ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ESTUDIO, CONTROL E IMPLEMENTACIÓN DE SISTEMAS ROBÓTICOS AVANZADOS

IMPLEMENTACIÓN DE UN ROBOT PARALELO VIRTUAL EN BASE A LA PLATAFORMA QUE ESTÁ SIENDO EMPLEADA EN EL PROYECTO DE INVESTIGACIÓN GRUPAL PIGR-20-01

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y AUTOMATIZACIÓN

JONATHAN PAUL CATOTA PAREDES

jonathan.catota@epn.edu.ec

DIRECTOR: ING. PATRICIO JAVIER CRUZ DÁVALOS, Ph.D.

patricio.cruz@epn.edu.ec

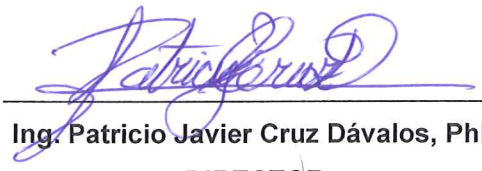
DMQ, Octubre 2022

CERTIFICACIONES

Yo, Jonathan Paul Catota Paredes declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Jonathan Paul Catota Paredes

Certifico que el presente trabajo de integración curricular fue desarrollado por Jonathan Paul Catota Paredes, bajo mi supervisión.



Ing. Patricio Javier Cruz Dávalos, PhD

DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Jonathan Paul Catota Paredes

Ing. Patricio Javier Cruz Dávalos, PhD

Ing. Jeampool Arcos

Ing. Kleber Vicente

DEDICATORIA

A toda mi familia por haberme brindado su completo apoyo dentro de mi carrera universitaria, depositando toda su confianza en mí, y en especial a mis padres Néstor y Mercedes que siempre han sido los pilares en todos los proyectos de vida que me he propuesto y que hasta el día de hoy he logrado culminar.

Jonathan Paul Catota Paredes

AGRADECIMIENTO

Agradezco a Dios por haberme dado la salud y vida necesaria para poder culminar con todos mis objetivos.

A mis padres, Néstor y Mercedes, que han sido siempre mi inspiración y mi más grande apoyo al momento de realizar todo proyecto que me he propuesto en la vida, estando siempre para mi sin negarme nunca sus sabios consejos y dándolo todo por apoyarme en todo el trayecto de mi carrera universitaria.

A mis hermanos, Diego y Alex, que a pesar de todo siempre han estado presentes dándome ánimos para poder seguir adelante y nunca dejar que me desvié de mi camino para poder lograr mi más anhelado sueño de culminar mi carrera universitaria.

A mis mejores amigos John y Edwin, que a pesar de ser amigos los considero como mis hermanos, y nunca han dudado en brindarme su ayuda o un buen consejo en los momentos que más lo necesitaba.

Dentro de la universidad agradezco a todos los docentes que supieron impartirme los conocimientos para poder avanzar y pode formarme como un profesional, y a todos los compañeros y amigos que, a pesar de las dificultades encontradas en el ámbito universitario, siempre supimos colaborar en conjunto para poder superarlas.

Agradeciendo de una manera muy especial al director de este proyecto, el Phd. Patricio Cruz, quien me permitió participar dentro de este proyecto, apoyándome y guiándome a lo largo de toda la elaboración y desarrollo de este trabajo de una manera desinteresada, permitiéndome desarrollar nuevas destrezas y poder aprovechar todo lo aprendido para mejorar mis cualidades en el ámbito profesional.

Por último, agradezco a los ingenieros Jeampool y Klever que fueron piezas fundamentales a la hora de poder desarrollar y culminar este trabajo.

Jonathan Paul Catota Paredes

ÍNDICE DE CONTENIDO

RESUMEN

El presente trabajo se enmarca dentro del Proyecto de Investigación Grupal PIGR-20-01, en el cual se ha implementado un robot paralelo enfocado a la realización del tratamiento y rehabilitación de rodilla. La problemática a solventar se centra en la necesidad de contar con un robot paralelo virtual que permita probar en el mismo, los diferentes tipos de controladores diseñados para el robot físico. Por lo cual, en este trabajo se emplean los softwares ROS2 y el simulador GAZEBO. En particular, se analizan los archivos base existentes del robot, desarrollados en ROS, para migrarlos a ROS2, y diseñar una estructura URDF que permita la implementación de las diferentes partes creadas para la conformación del robot. Además, utilizando los modelos dinámico y cinemático, se diseñan y sintonizan controladores PID para cada uno de los actuadores del robot, así como un esquema de control que unifique los mismos, para comandar la plataforma móvil a una posición deseada. Para comprobar el funcionamiento se realizan pruebas añadiendo perturbaciones en el entorno de simulación y analizando el desempeño del sistema de control de manera que permita que el robot paralelo pueda mantener o alcanzar una posición deseada.

PALABRAS CLAVE: ROS2, GAZEBO, Robot Paralelo, Controladores PID

ABSTRACT

This work is part of the research project PIGR-20-01, in which a parallel robot focused on performing knee treatment and rehabilitation has been implemented. The problem to solve is the requirement of having a virtual parallel robot that allows to test the different control architectures designed for the physical robot. To accomplish this goal, this work uses ROS2 and the GAZEBO simulator. In particular, the pre-existing base files of the robot developed in ROS are analyzed and then they are migrated to ROS2. Also, a URDF structure is designed to allow the implementation of the different parts created for the robot conformation. In addition, by using the dynamic and kinematic robot models, PID-type controllers are designed and tuned for each one of the robot's actuators. Furthermore, a control scheme that unifies these controllers is implemented to command the robot's mobile platform to reach a desired position. To validate the system, tests are carried out by adding disturbances in the simulation environment to analyze the performance of the control architecture to fulfill the goal of reaching and keeping a desired position.

KEYWORDS: ROS2, GAZEBO, Parallel Robot, PID Controller

1. INTRODUCCIÓN

En la actualidad la robótica está en continua evolución y sus necesidades cambian para hacer frente a nuevos retos. En el ámbito de la robótica para hacer frente a su continua evolución se ha desarrollado un sistema operativo dedicado a robots, denominado ROS (Robot Operating System), el cual posee un amplio campo de herramientas y librerías para la creación y manipulación de diversos modelos robóticos. Dicho sistema operativo cuenta con variadas distribuciones o versiones como ROS1 y su reciente versión ROS2. El objetivo de ROS2 es adaptarse a las modificaciones que día a día surgen en la robótica, aprovechando las mejores características de ROS1 y mejorando sus desventajas [1]. El uso de ROS2, permite implementar simulaciones que ayudan a descartar errores que se pueden obtener previo al uso de una plataforma real, permitiendo así corregirlos de manera rápida sin el riesgo de comprometer el funcionamiento de las partes mecánicas. Dicho sistema operativo también cuenta con simuladores como GAZEBO, el cual permite realizar modificaciones rápidas en el robot virtual para probar así su funcionalidad y ver si es viable realizar dicha modificación en el robot físico.

El sistema operativo aplicado a robótica ROS2 y el simulador GAZEBO, utilizan código libre por lo que son ejecutadas en el sistema operativo Linux con la versión 20.04 LTS en específico. Al ser un software libre de código abierto, este presenta ventajas como la perduración del entorno virtual desarrollado, ya que al permitir el acceso libre al código fuente del mismo, este podrá ser editado por distintos desarrolladores contribuyendo en la mejora y actualización del software del robot paralelo virtual.

El robot paralelo con el que se cuenta dentro del proyecto de investigación grupal PIGR-20-01 tiene una disposición 3UPS+1RPU y que está enfocado al apoyo en la rehabilitación de rodilla. Por este motivo es esencial que se cuente con un modelo virtual que permita simular los diferentes movimientos que este realizará. Justamente, la implementación del robot paralelo virtual permitirá verificar su funcionamiento dentro del entorno de simulación GAZEBO, este facilitará que el usuario pueda modificar dicho entorno añadiendo perturbaciones a diferentes partes del robot, corroborando que los diferentes controladores programados se encuentran funcionando de manera correcta [2]. Los controladores, sintonizados mediante el método más adecuado, serán programados en el robot virtual mediante nodos en ROS2, para así poder verificar su funcionamiento en la plataforma virtual y que estos puedan ser utilizados en el robot físico.

1.1 OBJETIVO GENERAL

Implementar un robot paralelo virtual, mediante las plataformas ROS2 y GAZEBO, para el control de posición del robot paralelo 3 UPS+RPU empleado en el Proyecto PIGR-20-01.

1.2 OBJETIVOS ESPECÍFICOS

1. Realizar una revisión bibliográfica acerca de las generalidades del robot paralelo 3UPS+RPU, empleado en el Proyecto PIGR-20-01, y los softwares de simulación de robótica ROS2 y GAZEBO.
2. Implementar diferentes Scripts para la manipulación y configuración de las distintas partes del robot paralelo 3 UPS+RPU y un Script general para la incorporación de las mismas dentro del entorno virtual.
3. Verificar que la implementación del robot en el entorno virtual funcione adecuadamente, comprobando que la estructura y disposición de todos los elementos que conforman un robot paralelo con disposición 3UPS-RPU, no se vean alterados al momento de aplicar movimientos o perturbaciones al robot realizando repetidas simulaciones.
4. Implementar un esquema de control funcional en el sistema operativo ROS2, basado en la correcta programación de nodos, que facilita la sintonización y prueba de controladores.
5. Verificar que el esquema de control implementado posicione al robot paralelo virtual acorde a las entradas de referencias dadas mediante pruebas de simulación que incluyan perturbaciones aplicadas al robot.

1.3 ALCANCE

- Se realiza un análisis bibliográfico que permita conocer el modelo cinemático y dinámico del robot paralelo 3UPS-1RPU, para así entender la disposición física y los parámetros con los cuales se podrá implementar el robot virtual, asemejando su forma al del robot físico.
- Se realiza una revisión bibliográfica que permite familiarizarse con el sistema operativo Linux 20.04 LTS, además de comprender como se ejecutan los paquetes de ROS2 y GAZEBO dentro del mismo, para realizar la programación de las partes del robot y obtener una simulación correcta del mismo.

- Se implementa un modelo virtual en 3D del robot paralelo con el uso de las plataformas ROS2 y GAZEBO, que permite simular su funcionamiento ante perturbaciones aplicadas al mismo, logrando verificar el funcionamiento del esquema de control que se diseña para dicho robot.
- Se implementan controladores tipo PID individuales para cada uno de los actuadores del robot, además de un esquema tipo cascada que unifica los mismos obteniendo un control único de posición del robot, que corre empleando el sistema operativo ROS2 y permite visualizar su funcionamiento mediante el simulador GAZEBO.
- Se realizan pruebas independientes para cada uno de los actuadores y de todo el robot virtual, junto al esquema de control, para asegurar su correcto posicionamiento y analizar su respuesta ante perturbaciones.

1.4 MARCO TEÓRICO

1.4.1 ROBOT PARALELO 3UPS+1RPU

El proceso de rehabilitación de rodilla, exige que se haga uso de un robot con al menos 4 grados de libertad para realizar los movimientos correspondientes a los ejercicios requeridos. Gracias a la estructura que presenta un robot paralelo, estos poseen una relación de carga/peso relativamente alto, lo cual se traduce en una alta precisión en el sistema al momento de que el paciente realice esfuerzos sobre la plataforma móvil. Para poder regular y controlar el sistema móvil del robot al momento de recibir dichos esfuerzos, se requiere entender la cinemática y dinámica del robot [3].

Un robot paralelo es un mecanismo de cadena cinemática cerrada en el que una base móvil está unida a su base fija por varias cadenas cinemáticas independientes. Esta configuración de cadena cinemática cerrada ofrece a los robots paralelos ciertas ventajas sobre los robots en serie, en términos de rigidez, velocidad, precisión e inercia de movimiento. La principal desventaja de los robots paralelos en comparación con los robots en serie es su pequeña área de trabajo, dado que las diversas cadenas cinemáticas cerradas que poseen, limitan de gran manera su desplazamiento [4].

Existen variadas estructuras de robots paralelos, por lo que no existe una notación específica para cada uno de ellos. De forma general estos tipos de robot se clasifican según el número de grados de libertad o el tipo de movimiento que presentan los mismos.

Una de las nomenclaturas para identificar este tipo de robots consiste en utilizar un número para indicar el número de piernas o brazos que posea el robot, seguido de letras que caracterizan el tipo de articulaciones que conforman los mismos, estas permiten las diferentes uniones que posee cada una de las piernas. Además, en ciertos tipos de robots que utilizan un brazo central o central mast, comúnmente, se escribe en primer lugar el número de piernas y luego la pierna central, separándolas mediante el uso de un guion.

El robot paralelo en el que se centra este proyecto de integración curricular, como se menciona, es el de estructura 3UPS-1RPU, mismo que posee 4 grados de libertad para realizar los movimientos requeridos al momento de realizar sus funciones en el área de rehabilitación de rodilla donde pretende usarse dicho robot, este forma parte del Proyecto de Investigación PIGR-20-01. Haciendo uso de la nomenclatura explicada anteriormente para el robot en específico 3UPS-1RPU, se puede describir que se tiene un robot de 3 piernas con articulaciones universal, prismática y esférica; y, con una pata central con articulaciones del tipo rotacional, prismática y universal [5], como se muestra en la Figura 1.1.

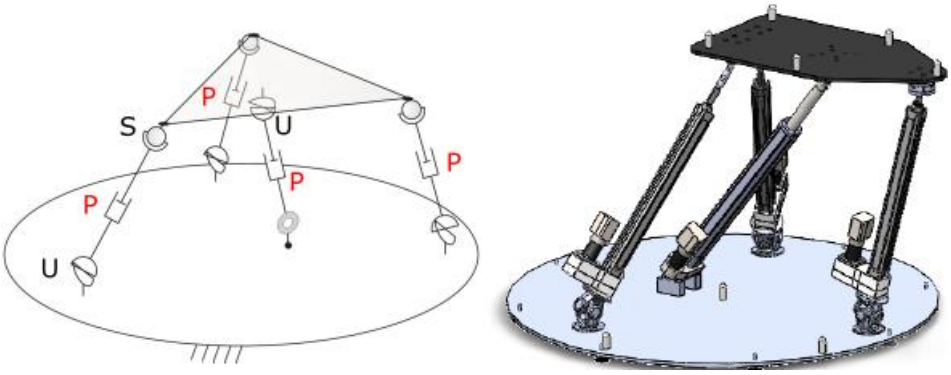


Figura 1.1 Robot Paralelo 3UPS+1RPU

La distribución de las patas del robot tanto en la base fija, así como en la móvil, es un parámetro a considerarse dentro de la obtención de las matrices dinámicas así como de las matrices de inercia, esto para poder realizar la sintonización de los correspondientes controladores. A continuación, en la Figura 1.2, se puede apreciar la distribución utilizada por el robot paralelo 3UPS-1RPU en el actual proyecto de integración curricular.

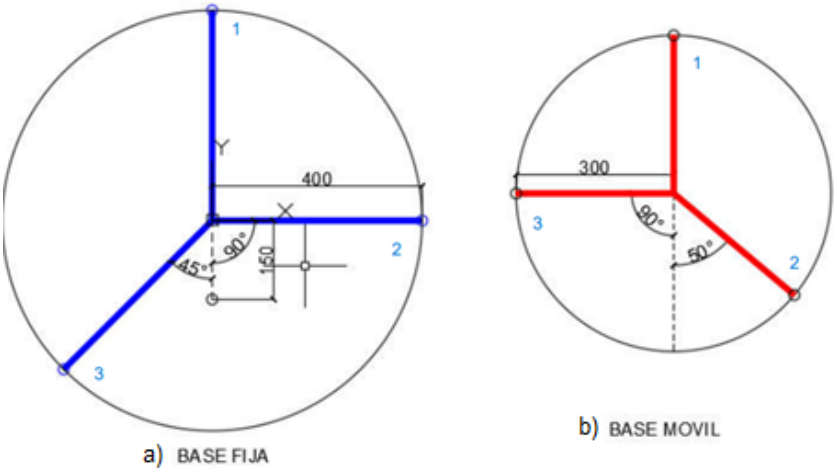


Figura 1.2 Distribución de las extremidades del robot tanto en a) base fija y b) base móvil.

La distribución en la Figura 1.2, describe los radios desde el centro de la plataforma hacia los puntos de conexión de las diferentes extremidades del robot, para la base fija las 3 extremidades se ubicaran a un radio de 400 mm desde el centro mientras que el central mast o pata central se ubica a 150 mm a partir del centro y para la base móvil las 3 extremidades se ubican en un radio de 300 mm a partir del centro de la plataforma mientras que la pata central se ubicara en el centro, también se puede apreciar los ángulos con los que van a ser ubicadas dichas extremidades en ambas plataformas.

Las diferentes distribuciones en las cuales se podría ubicar las extremidades del robot tanto en la plataforma móvil como en la fija, deben ser analizadas con anterioridad, ya que una mala distribución de las mismas podría dar como resultado el surgimiento de un sin numero de singularidades, que limitarían aun mas el movimiento del robot.

1.4.1.1 Articulaciones

Dentro de un robot paralelo se pueden encontrar de manera común el siguiente tipo de articulaciones [5]:

Articulación Prismática (P-Prismatic): Articulación con 1 grado de la libertad, se encuentran formadas por dos uniones anidadas que se desplazan hacia dentro y a lo largo de cada una, uno de los ejemplos más básicos y utilizados de este tipo de articulación, son los cilindros acompañados de su vástago, mismo que se desplaza hacia dentro y fuera del cilindro. Para el actual trabajo esta articulación es de vital importancia para proveer de movimiento al robot.

Articulación rotacional (R-Rotational): Articulación con 1 grado de libertad, estas permiten que la unión gire alrededor de un único eje, asemejándose al movimiento que realiza una puerta alrededor de su bisagra.

Articulación esférica (S-Spherica): Articulación que posee 3 grados de libertad, esta permite el movimiento de alrededor de 3 ejes, permitiendo un sin número de combinaciones de giro, en exactamente 3 direcciones que se encuentras perpendiculares al espacio.

Articulación universal (U- Universal): Articulación de 2 Grados de libertad.

Articulación cilíndrica (C- Cylindrical): Articulación con 2 grados de libertad, donde el primero permite la acción de traslación y el segundo permite la acción de rotación.

Para el robot paralelo con distribución 3UPS-1RPU, como se mencionó anteriormente, se hace uso de 4 articulaciones de las más comunes que son la Prismática, Rotacional, Esférica y Universal, donde la articulación principal y la cual será utilizada para realizar el control y posicionamiento de la plataforma móvil, será la prismática o también denominada articulación activa. Debido a que este tipo de articulaciones presentan un movimiento lineal, serán utilizadas mediante actuadores o motores colocados en cada una de las patas del robot, para poder ejecutar el movimiento de los vástagos y así poder posicionar la plataforma móvil. Las demás articulaciones que son la esférica, rotacional y universal, serán denominadas articulaciones pasivas, debido a que los movimientos que estas realicen dependerán de los movimientos generados por las articulaciones activas. En la Figura 1.3, se puede observar las articulaciones a ser utilizadas por el robot 3UPS-1RPU.

Figura 1.3 Articulación Prismática (a), Articulación rotacional (b), Articulación esférica (c), Articulación Universal (d)

1.4.1.2 Actuadores

Los actuadores a utilizar en el robot paralelo 3UPS-1RPU, son motores del tipo lineal, los cuales serán ubicados en cada una de las patas del robot, estos impulsaran los diferentes ejes o vástagos de los cilindros que conforman dichas patas, para poder posicionar la

plataforma móvil. El cilindro que conforma cada una de las patas del robot, así como su respectivo actuador se puede apreciar en la Figura 1.4.

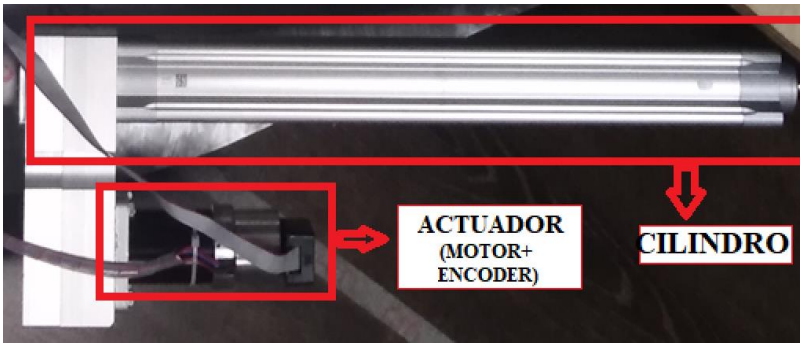


Figura 1.4 Cilindro Festo ESBF BS-32-100-10P y Motor Maxon 148877

Como se puede apreciar en la Figura 1.4, el actuador utilizado para las patas del robot, se conforma de un motor lineal en conjunto con un encoder que facilita el control de posición del vástago, mediante el tratamiento de los datos enviados por el encoder.

Para realizar la implementación virtual de los componentes descritos anteriormente, así como de las diferentes partes que conforman el robot, como articulaciones o actuadores, se debe contar con diferentes softwares que permitan la incorporación de todo el conjunto de elementos, además de poder simular en un entorno virtual las diferentes acciones de control y movimiento del mismo.

Dentro de la implementación virtual de cada una de las piezas que conforman el robot, se debe incluir las diferentes características dinámicas que predominan a sus piezas, para que de esta manera nuestra planta o robot simulado, nos permita obtener resultados de control similares a los que se obtienen mediante el robot físico real.

1.4.2 SOFTWARES DE IMPLEMENTACIÓN

1.4.2.1 ROS2-Galactic Geochelone

El mundo de la Robótica va formando parte de la vida cotidiana, por lo que se encuentra día a día en una constante evolución, lo que le permite ir mejorando determinadas características con el transcurso del tiempo, por ello se ha desarrollado un sistema operativo dedicado a robots, el cual se denomina ROS (Robotic Operative Systems). Este sistema posee varias distribuciones y actualizaciones, en el actual proyecto de integración curricular se hará uso de la distribución Galactic Gouchelone, misma que es parte del sistema operativo actualizado ROS2.

El sistema ROS2 será utilizado dentro o en conjunto con el sistema operativo LINUX en su versión 20.04 LTS, este se trata de un derivado de UNIX que utiliza una libre

distribución, y el lenguaje principal de programación de este sistema es el denominado lenguaje C. Tener un previo conocimiento de este tipo de lenguaje, así como su variante C++, es uno de los requisitos para el entendimiento de los scripts desarrollados para el manejo del robot paralelo 3UPS+1RPU. Este sistema operativo ha ido creciendo rápidamente gracias a su característica de ser un software de código abierto, lo cual permite que un sin número de desarrolladores puedan contribuir a la mejora y actualización del sistema, creando y mejorando diversos componentes del sistema [6], así como también permite la perduración de programas o archivos generados en el mismo. Esta característica permite que los archivos generados para el desarrollo del robot puedan ir siendo mejorados y actualizados por diferentes desarrolladores, y así obtener un mejor modelo virtual del mismo.

ROS2 brinda la capacidad a los usuarios de poder implementar, desarrollar y hacer uso de un sin número de prototipos de robots, gracias a su conjunto de bibliotecas y un sin número de herramientas, como controladores y algoritmos de desarrollo, permitiendo de esta manera establecer la interacción entre el usuario y un mundo virtual donde se desenvuelve el robot diseñado. De esta manera se puede obtener un aprendizaje tanto teórico como práctico, en el mundo de la robótica, el entorno proporcionado por ROS2, brinda una infinidad de recursos que optimizan los parámetros de desarrollo y prueba de las características de un robot [7].

