Diseño y construcción de un prototipo de robot bípedo basado en dinámica simplificada Cart-Table y el sistema operativo ROS para la experimentación de trayectorias articulares de marcha *off-line*

Jaime Andres Tabango López.
Franco Esneider Gutierrez Lasso.

Universidad del Cauca.
Facultad de Ingeniería Electrónica y Telecomunicaciones.
Departamento de Electrónica, Instrumentación y Control.
Programa de Ingeniería en Automática Industrial.
Popayán.
2022

Diseño y construcción de un prototipo de robot bípedo basado en dinámica simplificada Cart-Table y el sistema operativo ROS para la experimentación de trayectorias articulares de marcha *off-line*

Trabajo de Grado, Requisito para optar al Título
de Ingeniero en Automática Industrial.

Jaime Andres Tabango López.
Franco Esneider Gutierrez Lasso.

Director:
Msc. Jeison Tacué González

Codirector:
PhD. Diego Bravo Montenegro

Universidad del Cauca.
Facultad de Ingeniería Electrónica y Telecomunicaciones.
Programa de Ingeniería en Automática Industrial.
Popayán, 2022

Agradecimientos

En primer lugar, queremos dar gracias a Dios por darnos la sabiduría, la fortaleza y la paciencia para realizar cada etapa de nuestro proyecto.

En segundo lugar, agradecer a nuestros padres por su apoyo moral, económico, por sabernos guiar con sus buenos consejos a lo largo de nuestra carrera universitaria.

A nuestro Director de tesis y amigo Jeison Tacué, como también a nuestro codirector Diego Bravo.

Por último, a todos los docentes del alma mater quienes compartieron sus conocimientos, brindarnos apoyo y sabernos orientar en este camino, para alcanzar esta nueva faceta.

Índice de contenido

Tabla de Figuras

Índice de Tablas:

Glosario

COP

Centro de presión.

ROS

Sistema operativo robótico.

PLA

Ácido poliláctico.

CoM

Centro de masa.

ZMP

Punto de momento cero.

CAD

diseño asistido por computadora,

ADC

Conversor analógico digital.

IMs

Creadores interactivos.

URDF

Formato de descripción de robot unificado.

RVIZ

Herramienta de visualización para ROS.

MIMO

Múltiples entradas y múltiples salidas.

API

Interfaz de programación de aplicaciones.

Capítulo 1: Aspectos Generales del Proyecto

1.1. Planteamiento del problema

El auge de la robótica en los últimos años ha logrado que los robots sean cada vez más comunes en ambientes no industriales, un caso particular son los robots bípedos tipo humanoide, los cuales trabajan e interactúan con el ser humano y poseen la habilidad de desempeñarse en espacios diseñados para los hombres [1]. Es así que desde hace más de una década han surgido prototipos comerciales de robots bípedos. En la actualidad se encuentran en el mercado robots de bajo presupuesto y de pequeño formato (ROBONOVA, KONDO, BIOLOID, entre muchos otros), de mediana y alta funcionalidad (Qrio, Hoap, NAO, Darwin) y en paralelo grandes proyectos de presupuestos exorbitantes (ASIMO, HUBO, HRP, ATLAS) [1]. Algunos robots como BIOLOID (ejemplar que posee el grupo de investigación en Automática Industrial), RoBoHoN, NAO, Poppy y Qrio han sido utilizados para educar, otros como el ASIMO, Geminoid F, Robothespian, Romeo y Wakamaru han sido diseñados principalmente para comunicarse con humanos, buscando proporcionar información y asistencia, otros como el ATLAS, DRC-HUBO y Petman se enfocan en brindar protección en diversos aspectos [2].

Los robots bípedos diseñados para la educación y desarrollo, son de gran utilidad para generar avances de carácter científico, sin embargo, en diversas situaciones estos dispositivos presentan limitaciones, un claro ejemplo son los humanoides BIOLOID, ROBONOVA y KONDO, si bien sus interesantes precios no rebasan los 2.000 USD, sus prestaciones físicas y de procesamiento lógico-matemático basadas en microprocesadores de baja y mediana gama son escasas y condicionadas [3], esta situación se ha evidenciado en los múltiples estudios acerca de la marcha humana implementada en robots, realizados por el grupo de investigación en Automática Industrial, e incluso en proyectos como el de la Universidad de Middlesex, en el que fue necesario adaptar un nuevo sistema embebido, tipo computadora, para dar continuidad a la experimentación de técnicas de control de equilibrio avanzadas y generación de marcha *on-line* sobre el robot BIOLOID [4], esto se debe en gran parte a que en la marcha bípeda se necesita la ejecución de algoritmos *on-line* y *off-line* para la generación de trayectorias de referencia de marcha, en donde es necesario transformar durante cada instante de muestreo la posición cartesiana de los pies en una posición articular conjunta, lo cual implica solucionar iterativamente los modelos cinemático directo e inverso del robot [5], de forma paralela, se debe ejecutar alguna técnica de control de equilibrio para efectuar la trayectoria de marcha previamente generada, por tanto, los dispositivos originales de procesamiento no son capaces de brindar una solución completa y eficiente, puesto que se requiere alta precisión de cálculo y velocidad de procesamiento [5]. Un caso particular de esta situación es la del robot BIOLOID y su procesador CM530 [6], sobre el cual únicamente se ha logrado estudiar técnicas de generación de marcha *off-line*, en los trabajos de pregrado y posgrado realizados sobre esta plataforma al interior de la Universidad del Cauca [6], [7], [8], [9]. Dicho planteamiento también ha permitido observar que la complejidad de los modelos dinámicos

del robot bípedo son una elección importante a tener en cuenta para generar técnicas robustas de marcha, ya que existe una relación proporcional entre la rigurosidad del modelo y el gasto computacional [10], [11], [12]. Los estudios realizados han indicado que al usar modelos matemáticos completos se requiere conocimiento preciso de la dinámica multicuerpo del robot, incluyendo masas, centros de masa, inercias de los actuadores y otros datos difícilmente conocidos [10], no obstante, respecto a la concordancia entre el modelo y el robot real, la reproducción de la dinámica de robots bípedos bajo este enfoque es medianamente superior en comparación al uso de enfoques como el de dinámica simplificada [7], [13], [14], [15], [16], [17]. Este último es ampliamente conocido como modelado de dinámica simplificada, y utiliza en gran parte las propiedades geométricas y cinemáticas del robot junto a modelos de sistemas físicos conocidos, como son el péndulo [18], el *Cart-Table* [14], [19], [20] entre otros, usados para generar un modelo dinámico aproximado del balance al caminar, este ha sido vastamente aceptado ante la opinión de muchos autores respecto al control y generación de marcha de robots bípedos [6], en comparación con los modelos de dinámica completa, la dinámica simplificada logra que el proceso de generación de trayectorias, cálculo matemático y uso de recursos de procesamiento sean menores, por lo cual esta brecha podría ser una buena alternativa para generar técnicas de marcha *on-line* sobre robots académicos de gama baja y media, con algunas características complementarias [5], [6], [10], [21].

Un aspecto adicional y combinatorio a la dinámica simplificada, que se desea experimentar en este proyecto, consiste en el uso del *Framework* del sistema operativo robótico (*ROS*, Robot Operating System), para el desarrollo de software para robots [22], [23], el cual ha demostrado ser una herramienta versátil a la hora de utilizarse en robots de diferente índole, puesto que ha sido diseñado para ser ligero en ejecución y procesamiento, utilizando una arquitectura de nodos distribuidos, que le permite a los algoritmos del robot ser ejecutados de forma separada por funcionalidades específicas que se ejecutan en paralelo tras ser invocadas, todo sin intervenir en la ejecución de un nodo con otro [22], [23], [24], [25]. *ROS* también incorpora un conjunto de paquetes denominados *ROS contro*l que proporcionan la capacidad de aplicar y gestionar controladores para robots con un enfoque de rendimiento en tiempo real, útil en trabajos futuros acerca de control [22], [26], [27], y además facilita la integración entre Matlab y los robots que operan bajo este *Framework* [28].

A partir del argumento planteado, acerca de la generación de marcha para robots bípedos, específicamente combinando las problemáticas de modelado dinámico, el procesamiento y el *Framework* de soporte, se ha planteado la siguiente pregunta de investigación: ¿Cuáles deben ser las características de un prototipo de robot bípedo de ámbito académico, que adopte un modelo dinámico simplificado el cual se ejecute sobre el sistema operativo *ROS* para la experimentación de trayectorias articulares de marcha *off-line*?

1.2. Objetivos

1.2.1. Objetivo general

- Proponer un robot bípedo para la experimentación de trayectorias articulares de marcha, basado en el sistema operativo ROS y el modelo dinámico simplificado *Cart-Table.*

1.2.2. Objetivos específicos

- Especificar las características electrónicas, estructurales y de software para el robot bípedo propuesto.
- Calcular los modelos matemáticos cinemático y dinámico simplificado del robot bípedo propuesto.
- Determinar experimentalmente la capacidad de seguimiento de trayectorias articulares predefinidas para el robot bípedo propuesto.

1.3. Aportes investigativos

En este proyecto se busca adaptar un modelo dinámico simplificado ampliamente estudiado en la literatura, como es el caso del modelo *Cart-Table* en la generación de trayectorias articulares *off-line* de caminata, sin considerar obstáculos en el trayecto, aplicado sobre un prototipo experimental de robot bípedo concebido bajo plataformas de hardware y software libre, como es el sistema operativo para robots ROS, con lo cual se pretende alcanzar prestaciones mejoradas de funcionalidad, cálculo computacional, posibilidad de continuar proyectos de control *on-line* y bajo costo de construcción, en comparación con las plataformas comerciales actuales como: NAO o BIOLOID Premium Kit, usado por el grupo de Investigación en Automática Industrial, para la ejecución de proyectos de investigación de pregrado y posgrado. Cabe resaltar que la propuesta de diseño del robot ha sido pensada en la realización de trabajos futuros, como, por ejemplo, el control de equilibrio, la generación de trayectorias de marcha *on-line* y la experimentación de nuevos algoritmos de control. Lo cual restringe el proyecto actual al diseño, construcción, modelado matemático y experimentación articular.

1.4. Marco teórico

1.4.1. Conceptos básicos

- **Robots Bípedos (humanoides**): Los robots humanoides son robots bípedos con apariencia general basada en el cuerpo humano [29]. La definición más restrictiva del robot humanoide indica que se trata de un robot bípedo actuado con torso, brazos y cabeza, diseñado para lograr algunas capacidades de movimiento humano [30]. Estos robots tienen orígenes que se remontan a la antigua mitología griega y durante muchos siglos se ha buscado crear máquinas artificiales similares a los humanos [31].

- **ZMP:** Punto de Momento Cero (ZMP, *Zero Moment Point),* es el momento donde las fuerzas de reacción ejercidas por la superficie plana sobre el pie del robot bípedo, neutralizan el momento total ejercido por el cuerpo del robot durante la marcha. Adicionalmente, es considerado como un criterio de estabilidad que ha sido

usado para el seguimiento de trayectorias articulares y garantizar una caminata robusta en robots bípedos, porque la existencia de este punto dentro del polígono de soporte garantiza que el robot esté en equilibrio dinámico [14], [15], [16], [17].

- **CoM:** Centro de Masa (CoM, *Center of Mass*), se encuentra dentro del robot y es el punto de mayor concentración de masa [16], [17].

- **Polígono de Soporte:** Es la figura geométrica plana formada sobre el suelo al estar uno o los dos pies del robot en contacto con el suelo, como se muestra en la Figura 1.1 [14], [17].

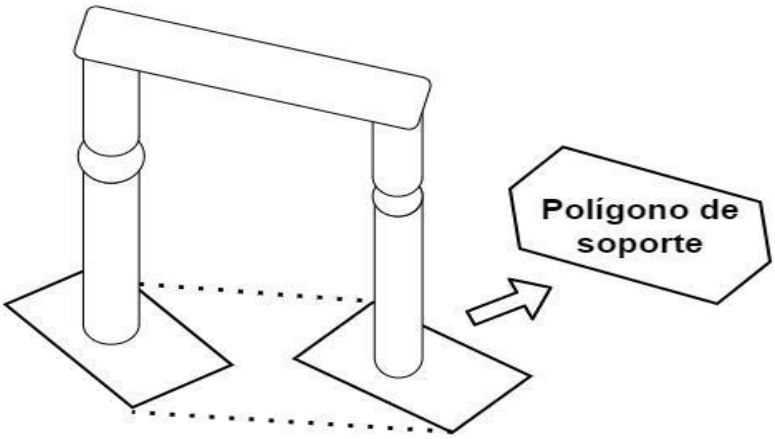


Figura 1.1 Polígono de soporte

Modelo Cinemático: El modelo cinemático describe relaciones geométricas que están presente en el sistema [32], describe las velocidades de las articulaciones del robot en el espacio cartesiano en función de las velocidades de estas articulaciones expresadas en el espacio articular [33]. El modelo cinemático directo se expresa como:

$$\dot{X} = J(q)\dot{q} \tag{1}$$

Mientras el inverso:

$$\dot{q} = J^{-1}(q)\dot{X} \tag{2}$$

Donde, $\dot{X}$ representa las velocidades de las articulaciones del robot en el espacio cartesiano, $\dot{q}$ son las velocidades articulares de las articulaciones del robot y $J(q)$ es la matriz Jacobiana [34]. Cabe resaltar que el modelo cinemático directo de primer orden permite expresar las velocidades lineal y angular mientras el de segundo orden permite expresar las aceleraciones lineal y angular de cada pie en

función del vector de aceleraciones generalizadas [35]. Adicional a esto se ocupa de la configuración de robots en su espacio de trabajo, investiga los momentos de inercia, las fuerzas y pares que producen el movimiento. Claramente, la cinemática juega un papel básico en análisis de movimiento, generación y control, debido a que es bastante empleado en algoritmos de control para el movimiento en robots [36], [37].

- **Modelo Dinámico:** Este modelo es importante para el diseño, dimensionamiento de actuadores, y evaluación de la estructura del robot bípedo [79], debido a que permite conocer los componentes que intervienen durante la marcha del robot como son: inercias, parámetros de velocidad, coriolis, entre otros. Para la solución de este modelo existen varios métodos, entre los más comunes están los métodos Newtoniano y Lagrangiano [38], [39].

- **Modelo Dinámico Simplificado**: Estos modelos buscan reducir la complejidad del estudio de la marcha humana. Es por esto que se generan patrones de caminata basados en sistemas más sencillos, como son el modelo *Cart-Table*, péndulo invertido, modelo de eslabones, entre otros. Además de ser sistemas simples para el estudio de la marcha bípeda, se ha demostrado eficiencia para la generación y control de patrones cíclicos, como también de dar estabilidad al robot bípedo al caminar [40], [41].

1.4.2. Cinemática del robot bípedo

1.4.2.1 Cinemática Directa:

El objetivo del modelo cinemático directo consiste en encontrar el valor de la posición y orientación del efector final del robot humanoide, en este caso cada uno de sus pies, conociendo los ángulos y longitudes de cada articulación que componen cada pierna [1], [3]. Este modelo puede ser determinado por el método geométrico o el método basado en cambios de sistemas de referencia, siendo este último el más usado ya que permite obtener la cinemática directa sin importar la cantidad de grados de libertad con los que cuente el robot [2].

1.4.2.2 Cinemática Inversa:

El objetivo del modelo cinemático inverso, consiste en encontrar el valor de los ángulos que debe adoptar cada articulación, para que una de las extremidades del robot alcance la posición y orientación deseada [2], [4], [5]. Para encontrar dichos valores se debe conocer la posición deseada del efector final, el número de articulaciones y la longitud de los segmentos que unen a cada articulación [3].

1.4.3. Dinámica del robot bípedo

El modelo dinámico es indispensable para evaluar y diseñar la estructura mecánica del robot, por medio de este se conoce el comportamiento dinámico del robot ante fuerzas externas a las que se ve expuesto [24], [25]. Para el cálculo de este modelo existen procedimientos de dinámica completa descritos por métodos como:

- **Método Newton-Euler:** Este método parte de la ley de conservación de par y fuerza para cada componente [22], adicional a esto permite conocer la dinámica directa e inversa del robot, de las cuales es posible establecer la posición, velocidad y la aceleración de cada una de las articulaciones del robot. Como también desarrollar sistemas de control de movimiento, generación de trayectorias y el diseño mecánico del robot [22], [24], [25].
- **Modelo Euler–Lagrange:** Este método es basado en la representación de Denavit-Hartenberg que se basa en las matrices de transformación homogénea [26], en el cual describe las ecuaciones del movimiento en términos del trabajo y de la energía del sistema [34], este método permite conocer los pares y fuerzas que intervienen en la dinámica del robot como son fuerzas de inercias, fuerzas centrífugas, efecto coriolis, gravedad, entre otras. La principal desventaja es el gasto computacional, debido a que el número de operaciones crece de manera exponencial en relación con el número de grados de libertad (GDL) [22], [25].

  De forma alterna existen procedimientos de dinámica simplificada, con los cuales se puede describir la dinámica del robot, a continuación, se conceptualizan algunos de estos:

- **Método de Espacio Articular:** Este método consiste en calcular trayectorias temporales entre los límites articulares definidos con anticipación para los movimientos durante la marcha. Este método fue usado en el robot LEROI, donde se presenta una forma simple de calcular los cambios angulares en la marcha del bípedo, para esto se hace necesario conocer la longitud deseada del paso, la velocidad deseada para dar el paso y el tipo de movimiento (caminar en superficie plana o inclinada) [42].

- **Modelo Carro Mesa (*Cart-Table*):** Este modelo se plantea en el plano horizontal y se utiliza principalmente en la planificación y el control del movimiento de robots bípedos mediante el control del ZMP [26] ,[27]. Este modelo consiste en un móvil (carro) que se mueve a lo largo de una mesa [23], [6] como se aprecia en Figura 1.2. El propósito de este modelo consiste en encontrar el punto *p* sobre el suelo, donde la sumatoria de los torques debidos a las fuerzas de reacción del suelo contra el pie sea nula, es decir donde el ZMP es estable [27], [6].

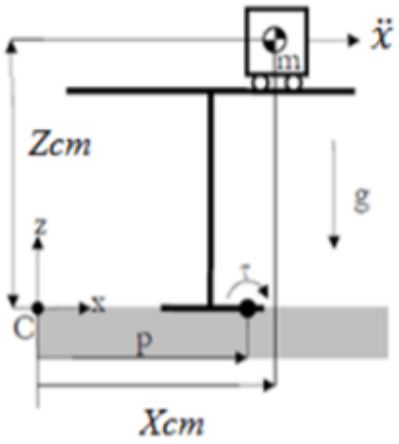


Figura 1.2. Modelo Cart-Table [6]

- **Péndulo Invertido:** Es el modelo más simple para la representación de la marcha humana introducido por Kajita y Tani en 1991 [23], [26], [27]. Este modelo permite entender el movimiento durante la fase de apoyo en la marcha [28]. La desventaja de este método es la dinámica no lineal dada por la inclusión de la función trigonométrica seno, ya que restringe algunas técnicas de control [27].

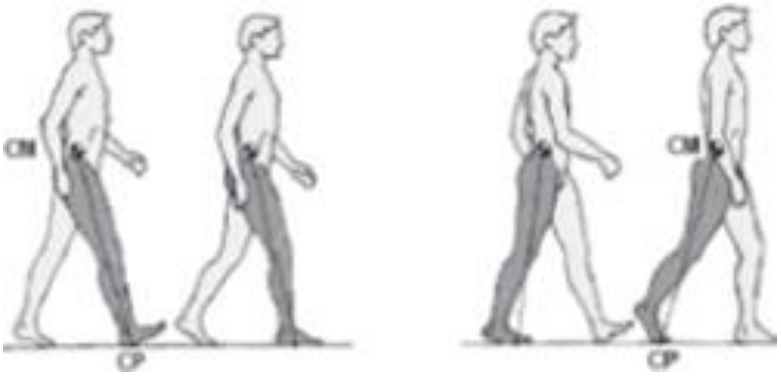


Figura 1.3. Ilustración del péndulo invertido en la fase de apoyo

- **Péndulo Invertido Lineal (LIP):** Este modelo determina la fuerza de la pierna para compensar la gravedad, resultando en una aceleración vertical igual a cero [43], Esta representación fue introducida por primera vez por Kajita y Tani en 1996 [44], y también se basa en el péndulo invertido, sin embargo para modelar la elasticidad de la pierna y eliminar la no linealidad del péndulo anterior se aumenta un actuador prismático, que permite al péndulo estirarse o encogerse, de tal forma que el centro de masa se restringe a un plano horizontal de altura constante [6].

- **Modelo Lineal de Péndulo Invertido 3D (3D-LIMP):** Este modelo se deriva de un péndulo invertido tradicional y es usado con frecuencia en el análisis de la marcha bípeda, ya que permite conocer aspectos importantes que intervienen en esta, por ejemplo, el límite del ZMP y el efecto de la gravedad [23], [27], [45], [46]. Además, el movimiento del 3D-LIMP se encuentra limitado en un plano arbitrariamente definido, logrando el diseño de controlador por separado para el movimiento sagital (x-z) y lateral (y-z) [46]. Por otra parte, a partir de este modelo es posible calcular las fuerzas necesarias para que el CoM del robot se encuentre en equilibrio, y compensar factores que lleven a caídas del robot [27].

1.4.4. Marcha humana implementada en robots

1.4.4.1. Locomoción humana

La locomoción humana hace referencia al aparato encargado de la movilidad de los humanos e incluso en animales [8]. Su estudio es de gran importancia en diferentes áreas de investigación como lo es el análisis fisiológico y aplicaciones médicas, en computación para desarrollar animaciones virtuales y reconocimientos virtuales, y en la rama de la robótica concerniente a desarrollos de exoesqueletos, humanoides entre otros [6], [7], [11].

1.4.4.2. Ciclo de marcha

El andar en los humanos es un proceso repetitivo, en el cual cada una de las piernas realiza un movimiento periódico, pasando de estar en apoyo a uno en movimiento [6], [7], [14]. Este ciclo inicia con el contacto de uno de los pies del robot con el piso, y termina una vez que este mismo pie vuelva a tener contacto con el suelo, esto se garantiza una vez que la pierna ha realizado la etapa de apoyo (el pie está en contacto con el piso) y la etapa de balanceo (el pie no tiene contacto con el suelo) [10], [17]. En [18] se analiza la marcha humana en cuatro fases o tiempos como se muestra en la figura 1.4, este movimiento se le puede dividir solamente en dos fases ya que el movimiento de una de las piernas es el mismo para la otra.

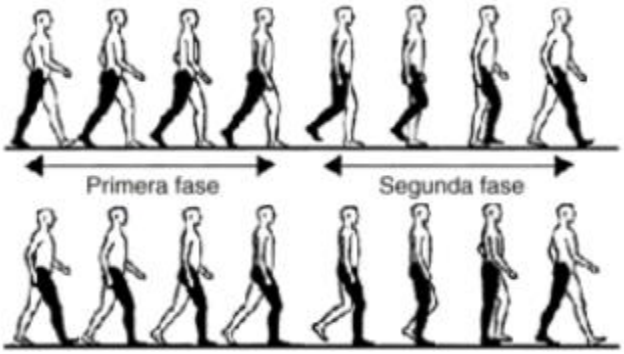


Figura 1. 4. Ciclo de marcha humana [65]

1.4.4.3. Fases de la marcha

Existen dos fases elementales en el ciclo de marcha en los humanos analizando un paso al caminar, la primera de ellas es la fase de soporte único y la segunda es la fase de doble soporte. La primera fase se distingue por estar una pierna en apoyo, soportando todo el peso del cuerpo mientras que la segunda pierna se encuentra en movimiento, y en la segunda fase las dos piernas se encuentran realizando el soporte del cuerpo [6], [9], [10], [11], [14], [16].

1.4.4.4. Locomoción Pasiva

La locomoción en robots pasivos se caracteriza principalmente por la ausencia de energía, esto se debe a que no usa actuadores (servomotores, sensores, etc.) para desplazarse. En estos robots, el movimiento depende en gran parte por la pendiente del terreno, el uso de la gravedad y su diseño, otras ventajas en comparación con los robots activos es su peso y el bajo coste total en su fabricación, por otra parte, su desventaja es la poca versatilidad y el no poder subir pendientes [12], [13], [14].

1.4.4.5. Locomoción Activa

Como ya se mencionó, la locomoción activa es mucho más versátil que la locomoción pasiva, esto se debe en gran parte por el uso de actuadores para generar su movimiento, esto le permite explorar superficies irregulares, subir pendientes, saltar, correr, entre otras acciones, sin embargo, unas de sus principales desventajas es el gasto energético, el sistema de control es de alta complejidad y el alto coste [12], [13], [14].

1.4.5. ROS:

ROS es un *framework* de código abierto, modular, flexible e integral, que permite desarrollar algoritmos para robots [19]. ROS es una colección de herramientas, bibliotecas y paquetes que tienen como objetivo simplificar la tarea de crear robots complejos y versátiles en comportamiento [19], [20], [21], lo antes expuesto, es llevado a cabo gracias a la comunicación entre los componentes del sistema operativo, y a continuación se conceptualizan los más importantes de la herramienta.

- **Paquete(*package*)**: ROS se encuentra organizado en paquetes, los cuales pueden contener nodos, librerías, bases de datos, archivos de configuración, archivos de definición de mensajes y servicios, entre otros elementos usados dentro del proyecto [22], [23].
- **Espacio de trabajo:** Es la carpeta creada para el desarrollo del proyecto, en esta se encuentran los archivos (CMakeList.txt y Package.xml) que gestionan las dependencias y la estructura de los diferentes paquetes, además es el sitio donde se instalan, editan y compilan los paquetes que intervienen en el proyecto [22].
- **Repositorio:** Son ficheros los cuales están formados por uno o varios paquetes, los cuales son elementales para el desarrollo del proyecto [23].
- **Nodos:** Son códigos de programación ejecutables que describen los procesos a realizar y se encuentran ubicados dentro de los paquetes, estos nodos pueden ser publicadores o suscriptores, pueden comunicarse de manera unidireccional y con

nodos programados en distintos lenguajes, su ejecución se puede realizar en paralelo o individualmente [22], [23].

