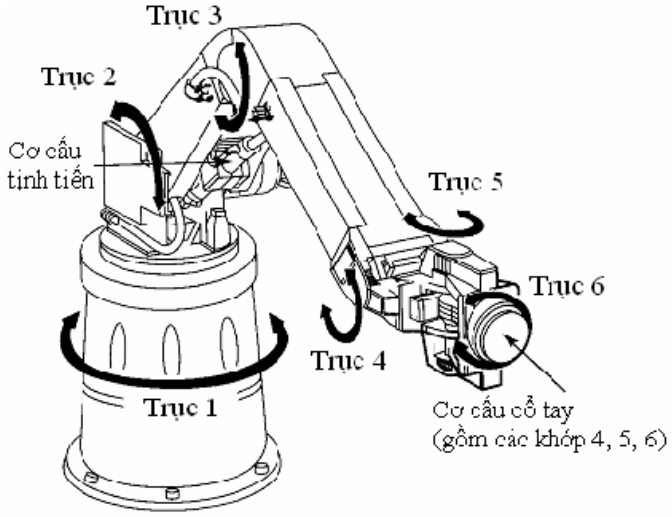


Hình 2.2: Cánh tay robot công nghiệp

a) Tay máy:

Đây là một bộ phận dùng để thực hiện các thao tác, tác động và hoàn thành công việc. Là bộ phận gắn kết các chi tiết khác với nhau nhờ các khớp (ví dụ với các khớp như trên Hình 2.3). Được chế tạo với khả năng thực hiện chuyển động dễ dàng,

cổ tay linh hoạt, khéo léo, … Cánh tay robot được lắp đặt một cách chắc chắn đảm bảo đủ độ an toàn và gắn kết.



Hình 2.3: Các khớp liên kết trong tay máy robot

b) Hệ thống điều khiển

Đây chính là bộ phận được dùng để điều khiển các thao tác sau khi tiếp nhận và xử lý các tín hiệu đến từ bên ngoài. Đối với mỗi yêu cầu hoạt động khác nhau, chúng ta sẽ có những chức năng điều khiển cánh tay robot khác nhau. Đảm nhận các hoạt động theo nhiều cấp độ như xác định vị trí, tọa độ hàng, các nơi đi qua, ...Hoặc cũng có thể tính toán các thông số như động lực, các bài tính toán, các quỹ đạo, xử lý các lỗi bắt gặp, …

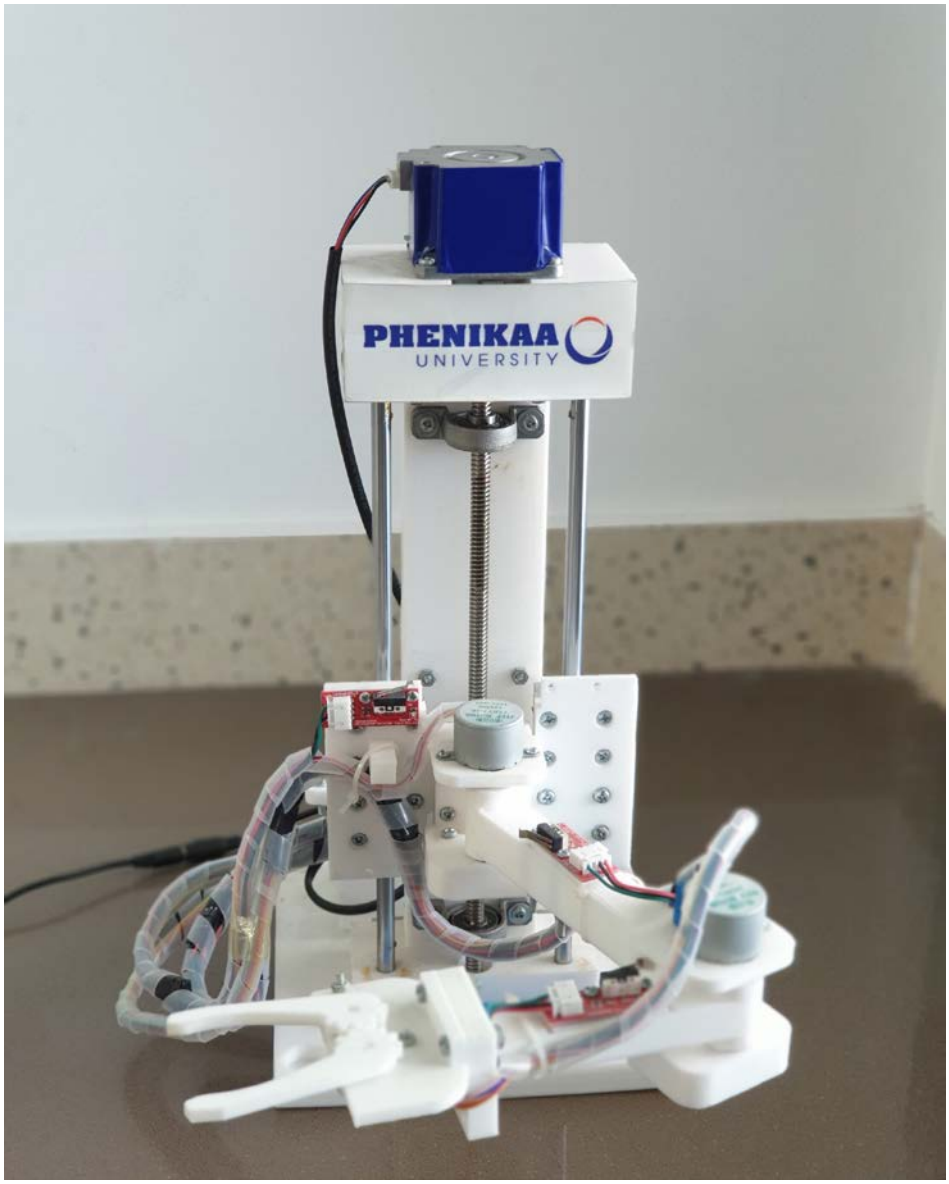
c) Phần mềm quản lý và vận hành

Là không gian lập trình, thường sẽ được sử dụng để nhận và truyền đạt thông tin về mệnh lệnh cũng như yêu cầu đến với robot. Phần mềm lập trình yêu cầu phải đồng bộ với ngôn ngữ lập trình, dễ sử dụng.

2.2.2. Tay máy robot mini cỡ nhỏ

Sau khi xem kết cấu tay máy robot công nghiệp có dạng thiết kế như nào như vậy ta đi sang việc xây dựng tay máy robot mini phục vụ đào tạo. Ở đây giới thiệu tay máy robot mini đã được chế tạo tại Khoa Cơ khí – Cơ điện tử như trên Hình 2.4. Giống như tay máy công nghiệp, tay máy robot mini có thể tịnh tiến lên xuống và quay quanh trục trong không gian. Để phục vụ cho đào tạo ta lựa chọn mô hình thiết kế kích thước sử dụng những bộ phận có sẵn như cụm trục vít, đai ốc, bo mạch Arduino, Module công tắc hành trình. Tay máy này cơ bản gồm các bộ phận:

- Khâu 0: khâu đế cố định
- Khâu 1: di chuyển tịnh tiến lên xuống nhờ truyền động trục vít đai ốc, được điều khiển bởi động cơ bước 23KM-K255-G3V đặt ở trên đỉnh robot.
- Khâu 2 và 3: là các khâu dạng thanh được liên kết với nhau bằng khớp quay, khâu 2 được liên kết với khâu 1 nhờ một khớp quay với trục quay thẳng đứng. Các chuyển động quay này được điều khiển bởi động cơ bước 28BYJ-48.
- Tay kẹp: gồm hai ngón thực hiện đóng mở để kẹp vật. Hai ngón tay được liên động với nhau bằng ăn khớp răng và được điều khiển bằng động cơ bước 28BYJ-48.