El entorno como tal, permite obtener una simulación confiable del robot que se acomoda físicamente a todas las características del mismo, permitiendo incluir sobre este influencias externas o perturbaciones, que puedan afectar a las diferentes partes que conforman el robot. Tanto el diseño como la manipulación del robot se logran en ROS2 utilizando diferentes algoritmos, que se pueden programar en diferentes lenguajes como Python o C++ para implementar una aplicación en un lenguaje común para varios desarrolladores que pueden contribuir con la mejora del modelo. ROS2 puede trabajar en conjunto con simuladores como GAZEBO o RVIZ2 para proporcionar una representación 3D precisa del robot [2]. Aunque se debe tomar en consideración que ambos simuladores no ofrecen las mismas características de simulación, dado que el resultado esperado es obtener un modelo virtual que asemeje al robot físico, el uso de GAZEBO es el mas recomendado debido a que permite simular diversas características dinámicas, así como un sin numero de factores o perturbaciones que afectan al sistema.

ROS2 basa su arquitectura en grafos. Un grafo consiste en una red de elementos que comparten una serie de datos o procesos en un mismo instante, donde dichos procesos se realizan mediante el uso de nodos. Es recomendable que cada uno de los procesos o

actividades que realice el robot, se los implemente programando nodos independientes, estos permiten la interacción entre el modelo robótico con el usuario. Los nodos pueden recibir o enviar diferentes mensajes mediante el uso de tópicos, acciones o servicios [7] [8]. En la Figura 1.5 se puede observar el esquema de funcionamiento del uso de nodos en ROS2.

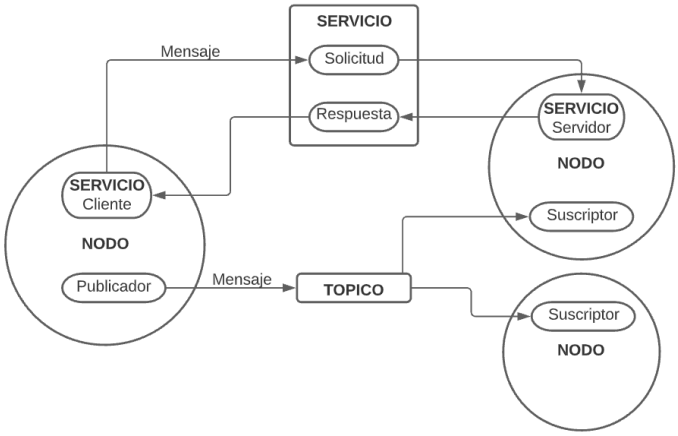


Figura 1.5 Esquema de funcionamiento de ROS2 [9].

Los conceptos más generales e importantes dentro del uso de ROS y sus diferentes actualizaciones como ROS2 son los siguientes [8]:

• **Nodos:** Procesos principales en los cuales se realizan variados cálculos. Cada nodo es responsable de una tarea en específico del robot, como por ejemplo el movimiento del robot puede ocupar un nodo, mientras que otro se ocupa de la lectura y retroalimentación de los diferentes sensores disponibles, entre otros procesos. Este sistema de nodos permite obtener una buena modularidad logrando así que ROS obtenga un diseño más eficiente.

• **Mensajes:** Estructura del tipo de datos, con la cual se codifica la comunicación entre los nodos y define como se deben incorporar los datos. Una de las estructuras más comunes para integrar este tipo de datos es el uso de componentes escritos en lenguaje C. Dentro de ROS se puede hacer uso de un sin número de mensajes preestablecidos, así como la creación de mensajes personalizados que cuenten con una estructura en específico para el establecimiento de la comunicación.

• **Tópicos:** Los nodos se enrutan o enlazan a través de un sistema de publicador/suscriptor, un ejemplo fácil de entender es una red social como YouTube donde el publicador es quien sube un video y suscriptor quien lo observa asimilando el contenido. Los tópicos actúan como un bus de datos o cables de interconexión, por los cuales se realiza el intercambio de los diferentes tipos de mensajes. Cada uno de los

nodos configurado como publicador, ubicará mensajes en un tópico determinado, el nombre con el cual se designe dicho tópico será utilizado por el o los nodos suscriptores, para acceder a la información de dicho mensaje. Un único tópico puede conllevar un sin número de publicadores y suscriptores.

• **Servicios:** El uso de servicios en ROS, se destina para hacer uso de un método diferente de comunicación entre la estructura de nodos. Los servicios poseen un modelo basado en llamada y respuesta, a diferencia de los tópicos que permiten una suscripción a un flujo de datos que se actualiza continuamente, los servicios brindaran información únicamente cuando se realice una llamada especifica a dichos datos [7].

1.4.2.2 GAZEBO

Para el actual trabajo de integración curricular se hace uso del simulador GAZEBO, debido a sus grandes ventajas con respecto a la simulación de propiedades dinámicas, en conjunto con ROS2 en su versión “Galactic Gouchelone”. Dicho simulador es una de las herramientas que permite la implementación de un modelo en 3D con características dinámicas propias del robot físico a utilizar. Una de las ventajas principales del simulador GAZEBO es que permite colocar al modelo en 3D del robot, en diferentes entornos simulados que posean características de interior o exterior, donde se puede añadir variados parámetros como el efecto de la gravedad, fuerzas y torques aplicados a distintas partes del robot entre otras variadas perturbaciones que afectaran a las diferentes piezas que conforman el robot [8].

Mediante la incorporación al simulador GAZEBO de los archivos URDF, que brindan la descripción física del robot, dividiéndolos en un conjunto de eslabones (links), articulaciones (Joints), colisiones (Collision) y transmisiones (Transmission), permitirá incorporar una descripción tanto a nivel cinemático como dinámico del robot, obteniendo una simulación realista y confiable del comportamiento del mismo. A más de ello el simulador permite la incorporación de una serie de sensores y actuadores, que permitirán el monitoreo y control del robot.

Gazebo cuenta con un modo de edición de modelo basico (Model Editor), el cual permite realizar modificaciones directas en el modelo virtual del robot, haciendo uso directamente del simulador como tal, evitando así la modificación de código dentro de los scripts o URDFs mencionados anteriormente. Pero es recomendable tener conocimiento de cómo se estructura el archivo URDF, ya que en este se pueden incluir las diferentes características dinámicas que debería tener el robot para que el grado de fidelidad de la simulación aumente [8].

Entre las principales características que dispone el simulador GAZEBO se tiene las siguientes[10]:

- Software libre, lo que implica que varios desarrolladores puedan modificar, ampliar y actualizar el entorno.
- Se trata de un simulador compatible con ROS, lo que permite hacer uso de APIs de comunicación para transmitir y recibir información entre ambas plataformas, permitiendo la manipulación de los diferentes tipos de robots implementados con mayor facilidad.
- Implementa simulaciones fiables y realistas del robot, ya que el simulador puede implementar varias características dinámicas del robot, permitiéndolo interactuar de manera interna y externa con el entorno creado.
- Permite hacer uso de archivos URDF, lo cual facilita la implementación de robots propios del usuario.
- Gazebo brinda la posibilidad de implementar y modificar el mundo completo donde va a interactuar el robot, permitiendo así probar el desenvolvimiento del robot como si lo hiciera en el mundo real.
- El simulador cuenta con un sin número de plugin que permite colocar diversos sensores al modelo del robot implementado, para así poder desarrollar controladores que aumenten la eficiencia de manipulación sobre el mismo.

Las versiones de los softwares a utilizar para este trabajo de integración curricular, se los pueden identificar por sus logos mostrados en la Figura 1.6.

(a) (b) (c)

Figura 1.6 Softwares: (a) Linux-Ubuntu 20.04, (b) ROS2 Galactic Geochelone, (c) Simulador Gazebo.

1.4.3 SISTEMA DE CONTROL

Un sistema de control es aquel que se encuentra conformado por un conjunto de varios dispositivos, estos pueden ser de diversos tipos como eléctricos, mecánicos, etc. Para la correcta implementación de un sistema de control robótico, sea real o simulado, hace falta la programación correcta del controlador teniendo en cuenta que la ley de control a aplicar se establezca de manera correcta, pero también existe la posibilidad de poder programar una serie de instrucciones que deberá seguir el robot para cumplir con sus tareas correspondientes [11].

Para poder realizar la implementación de un sistema de control para un robot paralelo, se pueden considerar 3 puntos importantes:

- Conocer la estructura del sistema a controlar.
- Obtener el modelado dinámico del robot.
- Obtener las diferentes leyes de control

El objetivo principal de este sistema es poder obtener un control sobre el posicionamiento de partes en específico del sistema, para este caso en particular, el sistema se trata del modelo virtual del robot paralelo. Para el cual se deberá obtener controladores del tipo más adecuado que permita posicionar a los diferentes ejes de los cilindros en una posición deseada, mediante la acción de los actuadores sobre las distintas articulaciones prismáticas [11].

Los principales elementos dentro de un sistema de control son:

Sensores: Dispositivos colocados en el sistema o para este caso específico en el robot, para poder captar información del entorno en el que se encuentra, el robot a utilizar cuenta con sensores denominados Encoders los cuales otorgan datos que al ser tratados computacionalmente podrán traducirse a la posición en la que se encuentra el eje del cilindro o extremidad del robot.

Controlador: Se trata de un dispositivo electrónico o mecanismo, que entrega las acciones de control necesarias para poder manejar los distintos actuadores ubicados en el robot. Este actúa con respecto a los valores de entrada o referencias de posición dada y las lecturas del sensor.

Actuador: El actuador es el elemento encargado de realizar la acción mecánica dentro del sistema o robot.

Dentro del ámbito de robótica, los sistemas de control se componen de varios sistemas electrónicos que permiten controlar las acciones que va a realizar el robot, pero para el actual trabajo de integración curricular todo el modelo robótico y entorno en el cual se va a desenvolver va a ser simulado mediante el uso de GAZEBO. Por ello los datos de retroalimentación que se recibirán, serán proporcionados por diferentes plugin que asemejan el comportamiento de los diferentes sensores, estos deberán ser programados dentro del entorno para poder recibir sus datos y así poder ejecutar las diferentes acciones de control establecidas para obtener un posicionamiento adecuado del robot

1.4.3.1 Control de posición

Teniendo en consideración el modelo dinámico del robot a controlar, de n gados de libertad, se debe implementar un sistema que ayude al robot a llegar a una posición articular deseada (q_d), misma que se considerara constante. Para este objetivo se debe tratar de encontrar una función vectorial (ζ), que permita que las posiciones q asociadas a las coordenadas articulares del robot puedan llegar hacia la q_d. Dicha función vectorial, es conocida en el ámbito de control como “Ley de control”. Para fines prácticos es recomendable que el controlador no depende de la aceleración articular $\ddot{q}$ [12]. En la Figura 1.7 se puede observar un control a lazo cerrado común en un robot.

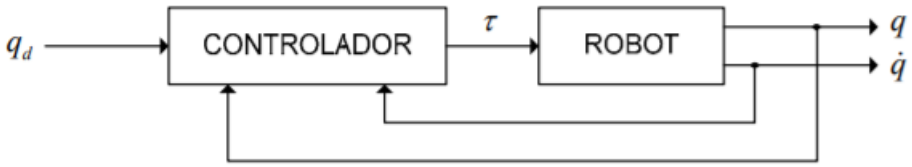


Figura 1.7 Control en lazo cerrado para un robot [12].

Existen controladores como lo son los del tipo PID con parámetros numéricos o de sintonización que se determinan en función del modelo del robot que se desea controlar, por lo que se dice que un control adecuado del robot debe basarse en las características físicas del mismo. Debido a las cadenas cinemáticas cerradas que posee el robot paralelo, su control se dificulta, debido a que las conexiones finales de cada una de las patas hacia la plataforma móvil, limita y afecta al movimiento de las uniones o articulaciones prismáticas que brindan de movimiento a la plataforma.

1.4.3.2 Control PID

En aplicaciones prácticas, debido a su facilidad de implementación y alta eficiencia en varios campos, es probable que encuentre un controlador PID ya que es el más utilizado en diversas industrias. PID proviene del control proporcional, integral y derivado. Una de

las razones por las que estos controladores son tan populares es que han tenido éxito en diferentes tipos de aplicaciones, como sistemas mecánicos, sistemas neumáticos, electrónica analógica, electrónica digital y procesos químicos. Además, el controlador PID es relativamente simple y no requiere un modelo matemático explícito, dado que la sintonización de sus parámetros puede ayudar a establecer la respuesta de salida para obtener un control eficiente, sin embargo, abarca un campo bastante amplio y tiene muchas variaciones [13].

Para el actual proyecto de integración curricular se hará uso de controladores PID individuales para cada una de las extremidades del robot. Esto permitirá posicionar de manera independiente cada uno de los ejes que impulsaran la plataforma móvil del mismo, para luego implementar un esquema que unifique dichos controladores individuales, con el objetivo de poder comandar la posición de la plataforma móvil de manera conjunta. En la Figura 1.8, se puede observar el esquema de control individual que utilizara cada una de las extremidades del robot.

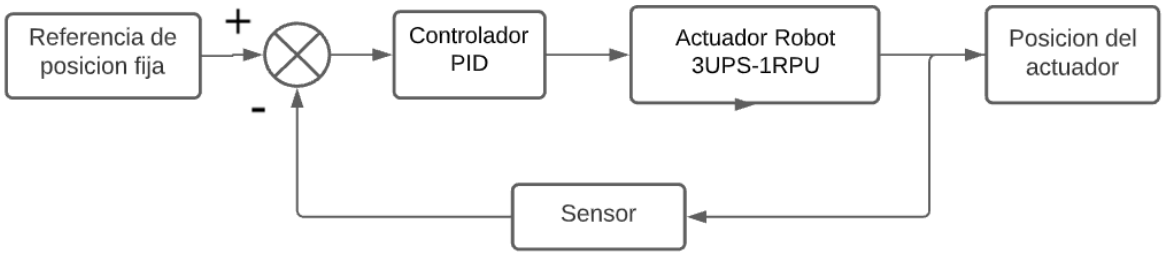


Figura 1.8 Control PID individual

Existen diversas combinaciones para hacer uso de un controlado PID, como pueden ser las disposiciones **P**, **PI**, **PD**, asignando las constantes a cero. Una vez que se haya implementado dentro del entorno ROS2-GAZEBO el esquema de control individual para cada actuador, se podrá sintonizar las constantes mencionadas anteriormente.

1.4.3.3 Control en Cascada

El esquema de control a utilizar para anidar los diferentes controladores PID individuales para las extremidades del robot paralelo 3UPS-1RPU, es el tipo cascada. Esta configuración se basa en sistemas de control realimentados uno dentro del otro, ayudando a eliminar las diferentes perturbaciones que pueden ingresar al sistema o robot [14].

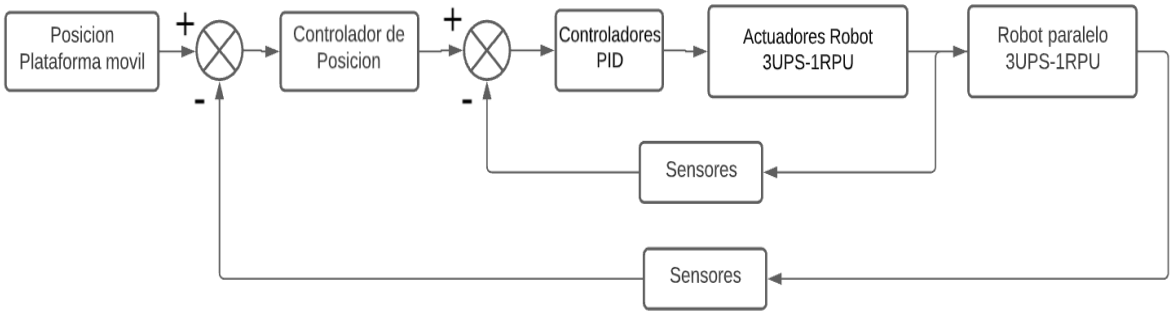


Figura 1.9 Control en Cascada Robot paralelo 3UPS-1RPU

El control en cascada mostrado en la Figura 1.9, muestra como al sistema ingresa una referencia de posición fija de la plataforma móvil. En base a esta, entrará en acción el controlador principal, que comandará directamente sobre el conjunto de controladores PID para poder manejar de manera conjunta los actuadores del robot, y así poder posicionar a la plataforma en la posición deseada.

Una vez descritas las características generales de funcionamiento del robot paralelo 3UPS-1RPU y comprendiendo los conceptos generales de los softwares a utilizar en el presente proyecto, para la implementación de un modelo virtual del mismo, se presenta a continuación, el desarrollo de la metodología necesaria para comprender la estructura dinámica y cinemática del robot. Además, se detallan la instalación, configuraciones y programación de archivos que se requiere a nivel de software para poder poner en marcha un modelo que asemeje las características del robot real, pero de manera virtual mediante el uso de simuladores. También se realizará la descripción de los archivos y paquetes de compilación requeridos para poder establecer o implementar controladores basados en las características que brinda el sistema ROS2, para que de esta manera se permita al usuario interactuar y manipular al robot.

2. METODOLOGÍA

2.1 MODELADO DE UN ROBOT PARALELO 3UPS+1RPU

2.1.1 CINEMÁTICA DEL ROBOT PARALELO 3UPS-1RPU

Dentro del estudio de la cinemática de un robot se presentan dos tipos de casos, la cinemática directa y la inversa. En el estudio de la cinemática directa, se obtienen parámetros derivados de la geometría del robot, donde las variables conocidas son las coordenadas articulares, enfocándose en encontrar la dirección y posición de la plataforma móvil con respecto al sistema de coordenadas. Mientras que, en la cinemática

inversa, se conoce la posición de la plataforma móvil y su objetivo se basa en encontrar las coordenadas articulares [15].

Para el robot paralelo 3UPS+1RPU, el cual permite realizar dos desplazamientos, así como dos rotaciones, posee un modelado cinemático realizado en [16], el cual se utilizará para el actual trabajo de integración curricular, mismo que fue obtenido mediante el método de Denavit Hartenberg. El modelo posee dos sistemas de referencia correspondientes a la base $\{O_F - X_F Y_F Z_F\}$, y el otro a la plataforma móvil $\{O_M - X_M Y_M Z_M\}$, cómo se puede observar en la Figura 2.1, haciendo uso del sistema de referencia fijo se podrá obtener los vértices de las extremidades del robot [3].

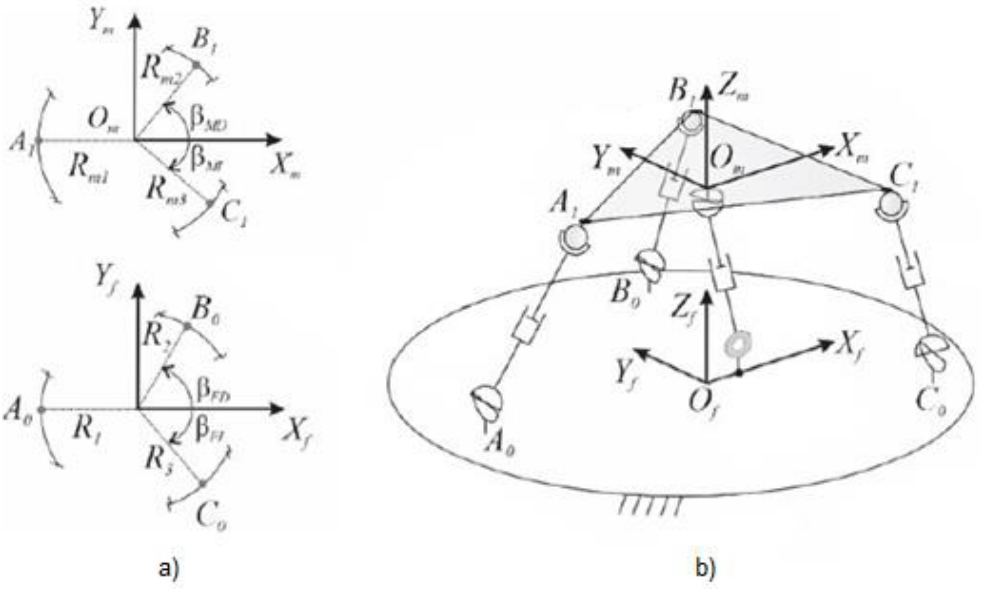


Figura 2.1 a) Posición de los vértices en las extremidades, b) Sistemas de referencia utilizados [14].

A continuación, se deberán ubicar los sistemas de referencia correspondientes a cada una de las patas, para así poder obtener los parámetros de Denavit-Hartenberg (DH), para cada articulación que permite la unión entre dos brazos. Estos valores definirán la relación existente entre la referencia de la barra $\{O_i - X_i\, Y_i Z_i\}$ y la referencia de la barra a la que se encuentra conectada $\{O_{i-1} - X_{i-1} Y_{i-1} Z_{i-1}\}$. Luego siguiendo los pasos del proceso de DH se calcularán y tabularán los parámetros obtenidos del sistema obteniendo los siguientes resultados para el robot 3UPS-1RPU [3].

Tabla 2.1 Parámetros DH de las patas UPS [3].

Barra i-esima	$\boldsymbol{\alpha_i}$	$\boldsymbol{a_i}$	$\boldsymbol{d_i}$	$\boldsymbol{\theta_i}$
1	$-\frac{\pi}{2}$	0	0	q_1

2	$\frac{\pi}{2}$	0	0	q_2
3	0	0	q_3	0
4	$-\frac{\pi}{2}$	0	0	q_4
5	$-\frac{\pi}{2}$	0	0	q_5
6	$-\frac{\pi}{2}$	0	0	q_6

Tabla 2.2 Parámetros DH para la pata central RPU [3].

Barra i-esima	$\boldsymbol{\alpha_i}$	$\boldsymbol{a_i}$	$\boldsymbol{d_i}$	$\boldsymbol{\theta_i}$
1	$-\frac{\pi}{2}$	0	0	q_1
2	$-\frac{\pi}{2}$	0	q_2	π
3	$\frac{\pi}{2}$	0	0	q_3
4	0	0	0	q_4

Gracias al uso de los parámetros obtenidos mediante DH se puede obtener la matriz homogénea ($H_{i1,i}$), mostrada en (2.1) [3].

$$H_{i1,i} = \begin{bmatrix} \cos(\theta_i) & -sen(\theta_i)\cos(\alpha_i) & sen(\theta_i)\cos(\alpha_i) & r_i\cos(\theta_i) \\ sen(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\cos(\alpha_i) & r_i sen(\theta_i) \\ 0 & sen(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.1)$$

A continuación, se debe definir la matriz de rotación de Euler correspondiente a la plataforma móvil del robot (fR_m), misma que se expresa en ángulos de Tait-Bryan [17], en (2.2). Fijando ángulos de inclinación iguales a cero en cuerpos que se encuentren ubicados de manera horizontal.

$$fR_m = R_x(\phi)R_y(\theta)R_z(\psi)$$

$$fR_m = \begin{bmatrix} \cos\theta\cos\psi & -\cos\theta\sin\psi & \sin\theta \\ \sin\psi & \cos\psi & 0 \\ -\sin\theta\cos\psi & \sin\theta\sin\psi & \cos\theta \end{bmatrix} \quad (2.2)$$

Utilizando la matriz homogénea (2.1) y la matriz de rotación (2.2), con respecto a las coordenadas cartesianas y angulares de la plataforma móvil $[X_m,\ Y_m,\ Z_m,\ \phi,\ \theta,\ \psi]$ se pueden calcular las coordenadas q_{i1}, q_{i2}, q_{i3} con $i = 1,2,3$ para las patas UPS y

q_{i1}, q_{i2} con $i = 4$ para la pata central RPU. Dichas coordenadas pueden ser determinadas al resolver el sistema que se define en función de los lazos o cadenas cinemáticas [3], que se pueden observar en la Figura 2.2.

