DANH MỤC BẢNG BIỂU

DANH MỤC HÌNH ẢNH

DANH MỤC CHỮ VIẾT TẮT

3D	:	3 chiều (3 direction)
ROS	:	Hệ điều hành robot (Robot Operating System)
TCP	:	Giao thức điều khiển truyền dẫn (Transmission Control Protocol)
IP	:	Giao thức Internet (Internet Protocol)

MỤC LỤC

CHƯƠNG I. XÂY DỰNG MÔ HÌNH 3D ROBOT VÀ MÔI TRƯỜNG

Để tạo mô hình 3D cho bản sao số có nhiều giải pháp tuy nhiên có 2 phương pháp phổ biến hiện nay là quét đối tượng hoặc vẽ đối tượng (bảng 1). Tùy vào điều kiện hiện tại có thể lựa chọn phương pháp khác nhau. Trong nghiên cứu này sử dụng phương pháp vẽ đối tượng vì có bản thiết kế của robot.

Bảng 1. *So sánh giữa phương pháp quét đối tượng và vẽ đối tượng*

	Quét đối tượng	Vẽ đối tượng
Ưu điểm	Tiết kiệm thời gian và chi phí đối với hệ thống lớn và cấu tạo phức tạp Độ chính xác cao trong việc tái tạo hình dạng và chi tiết vật thể Chỉ cần 1 máy quét 3D để thu thập dữ liệu	Yêu cầu máy tính có cấu hình cao trở lên
Nhược điểm	Yêu cầu máy tính cấu hình mạnh để xử lý Thời gian xử lý và tạo mô hình 3D từ phần mềm tốn nhiều thời gian	Cần nhiều thiết bị đo đạc Thời gian và chi phí cao đối với hệ thống lớn Yêu cầu khả năng hội tụ trong việc tạo hình và kỹ năng thiết kế của người dùng để tái tạo vật thể với độ chính xác cao

Trong báo cáo này sử dụng mô hình robot được xây dựng từ các chi tiết của bản thiết kế. Do phần mềm unity và solidword dựa trên nền tảng đồ họa khác nhau, vì vậy cần đưa mô hình về chuẩn đồ họa của unity. Có nhiều giải pháp cho vấn đề này và ở đây, phần mềm blender được sử dụng làm giải pháp trung gian giúp đưa mô hình về chuẩn đồ họa của unity. Mô hình 3D được trình bày tại hình 1.

Lưu ý rằng có rất nhiều chuẩn định dạng được hỗ trợ trong nền tảng game như obj, fbx, usd, urdf, … Việc sử dụng từng loại định dạng sẽ có những ưu nhược điểm và phương pháp xử lý riêng. Trong nghiên cứu này chúng tôi sử dụng định dạng URDF. Bảng 4.2 so sánh một số định dạng trong vấn đề này.

Bảng 2. *So sánh các loại định dạng cho đồ họa cho đồ họa hoạt hình*

Loại định dạng	Ưu điểm	Nhược điểm
FBX (Filmbox)	Hỗ trợ nhiều tính năng như animation, vật liệu, độ phân giải cao, ánh sáng, …	Kích thước tệp có thể lớn đối với mô hình phức tạp
	Có khả năng tương thích với nhiều công cụ mô hình hóa 3D	
OBJ (Wavefront Object)	Hỗ trợ định dạng mesh cơ bản, vật liệu và texture	Không hỗ trợ animation và các tính năng cao cấp khác
glTF (GL Transmission Format)	Định dạng nguồn mở, hỗ trợ nhiều tính năng như geometry, vật liệu, animation, ánh sáng, …	Có thể gặp một số vấn đề về tương thích và hỗ trợ trong nền tảng game
	Hỗ trợ kích thước tệp nhỏ và tương thích với các công nghệ web khác nhau	
Collada (Digital Asset Exchange)	Định dạng đa năng, hỗ trợ nhiều tính năng như geometry,	Có thể gặp một số vấn đề về tương thích và hỗ trợ trong nền tảng game

