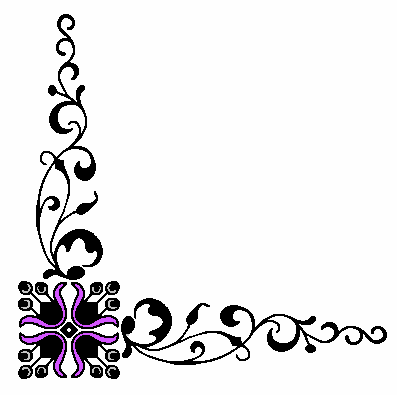
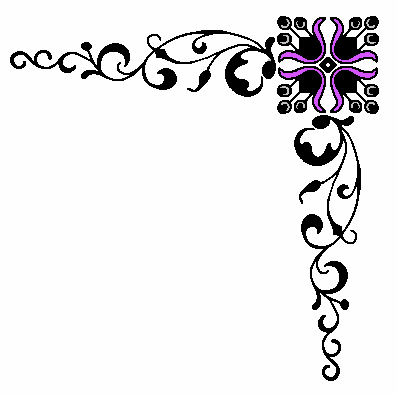
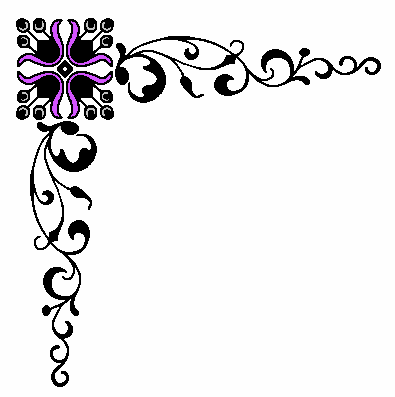
**TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH**



**KHOA ĐIỆN – ĐIỆN TỬ**

**BỘ MÔN ĐIỀU KHIỂN TỰ ĐỘNG**



**ĐỀ CƯƠNG LUẬN VĂN  
ĐIỀU KHIỂN VÀ ỨNG DỤNG ROBOT DENTAL**

**GVHD: Nguyễn Đức Hoàng**

**Họ và Tên MSSV**

**Lê Thanh Dương 1811809**

**Phan Mạnh Đức 1811980**

**TP.Hồ Chí Minh ngày 03 tháng 1 năm 2022**

1. **Giới thiệu tổng quan về Robot và Robot Delta**

Theo thời gian, con người đã tìm kiếm các sản phẩm thay thế có thể bắt chước hành vi của họ trong tương tác với môi trường. Từ robot được giới thiệu vào năm 1920 bởi Karel Capek, một nhà viết kịch người Séc, trong vở kịch “Rossum’s Universal Robots” của ông. Thuật ngữ robot có nguồn gốc từ từ robota trong tiếng Slav có nghĩa là lao động điều hành [1].



Cấu trúc của robot bao gồm các cơ quan cứng (liên kết) được kết nối với nhau bằng các khớp (khớp) cho phép khả năng di chuyển của robot. Cấu trúc cơ học của robot hoặc dây chuyền động học, có thể được thể hiện bằng cổ tay tạo sự khéo léo, một cánh tay đảm bảo tính di động và một bộ phận hiệu ứng cuối để thực hiện nhiệm vụ được yêu cầu [1].

Các robot được sử dụng trong các ứng dụng công nghiệp có thể là dây chuyền nối tiếp hoặc dây chuyền mở, và bộ điều khiển dây chuyền động học khép kín. Chuỗi động học mở khi chỉ một chuỗi liên kết nối hai đầu của chuỗi, trong khi chuỗi động học đóng khi một chuỗi liên kết tạo thành vòng.

Các bộ thao tác nối tiếp bao phủ một không gian làm việc lớn khi nó được mở rộng hoàn toàn và các khớp nối của chúng có thể được điều khiển độc lập. Nhưng nhược điểm lớn của nó là khả năng chịu tải thấp so với kết cấu và động cơ lớn của từng mắt xích, gây ra hiện tượng tăng giảm tốc chậm.

Robot delta được coi là một bộ điều khiển dây chuyền động học khép kín và nó có rất nhiều lợi thế khi so sánh nó với các bộ thao tác nối tiếp. Ví dụ, độ cứng cao hơn và khối lượng di động nhỏ, giúp tăng tốc nhanh hơn mà không làm mất độ chính xác. Ngoài ra, robot delta có thể di chuyển trọng tải nặng hơn so với khối lượng cơ thể của nó. Hạn chế chính của robot delta là phạm vi chuyển động hạn chế của chúng so với một bộ điều khiển nối tiếp.



Năm 1988, Giáo sư Reymond Clavel, tại Viện Công nghệ Liên bang Thụy Sĩ ở Lausanne, đã phát triển ý tưởng sử dụng các robot song song với ba bậc tự do tịnh tiến và một bậc quay, tạo ra robot delta [2].

Ý tưởng sử dụng robot delta là việc sử dụng các hình bình hành. Hình bình hành cho phép một liên kết đầu ra vẫn ở một vị trí cố định đối với liên kết đầu vào. Chức năng của ba hình bình hành là hạn chế tác động cuối, chỉ còn lại ba bậc tự do tịnh tiến.

Tạo thành một cấu trúc vòng khép kín, mỗi một trong số các cánh tay được kết nối với bệ cơ sở thông qua các khớp quay [3], khớp này chỉ cho phép quay một trong các phần thân so với phần khác, do đó, khớp này chỉ có một bậc tự do [4 ]. Vị trí của bộ tạo hiệu ứng cuối thay đổi tương ứng với vị trí góc của động cơ trên bệ cơ sở [5]. Cuối cùng, một chân thứ tư được sử dụng để truyền chuyển động quay (Hình 2) từ chân đế đến bộ tạo hiệu ứng cuối gắn trên bệ di động [6].



Ưu điểm cốt lõi của robot delta là tốc độ của nó, sự khác biệt giữa các cánh tay robot điển hình và robot delta là cánh tay robot điển hình phải di chuyển trọng tải và tất cả các servo trong mỗi khớp và robot delta chỉ di chuyển khung của nó, điều này thường được làm bằng vật liệu nhẹ. Do ưu điểm này, robot delta đạt được gia tốc lên đến 12g trong các ứng dụng công nghiệp [2], khiến nó trở thành ứng cử viên hoàn hảo cho các hoạt động nhặt và đặt các vật thể nhẹ.

1. **Động học của robot Delta**
   1. **Mô hình hóa robot delta**

Như đã đề cập trong phần trên, vị trí của bộ tạo hiệu ứng cuối thay đổi tương ứng với vị trí góc của động cơ, do đó, bằng cách sửa đổi các vị trí này, có thể kiểm soát vị trí của bộ tạo hiệu ứng cuối. Bộ tạo hiệu ứng được kết nối với các cánh tay trên bằng các khớp phổ quát [9].

Chuyển động học được sử dụng để mô tả chuyển động của robot delta thông qua các phương trình toán học, điều này với mục đích biết cách bộ tạo hiệu ứng cuối được dẫn đến vị trí mong muốn, dẫn đến hai khả năng, đó là: chuyển động tịnh tiến hoặc chuyển động thẳng và chuyển động học nghịch đảo.

* 1. **Chuyển động học trực tiếp**

Động học trực tiếp cho phép vị trí và hướng của tác động cuối được biểu thị dưới dạng hàm của các biến khớp của kết cấu cơ học đối với hệ quy chiếu [1].

Vị trí của bộ tạo hiệu ứng cuối có thể được tính toán bằng vị trí góc của động cơ [9], điều này có thể được thực hiện bằng cách sử dụng bộ mã hóa động cơ. Khi có các giá trị cụ thể của bộ mã hóa, cần xác định vị trí của bộ tạo hiệu ứng cuối. Một cách khác để tìm vị trí của bộ tạo hiệu ứng cuối trong robot delta là tìm giao điểm của ba phương trình hình cầu, trong đó điểm gốc của hình cầu được xác định bởi kích thước của các cánh tay trên và vị trí góc của động cơ [5].

Các chuyển động học trực tiếp có thể cung cấp tâm tấm di chuyển cho ba góc nhất định. Tâm của tấm được lập thành một bộ ba phương trình bậc hai ghép nối phải được giải đồng thời [3].

* 1. **Chuyển động học nghịch đảo**

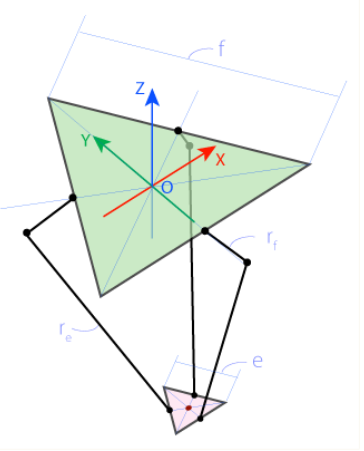
Với vị trí mong muốn của bộ tạo hiệu ứng cuối (các điểm trong tọa độ Descartes (𝑋, 𝑌, 𝑍)), cần phải xác định vị trí góc hoặc tịnh tiến của các liên kết của robot để đặt chúng vào đúng vị trí để định vị bộ tạo hiệu ứng cuối. Quá trình này được gọi là chuyển động học nghịch đảo. Giải pháp cho bài toán động học nghịch đảo của robot delta bao gồm việc chuyển đổi vị trí của tác động cuối thành các vị trí góc của ba cánh tay.

Bài toán động học nghịch đảo rất phức tạp vì các phương trình cần giải nói chung là phi tuyến tính và do đó có thể tồn tại nhiều nghiệm, không phải lúc nào cũng có thể tìm ra lời giải hoặc không có lời giải nào được phép theo cấu trúc bộ thao tác.

1. **Bài toán Động học nghịch của Robot Delta**

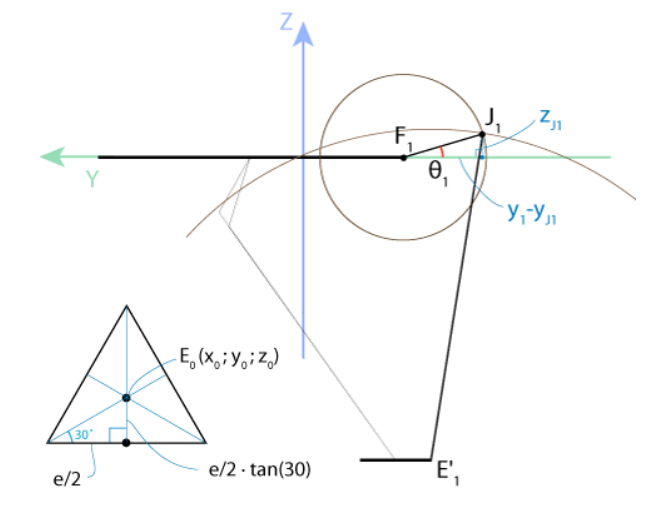
Trước tiên, chúng ta xác định một số thông số chính về hình học của robot. Ký hiệu cạnh của tam giác cố định là f, cạnh của tam giác hiệu ứng cuối là e, chiều dài của khớp trên là rf và chiều dài của khớp hình bình hành là re. Đây là các thông số vật lý được xác định bởi thiết kế của rô bốt của bạn. Hệ quy chiếu sẽ được chọn với gốc là tâm đối xứng của tam giác cố định, như hình dưới đây, do đó, tọa độ z của hiệu ứng cuối sẽ luôn âm.

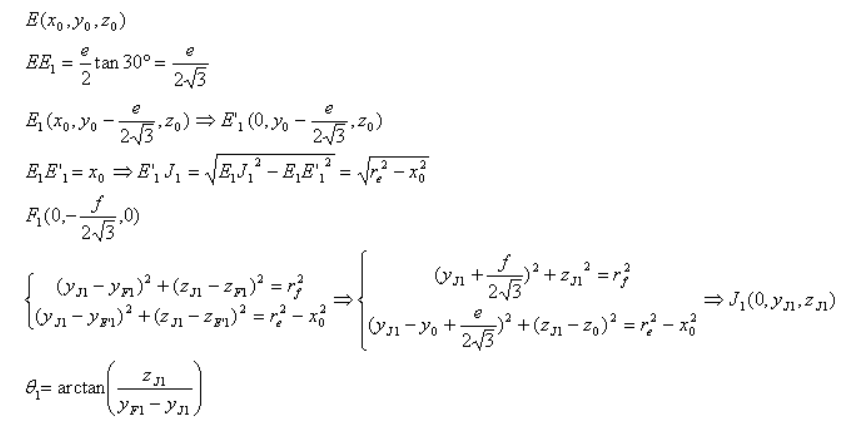
Do khớp thiết kế của rô bốt nên F1J1 (xem hình bên dưới) chỉ có thể quay trong mặt phẳng YZ, tạo thành vòng tròn có tâm ở điểm F1 và bán kính rf. Trái ngược với F1, J1 và E1 được gọi là khớp vạn năng, có nghĩa là E1 J1 có thể quay tự do tương đối so với E1, tạo thành hình cầu có tâm ở điểm E1 và bán kính lại.