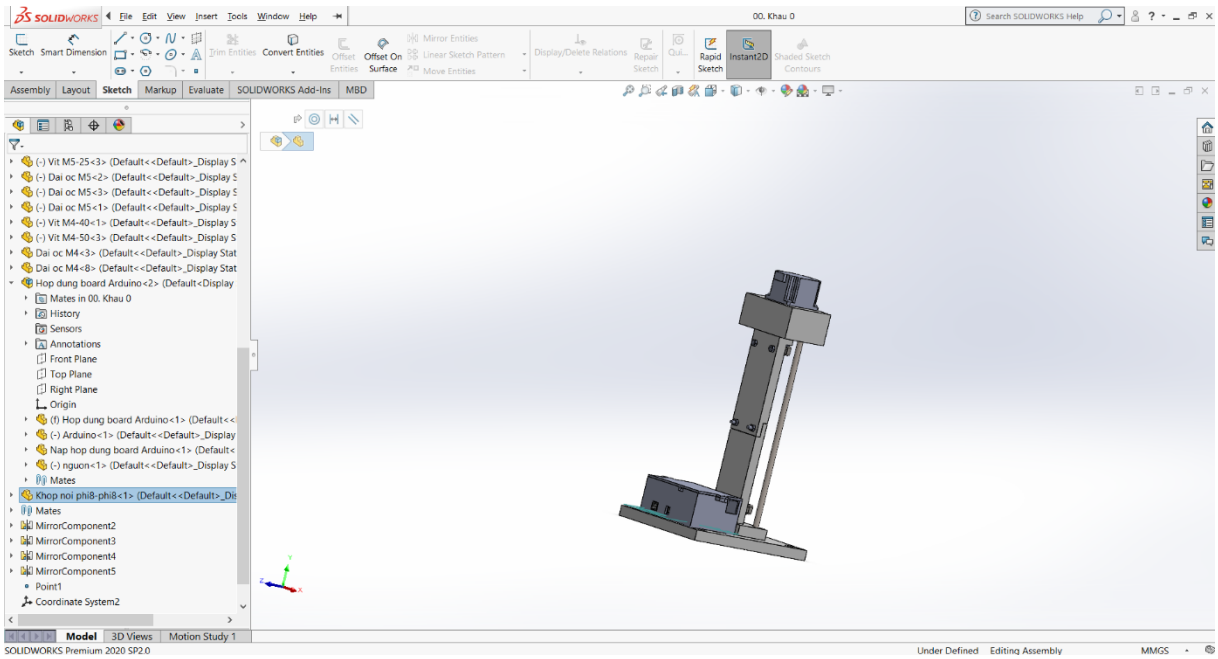


Hình 2.4: Hình ảnh tay máy robot mini

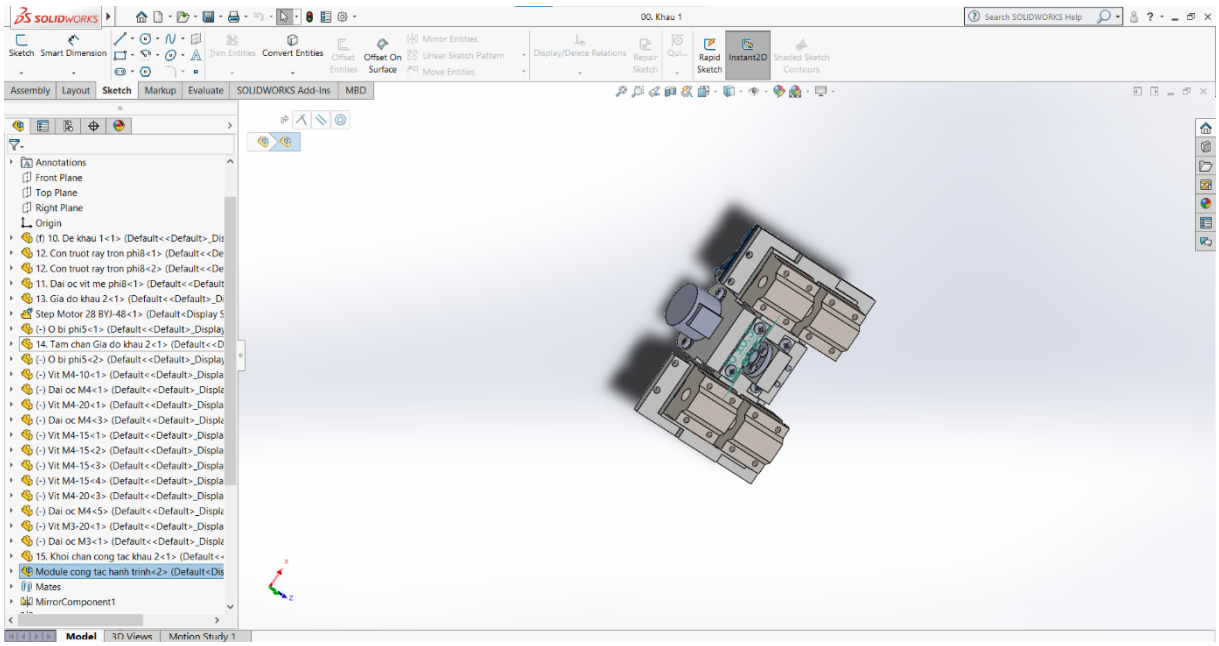
2.3. Xây dựng mô hình 3D tay máy robot cỡ nhỏ phục vụ đào tạo

Để có thể xây dựng mô hình cánh tay robot mini 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 cánh tay robot. Đầu tiên vẽ khâu 0 (khâu đế, Hình 2.5) của 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 cuả 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. Khi vẽ được đế, ta bắt đầu vẽ ray trụ, trụ đứng, trục vít(để có thể cắt được hình xoắn ốc ta sử dụng lệnh Curves và chọn Helix and Spiral), vít, đai ốc, hợp đựng mạch Arduino, khớp nối. Khi đã có đầy đủ các PART ta tiến hành lắp ráp chúng lại với nhau, đầu tiên ta lắp đế rồi đến gắn 2 ray trụ vào đế và trụ đứng cũng như vậy sau đó lắp 2 ổ để có thể cố định trục vít lại sau đó gắn Step Motor lên đỉnh của khâu 0, sau đó gắn các vít và đai ốc lại các điểm để cố định các PART và để tích kiệm thời gian ta chỉ gần gắn một bên và sử dụng Mirror để có thể tạo ra những vít và đai ốc đã gắn ở bên phía còn lại sau đó gắn khớp nối, cuối cùng là gắn hộp đựng bo mạch Arduino ở dưới phần đế để hoàn thành khâu 0.

Tiếp đến thực hiện vẽ khâu 1 (Hình 2.6), về nguyên lý các vẽ PART thì đều sử dụng Sketch và Extruded Boss giống như vẽ PART ở khâu 0. Để có khâu 1 ta cần vẽ khâu đế, con trượt ray tròn, đai ốc vít, Step motor, tấm chân giá đỡ khâu, ốc bi, đai ốc, vít, khối chân công tắc khâu, và module công tắc hành trình. Sau khi đã có đầy đủ các PART ta bắt đầu lắp ráp khâu 1.



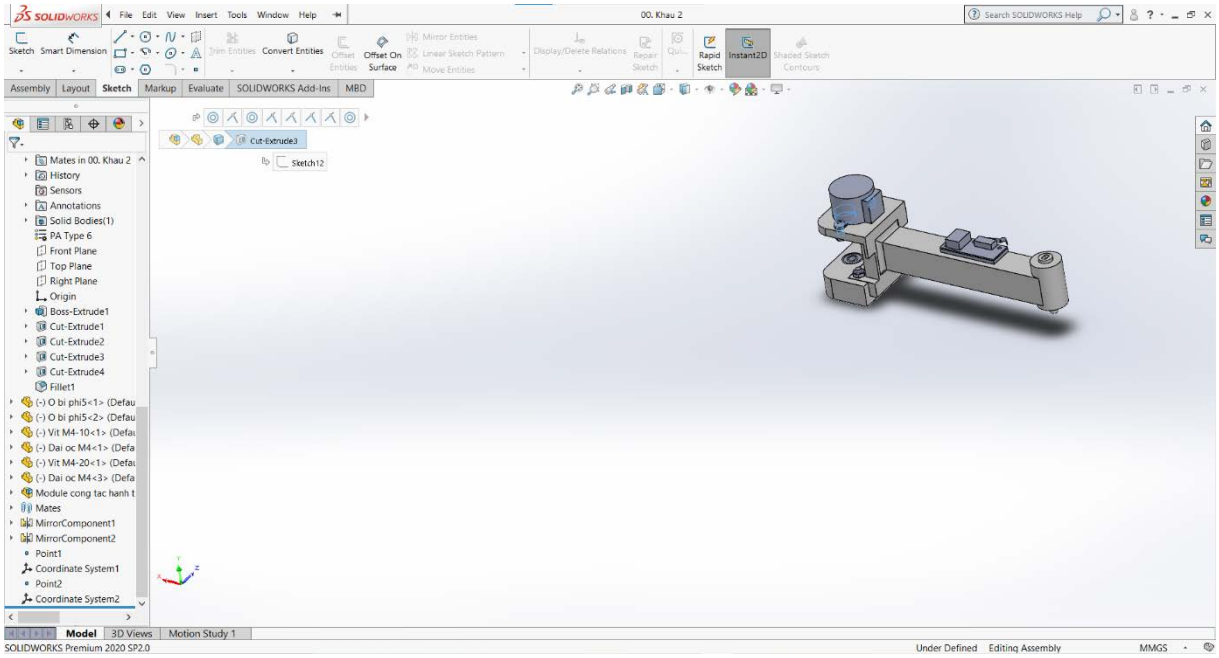
Hình 2.5: Khâu 0



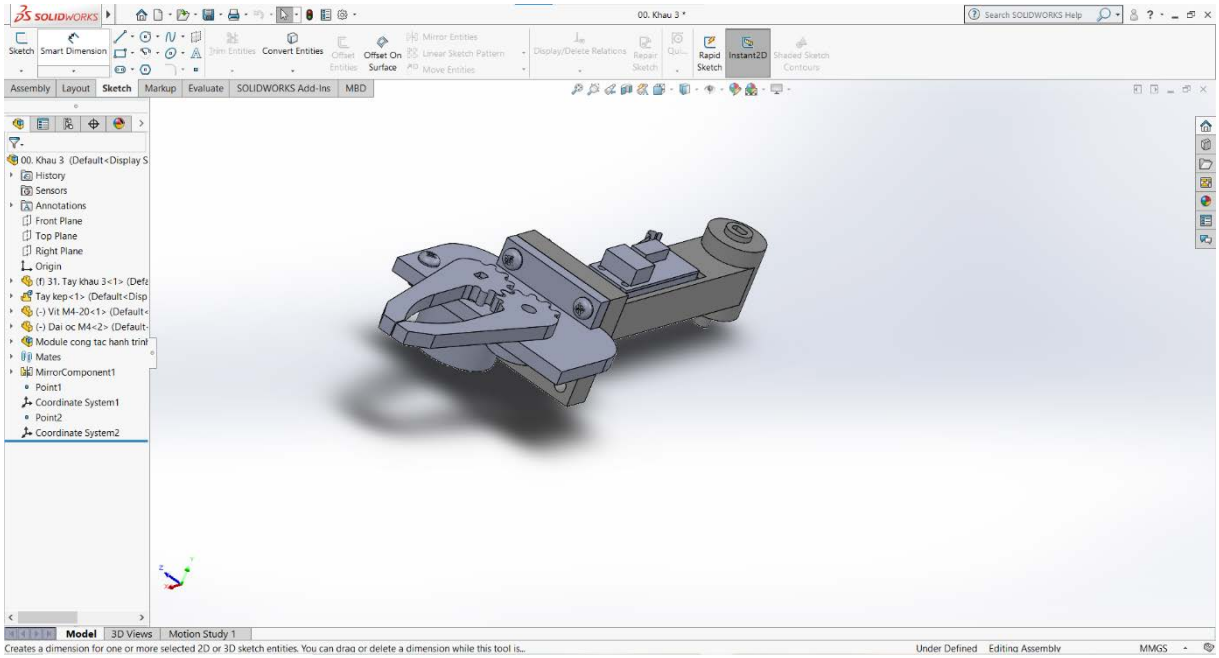
Hình 2.6: Khâu 1

Đến khâu 2 (Hình 2.7), vẽ tay khâu, đai ốc, vít, và module công tắc hành trình và lắp ráp chúng lại với nhau. Tiếp theo là khâu 3 được vẽ cũng như khâu 2 và thêm tay kẹp (Hình 2.8).