	vật liệu, animation, cấu trúc phân cấp, …	
	Có khả năng tương thích với nhiều công cụ mô hình hóa 3D	Kích thước tệp có thể lớn đối với mô hình phức tạp
DAE (COLLADA)	Định dạng nguồn mở, hỗ trợ nhiều tính năng như geometry, vật liệu, animation, cấu trúc phân cấp,…	Một số vấn đề có thể xảy ra trong quá trình đưa vào nền tảng game
	Có thể tương thích với nhiều công cụ mô hình hóa 3D	
3DS (3D Studio)	Có thể xuất bởi các phần mềm 3D khác nhau và hỗ trợ nhiều tính năng	Có thể mất một số thông tin và tính năng phức tạp khi đưa vào nền tảng game
BLEND (Blender)	cho phép lưu trữ tất cả các thông tin của mô hình như geometry, vật liệu, animation, ánh sáng, …	Không phải là định dạng chuẩn được hỗ trợ trực tiếp trong nền tảng game
STL (Stereolithography)	Hỗ trợ geometry ba chiều đơn giản và dễ sử dụng	Không hỗ trợ vật liệu, texture, animation hay tính năng phức tạp khác
UDS (Universal Scene Description)	cho phép lưu trữ mô hình 3D phức tạp với nhiều tính năng như animation, hiệu ứng, ánh sáng và vật liệu	sự hỗ trợ và tương thích có thể khác nhau giữa các phần mềm và nền tảng khác nhau
	UDS được tối ưu hóa để đạt hiệu suất tốt trong việc xử lý và truyền tải cảnh 3D phức tạp	Mô hình UDS có thể phức tạp và đòi hỏi hiểu biết và kỹ năng đặcifch để làm việc với nó
URDF (Unified Robot Description Format)	cho phép mô tả chi tiết về cấu trúc và các thành phần của robot	URDF không hỗ trợ mô tả hình dạng hình học phức tạp
	URDF cho phép mô hình hóa nhiều robot khác nhau trong cùng một tệp	Không cung cấp hỗ trợ đặc biệt cho mô tả các động cơ hoặc hình thức điều khiển chuyển động của robot

CHƯƠNG II. XÂY DỰNG MÔI TRƯỜNG ẢO

Để xây dựng một môi trường ảo cho bản sao số có thể sử dụng các mô hình 3D về animation có sẵn hoặc sử dụng phương pháp tương tự như xây dựng mô hình 3D ở trên. Tuy nhiên với các không gian tòa nhà hay phòng thí nghiệm có thể sử dụng phương pháp vẽ trên các nền tảng như Revit, Skepchup, … sau đó sử dụng các biện pháp trung gian để đưa vào môi trường ảo. Trong nghiêm cứu này sử dụng một bản thiết kế mẫu của Revit để làm không gian hoạt động cho đối tượng 3D.

Một giao diện người dùng cũng được xây dựng giúp người dùng giám sát, tương tác và điều khiển hệ thống với nhiều chức năng điều khiển khác nhau. Giao diện điều khiển gồm 2 chế độ xem chính:

1. *Chế độ giám sát và điều khiển:*

   Gồm 2 phần là phần giao diện điều khiển và phần giám sát (hình 3 và 4). Các thông số về trạng thái robot được thể hiện lên màn hình. Ngoài ra trạng thái của mô hình 3D cũng thay đổi để người dùng dễ hình dung trạng thái của robot. Giao diện điều khiển giúp cho việc tương tác và điều khiển với robot từ xa thuận tiện và dễ dàng.

2. Chế độ 360:

Tương tác và điều khiển góc quay của camera. Điều đó giúp tăng độ trải nghiệm và tính đắm chìm của hệ thống. Ngoài ra chế độ này áp dụng cho thực tế ảo.

Ngoài ra hệ thống còn thêm chế độ mô phỏng giúp người dùng xem trước được quá trình robot sẽ làm việc. Tuy nhiên do dữ liệu và tiết kiệm chi phí nên chức năng này so với thực tế còn cách xa nhau. Cần có thời gian để huấn luyện cho mô phỏng tiệm cận với thực tế.

CHƯƠNG III. XÂY DỰNG BỘ ĐIỀU KHIỂN

Bộ điều khiển phần ảo có chức năng điều khiển mô hình 3D của robot. Trong bộ điều khiển có 2 chế độ gồm: (1) Điều khiển mô hình 3D robot bằng việc sử dụng slider; (2) Điều khiển mô hình 3D robot sử dụng mã lệnh điều khiển với các thông số tùy chỉnh. Tuy nhiên các thông số bị giới hạn trong phạm vị hoạt động của robot. Nếu vượt quá giới hạn thì sẽ robot sẽ không hoạt động và gửi cảnh báo đến người dùng. Mã lệnh được trình bày tại bảng 3.

Ngoài ra việc xử lý va chạm cũng được xây dựng dựa trên lớp phủ vật lý giúp lấy thông tin khi có sự cố va chạm giữa các đối tượng trong môi trường ảo từ đó gửi thông báo đến bộ xử lý trung tâm để xử lý.

CHƯƠNG IV. THIẾT LẬP KẾT NỐI

Hệ thống có 2 kết nối chính: 1. Kết nối giữa bộ xử lý trung tâm và bộ điều khiển; Và 2. Kết nối giữa bộ xử lý trung tâm và bản sao số.

1. KẾT NỐI GIỮA BỘ XỬ LÝ TRUNG TÂM VÀ BỘ ĐIỀU KHIỂN:

Để kết nối bộ điều khiển với bộ xử lý trung tâm, có thể sử dụng các phương pháp kết nối sau đây:

Kết nối qua cổng Serial (UART): Bộ điều khiển được trang bị một cổng Serial (UART) để giao tiếp với các thiết bị khác. Có thể sử dụng một cáp USB-to-Serial (như USB-to-Serial TTL hoặc USB-to-Serial FTDI) để kết nối cổng USB của bộ điều khiển với cổng Serial trên bộ xử lý trung tâm. Sau đó, có thể sử dụng các thư viện phần mềm như PySerial trong Python, SerialPort trong C# hoặc rosserial trong ros để giao tiếp với bộ điều khiển thông qua cổng Serial.