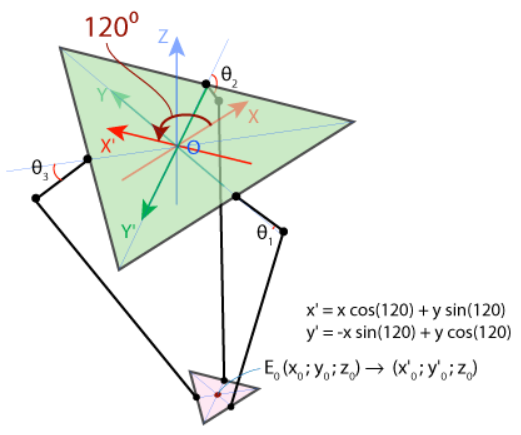


Giao của mặt cầu này và mặt phẳng YZ là một đường tròn có tâm là điểm E’1 và bán kính E’1J1, trong đó E’1 là hình chiếu của điểm E1 trên mặt phẳng YZ. Bây giờ có thể tìm thấy điểm J1 là giao điểm của các đường tròn có bán kính đã biết với các tâm ở E’1 và F1 (chúng ta chỉ nên chọn một giao điểm có tọa độ Y nhỏ hơn). Và nếu chúng ta biết J1, chúng ta có thể tính được góc theta1. Dưới đây, chúng ta có thể tìm thấy các phương trình tương ứng và chế độ xem mặt phẳng YZ :





Sự đơn giản đại số như vậy dựa trên sự lựa chọn tốt của hệ quy chiếu: khớp F1J1 chỉ chuyển động trong mặt phẳng YZ, vì vậy chúng ta hoàn toàn bỏ qua tọa độ X. Để tận dụng lợi thế này cho các góc còn lại theta2 và theta3, chúng ta nên sử dụng phép đối xứng của robot delta. Đầu tiên, hãy xoay hệ tọa độ trong mặt phẳng XY quanh trục Z qua góc 120 độ ngược chiều kim đồng hồ, như hình minh họa bên phải.



Chúng tôi đã có một hệ quy chiếu mới X’Y’Z’ và hệ quy chiếu này, chúng ta có thể tìm thấy góc theta2 bằng cách sử dụng cùng một thuật toán mà chúng tôi đã sử dụng để tìm theta1. Thay đổi duy nhất là chúng ta cần xác định tọa độ mới x’0 và y’0 cho điểm E0, điều này có thể dễ dàng thực hiện bằng cách sử dụng ma trận quay tương ứng. Để tìm góc theta3 ta phải quay hệ quy chiếu theo chiều kim đồng hồ.

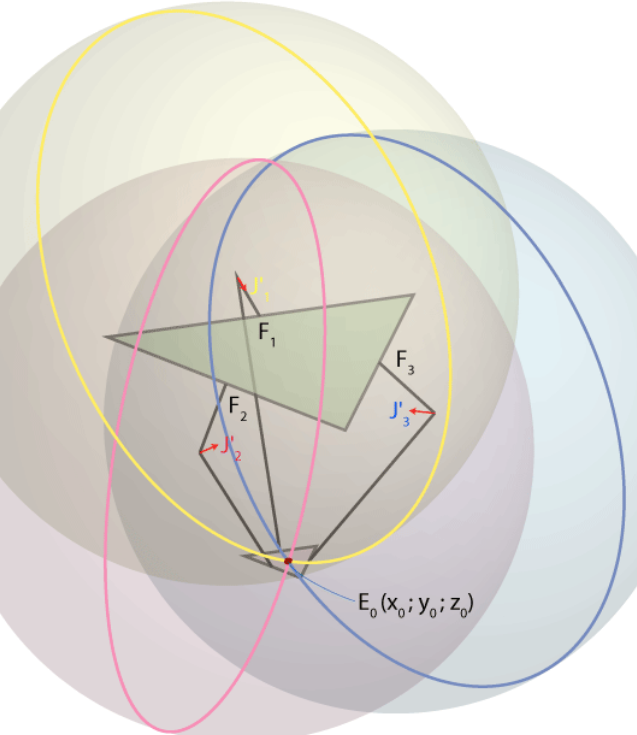
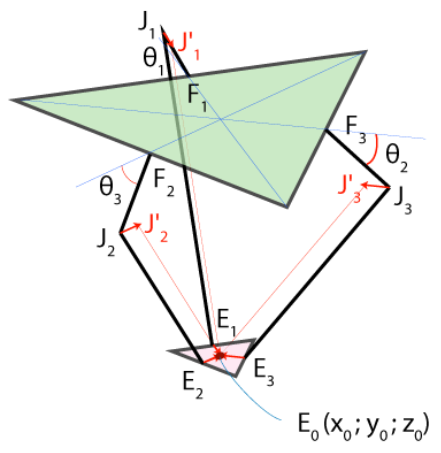
1. **Bài toán Động học thuận của Robot Delta**

Bây giờ ba góc chung theta1, theta2 và theta3 đã cho, và chúng ta cần tìm tọa độ (x0, y0, z0) của điểm hiệu ứng cuối E0. Như chúng ta đã biết về góc theta, chúng ta có thể dễ dàng tìm được tọa độ của các điểm J1, J2 và J3(xem hình bên dưới). Mối ghép J1E1, J2 E2 và J3E3 có thể quay tự do lần lượt quanh các điểm J1, J2 và J3 tạo thành ba mặt cầu có bán kính lại.

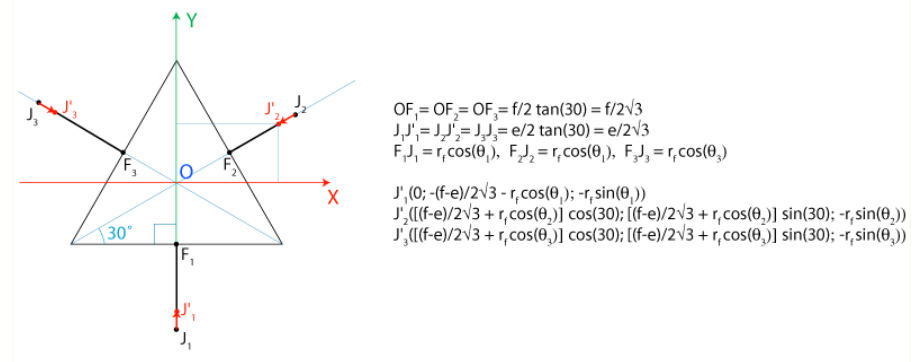
Bây giờ chúng ta hãy làm như sau: di chuyển tâm của các hình cầu từ các điểm J1, J2 và J3 đến các điểm J’1, J’2 và J’3 bằng cách sử dụng các vectơ chuyển tiếp E1E0, E2E0 và E3E0 tương ứng. Sau quá trình chuyển đổi này, tất cả ba hình cầu sẽ giao nhau tại một điểm: E0 (hình phía dưới)

Vì vậy, để tìm tọa độ (x0, y0, z0) của điểm E0, chúng ta cần giải bộ ba phương trình như

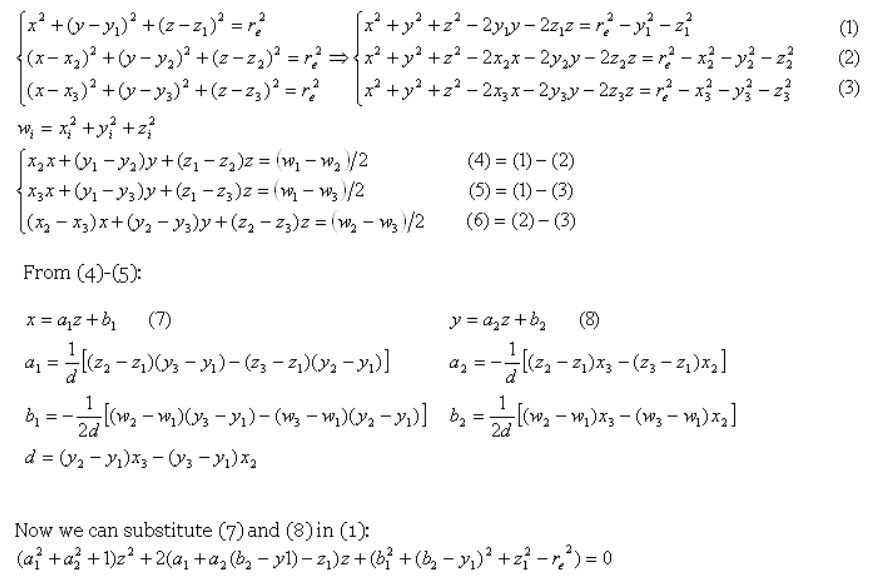
(x-xj)^2 + (y-yj)^2 + (z-zj)^2 = re^2 , trong đó tọa độ của các tâm hình cầu (xj, yj, zj) và bán kính lại được biết đến.



Đầu tiên, chúng ta sẽ tìm tọa độ của các điểm J’1, J’2, J’3 :



Trong các phương trình sau, chúng ta sẽ chỉ định tọa độ của các điểm J1, J2, J3 là (x1, y1, z1), (x2, y2, z2) và (x3, y3, z3) và x0 = 0. Dưới đây là phương trình của ba quả cầu:



Cuối cùng, chúng ta cần giải phương trình bậc hai này và tìm z0 (chúng ta nên chọn nghiệm nguyên nhỏ nhất của phương trình âm), sau đó tính x0 và y0 từ phương trình (7) và (8).

1. **Software Robot Operating System (ROS)**

Robot Operation System (ROS) là một framework để viết ứng dụng robot. Nó là một tập hợp các công cụ, thư viện và quy ước nhằm mục đích đơn giản hóa nhiệm vụ tạo ra các hành vi phức tạp và mạnh mẽ của robot trên nhiều nền tảng robot khác nhau.

Bởi vì việc tạo ra phần mềm robot thực sự hữu ích, có mục đích chung là rất khó. Từ các khía cạnh của robot, nhiều vấn đề có vẻ nhỏ nhặt đối với con người thực sự có thể bao gồm các biến thể phức tạp giữa các task và môi trường.

ROS là một khuôn khổ để phát triển phần mềm người máy. Phần mềm được cấu trúc theo mô hình mô-đun, nhỏ Các gói này giao tiếp với nhau thông qua các message. Mô hình này là khuyến khích tái sử dụng mã và hợp tác toàn cầu trên các môi trường cụ thể.



* 1. **Lịch sử**

ROS là một dự án tuyệt vời có nhiều tổ tiên và những người đóng góp. Rất nhiều Những người trong cộng đồng nghiên cứu robot cảm thấy cần phải có một khuôn khổ hợp tác mở. Một số dự án tại Đại học Stanford vào giữa những năm 2000 liên quan đến trí tuệ nhân tạo tích hợp và nhúng, chẳng hạn như chương trình Robot AI của Stanford (STAIR) và chương trình Robot cá nhân (PR), đã tạo ra nguyên mẫu nội bộ của các loại hệ thống phần mềm năng động và linh hoạt như nó vốn có ROS. Nó dựa trên Switchyard, là một phần của dự án STAIR và được viết bởi Morgan Quigley tại Stanford. Năm 2007, Willow Garage, Inc., một vườn ươm người máy gần đó, được cung cấp các nguồn lực quan trọng để mở rộng các khái niệm này hơn nữa và tạo các triển khai đã được kiểm tra tốt. Từ năm 2013 đến nay, ROS được duy trì vĩnh viễn bởi Quỹ người máy nguồn mở của Google và từ năm 2017 đổi tên thành Open Robotics.