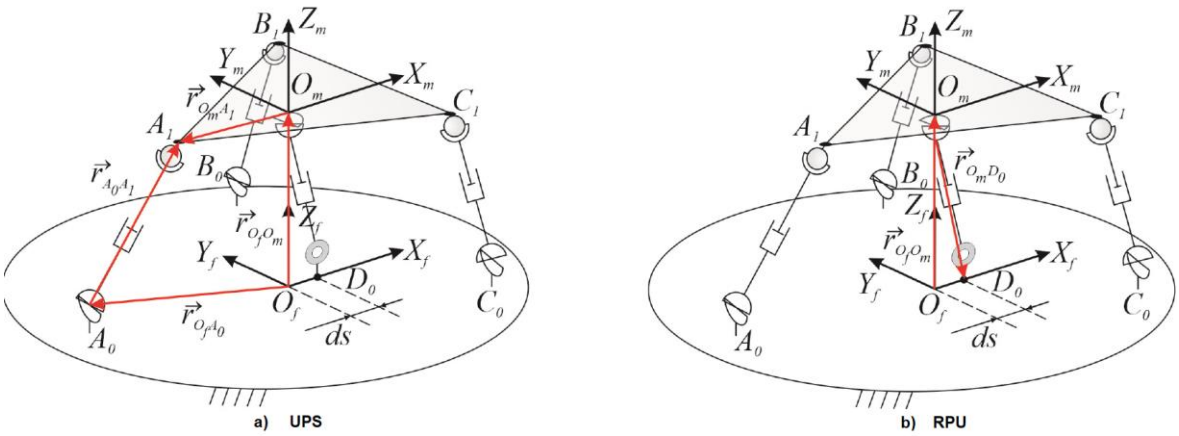


Figura 2.2 Lazos cinemáticos en las patas a) y b)[3].

De esta manera se puede realizar el cálculo de las coordenadas en los puntos A1, B1 y C1, mostradas en la Figura 2.2.

Coordenadas en A1:

$$\begin{bmatrix} X_m \\ Y_m \\ Z_m \end{bmatrix} + fR_m \begin{bmatrix} -R_m \\ 0 \\ 0 \end{bmatrix} = [H_{FP1} H_{01}(q_1) H_{12}(q_2) H_{23}(q_3)] \qquad (2.3)$$

Coordenadas en B1:

$$\begin{bmatrix} X_m \\ Y_m \\ Z_m \end{bmatrix} + fR_m \begin{bmatrix} R_m \cos(\beta) \\ R_m sen(\beta) \\ 0 \end{bmatrix} = [H_{FP2} H_{01}(q_1) H_{12}(q_2) H_{23}(q_3)] \qquad (2.4)$$

Coordenadas en C1:

$$\begin{bmatrix} X_m \\ Y_m \\ Z_m \end{bmatrix} + fR_m \begin{bmatrix} R_m \cos(\beta) \\ R_m sen(\beta) \\ 0 \end{bmatrix} = [H_{FP3} H_{01}(q_1) H_{12}(q_2) H_{23}(q_3)] \qquad (2.5)$$

Mientras que para la pata central se hará uso de la siguiente expresión:

$$\begin{bmatrix} X_m \\ Y_m \\ Z_m \end{bmatrix} = [H_{FP4} H_{01}(q_1) H_{12}(q_2) \qquad (2.6)$$

2.1.2 MODELO DINÁMICO DEL ROBOT PARALELO 3UPS-1RPU

El análisis dinámico también estudia el movimiento del robot, pero tomando en cuenta las fuerzas que lo producen. De la misma manera que en la cinemática la dinámica también puede ser directa o inversa, para el caso de la dinámica directa, se trata de determinar la respuesta del robot respecto a fuerzas aplicadas en las articulaciones, dicho de otra manera, se determina la posición del robot como función del tiempo conociéndose las fuerzas que se le aplican a sus articulaciones [15]. Mientras que la dinámica inversa consiste en encontrar la fuerza que requiere generar los actuadores para poder generar el movimiento deseado de la plataforma móvil [3].

Al igual que en el caso anterior se hará uso del modelado realizado en [16], donde se analizará las fuerzas y los momentos que afectan al robot a fin de obtener las fuerzas o acciones de control que se deberán aplicar a los actuadores.

Para los cálculos dinámicos intervendrán tanto la plataforma móvil, así como las 4 extremidades del robot, estas se dividirán en cilindro y vástago, como se observa en la Figura 2.3. Las propiedades dinámicas han sido obtenidas mediante el uso del software SolidWorks [3].

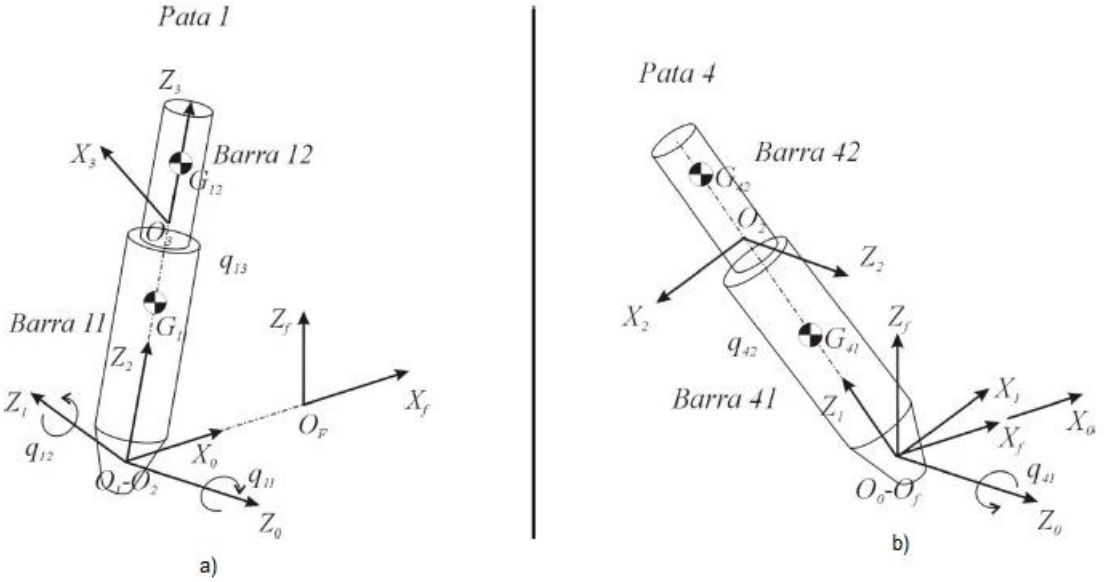


Figura 2.3 Modelado: a) Patas 1,2,3 b) Pata 4 [3].

Para continuar con la obtención del modelo dinámico del sistema se debe obtener las velocidades mostradas en (2.7) y aceleraciones mostradas en (2.8), de cada una de las barras en su centro de masa, dado que en este punto se aplican las fuerzas. Las velocidades y aceleraciones para la plataforma móvil se obtienen a partir de las derivadas

temporales de los ángulos de Euler, para verificar el procedimiento completo para obtener dichas matrices, se puede revisar lo explicado en [3].

$$\vec{\omega}_m = \begin{bmatrix} -\sin\Psi \cdot \dot{\Theta} \\ \cos\Psi \cdot \dot{\Theta} \\ \dot{\Psi} \end{bmatrix} \quad (2.7)$$

$$\vec{\alpha}_m = \begin{bmatrix} -\sin\Psi \cdot \ddot{\Theta} - \cos\Psi \cdot \dot{\Psi} \cdot \dot{\Theta} \\ \cos\Psi \cdot \ddot{\Theta} - \cos\Psi \cdot \dot{\Psi} \cdot \dot{\Theta} \\ \ddot{\Psi} \end{bmatrix} \quad (2.8)$$

La representación matemática general de la dinámica se puede expresar mediante la ecuación (2.9) [5].

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = f \quad (2.9)$$

Donde:

$M(q)$: Matriz de inercia

q: Vector de desplazamiento de las juntas

$C(q,\dot{q})$: Matriz de aceleraciones centrifugas y de Coriolis

$g(q)$: Vector de aceleraciones gravitacionales

f: Vector de fuerzas aplicadas en las juntas

2.2 USO DE LAS PLATAFORMAS ROS2-GAZEBO

Como se ha mencionado anteriormente, ROS2 se trata de un sistema operativo dedicado a robots. Este en conjunto con la plataforma de simulación GAZEBO, ayudan a realizar el modelo 3D del robot paralelo 3UPS-1RPU. Por ello se describirá como realizar la implementación del robot dentro de los softwares mencionados.

2.2.1 IMPLEMENTACIÓN DEL MODELO EN ROS2-GAZEBO

Para poder realizar la implementación del modelo o robot paralelo con disposición 3UPS-1RPU, se debe contar con los softwares ROS2, con la distribución Galáctica Gouchelone, y el simulador GAZEBO11 instalados de manera correcta en el sistema operativo Linux Ubuntu LTS 20.04. El procedimiento para realizar la correcta instalación de dichos softwares, se puede revisar en el ANEXO I.

En esta etapa de implementación se describirán los pasos a realizar en el sistema ROS2 para poder implementar el diseño del robot, además de poder cargarlo al entorno virtual GAZEBO.

Primero se debe crear un espacio de trabajo, con una carpeta directorio, en el cual se disponga de todos los paquetes requeridos, para obtener la incorporación y manejo del robot. Para ello en un nuevo terminal se deberá ejecutar los siguientes comandos.

```
mkdir dev_ws/src
```

Con este comando se crea una carpeta de trabajo denominada “dev_ws”, misma que contendrá la carpeta directorio mencionada anteriormente denominada “src”, donde se ubicaran todos los paquetes requeridos para el funcionamiento y puesta en marcha del robot.

Una vez creadas estas carpetas se debe continuar con la creación del paquete principal, que en este caso se denominará “parallel_robot”, haciendo uso del siguiente comando:

```
ros2 pkg create –build-type ament_cmake <package name>
```

En el comando anterior se debe reemplazar la parte de <package name> por “parallel_robot”, esto debido a que el nombre del paquete se incluirá como ruta para los archivos de ejecución en los diferentes scripts. Por este motivo en el actual proyecto se hará uso de este nombre para el paquete principal.

Una vez creado el paquete, mismo que posee una distribución del tipo Cmake, la cual permite hacer uso del lenguaje C++ como principal, se creara dentro del mismo las carpetas necesarias para almacenar los archivos de configuración, lanzamiento, entre otros, para poder modelar y cargar el modelo en 3D del robot paralelo, dentro del mundo de GAZEBO.

```
mkdir config launch meshes urdf control
```

Cuando se tengan listas las carpetas requeridas para los archivos de configuración y lanzamiento del modelo se podrá observar una pantalla como se puede apreciar en la Figura 2.4 Paquete principal del Robot paralelo, misma que corresponde al contenido del paquete principal “parallel_robot” que se ubica dentro del espacio de trabajo mencionado en secciones anteriores, los contenidos o archivos ubicados dentro de cada una de las carpetas, que son requeridos para la correcta ejecución del robot se presentaran y describirán en secciones posteriores.

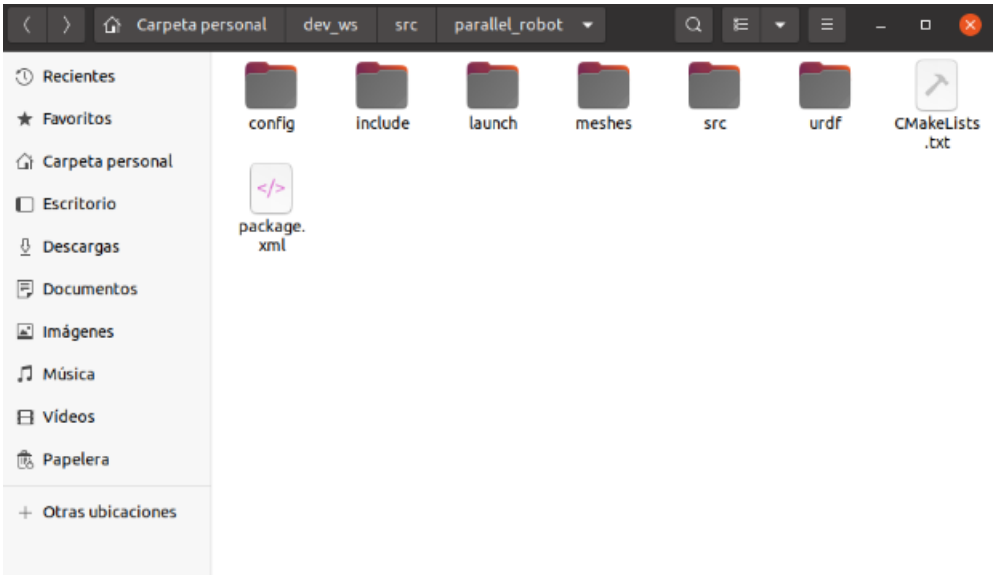


Figura 2.4 Paquete principal del Robot paralelo

2.2.2 DESCRIPCIÓN DEL ARCHIVO URDF

Dentro de la capeta URDF, se ubican los archivos URDF, mismos que realiza la descripción del robot, donde se debe tomar en cuenta detallar cada uno de los aspectos correspondientes al robot paralelo con configuración 3UPS-1RPU, incluyendo características dinámicas del mismo. Estas características dinámicas se pueden obtener de diversas formas, para el actual proyecto se hizo uso del modelo existente dentro del proyecto de investigación PIGR-20-01 realizado en SOLIDWORKS, de este se deben migrar las diferentes piezas que conforman al robot. Además, se debe recordar que en ROS2 se puede hacer uso de archivos de extensión XACRO, que son archivos que permiten obtener una estructura robótica del tipo modular, evitando así la complejidad de una descripción única del robot, es decir que se puede hacer uso de estos para reducir el código principal. Consecuente a ello los archivos XACRO poseen un soporte de macros, los cuales son archivos que forman parte del XACRO principal, manteniendo la condición de modularidad que ayuda a reducir el código principal y facilita la edición de las diferentes partes que conforman el robot [18].

La estructura URDF del robot paralelo a implementar se basa principalmente en 3 archivos del tipo XACRO, denominados “parallel_robot.xacro” , “parallel_leg.xacro” y “parallel_control”, mismos que se detallan a continuación.

2.2.2.1 Parallel_robot.xacro:

El archivo “parallel_robot.xacro”, será utilizado como archivo principal, es decir que este contendrá la descripción completa del robot, por esta razón este archivo será encargado

de abstraer los archivos macros, que permiten simplificar el código principal. Este también será el encargado de describir parámetros referentes a los simuladores y marco de control a utilizar para manipular al robot. La estructura principal simplificada de este archivo se puede apreciar en el diagrama de la Figura 2.5.

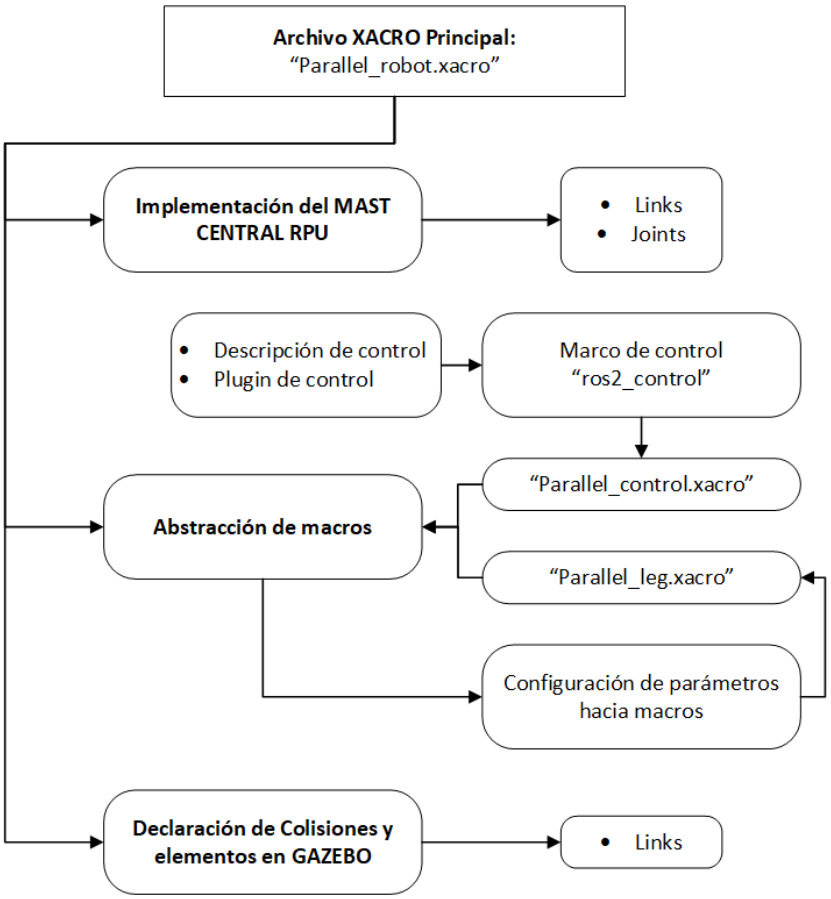


Figura 2.5 Estructura básica del archivo principal "Parallel_robbot.xacro"

A continuación se describe con mayor detalle las secciones que conforman el archivo principal "Paralle_robot.xacro":

Implementación del mast central: Como se puede apreciar en la Figura 2.5, la primera sección del archivo principal de extensión XACRO, se compone de la implementación de la cadena cinemática principal que corresponde a la pata central RPU. En esta se describen los parámetros dinámicos y cinemáticos correspondientes a los Links y Joints de dicha cadena. Para una descripción más profunda de esta parte referente al robot paralelo se puede revisar el ANEXO II, así como la documentación oficial de URDF en [19].

Abstracción de macros: Una vez culminada la sección anterior se continua con la sección correspondiente a la abstracción de macros o abstracción de los archivos de

extensión XACRO secundarios, de los cuales se obtiene la información de los módulos que componen al robot. De manera simplificada, los macros contienen tanto la información de la estructura completa de una de las patas exteriores UPS del robot, así como toda la configuración requerida para proveer de control al mismo. Tanto la estructura de la pierna externa del robot, así como la configuración de control, comprenden un sin número de líneas de código adicional que debería incluirse en el archivo principal, ocasionando la extensión de este archivo de manera significativa. Por ello al hacer uso de macros, estos podrán ser llamados desde el código o archivo principal en una sola línea de código, reduciendo de manera significativa dicho código, además de poder realizar modificaciones en ciertas partes del robot sin necesidad de modificar el archivo principal. Además, se utilizan variables para configurar determinados parámetros como el origen de posición y orientación, que deberán tener los Joints que se describen en los archivos macros, gracias a esto se facilita la reubicación de las partes del robot.

A continuación, para obtener un mayor entendimiento de dichos macros o archivos XACRO secundarios, se explica el contenido y funcionamiento de estos dos archivos:

Parallel_leg.xacro: Este macro, posee la descripción general incluyendo las características dinámicas de una de las patas exteriores del robot. Dado que el robot paralelo se conforma por 3 patas exteriores con configuración UPS, que poseen las mismas características difiriendo solamente en su ubicación, este archivo permite describir la implementación de una de sus patas hacia la plataforma fija, para de esta manera poder ser reutilizada y reconfigurar su ubicación y disposición desde el archivo principal. Se puede hacer una analogía comparando el uso de archivos macros en una descripción URDF como si se utilizara una función dentro de un código en C++.

Parallel_control.xacro: Este macro es el encargado de describir**,** tanto el tipo de hardware utilizado por el robot así como las articulaciones en específicos que se van a controlar. Dentro de este se deben incluir varias características. A continuación, se detalla la sección principal que conforma dicho macro, como se puede observar en la Figura 2.5, la información que se detalla en el archivo “Parallel_control.xacro”, es referida principalmente al marco de control “ros2_control”.

Marco de control ros2_control: En esta sección, se debe implementar la configuración requerida para poder obtener un método de control sobre ciertas articulaciones del robot. A más de indicar dichas articulaciones, se debe tener en cuenta ciertos parámetros que se describen con mayor claridad en la sección 2.3 del presente documento

Colisiones en GAZEBO: Esta última sección dentro del archivo XACRO principal, comprende la definición de los eslabones (Links), que presentan la característica de colisión con respecto al mundo de GAZEBO, al momento de simular al robot. Estas características de colisión son diferentes de las características de colisión que poseen todos los links descritos en URDF. Dado que las colisiones descritas en el formato URDF son establecidas para que haya colisión entre un eslabón padre (parent) y un eslabón hijo (child), dichas colisiones son suficientes cuando se trata de robots manipuladores o que no poseen cadenas cinemáticas cerradas. Para el actual trabajo, el robot implementado se trata de un robot paralelo el cual posee varias cadenas cinemáticas cerradas, por ello es importante definir dichas colisiones con respecto al mundo GAZEBO. Debido a que, si no se definen correctamente estas colisiones, los elementos o patas externas del robot podrían traspasar entre sí, lo que afectaría de manera considerable al funcionamiento del modelo virtual. A más de ello debido a las características de los robots paralelos, de poseer, varias cadenas cinemáticas cerradas, se dificulta la descripción URDF, esto dado que los eslabones que conforman las patas del robot terminan en un mismo eslabón correspondiente a la plataforma móvil del robot, lo que no es permitido dentro de una estructura URDF. Para poder solucionar dicho problema se establecen las joints o articulaciones finales que conectan los vástagos de los cilindros externos con la plataforma móvil directamente en el entorno de GAZEBO, haciendo uso de las características de colisión mencionadas anteriormente para poder cerrar dichas cadenas. Las articulaciones cerradas en el mundo GAZEBO, podrán ser visualizadas en secciones posteriores, donde el robot descrito mediante los distintos archivos XACRO mencionados, sea puesto en simulación mediante GAZEBO11. Los códigos completos de todos los archivos XACRO mencionados se podrán observar en el ANEXO III.

2.2.3 INCORPORACIÓN GENERAL DEL ROBOT EN GAZEBO:

Una vez que en la sección anterior se ha descrito el contenido de la descripción completa URDF, y el contenido que posee cada uno de los archivos XACRO de manera simplificada, ahora se procede a utilizar estos archivos mediante el uso de ROS2 y los paquetes requeridos que se instalaron en secciones anteriores, para poder lanzar o ejecutar los archivos mencionados dentro del simulador GAZEBO. La estructura que seguirá el modelo para la implementación en 3D del robot en el simulador GAZEBO se puede observar en la Figura 2.6.