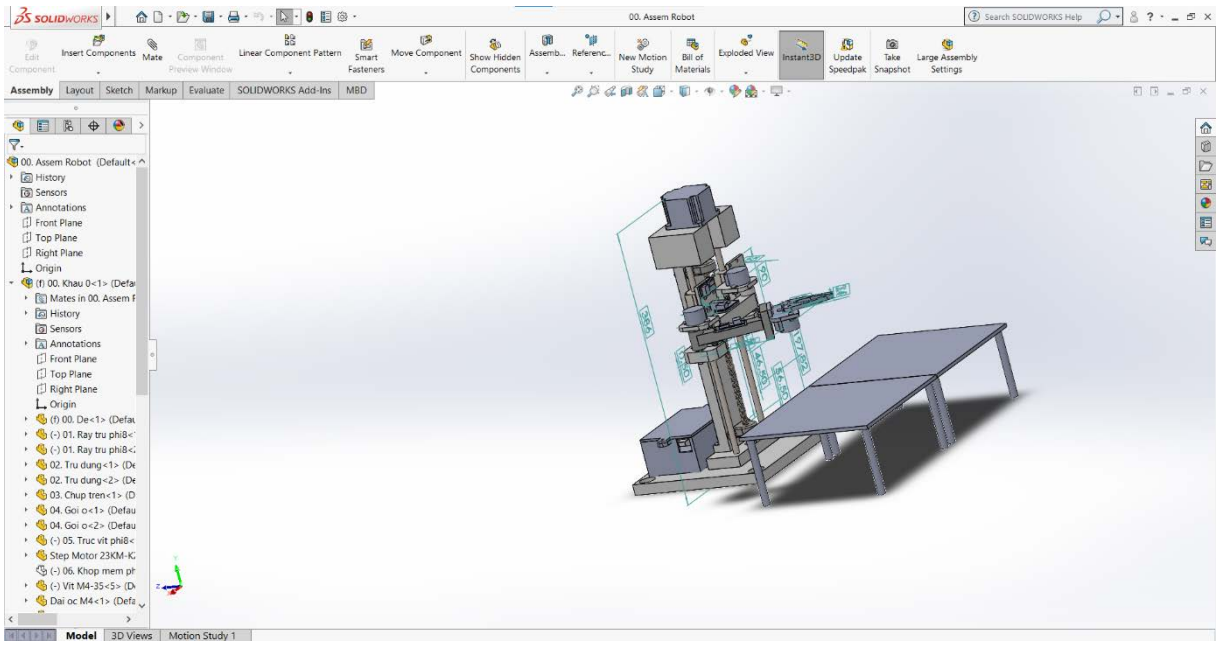
Sau khi đã vẽ đầy đủ 4 khâu ta bắt đầu lắp ráp 4 khâu lại với nhau đầu theo thứ tự từ khâu 0 đến khâu 3. Khi lắp ráp xong ta sẽ có một tay máy robot mini hoàn thiện trên Solidworks như ở Hình 2.9.



Hình 2.7: Khâu 2



Hình 2.8: Khâu 3

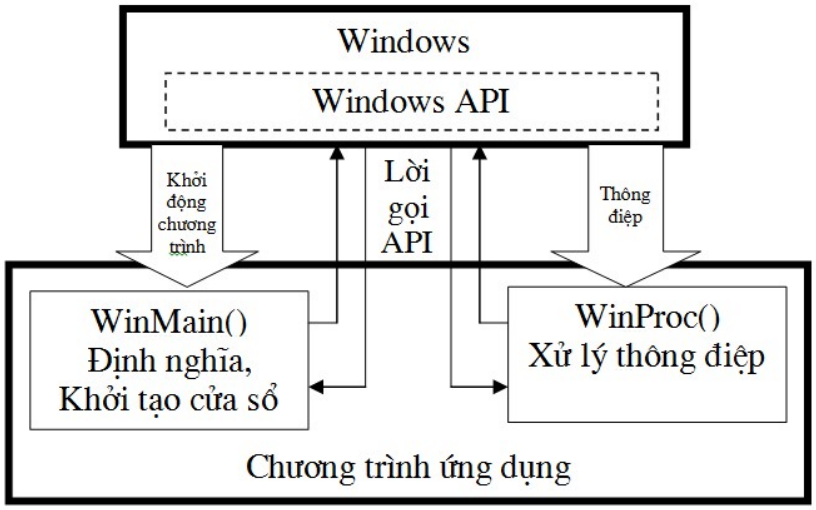


Hình 2.9: Mô hình tay máy robot mini

Chương 3. Xây dựng phần mềm mô phỏng robot

3.1. Tìm hiểu lập trình ứng dụng Windows bằng Visual Studio 2019

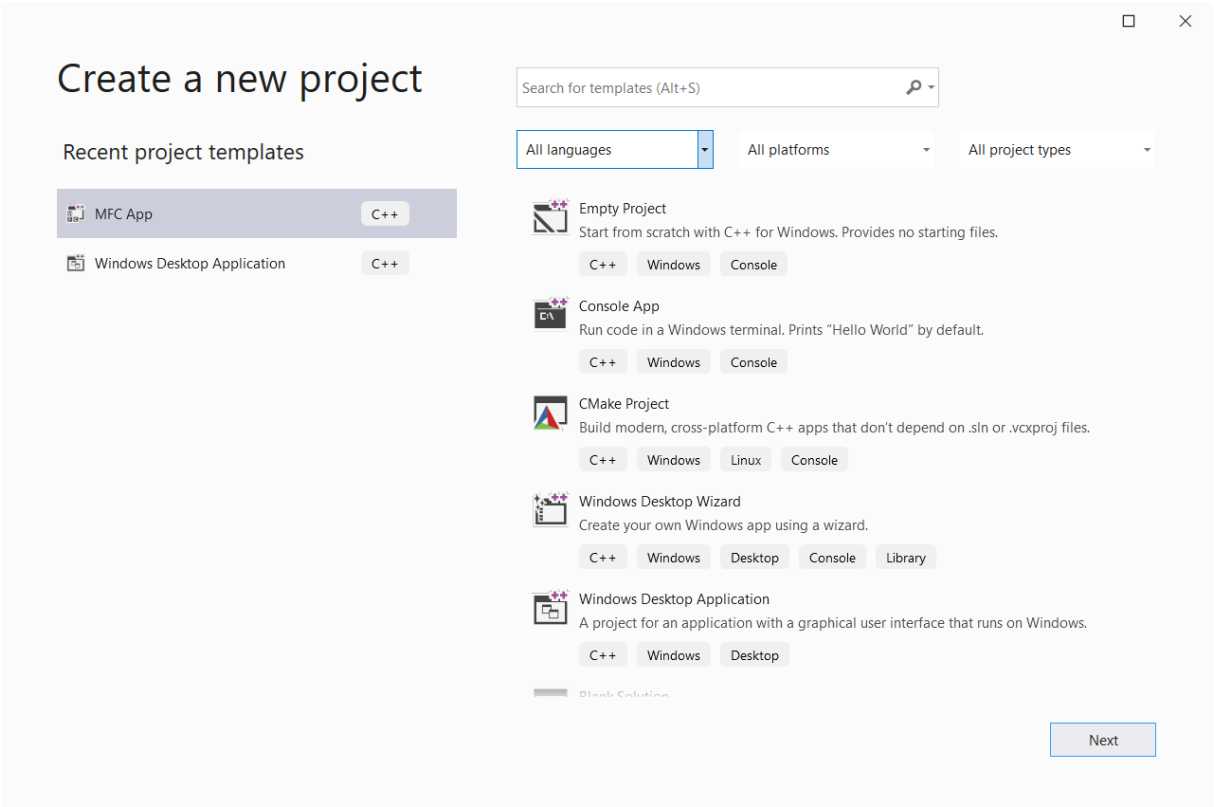
Nguyên lý cơ bản của lập trình Window là cơ sở xử lý thông điệp như mô tả trên Hình 4.1 [3]. Tuy nhiên để lập chương trình bằng các hàm API là vô cùng phức tạp và rất khó quản lý, do đó với lợi thế của ngôn ngữ lập trình hướng đối tượng, Microsoft đã tập hợp các hàm API đó lại và đóng gọi lại thành các 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

Để tạo đề án (project) khung cho chương trình mô phỏng, chúng tôi dùng tiến trình tạo đề án của bộ Microsoft Visual Studio với phiên bản Visual Studio 2019. 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 4.2. 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++