Hầu hết các công ty nghiên cứu robot và phòng thí nghiệm hiện nay chuyển phần mềm của bạn sang ROS. Xu hướng này cũng có thể nhìn thấy trong các robot công nghiệp, nơi các công ty đang dần chuyển từ các ứng dụng độc quyền sang ROS. Phong trào được gọi là ROS Industrial đã gia tăng trong những năm gần đây và Mục tiêu chính là mở rộng các khả năng nâng cao của ROS cho tự động hóa và robot công nghiệp. Dự án này bắt đầu như một nỗ lực hợp tác của Yaskawa Motoman Robotics. Vào tháng 1 năm 2012, Shaun Edwards của SwRI thành lập một phần mềm kho lưu trữ, được lưu trữ trên Github, nơi cộng đồng người máy có thể tìm thấy các giao diện cho người thao tác công nghiệp, máy gắp, cảm biến và mạng thiết bị

Kết thúc với lịch sử ngắn gọn của ROS, có thể nói rằng nó hiện đang được sử dụng trên toàn thế giới trong các tổ chức học thuật, công nghiệp và nghiên cứu. Các nhà phát triển đã đóng góp hàng nghìn gói, bao gồm các giải pháp từ một số chuyên gia hàng đầu thế giới trong các lĩnh vực cụ thể. Các công ty mới Robot cung cấp giao diện ROS cho các sản phẩm của họ và các công ty robot công nghiệp đã thành lập cũng đang giới thiệu ROS. Với sự chấp nhận rộng rãi ROS là phương pháp tiếp cận yếu tố tiêu chuẩn để lập trình robot, có một hy vọng mới về việc tăng tốc khả năng của robot theo cấp số nhân.

* 1. **Vấn đề chính**

-ROS được tạo ra với mục đích cho phép các nhà nghiên cứu phát triển nhanh chóng các hệ thống robot mới mà không cần phải phát minh lại mọi thứ, bằng cách sử dụng sử dụng các công cụ và giao diện tiêu chuẩn. Để thực hiện các quy trình của nghiên cứu và phát triển phần mềm rô bốt, 3 vấn đề sau được phát hiện cần giải quyết:

* + Lập trình tuần tự không phù hợp với thế giới không đồng bộ Robot.
  + Hệ thống robot phải xử lý độ phức tạp đáng kể
  + Việc trừu tượng hóa các chi tiết của một phần cứng cụ thể của robot rất khó

- Giải pháp cho vấn đề lập trình trình tự dựa trên "callback", chúng là một hàm chạy bất cứ khi nào có dữ liệu để xử lý. Chúng không đồng bộ, vì lệnh gọi lại có thể xảy ra bất kỳ lúc nào. Về vấn đề của phần mềm đa chức năng phức tạp và sự thu hút của các chi tiết phần cứng cụ thể, các quy trình được tách thành các "nút" giao tiếp với thông qua giao diện nhắn tin. Ví dụ, các quy trình của các phòng đã được tách biệt maras, đo oedometry, máy quét laser và tạo bản đồ và những thứ này tương tác với nhau thông qua một giao diện truyền đạt các quá trình này thông qua "chủ đề". Những khái niệm cuối cùng này là giải thích chi tiết trong các phần sau

* 1. **Giới thiệu cụ thể về ROS**

ROS là tên viết tắt của Robot Operating System và là một nền tảng phát triển ứng dụng trong lĩnh vực người máy cung cấp các kiểu lập trình (cụ thể là dựa trên các node được ghép và phân phối tự do); định nghĩa và mô hình giao diện để liên lạc giữa các node; định nghĩa giao diện để nhúng thư viện và gói; một bộ sưu tập các công cụ để trực quan hóa, gỡ lỗi, ghi dữ liệu và chẩn đoán hệ thống; một kho mã nguồn dùng chung; và kết nối với nhiều thư viện nguồn mở hữu ích và độc lập.

Mục tiêu chính của ROS là hỗ trợ tái sử dụng mã trong nghiên cứu và sự phát triển của người máy. ROS là một khuôn khổ phân tán của các quy trình (còn được gọi là dưới dạng các nút) cho phép các tệp thực thi được thiết kế và gắn đế riêng biệt tự do trong thời gian chạy. Các quy trình này có thể được nhóm lại thành các gói, chúng có thể dễ dàng chia sẻ và phân phối. ROS cũng tương thích với một hệ thống kho lưu trữ mã liên kết cũng cho phép chia sẻ chia sẻ. Thiết kế này, từ cấp hệ thống tệp đến cấp cộng đồng, cho phép Quyết định độc lập về phát triển và thực hiện, nhưng mọi người đều có thể được kết hợp với các công cụ cơ sở hạ tầng ROS

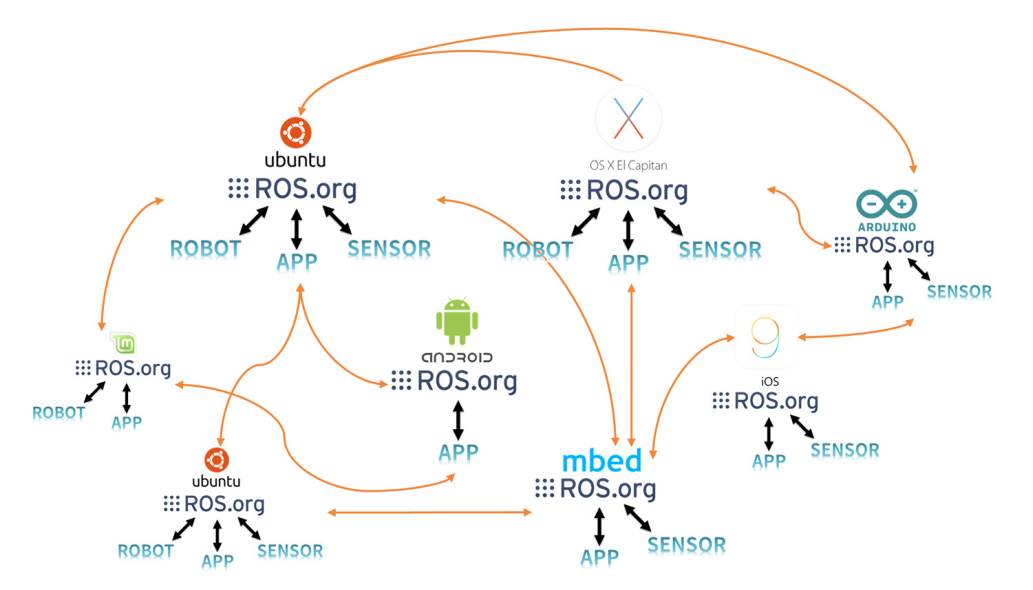
ROS không phải là một hệ điều hành thông thường như Windows, Linux và Android trong ý thức truyền thống về quản lý và lập lịch trình của các quá trình; đúng hơn, nó là một hệ thống meta hệ điều hành chạy trên hệ điều hành. Sự khác biệt giữa OS và ROS là được thể hiện trong hình bên dưới. ROS là một phần mềm trung gian, cung cấp các dịch vụ chờ đợi một hệ điều hành, bao gồm trừu tượng hóa phần cứng, điều khiển thiết bị cấp thấp, triển khai các chức năng thường được sử dụng, truyền thông báo giữa các quy trình và quản lý gói. Nó cũng cung cấp các công cụ và thư viện để lấy, biên dịch, viết và chạy mã trên nhiều máy tính.



*Sự khác biệt giữa hệ điều hành thông thường và ROS*

Các đặc điểm chính của ROS có thể được tóm tắt như sau:

* + ROS cung cấp cơ sở hạ tầng xuất bản và nhắn tin đăng ký được thiết kế để hỗ trợ xây dựng hệ thống nhanh chóng và dễ dàng phân phối máy tính
  + ROS cung cấp một bộ công cụ toàn diện để định cấu hình, khởi động, xem xét nội dung, gỡ lỗi, trực quan hóa, ghi nhật ký, kiểm tra và dừng các hệ thống phân phối máy tính
  + ROS cung cấp một bộ sưu tập thư viện phong phú triển khai các chức năng hữu ích của robot, tập trung vào tính di động, thao tác và nhận thức
  + ROS được hỗ trợ và nâng cao bởi một cộng đồng lớn, với một tập trung vào tích hợp và tài liệu. https://www.ros.org/ là một điểm tổng hợp để tìm kiếm và học hỏi về hàng nghìn gói ROS được có sẵn từ các nhà phát triển trên khắp thế giới.



*ROS và khả năng tương thích nền tảng khác*

Hình trên cho thấy rằng giao tiếp dữ liệu ROS không chỉ hỗ trợ với một hệ điều hành, nhưng cũng với nhiều hệ điều hành, phần cứng và các chương trình này rất phù hợp cho sự phát triển của Robot khi chúng được kết hợp với nhau phần cứng khác nhau

* 1. **Mục tiêu của ROS**

Tất cả các khuôn khổ phần mềm đều áp đặt triết lý phát triển của họ lên những người đóng góp trực tiếp hoặc gián tiếp, thông qua các thành ngữ và thông lệ chung của họ. Nói chung, ROS tuân theo triết lý phát triển phần mềm Unix ở một số khía cạnh mã khóa. Điều này có xu hướng làm cho ROS cảm thấy “tự nhiên” đối với các nhà phát triển đến từ Unix, nhưng hơi “khó hiểu” lúc đầu đối với những người chủ yếu sử dụng môi trường phát triển đồ họa trên Windows hoặc Mac OS X

Hệ thống ROS bao gồm nhiều chương trình máy tính nhỏ kết nối với nhau và trao đổi tin nhắn liên tục. Những thông điệp này đi travel trực tiếp từ chương trình này sang chương trình khác; không có dịch vụ định tuyến trung tâm.



* 1. **Các khái niệm chính**

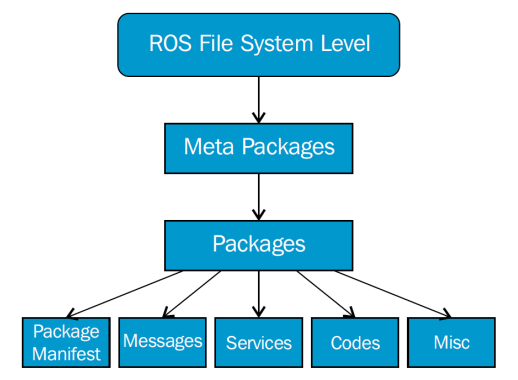
Có ba cấp độ khái niệm trong ROS. Cấp độ đầu tiên được gọi là "cấp độ hệ thống trong tổng số tệp ”đề cập đến cách các tệp khác nhau được tổ chức trong máy tính, các công cụ ROS để quản lý mã nguồn, hướng dẫn biên dịch và định nghĩa thông báo. Mức thứ hai được gọi là "mức đồ thị tính toán" là các thành phần của mạng ngang hàng của các nút ROS, tức là các quá trình. Cuối cùng, cấp thứ ba được gọi là "cấp cộng đồng" liên quan đến cách thức mà người dùng chia sẻ phần mềm để cho phép mọi người cùng sử dụng.



*Các cấp độ khái niệm ROS*

* + 1. **Cấp hệ thống tập tin.**

Tương tự như một hệ điều hành, các tệp ROS cũng được tổ chức trên đĩa theo một cách cụ thể. Ở cấp độ này, chứng ta có thể thấy chúng được tổ chức như thế nào các tệp trên đĩa. Hình dưới cho thấy cách tổ chức tệp và thư mục:



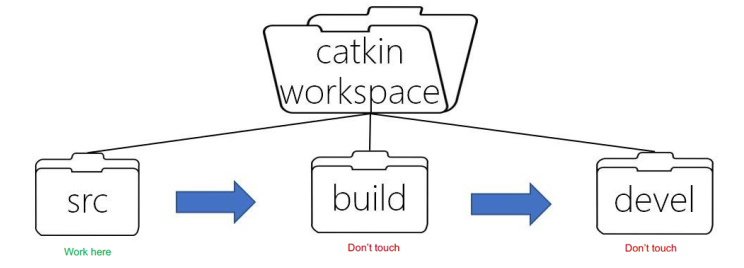
*Cấp hệ thống tệp ROS*

ROS dựa trên CMake và có thể được sử dụng bởi nhiều hệ điều hành như Linux hoặc Windows. CMake là một trình tạo ra các công cụ sáng tạo, nó là một Công cụ cấp cao nhất giúp đơn giản hóa quá trình xây dựng và xây dựng. Nó có thể xử lý các dự án lớn và có khả năng mở rộng tốt. Đối với một nền tảng trên quy mô lớn như ROS, đã mở rộng CMake cho một hệ thống xây dựng được gọi là Catkin. Catkin là hệ thống biên dịch ROS tạo ra các chương trình thực thi, thư viện và giao diện và cung cấp khái niệm về không gian làm việc trong xây dựng của mỗi dự án.