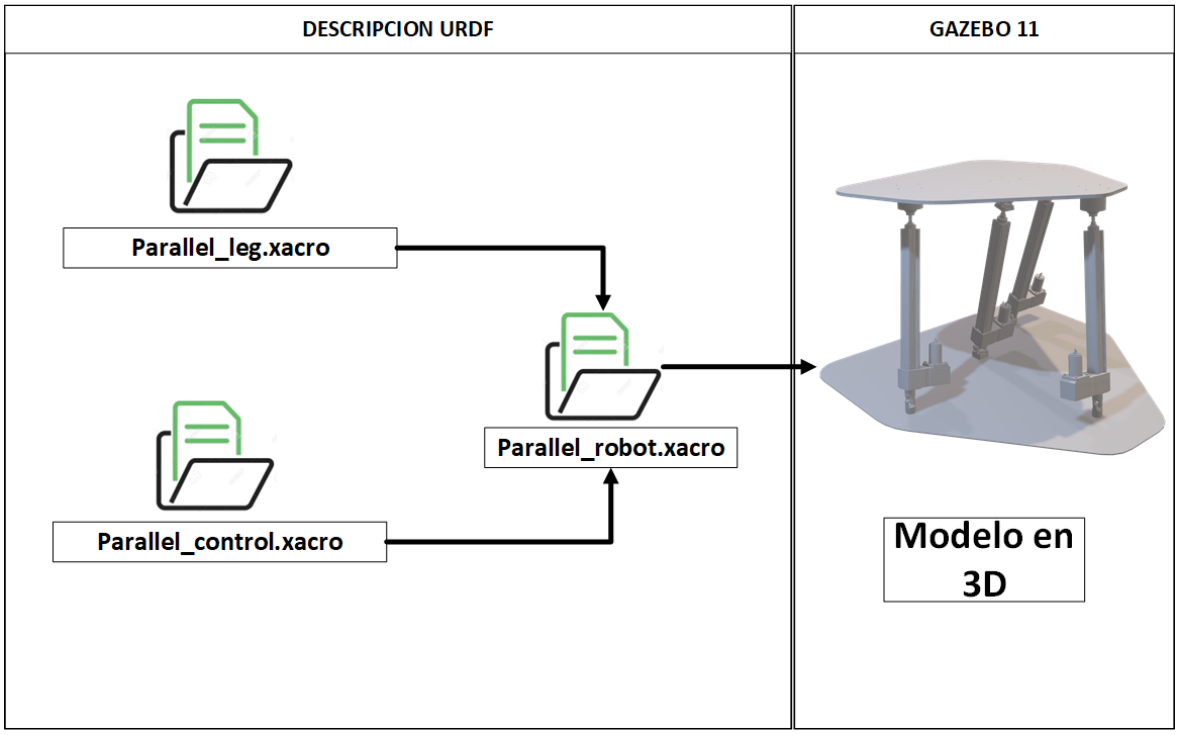


Figura 2.6 Estructura general del robot en 3D

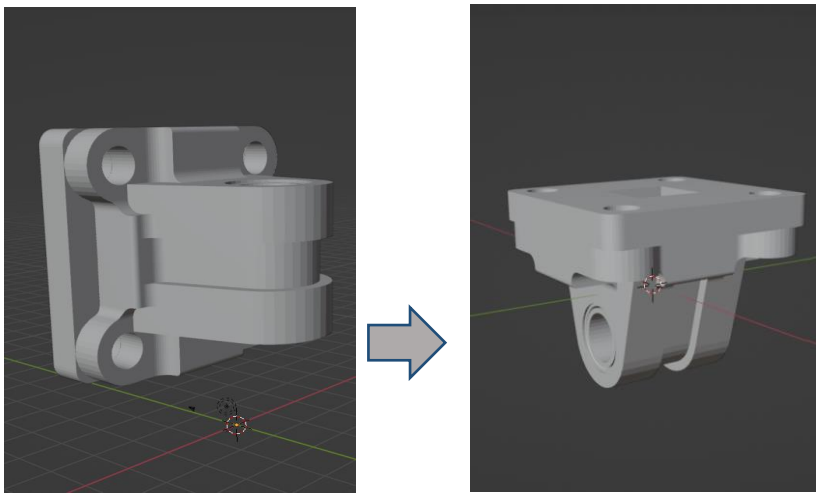
2.2.3.1 Generación de partes en 3D

Para poder visualizar en el simulador GAZEBO, un robot paralelo con la disposición 3UPS-1RPU de manera realista con respecto al robot real, se empleó el uso de mallas (meshes), estas se incorporarán en los archivos XACRO del robot paralelo descritos anteriormente, específicamente en los Links implementados. Para saber cómo incorporar dichos archivos se puede revisar la información detallada en el ANEXO II. Estos son archivos gráficos, mismos que para este presente trabajo se encuentran en formato DAE (Digital Asset Exchange), este es el formato más ligero y que mejor procesa el simulador a utilizar. Estos archivos permiten incorporar modelados en 3D de las piezas a utilizar por el robot y pueden ser modificados haciendo uso de softwares de diseño gráfico como lo son SolidWorks o Blender.

Para el presente trabajo se utilizó el software gráfico denominado Blender, mismo que se dedica de manera específica al modelado, animación y creación de modelos tridimensionales [20]. Una de las ventajas que presenta este programa es su uso gratuito ya que no requiere de licencia para su ejecución, además que es compatible con varios formatos como STL, BVH, DAE, etc. y con el sistema operativo Linux, mismo que se está utilizando para la generación del modelo virtual.

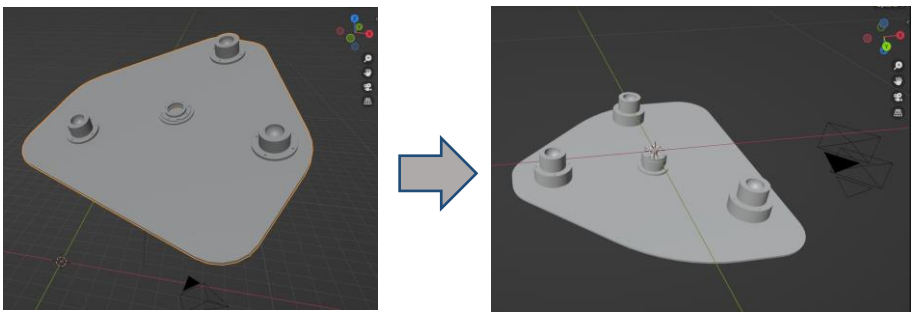
Los archivos base para la incorporación del robot, fueron creados en el proyecto grupal PIGR 20-01 en el programa SolidWorks, mismo que presenta un gran inconveniente al

momento de exportar las piezas, ya que SolidWorks no permite la exportación de archivos en formato DAE. Además, dichos archivos se exportan con errores de escala y ubicación de su punto de referencia en cada una de las piezas, mismo que es de vital importancia, ya que las características como los momentos de inercia se determinan en SolidWorks para un sistema de referencia fijo. Dado estos inconvenientes se hace uso de Blender, para modificar los parámetros de escala, así como la ubicación del centro de referencia y asi poder exportar dichas piezas en formato DAE y que estas puedan ser utilizadas dentro de los archivos XACRO para su simulación en GAZEBO. Las modificaciones realizadas en el robot real deben ser actualizados según las especificaciones dentro del modelo de solidwork, para de esta manera poder exportar las piezas en STL, tratarlas en Blender y asi obtener los archivos de malla (meshes) en formato DAE. La modificación de dichos archivos se puede apreciar en la Figura 2.7.



STL (Solidworks) DAE (Blender)

(a) Unión 4



STL (SolidWorks) DAE (Blender)

(b) Plataforma Móvil

Figura 2.7 Modificación de piezas utilizando Blender

Los archivos fueron modificados y exportados desde Blender en formato DAE (Digital Asset Exchangwe), como se puede apreciar en la Figura 2.7, los cambios que se realiza principalmente en Blender es el correcto escalamiento y la ubicación de los centros de referencia, además de la exportación de dichas piezas en archivos de formato DAE. Mismos que permiten el intercambio de información digital con un formato en 3D, y es el más adecuado para la transmisión de datos hacia el simulador GAZEBO, permitiendo obtener la apariencia exacta del robot [21]. Estos archivos deben ser ubicados en la carpeta meshes, ya que este directorio o ruta se empleará durante la incorporación en URDF.

Una vez ubicados los archivos XACRO en la carpeta URDF y los archivos DAE en la carpeta meshes, ya se dispone de la descripción completa del robot, para poder ubicarlo o incorporarlo al simulador GAZEBO.

Pero antes de probar la simulación se debe colocar correctamente las características dinámicas del robot dentro de la descripción URDF, principalmente sus momentos de inercia. Una vez que se obtiene las piezas en Blender con sus sistemas de referencia ubicados correctamente se podrá utilizar los parámetros dinámicos que nos brinda SolidWorks y que fueron obtenidos gracias al modelo existente dentro del proyecto de investigación PIGR 20-01, dichos parámetros se pueden observar tabulados dentro del ANEXO IV.

2.2.4 SIMULACIÓN DEL ROBOT PARALELO 3UPS-1RPU EN GAZEBO

Para poder simular y ubicar el robot dentro del entorno GAZEBO, se debe implementar un archivo de lanzamiento, mismo que dentro de ROS2 hace uso de lenguaje en Python, la descripción de como generar el archivo launch puede ser revisada a profundidad en [22].

A continuación, se puede observar en la Figura 2.8, la estructura del archivo de lanzamiento para poder extraer la descripción del robot de los archivos XACRO y poder simular el robot en GAZEBO:

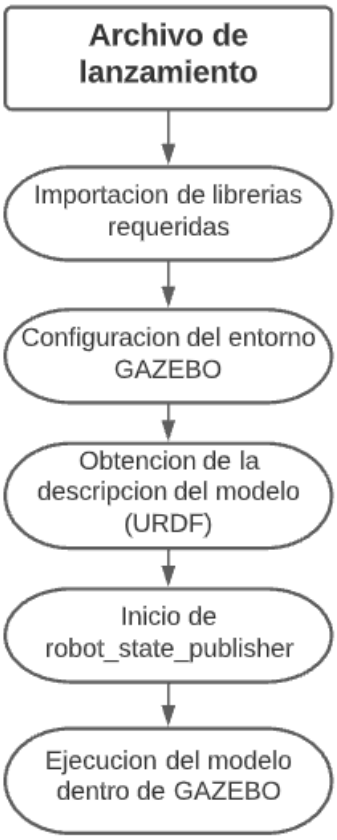


Figura 2.8 Estructura Archivo de lanzamiento para simular el robot en Gazebo

El archivo de lanzamiento deberá ser ubicado dentro de la carpeta “launch”, el nombre del archivo tendrá el siguiente formato “robot_paralelo.launch.py”. Una vez que se tenga dentro del paquete denominado “parallel_robot”, todos los archivos necesarios para la simulación se hará una compilación de todo el proyecto para lo cual se debe dirigir hacia la carpeta raíz del espacio de trabajo “dev_ws”, y se hará uso del siguiente comando:

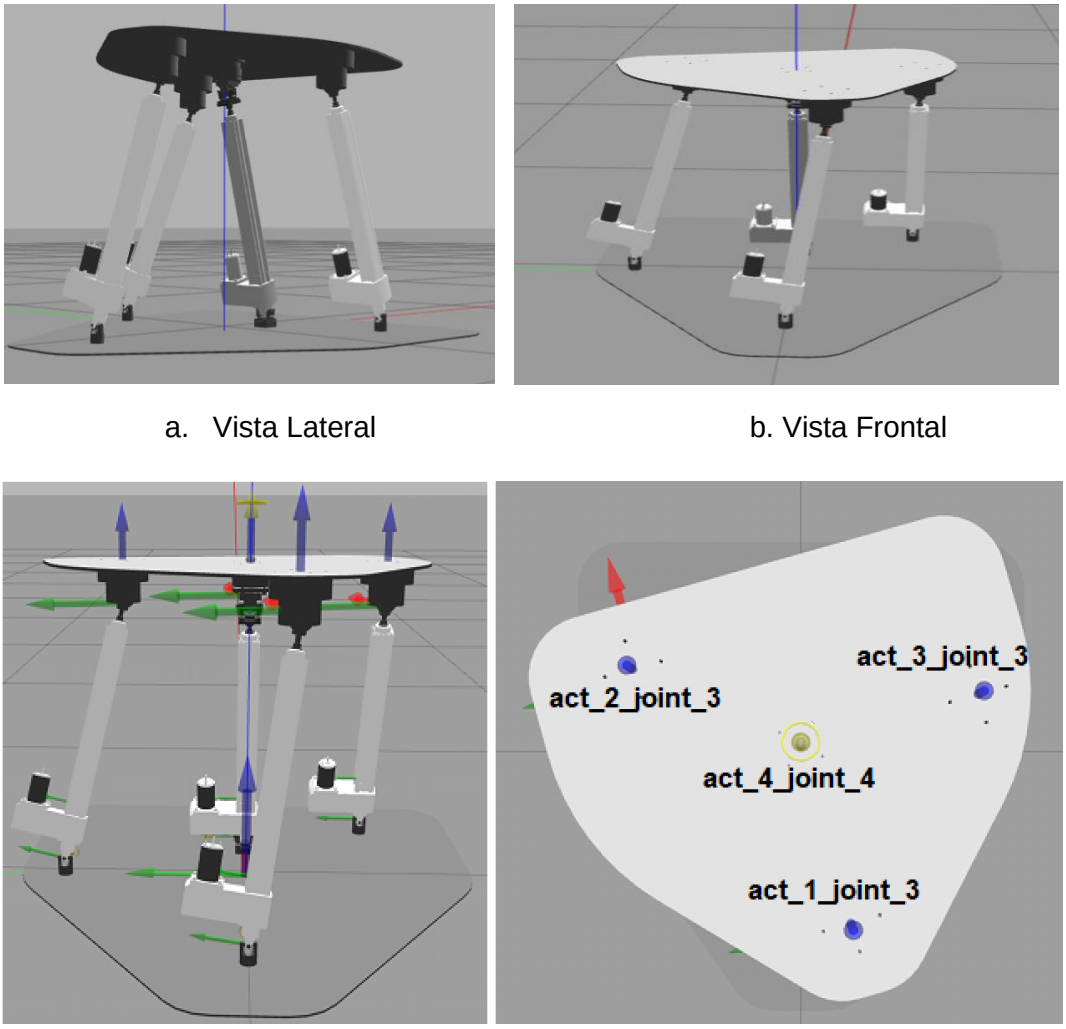
```
Colcon build
```

Antes de ejecutar el compilador Colcon, verificar que este se encuentre instalado dentro de ROS2, de lo contrario se debe instalarlo para evitar errores.

Cuando haya culminado la compilación del paquete, ya podremos ejecutar el archivo de lanzamiento creado para la simulación del robot, mediante el siguiente comando:

```
ros2 launch parallel_robot robot_paralelo.launch.py
```

A continuación de ingresado este comando se debe poder observar el modelo del robot dentro del entorno virtual de GAZEBO, como se muestra en la Figura 2.9.



a. Vista Lateral

b. Vista Frontal

c. Vista de articulaciones

Figura 2.9 Simulación en GAZEBO del robot paralelo 3UPS-1RPU

Como se puede observar en la Figura 2.9, se observan las articulaciones que conectan los vástagos tanto central como exteriores a la plataforma móvil. Como se mencionó en secciones anteriores, la descripción URDF limita el cierre de cadenas cinemáticas variadas en un robot, por ello las articulaciones del actuador 1, 2 y 3 que se aprecian en la imagen, se encuentran declaradas únicamente en el mundo GAZEBO. Esto limita al robot a ser utilizado dentro de otros simuladores como RVIZ compatibles con ROS2, pero permite un gran manejo a nivel de GAZEBO.

2.3 CONTROL DEL MODELO VIRTUAL DEL ROBOT PARALELO 3UPS-1RPU

Gracias al uso de las diferentes librerías existentes en el sistema operativo ROS2, se realizará un control individual y en conjunto de los diferentes actuadores que incorpora el

robot, para de esta manera poder posicionar la plataforma móvil del robot en una posición deseada.

2.3.1 CONTROL DEL ROBOT PARALELO 3UPS-1RPU

Para poder implementar una estructura de control en el ambiente de ROS2, se debe recurrir al marco de control ros2_control, mismo que permite implementar los controladores a utilizar para manejar las diferentes articulaciones del robot paralelo.

Los paquetes requeridos para hacer uso de la estructura o marco de control ros2_control se los puede encontrar en la página principal de Github, estos paquetes deberán ser clonados del repositorio Github y ubicados dentro de la carpeta src del ambiente de trabajo. Para obtener mayor información de como clonar y utilizar los paquetes del marco ros2_control puede revisar el contenido en [23].

Para poder incorporar el marco de control dentro del robot paralelo 3UPS-1RPU, se debe realizar tanto la configuración correspondiente dentro del archivo XACRO principal, así como la creación y edición de un archivo en formato YAML, todo este proceso para poder incorporar control dentro del robot se puede apreciar de manera simplificada y detallada en la Figura 2.10.

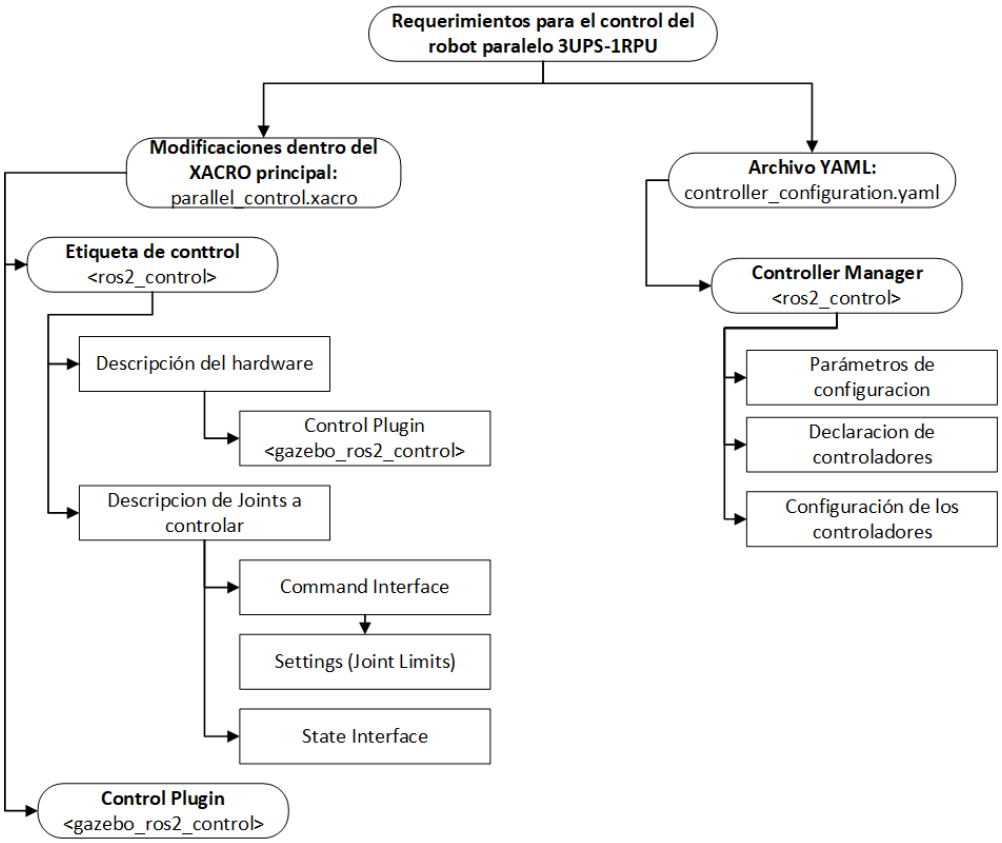


Figura 2.10 Implementación del marco de control hacia el robot paralelo

Como se puede observar en el esquema resumido, una de las partes principales para incorporar el control dentro del robot, es modificar su archivo URDF o en este caso el archivo XACRO principal denominado “parallel_robot.xacro”, así como la creación y edición del archivo de configuración YAML, que configurara los controladores a cargar en el robot, estos aspectos se detallaran de mejor manera en las secciones posteriores.

2.3.2 GAZEBO_ROS2_CONTROL

Dado que el presente proyecto se realiza a nivel de simulación, mediante el uso de GAZEBO11, se requiere de un plugin necesario para comunicar el entorno de simulación con el marco ros2_control, por ello se deberá hacer uso del paquete desarrollado en ROS2 denominado gazebo_ros2_control, este permitirá la comunicación del entorno virtual con el marco de control, permitiendo así adquirir el estado del hardware simulado y cargar en el mismo los diferentes controladores a utilizar para el manejo de las articulaciones del robot.

Una vez descrito el plugin que se utiliza para poder realizar el control del modelo virtual, se puede observar en la Figura 2.11, todos los paquetes requeridos para poder implementar los controladores y la estructura de control. Estos deben ser obtenidos de los repositorios oficiales de GitHub, para poder compilarlos y tener los recursos requeridos para controlar al robot. Para poder realizar el clonado, instalación y ejecución de dichos paquetes dentro del proyecto del robot paralelo puede seguir los pasos descritos en el ANEXO V.

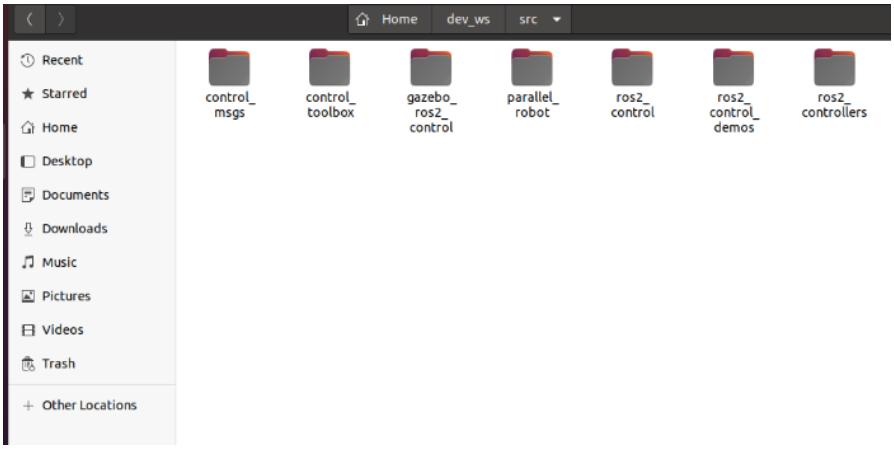


Figura 2.11 Paquetes requeridos para controlar el robot

2.3.3 CONFIGURACIÓN DE ROS2_CONTROL EN URDF

Una vez que se dispongan de todos los recursos requeridos para el control del robot dentro del espacio de trabajo, se procederá a realizar las modificaciones requeridas en el archivo URDF del robot, en el robot paralelo actual, modificaremos el archivo XACRO

principal denominado “parallel_robot.xacro”, añadiendo la siguiente etiqueta, para poder obtener el estado del hardware simulado dentro del entorno de GAZEBO, dicha etiqueta se puede observar en la Figura 2.12.