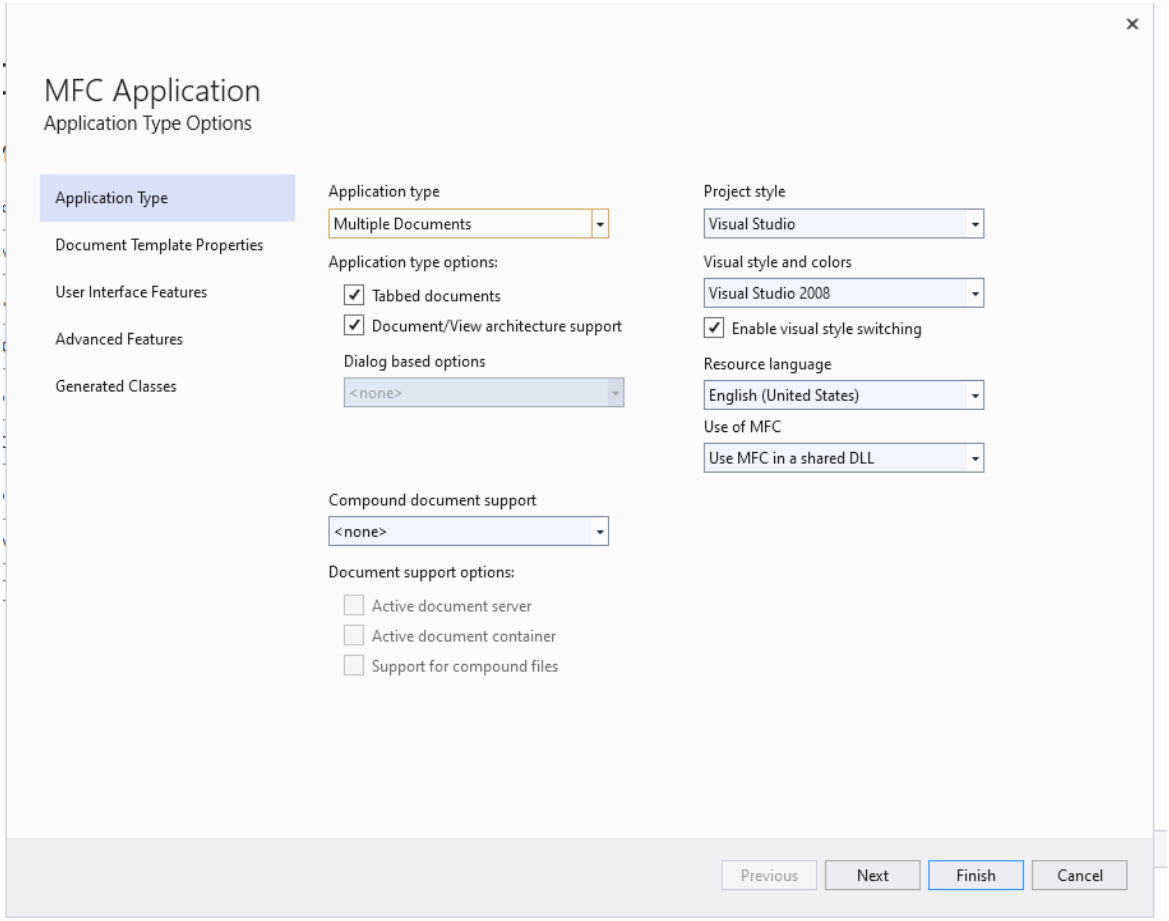
Bước 1 (Step 1): là bước đầu tiên của tiến trình, cho phép người dùng lựa chọn dạng khung giao diện ứng dụng muốn tạo (Hình 4.3).

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.

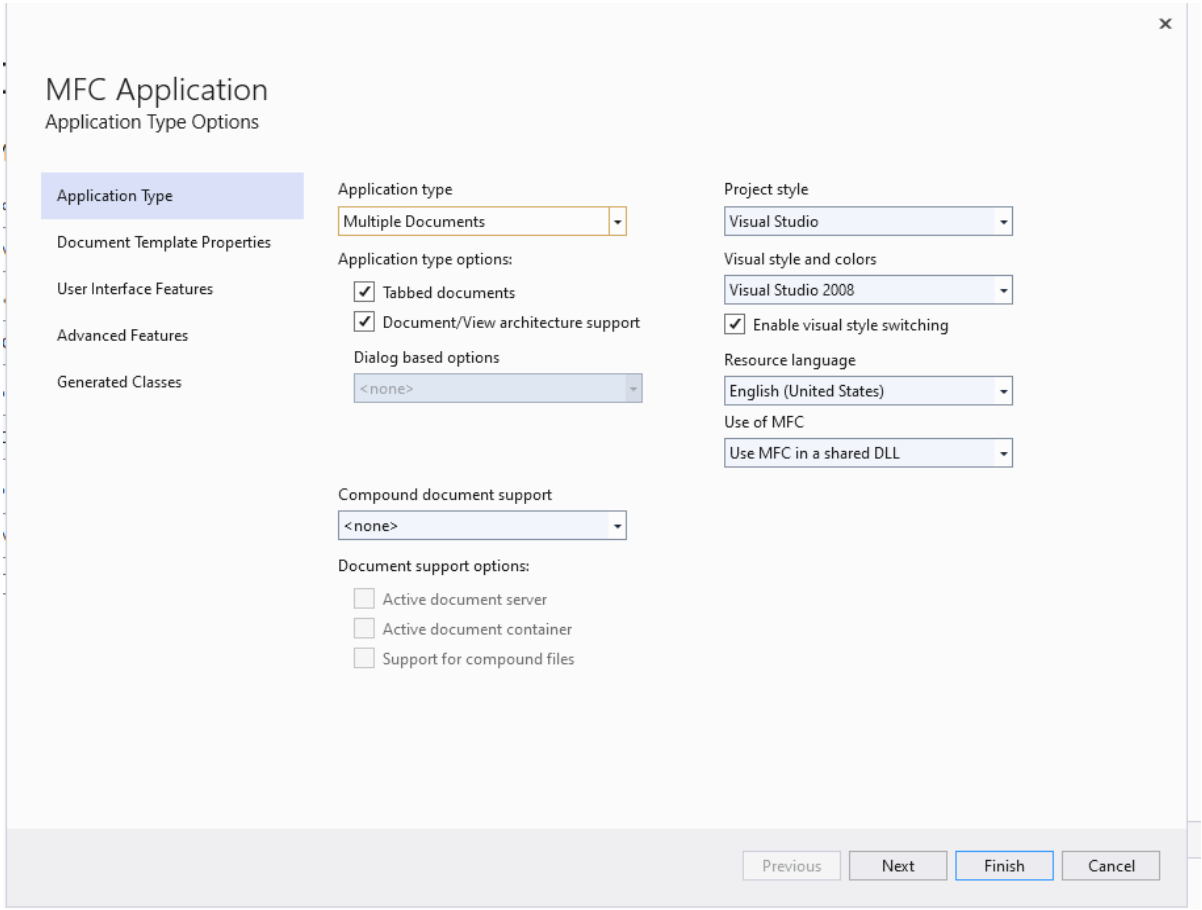
Với mục đích của chương trình mô phỏng ở đề tài này, chúng tôi lựa chọn dạng khung ứng dụng đơn tài liệu (Single Document).



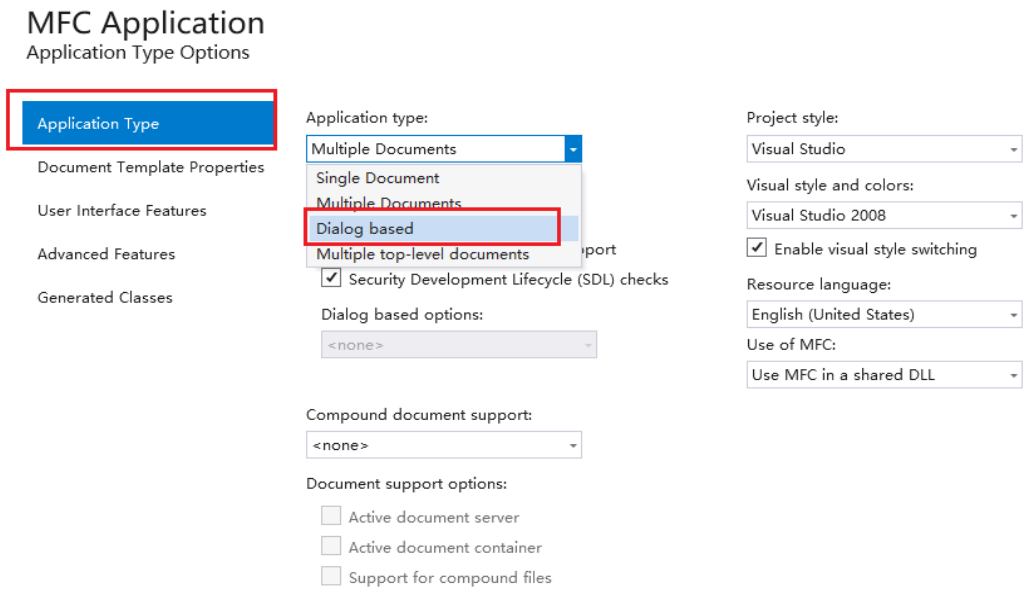
Hình 3.3: Giao diện lựa chọn dạng giao diện ứng dụng muốn tạo

Bước 2 và 3 có giao diện như trên Hình 4.4. 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 cần nhấn nút "Next" để chuyển qua bước tiếp theo.