Cấu trúc không gian làm việc catkin bao gồm các thư mục / src, / build, / devel. Các ba path này cũng có thể bao gồm những path khác trong một số tùy chọn xây dựng nhưng 3 thư mục là mặc định cho hệ thống xây dựng catkin. Vai trò của chứng cụ thể như sau:

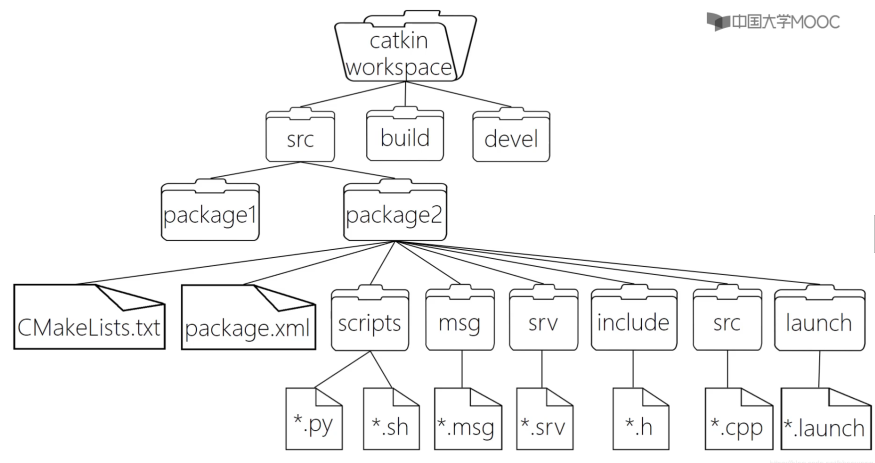
* + Không gian nguồn (/src): Chứa mã nguồn nơi người dùng có thể trích xuất, kiểm tra hoặc sao chép mã nguồn của các gói muốn xây dựng.
  + Không gian xây dựng (/build): Đây là không gian mà CMake yêu cầu xây dựng các gói trong không gian làm việc catkin. CMake và Catkin giữ thông tin bộ nhớ cache và các tệp trung gian khác.
  + Không gian phát triển (/ devel): Tìm thấy tệp đối tượng được tạo (bao gồm tệp tiêu đề, thư viện liên kết động, liên kết tĩnh, tệp thực thi, v.v.) và các biến môi trường.

Trong quá trình xây dựng, quy trình làm việc cho không gian làm việc catkin được hiển thị bằng các mũi tên màu xanh trong hình dưới. Hai quy trình cuối cùng được tạo và quản lý tự động bởi hệ thống catkin. Việc sử dụng chính cho người dùng ROS là thư mục src, nơi các gói được tạo hoặc sao chép được lưu trữ trong thư mục đó. Tại thời điểm xây dựng, hệ thống xây dựng của catkin tìm và biên dịch tất cả các gói mã nguồn từ đường dẫn / src.

**

*Catkin\_make workspace*

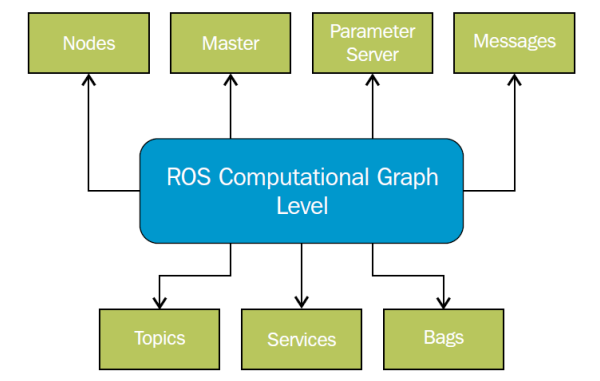
Khi tạo một gói, thông thường sẽ tìm thấy các thư mục thể hiện trong hình.



*Cấu tạo và đặc điểm của catkin\_make*

* / include /: Thư mục này chứa các tiêu đề (\* .h) và các thư viện (\* .lib, \* .so).
* / config /: Tất cả các cài đặt được lưu trữ trong thư mục này bao gồm
* các tham số xác định, chẳng hạn như bộ điều khiển rô bốt có phần mở rộng (\* .yaml).
* / launcher /: Chứa các tệp (\* .launch) để khởi chạy các nút đã chọn từ gói hoặc các gói khác bằng lệnh <include />.
* / msg /: Chứa kiểu dữ liệu của thông báo cho ứng dụng của chúng tôi (\* .msg).
* / srv /: Thư mục này chứa loại thông báo cho các dịch vụ (\* .srv).
* / action /: Chứa kiểu dữ liệu của thông báo được sử dụng trong các hành động (\* .action).
* / src /: Tất cả mã được tính trong gói phải ở bên trong thư mục có phần mở rộng (\* .cpp).
* package.xml: Mô tả gói và danh sách các phần phụ thuộc được sử dụng bởi nó.
* CmakeLists.txt: Danh sách các yêu cầu và phụ thuộc được trình khởi chạy gói biên dịch.
  + 1. **Mức đồ họa tính toán**

Mức đồ họa tính toán mô tả cơ sở hạ tầng mà các thành phần ROS sử dụng để giao tiếp. Nó dựa trên một mạng lưới các quy trình ngang hàng (các nút) và được tạo thành từ một số đơn vị, chẳng hạn như những đơn vị được hiển thị trong hình sau



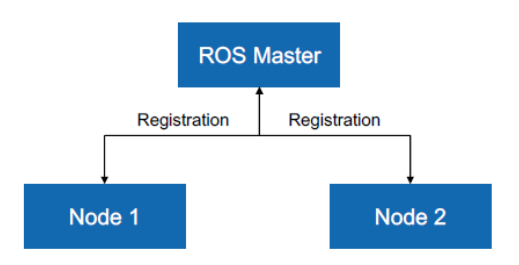
*Cấu trúc lớp đồ thị ROS*

* + - 1. **Node.**

Node có thể thực thi trong hệ thống ROS và thực hiện phần tính toán. Chúng có thể được kết nối với các nút khác thông qua 2 kiểu giao tiếp: Publisher/Subscriber hoặc Services. Mỗi Node có tên riêng để được công nhận và phân biệt với các nút khác. Trong ROS có nhiều ngôn ngữ để lập trình, chẳng hạn như C ++, Python hoặc Java.

* + - 1. **Ros Master.**

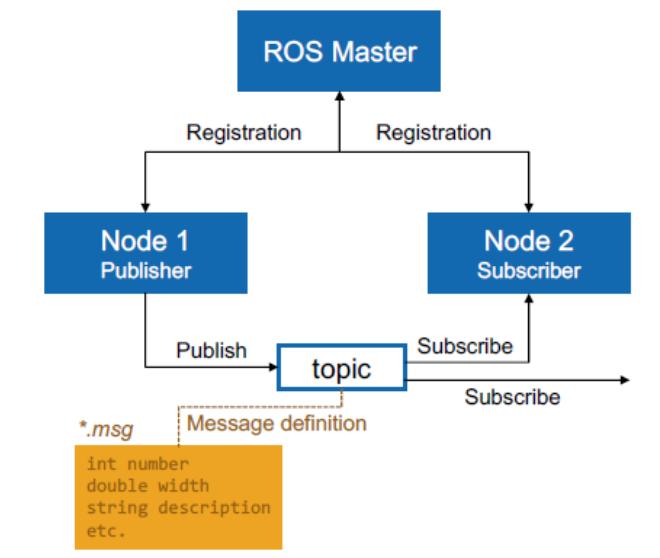
Ros Master cho phép hoạt động của mạng ROS và là quá trình đầu tiên phải được thực hiện khi chúng ta đang sử dụng ROS. Chức năng của nó là đăng ký tất cả các nút đang chạy, chủ đề và các dịch vụ hiện có. Điều này là cần thiết để kết nối các nút và giám sát tất cả các đơn vị trong hệ thống.

**

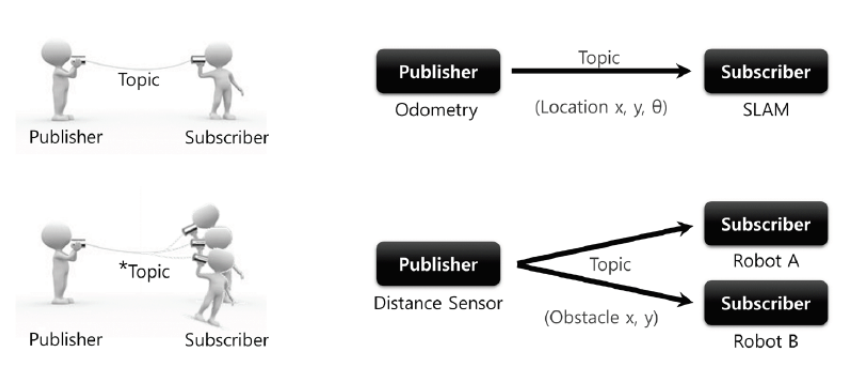
*Đăng ký Node trong ROS Master*

* + - 1. **Topics.**

Các Node chia sẻ thông tin bằng cách chuyển và nhận message. Những thông báo này được xuất bản dưới một cái tên đặc biệt được gọi là một Topic. Do đó, các Node có thể Pulishe hoặc Register một Topic cụ thể để có được message mong muốn. Nhiều Node có thể đăng ký cùng một Topic để nhận được thông điệp mà nó cần. Giao tiếp giữa các Node được thực hiện theo cách một chiều, vì vậy bất kỳ Node nào gửi thông tin về một chủ đề đều không thể nhận được trả lời trên cùng một kênh, cũng như không biết liệu nút nhận đã nhận được tin nhắn hay chưa

**

*Các Node giao tiếp thông qua Topic bằng ti message \* .msg*

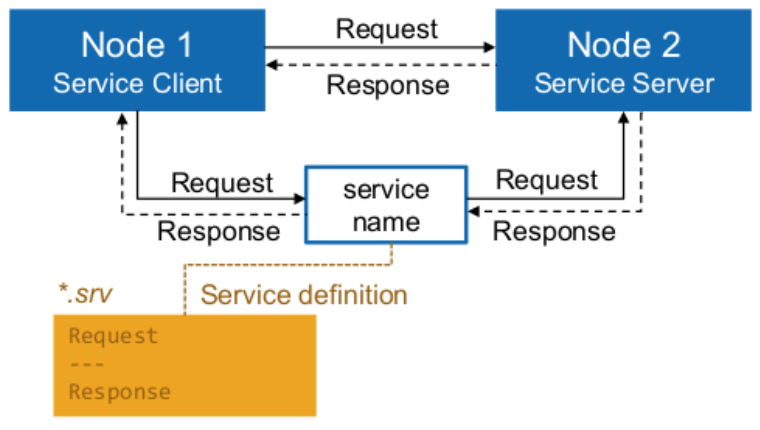
**

*Truyền thông điệp qua các Topic*

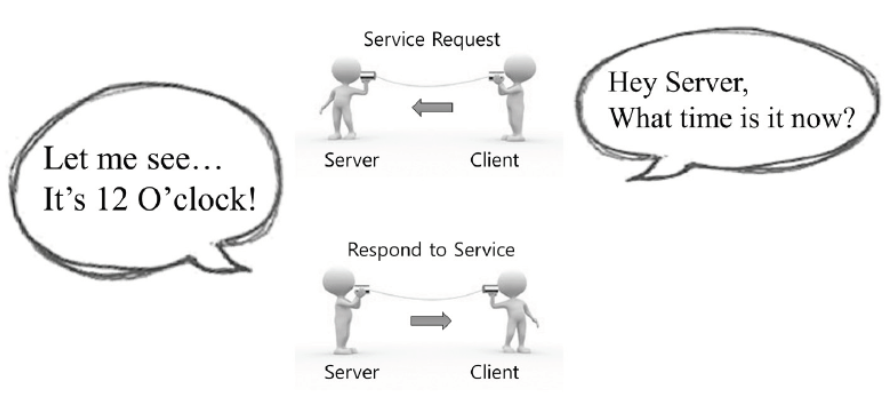
* + - 1. **Services.**

Mặc dù phương thức giao tiếp ở giữa thông qua các Topic đã được giải thích ở trên, có một phương pháp khác trong ROS được gọi là Services. Các Service trong ROS khác với chủ đề do các đặc điểm sau: chúng là hai hướng và một đối một.