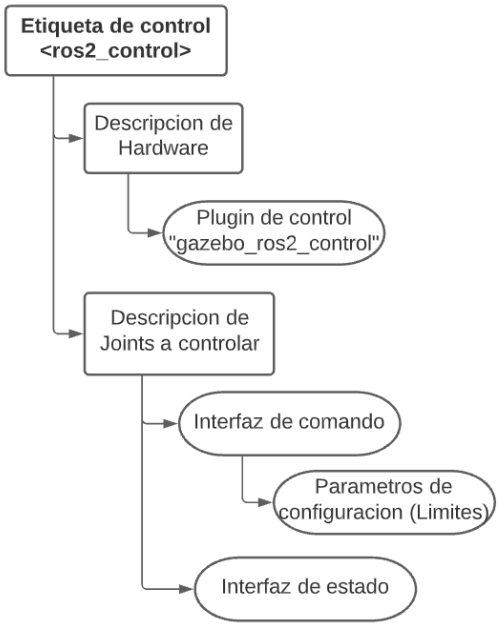


Figura 2.12 Configuración de etiqueta de control en archivo URDF

La siguiente etiqueta permite implementar el marco de control en ROS2, esta debido a que se la utilizara a nivel de simulación y no con hardware físico, se la utilizara en tipo “System”, dentro de este sistema ubicaremos las articulaciones que se requieren controlar para ubicar la plataforma móvil, cabe recordar que el control será realizado sobre las articulaciones prismáticas correspondientes a los actuadores del robot, para dicho control de las articulaciones se las deberá añadir siguiendo el ejemplo y estructura del “joint” dentro de la etiqueta “ros2_control”.

Estructura etiqueta “ros2_control”:

Hardware: Dentro del hardware para este caso en específico se hará uso del Plugin que se encuentra dentro del paquete “gazebo_ros2_control”.

Joint: Dentro del joint se debe añadir diferentes parámetros relacionados con la articulación que se desea controlar.

- **Command_Interface:** Se debe especificar el comando o el parámetro con el que se comandará la articulación, en este caso el entorno de ros2_control, permite comandar la articulación mediante “position”, “effort” o “velocity”, para este caso en específico se hará uso del esfuerzo.

- **Param:** Se deberá colocar los límites de posición, fuerza o velocidad, que permite la articulación a controlar.
- **State_interface**: En este campo se debe ubicar el estado que se publicara de la articulación, este permite visualizar al igual que en casos anteriores la posición, fuerza o velocidad de la articulación.
- Además de la etiqueta de control, se deberá incluir dentro del archivo XACRO principal la etiqueta para activar el plugin que permite la comunicación entre GAZEBO y ros2_control, misma se puede observar en la Figura 2.13.

```
<!-- PLUGIN DE CONTROL-->
    <gazebo>
        <plugin filename="libgazebo_ros2_control.so" name="gazebo_ros2_control">
        <robot_param>robot_description</robot_param>
        <robot_param_node>robot_state_publisher</robot_param_node>
        <parameters>$(find parallel_robot)/config/parallel_control2.yaml</parameters>
        </plugin>
    </gazebo>
<!-- -->
```

Figura 2.13 Plugin de control

Dentro del plugin de control, se deberán colocar la ubicación donde se encontrarán los parámetros de control utilizados por el “Controller_manager”, para configurar los diferentes controladores a utilizar. Podemos observar un ejemplo de configuración en la Figura 2.14

Figura 2.14 Configuración de archivo YAML para Controller_manager

Dentro del archivo YAML, se debe configurar los controladores a utilizar, las interfaces para comandar la articulación y los estados que se publicaran de la articulación, el archivo YAML una vez configurado, permitirá controlar la posición de las articulaciones prismáticas del robot, y permite visualizar estados de la posición y velocidad que adopta dicha articulación.

Una vez que se configure el archivo URDF, del robot paralelo, podremos interactuar con el robot simulado mediante el uso del marco ros2_control haciendo uso de los comandos desde la terminal para poder verificar el estado de los controladores incorporados a la simulación.

Los comandos requeridos para poder implementar el control y manejar las articulaciones del robot son las siguientes [24]:

- ros2 control list_hardware_interfaces (Obtener lista de interfaces de hardware)
- ros2 control list_controller_types (Obtener la descripción del tipo de controladores utilizados)
- ros2 control list_controllers (Obtener una lista de controladores utilizados en simulación)
- ros2 control reload_controller_libraries (Recargar las librerías de controladores)
- ros2 control load_controller <controller_name> (Cargar un controlador)
- ros2 control unload_controller <controller_name> (Quitar un controlador)
- ros2 control set_controller_state <controller_name> <configure |start | stop> (Setear el estado del controlador)
- ros2 control switch_controllers −−stop <controller_list> −− start <controller_list> (Iniciar o parar uno o más controladores)

Una vez configurado todos los archivos correspondientes, al paquete o proyecto del robot paralelo, se puede iniciar la simulación, para verificar que esta se ejecuta de manera correcta, se deberá cargar los controladores configurados en el archivo YAML.

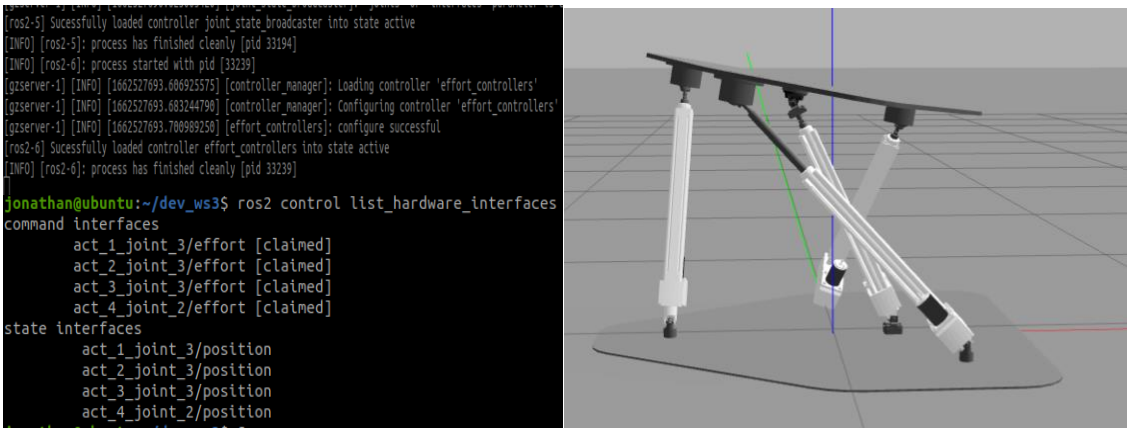


Figura 2.15 Carga de controladores en GAZEBO

Como se puede observar en la Figura 2.15, los controladores se encuentras cargados correctamente dentro del entorno de GAZEBO, además estos proporcionan una interfaz de hardware de esfuerzo, y publican estados de posición y fuerza, para poder comprender como se está desenvolviendo la ejecución de nodos dentro del entorno completo, se hará uso de la herramienta “rqt_graph”, para verificar el estado de los nodos que se encuentran en ejecución.

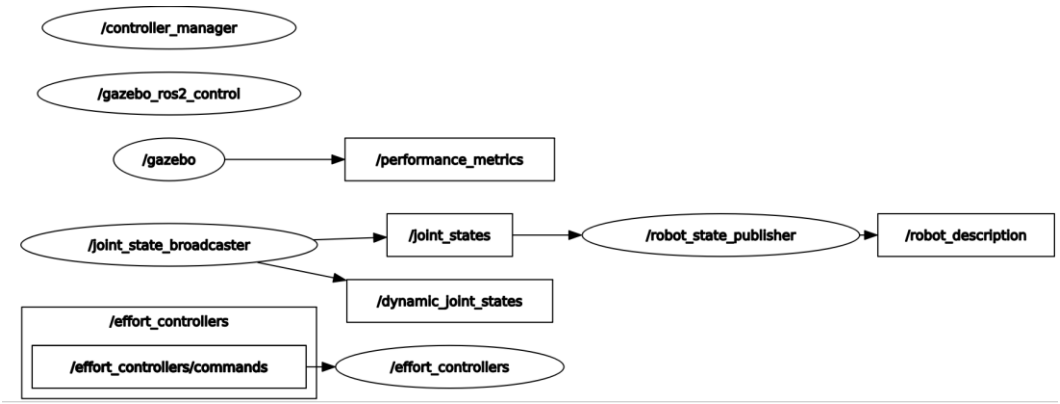


Figura 2.16 Ejecución de nodos en el sistema

Como se puede observar en la Figura 2.16, mediante el esquema simplificado, podemos observar cómo se sincronizan los nodos de ejecución de gazebo y gazebo_ros2_control, a continuación se hace uso de dos controladores preestablecidos en el paquete “ros2_controllers”, mismos que son el “joint_state_broadcaster”, encargado de publicar y suscribir el estado de la articulaciones del robot, mediante los tópicos “/joint_states” y “/dynamic_joint_states”, gracias a este se obtiene el estado de posición de cada una de las articulaciones prismáticas. El segundo controlador preestablecido a utilizar es el “effort_controllers” o controlador de esfuerzo conjunto, este controlador es clave al momento de controlar la posición de los actuadores, proceso que se explicara a continuación.

Gracias al controlador “effort_controller”, obtenemos el tópico de comando denominado “effort_controllers/commands”, mismo que permite enviar hacia el sistema o robot estados de esfuerzo para poder impulsar los actuadores a diversas posiciones, para enviar una instrucción a este tópico se puede utilizar le siguiente comando:

```
ros2 topic pub /effort_controller/commands std_msgs/msg/Float64MultiArray
"data:
- 100
- 30
- 40
- 50" -1
```

El comando anterior publica valores de posición en el tópico, a los cuales deberán llegar los diferentes actuadores, para este caso en particular se manejan las 4 articulaciones prismáticas que predominan la plataforma móvil, por ello se maneja un arreglo MultiArray o vector de 4 valores, un valor para cada una de las articulaciones.

2.3.4 CREACION DEL CONTROLADOR DE ESTADO CONJUNTO PID

A continuación, se describirá la implementación del nodo de control PID, para las articulaciones prismáticas del robot, mismo que recibirá como parámetro de entrada la posición deseada del usuario y a su salida obtendremos un valor de esfuerzo que deberá enviarse a la articulación para llegar a la posición deseada.

Para este tipo de metodología se hace uso del marco de control ros2_control con sus controladores preestablecidos para poder acceder a la interfaz de hardware virtual. Luego de esto se procede a generar un nodo que trabaje externamente retroalimentando los estados y preparando las acciones de control de esfuerzo que deben ser remitidas hacia el modelo virtual, para el posicionamiento de los vástagos. En la Figura 2.17, podemos observar de manera comprimida el funcionamiento del nodo PID.

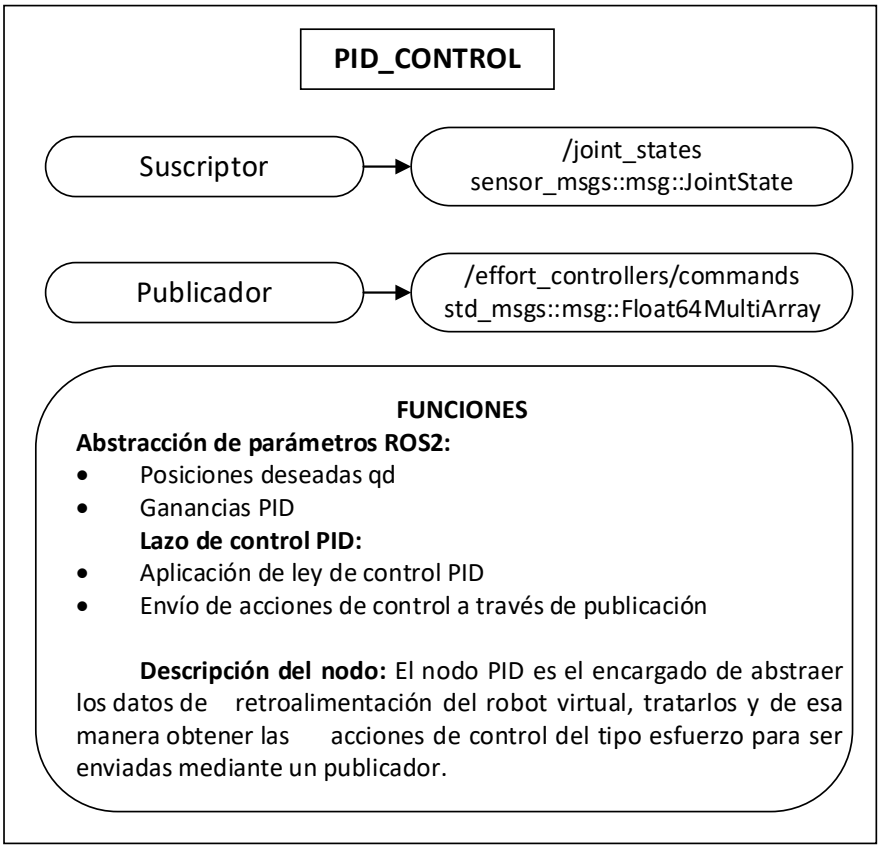


Figura 2.17 Nodo de control PID

Como se puede observar el funcionamiento principal del nodo es suscribirse a la retroalimentación de estados que nos entrega el controlador “joint_state_broadcaster”, abstraer los parámetros requeridos como las posiciones deseadas, aplicar la ley de control PID mostrada en (2.10), para de esta manera obtener un valor de esfuerzo a la salida, y mediante la sintonización de parámetros lograr que dicho valor de esfuerzo se publique en el controlador “effort_controllers” y de esta manera lograr que los vástagos lleguen a la posición deseada.

$$\tau = k_p(q_d - q) + k_i\,\dot{q} + k_d \int (q_d - q) \qquad (2.10)$$

Donde:

k_p, k_i, k_d: Ganancias PID

q_d: Posición deseada

q: Posición final

$\dot{q}$: Derivada de la posición final

2.3.5 EJECUCIÓN DEL MODELO VIRTUAL EN CONJUNTO CON EL CONTROLADOR PID

Una vez realizado el script con el algoritmo del nodo PID, se puede realizar la ejecución del modelo virtual en GAZEBO, para después proceder a correr el nodo encargado del control del estado conjunto de las articulaciones prismáticas, “pid_control”, una vez ejecutado ambos nodos se obtiene el siguiente esquema de trabajo mediante el “rqt_graph”, mostrado en la Figura 2.18.

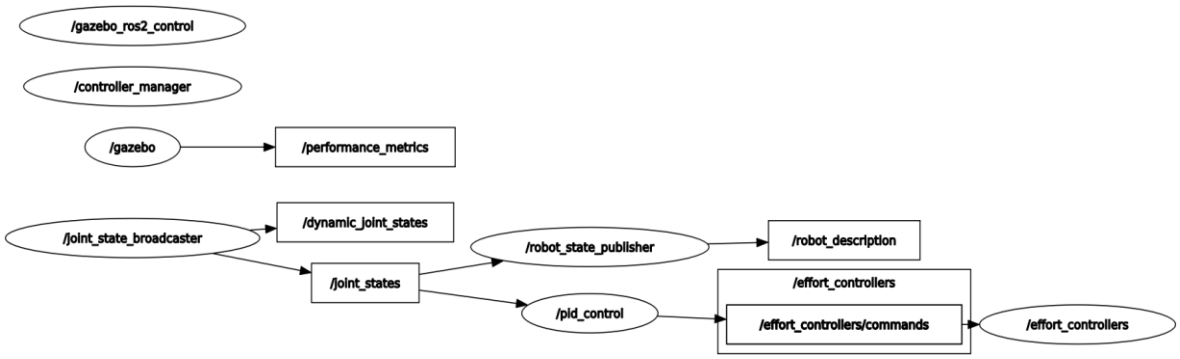
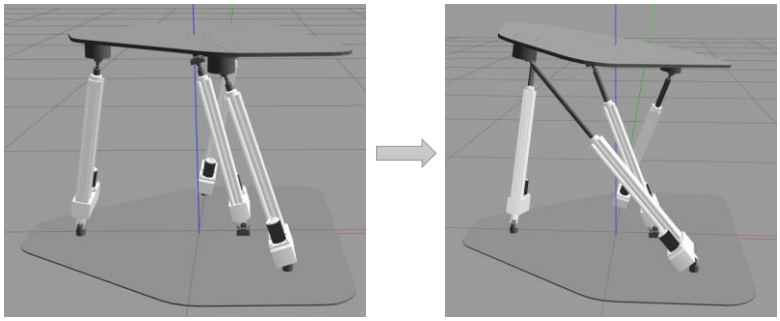


Figura 2.18 Esquema completo de control PID en ROS2

Como podemos observar en la Figura 2.18, el nodo “/pid_control” se encuentra activo realizando las diversas funciones mencionadas en secciones anteriores, cabe aclarar que este nodo a pesar de tratarse de un nodo de control, también se encarga de abstraer la

información del usuario como la posición deseada y las constantes del controlador. Dichos parámetros en ROS2 pueden ser configurados directamente desde el script, así como también pueden ser seteados directamente desde consola mediane los diversos comandos otorgados por ROS2, específicamente en la sección de parámetros, para poder ver esta información de manera más detallada, así como el desarrollo del script del nodo PID, se recomienda revisar el manual de usuario que se encuentra en el ANEXO VI.



a) Posición Inicial b) Posición Final

Figura 2.19 Posicionamiento del robot mediante PID conjunto (Secuencia de imágenes)

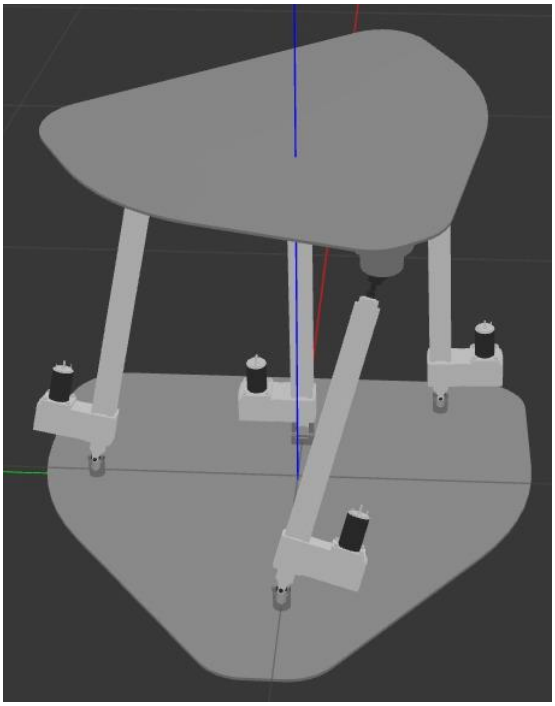
Como se puede observar en la Figura 2.19 Posicionamiento del robot mediante PID Figura 2.19 , una vez que ingresamos la posicion deseada de los vastagos o articulaciones prismaticas estas tratan de alcanzar la referencia, para lo cual se debe realizar una correcta sintonizacion para poder de esta manera determinar que el posicionamiento de la plataforma movil efectivamente se realiza.

3. RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

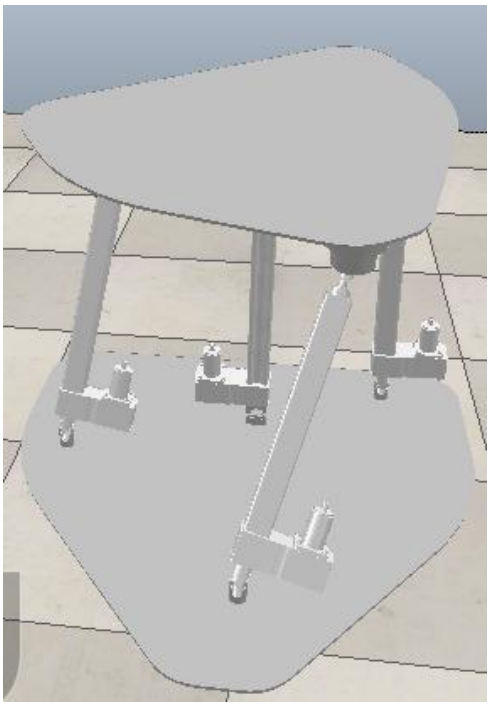
3.1 RESULTADOS

3.1.1 COMPROBACIÓN DEL MODELO VIRTUAL EN GAZEBO

Una vez compilado el modelo virtual y puesto en marcha dentro del mundo de GAZEBO, la primera prueba que se realizo fue la de posicionar el robot en el punto inicial correspondiente, y compararlo con respecto al que alcanza el robot físico real además del robot virtual existente en CoppeliaSim cuando todos sus vástagos inician en la posición inicial de cero es decir todos hacia dentro. Para ello se obtuvo la posición del centro de la plataforma móvil del robot simulado dentro de GAZEBO, mismo que se puede observar en la Figura 3.1 y se la comparo con la referencia del punto inicial obtenido del robot real y de CoppeliaSim.



a) Robot en GAZEBO



b) Robot en CoppeliaSim

c) Robot real

Figura 3.1 Posición inicial del robot simulado en GAZEBO y CoppeliaSim vs Robot Real

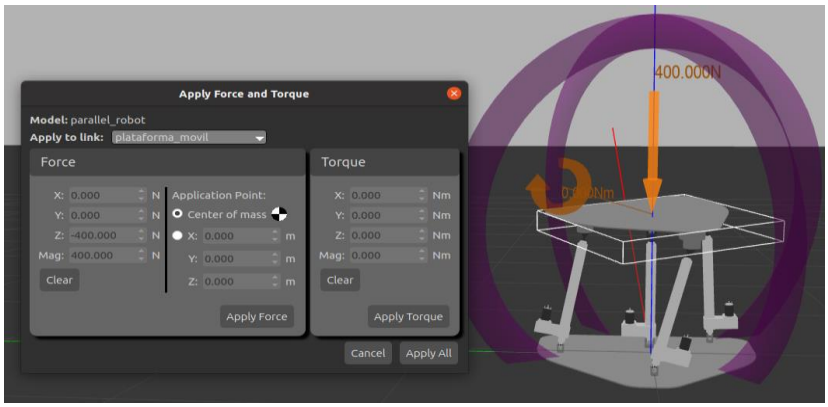
Tabla 3.1 Posición Inicial alcanzada por la plataforma móvil

Posición Inicial alcanzada por el centro de la plataforma móvil					
GAZEBO		**Robot real**		**CoppeliaSim**	
Xm	0.0106119	**Xm**	0.0117	**Xm**	0.02272
Ym	0	**Ym**	0	**Ym**	0
Zm	0.694217	**Zm**	0.681200	**Zm**	0.5812

Una vez obtenida la distancia del centro de la plataforma con respecto al mundo de GAZEBO se la comparo con la referencia de posición inicial alcanzada por el centro de la plataforma en el robot real. Comparadas las mediciones de la Tabla 3.1, se obtuvo un error del 1.87% para el eje Z y un error del 10% para el eje X, por lo que se puede asumir que el robot obtenido asemeja de manera correcta las características físicas del robot real, al menos en su posición inicial, dado que el punto inicial se obtiene con los vástagos iniciados en cero y con el robot en reposo es decir sin controlador, una vez validado el modelo en su posición inicial se procedió a realizar las pruebas correspondientes a la estabilidad del robot aun sin controlador, en secciones posteriores se añadió el control para poder comparar sus posiciones, nuevamente con las del robot real utilizando diferentes posiciones de los vástagos.

3.1.2 COMPROBACION DE ESTABILIDAD DEL MODELO VIRTUAL

Una vez comprobada la posición inicial del robot virtual, se procedió a comprobar la estabilidad del mismo, como se mencionó anteriormente dado a la dificultad de cerrar cadenas cinemáticas en URDF, estas fueron cerradas dentro del simulador GAZEBO, por lo que se realizaron pruebas de ejecución del modelo y aplicación de fuerzas externas hacia el mismo para determinar si el robot virtual cedía, se desarmaba o existía algún problema de colisiones en sus piezas ante la aplicación de dichas fuerzas externas, obteniendo los siguientes resultados.