Kết nối qua USB: Một số bộ điều khiển hỗ trợ kết nối trực tiếp qua cổng USB. Có thể sử dụng các thư viện phần mềm như PySerial, hoặc SerialPort để thiết lập kết nối và giao tiếp với Arduino qua cổng USB.

Kết nối qua giao thức nối tiếp khác: Ngoài kết nối Serial, bộ điều khiển cũng có thể được kết nối với bộ xử lý trung tâm thông qua các giao thức nối tiếp khác như I2C, SPI hoặc Ethernet. Điều này yêu cầu bộ điều khiển và bộ xử lý trung tâm cùng hỗ trợ các giao thức này. Cần sử dụng các thư viện phần mềm và giao thức tương ứng để giao tiếp với bộ điều khiển qua giao thức nối tiếp được chọn.

Trong nghiên cứu này chúng tôi sử dụng thư viện rosserial trong ros để kết nối với bộ điều khiển thông qua công Serial (UART).

2. KẾT NỐI GIỮA BỘ XỬ LÝ TRUNG TÂM VÀ BẢN SAO SỐ:

Để kết nối giữa bộ xử lý trung tâm và bản sao số, có thể sử dụng giao thức giao tiếp ROS để truyền và nhận dữ liệu giữa hai môi trường này. Dưới đây là một phương pháp phổ biến để thực hiện kết nối này:

Cài đặt ROS: Đầu tiên, cần cài đặt và cấu hình ROS trên bộ xử lý trung tâm. ROS cung cấp các thư viện và công cụ để phát triển và chạy các ứng dụng robot. Có thể tải xuống phiên bản ROS phù hợp với hệ điều hành của bạn từ trang web chính thức của ROS.

Xác định các gói ROS cần thiết: Cần xác định các gói ROS muốn sử dụng trong ứng dụng bản sao số. Các gói này cung cấp các thông điệp, dịch vụ và hành động cần thiết để truyền dữ liệu giữa bộ xử lý trung tâm và bản sao sô. Có một số gói ROS phổ biến cho việc kết nối như "rosbridge_suite", "rosbridge_library", "ros-sharp" và “ROS-TCP”.

Xây dựng các chương trình bộ xử lý trung tâm và bản sao số tương thích: Trong ROS, cần viết các nút ROS để gửi và nhận thông tin giữa bộ xử lý trung tâm và bản sao số. Cần định nghĩa các thông điệp ROS, dịch vụ và hành động cần thiết cho ứng dụng bản sao số. Trong bản sao số, cần xây dựng các chương trình để giao tiếp với bộ xử lý trung tâm qua giao thức TCP/IP.

Phương pháp này cho phép tích hợp giữa bộ xử lý trung tâm và bản sao số trong cùng một mạng cục bộ, giúp truyền dữ liệu và nhận dữ liệu giữa bộ xử lý trung tâm và bản sao số.

3. KIỂM TRA VÀ THÔNG BÁO LỖI KẾT NỐI

Khi làm việc với kết nối giữa các hệ thống như kết nối giữa bộ xử lý trung tâm và bản sao số, có một số lỗi thường gặp có thể xảy ra, bao gồm lỗi kết nối, tràn dữ liệu, sai kiểu dữ liệu và không truyền hoặc nhận được dữ liệu. Dưới đây là một số cách để kiểm tra và thông báo về các lỗi này:

1. *Kiểm tra kết nối:*

   Kiểm tra xem cả hệ thống đang chạy và được kết nối đúng cách.

   Sử dụng các công cụ như lệnh "rostopic list" trong ROS để kiểm tra xem các topic ROS có đúng và sẵn sàng cho việc truyền dữ liệu hay không.

2. *Xử lý lỗi kết nối:*

   Sử dụng các hàm callback hoặc xử lý ngoại lệ để xử lý các lỗi kết nối.

   Đảm bảo rằng các thiết bị mạng hoặc cổng kết nối được cấu hình đúng và hoạt động ổn định.

3. *Xử lý lỗi tràn dữ liệu:*

   Kiểm tra kích thước dữ liệu được truyền qua kết nối để đảm bảo rằng không có tràn dữ liệu xảy ra.

   Sử dụng các công cụ như buffer hoặc cơ chế đồng bộ hóa để quản lý dữ liệu và tránh tràn dữ liệu.

4. *Xử lý lỗi chờ kết nối:*

   Sử dụng các hàm callback hoặc hệ thống đồng bộ hóa để kiểm tra và chờ đợi kết nối trước khi truyền dữ liệu.

   Đặt thời gian chờ tối đa cho kết nối và xử lý các trường hợp lỗi khi không kết nối được trong khoảng thời gian đó.

5. *Thông báo lỗi:*

   Sử dụng hệ thống thông báo hoặc ghi log để thông báo và ghi lại các lỗi kết nối, tràn dữ liệu và chờ kết nối.