Quá trình giao tiếp đồng bộ này dựa trên hệ thống máy khách / máy chủ hoặc yêu cầu / phản hồi, trong đó một nút máy khách gửi một số dữ liệu được gọi là yêu cầu tới một nút máy chủ và đợi cho đến khi máy chủ có thể phản hồi. Máy chủ, có đã nhận được yêu cầu này, hãy thực hiện một số hành động và gửi một số dữ liệu được gọi là phản hồi tới nút máy khách Mô tả của dịch vụ được lưu trữ trong tệp định nghĩa của Dịch vụ \* .srv, được lưu trữ trong thư mục con / srv.



*Giao tiếp request/response giữa các node thực hiện Service*

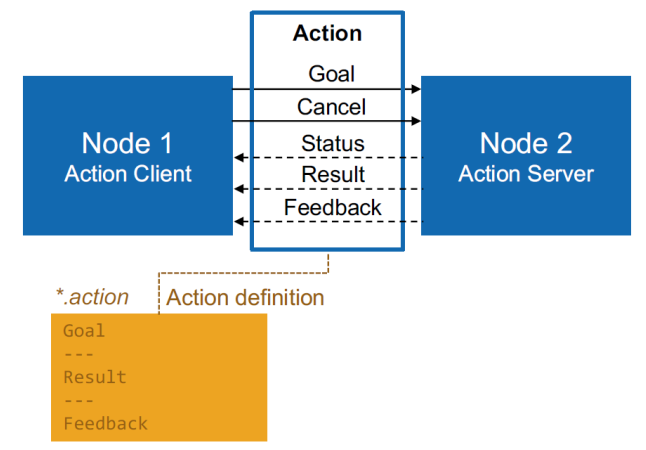


*Trao đổi message với nhau thông qua message*

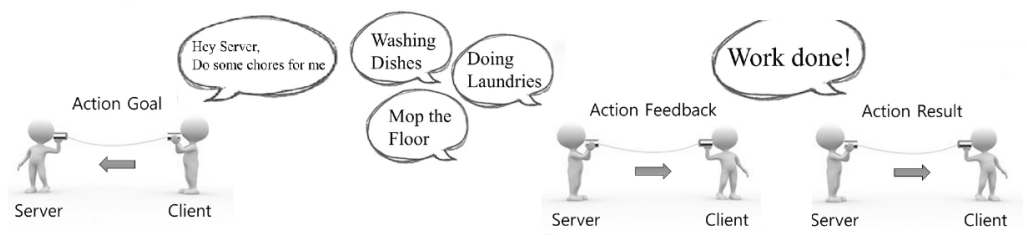
* + - 1. **Actions.**

Một Actions là giao tiếp hai chiều không đồng bộ giữa các Node hoạt động dựa trên action's client-based active nodes và the server system of the action. Hệ thống actions, chẳng hạn như các service đã giải thích ở trên, gửi một số ứng dụng dữ liệu action client đến action server được gọi là đích và action server phản hồi một kết quả. action khác với service vì server có thể cung cấp phản hồi và trạng thái cho client bất kỳ lúc nào về quá trình đang được thực hiện và node client có thể hủy mục tiêu trước đó cần thiết cho server bất kỳ lúc nào. Như trong các cuộc gọi service ROS, các action sẽ xác định một số thông báo để giao tiếp server. Điều này có thể thực hiện được nhờ vào các thông số kỹ thuật của các action, xác định các thông báo này trong một tệp action có phần mở rộng là \* .action. Các tệp này được lưu trữ trong một thư mục con /action/ trong thư mục gói. Tệp \* .action được chia thành 3 phần.

Mỗi phần cách nhau 3 dấu gạch ngang (-). Trong phần đầu tiên, mục tiêu được xác định, trong phần thứ hai, phản hồi được thiết lập và cuối cùng, định nghĩa của kết quả được chỉ định.

**

*Giao tiếp hai chiều giữa các node thông qua action*

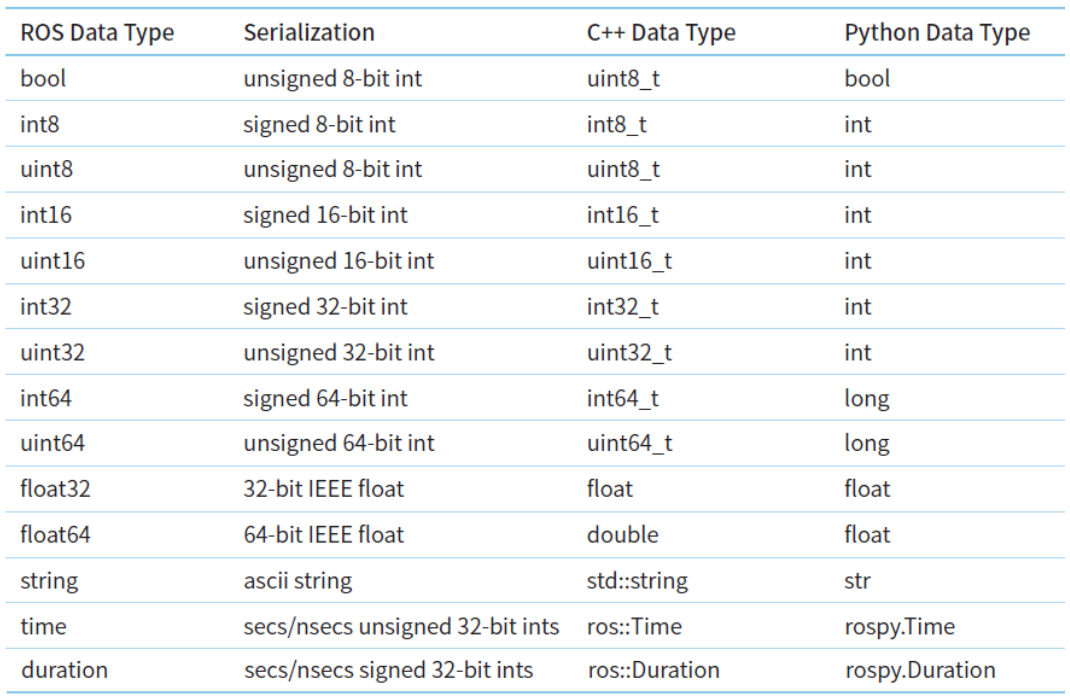


*Thông điệp hành động*

* + - 1. **Messages.**

Messages chỉ đơn giản là một cấu trúc dữ liệu, bao gồm các trường được viết. Các kiểu nguyên thủy tiêu chuẩn (số nguyên, dấu phẩy động, boolean, v.v.). Trong nhiều gói được sử dụng trong ROS các message đặc biệt được gọi là 'tiêu đề' được bao gồm, về cơ bản là các thông báo do người khác tạo ra. Có ba trường được xác định trong thông báo tiêu đề

* + uint32 seq: Tương ứng với số lượng tin nhắn được gửi bởi một Publisher
  + time stamp: Thời gian chính xác mà tin nhắn được gửi đi
  + string frame\_id: Hiển thị hệ quy chiếu được sử dụng

**

*Các kiểu dữ liệu cơ bản cho thông báo trong ROS*

* + - 1. **Parameter Server.**

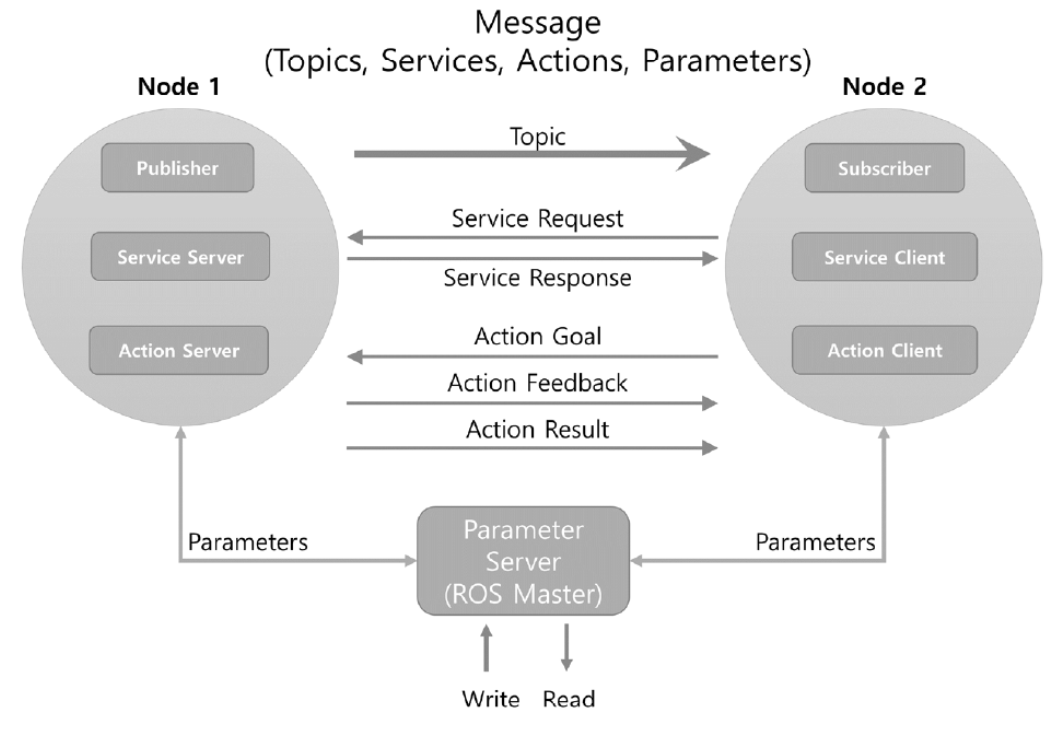
Parameter Server cho phép nhập dữ liệu ở một vị trí, là một phần của Master. Về cơ bản nó là một kho lưu trữ các hằng số.

* + - 1. **Bags.**

Bags là một định dạng để lưu trữ và tái tạo dữ liệu Tin nhắn ROS. Bags là một cơ chế quan trọng để lưu trữ dữ liệu, chẳng hạn như dữ liệu cảm biến, có thể khó thu thập, nhưng rất cần thiết. để phát triển và kiểm tra các thuật toán.

* + - 1. **Sơ đồ tóm tắt.**

Hệ thống ROS cho phép các nút khác nhau giao tiếp với nhau, trao đổi thông tin và dữ liệu. Nhưng tuy nhiên, toàn bộ hệ thống cần một ROS Master đang chạy để nhận thấy sự tồn tại của và bắt đầu giao tiếp với nhau. ROS Master cho phép các nút riêng lẻ xác định vị trí của nhau trong hệ thống, theo dõi các nhà xuất bản và người đăng ký chủ đề và dịch vụ. Một nút thường là một chương trình nhỏ được viết bằng Python hoặc C ++ thực thi một số tác vụ hoặc quy trình tương đối đơn giản. Các nút có thể được bắt đầu và dừng độc lập với nhau và giao tiếp bằng cách chuyển các tin nhắn. Một nút có thể đăng thông báo các nút khác về các chủ đề, dịch vụ hoặc hành động nhất định riêng biệt

**

*Message communication between nodes*

* 1. **Công cụ**

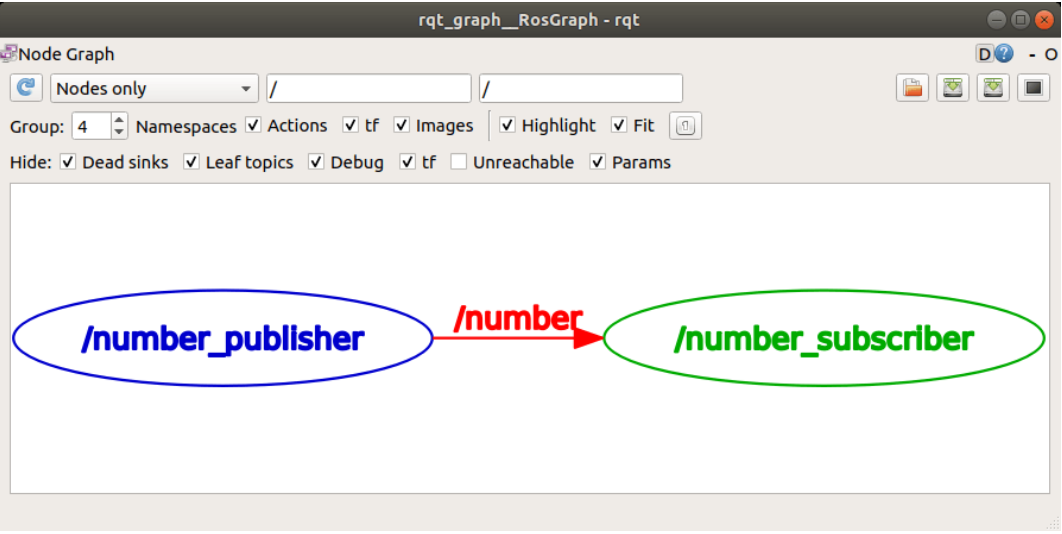
Có một số công cụ có thể trợ giúp khi sử dụng ROS. Phải có coi các công cụ GUI này là bổ sung cho các công cụ trực tuyến yêu cầu. Có rất nhiều công cụ ROS, bao gồm các công cụ mà người dùng ROS cũng đã phát hành cá nhân.