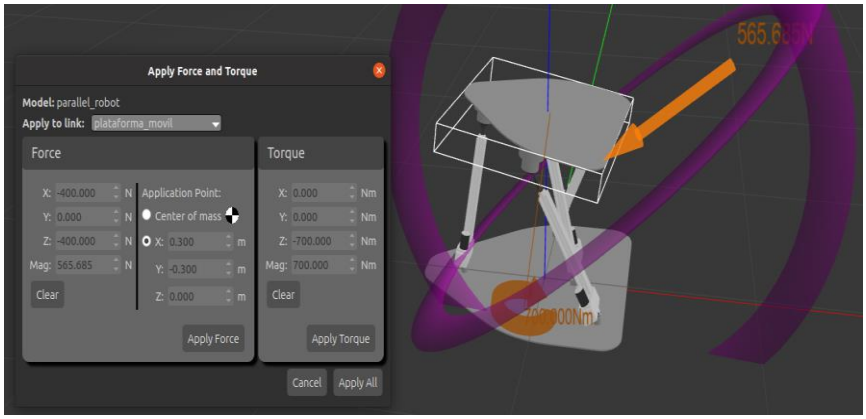


Figura 3.2 Aplicación de fuerzas externas hacia distintos puntos de la Plataforma móvil

Como se puede observar en la Figura 3.2, gracias a la interfaz que posee GAZEBO se logro aplicar fuerzas y torques en puntos estratégicos de la plataforma móvil, se realizaron repetidas simulaciones de prueba al modelo sin controlador, es decir en reposo. De esta manera se logró comprobar si el modelo virtual se logró incorporar dentro del simulador sin ningún inconveniente con respecto al ensamble de las piezas diseñadas para su ensamble, podemos observar las estadísticas de las pruebas realizadas en la Tabla 3.2

Tabla 3.2 Posiciones y magnitudes de fuerzas aplicadas en la Plataforma móvil con el robot en reposo

Nro. Prueba	**Plataforma móvil**								
1	**Punto de aplicación [m]**			**Fuerza [N]**			**Torque [N]**		
	x	**y**	**z**	**x**	**y**	**z**	**x**	**y**	**z**
	0	0	0	0	0	-400	0	0	0
2	**Punto de aplicación [m]**			**Fuerza [N]**			**Torque [N]**		
	x	**y**	**z**	**x**	**y**	**z**	**x**	**y**	**z**
	-0.3	0	0	0	0	-400	0	0	-500
3	**Punto de aplicación [m]**			**Fuerza [N]**			**Torque [N]**		
	x	**y**	**z**	**x**	**y**	**z**	**x**	**y**	**z**
	0.3	-0.3	0	-400	0	-400	0	0	-700
4	**Punto de aplicación [m]**			**Fuerza [N]**			**Torque [N]**		
	x	**y**	**z**	**x**	**y**	**z**	**x**	**y**	**z**
	-0.3	-0.1	0	600	-500	-2000	0	0	600
Observaciones	Las fuerzas aplicadas sobre la plataforma móvil fueron soportadas								

	sin ningún inconveniente, y el robot presento un nivel de estabilidad elevado

Una vez aplicadas dichas fuerzas a la plataforma móvil y ver que el robot no presento inconvenientes de estabilidad al momento de soportarlas, o algún tipo de desarme, se corroboro que el robot se encuentra ensamblado de manera correcta, y que las piezas correspondientes sujetan y se desenvuelven correctamente como si se trataran de las piezas reales, se puede determinar que el robot paralelo se encuentra ensamblado y listo sin ningún inconveniente, para proceder con las pruebas de control.

3.1.3 COMPROBACIÓN DE CONTROLADORES PID INDIVIDUALES

Una vez que se tiene listo el robot comprobando que sus piezas funcionan de manera adecuada, se realizaron las pruebas correspondientes a los controladores individuales del mismo, como se mencionó en secciones anteriores se hizo uso de un control PID para el manejo de cada una de las articulaciones prismáticas.

Una vez que se realizó la correcta sintonización de los parámetros del controlador PID, se debe tomar en cuenta que este trabajo de sintonización se lo realizo dos veces, una sintonización para una de las patas externas y otra para la pata central o mast central, ya que los parámetros o ganancias obtenidas para el controlador de una de las patas externas pueden ser utilizados sin problema en las dos restantes, dado que en el archivo XACRO principal se hace el llamado de un mismo archivo XACRO secundario para las piernas externas del robot, en consecuencia se puede decir que al obtener el control de una de las patas externas se logró también obtener el control sobre las dos restantes. Mientras que la pierna central necesito de otra fase de sintonización debido a que esta cambia en su estructura con respecto a las patas externas.

3.1.3.1 Sintonización de controlador PID individual

Una vez incorporada la ley de control mencionada en (2.10), dentro del modelo virtual del robot, se realizó mediante prueba y error una variación y tabulación de parámetros utilizados, para así poder obtener la mejor y más estable respuesta del sistema,

Tabla 3.3 Sintonización de ganancias Kp

N° Prueba	Kp	Kd	Ki
1	10	0	0
2	100	0	0
3	250	0	0

4	500	0	0
5	1000	0	0
6	1500	0	0
7	2000	0	0
8	2500	0	0

Primero determinamos el parámetro de la ganancia kp o proporcional para de esta manera verificar que el robot logre alcanzar la referencia establecida, para esta y las demás pruebas de sintonización individual se optó por retirar la plataforma móvil del robot para poder, obtener un control satisfactorio de cada uno de los actuadores, para de esta manera asegurar que cuando trabajen en conjunto para desplazar la plataforma móvil se obtenga el menor error posible.

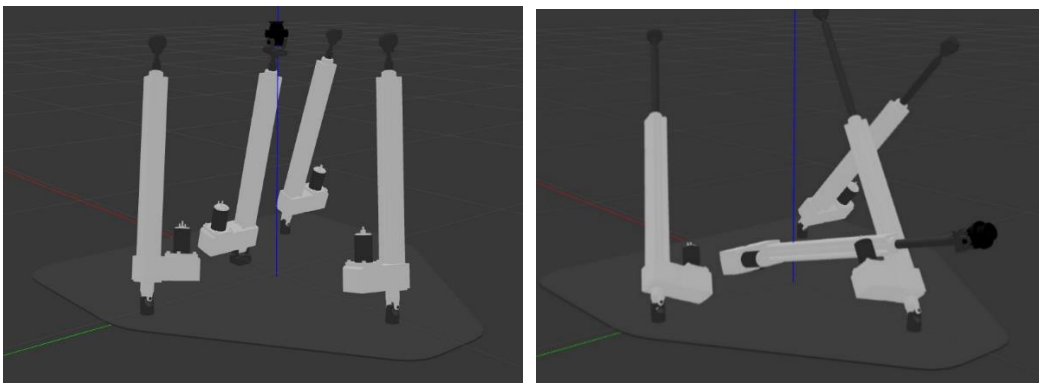
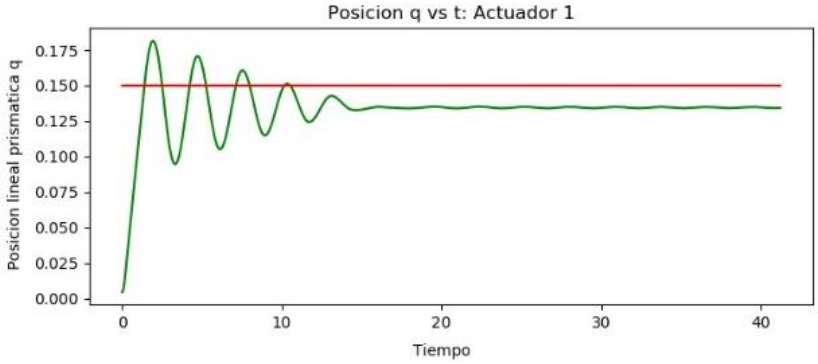
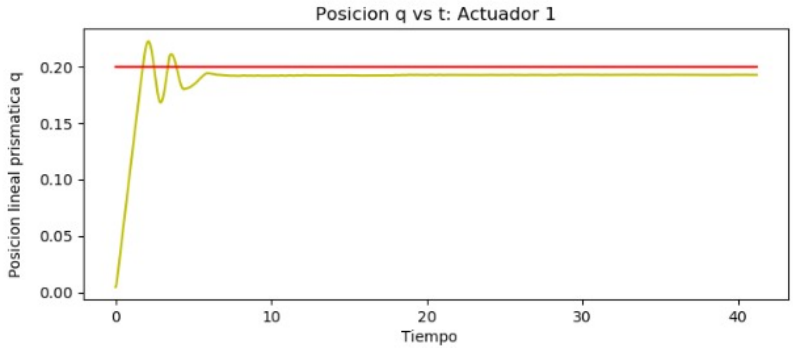


Figura 3.3 Robot sin plataforma móvil para sintonización de controladores individuales

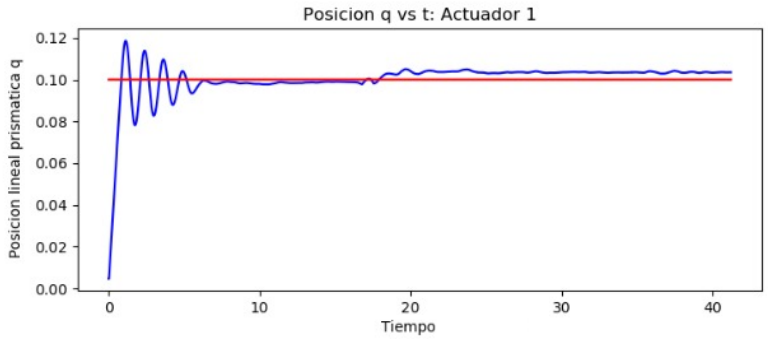
Como se puede observar en la Figura 1.1, se retiró la plataforma móvil con el fin de comprobar el control de cada uno de los vástagos individuales de las patas externas, una vez que se la retiro se procedió a variar kp, y se obtuvieron los siguientes resultados:



a) Kp=100, kd=0 ki=0



b) Kp=1500, kd=0 ki=0



c) Kp=2500, kd=0 ki=0

Figura 3.4 Respuestas de las articulaciones prismáticas con variación de Kp

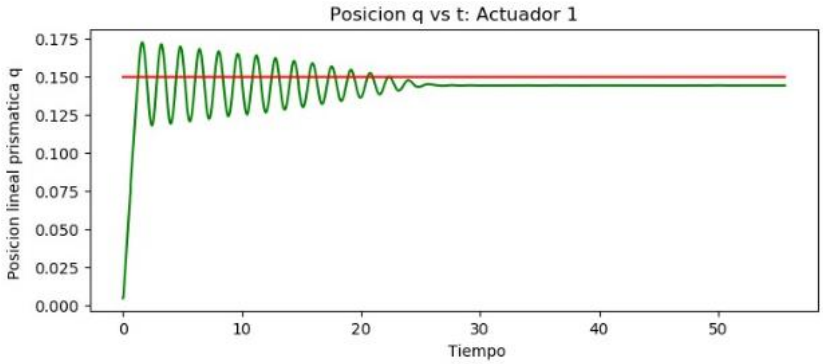
Como se puede observar en la Figura 3.4, se obtuvo como mejor resultado o respuesta estable el parámetro de 1500 como Kp, dado que a medida que se incrementa la constante proporcional se llega a la referencia pero se pierde estabilidad, ya que se puede observar que el modelo comienza a recibir vibraciones, lo cual no es una buena característica a obtener, de esta manera se decidió utilizar el parámetro de 1500, y a continuación se procedió a sintonizar los parámetros de Ki y Kd, para mejor la respuesta en estado estable y el kd para poder disminuir el error.

Tabla 3.4 Sintonización del parámetro Ki

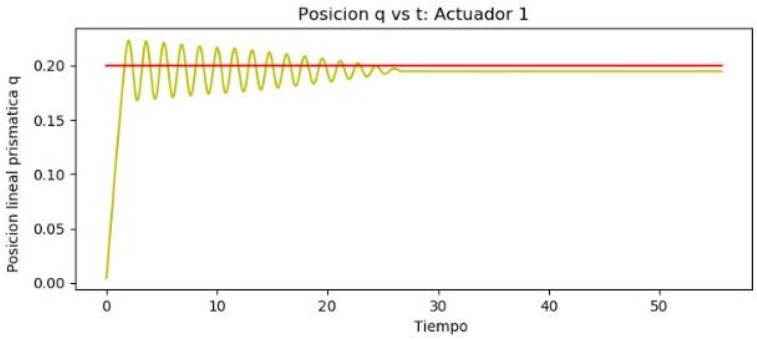
N° Prueba	Kp	Ki	Kd
1	1500	1	0
2	1500	0.1	0
3	1500	0.01	0

Como se puede apreciar en la Tabla 3.4, se ha tabulado los valores de 3 pruebas, pero se realizaron muchas más, pero para reducir la extensión del presente trabajo se

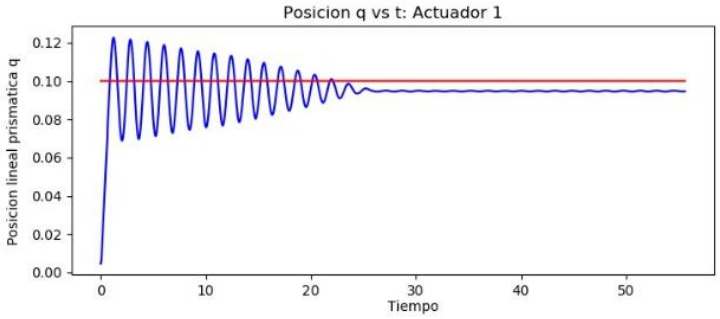
presentan las tabulaciones de los valores con mejores resultados, a continuación, se presentan las respuestas obtenidas por el sistema:



a) Kp=1500, ki=1 kd=0



b) Kp=1500, ki=0.1 kd=0



c) Kp=1500, ki=0.01 kd=0

Figura 3.5 Sintonización de parámetro Ki, para mejorar el establecimiento

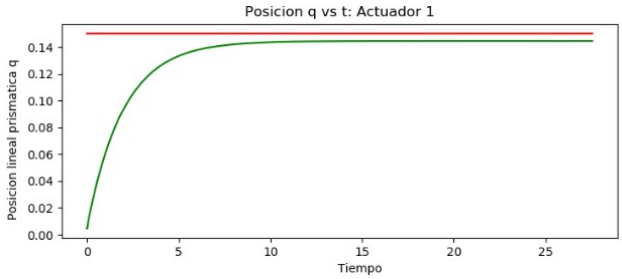
Como se puede observar en la Figura 3.5, el parámetro de Ki con el que hubo una mejor significativa en la respuesta estable es el de 0.1, por ello se hizo uso de Kd=1500 y Ki=0.1, para de esta forma realizar la variación de Kd y de esta forma poder reducir el error en estado estacionario, de esta manera se consiguió obtener los resultados esperados, a continuación, en la Tabla 3.5, se presentan las pruebas con las cuales se

obtuvo los mejores resultados al momento de disminuir el error en estado estacionario del sistema:

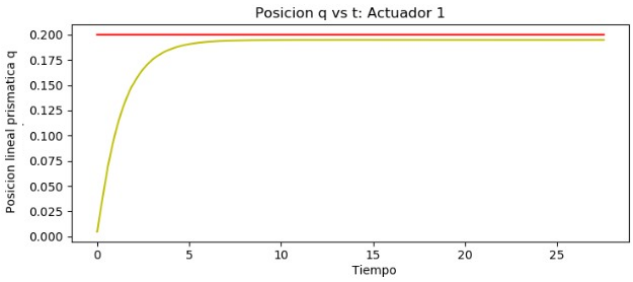
Tabla 3.5 Sintonización del parámetro Kd

N° Prueba	Kp	Ki	Kd
1	1500	0.1	10
2	1500	0.1	20
3	1500	0.1	30

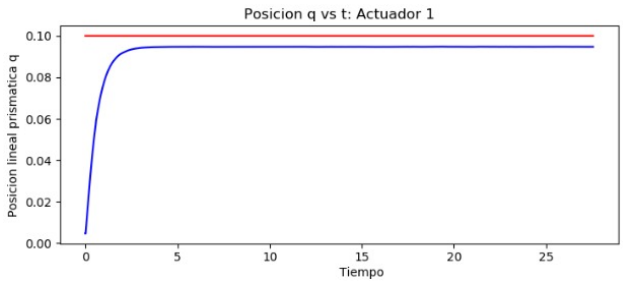
Las tres respuestas del sistema obtenidas durante la variación del parámetro kp fueron las siguientes:



a) Kp=1500, ki=0.1 kd=10



b) Kp=1500, ki=0.1 kp=20



a) Kp=1500, ki=0.1 kd=30

Figura 3.6 Resultados con la sintonización del parámetro Kd

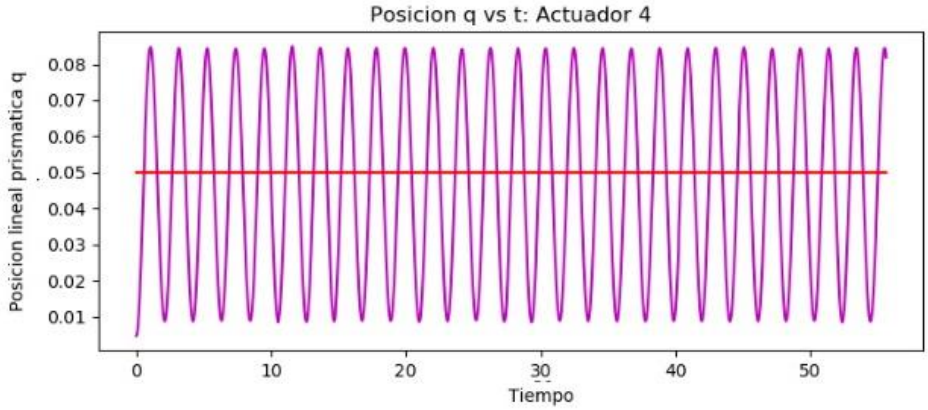
Como se puede observar en la Figura 3.6, la respuesta del sistema que mejor características presenta es la que se obtuvo haciendo uso de los parámetros kp=1500, ki=0.1 y kd=20, haciendo uso de los parámetros obtenidos durante esta sintonización se logró realizar el perfecto control individual de las articulaciones prismáticas mediante el bucle PID, permitiendo asemejar el sistema a una respuesta de un sistema de primer orden, logrando llegar sin problema a la referencia dada.

Una vez que se obtuvo los parámetros para el controlador PID de las patas exteriores se procedió a realizar la sintonización del mast central, para lo cual se utilizó la misma metodología explicada anteriormente, para simplificar el procedimiento se presenta a continuación la tabulación del os valores que se utilizaron durante la sintonización del mast central, así como las respuestas que se obtuvieron del sistema.

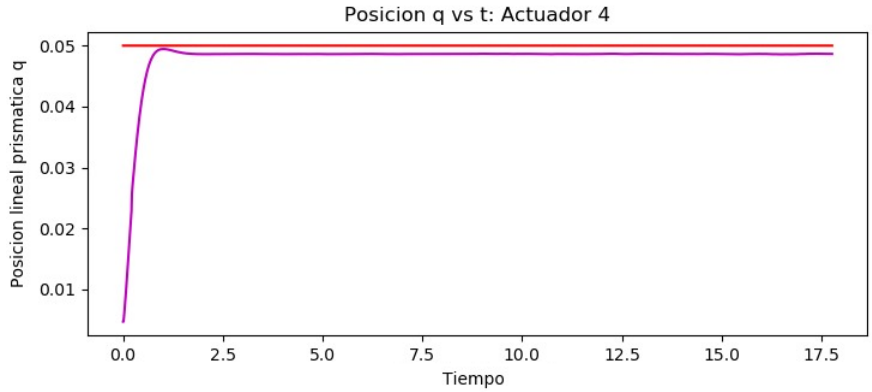
Tabla 3.6 Tabulación de parámetros PID para control del Mast Central

N° Prueba	Kp	Ki	Kd
1	1000	0	0
2	1500	0	0
3	3000	0	0
4	3000	0.1	0
5	3000	0.01	0
6	3000	0.01	10
7	3000	0.01	20
8	3000	0.01	30

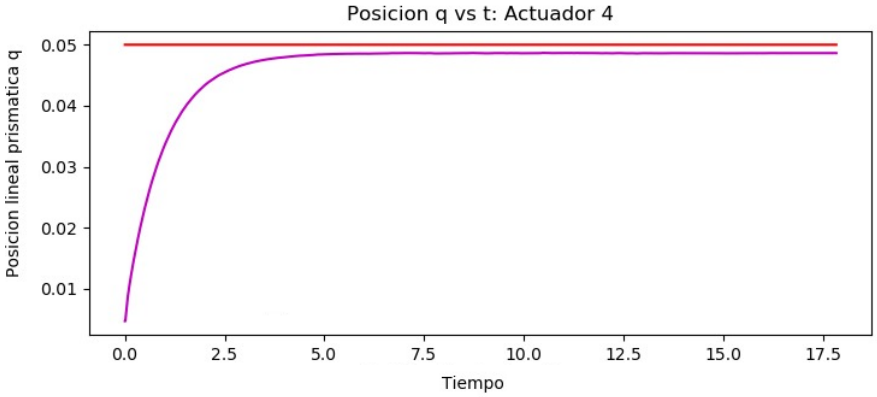
En la Tabla 3.6, se puede apreciar como se fueron sintonizando los diferentes parametros en variadas pruebas realizadas con el robot, a continuacion se muestran los resultados obtenidos durante algunas de las pruebas realizadas con los valores tabulados.



a) Kp=3000, ki=0, kd=0



b) Kp=3000, ki=0.1, kd=0



c) Kp=3000, ki=0.1, kd=30

Figura 3.7 Respuesta del sistema con parámetros PID para el mast central

Como se puede apreciar en la Figura 3.7, en c), los parámetros que se obtuvieron para obtener la mejor respuesta del sistema y llegar a estabilizar la posición es con kp=3000, ki=0.1 y kd=30, de esta manera se realizó la correcta sintonización para poder obtener un control óptimo de las articulaciones prismáticas del robot paralelo 3UPS-1RPU.

3.1.4 COMPROBACIÓN DEL CONTROL PID CONJUNTO

Una vez que se obtuvo la sintonización de los controladores PID individuales para cada uno de los actuadores, se implementó un controlador conjunto del tipo PID, para poder llevar al sistema o plataforma móvil a una posición en específico. Para esta sintonización se colocó de nuevo la plataforma móvil sobre el robot además de establecer o cerrar las cadenas cinemáticas, para de esta forma recibir la retroalimentación de movimiento.