- **Tópicos:** Los tópicos son los canales (entrada o salida) encargados de llevar los mensajes de un nodo (llamado publicador) a otros (llamado suscriptor) de manera unidireccional. Un tópico puede ser usado por varios publicadores o suscriptores a la vez [22], [23].
- **Nodo Publicador:** Se le otorga este nombre al nodo encargado de colocar el mensaje en un tópico de salida para ser llevado a otro [23].
- **Nodo Suscriptor:** Este nodo es el encargado de recibir el mensaje que entrega el tópico de entrada [23].
- **Mensaje:** Los mensajes son usados para la comunicación entre un nodo suscriptor y un nodo publicador, estos son estructuras de datos, que comprende los tipos de campos. Los mensajes pueden incluir estructuras arbitrariamente anidadas y matrices [23].
- **Servicio:** Es otro tipo de comunicación entre nodos, en la cual interviene un nodo llamado cliente y otro llamado servidor. En este proceso se utiliza dos tipos de mensajes, uno para la solicitud o propuesta (acción realizada por el nodo cliente), y otro que es la respuesta a dicha propuesta enviada por el nodo servidor [22], [23].
- **Maestro**: Es el encargado de llevar el registro de nombres y la búsqueda de los distintos nodos creados en el proyecto, sin el maestro los demás nodos no podrían comunicarse entre sí, ya que los mensajes no se encontrarían, como también no se podría invocar servicios, es por esto que es indispensable para la ejecución de cualquier tipo de programa [23].

I.5. Estado del arte

Muchos enfoques populares utilizados en planificación de trayectorias, para la locomoción bípeda; se basan en el indicador de estabilidad ZMP, el modelo simplificado *Cart-Table* y la estrategia de control *preview control* [14]. Este modelo es presentado por Kajita en 2003 [19], [20], el cual consiste en la representación de un móvil para asemejar la aproximación de toda la masa del robot humanoide a una masa localizada en su centro (CoM), por otro

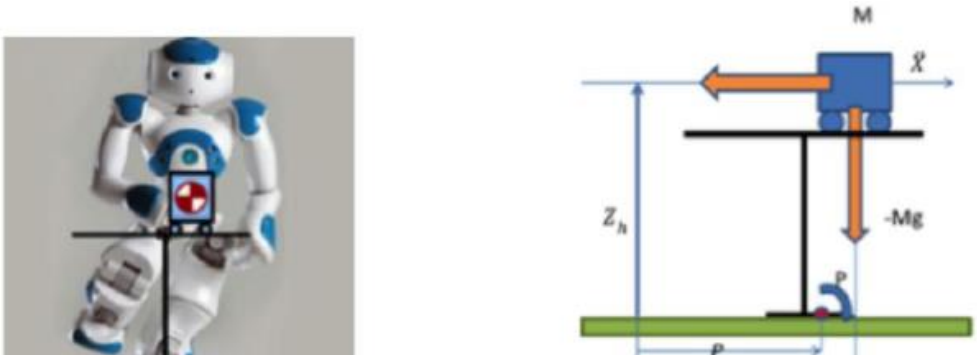


Figura 1.5. Comparación entre el robot humanoide y el modelo Cart-Table [47]

lado, asume que la pierna que se encuentra en fase de apoyo no tiene masa y la representa por una mesa sin masa, tal como se muestra en Figura 1.5 [47].

Una vez obtenido esta representación, el problema de controlar el movimiento del robot humanoide se reduce a controlar un péndulo invertido y con esto es posible la creación de patrones estables de marcha, usando una trayectoria específica del ZMP [7], [14], [19], [20], [45], [46]. Para realizar lo ya mencionado, se necesitan dos modelos *Cart-Table* para cada pierna, uno para el plano sagital (x, z) y otro para el plano frontal (y, z), adicional a esto para el uso de este método hay que planificar y definir la posición del pie durante la marcha [6], [14], [42], [47]. En 2005 M. Morisawa realizó de manera experimental la generación de trayectorias de marcha rectas con base en el criterio de estabilidad ZMP, utilizando el modelo parametrizado *Cart-Table* donde la altura del CoM no es constante, estas trayectorias se verificaron con los resultados del modelo cinemático del robot HRP-2 [13]. Esto sirvió de referencia para posteriores investigaciones, una de ellas es la generación de trayectoria mediante un arco de circunferencia, que imita el movimiento iterativo de los extremos de los pies; la cual se validó experimentalmente en el robot bípedo ScoutTM obteniendo buenos resultados [13], cabe resaltar, que en esta investigación se ejecutó una estrategia para reducir el error entre el ZMP calculado a partir del modelo completo del robot y el ZMP correspondiente al modelo parametrizado *Cart-Table*, esta estrategia consiste en hacer la retroalimentación del error, sumándose al ZMP de referencia y este nuevo ZMP compensado, es utilizado para generar la trayectoria de referencia del CoM del robot ScoutTM [13]. De igual forma, usando el modelo *Cart-Table*, se desarrolló un método general de planificación aleatorio de movimientos restringidos para la caminata sin colisiones en robots humanoides; el cual fue probado en un robot HRP-2 para resolver problemas de manipulación, en dicha investigación se demostró en primer lugar que el caminar dinámico conlleva a que los robots humanoides sean controlables en pequeños espacios, en segundo lugar que el modelo se adapta de manera eficiente al robot humanoide HRP-2 y por último que la diferencia entre este modelo y el robot humanoide, recae en la derivada del momento angular y de la aceleración vertical del centro de masa [20]. Del mismo modo se ha desarrollado un algoritmo de generación de patrones de caminata fuera de línea para rastrear una trayectoria deseada del ZMP, la cual se aplica al robot humanoide HRP4-C, con el fin de compensar el error entre la simulación del modelo simplificado y el modelo multicuerpo del robot, se optó por combinar la parte no lineal y la parte lineal, mostrando que el modelo lineal relacionado con el modelo simplificado se puede mejorar capturando el diferencial dinámico entre el modelo multicuerpo del robot humanoide y el del modelo *Cart-Table* [43]. Por otra parte, en la plataforma BIOLOID Premium se generaron trayectorias articulares dinámicamente estables *off-line*, utilizando el modelo *Cart-Table* [7], en dicho estudio se evidenció que el modelo muestra valores menores de pérdida por efecto Joule de los actuadores, frente a otros modelos simplificados para la generación de trayectorias [7]. Igualmente, usando el modelo *Cart-Table* en un robot humanoide NAO se ha realizado una caminata estable omnidireccional, donde se logra su ejecución en tiempo real y se desarrolla una optimización del modelo produciendo pasos más suaves y estables, lo que significa que el robot podrá alcanzar velocidades más altas

[48]. También, se ha desarrollado un sistema neuronal con la capacidad de generar perfiles de movimiento para un bípedo de 6 GDL en el plano sagital, el modelo *Cart-Table* en esta investigación es usado para entrenar el sistema neuronal con el fin de generar nuevas trayectorias de las articulaciones y posiciones del ZMP basado en la trayectoria del pie oscilante [46]. Incluso, se ha realizado una investigación en la cual se considera el problema de construir la trayectoria óptima de ZMP y la entrada correspondiente para sistema *Cart-Table* durante el ciclo de paso, en efecto se proporciona un método formal para evaluar la elección de la trayectoria ZMP deseada durante la fase de soporte único, debido a que esta impacta significativamente la energía total gastada durante el paso del ciclo [49]. También dicho modelo se adaptó a un robot NAO, en donde la experimentación mediante simulación mostró resultados favorables. Por otro lado, en [50], se propone un modelo de *Cart-Table* extendido para habilitar la marcha lateral en humanoides, a partir de este modelo y el uso de ZMP extendido como una nueva variante en esta investigación, se construye e implementa un generador de patrones para robots bípedos, el cual se valida mediante simulación en el robot Atlas, donde los resultados demuestran que el robot puede caminar de manera estable en terrenos bastante complejos. Estos experimentos indican que el método es exitoso para generar estabilidad lateral y diagonal, es decir que el robot es capaz de caminar en cualquier dirección [14], sin embargo, se notó que el principal problema de la aplicación del modelo *Cart-Table*, sucede cuando la trayectoria es calculada al mismo tiempo en que el robot bípedo real efectúa la caminata, debido a que la trayectoria obtenida del CoM es muy sensible a la variación de la marcha andante [14], a pesar de ello evidenció que se generan trayectorias óptimas utilizando una ganancia de alto rendimiento [14]. Igualmente, otra investigación realizada en el robot NAO mediante simulación, ha sido un motor de marcha omnidireccional basado en un planificador del ZMP mediante un bucle de equilibrio activo, de esta investigación, se obtienen resultados alentadores debido a que el robot ha podido caminar de manera rápida y estable en cualquier dirección con prestaciones que se comparan con los mejores robots usados en RoboCup 2012 [51]. Otras aplicaciones que ha tenido el método Cart-Table, en la que se realizó su experimentación mediante simulación, es la del robot KDHSR, en el cual se intenta seguir trayectorias ZMP, donde se demuestra que una de las ventajas de este método, es la facilidad para planificar la marcha de acuerdo con las trayectorias ZMP deseadas [15]. Además, se ha realizado un modelo de control predictivo (MPC, *Predictive Control Model*) con restricciones para generar una caminata dinámica para el humanoide COMAN, la dinámica del robot se representa utilizando el modelo *Cart-Table* que permite la generación de una marcha equilibrada dado un patrón de caminata planificada teniendo en cuenta el ZMP [19]. Otro caso de estudio realizado en un simulador dinámico, es la generación de patrones de marcha mediante el uso de un control preview control del ZMP en un HRP-2p, evidenciando que se puede compensar el error del ZMP causado por la diferencia entre el modelo *Cart-Table* y el modelo multicuerpo usando un controlador de *preview control* [42], también, se ha elaborado un generador de patrones de movimientos para subir pendientes en simulación dinámica 3D, usando control de *preview control* del ZMP a través del modelo *Cart-Table*, los resultados de la simulación dinámica mostraron que el robot puede caminar satisfactoriamente en muchos tipos de terrenos irregulares usando el CoM de las trayectorias generada [52]. Adicionalmente, se ha demostrado la eficiencia del modelo

Cart-Table a través de un prototipo de robot bípedo, el cual demuestra cómo se ha aplicado la base de esta teoría del modelo simplificado para: el diseño, el movimiento y el control de una marcha estable y rápida, e incluso en este estudio se ha propuesto un método de transformación del ZMP y del CoM de un patrón de caminata a otros patrones, al considerar la situación de caminata en tiempo real para la estructura de un robot [16], [53]. Los estudios mencionados con anterioridad acerca del uso del modelo *Cart-Table* para la generación de marcha, permiten evidenciar la obtención de buenos resultados tras su aplicación, siendo un modelo de fácil comprensión en comparación a otros modelos y el cual se puede extrapolar a un robot bípedo común.

En la generación de trayectorias de marcha, un aspecto igualmente importante, reside en la correcta elección de la herramienta software que ejecuta los algoritmos de marcha, en este caso se ha indagado sobre ROS como una nueva apuesta en la generación de marcha. En los últimos años se han realizado diversas investigaciones con este sistema operativo, como el control de robots industriales, estudios en la NASA para robots espaciales, e incluso se ha trabajado en la industria 4.0 [54]. En la actualidad este sistema operativo, se ha convertido en una plataforma dominante para la investigación en robótica, se han desarrollado aplicaciones significativas y ha permitido el diseño de sistemas de control para robots, con el objetivo de un manejo dinámico, rápido y un desarrollo confiable, basado en software de código abierto, además ROS facilita la integración de varios algoritmos de control y manipulación de sensores y actuadores disponibles como paquetes distribuido [23]. Por ejemplo los humanoides TALOS, tulipán y el NAO son algunos robots que actualmente utilizan el sistema operativo ROS para su funcionalidad [54], [55], en estos dispositivos, los algoritmos de control se realizan mediante nodos funcionales que están interactuando y ejecutándose simultáneamente, llevando a cabo, el intercambio de información, para darle movimiento a cada una de las articulaciones del humanoide [24], [25], otra muestra de robots implementados con ROS, es el robot NimbRo-OP desarrollado por la Universidad de Bonn en Alemania, este mide 95 centímetros de estatura, pesa 6.6 kilogramos y utiliza servomotores Dynamixel, su característica más sobresaliente es que constituye un proyecto de plataforma abierta, diseñado para trabajos de investigación en campos como el control clásico, la inteligencia artificial y robótica, dicha temática es similar a la pretendida en este proyecto [56], también de manera similar se ha buscado mejorar la capacidad del robot comercial BIOLOID Premium, usando el *Framework* ROS y cambiando su tarjeta controladora, ROS se utiliza como marco de desarrollo y el software impregnado con bibliotecas Python se emplea para integrar funciones robóticas, realizando en este *Framework* los cálculos necesarios para su adecuado funcionamiento y una interfaz de programación de aplicaciones (API, *Application Programming Interface*) de alto nivel [4]. Otra implementación en ROS, es un sistema de localización y navegación de un robot humanoide, en la cual se realiza la planificación de la ruta basada en la exploración de límites, obteniendo buenos resultados en el momento de la experimentación [57]. También se ha realizado el diseño de un robot que combina elementos activos y pasivos, para su funcionamiento y coordinación de movimientos, se creó una interfaz, que al igual que la programación del robot, fueron desarrollados en ROS, lo que facilita considerablemente la puesta en marcha de pruebas de caminata [32], además, se observa la programación de

nodos que permiten manipular completamente los parámetros del robot, adquirir y almacenar los datos de sensores y controladores y una fácil extensión del sistema mediante nodos ROS [32]. De igual forma, se ha realizado el diseño del controlador de un humanoide en ROS para seguir trayectorias de marcha, este diseño asemeja a un sistema operativo distribuido, los resultados de la simulación mostraron que el robot humanoide sigue la onda predefinida para completar los recorridos de su desplazamiento [44]. De modo similar, se ha realizado un controlador ROS con una capa de interfaz IPC híbrida y un controlador de robot diseñado para ejecutarse en THOR y ESCHE [45]. Otro desarrollo, ha sido un algoritmo de control que permite el movimiento de un robot bípedo en terrenos inclinados o irregulares, el algoritmo es escrito en ROS buscando que pueda transferirse fácilmente a otro humanoide con estructura similar, cabe resaltar que se comprobó en un robot REEM-C, para el cual se diseñaron controladores PID buscando lograr una mejor estabilidad en la posición del pie [46]. Igualmente, en [41], los autores construyeron un bípedo para el cual se implementó un controlador con múltiples entradas y múltiples salidas (MIMO, Multi-Input Multi-Output) no lineal, en esta investigación el simulador Gazebo se utiliza para validar el modelo matemático y el esquema de control, evidenciando que durante este proyecto el uso de ROS redujo el tiempo de implementación debido a que se pueden reutilizar algunos componentes [41]. Incluso se ha desarrollado e implementado trayectorias de manera virtual en el simulador Gazebo, uno de ellos es la simulación del humanoide Robotis OP2, en el cual se llevó a cabo un modelo de alta fidelidad, su controlador implementado en el lenguaje de programación C++, el cual usó los nodos de mensajería ROS para interactuar con Gazebo [47], otro caso de estudio en el simulador Gazebo y ROS, es el desarrollo de un algoritmo de modelado, identificación, control y estabilidad de sistemas bípedos para facilitar el diseño y control de robots humanoides, este algoritmo realiza la selección de parámetros de marcha para la velocidad del caminar y trayectorias omnidireccionales, la validación experimental se realizó en un robot humanoide Tulipán simulado, observando que este marco de simulación es computacionalmente rápido e incorpora todos los aspectos dinámicos importantes [48]. Igualmente, se creó un método de planificación tridimensional de marcha bípeda humanoide utilizando la geodésica, para verificar el patrón de caminata propuesto se implementó en el robot RoboErectus Senior Adult Si [49]. De forma similar, otra herramienta que posee ROS para realizar simulaciones es la herramienta de visualización para ROS (RVIZ, *visualization tool for ROS*), es preciso aclarar que esta herramienta es estrictamente de visualización, que permite observar la forma que va adquiriendo el diseño robótico, esta herramienta no cuenta con esquemas de simulación de fricción, gravedad, ni tampoco es posible la construcción de *worlds* o escenarios de interacción [58]. Sin embargo, esta herramienta puede ser complementada con otras herramientas de ROS como por ejemplo creadores interactivos (IMs, *interactive markers*), la cual le permite a RVIZ crear interfaces bidireccionales, es decir permite tanto la visualización como también la interacción con el sistema robótico [59]. Otro ejemplo de plataformas virtuales desarrollada en RVIZ se encuentra en [54], en la cual permite mirar la configuración física, los movimientos y operaciones que realiza un brazo robótico en tiempo real, por otro lado, esta plataforma también permite visualizar y representar las lecturas de los respectivos sensores del sistema, permitiendo una mejor manipulación y control del brazo. También se ha

realizado un estudio donde se ejecutan trayectorias para robots bípedos basado en la planta del pie del robot, este estudio proporciona no sólo las fuerzas resultantes aplicadas en los tobillos sino también una forma precisa de cómo es la distribución de la presión aplicada en el suelo [50]. Cabe destacar que este sistema operativo cuenta con un paquete denominado formato de descripción de robot unificado (URDF, *Unified Robot Description Format*) que permite la descripción de la cinemática, la dinámica, el modelo de colisiones, materiales, color y textura del robot [41]. Demostrando una vez más que ROS es una herramienta útil, para la simulación de robots, sin embargo, aún se queda corto en la experimentación aplicada a robots bípedos reales que presentan características open source.

Por otro lado se ha realizado diferentes desarrollos en robots bípedos, que no necesariamente se han centrado en la implementación de trayectorias, por ejemplo, se ha creado un entorno dinámico donde dos robots juegan fútbol [19], también la plataforma robótica humanoide bípeda con visión artificial de bajo coste para la investigación, llamada robot Clank [22] y robots bípedos que integran sistemas de aprendizaje de localización y mapeo simultáneo (SLAM, *simultaneous locating and mapping*), la teleoperación de un robot humanoide basado en una red distribuida desarrollada en ROS, que une los dispositivos hápticos y el simulador en tiempo real [25] y la teleoperación del robot NAO, esta se realiza empleando la voz y gestos [51]. Otro desarrollo es robotaxBot, este es un robot que posee un controlador específicamente desarrollado en ROS, que facilita el uso de varias herramientas para el análisis de datos y la interacción entre múltiples robots, sensores y dispositivos de tele operación [60]

Capítulo 2: Diseño hardware y software del robot bípedo propuesto.

2.1. Estructura de soporte del Robot Bípedo

El diseño del bípedo involucra pensar en etapas futuras que darán continuidad al proyecto, por tanto, este debe ser capaz de desplazarse en línea recta y realizar giros, Igualmente se contrasta el diseño articular del robot con la estructura articular de los seres humanos, debido a que somos capaces de realizar los movimientos requeridos, en consecuencia, se planteó desarrollar un robot bípedo con 6 GDL por cada pierna, distribuidos de la siguiente forma: 3 GDL en la cadera, 1 GDL en la rodilla y 2 GDL en el tobillo. Al tener claridad del número de articulaciones y en vista de que el Semillero de Robótica de la Universidad del Cauca cuenta con dos estructuras metálicas para robots bípedos y seriales cada una de 6GDL, se planteó lo siguiente: usar las estructuras disponibles y realizar modificaciones que permitan obtener una estructura de soporte para el prototipo. Inicialmente se elaboró el modelo diseñado asistido por computadora (CAD, *Computer-Aided Design*), con el fin de establecer las dimensiones geométricas de manera simulada y posteriormente realizar la implementación física del prototipo. Cabe resaltar que las longitudes aportadas Drillins y Contini parten de la estimación estadística de las longitudes del cuerpo humano a partir de su estatura, por este motivo, dichas longitudes son un factor multiplicativo con respecto a la altura.

2.1.1. Modelo CAD del robot bípedo:

Respecto al diseño estructural del robot bípedo, se consultó documentación bibliográfica relacionada, a establecer los parámetros necesarios para que el robot pueda desplazarse en línea recta y realizar giros, tomando como base las dimensiones de la estructura del ser humano, debido a esto, se ha optado por seguir la metodología sugerida por Drillins y Contini en 1996, en consecuencia, a los casos de éxito que se ha obtenido en experimentos de robótica anteriores [61]. Las simulaciones de las piezas del robot bípedo se crearon en el programa Autodesk Inventor Professional 2018, el cual es un software especializado para diseño, modelado y simulación mecánica. A continuación, se describen cada una de ellas:

Pieza soporte cadera u: Es una pieza rígida que tiene por función soportar los eslabones que se conocen como las piernas del robot, además de albergar algunos dispositivos como las tarjetas controladoras, baterías, entre otras. Esta pieza se muestra en la Figura 2.2.

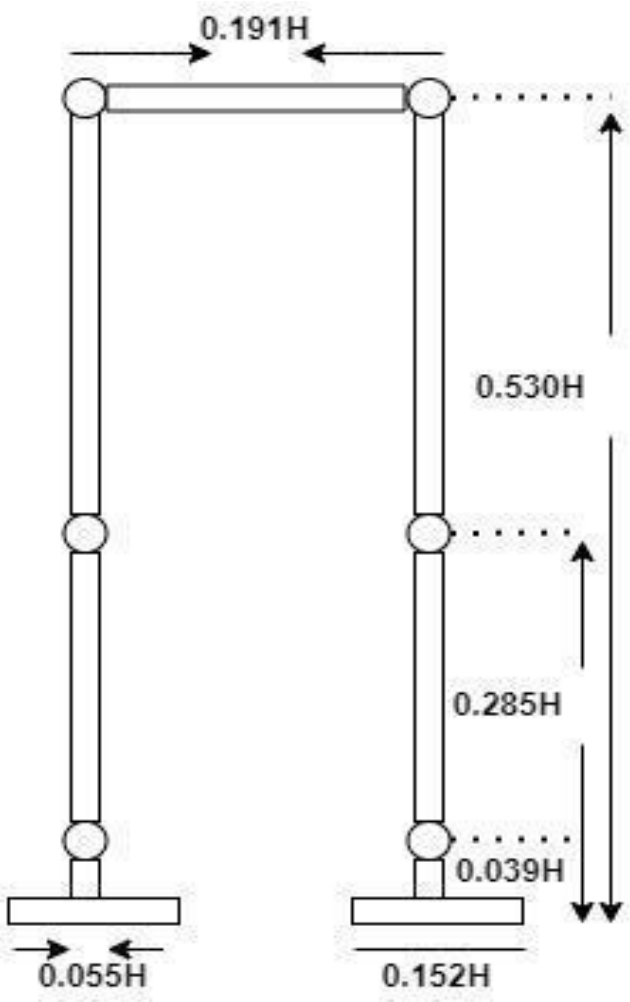


Figura 2.1. Longitudes aportadas por Drillins y Contini (1996). Cabe resaltar que el factor "H" corresponde a la altura total del robot.

$$Thigh = (0.530 - 0.285) * H = 23{,}1\ cm \quad (3)$$

$$Fibula = \ ("0.285 - 0{,}039")" * H = 23{,}2\ cm" \quad (4)$$

$$Ankle \ = \ 0.039 * H = 3{,}68\ cm \quad (5)$$

$$Foot\ width \ = \ 0.055 * H = 5{,}18\ cm \quad (6)$$

$$Long\ foot \ = \ 0.152 * H = 14.34\ cm \quad (7)$$

$$Hips \ = 0{,}191 * H = 18{,}01\ cm \quad (8)$$

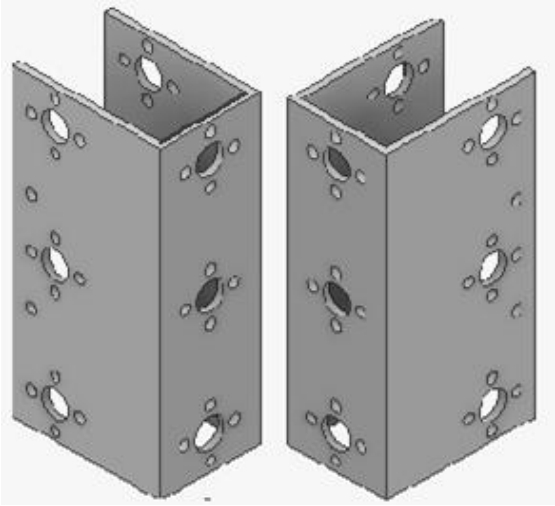


Figura 2.2. Pieza soporte cadera u

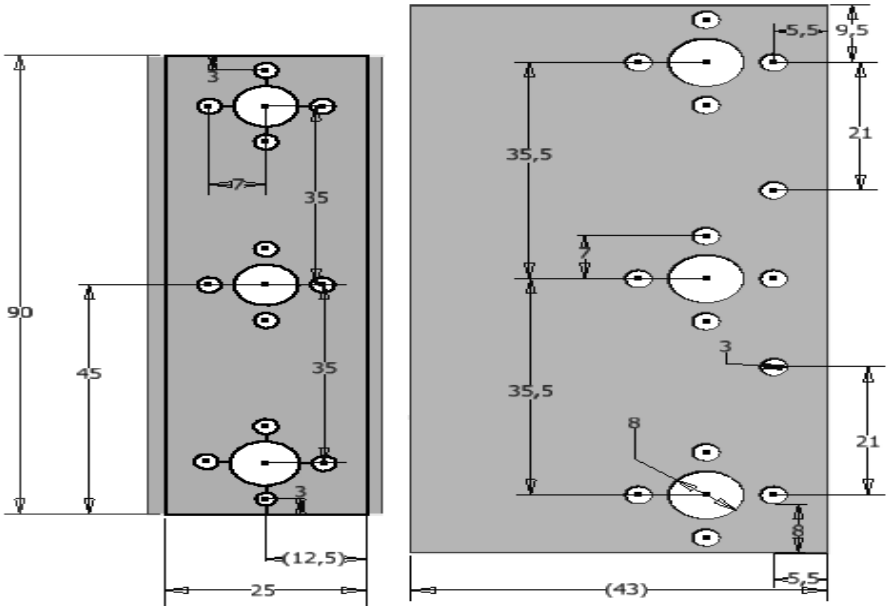


Figura 2.3. Medidas de Pieza soporte cadera

Pieza cadera: El diseño de esta pieza se centra especialmente en aumentar la longitud del ancho de cadera con el propósito de cumplir con la longitud de los parámetros establecidos por Drillins y Contini, además, poder unir las piezas de soporte cadera u presentado en la

Figura 2.2, pieza con las que ya se contaba. En la Figura 2.4, se puede apreciar la forma y dimensiones de esta pieza.