* + 1. **Launch Files.**

Trong một dự án ROS, có khi có thể chạy nhiều nút ROS cùng một lúc thời gian để tạo ra các hệ thống phức tạp hơn. ROS có một công cụ gọi là roslaunch cho phép người dùng chạy nhiều nút, thiết lập các tham số của định cấu hình từng nút, đổi tên chủ đề mặc định và thậm chí đổi tên nút. Mục đích của việc này là dễ dàng cấu hình hệ thống toàn cầu.

* + 1. **RQT Graph**

RQT Graph cung cấp hình ảnh trực quan của hệ thống ROS, hiển thị các nút và kết nối giữa chúng cho phép gỡ lỗi và hiểu hệ thống đang chạy và nó hoạt động như thế nào.



*Hình ảnh hóa giao tiếp giữa node publisher (màu xanh lam) và node subcriber (màu xanh lá) thông qua số chủ đề (màu đỏ) trong Biểu đồ RQT*

* + 1. **RQT Rosbag**

Rosbag là một bộ công cụ để ghi và phát dữ liệu chủ đề ROS. Dữ liệu được lưu trữ trong các tệp bag và có sẵn các công cụ trực tuyến. lệnh để làm việc với bag. Rosbag ngăn chặn quá trình deserialization và tái thiết lập thông điệp và các công cụ chính của nó là:

* + rosbag info: Hiển thị nội dung của các tệp túi, chẳng hạn như các chủ đề đã ghi, thời gian bắt đầu và kết thúc, số lượng tin nhắn, tần suất và số liệu thống kê về nén
  + rosbag record: Viết nội dung của tất cả các tin nhắn được xuất bản trên những chủ đề mà chúng tôi muốn đăng ký và thông tin được lưu trữ trong tệp .bag.
  + rosbag play: Đọc tệp bag và đăng thông tin về các chủ đề ROS đồng bộ trong thời gian. Hệ thống ROS có thể sử dụng thông tin này như thể nó ở trong thời gian thực.
    1. **RQT reconfigure**

Điều chỉnh của gói " Dynamic reconfigure" cho phép sửa đổi trong thời gian thực tất cả các tham số nút, để sửa đổi nhanh chóng. Cấu hình của các tham số có thể được xuất để sử dụng trong tương lai.

* + 1. **MoveIt.**

Phần mềm dựa trên lập kế hoạch chuyển động, xem xét tìm kiếm khách hàng tiềm năng 3D, động học, điều khiển và chuyển hướng. Cung cấp nền tảng để phát triển ứng dụng rô bốt, đánh giá các thiết kế rô bốt và sản phẩm xây dựng mới Robot tích hợp cho các ứng dụng công nghiệp, R & D, v.v.



*Logo của MoveIt*

* 1. **Visualization.**
     1. **Giao diện hiển thị Rviz**

RViz là viết tắt của ROS visualization. Đó là một môi trường xem 3D sử dụng chung cho rô bốt, cảm biến và thuật toán. Cho phép xem bản đồ, rô bốt, đối tượng, dữ liệu laser, hình ảnh máy ảnh, đám mây điểm và điểm đánh dấu. Giống như hầu hết các Các công cụ ROS, có thể được sử dụng cho bất kỳ Robot nào và được thiết lập nhanh chóng cho một ứng dụng cụ thể



Logo của Rviz

RViz cung cấp một giao diện đơn giản để chọn thông tin mà muốn được hiển thị. Màn hình hiển thị là một cái aplication vẽ thứ gì đó trong hình ảnh 3D của RViz có thể có một vài tùy chọn có sẵn trong danh sách các màn hình. Một ví dụ là đám mây điểm, trạng thái của Robot, mô hình hóa các bộ phận cơ khí của rô bốt, tọa độ tf, cảm biến, máy ảnh, kích thước lưới, v.v.



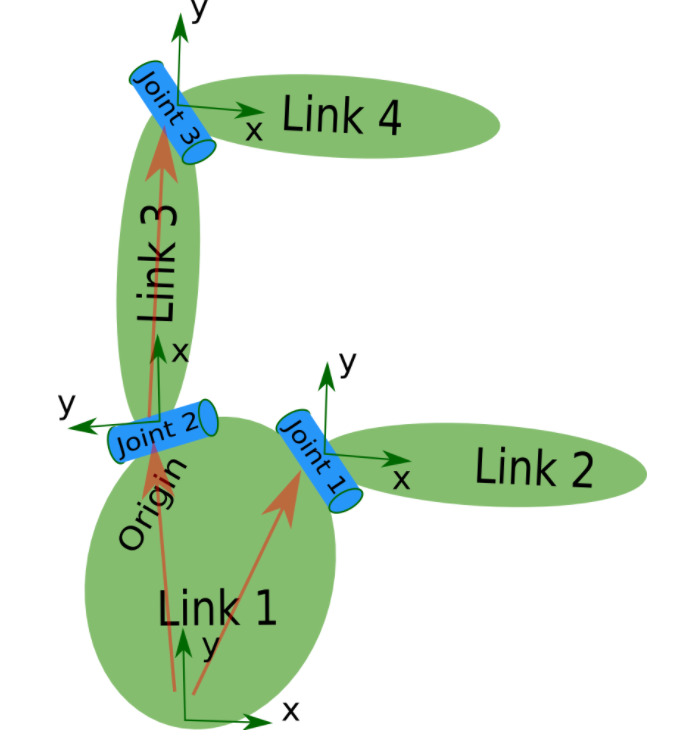
*Mô hình Robot trên Rviz*

* + 1. **Unified Robot Description Format (URDF)**
       1. **Giới thiệu URDF**

Trong ROS, có thể hình dung mô hình của Robot bằng cách sử dụng các tệp định dạng mô tả Robot (URDF). Tuy nhiên, chỉ những robot có các liên kết cứng được kết nối bằng các khớp mới có thể được mô tả sử dụng các mô hình URDF. Mô hình URDF có thể được sử dụng để tính toán động học, thêm các khung tọa độ mới và di chuyển chúng theo các giá trị mã hóa của rô bốt. Ngoài ra, nó có thể bao gồm các tính chất vật lý khác như quán tính, va chạm, động lực học chung, v.v.

Các tệp URDF dựa trên ngôn ngữ XML và được cấu thành bởi file với phần mở rộng .xml và có thể được đọc để trích xuất thông tin. Các đọc tệp để trích xuất thông tin quan trọng từ mô hình được gọi là phân tích cú pháp và chương trình hoặc chức năng thực hiện nó, phân tích cú pháp.

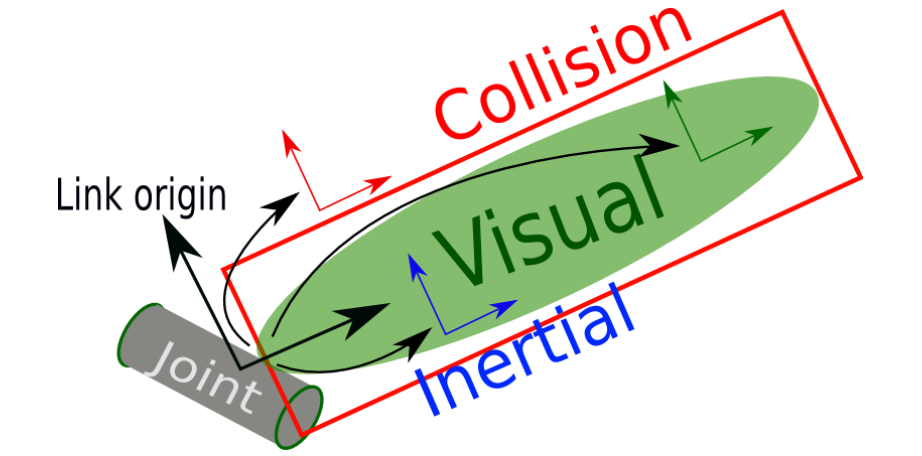
Mô tả của mô hình về cơ bản bao gồm việc kết hợp hai tập hợp: tập hợp of links (liên kết) và set of union (liên kết). Cách xây dựng và hình dung một mô hình robot trong URDF là viết và biên dịch tệp URDF. Sau khi nó được tạo ra kết xuất 3D của rô bốt bằng cách sử dụng tệp URDF, có thể sử dụng biểu diễn mô phỏng chuyển động của Robot. Để làm điều này, người dùng phải xuất bản các điều kiện Robot trong tf, sử dụng một node (hoặc các node) để xuất bản thông tin chuyển đổi.



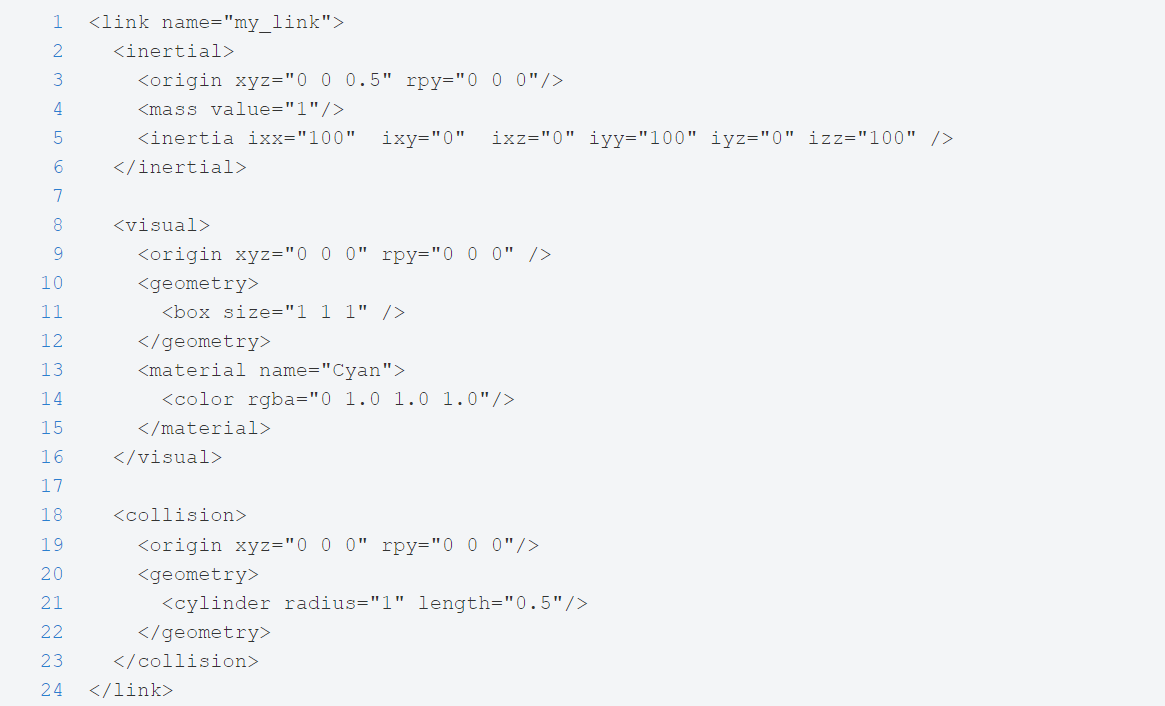
*Các yếu tố cơ bản của trực quan hóa URDF: Link và joint*