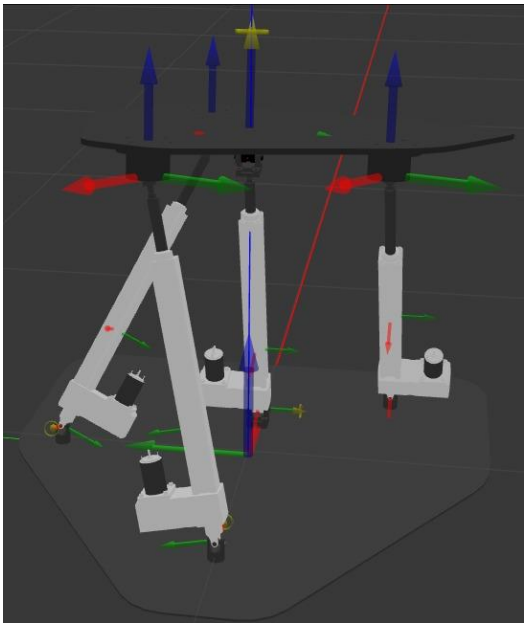


Figura 3.8 Colocación de la plataforma móvil y cierre de cadenas cinemáticas

Como se puede observar en la Figura 3.8, se utilizó el robot completamente armado e incluidos ya los controladores individuales para proceder a sintonizar le controlador PID conjunto con el fin de que se pueda movilizar la plataforma móvil, y los 4 actuadores en conjunto puedan llegar a la referencia dada. Para esto se envió a los actuadores un vector con las diferentes posiciones, que estas deben alcanzar, de manera de que la plataforma se posicione, para esto al igual que en la sintonización individual se realizado mediante prueba y error una tabulación de valores para poder establecer las ganancias del controlador conjunto.

Tabla 3.7 Tabulación de valores Kp. Ki y Kd para sintonización del controlador PID conjunto

N° Prueba	Kp	Ki	Kd
1	100	0	0
2	500	0	0
3	800	0	0
4	1200	0	0
5	1500	0	0
6	1500	1	0
7	1500	0.1	0
8	1500	0.01	0
9	1500	0.1	10
10	1500	0.1	20

Al igual que en el caso de la sintonización individual, la Tabla 3.7, presenta los valores más aproximados con los cuales se realizaron las diferentes pruebas, llegando así a la conclusión de que los parámetros adecuados para el correcto posicionamiento de la plataforma móvil son kp=1500, ki=0.1 y kd=20. De esa manera procedimos a dar un vector de valores de referencia para ubicar a la plataforma móvil a la posición inicial de trayectoria que es utilizada por el robot real, dicho vector de datos se obtuvo de variadas pruebas realizadas con el robot real en el proyecto de investigación PIGR-20-01.

El vector de posición deseada que se utilizó para la prueba es el siguiente:

$$qd = [0.083, 0.1, 0.15, 0.065] \qquad (3.1)$$

Una vez ingresado el vector de posiciones deseadas hacia el controlador PID de posición conjunta, se pueden observar en la Figura 3.9, que los actuadores alcanzaron de manera satisfactoria las referencias dadas, con un pequeño error, además se puede observar como en el grafico correspondiente al actuador 4, se nota que es el que mas error tuvo, y esto se debe a que se trata de la articulación o actuador central, mismo que soporta el peso del movimiento de los actuadores o patas externas, por este motivo estos ejercen presión sobre el vástago central, haciendo muy difícil que este llegue a la referencia de manera exacta. Aunque a pesar de ello lo hace de manera satisfactoria con un error mínimo.

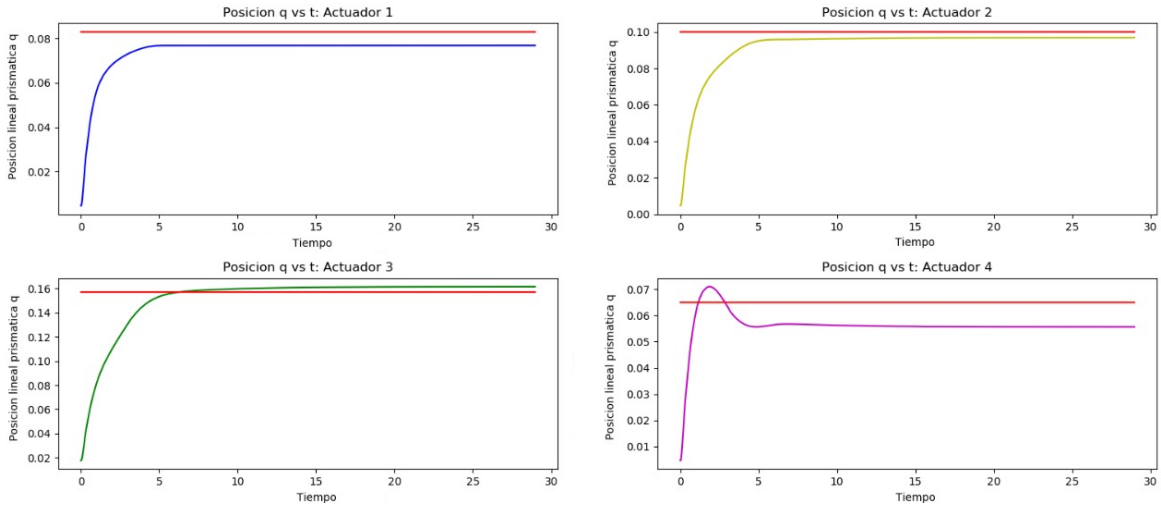


Figura 3.9 Ubicación de los actuadores en las posiciones q deseadas.

A continuación, se presenta la posición que se alcanzó durante la simulación por el modelo virtual del robot paralelo 3UPS-1RPU, mismo que se presenta como una serie de capturas en la Figura 3.10.

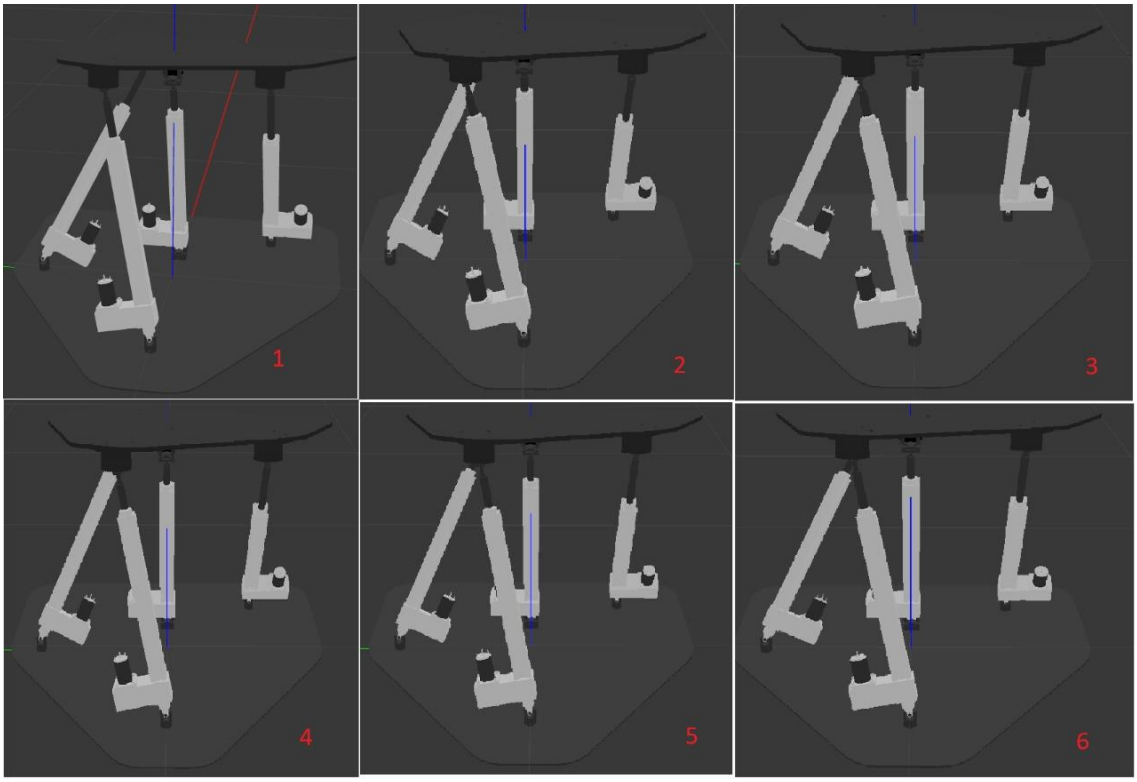


Figura 3.10 Posición alcanzada por la plataforma móvil

Una vez que se pudo observar el alcance de la posición de los vástagos sin inconveniente se puede decir que la plataforma móvil se ubica de manera correcta.

A continuación, una vez que la plataforma móvil alcanzo la posición deseada, se hizo uso de las herramientas de GAZEBO mencionadas en secciones anteriores, para poder ingresar perturbaciones al sistema y verificar que el controlador de estado conjunto permita mantener la posición alcanzada por la plataforma móvil.

3.1.4.1 Ingreso de perturbaciones al sistema

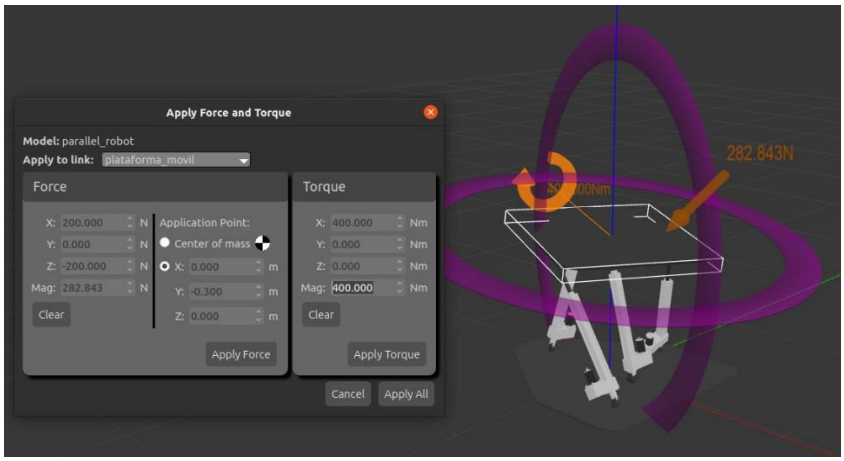


Figura 3.11 Ingreso de perturbaciones al sistema.

Una vez que la plataforma móvil se posiciono correctamente, se ingresaron fuerzas externas como perturbaciones al sistema tal y como se puede observar en la Figura 3.11, y a continuación se presentan los siguientes resultados obtenidos del controlador con respecto a las perturbaciones mencionadas:

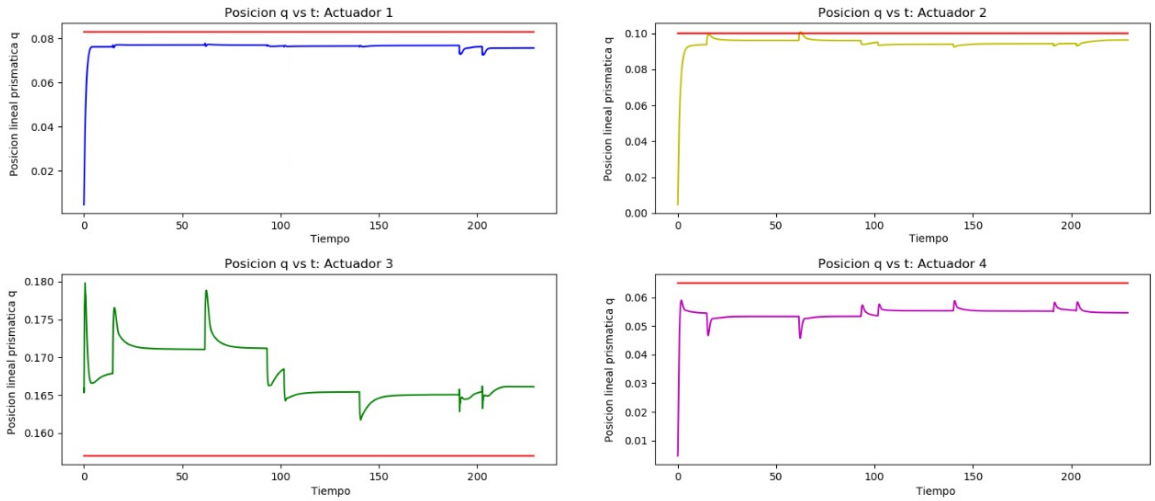


Figura 3.12 Respuestas del sistema ante entrada de perturbaciones

Como se puede observar en la Figura 3.12, las perturbaciones fueron anuladas de manera correcta por el controlador PID, mismo que permitió que el robot permaneciera estable y se mantenga en la posición enviada, para poder obtener una mayor mejora en la estabilidad del sistema ante perturbaciones de mayor tamaño se podría optar por diversos métodos de control, que podrían implementarse en conjunto con os nodos realizados para el actual proyecto.

Como podemos observar debido a la plataforma móvil, misma que se encuentra unida a todos los actuadores, en ocasiones la misma tiende a inclinarse a un solo lado debido a sus características dinámicas como el peso, por ello a veces el peso del resto de actuadores se concentra en una única articulación dificultando de mayor manera su control, por ello como se ve en la Figura 3.12, el actuador 3, presenta mayor dificultad para establecerse esto debido a lo mencionado anteriormente.

3.2 CONCLUSIONES

- El robot paralelo 3UPS-1RPU, posee una gran estabilidad debido a la disposición de sus patas y la existencia de varias cadenas cinemáticas cerradas, por lo que presenta grandes ventajas al momento de ser utilizada para la rehabilitación del miembro inferior, ya que su plataforma móvil permite ubicarse en posiciones que

no fuerzan en alto grado a las articulaciones que intervienen en el movimiento de rehabilitación.

- El uso de ROS2 para la implementación del robot paralelo, brinda una amplia gama de librerías y acepta una amplia gama de archivos con diferentes lenguajes de programación, que facilita estructurar el robot incluyendo sus características dinámicas, permitiendo simularlo de manera correcta en GAZEBO, para su correcto modelado es imprescindible obtener datos dinámicos que asemejen las características del robot real.

- El uso modular de archivos XACRO que permite utilizar ROS2, permite simplificar el trabajo de implementación, dado que, en el caso particular del robot paralelo, este posee 3 patas con características similares difiriendo solo en su orientación, por ello gracias al archivo XACRO, se puede hacer uso de la misma extremidad alterando únicamente los parámetros de ubicación desde el archivo principal.

- El marco ros2_control, permite obtener una ayuda para acceder al hardware virtual del robot paralelo, gracias al uso de sus controladores preestablecidos que se adaptan a las características del robot descrito, para luego realizar un control externo que permita controlar las articulaciones prismáticas de tal manera que la posición alcanzada sea la establecida reduciendo su error a lo más mínimo posible.

- La utilización de las herramientas de GAZEBO para realizar diversas pruebas con el robot virtual con respecto a su posicionamiento, brinda opciones de interfaces gráficas amigables al usuario además de un reseteo del mundo virtual, que permiten poner a prueba el comportamiento del robot ante diferentes fuerzas y a diferentes partes del robot sin necesidad de tener que reiniciar los procesos directamente desde el terminal.

3.3 RECOMENDACIONES

- Hacer uso de las diferentes herramientas gráficas que ofrece ROS2, como RVIZ2 y MOVELT, permitiría realizar un manejo más práctico del robot, pero para ello se deben analizar la implementación de características como el uso de varias cadenas cinemáticas cerradas en dichos simuladores.

- Para poder evitar singularidades que se deben a las limitaciones de las articulaciones del robot, se debe considerar realizar una interfaz de usuario, que permita modificar la redistribución de las patas del robot, a una predisposición

adecuada que permita evitar dichas singularidades, para el actual trabajo lo recomendable es hacer uso de posiciones o trayectorias ya probadas en otras simulaciones parte del proyecto de investigación PIGR-20-01, con el fin de esquivar dichas singularidades.

- Para el actual trabajo de integración curricular, se podrían generar diversas interfaces de usuario que faciliten la navegación entre programas así como la implementación de paquetes demos para la demostración de la correcta utilización de los paquetes preestablecidos a utilizar, como lo son los pertenecientes al marco de control ros2_control.

4. REFERENCIAS BIBLIOGRÁFICAS

[1] "GitHub: Where the world builds software · GitHub." https://github.com/ (accessed Nov. 28, 2021).

[2] C. C. Cárdenas Rodríguez and A. S. Méndez Chipatecua, "Documentación de aplicaciones robóticas en ROS," Jan. 2020, Accessed: Nov. 28, 2021. [Online]. Available: http://repository.unipiloto.edu.co/handle/20.500.12277/6889.

[3] D. Nedosseikine, "Modelado cinemático y dinámico de un robot paralelo con arquitectura 3UPS-RPU. Aplicación en la simulación de robots de rehabilitación de miembro inferior," Sep. 2021, Accessed: Nov. 28, 2021. [Online]. Available: https://riunet.upv.es/handle/10251/172609.

[4] "Reseña histórica de los Robots Paralelos." https://www.milenio.com/opinion/varios-autores/universidad-politecnica-de-tulancingo/resena-historica-de-los-robots-paralelos (accessed Nov. 28, 2021).

[5] K. Duarte, C. Borrás, and H. Vacca -Visión Electrónica, "Generalidades de robots paralelos," *Visión electrónica*, vol. 10, Jun. 2016, doi: 10.14483/22484728.11711.

[6] R. Mª Gómez Labrador Mayo, "INTRODUCCIÓN A LINUX," Accessed: Dec. 07, 2021. [Online].

[7] "Documentación ROS 2 - Documentación ROS 2: Documentación galáctica." https://docs.ros.org/en/galactic/index.html (accessed Jan. 06, 2022).

[8] P. Fin De Grado, M. Vargas, and V. Sevilla, "Equation Chapter 1 Section 1," 2017, Accessed: Dec. 07, 2021. [Online].

[9] “Comprender los nodos de ROS 2 — Documentación de ROS 2: Documentación galáctica.” https://docs.ros.org/en/galactic/Tutorials/Understanding-ROS2-Nodes.html (accessed Feb. 08, 2022).

[10] “Entorno de simulación ROS/Gazebo - PDF Free Download.” https://docplayer.es/97098786-Entorno-de-simulacion-ros-gazebo.html (accessed Jan. 06, 2022).

[11] “Control y robótica | Blog de Tecnología – IES José Arencibia Gil – Telde.” https://www3.gobiernodecanarias.org/medusa/ecoblog/fsancac/2015/02/04/control-y-robotica/ (accessed Dec. 15, 2021).

[12] C. de Investigación, Y. Desarrollo, and D. T. Digital, “INSTITUTO POLITÉCNICO NACIONAL ‘DISEÑO DE CONTROLADORES CON COMPENSACIÓN ADAPTABLE DE GRAVEDAD PARA ROBOTS MANIPULADORES.,’” Accessed: Dec. 15, 2021. [Online].

[13] “PID CONTROL – Blog de robótica y programación.” http://blog.pucp.edu.pe/blog/robotica/2018/02/17/pid-control/ (accessed Dec. 15, 2021).

[14] “Tema: Control en Cascada by Ing Electronica.” https://prezi.com/5i5kfq3igs4z/tema-control-en-cascada/ (accessed Jan. 06, 2022).

[15] K. Duarte Barón and C. Borrás Pinilla, “Generalidades de robots paralelos,” *Visión electrónica*, vol. 10, no. 1, pp. 102–112, Jun. 2016, doi: 10.14483/22484728.11711.

[16] J. L. Pulloquinga Zapata, “Análisis de compatibilidad del modelo dinámico de un robot paralelo tipo 3UPS+1RPU para rehabilitación de rodilla, mediante simulación de mecanismos en MSC-ADAMS,” Jul. 2018, Accessed: Feb. 07, 2022. [Online]. Available: http://bibdigital.epn.edu.ec/handle/15000/19567.

[17] *Angulos de Euler y de navegacion*. Departamento de Fisica Aplicada, Universidad de Sevilla.

[18] “xacro - ROS Wiki.” http://wiki.ros.org/xacro (accessed Jan. 07, 2022).

[19] “urdf/XML - ROS Wiki.” http://wiki.ros.org/urdf/XML (accessed Jan. 07, 2022).

[20] “Blender 2.80: Removed Features,” *Blender wiki*, Accessed: Feb. 14, 2022. [Online]. Available: https://wiki.blender.org/wiki/Reference/Release_Notes/2.80/Removed_Features.

[21] “¿Qué es un archivo DAE y cómo se abre uno? | TecnoNautas.” https://tecnonautas.net/que-es-un-archivo-dae-y-como-se-abre-uno/ (accessed Feb. 07, 2022).

[22] “es/roslaunch/XML/launch - ROS Wiki.” http://wiki.ros.org/es/roslaunch/XML/launch (accessed Jan. 07, 2022).

[23] “GitHub - ros-controls/ros2_control_demos: This repository aims at providing examples to illustrate ros2_control and ros2_controllers.” https://github.com/ros-controls/ros2_control_demos (accessed Feb. 07, 2022).

[24] “Welcome to the ros2_control documentation! — control.ros.org documentation.” http://control.ros.org/ (accessed Jan. 07, 2022).

[25] “Installing ROS 2 via Debian Packages — ROS 2 Documentation: Galactic documentation.” https://docs.ros.org/en/galactic/Installation/Ubuntu-Install-Debians.html (accessed Jan. 07, 2022).

[26] “Gazebo: Tutorial: Instalación de gazebo_ros_pkgs (ROS 2).” http://gazebosim.org/tutorials?tut=ros2_installing&cat=connect_ros (accessed Jan. 07, 2022).

[27] “Gazebo : Tutorial : Ubuntu.” https://classic.gazebosim.org/tutorials?tut=install_ubuntu (accessed Jun. 21, 2022).

5. ANEXOS

ANEXO I. Instalación de ROS2 Galactic Gouchelone y GAZEBO.

ANEXO II. Descripción URDF

ANEXO III Manual de usuario

ANEXO IV Propiedades dinámicas de los componentes del robot paralelo 3UPS-1RPU

ANEXO V. Instalación de paquetes y plugin para hacer uso de ros2_control en conjunto con GAZEBO.

ANEXO VI Enlaces a los archivos del PFD.

ANEXO I

Instalación ROS2 Galactic Gouchelone.

Para realizar la instalación en el sistema operativo Linux 20.04 LTS, de ROS2 Galactic Gouchelone, no disponemos de una única forma de instalación, pero la más practica y recomendable para ser instalada de manera inmediata en el computador es mediante el uso de paquetes Debian, que son repositorios colgados en línea donde el usuario podrá acceder fácilmente, mediante el terminal de su computador con sistema operativo Linux.

Primero se deberá establecer una configuración regional para que se pueda admitir el formato de codificación UTF-8 mediante el uso del os siguientes comandos [25]:

```
locale  # check for UTF-8

sudo apt update && sudo apt install locales
sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8

locale  # verify settings
```

A continuación, Deberá habilitar el repositorio Universal para la instalación de los paquetes con los siguientes comandos:

```
sudo apt install software-properties-common
sudo add-apt-repository universe
```

Una vez realizados estos pasos se deberá instalar finalmente el paquete ROS2 con la distribución Galactic Gouchelone, haciendo uso de los siguientes comandos:

```
sudo apt update
sudo apt install ros-galactic-desktop
sudo apt install ros-galactic-ros-base
```

Para poder hacer uso de los comandos, paquetes, tutoriales y demás utilidades de ROS2 en cada nuevo terminal deberá obtener la fuente de trabajo con el siguiente comando y ya podrá hacer uso de ROS2 en su computador.

```
source /opt/ros/galactic/setup.bash
```

Para obtener información más detallada del proceso puede dirigirse a:

https://docs.ros.org/en/galactic/index.html

Instalación GAZEBO 11:

Una vez que ya se haya realizado la instalación de ROS2 sin ningún inconveniente en su computador, se puede realizar la instalación de GAZEBO haciendo uso al igual que en el caso anterior de paquetes DEBIAN, para ello en un nuevo terminal se deben ejecutar las siguientes líneas de comando que instalaran el [26] simulador GAZEBO11 a más de ciertas opciones de desarrollador [27], para obtener un mayor número de herramientas en el sistema.

```
sudo apt-get install gazebo11
# For developers that work on top of Gazebo, one extra package
sudo apt-get install libgazebo11-dev
```

Una vez instalado correctamente el simulador GAZEBO se debe comprobar que este se ejecute sin ningún inconveniente en el sistema, haciendo uso de un nuevo terminal en el cual se ejecutara el siguiente comando:

```
gazebo
```

Cuando se haya verificado que el simulador GAZEBO se ejecuta de manera correcta dentro del sistema, se procederá a instalar los paquetes requeridos para poder vincular ROS2 con el simulador GAZEBO, y así poder colocar el modelo realizado mediante los archivos específicos de ROS en el simulador mencionado, esto se logra ejecutando el siguiente comando en un nuevo terminal [26].

```
sudo apt install ros-galactic-gazebo-ros-pkgs
```

Una vez que se haya ejecutado el comando el entorno de ROS2 y GAZEBO están listos para ser utilizados y poder realizar la implementación del robot paralelo con estructura 3UPS-1RPU.