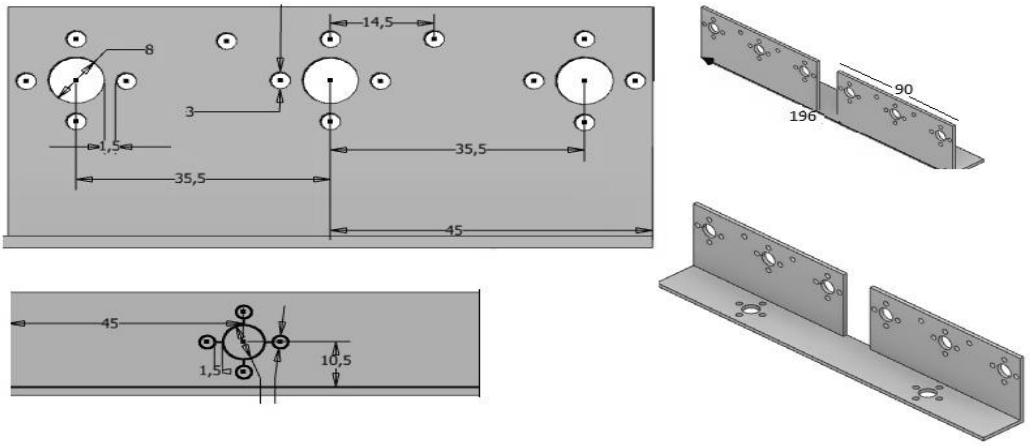


Figura 2.4. Pieza y medidas Cadera

Pieza peroné/muslo: A continuación, se presenta la estructura y dimensiones de las piezas peroné (Figura 2.5) y muslo (Figura 2.6). Estas fueron diseñadas con el propósito de aumentar la longitud de las piernas y así cumplir los parámetros estructurales sugeridos en la literatura, cabe resaltar que, estas piezas son construidas en filamentos de ácido poliláctico (PLA, *polylactic acid)* bajo impresión 3D y poseen un agujero cilíndrico, a través del cual se inserta un tubo metálico de aluminio que agrega soporte rígido, con el objetivo de brindar mayor resistencia, estabilidad y evitar que los ensambles de estas piezas se separen, además de poder introducir parte del cableado por este orificio.

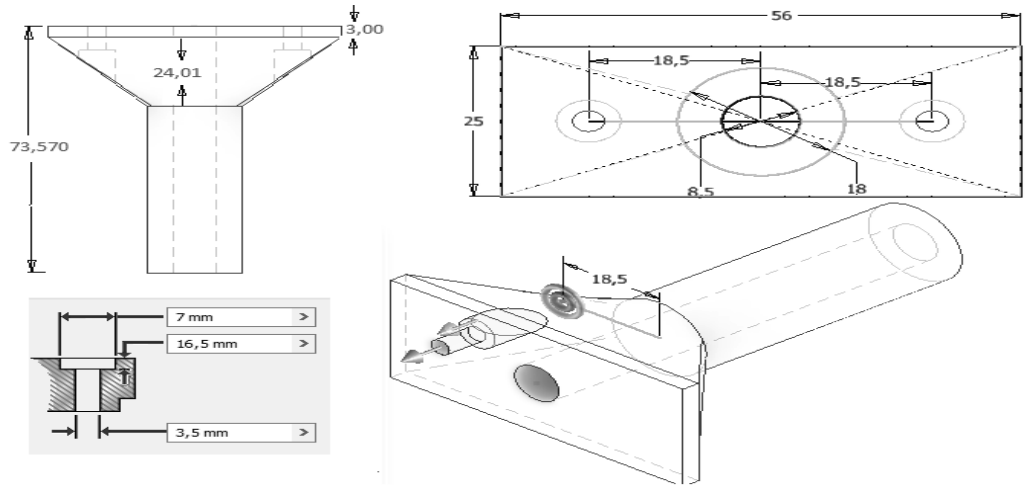


Figura 2.5. Pieza peroné

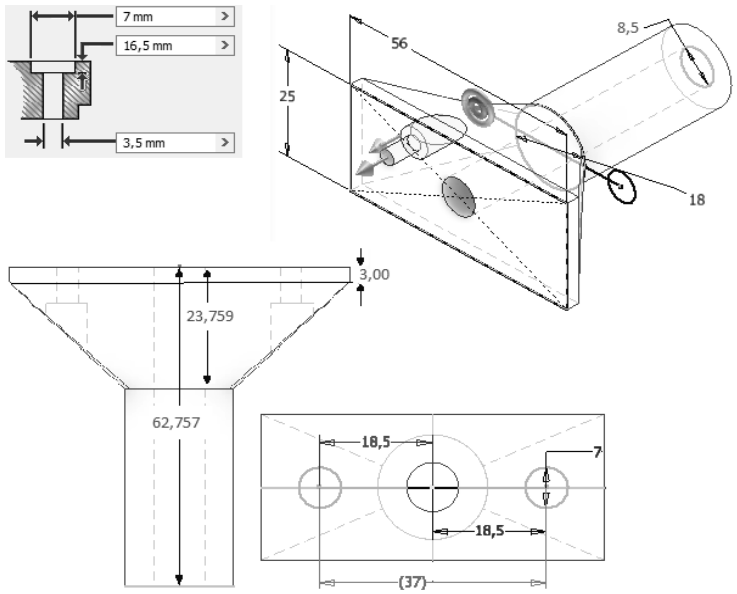


Figura 2.6. Pieza muslo

Pieza L: Es una pieza, cuyo propósito es el agarre del eslabón peroné, su estructura y dimensiones se observan en la Figura 2.7.

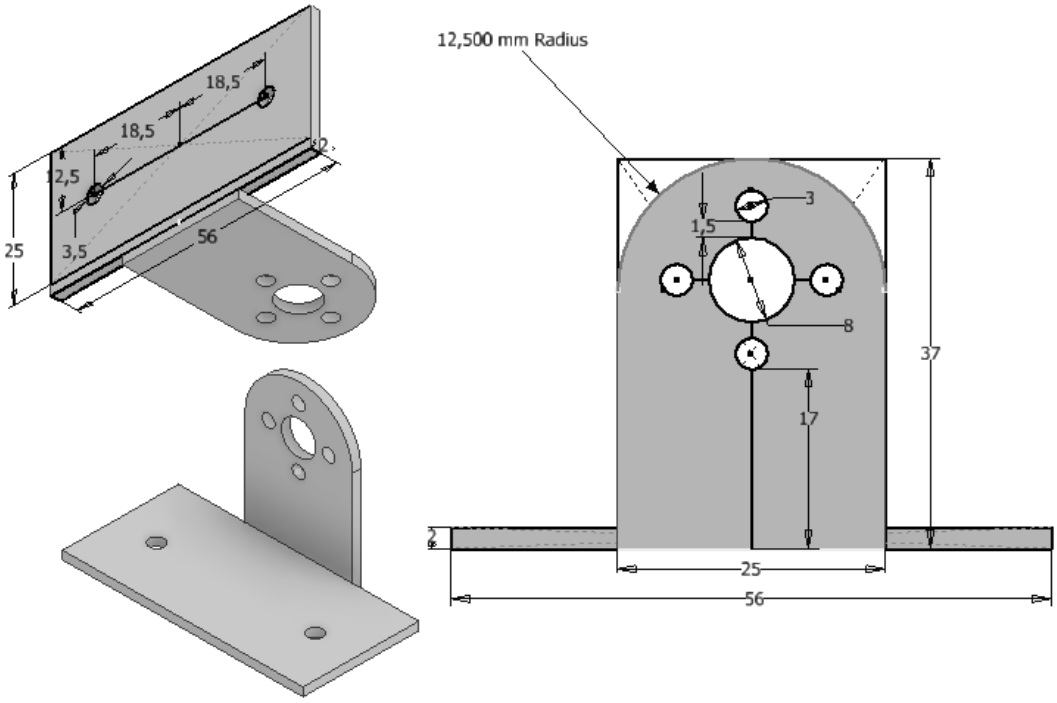


Figura 2.7. Pieza L

Pieza unión: Su utilidad es la unión de dos elementos, en este caso dos piezas peroné, como también dos de Pieza muslo, se muestra en la Figura 2.8.

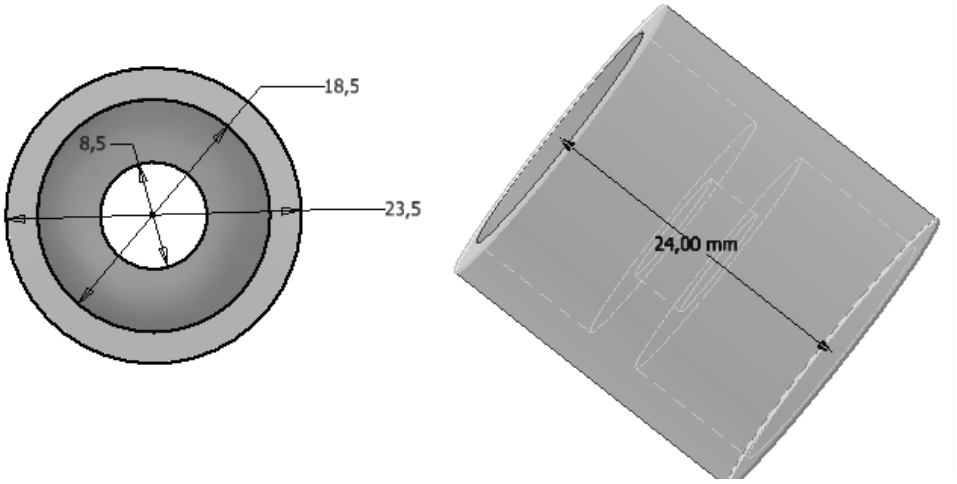


Figura 2.8. Pieza unión

Sujetador de motor: En la Figura 2.9 y Figura 2.10 se muestran las características de diseño de sujetador de motor, su funcionalidad es el agarre del servomotor o articulación motorizada de tal manera que quede completamente fijo al resto de la estructura.

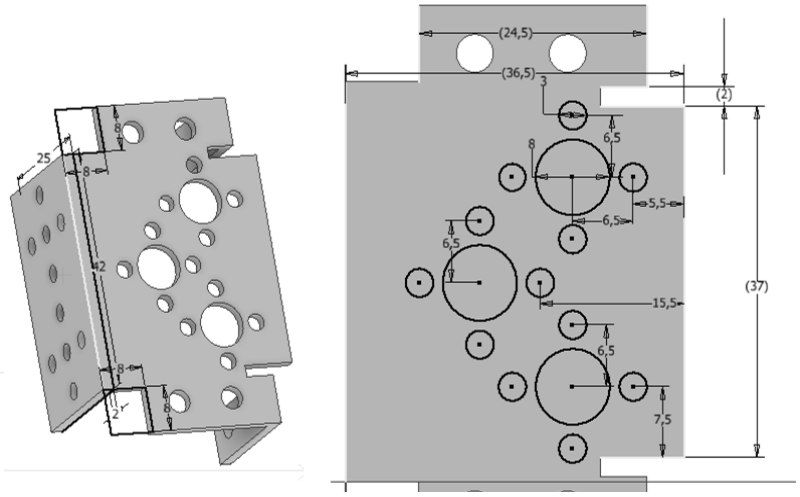


Figura 2.9. Sujetador de motor vista lateral

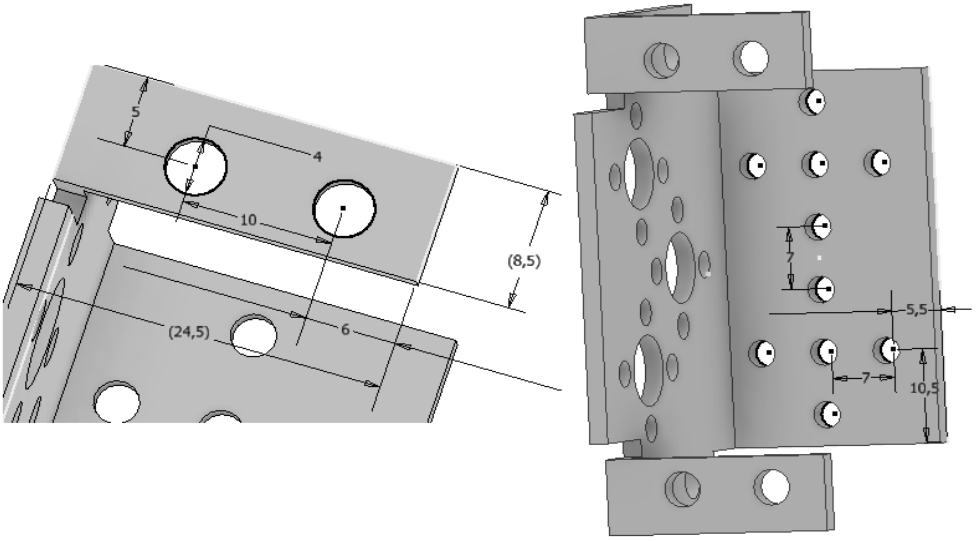


Figura 2.10. Sujetador de motor vista frontal

Soporte u largo / corto: Como se observa en la Figura 2.11, *es una p*ieza en forma de u, la cual sujeta a los ejes con accesorios del motor y otras piezas, con el propósito de que los eslabones puedan seguir el movimiento del motor.

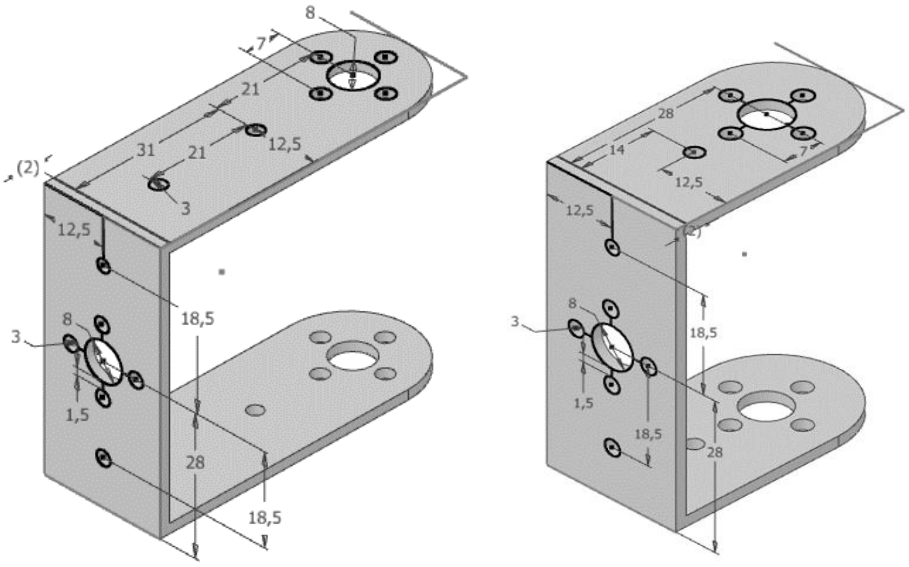


Figura 2.11. Estructura y dimensiones de Soporte u largo/ corto

Planta pie: Es una estructura plana, su funcionalidad es darle soporte a toda la estructura del robot bípedo, mediante el contacto plano entre el suelo y esta pieza. Esta permite que el robot permanezca en pie, su diseño y dimensiones se muestran a continuación.

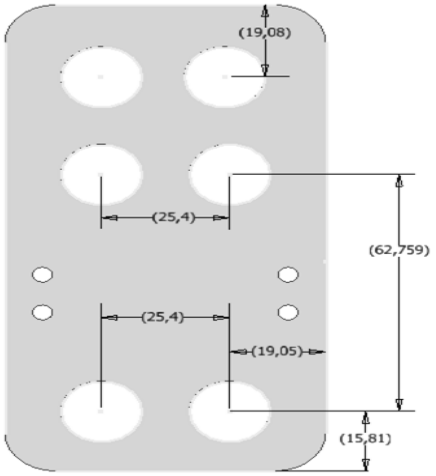


Figura 2.12. Planta pie

Zapato: Esta pieza consta de dos componentes; el primero se puede apreciar en la Figura 2.13, diseñada en material PLA bajo impresión 3D, con el fin de contener los sensores de presión en los vértices del zapato, es por esto que presenta en una de sus caras muelles por donde pasa el cableado, además por su cara posterior (estará en contacto con el piso) se realizaron ciertas perforaciones de tal manera que, al ensamblar las dos piezas con tornillos, no tengan contacto con el piso para evitar inestabilidad por contacto no plano. El segundo componente del zapato presentado en la Figura 2.13, es el que mantiene fija la pieza planta pie.

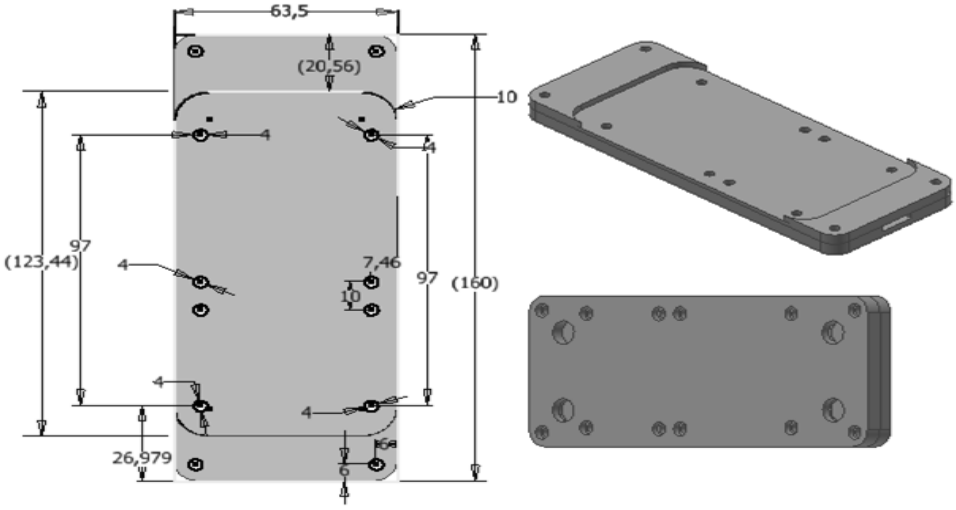


Figura 2.13. a) vista frontal del segundo componente del zapato b) zapato en su vista

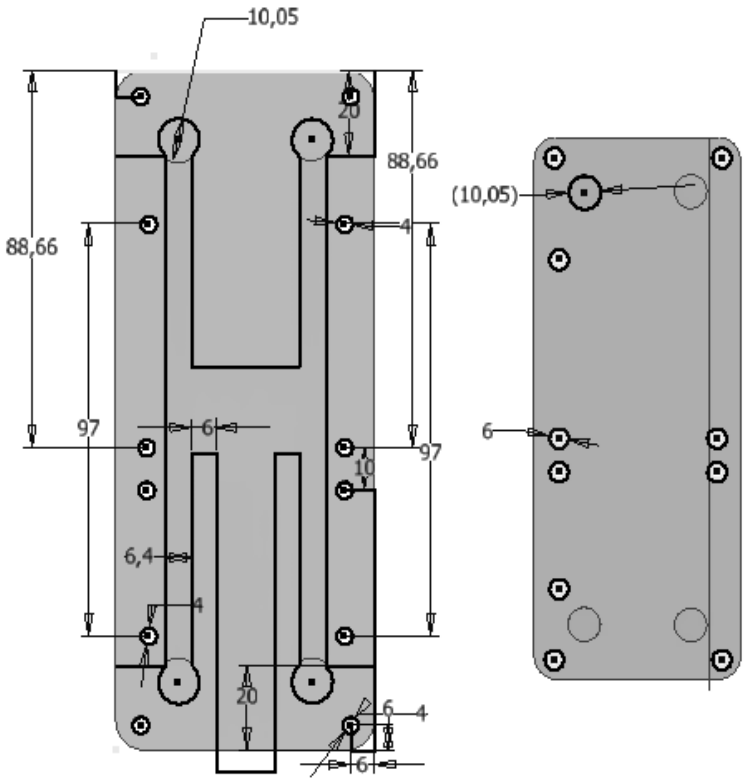


Figura 12. Vista frontal y posterior del primer componente del zapato

Eslabón cadera: Este eslabón es un ensamble de Figura 2.2 y Figura 2.4, como se aprecia a Figura 2.15, el cual busca ampliar la longitud de la cadera.

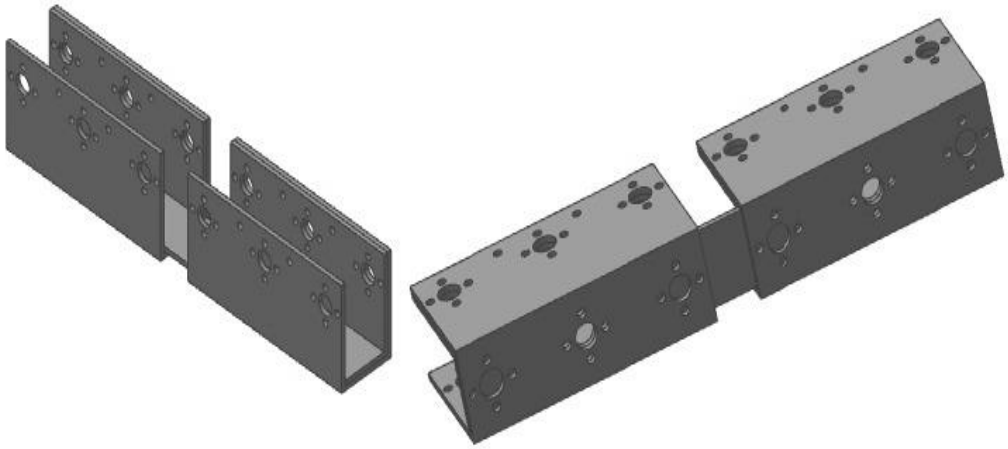


Figura 2.15. Eslabón cadera

Eslabón muslo/peroné: Como se puede apreciar en Figura 2.16, los eslabones muslo y peroné son iguales en cuanto diseño y composición, la única diferencia entre estas es la longitud, siendo el muslo el más pequeño. Estos eslabones son la unión de tres piezas, ya sean dos piezas peroné o dos de muslo y una pieza de unión.

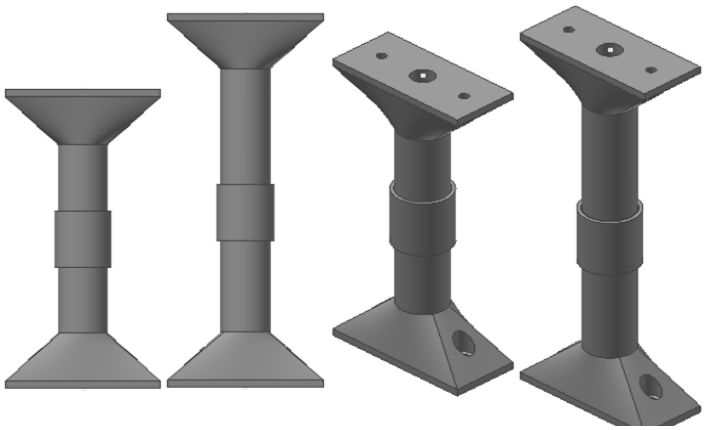


Figura 13. Eslabón muslo y Peroné

2.1.2. Articulaciones y grados de libertad del robot:

Cadera: La cadera cuenta con 3 GDL, uno en el eje "x" que permite llevar la pierna hacia delante y hacia atrás, el siguiente en el eje "y" para levantar la pierna y el último en el eje "z" con el fin de que la pierna pueda rotar sobre su propio eje. El objetivo de estos 3 GDL es darle al robot la capacidad de marchar en línea recta y el poder girar. Por otro lado, la cadera está formada por la pieza Eslabón cadera, 5 piezas U largas, 2 piezas U cortas y 6 sujetadores de motor, como se muestra en Figura 2.17.

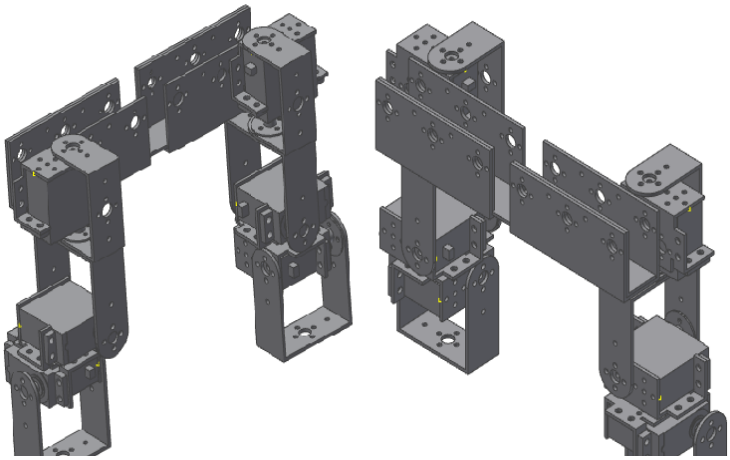


Figura 14. Articulación cadera

Rodilla: La rodilla posee un grado de libertad cuyo eje de rotación es “x”, que permite mover la rodilla hacia delante y hacia atrás y está compuesta por una pieza L, una pieza en U y un de sujetador de motor. A continuación, se muestra el ensamble de las piezas mencionadas anteriormente.

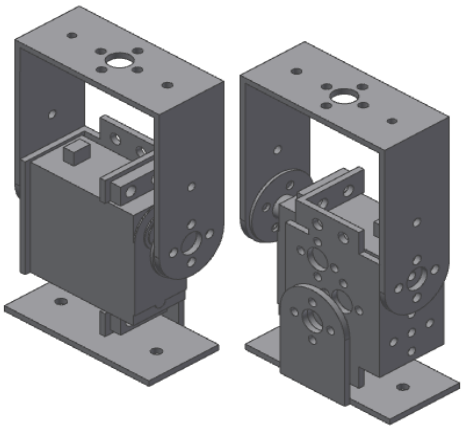


Figura 16. Articulación rodilla

Tobillo*:* En la Figura 2.19 se puede apreciar el diseño del tobillo izquierdo y derecho respectivamente. Estos poseen 2 GDL, uno para el eje “x” que permite moverlo hacia adelante y hacia atrás, y otro grado de libertad en el eje “y” para el movimiento hacia arriba y hacia abajo, también consta de cinco piezas; dos de Pieza en U (la primera de tamaño

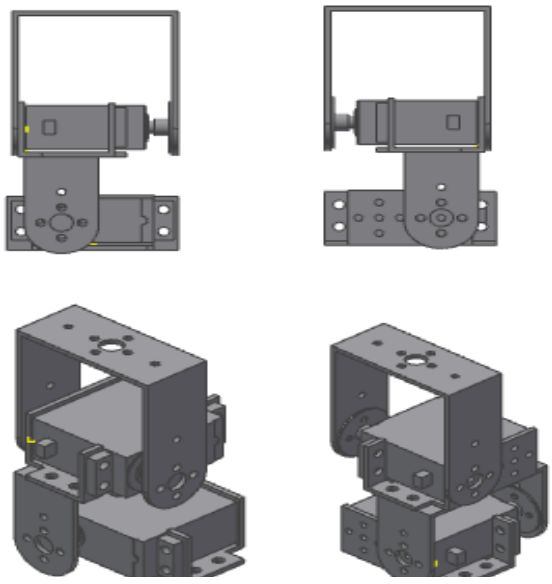


Figura 15. Articulación tobillo

pequeño y la segunda de tamaño grande si las enumeramos en forma ascendente) y dos piezas sujetador de motor.