* + - 1. **Links**

Các liên kết hoặc liên kết (tập hợp liên kết) mô tả phần vật lý cứng của rô bốt và cho phép chỉ định các thuộc tính của nó, chẳng hạn như kích thước, hình dạng, màu sắc hoặc một lưới 3D phức tạp nhập khẩu. Nó bao gồm ba yếu tố:



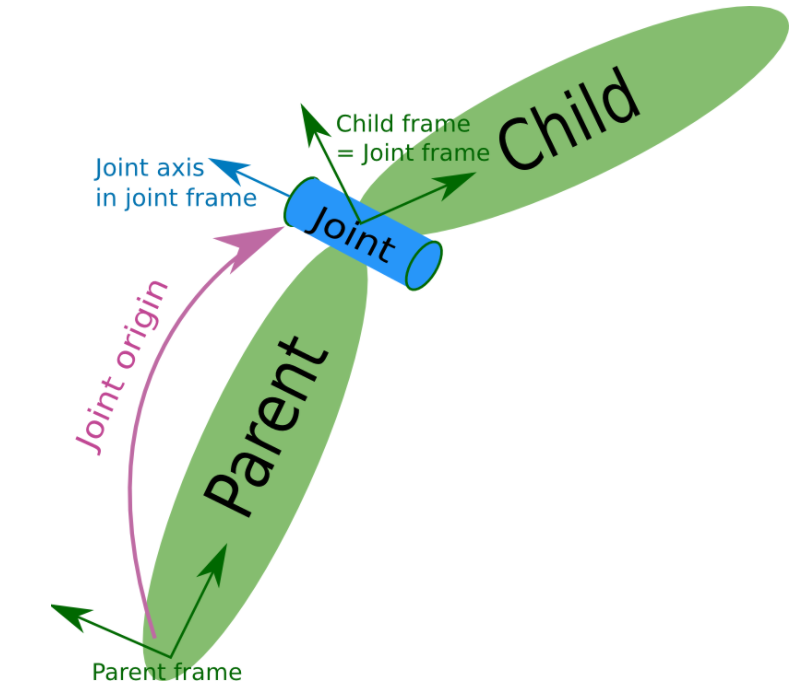
*Biểu diễn trực quan, quán tính và va chạm của một liên kết với URDF*



*Mô tả với file.xml*

* + - 1. **Joints.**

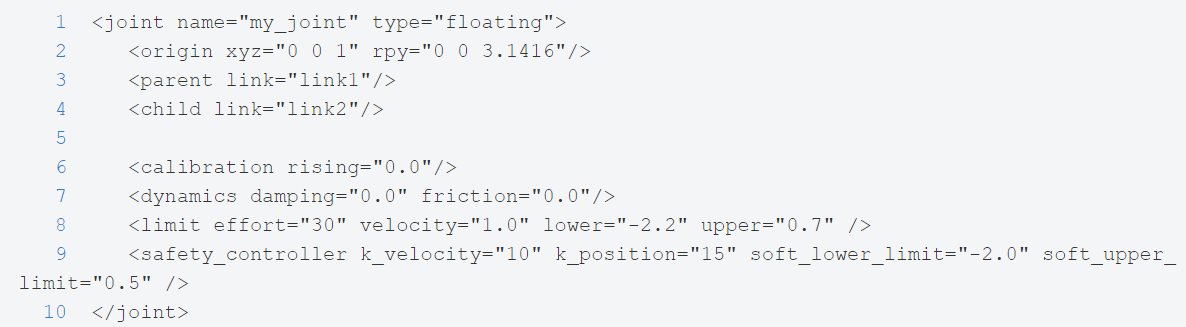
Các mối nối (joints) chỉ mối quan hệ giữa các link của Robot. Chúng mô tả động học và động lực học của từng khớp, ngoài ra để xác định giới hạn va chạm của robot. Một khớp được xác định khi nó các link chỉ định 2 liên kết gắn liền với nó: liên kết đầu tiên được gọi là parent link và thứ hai được gọi là child link .



*Mối quan hệ chung và mối quan hệ parent-link của nó với các liên kết*

Các loại khớp là:

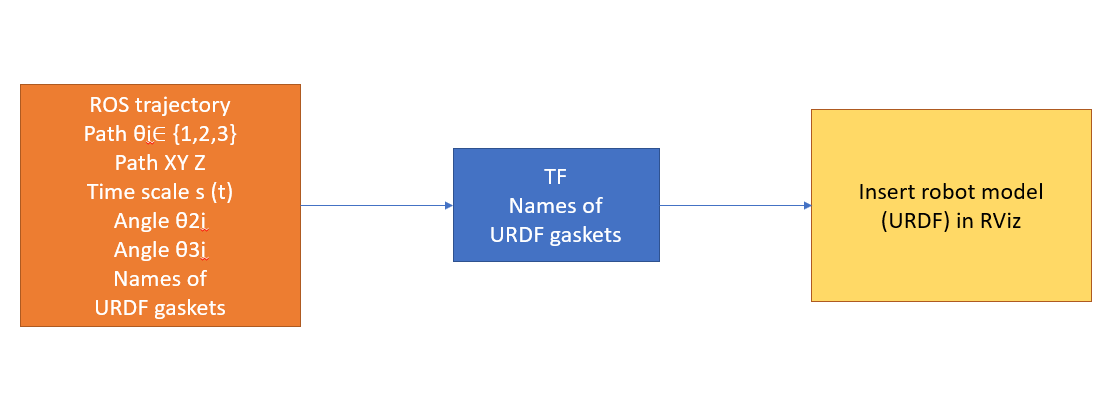
* + Revolute: Cho phép hai chất rắn quay quanh một trục chung và có một đường xoắn giới hạn được chỉ định bởi một giới hạn trên và một giới hạn dưới.
  + Continuous: Đó là một cuộc nổi dậy nhưng không có giới hạn.
  + Prismatic: Một khớp sẽ trượt dọc theo một trục. Có phạm vi
  + bị giới hạn bởi một giới hạn dưới và một giới hạn trên.
  + fixed: Nó không thực sự là một khớp vì nó ngăn cản chuyển động. Liên minh không không có bậc tự do. Loại khớp này không yêu cầu bất kỳ các yếu tố mô tả mối nối.
  + Floating: Liên minh này giữ cho tất cả các bậc tự do được tự do.
  + Planar: Khớp cho phép chuyển động trong một mặt phẳng.



*Mô tả joint với file .xml*

1. **Giao diện hiển thị Rviz**

* RViz cung cấp khả năng hình dung các thành phần cơ khí của robot từ một cách đơn giản hơn. Trong luận án này, chúng tôi muốn mô phỏng quỹ đạo tuyến tính của nền tảng di động của rô bốt delta. Để đạt được mục tiêu này, bạn cần mọi lúc của quỹ đạo giá trị của các góc của khớp nối các thành phần và vị trí xyz của trọng tâm của bệ chuyển động. Vị trí trong không gian của cánh tay và cẳng tay được xác định từ các góc của mỗi khớp so với các bộ phận khác cơ học và vị trí của nền tảng di động được xác định bởi các tọa độ của xyz centroid so với hệ quy chiếu toàn cục. Do đó, hình dung các bộ phận cơ khí trong 2 nhóm riêng biệt (3 chuỗi hệ thống truyền lực tics và nền tảng di động). Nếu chuyển động học nghịch đảo là đúng, thì nền tảng Di động phải vừa khít hoàn toàn giữa xương bánh chè (phần cuối của cẳng tay) nối với cánh tay với nền tảng di động.
* Kết nối giữa ROS và Rviz:
  + Về cơ bản, Ta có thể hình dung quỹ đạo Robot Delta qua các bước sau:



*Biểu đồ các bước để có có thể mô phỏng Robot Delta*

*Bước 1*: Dữ liệu sau được xuất từ ​​quỹ đạo được tạo trong ROS: quỹ đạo trong không gian góc của động cơ θi∈ {1,2,3}, quỹ đạo trong không gian Descartes của trọng tâm của end-effector XYZ và thang thời gian được sử dụng s (t). Sau đó, các góc bên trong θ2i và θ3i được xác định của 3 chuỗi động học. Điều này có thể thực hiện được bằng cách tạo một nút có tên Poser\_rviz\_realtime\_tm1\_adams.py dự kiến ​​làm dữ liệu đầu vào dữ liệu đường dẫn θi∈ {1,2,3}, XY Z và s (t) và khi xuất ra một đối tượng kiểu JointState (). Đối tượng cuối cùng này chứa dữ liệu về các góc bên trong tính bằng radian, góc của động cơ tính bằng radian và vị trí xyz của end-effector. Ngoài ra, mỗi góc và tọa độ được gán một tên duy nhất là loại chuỗi được sử dụng để liên kết các khớp được cấu hình trong RViz và mô hình URDF liên quan đến giá trị số của mỗi mô hình

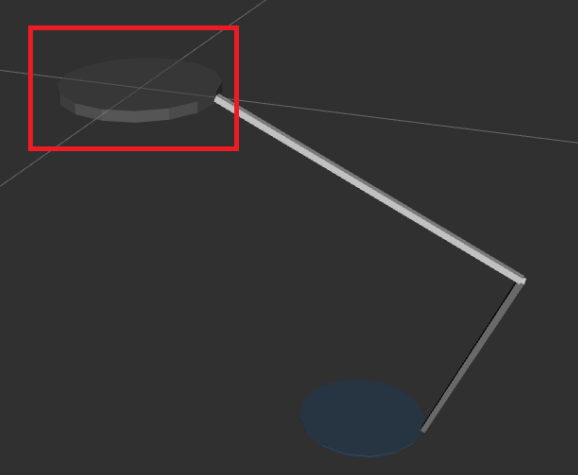
*Bước 2*: Chuyển đổi các giá trị của từng khớp và tọa độ có được từ bước 1 đến tf. RViz có một nút tích hợp tên là estados\_robot\_pub.py trông giống như yêu cầu đầu vào các giá trị của mỗi bảng với mỗi tên duy nhất tương ứng và khi xuất ra các giá trị tf với mỗi tên duy nhất tương ứng. Các cấu hình tf của rô bốt delta là nhờ vào mô hình URDF

*Bước 3*: Trước khi tạo đường dẫn trong ROS, Robot Delta phải được định cấu hình ở định dạng URDF. Tên của từng khớp và tọa độ của bộ tạo hiệu ứng trong Các tệp URDF ở định dạng .xml giống hệt như bước 1 và 2. Mô hình rô bốt URDF được đưa vào RViz để các giá trị tf được tính trong bước 2 được sử dụng để cung cấp chuyển động cho các phần được tạo bởi tệp URDF

* Tên các khớp của rô bốt delta trong URDF :

|  |  |  |
| --- | --- | --- |
| Ký Hiệu | Tên | Mô Tả |
| θ11 | base\_brazo1 | Góc động cơ trên hệ thống truyền lực 1 |
| θ12 | base\_brazo2 | Góc động cơ trên hệ thống truyền lực 2 |
| θ13 | base\_brazo3 | Góc động cơ trên hệ thống truyền lực 3 |
| θ21 | codo1\_a | Góc trong 2 của khớp bi nối cánh tay với cẳng tay trên chuỗi động học 1 |
| θ31 | codo1\_b | Góc trong 3 của khớp bi nối cánh tay với cẳng tay trên chuỗi động học 1 |
| θ22 | codo2\_a | Góc trong 2 của khớp bi nối cánh tay với cẳng tay trên chuỗi động học 2 |
| θ32 | codo2\_b | Góc trong 3 của khớp bi nối cánh tay với cẳng tay trên chuỗi động học 2 |
| θ23 | codo3\_a | Góc trong 2 của khớp bi nối cánh tay với cẳng tay trên chuỗi động học 3 |
| θ33 | codo3\_b | Góc trong 3 của khớp bi nối cánh tay với cẳng tay trên chuỗi động học 3 |
| X | act\_x | Vị trí X của end-effector |
| Y | act\_y | Vị trí Y của end-effector |
| Z | act\_z | Vị trí Z của end-effector |

* Tệp Định dạng mô tả rô bốt hợp nhất (URDF) dựa trên xml được sử dụng để hiển thị mô hình Robot
  + *Base\_link: Base\_link* là link cố định và được biểu diễn dưới dạng hình trụ với bán kính là Ra



<!--BASE SUPERIOR:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-->

    <link name="base\_link">

        <visual>

            <geometry>

                <cylinder length="0.005" radius="0.210"/>  <!--f/2-->

            </geometry>

            <material name="gris">

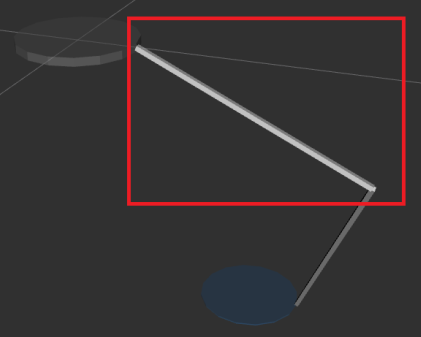
                <color rgba="0.55 0.55 0.55 0.5"/>

            </material>

        </visual>

    </link>

* + *Cánh tay:* Các nhánh là các liên kết được đặt tên là "brazo1, brazo2 và brazo3" và được biểu thị dưới dạng hình trụ có chiều dài là La. Các cánh tay này được gắn vào base\_link cố định và thông qua một khớp xoay đại diện cho các động cơ hoặc bộ truyền động của Robot Delta. Các cánh tay với tên " base\_brazo1, base\_brazo2 và base\_brazo3" cho mỗi cánh tay tương ứng.