   Hiển thị thông báo lỗi trực tiếp trên giao diện người dùng của bản sao số hoặc thông qua giao diện dòng lệnh của ROS.

Việc kiểm tra và thông báo về các lỗi này là quan trọng để quản lý và xử lý các vấn đề giao tiếp của hệ thống. Bằng cách sử dụng các cơ chế kiểm tra, xử lý lỗi và thông báo, có thể cải thiện tính ổn định và khả năng phản ứng của ứng dụng kết nối này. Hình 5.1 và 5.2 trình bày thông báo khi kết nối thành công và hiển thị thông báo lỗi khi không kết nối.

Bộ xử lý được xây dựng và phát triển trên một máy tính, vi xử lý có cài đặt ROS. Phần mềm xử lý được đóng gói thành giúp cho người dùng dễ dàng sử dụng với các câu lệnh đơn giản.

1. CÔNG DỤNG BỘ XỬ LÝ TRUNG TÂM

Kết nối và truyền thông dữ liệu 2 chiều theo thời gian thực giữa mô hình vật lý và mô hình số.

Xử lý lệnh điều khiển từ người dùng (Xem mã lệnh tại bảng 4).

Tính toán động học robot arm.

Tính toán và gửi tín hiệu điều khiển bộ điều khiển.

Xử lý lỗi phát sinh của hệ thống.

Lưu trữ dữ liệu cục bộ và sao lưu dữ liệu trên cloud.

Bảng 3. *Mã lệnh của robot*

STT	Mã lệnh	Mô tả
1	1	Điều khiển dựa trên góc quay
2	2	Điều khiển dựa trên điểm cuối tay máy

2. ƯỚC TÍNH VỊ TRÍ CỦA GÓC QUAY ĐỘNG CƠ

Việc ước tính vị trí động cơ giúp giảm sai số giữa mô hình thực và ảo ảnh hưởng bởi độ trễ của hệ thống. Dữ liệu đầu vào là vị trí góc tại thời điểm trước đó và vị trí góc tại thời điểm hiện tại, một điểm trong tương lai được tính toán. Thuật toán được trình bày như sau:

Đầu vào	:	P_{cur}, P_{pre}, v_{pre}, t_{cur}, t_{pre}, t_{fur}
Đầu ra	:	P_{fur}, P_{pre}, v_{pre}
Xử lý	:	$v = \frac{P_{cur} - P_{pre}}{t_{cur} - t_{pre}}$ $a = \frac{v_{cur} - v_{pre}}{t_{cur} - t_{pre}}$ $P_{fur} = P_{cur} + v * (t_{fur} - t_{cur}) + \frac{a*(t_{fur} - t_{cur})^2}{2}$ $P_{pre} = P_{cur}$ $v_{pre} = v$

Với: P_{cur} là vị trí góc quay hiện tại của khớp. P_{pre} là vị trí góc quay trước đó của khớp. v_{pre} là vận tốc tại thời điểm trước đó, t_{cur} là thời gian tại lúc lấy P_{cur}. t_{pre} là thời gian trước đó, t_{fur} là thời gian mà góc quay của khớp ảo được cập nhập. P_{fur} là giá trị góc quay tại thời điểm t_{fur}. v là giá trị vận tốc tức thời tại thời điểm t_{cur}. a là gia tốc tức thời tại thời điểm t_{cur}.

3. XỬ LÝ LỖI HỆ THỐNG

Khi chưa có kết nối hệ thống sẽ cố gắng kết nối với bộ điều khiển và bản sao số. Khi nào kết nối thành công thì sẽ kiểm tra các kết nối bên trong hệ thống đã đảm bảo chưa và hiển thị ra thông báo. Nếu chưa thì sẽ hiển thị thông báo lỗi kết nối.

Ngoài ra hệ thống cũng kiểm tra kết nối với bản sao số và bộ điều khiển. Nếu trong khoảng 10s không có dữ liệu nào được truyền về thì hệ thống sẽ thông báo lỗi mất kết nối và dừng hoạt động.

Trong quá trình hoạt động, nếu động cơ được lệnh di chuyển mà trong 20s không có sự thay đổi thì hệ thống sẽ thông báo lỗi và ngắt hệ thống. Điều này đảm bảo an toàn cho hệ thống khi có sự cố về động cơ xảy ra.

4. LƯU TRỮ DỮ LIỆU

Để lưu trữ dữ liệu cục bộ có thể sử dụng các phương pháp như lưu trữ trong tệp văn bản, lưu trữ trong cơ sở dữ liệu, hoặc sử dụng các cấu trúc dữ liệu trong ngôn ngữ lập trình đang sử dụng.

1. *Lưu trữ trong tệp văn bản:*

   Lưu trữ dữ liệu trong các tệp văn bản như CSV, JSON hoặc các định dạng tương tự. Bạn có thể ghi dữ liệu vào tệp và đọc dữ liệu từ tệp bằng cách sử dụng các hàm đọc/ghi tệp trong ngôn ngữ lập trình của bạn.