Ensamble completo del robot bípedo: En figura 2.20, se encuentra el ensamble del robot bípedo, en esta se muestra una vista frontal, lateral y posterior del robot respectivamente.

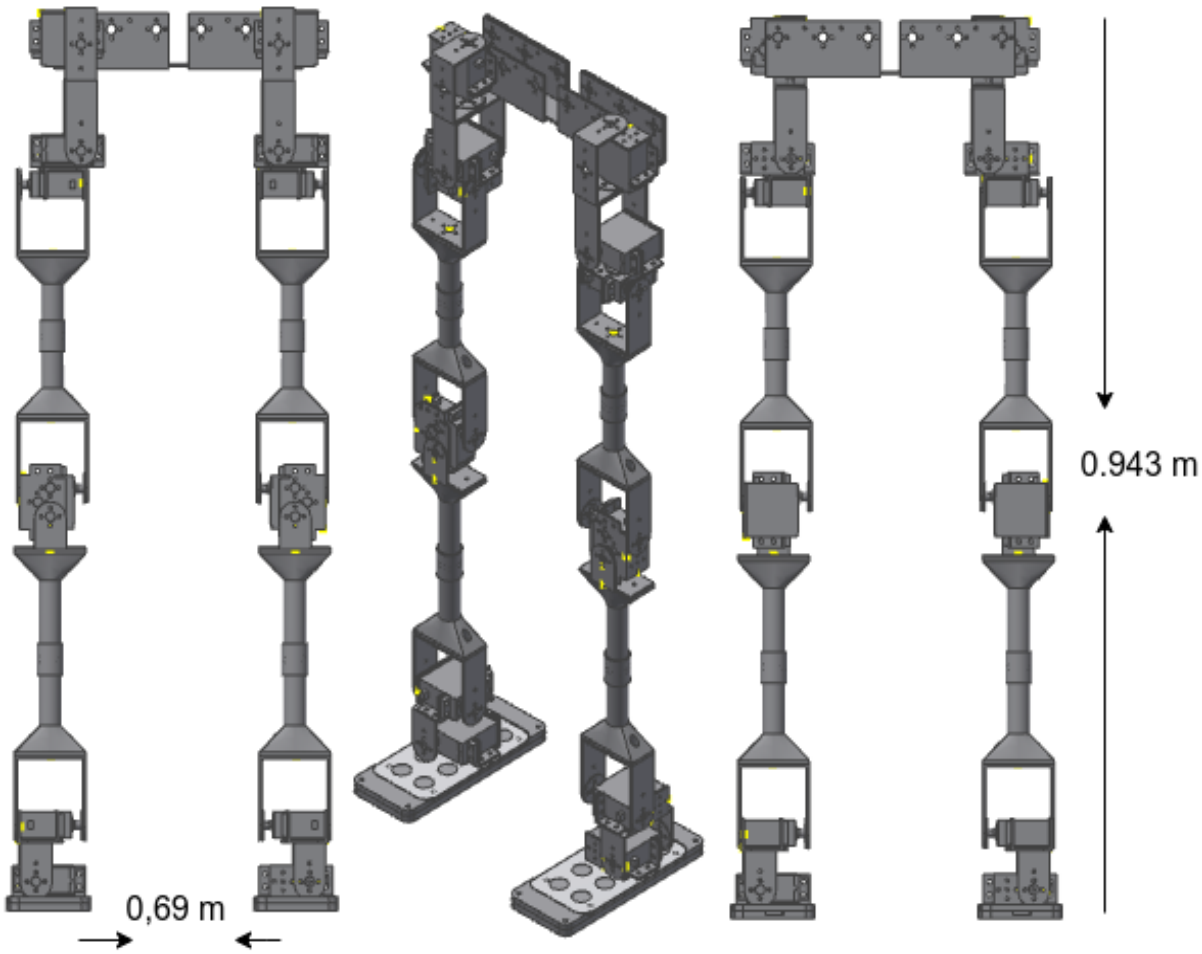


Figura 17. Robot bípedo diseñado en CAD

2.2. Implementación de la estructura de soporte del robot:

El diseño estructural y modelado CAD del Capítulo 2.1 se utilizó como guía para acoplar cada una de las piezas, eslabones, articulaciones y demás componentes de la estructura esquelética del robot, para esto se usaron todas las piezas de las dos estructuras metálicas y se construyeron algunas piezas, unas de ellas en impresión 3D sobre material PLA y otras en aluminio. Las especificaciones de las piezas se detallan a continuación:

Estructuras metálicas de 6 GDL*:* En el Semillero de Robótica de la Universidad del Cauca se cuenta con dos estructuras de piezas fabricadas en aluminio, estos fueron usados para realizar la construcción real del bípedo y se listan a continuación.

- 2 unidades de planta de pie
- 12 unidades de soportes de motor
- 8 unidades de soportes u largo
- 4 unidades de soportes u corto
- 4 unidades de soportes servo tipo l
- 2 unidades de soportes de cintura robot tipo u

Piezas impresas en 3D: En seguida, se menciona las piezas diseñadas y la cantidad elaborada en CAD e impresas en material PLA. Se optó por este material, con el fin de que las piezas sean resistentes, livianas y rígidas.

- 4 unidades de Pieza unión
- 4 unidades de Piezas peroné
- 4 unidades de Piezas muslo

Piezas elaboradas en metal*:* Las siguientes piezas se elaboraron en metal, con el fin de evitar la ruptura al ser delgadas y dar soporte a eslabones que las sujetan. Estas piezas son las siguientes:

- 1 unidad de Cadera 2
- 2 unidades de Pieza en L

2.3. Características eléctricas de los componentes del robot

Tarjeta ESP32: Es un módulo para aplicaciones con WIFI, Bluetooth y soporte amplio de periféricos, posee un procesador Tensilica Xtensa 32 bits LX6 hasta 240 MHz, una ROM de 448 KB, tiene Periféricos ADC, DAC, I2 C, UART, Interfaz CAN 2.0, SPI, I2S, RMII y PWM entre otros, su Voltaje de trabajo es de 3.3 VDC y su alimentación y envió de datos son vía conector micro USB 5 VDC. Esta tarjeta cuenta con varias funcionalidades en el proyecto, la primera es recibir la información de los sensores, mediante señales analógicas y enviarlas como cadena a la tarjeta Raspberry, y la segunda funcionalidad es la recepción y la entrega de la señal dada por la modulación de ancho de pulso (PWM, *Pulse Width Modulation)* hacia los motores.

***Turnigy power systems 20-30c discharge, high discfarge lipo battery*:** Batería lipo recargable, de 11.1 V, con una corriente de 1000 mA, posee una descarga constante de y su pico de descarga es de 40 C en 10 s, mediante cargas de alta corriente. Esta batería proporciona la alimentación a los motores del robot bípedo mediante un voltaje DC.

Módulo LM2596 Convertidor de Voltaje DC-DC: Es un regulador DC-DC *Step Down* LM259, lo que quiere decir que es una fuente de alimentación conmutada, está basada en el regulador LM2596, su voltaje de entrada está entre 4.5-40 V y su voltaje de salida entre 1.5-35 V (Ajustable), su corriente de salida máxima es de 3 A.

MG966r: Este servomotor gira 120° (60° a la derecha y 60° a la izquierda), posee un torque de 9.4 kgf a 4.8 V y 11 kgf a 6 V, posee una velocidad de operación de 0.17 s/60º a 4.8v y, 0.14 s/60º a 6 V, sus voltajes de operación son de 4.8 V a 7.2 V, su corriente en funcionamiento es de 500 mA y su corriente en reposo es de 2.5 mA a 6 V, este motor se encuentra en las articulaciones 1, 5, 6, 7, 11, 12.

1501MG: Este servomotor posee un ángulo de funcionamiento de aproximadamente 165º, posee un torque de 15.5 kgf a 4.8 V y 17 kgf a 6 V y tiene un voltaje de operación de 4.8 V a 6 V, este servomotor se encuentra en las articulaciones 2, 3, 4, 8, 9,10.

Sensor de fuerza de presión DF9-40: El rango de medida de este sensor es de 0-5 kg, también cuenta con una precisión del 22.5% (rango de medida 85%) y su tiempo de respuesta es de 1 ms. El DF9-40 cuenta con una resistencia inicial de 10 MΩ (sin carga), y su tensión de prueba es de 3.3 Vcc.

2.4. Diagrama eléctrico general del robot Bípedo

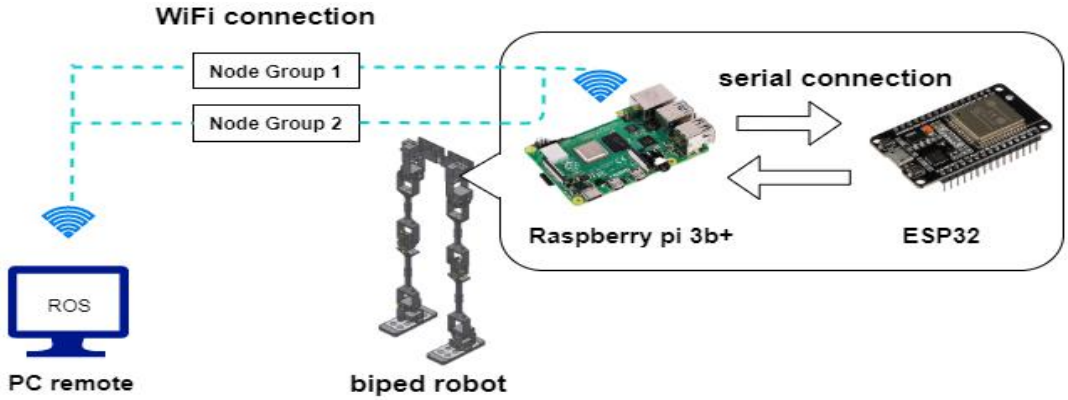


Figura 18. Software del robot bípedo

En la Figura 2.22 se muestra como están cableados los motores y los sensores de fuerza del robot. El robot es energizado mediante una batería de 11.1 V a 1000 mA, a su vez este voltaje es regulado a 5 V mediante el módulo x1406b para alimentar los servomotores, mientras que la tarjeta ESP32 permite alimentar los sensores de presión mediante el pin de suministro de 3.3 Vcc. Cabe resaltar que el bloque denominado CITO_SENSOR#, es mostrado detalladamente en la Figura 2.23, el cual se replica ocho veces (uno por cada sensor); este, es un circuito seguidor de tensión o amplificador de ganancia unitaria y es considerado como un filtro de lectura para los sensores, puesto que posee un amplificador

operacional para el aislamiento, cuya ganancia es unitaria, se utiliza porque su resistencia de entrada es alta, por lo tanto, extrae una corriente despreciable de la fuente de señal y se utiliza para acondicionar la señal eliminando el ruido de alta frecuencia, presentes en las lecturas entregada por los sensores hacia la tarjeta de adquisición, mediante un acople o desacople de la señal.

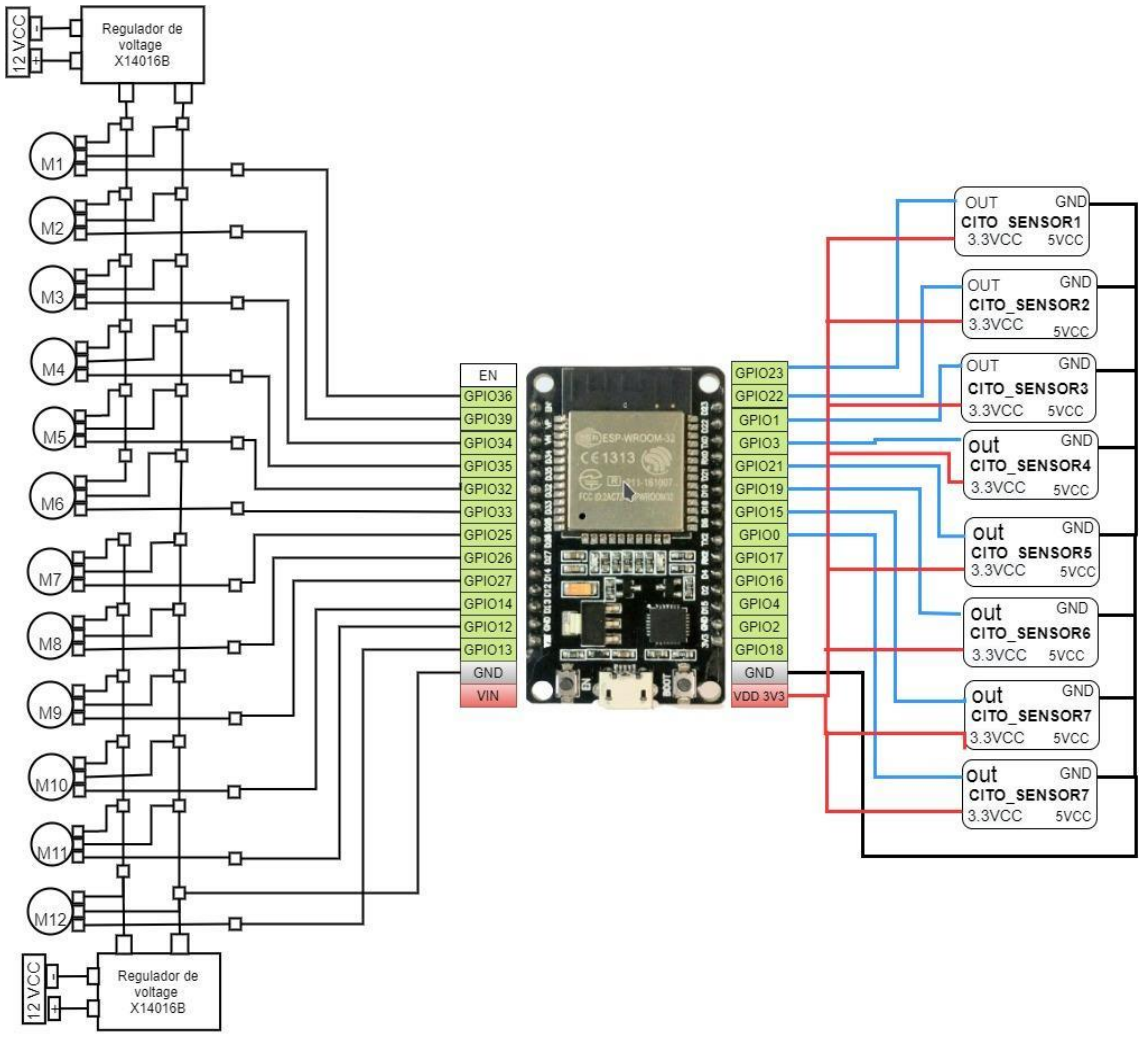


Figura 19. Cableado motores y sensores

2.5. Caracterización de los sensores de presión.

El sensor elegido para medir las fuerzas distribuidas sobre extremos de cada pie del robot, es el *Film Pressure Sensor* (DF9-40), en la Figura 2.23, se presenta el diagrama de conexión eléctrica del sensor.

Caracterización: Este proceso consiste en capturaron 25 muestras de voltaje entregadas por el sensor de fuerza ante diferentes pesos, estas estaban dentro del intervalo de 0 a 5 kg, con el propósito de hallar la curva característica de respuesta del sensor, la cual relaciona el voltaje de salida contra la masa (dada en gramos) aplicada sobre el dispositivo. A dicha curva se ajusta un polinomio característico del sensor. Para el análisis gráfico de estas muestras se usó la herramienta de cálculo matemático de Matlab, donde se procede a diseñar un algoritmo que por medio de ajuste de datos encuentra el polinomio de orden 3 que más se ajusta a las muestras entregadas por el sensor, a través del comando *Polyfit*.

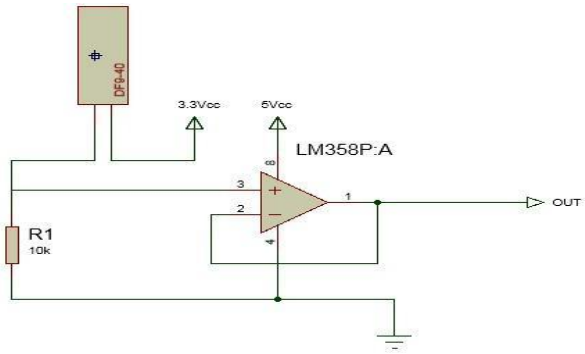


Figura 21. Circuito del sensor de presión

Del proceso antes descrito se obtuvo el siguiente polinomio característico, donde "y" es la cantidad de fuerza ejercida por el peso del robot y "x" es el voltaje de entrada del sistema.

$$y = \ 0.0004x^2 + \ 0.0102x^3 \qquad (9)$$

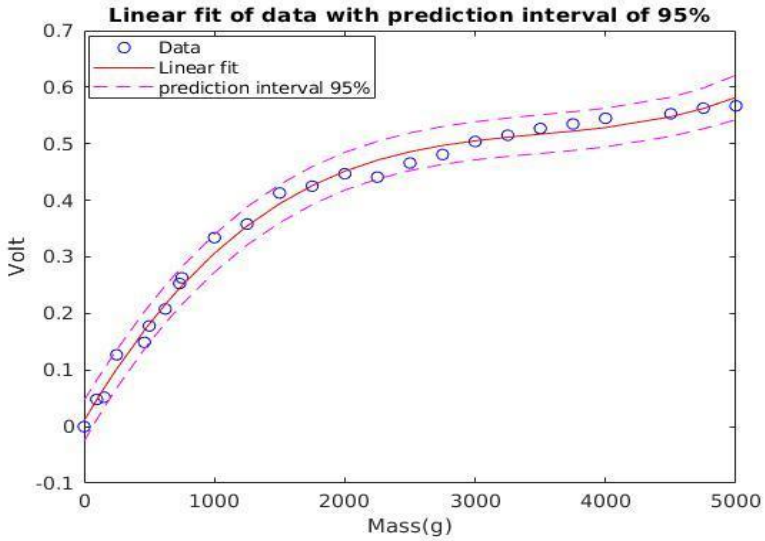


Figura 20. Curva característica del sensor DF9_40

En total se usaron 8 sensores, 4 por cada pie. Estos se situaron en los vértices de cada zapato con la finalidad de encontrar las coordenadas ZMP_x y ZMP_y, que definen el ZMP del robot bípedo sobre el plano de avance.

$$ZMP_x = \frac{\sum_{i=0}^{4} R_i X_i}{\sum_{i=0}^{4} R_i} \tag{10}$$

$$ZMP_y = \frac{\sum_{i=0}^{4} R_i Y_i}{\sum_{i=0}^{4} R_i} \tag{11}$$

Para el cálculo de estos componentes, se utilizó Matlab, en este se obtienen las lecturas de fuerza de cada sensor, se definen las distancias de los sensores respecto a los ejes *X* e *Y*, se crea el polígono de soporte de referencia y se realiza la sumatoria vectorial de fuerzas por cada componente, es así como se obtiene el *ZMP* medido del robot bípedo.

2.6. Prueba de los sensores de presión

Para conocer el CoP del robot se instaló un sensor de presión en cada esquina del zapato del robot propuesto, para comprobar el correcto funcionamiento de estos dispositivos. Para esto se realizó una prueba, la cual consiste en energizar el robot y posicionarlo en forma vertical, posteriormente se realizó el cálculo del centro de presión con la Ecuación (12) y (13) en Matlab. Adicionalmente, se plantearon tres referencias, la primera en forma de triángulo, la segunda en forma circular y la tercera en forma rectangular, las cuales se buscan seguir al manipular manualmente el robot bípedo, para mostrar el comportamiento del CoP según la posición del peso del robot, y de esta manera comprobar el correcto funcionamiento de los sensores.

$$xp = \frac{1}{F}\sum_{j=1}^{i=8} fs_i * ax_j \tag{12}$$

$$yp = \frac{1}{F}\sum_{j=1}^{i=8} fs_i * bx_j \tag{13}$$

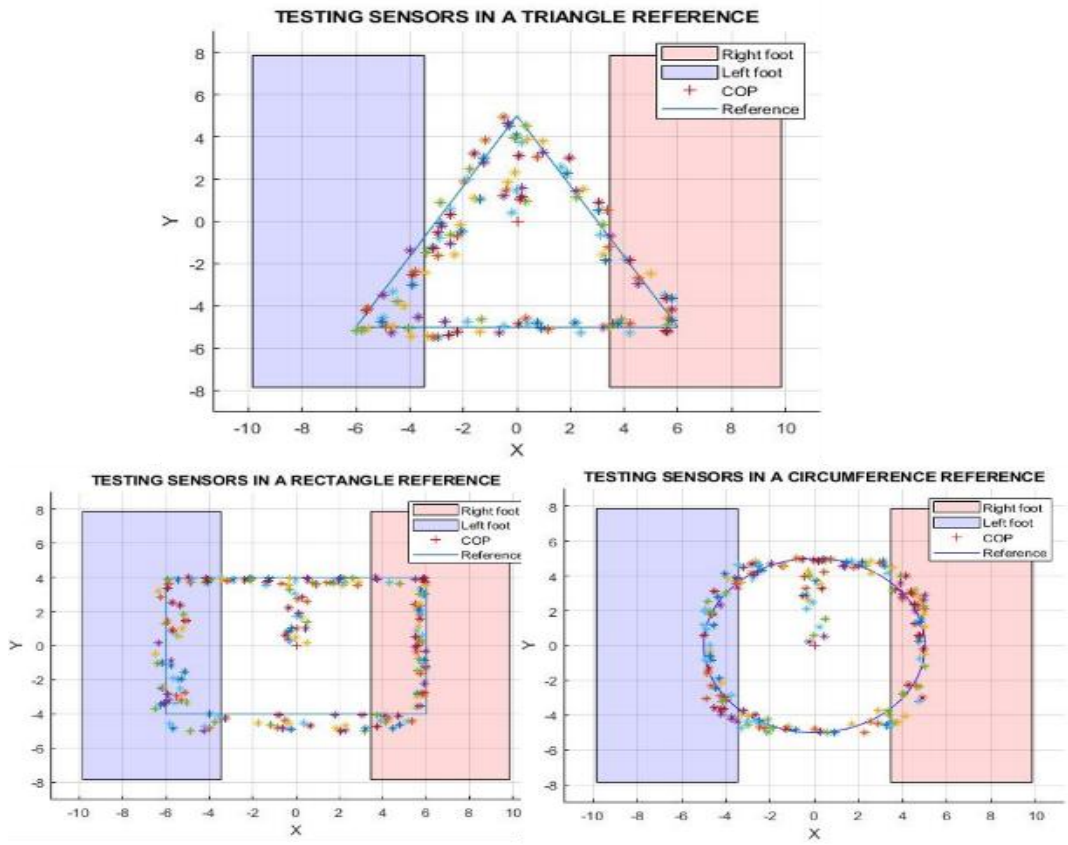


Figura 22. Prueba de sensores en una referencia triangular, rectangular y circular

En la Figura 2.25 el CoP cambia de posición, según la presión ejercida sobre el robot, el CoP está representado por el carácter simbólico asterisco (*). Los gráficos resultantes muestran el correcto funcionamiento de los sensores ya que es posible llevar los CoP del robot a posiciones cercanas a las referencias establecidas. Es importante decir que, debido a la ausencia de un controlador, los movimientos del centro de masa CoM son experimentos con apoyo manual, es decir, se manipulaba el robot posicionándolo en el piso, para luego realizar movimientos que conducen a cambios de presión en los sensores ZMP. Cabe señalar que, al hacerlo manualmente, las lecturas de presión ejercida sobre el robot presentan algunas desviaciones con respecto a las referencias. Lo anterior es de suma importancia ya que con esto se podrá implementar un controlador ZMP para el andar del robot bípedo en futuros trabajos.

2.6. Estructura real del robot bípedo

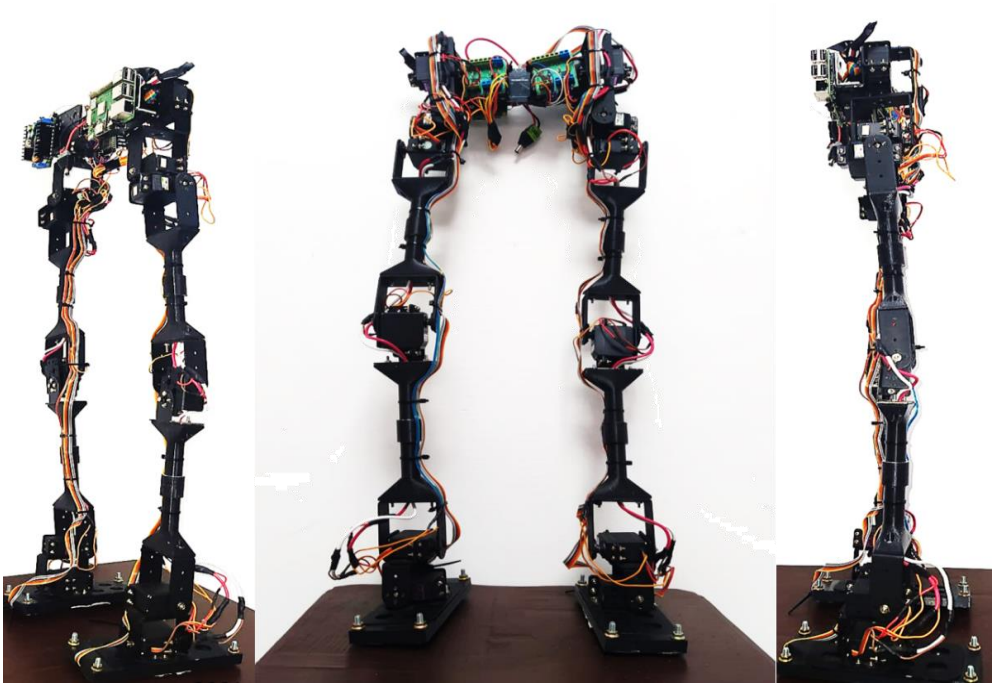


Figura 23 Robot bípedo propuesto.