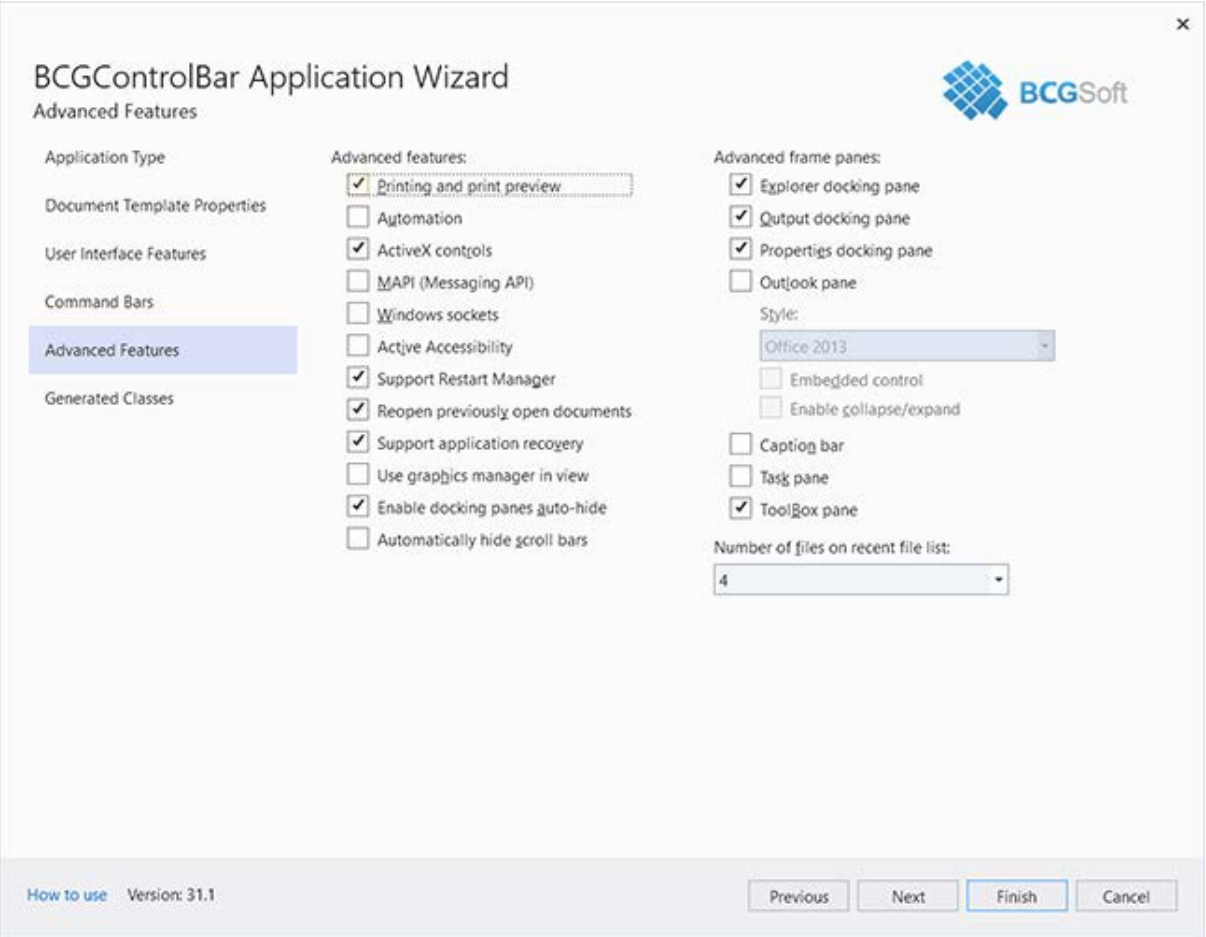
Bước 4: cho phép tùy chọn một số thể hiện trên giao diện của ứng dụng sẽ được tạo lập (Hình 4.5). Chẳng hạn như có tạo ra các thanh công cụ có khả năng thả nổi và bắt dính vào khung ứng dụng hay không, có thanh trạng thái hay không, có chức năng in và quan sát trước khi in hay không, …



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



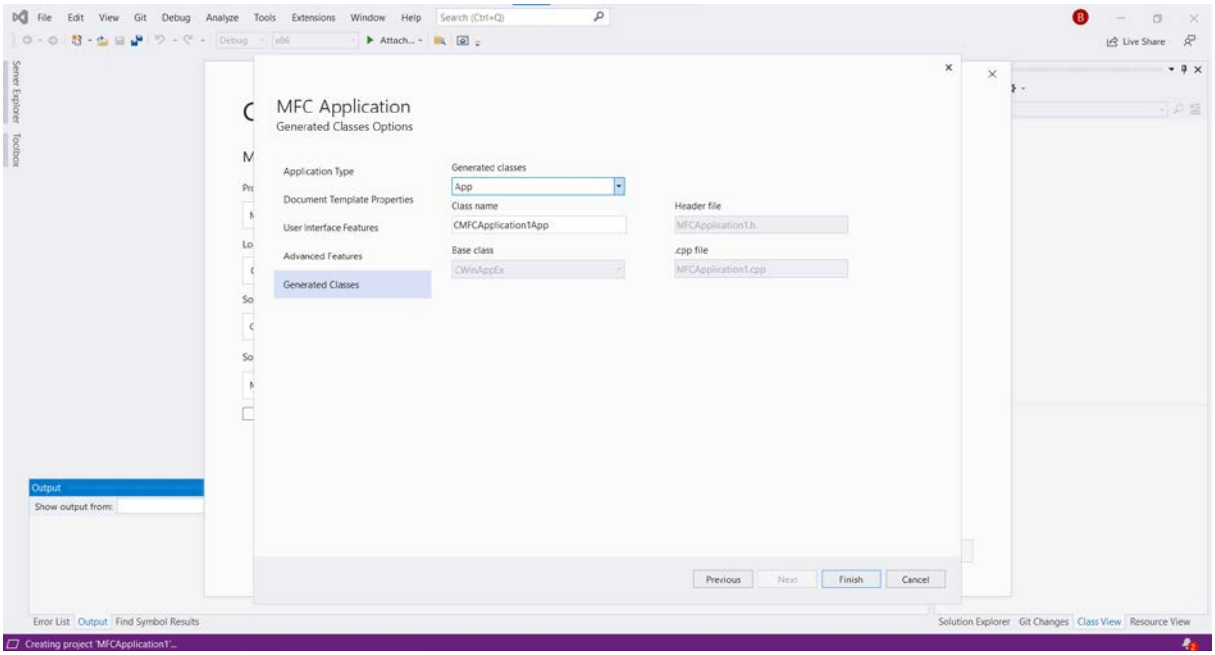
Hình 3.5: Giao diện của bước 4 trong tiến trình tạo đề án



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 4.6. 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 4.7, 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 4.8. 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".

3.1.1. Làm việc với Menu, ToolBar, Status

3.1.1.1. 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).

3.1.1.2. 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.

3.1.2. Vẽ trên Windows

Để thực hiện công việc vẽ bạn cần có biến quản lý công việc vẽ thuộc kiểu CDC hoặc CPaintDC, CClientDC, ... (đều kế thừa từ CDC), lớp này cần được nhận được con trỏ của lớp điều khiển sẽ được vẽ lên đó.

CDC : Class of Device-Context, lớp ngữ cảnh thiết bị, để vẽ tất cả các thứ đều thông qua các hàm thành viên của lớp này. CDC cung cấp các chức năng thao tác ngữ cảnh thiết bị, làm việc với các công cụ vẽ, lựa chọn các đối tượng giao diện thiết bị đồ hoạ (GDI : Graphics Device Interface), làm việc với mầu và bảng mầu.

Vẽ trong cửa sổ của chương trình Dialog thông qua hàm OnPaint(), còn với khung View thì thông qua hàm OnDraw(CDC *pDC). Một số hàm vẽ của CDC :

```
ArcTo LineTo        Polygon        TextOut
Ellipse MoveTo Rectangle  DrawText
FillRect        Pie      RoundRect    ….
FloodFill       PolyBezier    SetTextColor
```

CPen quản lý các nét vẽ với kiểu đường, độ rộng và mầu vẽ.

CBrush quản lý vùng tô với kiểu tô, mầu tô.

CFont quản lý kiểu chữ.

Để chọn một kiểu bút và chổi tô mới cho CDC sử dụng hàm :

CDC::SelectObject

Nếu sử dụng các đối tượng bút, chổi tô và font chữ được định nghĩa sẵn trong Window:

virtual CGdiObject* SelectStockObject(intnIndex);

với nIndex là một trong những giá trị:

BLACK_BRUSH DKGRAY_BRUSH GRAY_BRUSH

HOLLOW_BRUSH LTGRAY_BRUSH NULL_BRUSH

WHITE_BRUSH BLACK_PEN NULL_PEN

WHITE_PEN DEVICE_DEFAULT_FONT SYSTEM_FONT

```
void CDlgTest::OnPaint()
{
CRect clientRect;
GetClientRect(clientRect);
CPaintDC dc(this);
clientRect.InflateRect(-5,-5,-5,-5);
dc.Rectangle(clientRect);
dc.DrawText("Draw Text : My HelloWorld program.",-1, clientRect,
DT_SINGLELINE|DT_CENTER|DT_VCENTER);
dc.MoveTo(5,5); dc.LineTo(100,100);
}
```

3.1.2.1. Bút vẽ

Cách đơn giản nhất để tạo bút là sử dụng đối tượng CPen và truyền cho chúng các tham số định nghĩa bút :

CPenpen(PS_SOLID,1,RGB(255,0,0));

Phương pháp thứ hai là xây dựng đối tượng bút chưa truyền tham số rồi gọi hàm thành phần CreatePen để khởi tạo bút :

CPenpen;

pen.CreatePen(PS_SOLID,1,RGB(255,0,0));

Phương pháp thứ ba là khởi tạo đối tượng CPen nhờ cấu trúc LOGPEN bằng hàm CreatePenIndirect:

CPenpen;

LOGPEN lp;

lp.lopnStyle=PS_SOLID;

lp.lopnWidth=1;

lp.lopnColor=RGB(255,0,0);

pen.CreateIndirect(&lp);

3.1.2.2. Chổi tô

Chổi tô có 3 loại cơ bản: đồng nhất (solid), sọc (hatch) và mẫu (pattern).