2. *Lưu trữ trong cơ sở dữ liệu:*

   Sử dụng các hệ quản trị cơ sở dữ liệu như MySQL, SQLite, PostgreSQL, hoặc MongoDB để lưu trữ và truy vấn dữ liệu cục bộ. Các hệ quản trị cơ sở dữ liệu này cung cấp các API và truy vấn để tương tác với dữ liệu.

3. *Sử dụng các cấu trúc dữ liệu trong ngôn ngữ lập trình:*

   Một số ngôn ngữ lập trình cung cấp các cấu trúc dữ liệu như danh sách (list), bộ (tuple), từ điển (dictionary) hoặc các cấu trúc dữ liệu tương tự. Bạn có thể lưu trữ dữ liệu trong các cấu trúc này và thao tác với chúng theo nhu cầu.

Trong nghiên cứu này sử dụng phương pháp kết hợp phương pháp 1 và 2 để lưu trữ và sao lưu dữ liệu lên cloud. Dữ liệu được lưu vào các tệp csv, sau một khoảng thời gian nhất định thì dữ liệu được sao lưu trên nền tảng cloud. Nền tảng cloud được xây dựng dựa trên MongoDB.

5. KIỂM TRA VÀ THÔNG BÁO LỖI HỆ THỐNG

Các lỗi khi vận hành cũng được chuẩn hóa thành các Error Code (EC) tại bảng 4. Điều đó giúp việc xác định và giải quyết lỗi một cách nhanh chóng và dễ dàng.

Bảng 4. *Mã lỗi của hệ thống*

STT	EC	Loại lỗi
1	000	Không có kết nối với mạch điều khiển
2	001	Không có kết nối với bản sao số
3	002	Không tìm thấy giá trị quỹ đạo
4	003	Lỗi động cơ 1 không hoạt động
5	004	Lỗi động cơ 2 không hoạt động
6	005	Lỗi 2 động cơ không hoạt động
7	006	Lỗi truyền dữ liệu hệ thống

CHƯƠNG VI. PHƯƠNG PHÁP ĐÁNH GIÁ

1. ĐỘ TRỄ HỆ THỐNG

Dữ liệu được thu thập và phân tích về mặt thời gian và giá trị tại thời điểm dữ liệu được nhận. Dựa trên dữ liệu đó sẽ được xử lý và chuẩn hóa sau đó vẽ thành biểu đồ để đánh giá sai số.

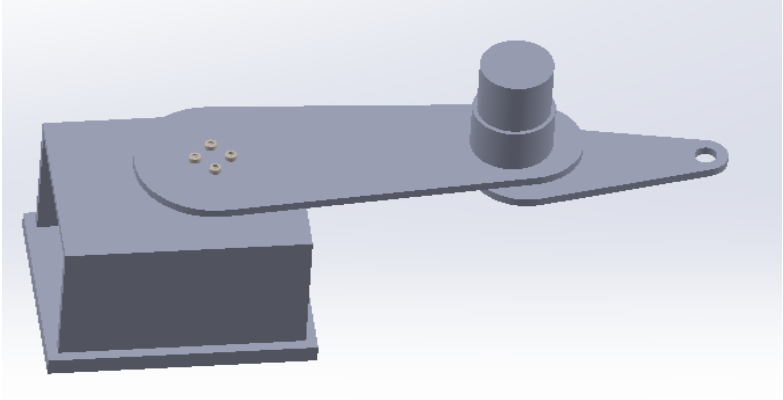
2. SAI SỐ HỆ THỐNG

Dữ liệu được thu thập và phân tích về mặt thời gian. Dựa trên dữ liệu thời gian thu thập được sẽ vẽ thành các biểu đồ hỗ trợ việc phân tích. Thuật toán tính toán độ trễ hệ thống được trình bày như sau:

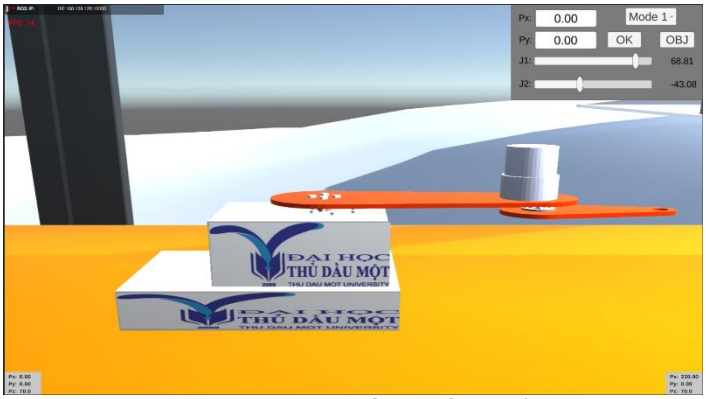
Đầu vào:	t_{pre}, t_{cur}, t=0
Đầu ra	Δ_{avd}, Δ_t
Vòng lặp:	$\Delta_t = t_{cur} - t_{pre}$ $i = i + 1$ $t = t + \Delta_t$ $\Delta_{avd} = \frac{t}{i}$

Trong đó: t_{pre} là thời gian tại thời điểm trước đó; t_{cur} là thời gian tại thời điểm xét; Δ_t là độ trễ; Δ_{adv} là độ trễ trung bình; i là số lần dữ liệu được truyền và nhận; t là tổng độ trễ tương ứng với số lần i.