2.7. Software del robot Bípedo

En esta sección se presentan los componentes software de comunicación del robot, tratando toda su programación, desde las instrucciones de alto nivel hasta las órdenes de movimiento a los motores, pasando por el procesamiento de la información de sus sensores.

ROS es un marco distribuido de código abierto de procesos, el cual posee nodos que se comunican entre sí, para ejecutar funciones como control de hardware, planificación de movimiento e integración de sensores, este marco permite varios lenguajes de programación y se puede ejecutar desde diferentes dispositivos. En este proyecto se creó el nodo maestro desde un PC remoto usando el sistema de cómputo numérico Matlab, el cual se encarga de recibir la información de los sensores, realizar el cálculo del ZMP sobre el suelo y enviar al robot posiciones de referencia a los motores mediante un nodo suscriptor y un publicador por medio de WIFI. En el robot se encuentra una tarjeta Raspberry Pi 3B+ y una tarjeta ESP32, las cuales se comunican mediante el protocolo rosserial [18]. La tarjeta ESP32 cumple la función de enviar las posiciones articulares de referencia traducidos a señales PWM hacia los motores y también obtener la lectura analógica de los sensores debido que la Raspberry Pi 3B+ solo cuenta con pines GPIO y no posee la cantidad necesaria de conversores analógicos digital (ADC, Analog Digital

Converter). Por otro lado, en la tarjeta Raspberry se encuentran dos nodos creados en Python, el primero hace el papel del publicador; encargado de enviar la información obtenida de los sensores, y el segundo es un nodo suscriptor el cual tiene como funcionalidad el recibir la información desde el PC remoto como estación de control.

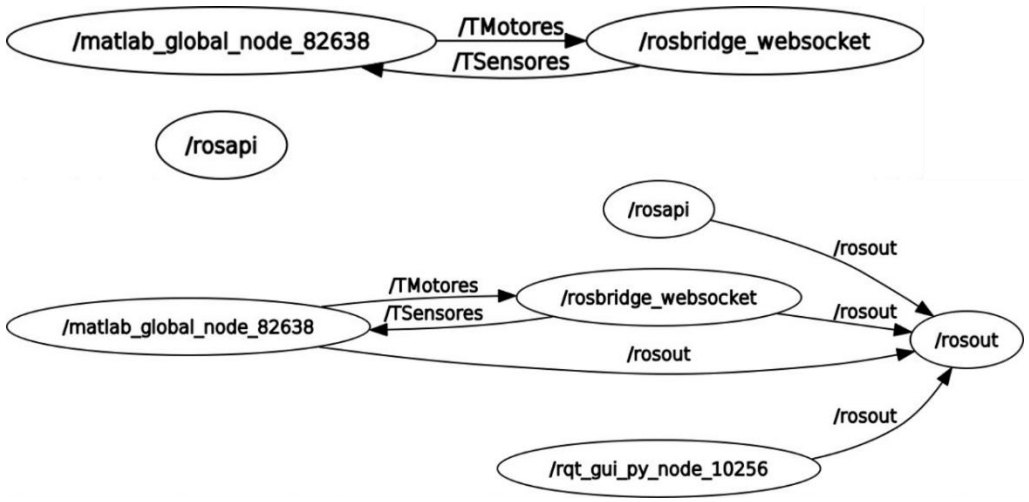


Figura 247. Diagrama de nodos, servicios y tópicos.

Cabe resaltar que los nodos; publicador y suscriptor tienen una similitud con una comunicación cliente servidor, en este caso, el suscriptor (cliente) solicita información al publicador (servidor) y éste comparte dicha información. La comunicación realizada para el intercambio de información para manipular el comportamiento articular del robot bípedo es la siguiente: Inicialmente se crea el servidor de ROS bridge, el cual permite la comunicación desde el centro de control remoto hasta el robot a través de WIFI. Luego de establecer la comunicación mencionada, es necesario la obtención y envío de las señales de los sensores con la tarjeta ESP32 y compartirla con el centro de cómputo desde la tarjeta Raspberry Pi, mediante el nodo PubSensores y el tópico TSensores. Después se realiza la recepción, la manipulación de los datos y la devolución de información desde el centro de cómputo hacia el robot, teniendo en cuenta que la recepción se hace por el tópico TSensores y el nodo SusSensores, y la devolución se efectúa por el tópico TMotores y el nodo PubMotores. Desde la tarjeta Raspberry Pi se hace la captura de los datos enviados desde el centro de cómputo y se procede a compartirlos con la tarjeta ESP32, la captura de los datos se hace mediante el tópico TMotores y el nodo SusMotores, y se comparten con la ESP32 por el puerto serial. Finalmente se pasa la señal de referencia articular a cada uno de los motores desde la ESP32, como se muestra en la Figura 2.28. A continuación, se explica detalladamente los nodos y algoritmos de programación que se encuentran interactuando para la comunicación del robot bípedo:

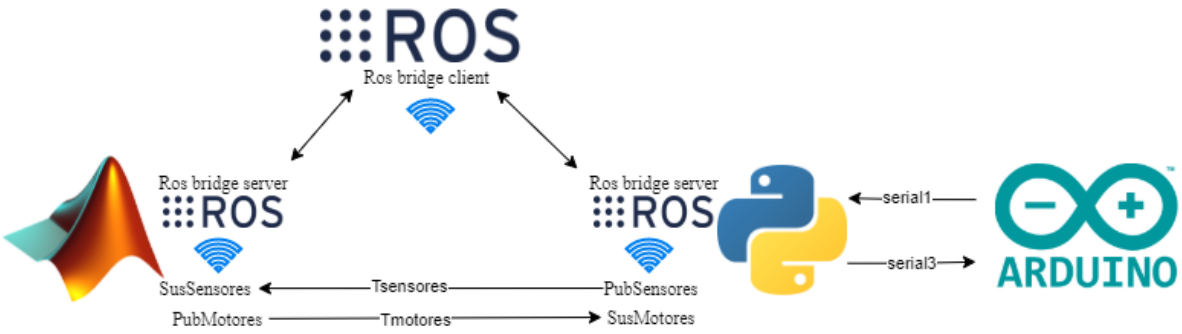


Figura 25. Comunicación del robot bípedo

2.7.1. Rosbridge:

Debido a que el robot trabaja de manera remota desde un centro de cómputo donde se realizan los cálculos y análisis numéricos, es necesario mantener una comunicación hacia el robot, en este caso se realiza por WIFI, de ahí surge la necesidad de emplear rosbridge, que es un protocolo que permite intercambiar información entre ROS y cualquier otra aplicación o programas capaces de analizar de objetos de JavaScript (JSON, *JavaScript Object Notation*). Hay una variedad de interfaces que interactúan con rosbridge, incluidos los servidores *WebSocket*, para interactuar desde navegadores web. Este protocolo utiliza JSON como transporte de mensajes para permitir el acceso a las funcionalidades de ROS como son: publicación, suscripción, llamadas de servicio, entre otros. Cabe resaltar que, para iniciar la comunicación del robot, se debe asegurar de iniciar todos los servicios.

2.7.2. Obtención y envío de señales:

La recepción de las señales de los sensores y él envió de señales a los motores, se lleva a cabo con la tarjeta ESP32, que se encuentra ubicada en el robot. La programación de esta tarjeta se realiza en el entorno de desarrollo integrado de Arduino. La ESP32 interactúa con la tarjeta Raspberry Pi donde se encuentran los nodos de ROS que se comunican por WIFI con el centro de control. Inicialmente se debe obtener la información de los sensores de presión, los cuales emiten una señal analógica que son leídos por la tarjeta ESP32, la cual recibe datos en un rango de 0 a 1023, sin embargo, es necesario escalar estos valores a un rango de 0 a 5 voltios, para ello se utiliza la función “map ()”,es preciso mencionar que el proceso se repite ocho veces, una por cada sensor. Posterior a esto se almacenan estos valores en una variable de tipo cadena, la cual será enviada mediante comunicación serial a la tarjeta Raspberry Pi con el comando “println ()”.
Por otro lado, en este algoritmo también se recibe de la tarjeta Raspberry una cadena por el puerto serie, que contiene datos que se envían mediante una señal PWM a los motores del robot. Para recibir la información enviada desde Python a la plataforma Arduino. Después de recibir esta cadena, es necesario separar la información correspondiente para

cada motor, por ende, se utiliza el comando “indexOf ()”, el cual localiza un carácter, en este caso es el “;”, encargado de separar los diferentes datos dentro de la cadena. También se utiliza el comando “substract()” cuya función es obtener una subcadena de una cadena, de esta manera se separan los datos y se van almacenando en una variable. Después de almacenarse se convierten en datos de tipo flotante para ser enviados como una señal PWM a cada motor.

2.7.3. Recibir información de la tarjeta ESP32 y envió al centro de cómputo desde la tarjeta Raspberry Pi:

La recepción de la información de señales desde la tarjeta ESP32 y el envío de esta información a Matlab se hace en la tarjeta controladora Raspberry Pi , la cual se encuentra en el robot, esta tiene un *script* desarrollado en Python para realizar las tareas anteriormente mencionadas, este algoritmo de programación utiliza las librerías time, serial y roslybpi, la primera es un módulo que proporciona varias funciones relacionadas con el tiempo, la segunda permite emplear el puerto serie y la última permite usar Python e IronPython para interactuar con ROS utilizando WebSockets con el fin de conectarse a rosbridge y dando paso a usar las diferentes características de ROS. Para establecer la conexión con el puerto serial, se realizó una función la cual será ejecuta posteriormente cuando se encuentre conectado al servidor de rosbrige, esta función se la llama “leerArduino()”, en esta función se crea un objeto PySerial**,** al cual se le pasan los parámetros del puerto serie con el que se está trabajando, como lo son el nombre del puerto y la velocidad, cabe resaltar que esta velocidad debe ser la misma con la que se trabaja en la tarjeta ESP32, para este caso se emplea el puerto '/dev/ttyACM0' y una velocidad de 115200 baudios, luego se emite el comando “open()” para abrir el puerto serial, después se da la orden “readline()” en el objeto serial para leer una línea enviada por Arduino, también se ejecuta el método “decode()” que decodifica la cadena de datos, luego se usa “flush()” para limpiar el puerto serial, y finalmente se cierra el puerto con “close()”**.** En primera instancia se crea un cliente el cual se conecta al servidor de rosbridge, para esto es necesario el nombre o dirección IP del host de ROSbridge y al puerto del ROSbridge y posteriormente se crea el nodo y el tópico por el cual otros nodos pueden acceder a la información que en este se albergar, a estos se les asignó el nombre de “PubSensores” y “TSensores” respectivamente, adicionalmente se especifica el tipo de variable que contiene, en este caso 'std_msgs/String'. Luego se crea un ciclo “while”, el cual se ejecuta mientras el cliente se encuentre conectado al servidor de rosbridge, su función es recibir la información del puerto serial y también publicar los datos.

2.7.4. Recepción y envío de información desde el centro de cómputo hacia el robot:

El centro de cómputo se comunica mediante WIFI con el robot bípedo usando rosbridge. Lo que permite la interacción entre el centro de control y la placa Raspberry Pi, este es el algoritmo que realiza las acciones de recibir y enviar la información, el cual está implementado en un script en Matlab. Primero es importante revisar si no hay nodos

ejecutándose con anterioridad, esto se hace con el comando “rosshutdown ()” y posteriormente se crea el cliente hacia rosbridge con sus respectivos parámetros. Para recibir la información se crea un nodo suscriptor llamado SusSensores con el comando “rossubscriber()”, después se usa la función “receive()”, para recibir los datos en forma de mensaje ROS, cabe resaltar que la información recibida es una cadena que se debe separar para conocer los valores individuales de cada sensor, por ende se usa el comando “strsplit()”, que divide la cadena en los delimitadores especificados, en este caso en un “;”, y finalmente se convierten los datos de string a doublé.

Para finalizar, se crea el nodo publicador denominado PubMotores con el comando “rospublisher()” y se asigna el tópico por el cual se hará el envío de la información del esfuerzo de control, en este caso es el tópico denominado TMotores, después se usa la función “rosmessage()” que crea un mensaje vacío determinado por el publicador, en este caso el publicador PubMotores, para posteriormente asignar la cadena que contiene los datos para cada motor y finalmente realizar el envío del mensaje con el comando “send()”. Cabe resaltar que cuando se envía o recibe mensajes mediante nodos ROS en Matlab se ejecutan una sola vez, por ende, se creó una función que se ejecuta mientras Matlab se encuentra conectado con rosbridge para que se realice la recepción y envío de datos.

2.7.5. Recepción de información desde el robot bípedo y envío a la tarjeta ESP32:

La recepción de información se hace en la tarjeta controladora Raspberry Pi la cual se encuentra en el robot, por medio de comunicación WIFI, cabe resaltar que el algoritmo de programación está en Python y las librerías que se usan son las ya mencionadas anteriormente. Lo primero que se realiza es la comunicación con rosbridge mediante el comando “roslibpy**.**Ros()”, posteriormente se hace la suscripción al tópico perteneciente al nodo en Matlab (TMotores) con la orden “roslibpy**.**Topic()”, además se hace la recepción de los datos a través de “subscribe()”.
Para enviar estos datos hasta la placa ESP32, se crea el objeto serial con el comando “serial.Serial#()”, al cual se le asignan sus respectivos parámetros, posteriormente se limpia el puerto con el comando “flush()”, luego se le asigna los datos a enviar mediante la función “write()”, después se decodifica la cadena usando “readline()” en este caso mediante 'utf-8’ y finalmente se envía con el comando” print()”.

Capítulo 3: Modelo Cinemático y dinámico simplificado del robot

En este capítulo se describen los modelos cinemático y dinámico del robot bípedo, modelos que constituyen elementos de base para la experimentación de trayectorias articulares de marcha en etapas futuras que darán continuidad al proyecto. El modelo geométrico directo se basa en la formulación de Denavit-Hartenberg, el cual determina la posición y orientación que toma el pie del robot bípedo, mientras el inverso se basa en el método de Paul para encontrar las diferentes posiciones que deben tomar cada una de las articulaciones que componen al robot humanoide, con el objetivo de llevar al órgano terminal a una posición y orientación deseada, estos modelo son realizados para un robot de 12 GDL, además para calcular el modelo dinámico del robot se optó por el sistema *Cart-Table*, técnica que permite encontrar la relación entre el ZMP y el CoM.

3.1. Modelo Geométrico del robot bípedo:

Para representar geométricamente las características físicas del robot bípedo propuesto, se ha optado por el método de Denavit-Hartenberg (1955). En primera instancia se han referenciado los ejes X y Z para cada articulación como se muestra en Figura 3.1 adicionalmente en Figura 3.2 se representa la distribución de distancias de separación que hay entre cada articulación. Cabe resaltar que el marco de referencia cero (Xo, Yo) del robot se encuentra en el centro de la cadera, mientras que el marco de referencia inercial (Xr, Yr) está ubicado sobre el suelo, en el punto medio entre los pies. El marco de referencia inercial se encuentra desplazado en el plano Z con respecto al marco de referencia cero

En la Tabla 1 se encuentra especificado el valor de la longitud existente entre cada articulación, es preciso decir que estas medidas son exactamente iguales tanto en la pierna derecha como izquierda. Por otro lado, al tomar las diferentes longitudes se referenció respecto al centro de los soportes de sujeción de cada servomotor, teniendo ese punto como la posición donde se encuentra la articulación.

Retomando el proceso propuesto por Denavit-Hartenberg, y siguiendo el procedimiento y recomendaciones dadas por el mismo, se prosigue a encontrar la tabla de parámetros geométricos del robot humanoide, los cuales se pueden evidenciar en la Tabla 2.

3.1.2. Modelo geométrico directo:

Este modelo (T_n^0), determina la posición y orientación que toma el pie del robot bípedo, a partir del conocimiento previo de las posiciones en las que se encuentran cada una de las articulaciones del robot. Para dar solución a este modelo se requiere de dos componentes,

el primero es la matriz de transformación (T_j^{j-1}) y el segundo es la tabla de parámetros de Denavit-Hartenberg presentados en la Tabla 2, con estos se precede de la siguiente manera:

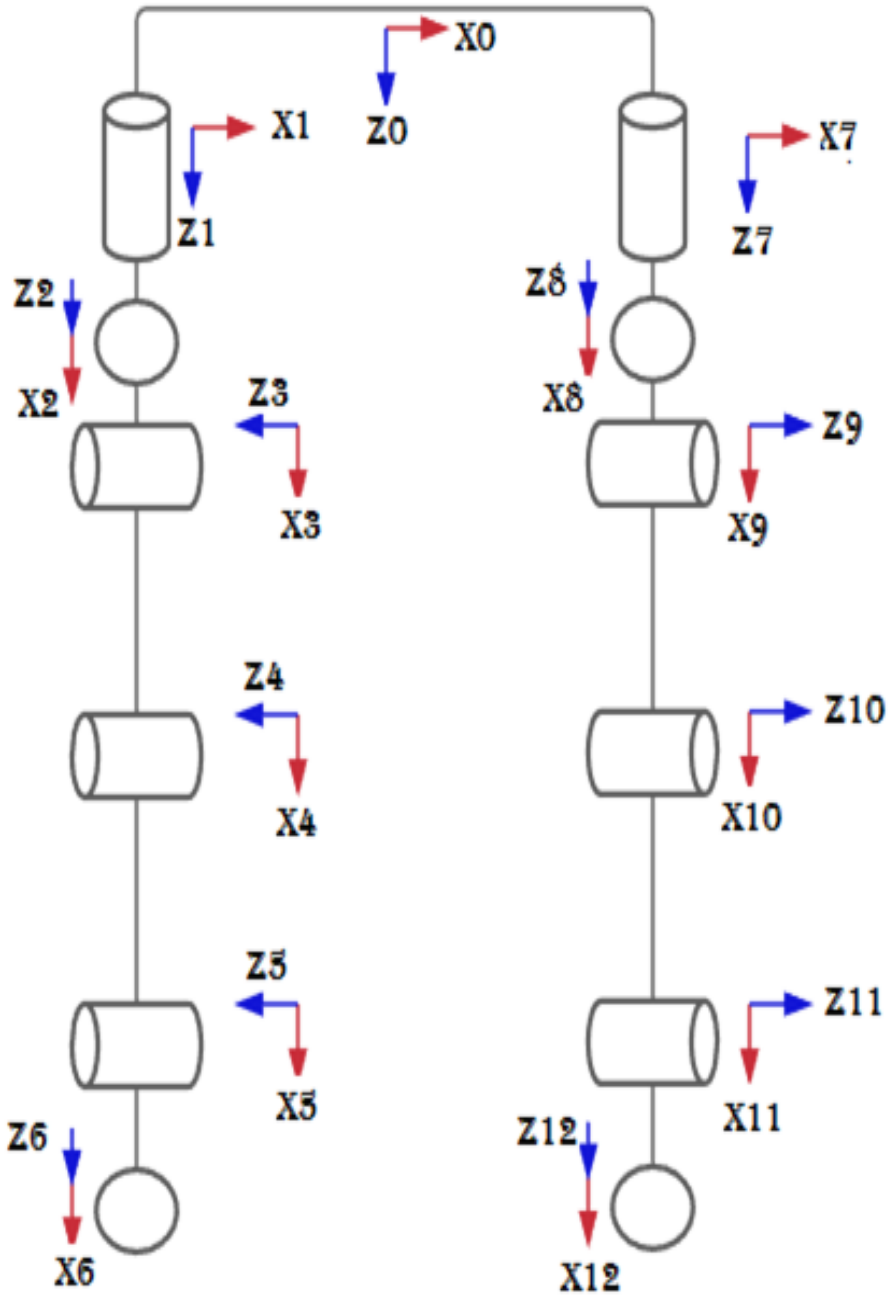


Figura 26. Asignación de los ejes X y Z para cada articulación

.

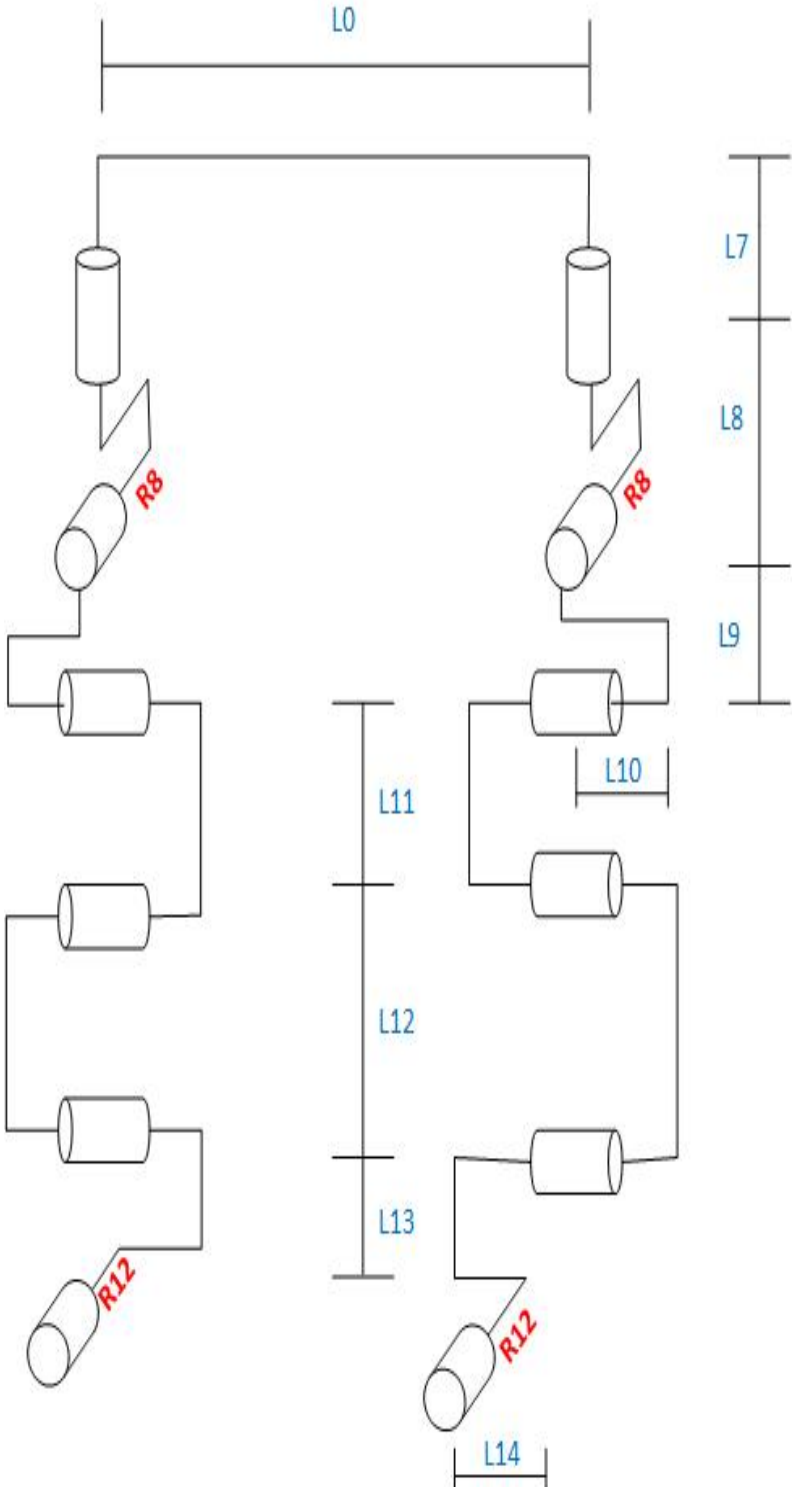


Figura 27. Distancias entre articulaciones

Li	Nombre	Medida(mm)
L0	longitud 0 - longitud pelvis	167.60
L7	longitud 1 - longitud cadera 1 izquierda	27.89
L8	longitud 2 - longitud cadera 2 izquierda	84.71
L9	longitud 3 - longitud desplazamiento en cadera izquierda	31.23
L10	longitud 4 - longitud fémur izquierdo	14.20
L11	longitud 5 - longitud peroné izquierdo	234
L12	longitud 6 - longitud tobillo izquierdo	247.5
L13	longitud 7 - longitud desplazamiento tobillo izquierdo	41.21
R12	longitud 8 - longitud cadera 1 derecha	9.64

Tabla 1: Longitudes de los componentes del robot

J	Σ_j	α_j	D_j	θ_j	R_j
1	0	0	Lo/2	θ1	L7
2	0	90	0	θ2	0
3	0	90	L9	θ3	L10
4	0	0	L11	θ4	R8
5	0	0	L12	θ5	0
6	0	-90	L13	θ6	R12
7	0	0	Lo/2	θ7	L7
8	0	-90	0	θ8	R8
9	0	-90	L9	θ9	L10
10	0	0	L11	θ10	0
11	0	0	L12	θ11	0
12	0	90	L12	θ12	R12

Tabla 2: Tabla de parámetros geométricos del robot bípedo propuesto

Inicialmente, se obtiene de la tabla de parámetros los valores de θ, α, d y r, y se remplazan en la ecuación 12, donde *j* representa el número de la articulación del robot asociada a estos valores. Por ejemplo, para la articulación 1 ($j = 1$) se tienen los valores en tabla 3.

$$T_j^{j-1} = \begin{bmatrix} C\theta_j & -S\theta_j & 0 & d_j \\ C\alpha_j S\theta_j & C\alpha_j C\theta & -S\alpha_j & -r_j S\alpha_j \\ S\alpha_j S\theta_j & S\alpha_j C\theta_j & C\alpha_j & r_j C\alpha_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

j	**Σ_j**	**α_j**	**D_j**	**θ_j**	**R_j**
1	0	0	Lo/2	θ1	L7

Tabla 3: *Tabla de parámetros geométricos para la articulación 1*

Reemplazando en la Ecuación (13), se obtiene la matriz de transformación correspondiente a la primera articulación del robot bípedo propuesto.

$$T_1^0 = \begin{bmatrix} C\theta_1 & -S\theta_j & 0 & Lo/2 \\ S\theta_1 & C\theta & 0 & 0 \\ 0 & 0 & 0 & \text{L7} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}$$

El robot posee 12 GDL, es por esto que se deben encontrar cada una de las matrices de transformación, siguiendo el proceso antes visto para cada una de las articulaciones. A continuación, se presentan las matrices de una de las piernas del robot. Hay que resaltar que se trabaja con una sola pierna ya que la representación y el comportamiento de la otra es idéntico.

$$TR = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & hr \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

$$T_1^0 = \begin{bmatrix} C\theta_1 & -S\theta_1 & 0 & Lo/2 \\ S\theta_1 & C\theta_1 & 0 & 0 \\ 0 & 0 & 1 & L7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{16}$$

$$T_2^1 = \begin{bmatrix} C\theta_2 & -S\theta_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ S\theta_2 & C\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{17}$$

$$T_3^2 = \begin{bmatrix} C\theta_3 & -S\theta_3 & 0 & L9 \\ 0 & 0 & -1 & -L10 \\ S\theta_3 & -S\theta_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (18) \qquad T_4^3 = \begin{bmatrix} C\theta_4 & -S\theta_4 & 0 & \mathrm{L11} \\ S\theta_4 & C\theta_4 & 0 & 0 \\ 0 & 0 & 1 & R8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (19)$$

$$T_5^4 = \begin{bmatrix} C\theta_5 & -S\theta_5 & 0 & L12 \\ S\theta_5 & C\theta_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (20) \qquad T_6^5 = \begin{bmatrix} C\theta_6 & -S\theta_6 & 0 & L13 \\ 0 & 0 & 1 & R12 \\ -S\theta_6 & -C\theta_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (21)$$

Continuando con el cálculo del MGD del robot, se efectúan multiplicaciones sucesivas de las seis matrices de transformación encontradas anteriormente ($MGD = T_6^0 = T_1^0 T_2^1 \dots T_5^4 T_6^5$), estos procedimientos se realizan en el programa Matlab, donde se introducen cada una de las matrices y se efectúan los cálculos.