Chổi đồng mầu được lập chỉ với tham số giá trị mầu COLORREF :

CBrushbrush(RGB(255,0,0)); hoặc có thể thực hiện khai báo không truyền tham số và khởi tạo chổi tô đồng mầu với hàm CreateSolidBrush :

CBrushbrush;

brush.CreateSolidBrush(RGB(255,0,0));

Chổi sọc được khởi tạo ngoài tham số giá trị mầu còn cần thêm tham số xác định kiểu sọc :

CBrushbrush(HS_BDIAGONAL,RGB(255,0,0)); hoặc khởi tạo với hàm CreateSolidBrush :

CBrushbrush;

brush.CreateHatchBrush(HS_BDIAGONAL,RGB(255,0,0)); kiểu gạch có thể :

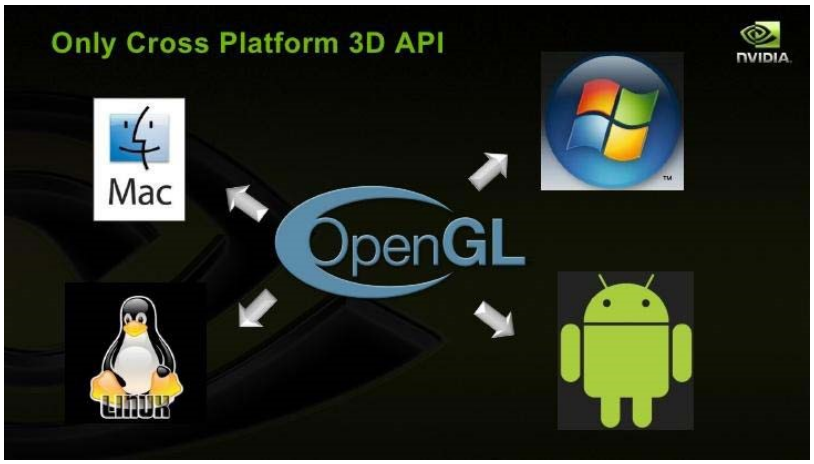
HS_BDIAGONAL, HS_CROSS, HS_DIAGCROSS,

HS_FDIAGONAL, HS_HORIZONTAL, HS_VERTICAL

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

3.2.1. OpenGL là gì?

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 [4]. 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) hộ trợ cho lập trình đồ họa.



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>

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

#pragma comment(lib, "opengl32.lib")

#pragma comment(lib, "glu32.lib")

#pragma comment(lib, "glaux.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 (rendering context)

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 1.1.

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.

Bảng 3.1: Các hàm quản lý ngữ cảnh diễn tả

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 1.2. 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.

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.

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

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 CChapter4View::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 CChapter4View::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
CChapter4View::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++.

3.2.7.1. 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 [3], 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>
#include <gl\glaux.h>
```

- Các tệp tin thư viện :

opengl32.lib

glu32.lib

glaux.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().

3.2.7.2. 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) [5], đã đóng gói toàn bộ các bước trên, do đó người lập trình không phải quan tâm tới các công việc đó nữa. 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) [5]. 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;

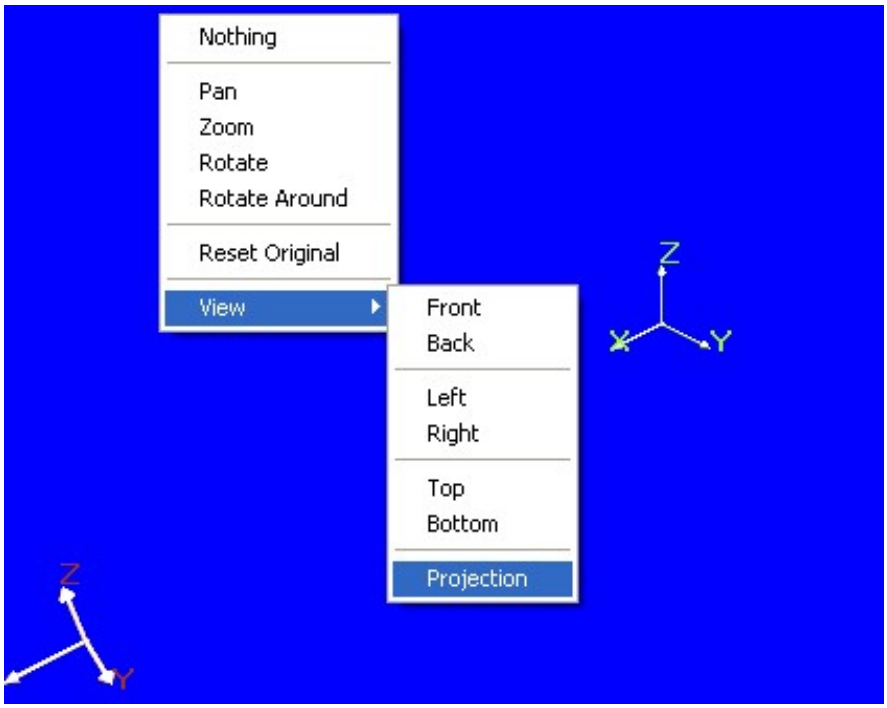
đặt trong khai báo của lớp quản lý giao diện.

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();
```

3.2.7.3. 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.

3.2.7.4. 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

3.2.8.1. 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

3.2.8.2. 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 “stdafx.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);

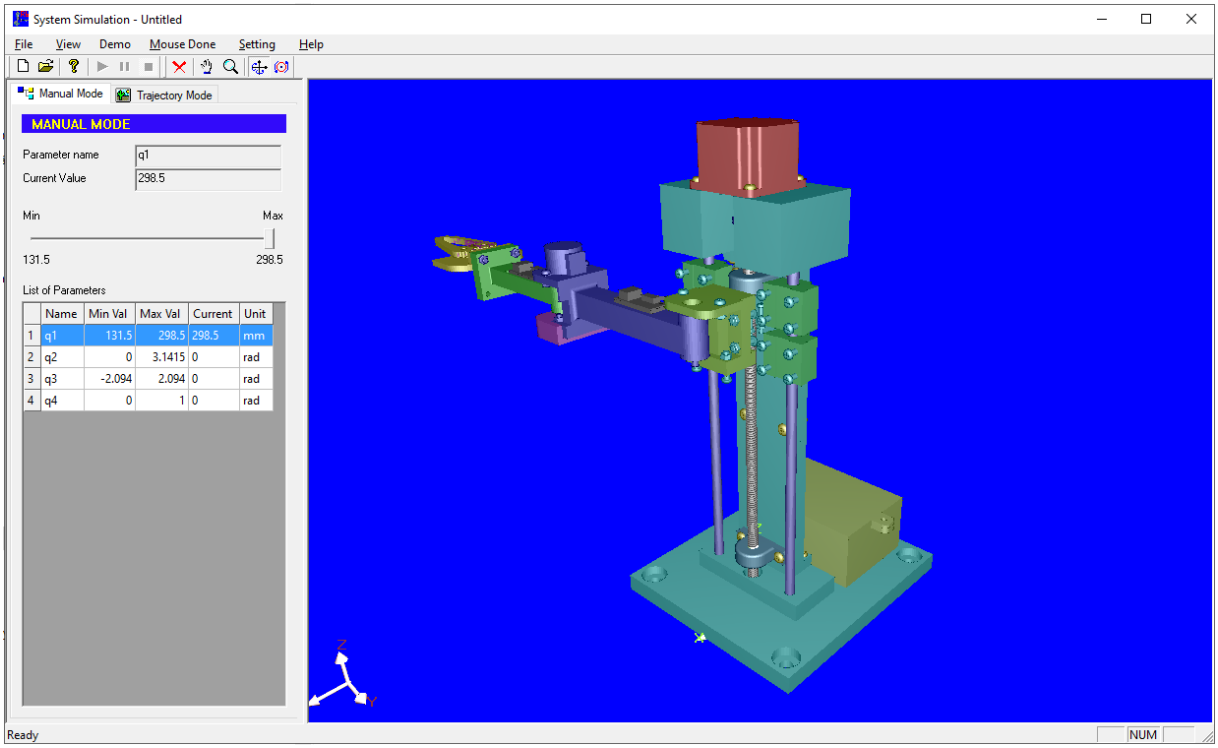
dlg.DoModal();

Sử dụng hộp thoại đặt vật liệu CDlgSettingMaterial tương tự.

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

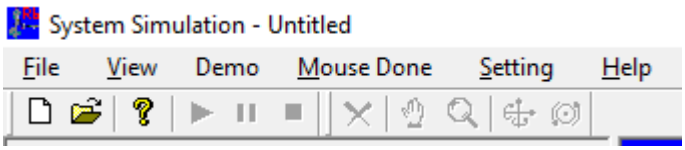
Chương trình "System Simulation" được lập trình bằng ngôn ngữ C++ trên môi trường Microsoft Visual Studio.

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 ban đầu của 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.