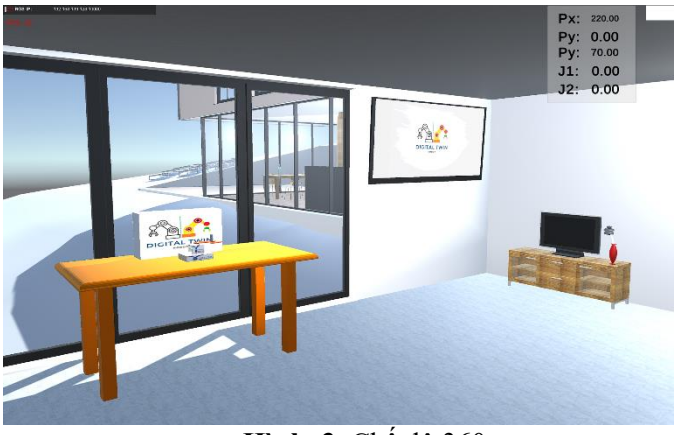
CHƯƠNG VII. KẾT QUẢ



Hình 1. Mô hình 3D robot



Hình 2. Chế độ điều khiển



Hình 3. Chế độ 360

CHƯƠNG VIII. ĐÁNH GIÁ ĐỘNG HỌC ROBOT

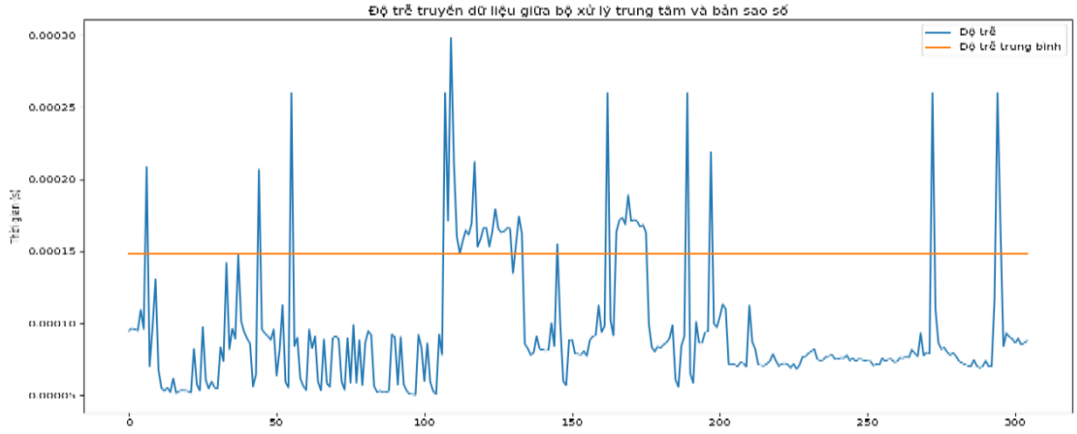
Bảng 7.1 cho thấy sự so sánh giữa việc tính toán giữa bộ xử lý trung tâm và công cụ hỗ trợ là gần giống nhau. Điều đó chứng tỏ việc tính toán động học robot của bộ xử lý trung tâm được chấp nhận.

Bảng 5. So sánh động học giữa bộ xử lý trung tâm và công cụ

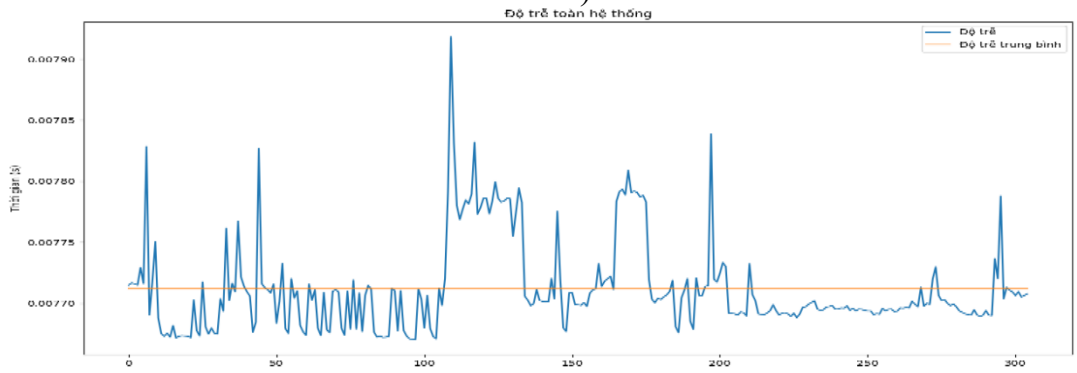
STT	Bộ xử lý trung tâm		Công cụ		Vị trí điểm cuối tay máy
	Bộ giá trị góc 1	Bộ giá trị góc 2	Bộ giá trị góc 1	Bộ giá trị góc 2	
1	[50, 50]	[85.5, -50]	[50, 50]	[85.5, -50]	[76.1, 186.03 ,70]
2	[49.5, 50]	[85, -50]	[49.49, 50]	[85.01, -50]	[77.73, 185.35, 70]
3	[9.99, 10]	[17.27, -10]	[10, 10]	[17.26, -9.99]	[213.05, 51.67, 70]
4	[-76, 89]	[-17, -89]	[-75.99, -89]	[-16.99, -89]	[111.82, -117.85, 70]
5	[-46.49, 88]	[12, -88]	[-47, 87.99]	[11.99, -88]	[156.29, -48.52, 70]
6	[58.28, 12]	[67, -12]	[58.28, 12.01]	[67, -12.01]	[100.59, 194.4, 70]