Como resultado, en la Ecuación (21) se aprecia el MGD del robot propuesto. Hay que resaltar que los valores de las constantes se encuentran en la Tabla 1. Adicionalmente, para comprender y dar un mejor orden al documento, no se indican los resultados al operar entre las constantes por su extensión.

$$T_6^0 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -C\theta_6 K_3 - S\theta_1 S\theta_2 S\theta_6 & S\theta_6 K_3 - C\theta_6 S\theta_1 S\theta_2 & K_4 & R_{12}K_4 - L_{13}K_3 - L_{11}K_7 - L_{12}K_5 + L_9 C\theta_2 S\theta_1 + K_{10} \\ C\theta_2 S\theta_6 + K_1 C\theta_6 S\theta_2 & C\theta_2 C\theta_6 - K_1 S\theta_2 S\theta_6 & -K_2 S\theta_2 & L_7 + h_r - L_{10} C\theta_2 - R_8 C\theta_2 + L_9 S\theta_2 + K_{11} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (22)$$

Donde:

$$K_1 = C(\theta_3 + \theta_4 + \theta_5) \quad (23)$$

$$K_2 = S(\theta_3 + \theta_4 + \theta_5) \quad (24)$$

$$K_3 = C\theta_5 K_5 - S\theta_5 K_6 \quad (25)$$

$$K_4 = C\theta_5 K_6 + S\theta_5 K_5 \quad (26)$$

$$K_5 = C\theta_4 K_7 - S\theta_4 K_8 \quad (27)$$

$$K_6 = C\theta_4 K_8 + S\theta_4 K_7 \quad (28)$$

$$K_7 = K_9 - C\theta_2 C\theta_3 S\theta_1 \quad (29)$$

$$K_8 = K_9 - C\theta_2 S\theta_1 S\theta_3 \quad (30)$$

$$K_9 = C\theta_1 + S\theta_3 \quad (31)$$

$$K_{10} = L_{10} S\theta_1 S\theta_2 + R_8 S\theta_1 S\theta_2 \quad (32)$$

$$K_{11} = L_{12} C(\theta_3 + \theta_4) S\theta_2 + L_{11} C\theta_3 S\theta_2 + L_{13} K_1 S\theta_2 - R_{12} K_2 S\theta_2 \quad (33)$$

3.1.2. Modelo geométrico Inverso:

El objetivo de este modelo, es encontrar las diferentes posiciones que deben tomar cada una de las articulaciones que componen al robot humanoide, con el fin de llevar al órgano terminal a una posición y orientación deseada. Es preciso decir, que una de las desventajas de este modelo es la posibilidad de encontrar varias soluciones para las posiciones articulares ante una única posición cartesiana deseada.

El modelo geométrico inverso puede encontrarse por varios métodos, pero el más común y el que mejor se adecua a este trabajo es el Método de Paul. Este método parte de la premisa de conocer la posición y orientación que se quiere que tome el eje terminal u_0 y la Matriz de transformación (T_n^0) del robot, las cuales se presentan en las Ecuaciones (34) y (35) respectivamente.

$$U_0 = \begin{bmatrix} S_x & n_x & a_x & P_x \\ S_y & n_y & a_y & P_y \\ S_z & n_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{34}$$

$$T_n^0 = T_1^0 T_2^1 \dots T_n^{n-1} \tag{35}$$

El método consiste en despejar las incógnitas (q) que definen las coordenadas articulares del robot humanoide, para esto se realiza una serie de multiplicaciones de tal manera que cada término de la Ecuación (34) se vaya eliminando.

$$U_0 = T_1^0 \, T_2^1 \dots T_6^5 \tag{36}$$

$$U_1 = T_0^1 U_0 = T_2^1 T_3^2 \dots T_6^5 \tag{37}$$

Como se evidencia en la Ecuación (35) y Ecuación (36), cada término de la Ecuación (34) se va eliminando en cada multiplicación de dos matrices consecutivas, hasta encontrar las ecuaciones que permitan hallar las incógnitas en su totalidad, a continuación, en la Ecuación (38) se representa la solución para una de las piernas del robot humanoide de 12 GDL.

$$T_4^5 U_4 = T_6^5 \tag{38}$$

Finalmente se resuelve el modelo cinemático inverso por método de Paul [34]. Es preciso decir que los cálculos de este modelo no se han realizado manualmente, por el contrario, se utiliza la herramienta *Simscape Multibody* de Matlab, para agilizar el proceso de desarrollo y evitar errores de cálculo matemático, además pensando en que en esta etapa no se trabaja en control de equilibrio, control de marcha o control central donde se requiere de este modelo, sin embargo, se especifica cómo se realizan los cálculos para ser tenidos en cuenta en la etapa siguiente del proyecto.

3.1.3. Modelo Simplificado *Cart-Table*

Para modelar la dinámica del robot bípedo, se ha optado por el sistema *Cart-Table*, cuyo propósito es generar una ecuación simplificada de la dinámica de caminar. Este modelo permite encontrar la relación entre el ZMP, y el CoM. Está representado por medio de un carro que se desplaza sobre la superficie horizontal de una mesa. La masa M del carro representa la masa total concentrada del robot, y su ubicación representa la posición del CoM, el cual se desliza con aceleración $\ddot{x}$ sobre una mesa con masa nula y Z_c es la altura constante del centro de gravedad como se muestra en la Figura 1.2. Es importante destacar que el polígono de soporte formado en fase de apoyo simple y apoyo doble es representado por la base de sustentación de la mesa. De esta manera, cuando el robot se encuentra en fase de soporte simple la base de la mesa coincide con la planta del pie de soporte del robot [22] [6].

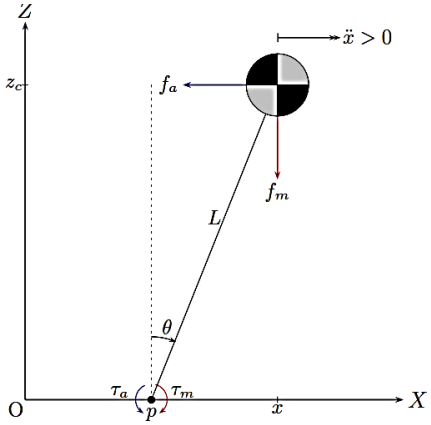


Figura 3.3. Diagrama de fuerzas ejercida sobre la masa M [6]

La aceleración debido a la gravedad de la tierra genera una fuerza de magnitud $F_m = Mg$ en sentido negativo al eje Z. Por otra parte, es indispensable mantener constante la altura del CoM en el nivel Z_c por tanto es necesario aplicar una fuerza de magnitud $F_a = m\ddot{x}$ en dirección contraria al movimiento de avance del robot [22].

Las fuerzas F_m y F_a generan un torque alrededor del punto p ubicado sobre el eje X, como se muestran en las Ecuaciones (39) y (40) [26]:

$$T_m(p) = F_m(x - p) \tag{39}$$

$$T_a(p) = F_a Z_c \tag{40}$$

A partir de las ecuaciones anteriores se puede obtener el momento total resultante alrededor del punto p.

$$T(p) \; = \; F_m(x - p) - F_a Z_c \tag{41}$$

$$T(p) = Mg(x - p) - M\,Z_c\,\ddot{x} \tag{42}$$

Existe un punto p donde se cancelan los torques $\tau_c(p)$, $\tau_m(p)$, y $\tau_a(p)$ denominado punto de momento cero P_{ZMP}, por lo cual $\tau(p)$ se hace cero, y el robot no tendrá fuerza alguna que lo desequilibre. De esta forma la ecuación anterior pasa a ser la siguiente:

$$P_{ZMP} = x - \frac{Z_c}{g}\ddot{x} \tag{43}$$

Posteriormente se aplica la transformación de Laplace a la ecuación anterior, para lo cual se consideran condiciones iniciales iguales a cero. Esto permite obtener la función de transferencia del sistema, Observando la relación del ZMP y CoM mencionada anteriormente [26].

$$\frac{x(s)}{P_{zmp}(s)} = \frac{1}{1 - \frac{Z_c}{g}s^2} \tag{44}$$

De la función de transferencia resultante, exactamente al examinar su polinomio característico, se evidencia que posee dos polos reales:

$$s_{1,2} = \pm\sqrt{\frac{g}{Z_c}} \tag{45}$$

Debido a que el polo S_1está ubicado en el semiplano derecho del plano complejo, se deduce que el sistema es inestable. Indicando que para una entrada limitada del ZMP deseado, la salida $x(t)$ resultante diverge. Por consiguiente, se debe solucionar el problema de estabilidad mediante un lazo de control en un nuevo proyecto. Por otro lado, se debe analizar el modelo *Cart-Table* tanto en el plano sagital como el frontal, en consecuencia, se deduce el nuevo sistema de ecuaciones diferenciales que describe el sistema en ambos planos [22]:

$$P_{ZMPx} = x - \frac{Z_c}{g}\ddot{x} \tag{46}$$

$$P_{ZMPy} = y - \frac{Z_c}{g}\ddot{y} \tag{47}$$

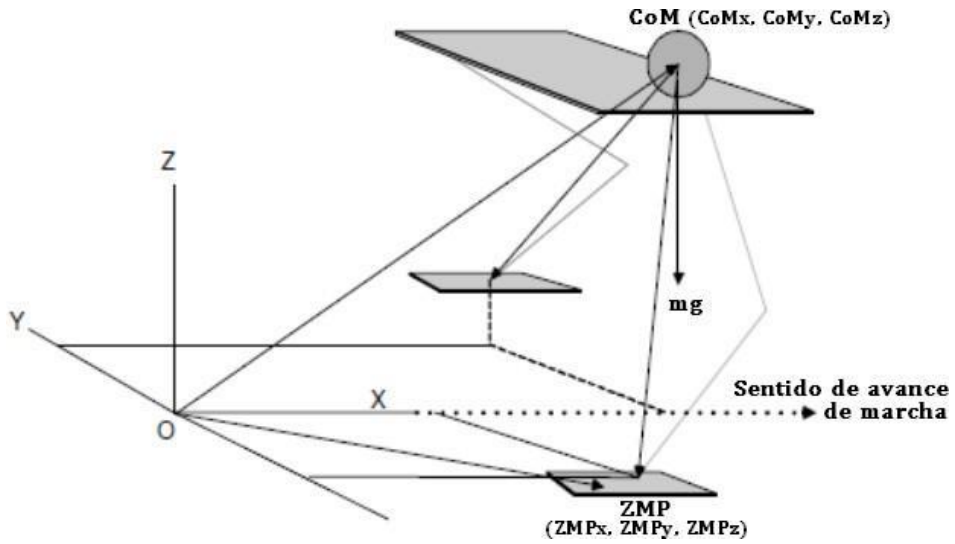


Figura 284. Representación del modelo Cart-Table en el espacio [6]

3.1.4. Validación cinemática del Robot propuesto

A través de la herramienta Simscape Multibody Link que ofrece la plataforma de Matlab, es posible comprobar en manera de modelado y simulación de movimientos, el correcto cálculo de la cinemática del robot propuesto.

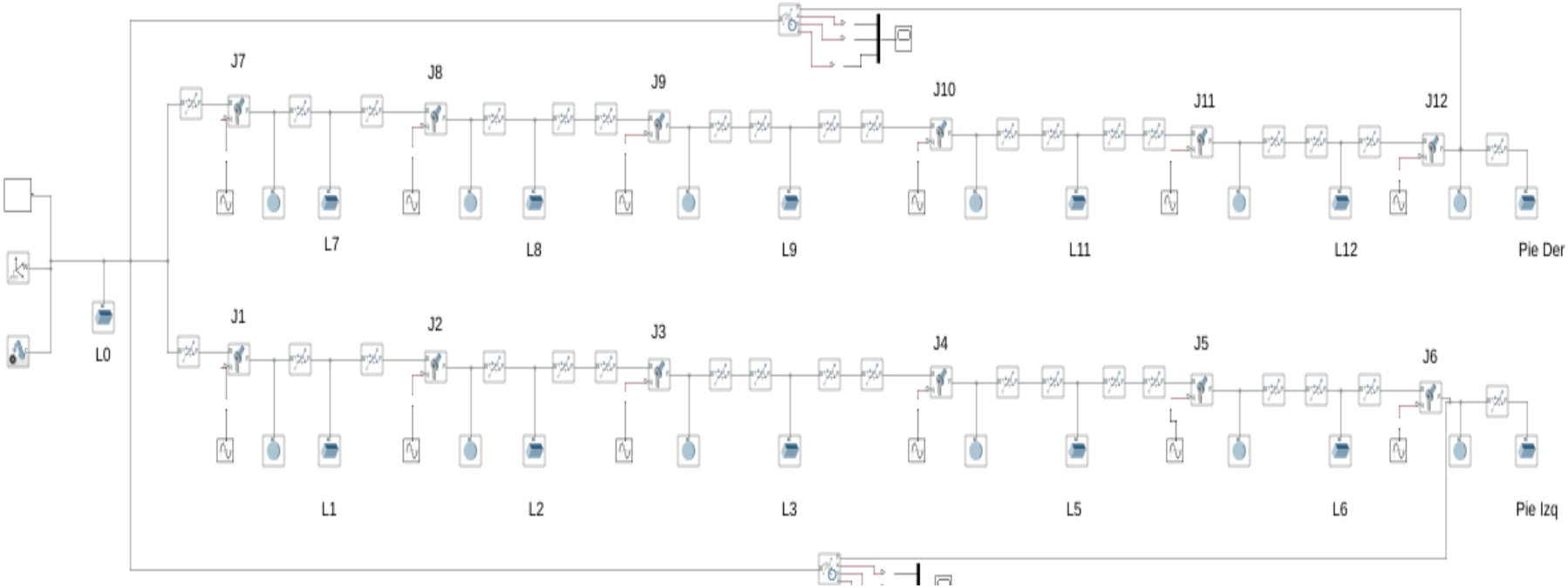


Figura 3.5. Modelo Multibody del robot propuesto

Para comprobar la cinemática del modelo del robot en Multibody, se ha optado por aplicar señales de tipo sinusoidales a cada articulación, con la finalidad de llevar los pies del robot a una posición deseada, como se puede evidenciar a continuación.

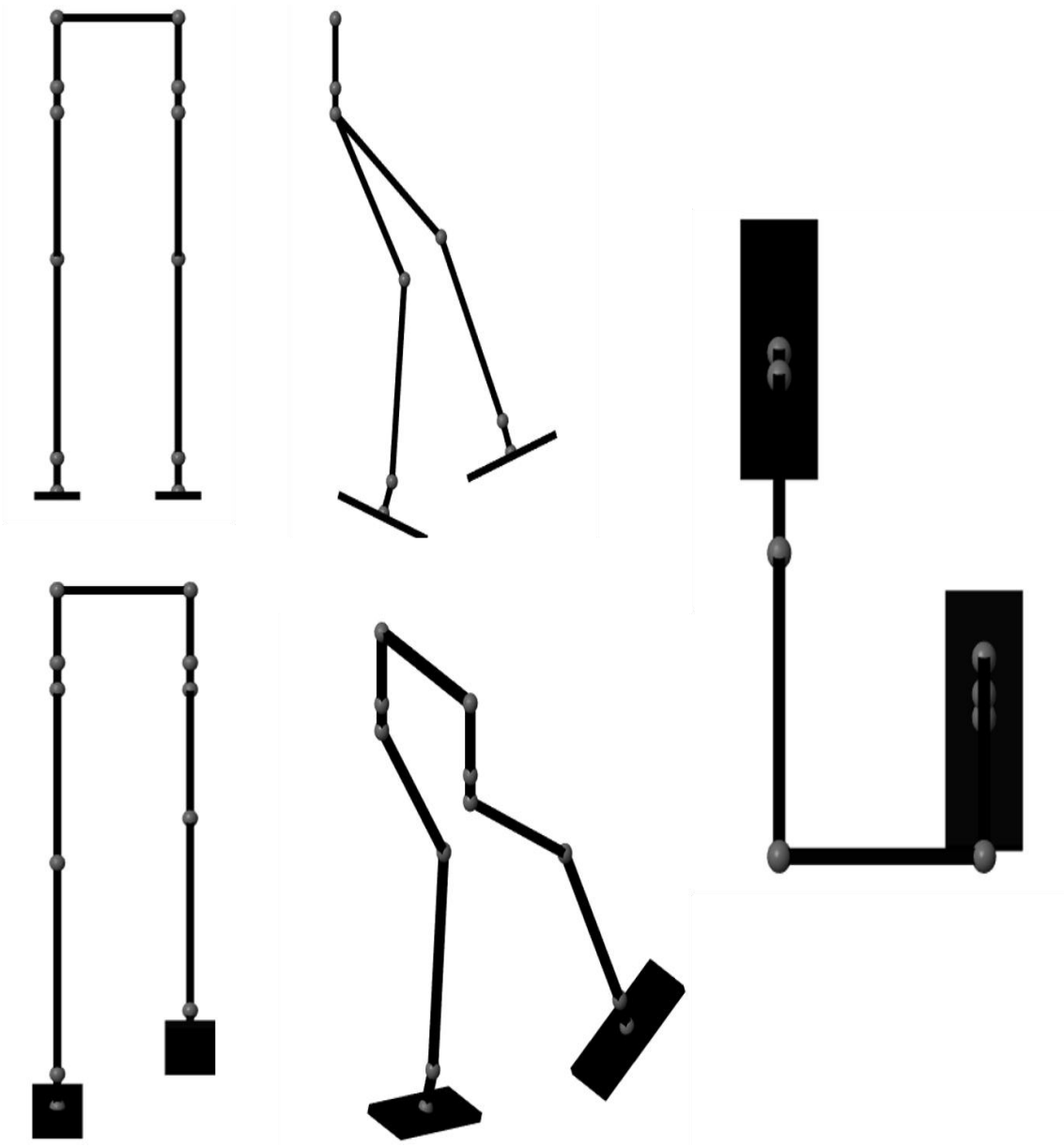


Figura 296. Modelo 3D en simscape multybody del robot, posturas ante ente entradas sinusoidales

Capítulo 4: Pruebas de funcionamiento del robot bípedo.

En este capítulo se encuentra documentado el método utilizado para encontrar las trayectorias articulares para el robot bípedo y finalmente los experimentos realizados para comprobar el funcionamiento del prototipo y evidenciar el seguimiento de las trayectorias articulares.

4.1. Generación de Trayectorias Articulares Mediante Ondas Sinusoidales acopladas

Con el propósito de determinar la capacidad de seguimiento de trayectorias articulares predefinidas a través del robot propuesto, es necesario el uso de alguna técnica para la generación de 12 trayectorias articulares que permitan comprobar lo anteriormente mencionado. Para esto se ha optado por el método de "Generación de Trayectorias Articulares Mediante Ondas Sinusoidales Acopladas" [33], debido a que la marcha de un robot bípedo es similar a la humana y algunas características como la marcha periódica y simétrica son la base del método anteriormente mencionado [33]. Este método, parte de que al realizar una caminata a velocidad constante, los movimientos de cada articulación durante un paso serán ciclos repetitivos y los movimientos de las piernas son simétricos con referencia al plano sagital cuando se realiza una marcha lineal [33], el movimiento de una pierna es simétrico al de la contraria, pero con un cierto retardo exactamente la mitad del tiempo que dura un paso. Para esto se incluye un tiempo de retardo entre los movimientos igual a la mitad del periodo del ciclo, con el objetivo de asegurar posiciones opuestas. Debido a lo anteriormente mencionado los movimientos de cada articulación pueden ser representados mediante una función cíclica con periodo constante. Por tal motivo se utiliza la señal sinusoidal de la siguiente forma:

$$y(t) = A * \sin(w * t + B) + D \tag{48}$$

Donde el tiempo se representa por *t*, la posición que toma la articulación en cada instante de tiempo por *y(t)*, *A*, es la amplitud de oscilación, *w* la frecuencia angular de oscilación, *B* la fase inicial de la señal y *D* es el punto fijo alrededor del cual oscila la función [35], [38].

Una manera de modelizar un paso es almacenar la secuencia de posiciones intermedias de todas las articulaciones que se emplean en la realización de un paso [33], [35], por esto se requieren 12 funciones parametrizadas en frecuencia, amplitud, fase y desplazamiento, una por cada articulación del robot propuesto, lo anterior lleva a que, por cada señal se tengan 4 parámetros que no se conocen y finalmente vendrían ser 48 incógnitas en total, cabe resaltar que las articulaciones 1 y 7 de las piernas, no poseen movimientos variantes en el tiempo durante el ciclo de marcha, en vista de que controlan la rotación de la cadera, sin embargo, mantienen un valor constante de posición, debido a que solo se probaran

caminatas en trayectorias rectas, es por esto que serían 40 incógnitas ya que es necesario encontrar 4 parámetros de 10 articulaciones.

Partiendo de que la caminata es un ciclo repetitivo, un paso debe empezar y terminar al mismo tiempo para todos los actuadores, para ello es necesario definir un valor común de frecuencia, permitiendo establecer la velocidad de caminata, sin embargo, esta debe tener relación con los rangos de velocidad de los actuadores para que estos puedan seguir la referencia. Lo que conlleva a que solo es necesario tener una frecuencia para todas las articulaciones, reduciendo así de 4 a 3 parámetros por cada actuador, quedando con 31 incógnitas.

Otro aspecto a tener en cuenta en las caminatas rectas, es el movimiento simétrico en las piernas, es por esto que, solo se requiere conocer la posición de las articulaciones de una de las piernas del robot, a partir de los movimientos de esta se pueden establecer el comportamiento de la otra pierna incluyendo un desfase fijo en las funciones de movimiento. Gracias a esto se reducen a 16 incógnitas, ya que vendrían a ser 3 parámetros de 5 articulaciones y una frecuencia común para la velocidad.

Cabe resaltar que para generar una marcha estable del robot bípedo el CoM debe mantenerse dentro del polígono de soporte [14], [17]. Debido a esto el parámetro *D* (que es el punto fijo alrededor del cual oscila la función sinusoidal), tiene que estar alrededor del punto donde el robot se encuentra en posición de genuflexión y su CoM en el centro del polígono de soporte, de esta manera se encuentra el parámetro *D* en las diferentes articulaciones del robot, llevando a tener 11 incógnitas.

Posteriormente el valor de desfase entre las articulaciones semejantes del pie contrario es de 180°, dado a que el movimiento de cada pie es opuesto, sin embargo, al estar opuestos los motores de dichas articulaciones semejantes no existirá tal desfase, encontrando de esta manera el parámetro *B* para todas las articulaciones. Finalmente, las cinco amplitudes sobrantes son calculas a partir de pruebas experimentales.

4.1.1 Posición de Genuflexión del robot bípedo

Para encontrar la posición de genuflexión, fue necesario mover de manera experimental los motores de las articulaciones 3, 4, 5, 9, 10 y 11 para llevar al robot a una posición en la cual se encuentre de manera erguido y que su COP se mantenga en el centro del polígono de soporte; para ello se trabajó con las articulaciones anteriormente mencionadas, ya que presentan cambios de posición angular al momento de aplicarles una trayectoria de marcha unidireccional, en este caso mediante ondas sinusoidales acopladas. Además, para comprobar que la posición del COP del robot durante la genuflexión se encontraba en el centro del polígono de soporte, fue necesario el uso de los sensores de presión ubicados en los pies del robot. Por otro lado, con las lecturas obtenidas de los sensores de presión se calculó el ZMP del robot. Como evidencia del proceso anteriormente descrito, la Figura 4.1 presenta el robot en el estado de genuflexión y la Figura 4.2 presenta la gráfica que

evidencia la ubicación del COP y el ZMP en dicha posición. Cabe resaltar que los valores de las posiciones encontrados en dicho experimento se encuentran en la tabla 4.