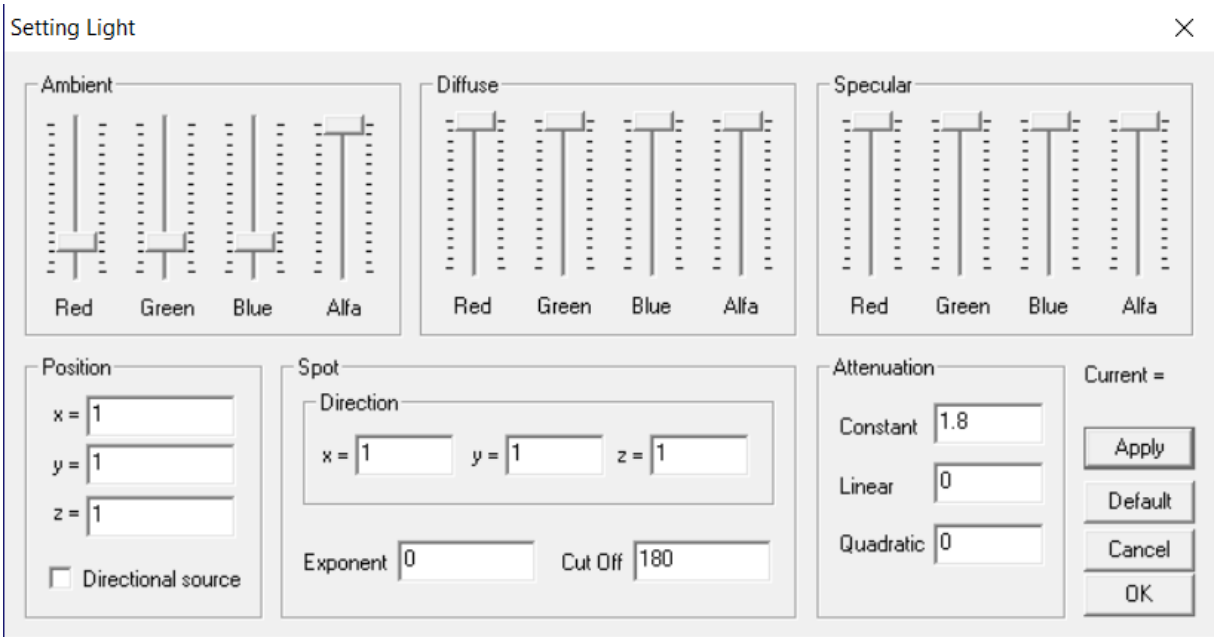
-  : dùng để phóng to hoặc thu nhỏ mô hình 3D .

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

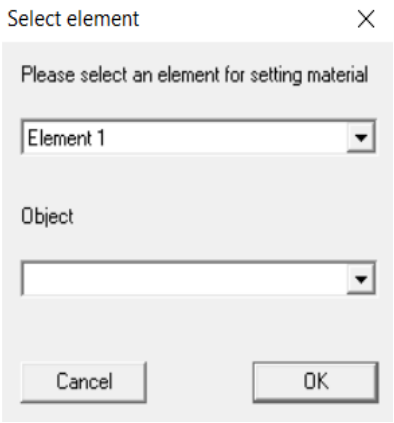
-  : dùng để xoay vật theo hướng vòng tròn.

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

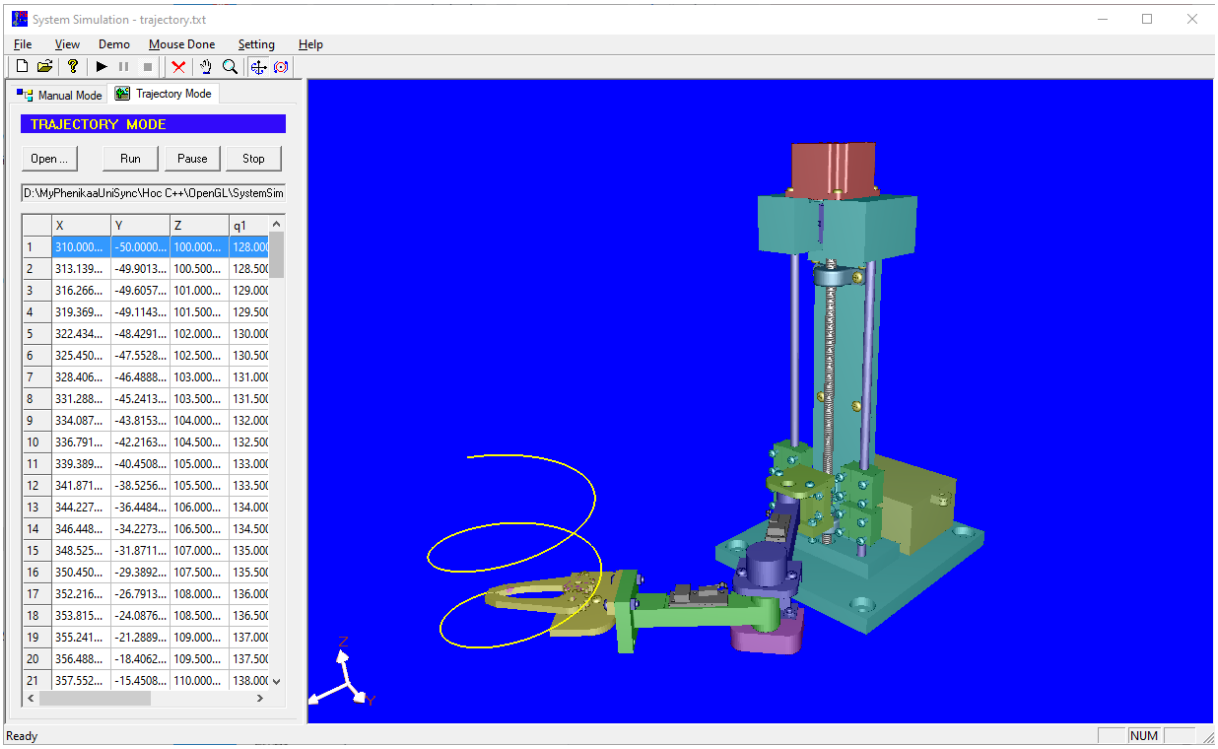


Hình 3.12: Giao diện hộp thoại lựa chọn đối tượng đặt thông số vật liệu

Trong chương trình gồm 2 chế độ đó là chế độ thủ công(Manual Mode) và chế độ quỹ đạo(Trajectory Mode).

Manual Mode cho phép ta xem được quỹ đạo chuyển động của từng khớp tay của mô hình.Khi chọn vào từng phần của bảng số liệu ta có thể điều khiển được từng khớp chuyển động giúp ta biết được giới hạn di chuyển và cách di chuyển của tay máy.

Trajectory Mode cho ta xem quỹ đạo chuyển động của tay máy bằng cách nạp vào một đoạn chương trình đã được viết sẵn để kích hoạt chuyển động tay máy.



Hình 3.13: Giao diện phần mềm ở chế độ trajectory mode

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

Mô hình tay máy robot mini 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 [6]. 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 theo ví dụ như sau:

```
[workspace]
400     ;Length of Axes
1       ;type of model file - 0: stl in ASCII ; 1: stl in binary ; 2: mdf
;num Elements
;0      1       2       3       4       5
[elements]
B0      B1      B2      B3      B4      B5
T0      T1      T2      T3      T4      T5
;
;name min     max     init    unit
[parameters]
q1      131.5   298.5   298.5   "mm"
q2      0       3.14150         "rad"
q3      -2.094  2.094   0       "rad"
q4      0       1       0       "rad"
;
[constants]
a1      55      "mm"
a2      150     "mm"
a3      190     "mm"
a4      -60     "mm"
d1      75      "mm"
```

d2 0.5 "mm"

d3 46.5 "mm"

d4 0 "mm"

;

;Transform matrixes

[T0]

1 0 0 0

0 1 0 0

0 0 1 0

0 0 0 1

[T1]

1 0 0 a1

0 1 0 0

0 0 1 q1-d1

0 0 0 1

[T2]

sin(q2) cos(q2) 0 a1+a2*sin(q2)

-cos(q2) sin(q2) 0 -a2*cos(q2)

0 0 1 q1-d1+d2

0 0 0 1

[T3]

sin(q2+q3) cos(q2+q3) 0 a1+a2*sin(q2)+a3*sin(q2+q3)

-cos(q2+q3) sin(q2+q3) 0 -a2*cos(q2)-a3*cos(q2+q3)

0 0 1 q1-d1+d2+d3

0 0 0 1

[T4]

sin(q2+q3-q4) cos(q2+q3-q4) 0
a1+a2*sin(q2)+(a3+a4)*sin(q2+q3)+10.2*cos(q2+q3)

-cos(q2+q3-q4) sin(q2+q3-q4) 0 -a2*cos(q2)-
(a3+a4)*cos(q2+q3)+10.2*sin(q2+q3)

0 0 1 q1-d1+d2+d3+d4

0 0 0 1

[T5]

sin(q2+q3+q4) cos(q2+q3+q4) 0 a1+a2*sin(q2)+(a3+a4)*sin(q2+q3)-10.2*cos(q2+q3)

-cos(q2+q3+q4) sin(q2+q3+q4) 0 -a2*cos(q2)-(a3+a4)*cos(q2+q3)-10.2*sin(q2+q3)

0 0 1 q1-d1+d2+d3+d4

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

[B0]

*0 0 0

"stl\00. Khau 0 - DE-TRU DUNG.STL" 0 1 1 0 0 0 0 0

"stl\00. Khau 0 - GOI O.STL" 0 0.5 0.75 0 0 0 0 0

"stl\00. Khau 0 - HOP DUNG ARDUINO.STL" 0.75 1 0.25 0 0 0 0 0

"stl\00. Khau 0 - KHOP NOI.STL" 0 0 1 0 0 0 0 0

"stl\00. Khau 0 - RAY TRU.STL" 0.25 0.25 1 0 0 0 0 0

"stl\00. Khau 0 - STEP MOTOR.STL" 1 0 0 0 0 0 0 0

"stl\00. Khau 0 - TRUC VIT.STL" 1 1 1 0 0 0 0 0

"stl\00. Khau 0 - VIT-DAI OC.STL" 1 1 0 0 0 0 0 0

;"stl\00. Khau 0.STL" 1 0 0 0 0 0 0 0

[B1]