1. ĐÁNH GIÁ ĐỘ TRỄ HỆ THỐNG

Do độ trễ xử lý nên đã được điều chỉnh khoảng 50 Hz với độ trễ khoảng 20ms để tránh tràn dữ liệu. Độ trễ để xử lý khi nhận được tín hiệu là khoảng 5 ms. Độ trễ khi bộ xử lý gửi dữ liệu đến khi bản sao số nhận được dữ liệu là khoảng 0.15 ms. Thời gian bản sao số xử lý khi nhận được tín hiệu từ bộ xử lý trung tâm là 2.57 ms. Thời gian độ trễ để một dữ liệu từ thu thập đến lúc hoàn thành xử lý trên toàn bộ hệ thống khoảng 7.72 ms. Độ trễ hệ thống được thể hiện tại hình 4.



a)

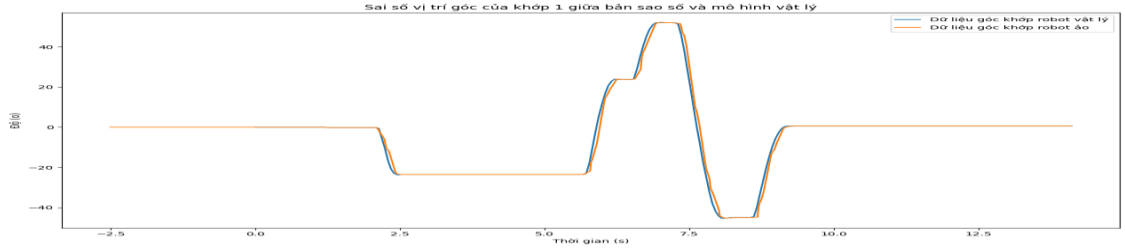


b)

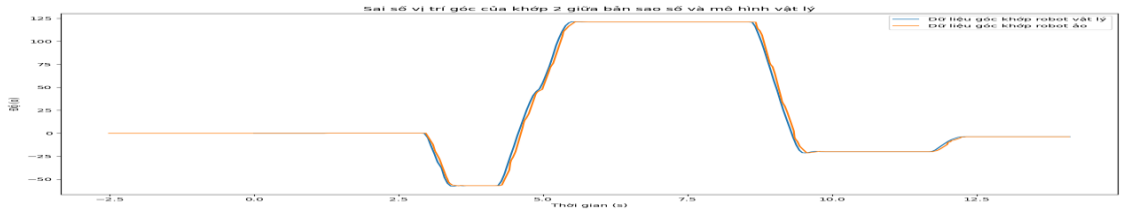
Hình 4. Độ trễ hệ thống. a) Độ trễ xử lý dữ liệu của bộ xử lý trung tâm trong khoảng 0.01-1.9 ms với độ trễ trung bình khoảng 0.15 ms; b) Đỗ trễ trên toàn hệ thống với độ trễ trung bình 7.72 ms.

2. ĐÁNH GIÁ SAI SỐ GIỮA MÔ HÌNH VẬT LÝ VÀ MÔ HÌNH ẢO

Sai số giữa mô hình thực vào ảo được thể hiện tại hình 6 và 7 khi lấy robot thực làm chuẩn. Có thể thấy rằng sai số giữa mô hình thực và ảo là khá nhỏ và gần như là trùng nhau khi sử dụng robot thực điều khiển robot ảo. Ngược lại thì lấy robot ảo làm chuẩn thì sai số giữa mô hình ảo và mô hình thực tăng lên thể hiện tại hình 8. Vấn đề này là do việc tính toán và điều khiển con ảo khác so với chạy thực tế ở con thực. Điều này có thể khắc phục bằng việc huấn luyện robot ảo từ dữ liệu robot thực.