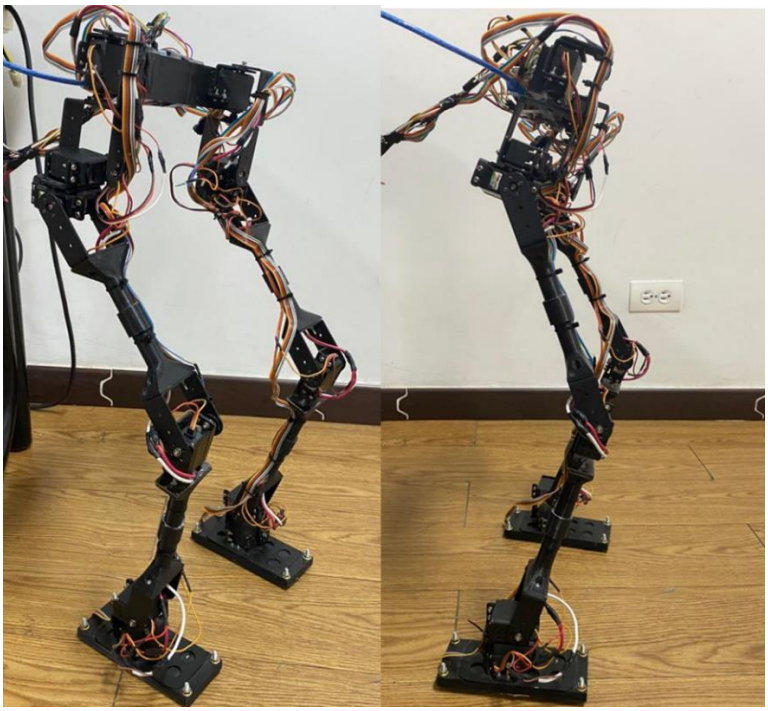


Figura 30. Robot bípedo en la posición de Genuflexión.

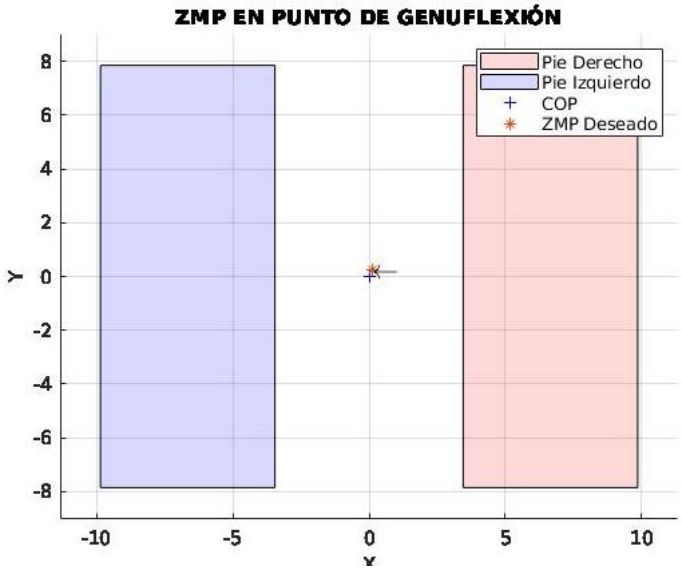


Figura 31. Posición del COP cuando el robot se encuentra en posición de genuflexión

4. 1.2 Parámetros de las Trayectorias Articulares Mediante Ondas Sinusoidales Acopladas

Finalmente, después de obtener la posición de cada articulación del robot en la cual se encuentra en genuflexión, se determina experimentalmente y con ayuda del simulador los parámetros detallados en la Tabla 4.

Articulación (j)	**Desplazamiento(D) (rad)**	**Amplitud (A)**	**Frecuencia(W) (Hz)**	**Fase (B)**
1	0	0	0,05	0
2	0	0	0,05	0
3	$40*\frac{\pi}{180}$	10	0,05	0
4	$-30*\frac{\pi}{180}$	5	0,05	0
5	$-10*\frac{\pi}{180}$	5	0,05	0
6	0	0	0,05	0
7	0	0	0,05	0
8	0	0	0,05	0
9	$-40*\frac{\pi}{180}$	10	0,05	0
10	$30*\frac{\pi}{180}$	5	0,05	0
11	$10*\frac{\pi}{180}$	5	0,05	0
12	0	0	0,05	0

Tabla 4: Parámetros de Trayectorias de Marcha Articulares mediante Ondas Sinusoidales Acopladas para el robot propuesto.

Las trayectorias articulares completas obtenidas en la simulación para el generador de patrones de marcha utilizando el método de trayectorias sinusoidales acopladas, se muestran en la Figura 4.3.

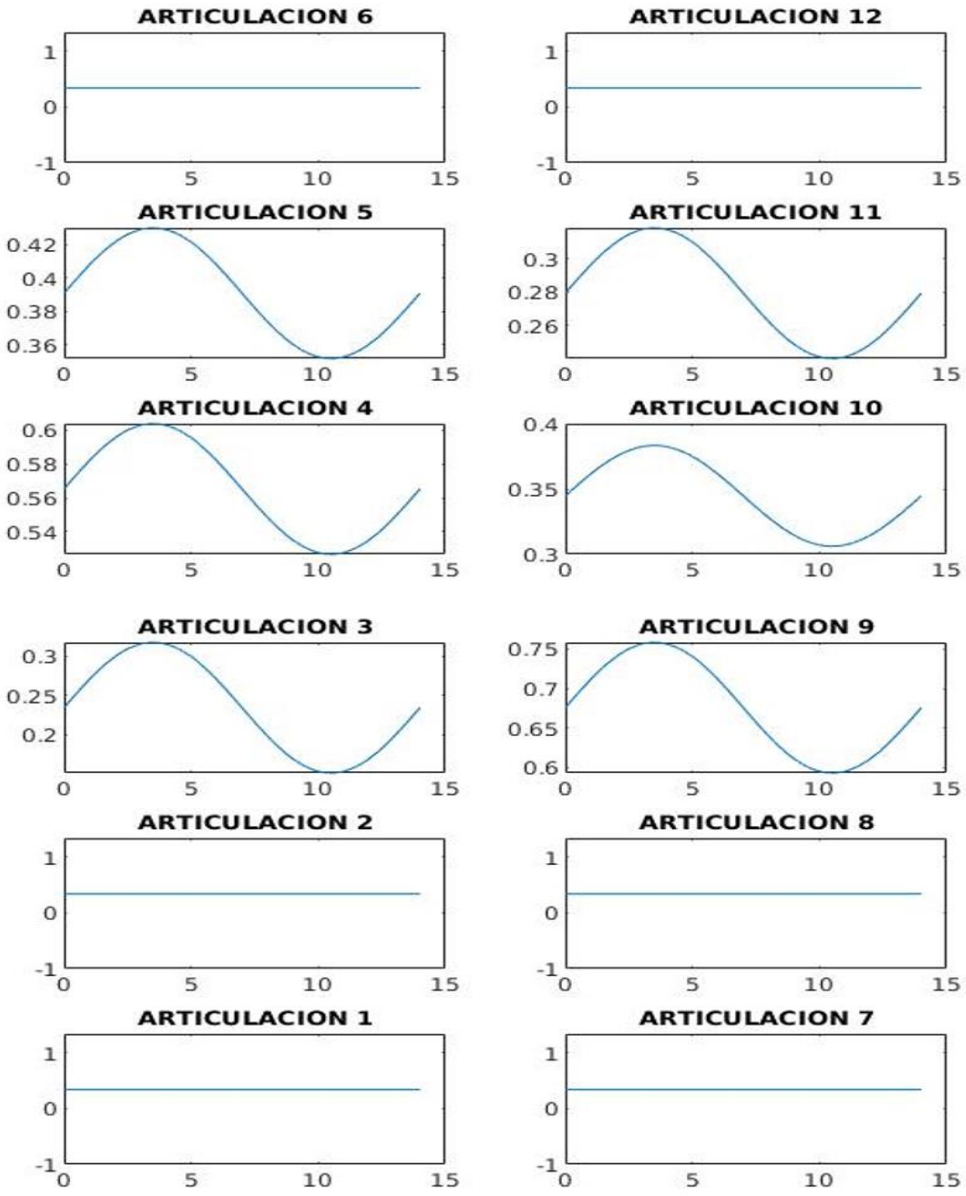


Figura 32.Trayectorias articulares de marcha mediante el método generación de trayectorias sinusoidales

Las articulaciones 1 y 7 no tendrán rotación, pero si se mantendrán un valor constante de posición, las articulaciones 2 y 8 también se mantendrán en un valor constante de posición, mientras las articulaciones 3 y 9, 4 y 10, 5 y 11 realizarán una trayectoria articular, la cual permite realizar los movimientos de marcha en el robot bípedo como se observa en la figura 4.3, finalmente, las articulaciones 6 y 12 mantendrán un valor constante.

4.1.3 Pruebas experimentales.

Para comprobar el funcionamiento del prototipo se plantearon 2 pruebas experimentales, la primera consiste en comparar el seguimiento de las trayectorias en el robot real con una simulación cuando el robot se encuentra suspendido y la otra prueba consiste en encontrar el error entre el COP ideal (0,0) y el COP mientras el robot propuesto se balancea con las trayectorias articulares.

4.1.3.1. Trayectorias de marcha articulares suspendiendo el robot propuesto.

Se planteó suspender el robot bípedo de una estructura colgante y enviar el patrón de marcha preestablecido y comparar los movimientos con el robot simulado en la herramienta *Simulink Multibody*, con el propósito de contrastar de manera visual la capacidad de seguimiento de trayectorias articulares.

Se establecen las siguientes consideraciones a tener en cuenta para este experimento:

- El robot se encuentra suspendido y su patrón de marcha es en línea recta.
- El robot simulado en *Simulink Multibody* presenta las mismas dimensiones, numero de articulaciones y trayectorias de marcha que el robot propuesto.
- Las trayectorias de marcha las cuales tendrán los robots se encuentran en la tabla 4.
- Se realizarán 4 pasos de trayectorias.

Como resultado de este experimento se realizo un video comparativo [62] en el cual se muestra el movimiento del robot real en comparación con el robot simulado, desde una vista frontal y lateral, sin embargo para datar los movimientos de manera formal en este documento de monografía, se capturaron los fotogramas en los cuales el robot se encuentra en la posición inicial, después cuando avanza con el pie izquierdo hacia delante mientras el derecho va hacia atrás, luego cuando retoma la posición inicial y finalmente cuando el pie derecho avanza mientras el izquierdo se encuentra atrás, en la Figura 4.4 se puede evidenciar dichos movimientos desde una vista frontal y la Figura 4.5 desde una vista lateral.

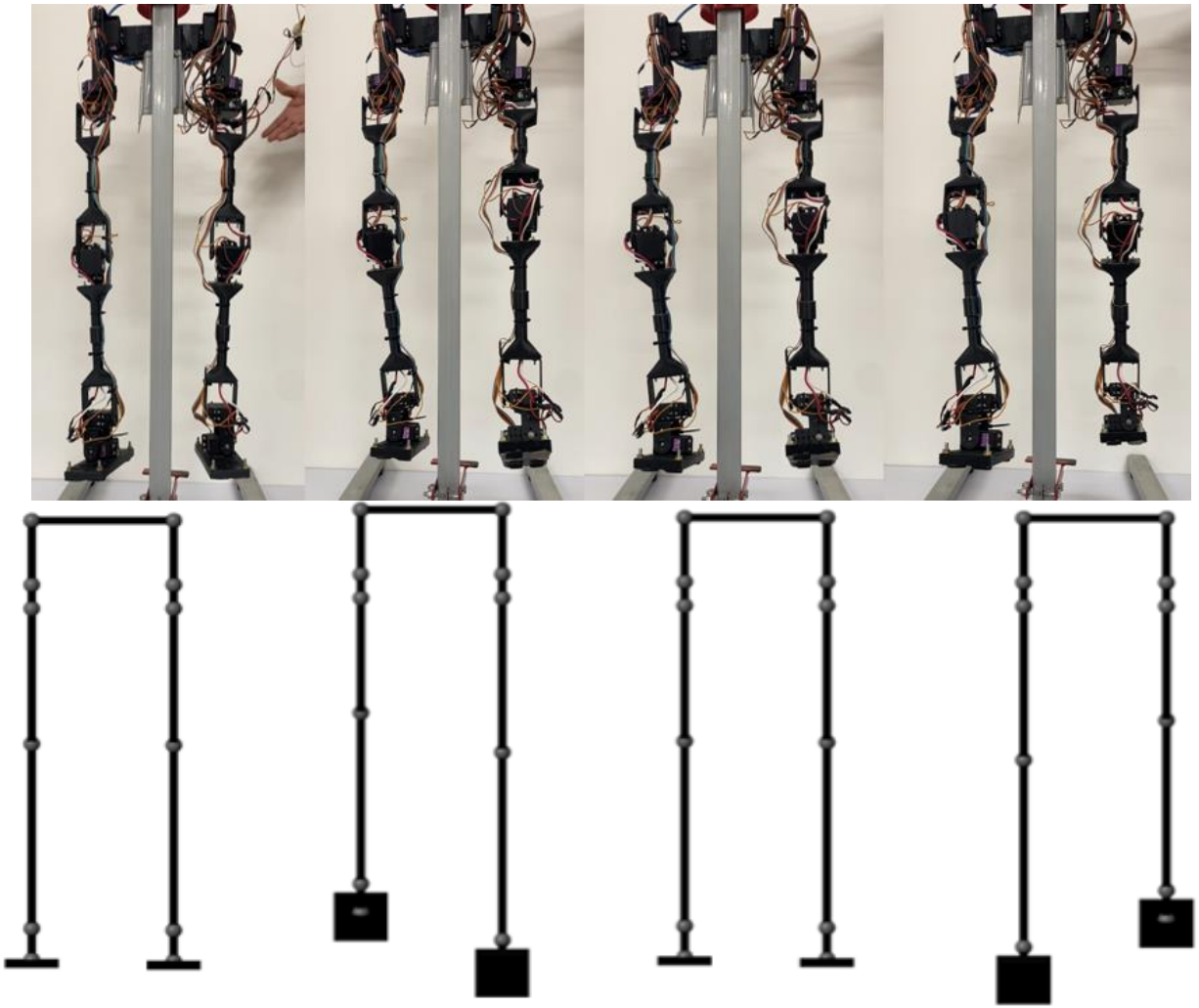


Figura 33. Trayectorias de marcha desde una vista frontal para el robot propuesto y el robot simulado.

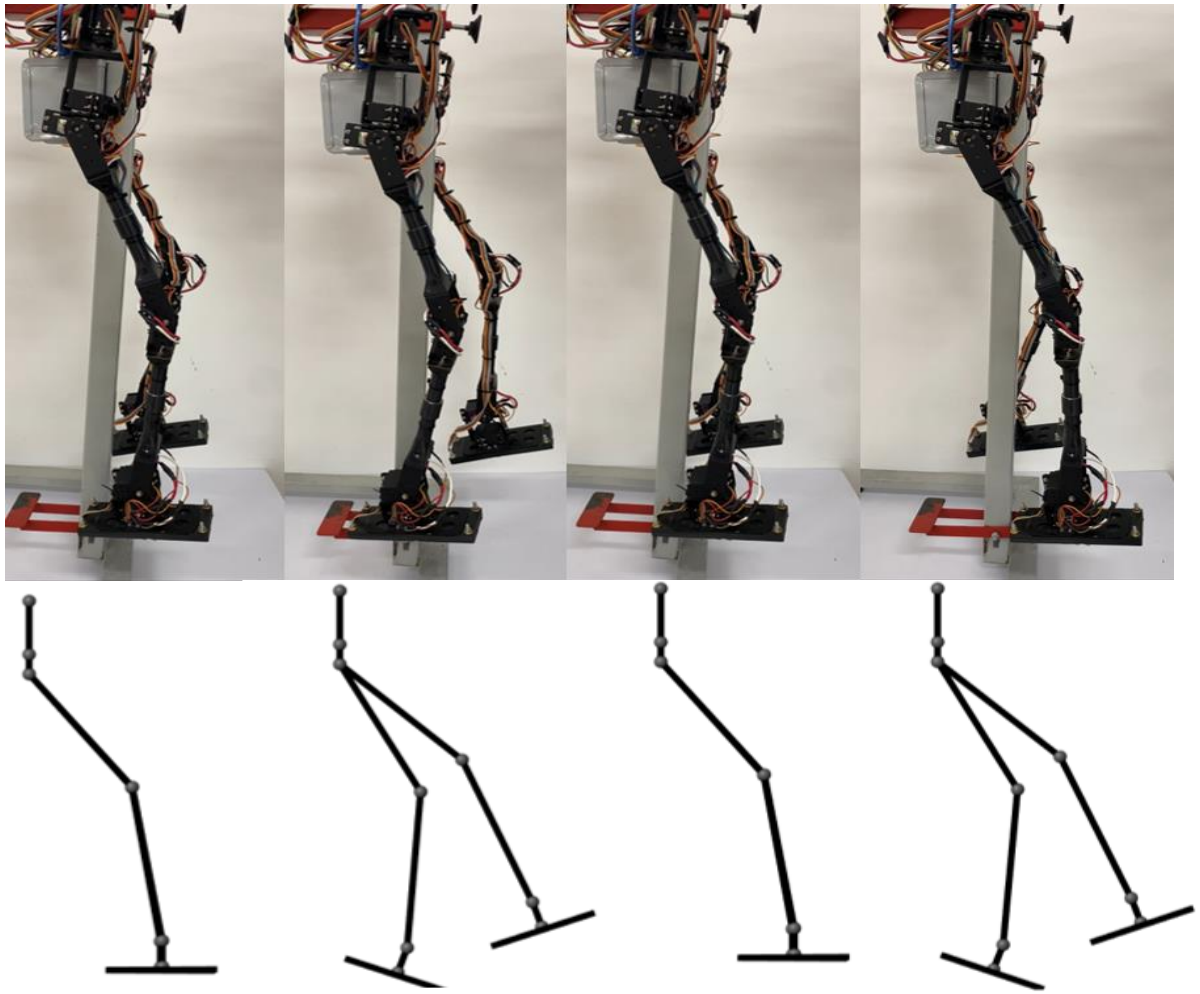


Figura 34. Trayectorias de marcha desde una vista lateral para el robot propuesto y el robot simulado.

De este experimento se pudo observar, que el robot bípedo sigue las posiciones articulares predefinidas por el generador de patrones sinusoidal, sin embargo al comparar estos movimientos con los simulados, se puede notar que algunos comportamientos no son tan pronunciados como en la simulación, esto es debido al peso del robot y el torque de los motores empleados en las articulaciones, pues en la simulación no se tiene en cuenta la gravedad y la fricción de los motores, otro aspecto a tener en cuenta es la velocidad del movimiento del robot, la cual es más rápida en la experimentación comparada con el robot simulado, esto se debe a que cuando ejecuta el movimiento de avanzar un pie y al momento de regresarlo lo hace más rápido por el peso, la gravedad y el torque de los motores, que entre un paso y el siguiente contribuyen a soportar el pie rápidamente hasta el suelo, a diferencia de cuando es necesario levantar el pie para iniciar la zancada. Por otro lado, a pesar de no trabajar con un control central para el robot propuesto, se cuenta con el control articular dado por el sistema de engranajes con el que cuenta cada servomotor, los cuales permiten llevar a una posición angular previamente establecida a cada articulación, de esta manera se puede afirmar que el robot sigue las posiciones articulares predefinidas.

4.1.3.2 Seguimiento del centro de presión mientras el robot propuesto se balancea con las trayectorias articulares.

Para esta prueba experimental, se planteó enviar las trayectorias articulares obtenidas al robot propuesto y observar cómo se perturba la posición del centro de presión (COP, *center of presion*) en el centro del polígono de soporte, cuando se encuentra parado en una superficie plana. Se realizó dicho experimento 10 veces, para 2 pasos de trayectorias, de cada prueba se obtuvieron 202 coordenadas que indican la posición del COP durante la ejecución de la trayectoria de los 2 pasos como se muestra en la Figura 4.6. Estas posiciones se obtuvieron mediante los sensores de presión del robot y el cálculo de las posiciones las coordenadas ZMP_x y ZMP_y a través de Matlab.

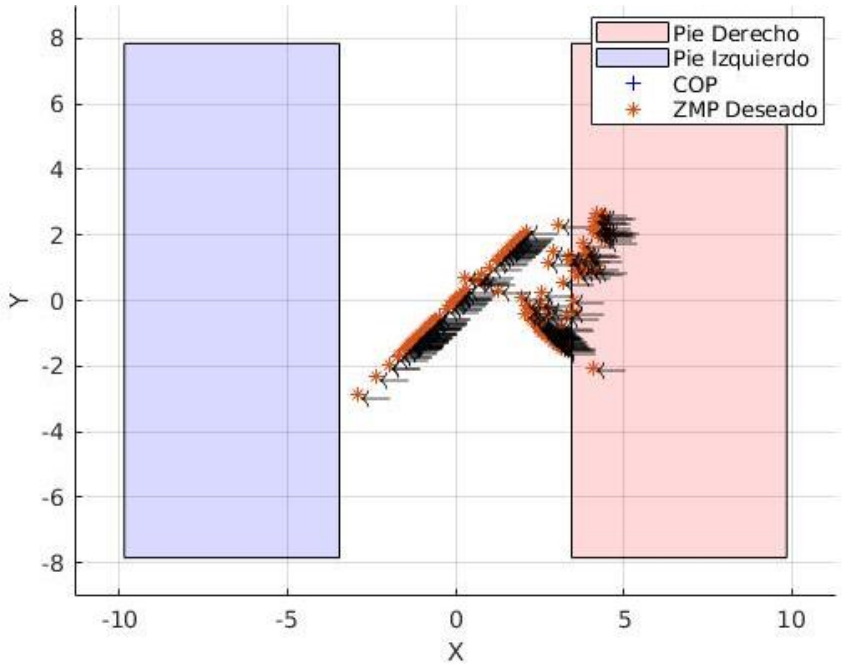


Figura 35. Recorrido de las diferentes posiciones del COP mientras realiza dos pasos de trayectoria

Después de obtener las coordenadas se obtuvo la distancia de cada una de estas posiciones con respecto al centro del plano cartesiano mediante la Ecuación (4.5) que representa la distancia entre dos puntos.

$$d(A,B) = \sqrt{(x2 - x1)^2 + (y2 - y1)^2} \tag{49}$$

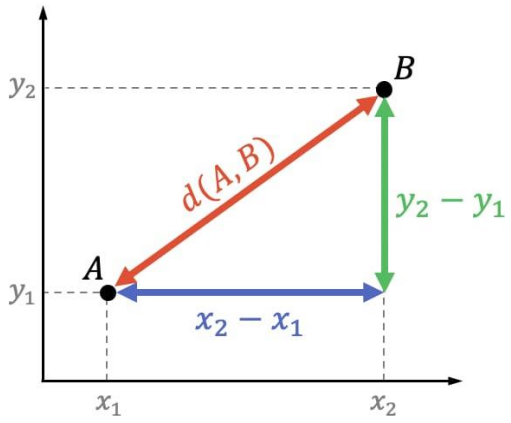
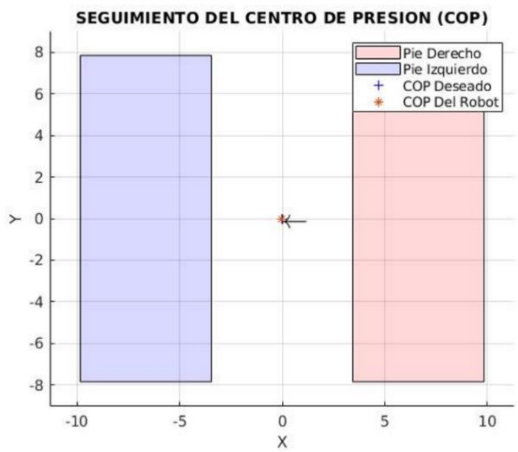


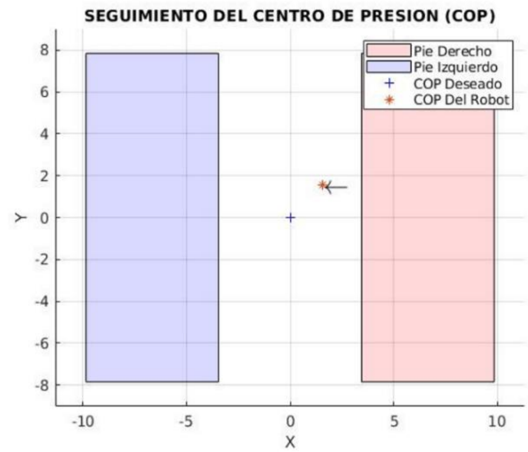
Figura 36. Distancia entre dos puntos en un plano cartesiano [63]

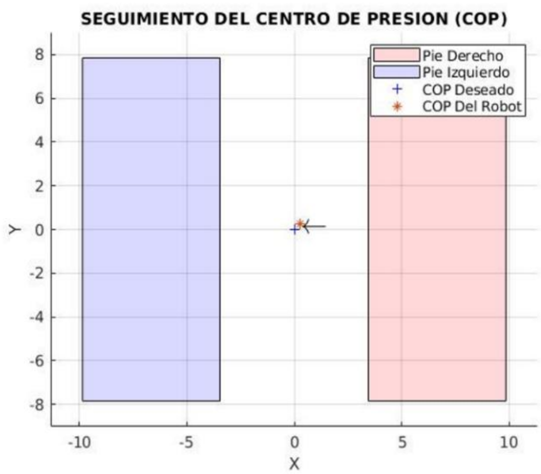
Posteriormente, al obtener dicha distancia se comparan estas longitudes con el COP ideal que para este caso será la posición (0,0); debido a que al realizar una caminata el COP debe encontrarse en el centro del polígono de soporte del robot bípedo, esta comparación se hace en Matlab mediante el cálculo del error cuadrático medio para cada una de las posiciones, obteniendo un porcentaje de error por cada prueba de experimentación, permitiendo analizar de manera matemática si el robot ejecuta las trayectorias e intenta mantener el COP en el centro del polígono de soporte del robot a pesar de que no existe un control central de equilibrio. En dichas pruebas se encontraron los siguientes resultados:

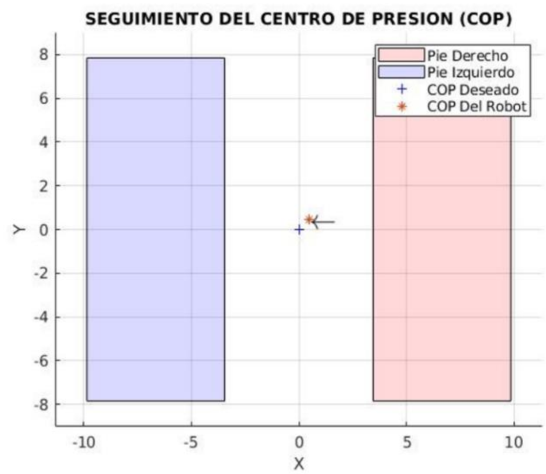
Numero de prueba de experimentación	**Error cuadrático medio (%)**
1	15.84
2	8.56
3	17.13
4	14.55
5	7.52
8	14.66
7	7.31
8	8.28
9	35.8
10	7.78