<!--BRAZOS SUPERIORES:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-->

    <!--BRAZO\_1-->

    <link name="brazo1">

        <visual>

            <origin xyz="0.310 0 0" rpy="0 0 0"/> <!--la = 150-->

            <geometry>

                <box size="0.620 0.003 0.003"/> <!--la = 150-->

            </geometry>

            <material name="blanco">

                <color rgba="0.9 0.9 0.9 1"/>

            </material>

        </visual>

    </link>

    <joint name="base\_brazo1" type="revolute">

        <axis xyz="0 1 0"/>

        <limit effort="100.0" lower="0.0" upper="1.8" velocity="0.5"/>

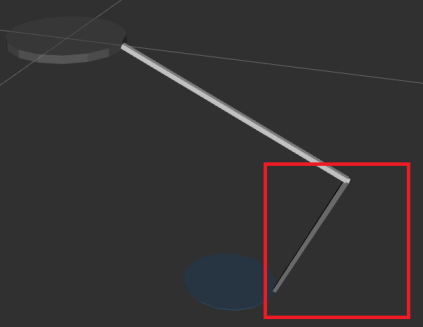
        <parent link="base\_link"/>

        <child link="brazo1"/>

        <origin xyz="-0.181865335 0.105 0" rpy="0 0 2.61799"/>

    </joint>

* + *Cẳng tay: Cẳng tay* là các link được đặt tên là "barra1\_b, barra2\_b và barra3\_b" được biểu diễn là hình trụ có chiều dài Lb. Những cánh tay này kết hợp lần lượt đến cánh tay tương ứng của nó thông qua hai khớp kiểu quay vòng đại diện cho khớp bi. Các khớp được gán các tên **codo1\_a , codo1\_b ,** **codo2\_a , codo2\_b , codo3\_a , codo3\_b** cho mỗi cánh tay tương ứng.



<!--BRAZOS INFERIORES:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-->

    <!--Ante\_BRAZO\_1-->

    <link name="barra1\_a"/>

    <joint name="codo1\_a" type="revolute">

        <axis xyz="0 -1 0"/>

        <limit effort="100.0" lower="-3.14" upper="3.14" velocity="0.5"/>

        <parent link="brazo1"/>

        <child link="barra1\_a"/>

        <origin xyz="0.620 0 0" rpy="0 0 0"/>

    </joint>

    <link name="barra1\_b">

        <visual>

            <origin xyz="-0.440 0 0" rpy="0 0 0"/> <!--0.175-->

            <geometry>

                <box size="0.880 0.003 0.003"/> <!--lb = 205-->

            </geometry>

            <material name="blanco"/>

        </visual>

    </link>

    <joint name="codo1\_b" type="revolute">

        <axis xyz="0 0 1"/>

        <limit effort="100.0" lower="-3.14" upper="3.14" velocity="0.5"/>

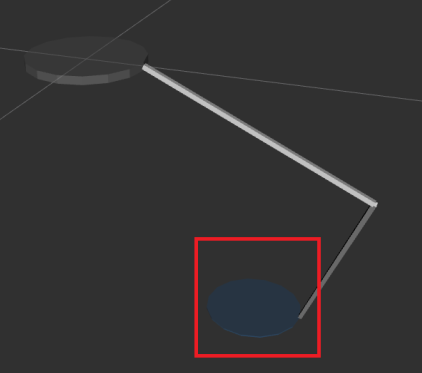
        <parent link="barra1\_a"/>

        <child link="barra1\_b"/>

        <origin xyz="0 0 0" rpy="0 0 0"/>

    </joint>

* + *End-effector:* là một link được đặt tên là ‘**actuador**’ và được biểu diễn dưới dạng hình trụ bán kính Rb. Vị trí và hướng của link này không liên quan đến bất kỳ link nào khác và chỉ được xác định trong mối quan hệ với hệ quy chiếu toàn cục. Để định vị tâm của hình trụ, 3 khớp kiểu lăng trụ được sử dụng, biểu thị các tọa độ X, Yvà Z tương ướng với joint là 'act\_x', 'act\_y' và 'act\_z’.



<!--ACTUADOR:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-->

    <link name="actuador">

        <visual>

            <origin xyz="0 0 0" rpy="0 -1.57075 0"/>

            <geometry>

                <cylinder length="0.001" radius="0.050"/>

            </geometry>

            <material name="colorte">

                <color rgba="0.15 0.55 0.95 0.3"/>

            </material>

        </visual>

    </link>

    <link name="aux1"/>

    <link name="aux2"/>

    <joint name="act\_x" type="prismatic">

        <parent link="base\_link"/>

        <child link="aux1"/>

        <limit effort="100.0" lower="-1" upper="1" velocity="0.5"/>

        <origin rpy="0 0 0" xyz="0 0 0"/>

    </joint>

    <joint name="act\_y" type="prismatic">

        <parent link="aux1"/>

        <child link="aux2"/>

        <limit effort="100.0" lower="-1" upper="1" velocity="0.5"/>

        <origin rpy="0 0 1.57075" xyz="0 0 0"/>

    </joint>

    <joint name="act\_z" type="prismatic">

        <parent link="aux2"/>

        <child link="actuador"/>

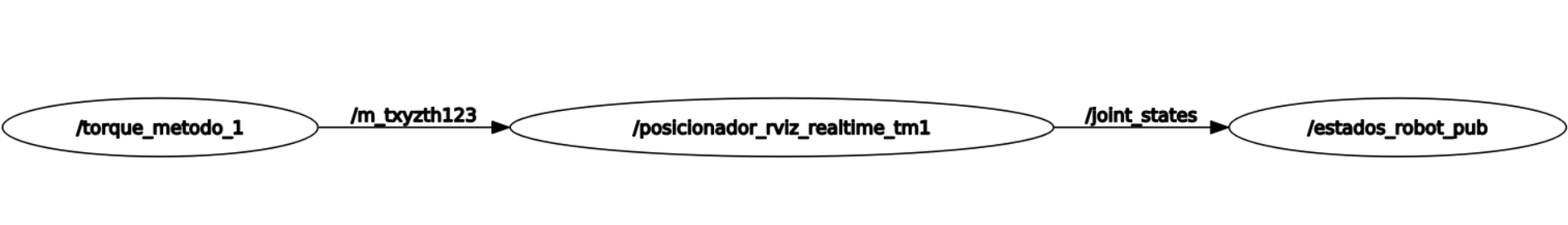
        <limit effort="100.0" lower="0" upper="1" velocity="0.5"/>

        <origin rpy="0 -1.57075 0" xyz="0 0 0"/>

    </joint>

</robot>

* *Topic và Node*

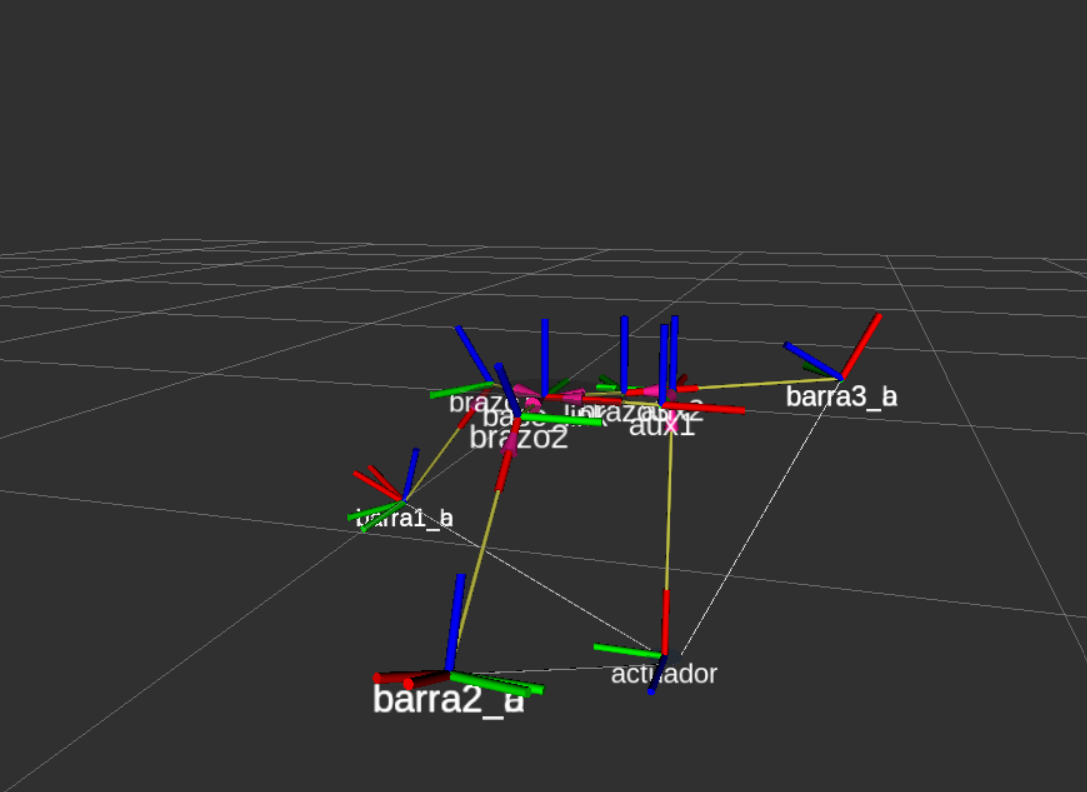


*Các Topic và Node trong ROS để hình dung Robot Delta.*

**Subscriber** : Topic /m\_txyzth123 được định cấu hình bởi một message được viết trong tệp có tên array\_path\_ls.msg. Các ma trận x, y, z là các điểm của đường thẳng trong không gian Descartes của cơ sở di động. Các ma trận th1, th2, th3 là quỹ đạo trong không gian chung của cơ cấu chấp hành. Ma trận thời gian là thang thời gian của quỹ đạo được mô phỏng.

**Node:** Node có tên Posonador\_rviz\_realtime\_tm1 phụ trách tính toán các góc của các khớp Robot Delta từ các tọa độ trong không gian Decate xyz của quỹ đạo của end-effector. Điểm xyz được sử dụng để mô phỏng chuyển động của end-effector và các góc của mỗi khớp cho chuyển động của 3 chuỗi động học.

**Publisher:** Topic Joint\_states được định cấu hình bằng một thông báo được viết trong tệp có tên JointState.msg. Đây là một thông báo chứa dữ liệu để mô tả trạng thái của một tập hợp khớp điều khiển mô-men xoắn. Mỗi khớp (xoay hoặc hình lăng trụ) được xác định duy nhất bằng tên với từng khớp, chức vụ là vị trí của khớp (rad hoặc m), vận tốc là tốc độ của khớp (rad / s hoặc m / s) và lực tác dụng lên các khớp nối (Nm hoặc N).



Kết quả mô phỏng