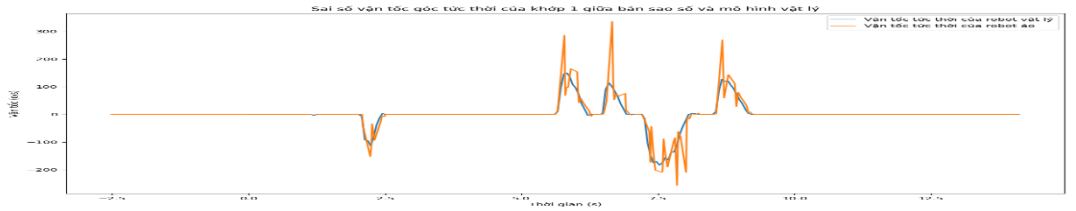


a)

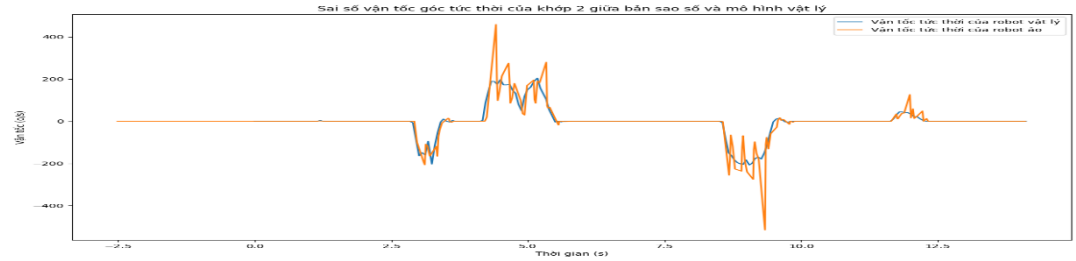


b)

Hình 5. Sai số vị trí góc của khớp giữa mô hình vật lý và mô hình bản sao số. a) Sai số vị trí góc tại khớp 1. b) Sai số vị trí góc tại khớp 2.

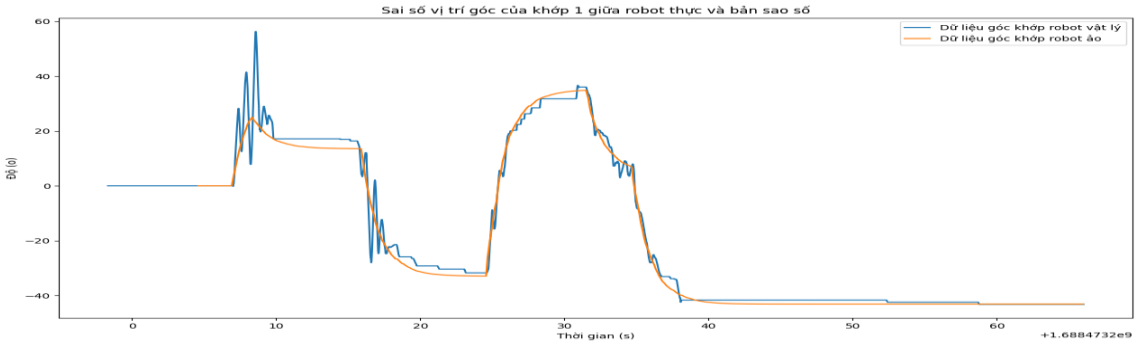


a)



b)

Hình 6. Sai số vận tốc góc tức thời giữa tính toán mô phỏng và thực tế. a) Sai số vận góc tốc tức thời tại khớp 1. b) Sai số vận tốc góc tức thời tại khớp 2.



a)

b)

Hình 7. Sai số vị trí góc của khớp giữa mô hình vật lý và mô hình bản sao số. a) Sai số vị trí góc tại khớp 1. b) Sai số vị trí góc tại khớp 2.

CHƯƠNG IX. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. KẾT LUẬN

Nghiên cứu xây dựng được mô hình bản sao số cho robot với các mục tiêu đề ra với độ trễ khoảng 7.72 ms và sai số khá thấp. Tuy nhiên hệ thống chỉ mới thu thập dữ liệu từ một loại cảm biến, chưa đề cập đến các vấn đề dự báo, xử lý va chạm hay huấn luyện robot ảo. Ngoài ra việc sử dụng tính toán để

tìm ví trí điểm cuối tay máy hay sự phụ thuộc vào encoder cũng là chưa đủ để đánh giá khách quan sai số giữa robot thực và ảo. Hệ thống cần được phát triển và cải tiến nhiều hơn nữa. Cần lưu ý khi chọn tần số lấy mẫu cảm biến cho bộ điều khiển trung tâm phải lớn hơn độ trễ của hệ thống để tránh tràn dữ liệu.

2. HƯỚNG PHÁT TRIỂN

Cải thiện các mô hình vật lý cũng như mô hình bản sao số để làm cho bản sao số giống mô hình thực hơn và cung cấp thông tin dự đoán bằng cách tích hợp các tính toán đa vật lý, phương pháp điều khiển hiện đại và áp dụng các kỹ thuật Máy học với nhiều dữ liệu thời gian thực hơn.

Ứng dụng VR/AR cho tương tác của hệ thống.