Si el procedimiento descrito anteriormente, no le permite realizar la instalación de GAZEBO o tiene inconvenientes con el uso de los paquetes que vinculan ROS2 con GAZEBO, se puede optar por otro método de instalación más completa del mismo y de su última versión dirigiéndose hacia [26], [27].

ANEXO II

Descripción URDF

La estructura básica que compone a un URDF divide al robot en 4 partes principales que son: los eslabones (links), las articulaciones (joints), la descripción de control (ros2_control) y los comandos o instrucciones a seguir dentro del entorno de Gazebo. El conjunto de todas estas estructuras permite generar el modelo completo de un robot. La estructura descrita anteriormente se presenta de forma simplificada en la Figura II.1

```
<robot name="nombre_del_robot">
    <link> ... </link>
    <joint> ... </joint>
    <transmission> ... </transmission>
    <gazebo> ... </gazebo>
</robot>
```

Figura II.1 Estructura URDF simplificada de un robot

Links: Define las características de los eslabones que conforman el robot.

Joint: Define el tipo de articulaciones a ser utilizadas por el robot.

Control: Descripción del hardware utilizado y las articulaciones en específico que van a ser controladas, mediante el uso del paquete de ROS2 denominado ros2_control.

Gazebo: Comandos requeridos para configurar el modelo durante la simulación en Gazebo.

El código anterior presenta subestructuras ligados a los elementos principales. A continuación, se detallará como se conforman dichos subprocesos, haciendo uso de la estructura implementada para el robot paralelo 3UPS-1RPU:

Estructura de Links:

```
<link name="plataforma_fija">

    <inertial>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <mass value="1"/>
        <inertia ixx="1" ixy="1" ixz="1" iyy="1" iyz="1" izz="1"/>
    </inertial>

    <collision>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <geometry>
            <mesh filename="package://parallel_robot/meshes/plataforma_fija.dae"/>
        </geometry>
    </collision>

    <visual>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <geometry>
            <mesh filename="package://parallel_robot/meshes/plataforma_fija.dae"/>
        </geometry>
    </visual>

</link>
```

Figura 5.1 Estructura de un Link

Los subcomponentes que conforman un Link se pueden observar en la Figura 2.5, los cuales se describirán a continuación:

Inertial: configura la posición en la que se ubica el centro de masa del eslabón, el valor de su masa en kilogramos y la matriz de inercias que describe la pieza.

Collision: Configura la propiedad de colision existente entre la geometría de una pieza consecuente con la otra.

Visual: Configura el aspecto visual del eslabón, el cual puede ser representado por cuerpos básicos o por estructuras más complejas definidas en un archivo externo.

Las subelementos visual y colisión del ejemplo emplean un archivo de formato DAE, el cual es un archivo que contiene la estructura de un objeto en 3D, para ser representado durante la simulación, este puede ser modificado según los requerimientos de visualización mediante el software BLENDER, mismo que se puede instalar en Linux 20.04. LTS [19].

Estructura de un Joint

El elemento joint define la articulación que conecta dos eslabones como se mencionó anteriormente, para su utilización requiere de un nombre y tipo el tipo de articulación con la que se conectan los eslabones como se observa en la Figura II.2 [19].

```
<joint name="joint_1" type="revolute">

    <origin  xyz="0 0 0" rpy="0 0 0"/>
    <parent link="link_padre"/>
    <child link="link_hijo"/>
    <axis xyz="0 1 0"/>
    <limit velocity="1" effort="5.0"/>
    <dynamics damping="1" friction="1" />
    <limit lower="${-pi/2}" upper="${pi/2}" velocity="1" effort="100"/>

</joint>
```

Figura II.2 Estructura de un Joint

El tipo de articulacion a utilizar puede ser definido en función del tipo de articulaciones existentes que se revisó en capítulos anteriores, recordando que para el robot a utilizar se hará uso principal de articulaciones del tipo rotacional, universal, esférica y prismática.

Subelementos de un Joint:

Origin: Referencia u origen de la articulación respecto al eslabón padre.

Parent: Eslabón de referencia a donde se conecta la articulación (Donde la articulación va a ser fija).

Child: eslabón hijo que se conecta a la articulación (Donde permite movimiento).

Axis: Ejes en donde actúa la articulación.

Dynamics: Parámetros dinámicos de la articulación, como la fricción que posee.

Limit: Límites de parámetros a definir como la velocidad o esfuerzo que soporta o alcanza la articulación, además, define el límite máximo y mínimo de movimiento a los que puede llegar.

Estructura de Control

Para el actual robot la etiqueta de control a utilizar dentro de la descripción URDF es la denominada ros2_control, misma que define el tipo de hardware a controlar, así como las interfaces de retroalimentación o estado y el tipo de acción de comando que se va a enviar, tanto para la interfaz de estado y comando podemos elegir entre velocidad, esfuerzo y posición.

Estructuras Gazebo

El elemento Gazebo contiene parámetros que configuran las articulaciones o los eslabones directamente dentro del entorno de simulación GAZEBO, estos parámetros

están definidos dentro de la descripción URDF como se muestra a continuación en la Figura II.3.

```
<gazebo>

    <plugin name="gazebo_ros_controller" filename="libgazebo_ros_control.so" >
        <robotNamespace>/parallel_robot</robotNamespace>
    </plugin>

</gazebo>
```

Figura II.3 Estructura Gazebo

ANEXO III

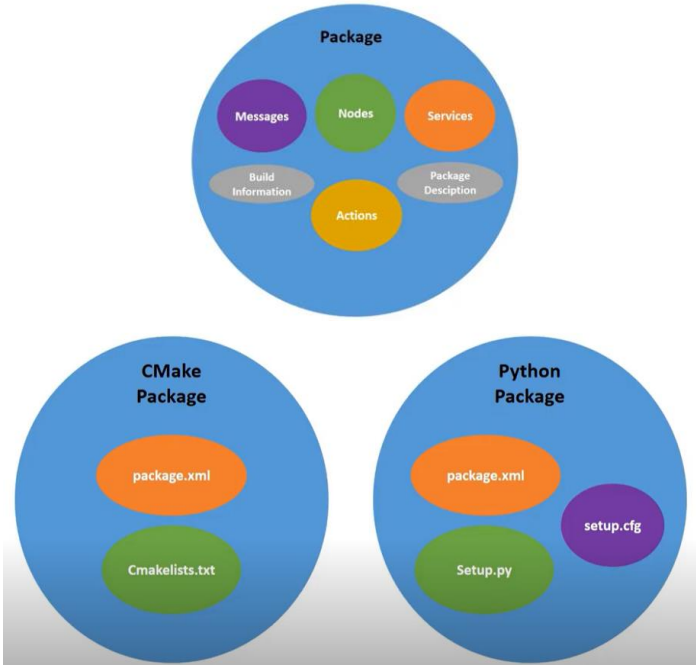
Manual de usuario:

Manual de usuario para el uso del robot paralelo 3UPS-1RPU

Manejo de ROS2

Ahora se aprenderá los elementos básicos a manejar en el entorno de ROS2, lo principal a manejar en este es el uso de paquetes, mismos que se ejecutaran dentro de un espacio de trabajo.

Un paquete es un contenedor donde ubicaremos todos los mensajes, nodos, servicios, etc que permitirán poner en funcionamiento nuestro proyecto, los paquetes pueden ser construidos ya sea utilizando lenguaje C++ o Phyton, esto deberá distinguirse al momento de crear el paquete.



Comandos para obtener espacio de trabajo:

`source /opt/ros/galactic/setup.bash` ##Inicialisamos con la fuente de uestro ROS

`source install/local_setup.bash` ##Obtenemos espacio de trabajo

Comandos utilizados para la creación de paquetes en C++ o Phyton:

```
ros2 pkg create --build-type ament_cmake <package_name>
```

```
ros2 pkg create --build-type ament_phyton <package_name>
```

Comandos para compilación de proyectos:

```
colcon build --packages-select my_package
```

```
colcon build
```

Dentro de los paquetes creados, se debe tomar en cuenta ciertos conceptos básicos que intervienen en el uso de programas dentro de ROS2:

Nodos:

Cada uno de los nodos implementados debe ser responsable de una tarea en específico del robot, y para poder intercomunicar los diferentes nodos se puede hacer uso de tópicos, servicios, acciones o parámetros.

Un robot o sistema robótico es en cierta forma un conjunto de nodos entrelazados y encargados de diferentes tareas, mediante un archivo de lanzamiento o launch, se pueden ejecutar uno o varios nodos de la forma que se requiera.

Tópicos:

Los tópicos en ros se traducen como un bus de datos o medio de transporte por donde viaja la información entre nodos, un nodo puede comportarse como publicador o suscriptor de información en diversos tópicos.

Creación del espacio de trabajo en ROS2 para el actual proyecto:

Creación de un nuevo directorio

```
mkdir -p ~/dev_ws/src
```

```
cd ~/dev_ws/src
```

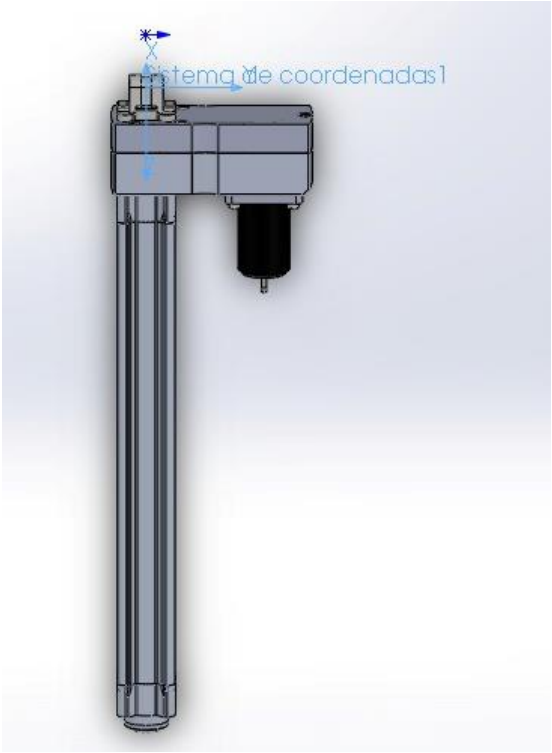
Nota: Recordemos que todos los paquetes que se creara para el proyecto deben ser ubicados dentro de la carpeta fuente del espacio de trabajo “src”.

Enlace manual completo: Manual de usuario_3UPS1RPU_PFD_Catota Jonathan.docx

ANEXO IV

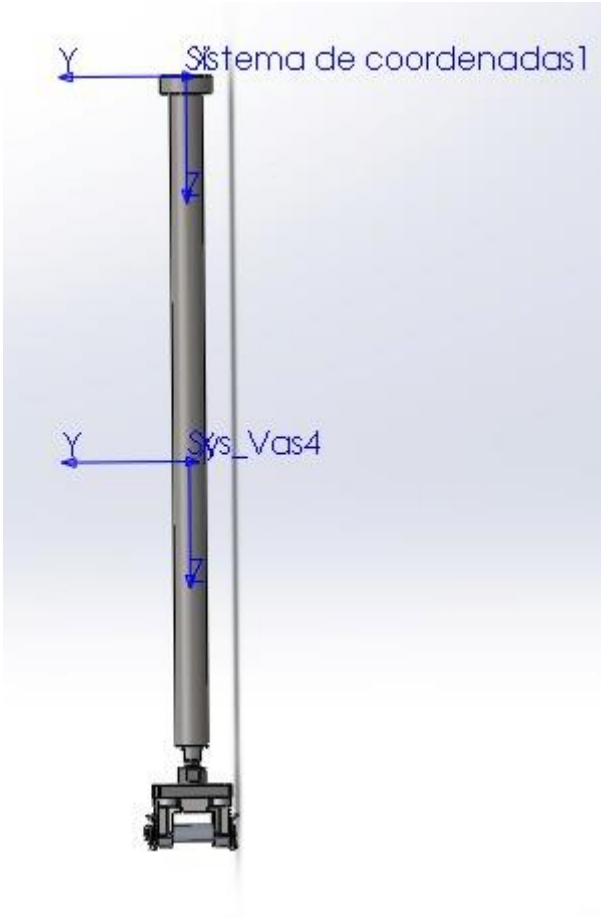
Propiedades dinámicas componentes del Robot Paralelo 3UPS-1RPU

Cilindro Central



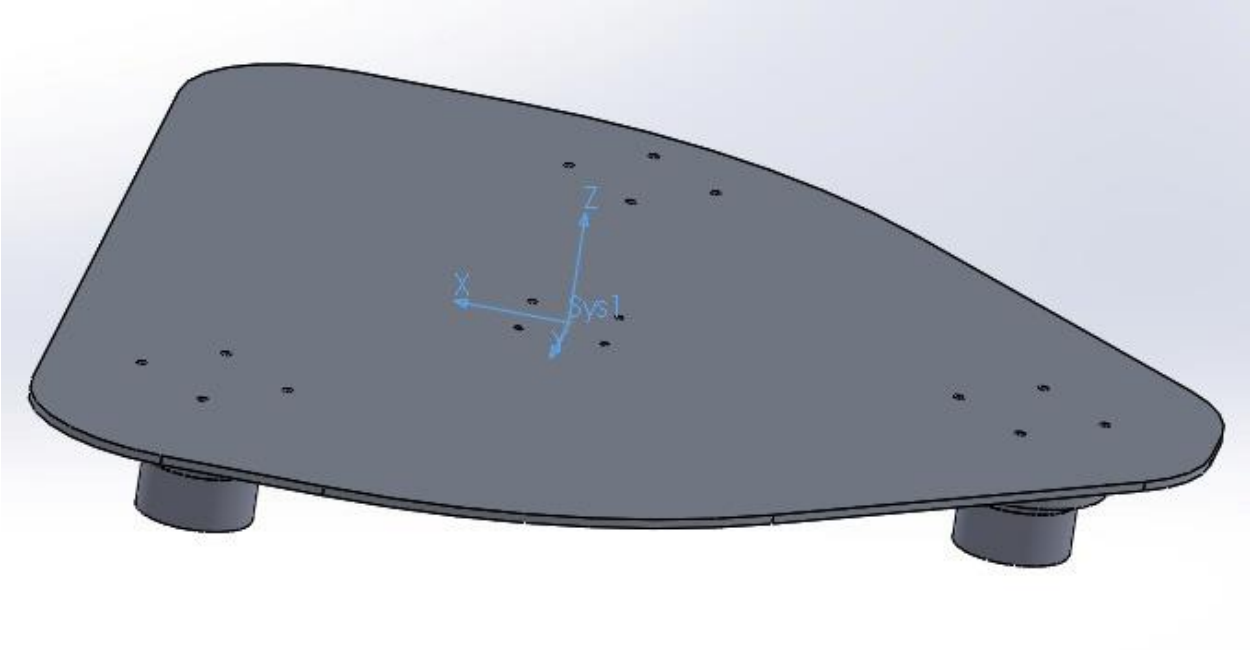
Propiedades Dinámicas			
Peso [Kg]	2.1656		
Centro de masa [m]	X: 0.0011	Y: 0.0283	Z: 0.1812
Momentos de inercia con respecto a los sistemas de referencia [Kg.m^2]	Ixx: 0.1280	Ixy: 0.0	Ixz: 0.0007
	Iyx: 0.0	Iyy: 0.1226	Iyz: 0.0046
	Izx: 0.0007	Izy: 0.0046	Izz: 0.0066

Vástago central 4



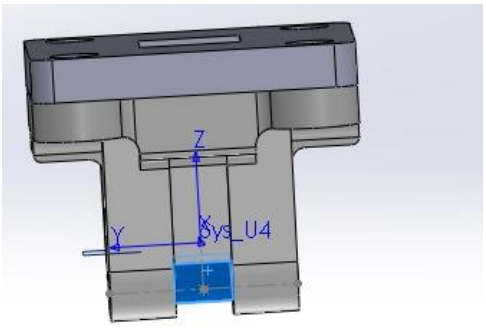
Propiedades Dinámicas			
Peso [Kg]	1.3915		
Centro de masa [m]	X: 0.0	Y: 0.0	Z: 0.2539
Momentos de inercia con respecto a los sistemas de referencia [Kg.m^2]	Ixx: 0.1211	Ixy: 0.0	Ixz: 0.0
	Iyx: 0.0	Iyy: 0.1211	Iyz: 0.0
	Izx: 0.0	Izy: 0.0	Izz: 0.0001

Plataforma Móvil



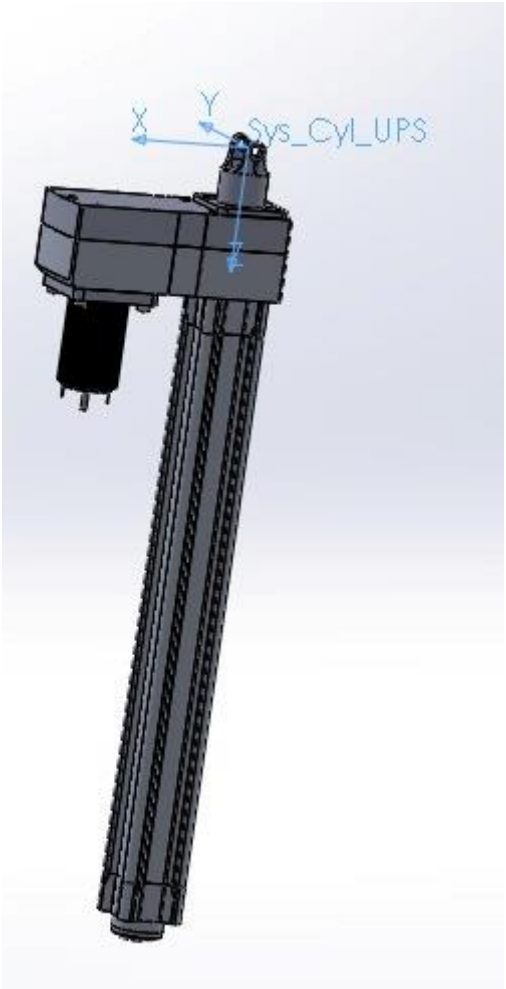
Propiedades Dinámicas			
Peso [Kg]	10.2553		
Centro de masa [m]	X: -0.0090	Y: -0.0068	Z: 0.0514
Momentos de inercia con respecto a los sistemas de referencia [Kg.m^2]	Ixx: 0.4772	Ixy: 0.0437	Ixz: -0.0024
	Iyx: 0.0437	Iyy: 0.4006	Iyz: -0.0021
	Izx: 0.0024	Izy: -0.0021	Izz: 0.8177

Unión 4



Propiedades Dinámicas			
Peso [Kg]	0.028409		
Centro de masa [m]	X: -0.000947	Y: 0.0	Z: 0.01579
Momentos de inercia con respecto a los sistemas de referencia [Kg.m^2]	Ixx: 0.000014	Ixy: 0.0	Ixz: 0.0
	Iyx: 0.0	Iyy: 0.000013	Iyz: 0.0
	Izx: 0.0	Izy: 0.0	Izz: 0.000008

Cilindro Externo UPS:



Propiedades Dinámicas			
Peso [Kg]	2.1929		
Centro de masa [m]	X: 0.0280	Y: 0.0011	Z: 0.1986
Momentos de inercia con respecto a los sistemas de referencia [Kg.m^2]	Ixx: 0.1388	Ixy: 0.0	Ixz: 0.0058
	Iyx: 0.0	Iyy: 0.1442	Iyz: 0.0007
	Izx: 0.0058	Izy: 0.0007	Izz: 0.0066

Vástago Externo:

Propiedades Dinámicas

Peso [Kg]	1.44		
Centro de masa [m]	X: 0.0	Y: 0.0	Z: 0.2673
Momentos de inercia con respecto a los sistemas de referencia [Kg.m^2]	Ixx: 0.1381	Ixy: 0.0	Ixz: 0.0
	Iyx: 0.0	Iyy: 0.1381	Iyz: 0.0
	Izx: 0.0	Izy: 0.0	Izz: 0.0001

ANEXO V

Instalación de paquetes y plugin para hacer uso de ros2_control en conjunto con GAZEBO.

El repositorio de ros2_control, comprende un sin número de paquetes con controladores prediseñados y herramientas útiles, para poder obtener un control preciso de cualquier robot diseñado en ROS2. Para hacer uso de estos paquetes y herramientas dentro del actual proyecto se debe dirigir primero a la carpeta de paquetes del espacio de trabajo principal haciendo uso del siguiente comando:

```
cd dev_ws/src
```

A continuación, en la carpeta src se deberá clonar los paquetes necesarios para hacer uso de ros2_control desde el repositorio oficial de GitHub, haciendo uso de las siguientes líneas de comando:

```
git clone https://github.com/ros-controls/ros2_control
git clone https://github.com/ros-controls/ros2_controllers
git clone https://github.com/ros-controls/ros2_control_demos
```

Una vez clonados los paquetes necesarios se realizará la instalación de dependencias:

```
rosdep install --from-paths src --ignore-src -r -y
```

Ahora se procederá a compilar todos los paquetes desde el directorio principal del espacio de trabajo y así ya se tendrán listos los componentes para realizar el uso del marco de control ros2_control.

```
cd dev_ws
colcon build --symlink-install
```

Plugin para incorporar ros2_control a GAZEBO

Además de los paquetes de ros2_control, se requiere de un paquete adicional para hacer uso del marco de control en conjunto con el simulador GAZEBO, por ello clonaremos el paquete de gazebo_ros2_control desde el repositorio oficial de GitHub, hacia la carpeta de paquetes de nuestro espacio de trabajo mediante los siguientes comandos:

```
cd dev_ws/src
```

```
git clone https://github.com/ros-controls/gazebo_ros2_control.git
```

Una vez que se haya clonado el paquete del plugin, solo se lo debe compilar y de esta manera ya se podrá hacer uso del mismo dentro del robot paralelo realizado para este proyecto.

```
cd dev_ws
colcon build --symlink-install
```

ANEXO VI

Enlaces a los recursos del producto final demostrable:

Video demostrativo 1: https://youtu.be/WLJQZQqTnAE

Video demostrativo 2: https://youtu.be/GkJIqskdUtQ

Video demostrativo 3: https://youtu.be/rBZBv9wfskg

PFD: PFD_TIC_3UPS-1RPU_Catota_Jonathan