*0 0 0

"stl\00. Khau 1 - DE.STL" 1 1 0 0 0 0 0 0

"stl\00. Khau 1 - GIA DO.STL" 0.5 1 0 0 0 0 0 0

"stl\00. Khau 1 - CON TRUOT.STL" 0 1 0.25 0 0 0 0
0

"stl\00. Khau 1 - VIT-DAI OC.STL" 0 1 1 0 0 0 0
0

;"stl\00. Khau 1.STL" 0 1 0 0 0 0 0 0

[B2]

*0 0 0

"stl\00. Khau 2 - MODULE.STL" 0.25 0.25 0.25 0 0 0 0 0

"stl\00. Khau 2 - STEP MOTOR.STL" 0 0 1 0 0 0 0
0

"stl\00. Khau 2 - TAM CHAN GIA DO KHAU.STL" 0.75 0 1 0 0
0 0 0

"stl\00. Khau 2 - TAY KHAU.STL" 0 0 1 0 0 0 0
0

"stl\00. Khau 2 - VIT-DAI OC.STL" 0 0 1 0 0 0 0
0

;"stl\00. Khau 2.STL" 0 1 1 0 0 0 0 0

[B3]

*0 0 0

"stl\00. Khau 3 - MODULE.STL" 0.25 0.25 0.25 0 0 0 0 0

"stl\00. Khau 3 - TAY KHAU.STL" 0 1 0 0 0 0 0
0

"stl\00. Khau 3 - VIT-DAI OC.STL" 0 0 1 0 0 0 0
0

"stl\00. Khau 3 - TAY KEP.STL" 1 1 0 0 0 0 0 0

"stl\00. Khau 3.STL" 1 1 0 0 0 0 0 0

[B4]

*0 0 0

"stl\02. Ngon phai.STL" 1 0 1 0 0 0 0 0

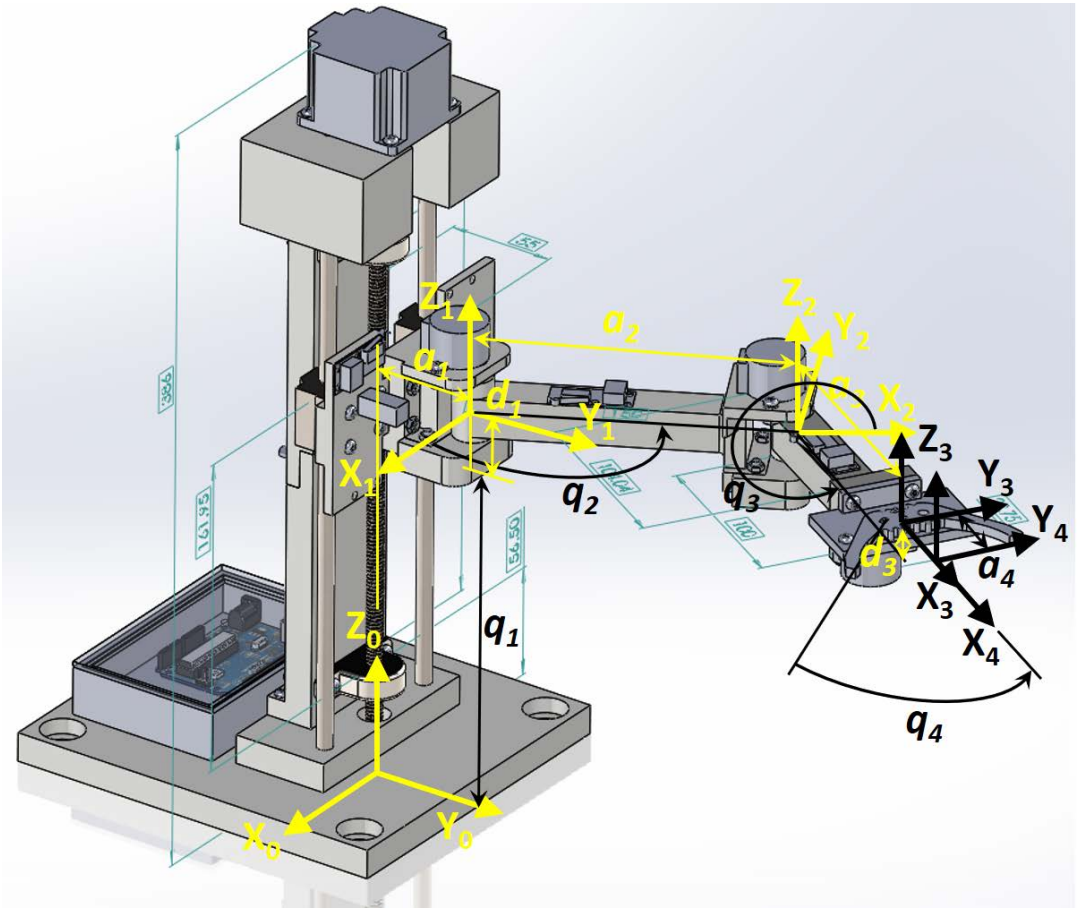
[B5]

*0 0 0

"stl\01. Ngon trai.STL" 1 0 1 0 0 0 0 0

;;

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 khâu. Từ phần dữ liệu này, ta sẽ có được các ma trận chuyển của các khâu là T0, T1, còn dữ liệu về các tệp tin chứa các chi tiết được quy định bởi các từ khóa B0, B1. Trong đó phần dữ liệu với từ khóa B0 và T0 sẽ chứa thông tin của khâu đế, khâu 0, còn phần dữ liệu với từ khóa B1 và T1 sẽ chứa thông tin của khâu 1 tức là cụm con trượt và đai ốc di chuyển lên xuống. 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 [8]. 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 3.14.



Hình 3.14: Hệ thống các tọa độ và trong tham số trong mô phỏng tay máy robot

3.3.3. Kết quả chương trình mô phỏngtay máy robot mini.

Kết quả cuối cùng của chương trình mô phỏng tay máy robot mini có giao diện như trên Hình 3.10 và 3.13. Khi chạy chương trình mô phỏng, người dùng sẽ có giao diện điều khiển mô hình theo các biến khớp (Hình 3.10) hoặc chạy mô phỏng theo quỹ đạo yêu cầu cho trước (Hình 3.13). Trong đó, quỹ đạo được cho trước là một đường xoắn ốc, từ đó các biến khớp đã được giải bằng excel và xuất kết quả ra tệp tin với định dạng sau:

X	Y	Z	q1	q2	q3	q4
310.000000	-50.000000	100.000000	128.000000	-4.099097	-1.413570	0.000000
313.139526	-49.901336	100.500000	128.500000	-4.113523	-1.385061	0.000000
316.266662	-49.605735	101.000000	129.000000	-4.127251	-1.356511	0.000000
319.369066	-49.114363	101.500000	129.500000	-4.140264	-1.328011	0.000000
322.434494	-48.429158	102.000000	130.000000	-4.152544	-1.299651	0.000000
325.450850	-47.552826	102.500000	130.500000	-4.164071	-1.271526	0.000000

328.406228	-46.488824	103.000000	131.000000	-4.174827	-1.243730	0.000000
331.288965	-45.241353	103.500000	131.500000	-4.184788	-1.216363	0.000000
334.087684	-43.815334	104.000000	132.000000	-4.193932	-1.189526	0.000000

...

Trong đó ba cột đầu thể hiện tọa độ của đầu tay kẹp, bốn cột sau thể hiện giá trị của các biến khớp tại mỗi thời điểm ứng với vị trí của tay kẹp. 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 tay máy robot mini.

Kết luận

Kết quả đạt được:

Đề tài này đã ứng dụng thành công phần mềm Solidworks để dựng mô hình 3D tay máy robot mini. Đây là tay máy robot mini phục vụ đào tạo đã được chế tạo thử tại Khoa Cơ khí – Cơ điện tử.

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 robot.

Đề tài đã xây dựng được phần mềm mô phỏng 3D hoạt động của tay máy robot theo quý đạo cho trước. Phân mềm được thiết kế để dễ dàng thay đổi và tích hợp cho các mô hình robot mới nhờ định dạng tệp tin cấu hình và mô hình 3D của robot. Đây là chức năng hữu ích giúp các sinh viên có thể thiết kế và mô phỏng hoạt động các robot mới, nhờ đó nâng cao khả năng học tập, thiết kế và sử dụng robot.

Hạn chế:

Đối với định hướng phát triển, hiện giờ chương trình mới thực hiện mô phỏng và quỹ đạo được giải trước bởi phần mềm khác chẳng hạn như file quỹ đạo được tạo ra từ file excel.

Phần mềm vẫn chưa thực hiện giải các bài toán động học ngược, tức là từ động tác từ yêu cầu chuyển động của tay máy robot mà tính ra được các vị trí của các khớp sẽ xoay bao nhiêu hoặc di chuyển bao nhiêu.

Đính hướng phát triển:

Mục tiêu đầu tiên là giải bài toán động học ngược để từ quỹ đạo ban đầu tính ra được các khớp.

Mục tiêu thứ hai, từ các chương trình chạy trên máy tính sau khi giải bài toán động học ngược, ta có thể điều khiển trực tiếp tay máy từ phần mềm.

Tài liệu tham khảo

[1] Đào Văn Hiệp, “Kỹ Thuật Robot”, chương 1, NXB khoa học và kỹ thuật, 2004

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

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

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

[5] 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.

[6] 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..

[7] 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.

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