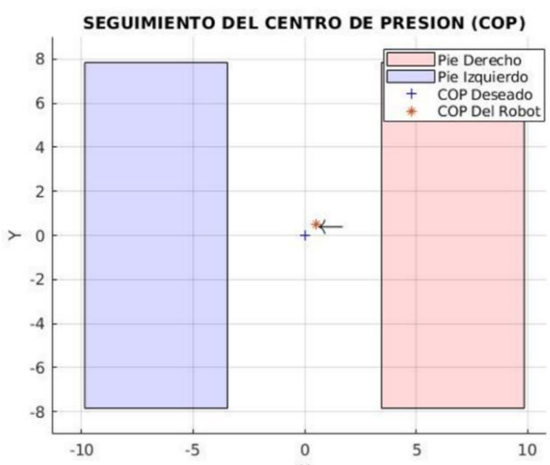
Tabla 5: Resultados del error medio cuadrático respecto al COP del robot propuesto y el COP deseado

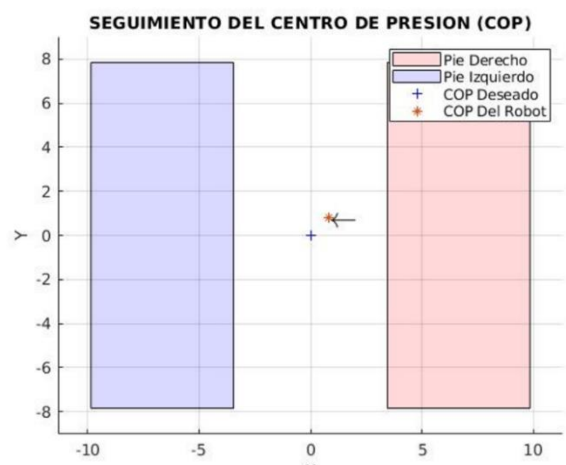

SEGUIMIENTO DEL CENTRO DE PRESION (COP)
Pie Derecho
Pie Izquierdo
COP Deseado
COP Del Robot
X
Y


SEGUIMIENTO DEL CENTRO DE PRESION (COP)
Pie Derecho
Pie Izquierdo
COP Deseado
COP Del Robot
X
Y


SEGUIMIENTO DEL CENTRO DE PRESION (COP)
Pie Derecho
Pie Izquierdo
COP Deseado
COP Del Robot
X
Y


SEGUIMIENTO DEL CENTRO DE PRESION (COP)
Pie Derecho
Pie Izquierdo
COP Deseado
COP Del Robot
X
Y


SEGUIMIENTO DEL CENTRO DE PRESION (COP)
Pie Derecho
Pie Izquierdo
COP Deseado
COP Del Robot
X
Y


SEGUIMIENTO DEL CENTRO DE PRESION (COP)
Pie Derecho
Pie Izquierdo
COP Deseado
COP Del Robot
X
Y

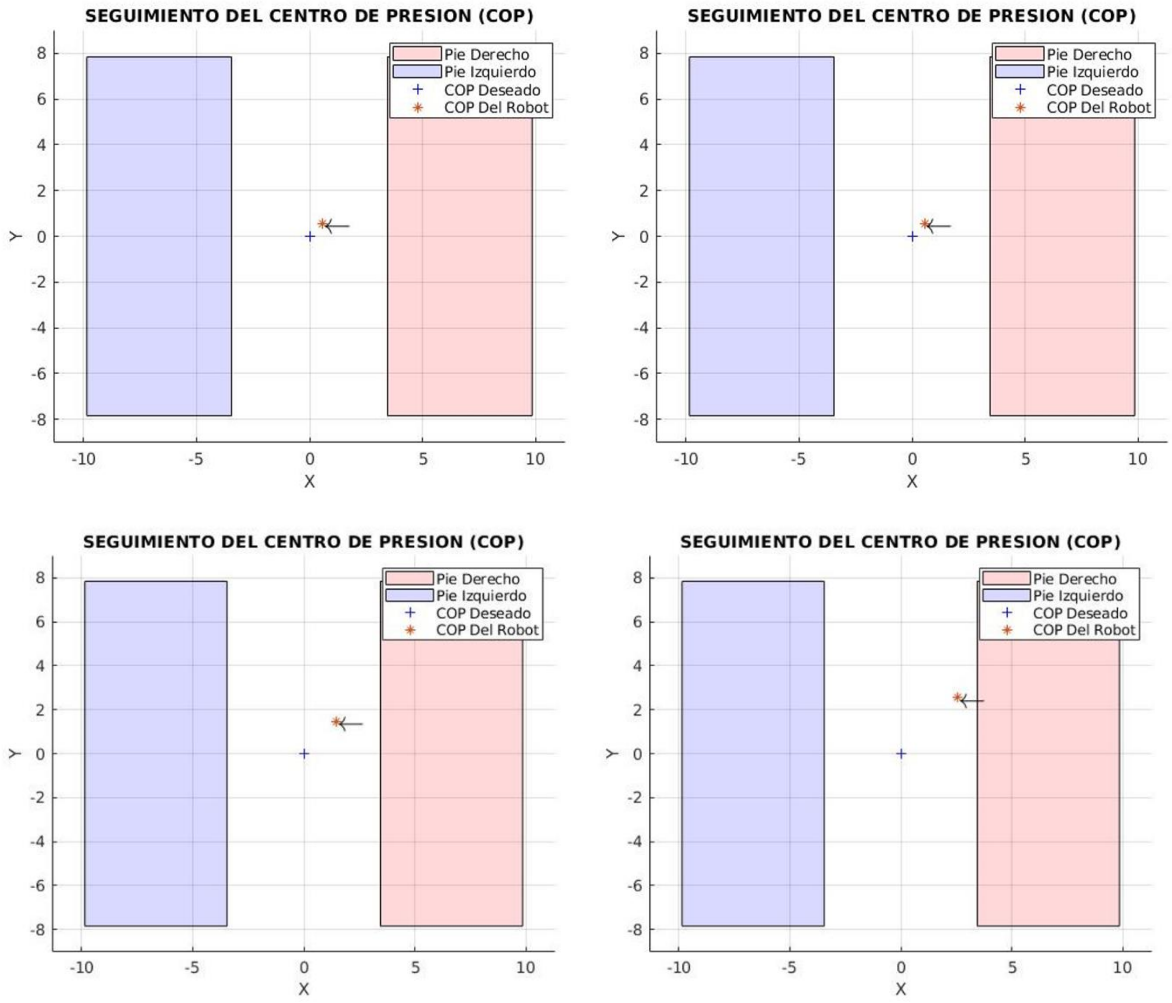


Figura 37. COP del robot bípedo durante cada prueba de experimentación.

De las pruebas podemos observar que 35.84% es el mayor porcentaje de error de la posición del COP del robot real con respecto a la posición de COP ideal (0,0) en el plano cartesiano y 7,785 es el mejor porcentaje de error, cabe resaltar que el mayor error de cuadrático posible, además, la media de estos errores es 13,74%. A partir de los valores encontrados y como se evidencia visualmente en la Figura 4.8. Podemos inferir que el COP del robot propuesto se encuentra cerca al COP ideal, además que el COP del robot se encuentra dentro del polígono de soporte, lo cual implica que el robot se mantendrá estable y podrá ejecutar la trayectoria de marcha articular.

5. Conclusiones y trabajo futuros

5.1 Conclusiones

- Mediante la herramienta Matlab es posible realizar el cálculo de algunos modelos matemáticos del robot, por ejemplo, en este proyecto se realizó el cálculo del modelo geométrico directo, sin embargo, al momento de realizar el modelo inverso es muy complejo despejar las incógnitas debido a que es un modelo de 12x12, debido a esto se decidió usar la herramienta Simulink Multibody, esta herramienta posee un bloque denominado Transform Sensor el cual nos permitió saber la posición de los pies del robot bípedo.
- La herramienta Simulink Multibody da la facultad de construir el modelo del robot con sus respectivas articulaciones, materiales y dimensiones permitiendo obtener los modelos geométricos del robot propuesto de manera simuladas sin necesidad de realizar cálculos matemáticos externos, cabe resaltar que para este proyecto no se importó el modelo CAD en esta herramienta, sin embargo, esto hubiera permitido obtener resultados más aproximados debido a que al usarlo se pueden abstraer más características del robot propuesto.
- El sistema operativo ROS permitió conectar diferentes nodos, lenguajes, y programas de forma remota, siendo muy útil por su versatilidad a la hora de realizar experimentos en robots bípedos, gracias a esto en este proyecto trabajamos en diferentes programas como ROS, Python, Arduino y Matlab, dando paso a ser una herramienta útil para expandir las características y funcionales del robot en futuros trabajos.
- El sistema operativo ROS usado en el sistema operativo Linux presenta más documentación y algunas características que permiten a los usuarios mayor facilidad para usar dicho framework
- Otro aspecto importante es él envió información a través del puerto serial en Phyton y Arduino, para esto se debe tener en cuenta el tiempo de envío de los mensajes debido a que en ocasiones se pierden los paquetes o se envían hasta cierto punto.
- El nodo rosbridge es de gran utilidad, debido a que permite generar un canal para comunicarse mediante wifi y de esta manera interconectar diferentes dispositivos, programas y lenguajes de programación para intercambiar información entre ROS y cualquier otra aplicación o programas capaces de analizar JSON.
- El robot propuesto sigue las trayectorias de marcha articulares realizadas mediante el método de Generación de Trayectorias Articulares Mediante Ondas Sinusoidales Acopladas, esto se pudo evidenciar cuando se suspendió el robot y se comparó con la simulación de Matlab y en el porcentaje del error medio cuadrático respecto al COP del robot propuesto y el COP deseado, debido a que es relativamente pequeño.
- Los motores que se usaron para el robot, no son los más funcionales, debido a que se necesitan motores con un mayor torque a causa del tamaño y peso del robot, por eso

para trabajos futuros se sugiere cambiar dichos motores con el fin de poder realizar los movimientos deseados sin ningún inconveniente.

- Un aspecto para mejorar en trabajos futuros es usar una tarjeta que posea las características necesarias para el funcionamiento del robot como lo son pines analógicos para los sensores, pines PWM para los motores y comunicación inalámbrica entre otras, esto permitirá reducir los tiempos de envió de información, reducir conexiones entre las tarjetas, minimizar el uso de diferentes programas entre otras.

5.2 Trabajos Futuros

Algunos trabajos futuros que se pueden realizar a partir de este proyecto son los siguientes:

- Realizar un algoritmo de control central para mantener el equilibrio del robot bípedo cuando se encuentre realizando trayectorias de marcha en línea recta.
- Realizar estudios de estabilidad para la marcha bípeda en la cual se pruebe un algoritmo que permita mantener el COP en el centro del robot cuando se encuentren perturbaciones en la marcha o en la superficie sobre la cual se encuentra el robot.
- Realizar el control central para trayectorias de marcha omnidireccional, partiendo desde encontrar dichas trayectorias hasta realizar las diferentes pruebas experimentales.

6. Referencia bibliográfica

[1] F. Martinez, E. Gómez y D. Zárate, «Concepto de robot humanoide antropométrico para investigación en control,» *Tecnura,* vol. 19, pp. 55-65.

[2] J. Linert y P. Kopacek, «Humanoid robots Robotainment,» *IFAC-PapersOnline,* vol. 51, nº 20, pp. 220-225, 2018.

[3] I. Zannatha, J. Cisneros, R. Garcia, L. Lavin y J. Yáñez, «Congreso internacion en innovación y desarrollo tecnológico (CIINDET),» de *Evasion de obstáculos basada en visión artificial y en sensores infrarrojos para un humanoide,* Cuernavaca, Mexico, 2008.

[4] G. Kalyani, *A Robot Operating System (ROS) based humanoid robot control,* Londres: Middlesex University, 2017.

[5] S. Gozález, J. Ramirez y E. Avella, «Técnicas de control para el balance de un robot bípedo: un estado del arte,» *Tecnura,* vol. 19, nº 43, pp. 139-162, 2015.

[6] J. Tacué, *Análisis comparativo del consumo de energía eléctrica en trayectorias de marcha offline para el robot bípedo bioloid,* Popayán, Colombia, 2018.

[7] D. Bravo, *Generación de trayectorias para un robot bípedo basadas en captura de movimiento humano,* Popayán, Colombia, 2016.

[8] J. Tacué y C. Rengifo, «Acerca del modelado, control y generación de marcha en robots bípedos,» *Ingenium Revista De La Facultad De Ingeniería,* vol. 19, nº 37, 2018.

[9] W. Medina, *Diseño e implementación de un sistema inalámbrico para monitorizar el consumo de energía eléctrica del robot bípedo bioloid premium kit,* Popayán, Colombia, 2018.

[10] M. Albero, F. Blanes, G. Benet, J. Simó y P. Perez, «YABIRO: PROTOTIPO DE ROBOT BÍPEDO AUTÓNOMO,,» de *XIV Jornadas de Automática,* Leon, España,, 2003. .

[11] E. Estrada, H. Becerra y C. Segura, «Optimization of Walking Control of a Biped Robot using Differential Evolution,» de *IEEE Latin American Conference on Computational Intelligence (LA-CCI), pp. 1-7,* Guadalajara, Mexico, 2018.

[12] J. Reher, E. Cousineau, A. Hereid, C. Hubicki y A. Ames, «Realizing dynamic and efficient bipedal locomotion on the humanoid robot DURUS,» de *2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 1794-1801*, 2016.

[13] F. Merino, E. Rocha y S. Hernández, «Implementación de un Patrón de Marcha Circular en un Robot Bıpedo de 12 GDL Internos,» de *Memorias del Cong. Anual AMCA*, 2016 .

[14] R. Ferreira, N. Shafii, N. Lau, L. Reis y A. Abdolmaleki, «Diagonal walk reference generator based on Fourier approximation of ZMP trajectory,» de *Lisbon*, 2013, 2013 13th International Conference on Autonomous Robot Systems .

[15] P. Shengjun, S. Haitao, L. Guang y M. Hongxu, «Walking gait planning of humanoid soccer robot based on the desired ZMP trajectories,» de *2010 The 2nd International Conference on Industrial Mechatronics and Automation*, Wuhan, 2010.

[16] W. Wei-guo y H. Yue-yang, «Research on rapid walking of biped robot based on parametric surface table cart model,» de *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Guilin, 2009 .

[17] J. Pantoja, *Diseño concurrente en la optimización de un mecanismo de locomoción bípedo,* Mexico D.F., 2019.

[18] S. Čelikovský y V. Lynnyk, «On the chaotic behavior of the hybrid inverted pendulum and its relation to the lateral dynamics of the walking robots,» *IFAC-PapersOnLine*, vol. 51, nº 33, pp. 15-21, 2018.

[19] J. Castano , A. Hernandez , Z. Li , C. Zhou y N. Tsaga, «Implementation of Robust EPSAC on dynamic walking of COMAN Humanoid,» *IFAC Proceedings Volumes*, vol. 47, nº 33, pp. 8384-8390, 2014.

[20] S. Dalibard, A. El Khoury, F. Lamiraux y A. Nakhaei, «Dynamic walking and whole-body motion planning for humanoid robots: an integrated approach,» *The International Journal of Robotics Research*, vol. 9, nº 10, pp. 1089-1103, 2013.

[21] J. Rodriguez, *Modelado matemático de un robot bípedo con equilibrio dinámico,* Bogotá, Colombia, 2017.

[22] M. Dominguez, *Diseño de un robot humanoide de bajo coste mediante robot operating system,* 2019.

[23] R. Delgado, B. You y B. Choi, «Real-time control architecture based on Xenomai using ROS packages for a service robot,» *Journal of Systems and Software,*, vol. 151, pp. 8-19, 2019.

[24] V. Tlach , I. Kuric, D. Kumičáková y A. Rengevič, «Possibilities of a Robotic End of Arm Tooling Control within the Software Platform ROS,» *Procedia Engineering*, vol. 192, pp. 857-880, 2017.

[25] H. Feng, C. Wong, C. Liu y S. Xiao, «ROS-Based Humanoid Robot Pose Control System Design,» de *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Miyazaki, Japan, 2018.

[26] J. Borrella, *DESARROLLO DE UN CONTROL VERSÁTIL DE TRAYECTORIAS Y MODOS DE MARCHA PARA UN ROBOT HEXÁPODO*, Madrid, España, 2017.

[27] A. Araújo, D. Portugal, M. Couceiro, J. Sales y R. Rocha, «Desarrollo de un robot móvil compacto integrado en el middleware ROS,» *Revista Iberoamericana de Automática e Informática Industrial RIAI*, vol. 11, nº 2, pp. 315-326, 2014.

[28] M. Galli, *Implementación de localización geométrica para robots móviles bajo ROS*, Madrid,, 2016.

[29] G. Spyros , «Introduction to Mobile Robot Control,» de *Mobile Robots: General Concepts*, elsevier, 2014, pp. 1-29.

[30] E. Yoshida, «Robots que parecen humanos,» *metode*, 2018.

[31] F. H. Martínez Sarmiento, E. J. Gómez y D. A. Zárate Días, «Concepto de robot humanoide antropométrico para investigación en control,» *Tecnura*, p. 19, 2015.

[32] K. Gregor, Z. Andrej, B. Sašo y Š. Igor, «Chapter 2 - Motion Modeling for Mobile Robots,» de *Wheeled Mobile Robotics*, Butterworth-Heinemann, 13-59, p. 2017.

[33] F. Rivas, J. Canas y J. González, «Aprendizaje automático de modos de caminar para un robot humanoide.,» *Proceedings of Robot*, pp. 120-127, 2011.

[34] O. A. Vivas Albán, Diseño y control de robots industriales: teoría y práctica, Buenos Aires: Elaleph.com, 2010.

[35] D. Hein, *Evolution of Biped Walking Using Physical Simulation.*, 2007.

[36] S. G., «2 - Mobile Robot Kinematics,» de *Introduction to Mobile Robot Control*, Elsevier, 2014, pp. 31-67.

[37] N. Dragomir, K. Atsushi y T. Teppei, «Chapter 2 - Kinematics,» de *Humanoid Robots*, Butterworth-Heinemann, 2019, pp. 15-82.

[38] J. Tacué y H. Naranjo, *Simulacion del cilo de marcha del robot bipedo bioloid en un entorno virtual 3D,* Popayán, Colombia, 2014.

[39] k. Gregor, z. Andrej, B. Sašo y Š. Igor , «Chapter 2 - Motion Modeling for Mobile Robots,,» de *Wheeled Mobile Robotics*, Butterworth-Heinemann, 2017, pp. 13-59.

[40] N. Dragomir, K. Atsushi y T. Teppei, «Chapter 5 - Balance Control,» de *Humanoid Robots*, Butterworth-Heinemann, 2019, pp. 203-302.

[41] N. Dragomir, K. Atsushi y T. Teppei, «Chapter 4 - Dynamics,» de *Humanoid Robots*, Butterworth-Heinemann, 2019, pp. 125-202.

[42] S. Kajita, «Biped walking pattern generation by using preview control of zero-moment point,» de *2003 IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, 2003.

[43] W. Suleiman, F. Kanehiro, K. Miura y E. Yoshida, «Enhancing Zero Moment Point-Based Control Model: System Identification Approach,» *Advanced Robotics,* vol. 25, pp. 427-446, 2011.

[44] R. Guadarrama, E. Dean, F. Bergner y G. Cheng, «Plantar Tactile Feedback for Biped Balance,» *International Journal of Humanoid Robotics,* vol. 17, nº 1, 2020.

[45] A. Palomino y G. Echeverry, *Medición y modelado bilineal del Zero Moment Point (ZMP) mediante la técnica de subespacios recursiva,* Puebla, 2019.

[46] J. Franco, A. De Lucio, K. Camarillo, G. Perez y J. Rivera, «Proceedings of the ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference,» de *Liquid State Machine to Generate the Movement Profiles for the Gait Cycle of a 6 DOF Bipedal Robot in a Sagittal Plane*, Quebec, Canada, 2018.

[47] N. Shafii , A. Abdolmaleki y R. Ferreira, «Omnidirectional Walking and Active Balance for Soccer Humanoid Robot.,» *Lectures notes in computer science,* vol. 8154, pp. 283-294, 2013.

[48] S. Piperakis, E. Orfanoudakis y M. Lagoudakis, «Predictive control for dynamic locomotion of real humanoid robots,» de *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, 2014.

[49] L. Lanari y S. Hutchinson, «Optimal double support zero moment point trajectories for bipedal locomotion,» de *016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, 2016.

[50] G. Sun, H. Wang y Z. Lu, «A Novel Biped Pattern Generator Based on Extended ZMP and Extended Cart-Table Model,» *International Journal of Advanced Robotic Systems*, 2015.

[51] N. Shafii, A. Abdolmaleki, R. Ferreira, N. Lau y L. Reis, «Omnidirectional Walking and Active Balance for Soccer Humanoid Robot,» *Lecture Notes in Computer Science*, vol. 8154, pp. 232-294, 2013.

[52] H. Weiwei, C. Chee-Meng, Z. Yu y H. Geok-Soon, «Pattern generation for bipedal walking on slopes and stairs,» de *Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots*, Daejeon, 2008.

[53] S.-M. Hong, «Método de conversión de trayectoria CoM / ZMP en tiempo real para robots humanoides que reflejan características estructurales,» *revista de la Sociedad Coreana de Navegación*, vol. 21, nº 1, pp. 132-137, 2017.

[54] S. Hernandez-Mendez, C. Maldonado-Mendez, A. Marin-Hernandez, H. Rios-Figueroa, H. Vazquez-Leal y E. Palacios-Hernandez, «Design and implementation of a robotic arm using ROS and MoveIt!,» de *2017 IEEE International Autumn Meeting on Power, Electronics and computing (ROPEC)*, Ixtapa, 2017.

[55] E. Maciel, R. Henriques y F. Lages, «Control of a Biped Robot Using the Robot Operating System,» de *2014 Joint Conference on Robotics: SBR-LARS Robotics Symposium and Robocontrol*, Sao Carlos, 2014.

[56] J. González, L. Reyes, A. López, P. Jesús, J. Vargas y M. Delgado, «Diseño, Modelado y Simulación de un Robot Bípedo.,» *La Mecatrónica en México*, vol. 3, nº 1, pp. 11-22, 2014.

[57] H. Zhicheng, L. Xiaokun y Z. Fusheng, «ORB-SLAM based humanoid robot location and navigation system,» *Vibroengineering PROCEDIA*, vol. 17, pp. 118-123, 2018.

[58] C. Cuevas, «Ros-gazebo. una valiosa Herramienta de Vanguardia para el Desarrollo de la Robótica,» *Publ. investig*, vol. 10, pp. 143-160, 2016.

[59] D. Gossow, A. Leeper, D. Hershberger y M. Ciocarlie, «Interactive Markers: 3-D User Interfaces for ROS Applications,» *IEEE Robot. Automat. Mag.*, vol. 18, pp. 14-15, 2011.

[60] «Humanoid Robot Field-programmable Gate Array Hardware,» *Sensors and Materials*, vol. 31, nº 6, pp. 1893-1904, 2019.

[61] D. A. Winter, de *Biomechanics and Motor Control of Human Movement*, Waterloo, Ontario, Canada, 2009.

[62] J. A. Lopez Tabano y F. E. Gutierrez Lasso, «TRAYECTORIAS DE MARCHA CON EL ROBOT BIPEDO SUSPENDIDO,» 21 12 2012. [En línea]. Available: https://youtu.be/aNMcR4VbnsE.

[63] GeometriaAnalitica.info, «Fórmula de la distancia entre dos puntos (geometría),» [En línea]. Available: https://www.geometriaanalitica.info/formula-de-la-distancia-entre-dos-puntos-geometria-ejemplos-y-ejercicios-resueltos/. [Último acceso: 13 12 2021].

[64] L. E. Contreras Bravo y L. F. Vargas Tamayo, «Generación de modelos de caminata bípeda a través de diversas técnicas de modelamiento,» [En línea]. Available: https://www.redalyc.org/pdf/4988/498850163003.pdf. [Último acceso: 2021].

[65] A. M. Nogueras, J. L. Calvo Arenillas, J. Orejuelas Rodriguez , F. J. Barbero Iglesias y C. Sánchez Sánchez, «Fases de la marcha humana,» *Revista Iberoamericana de Fisioterapia y Kinesiología*, pp. 44-49, 1999.

7. Anexos

7.1. Caracterización del sensor de presión DF9-40

En este aparatado se dan a conocer los algoritmos que permiten obtener un polinomio característico de orden tres para el comportamiento del sensor. El primero de estos, lee los datos emitidos por un sensor de presión expuesto a varios pesos (dados en gramos), los cuales definen la curva característica de este sensor, y con el fin de dar una medición más óptima de estos datos se toman 25 muestras de lecturas por prueba, una vez obtenidos estos valores se someten a un filtro; este hace la sumatoria de las muestras obtenidas y promedia, para obtener un único valor, el cual corresponde al peso(gr) aplicado al sensor en términos de voltaje, finalmente se realiza el escalizado de lectura y se muestra el valor en el monitor serial.

```
//Se declaran las variables
float pinsensor = 0;  // Indica el pin de lectura de los datos del
sensor
float lectura=0;     //  Dato asignado al peso aplicado al sensor de
presión

void setup() {
 Serial.begin(9600);
}

void loop() {
//Se toman 2 lecturas de voltaje a un mismo peso
lectura = FiltroLectura(25);
/* Se muestra en el monitor serial la lectura en voltios (0-3)
correspondiente a un peso en gramos
*/
Serial.print("lectura analogica = ");
Serial.print((lectura/1023),3);
Serial.println();

delay(60);
}
// Se realiza el filtro de datos
float FiltroLectura(int n)
{
  float suma=0;
  for(int i=0;i<n;i++)
  {
    suma=suma + analogRead(pinsensor);
  }
  float pro_lectura = suma/n;
  return(pro_lectura);
}
```

El segundo algoritmo que interviene en el proceso planteado anteriormente es diseñado en Matlab, y el objetivo de este código es encontrar el polinomio característico de orden 3 para el sensor de Presión DF9-40. Para ello se aplica al sensor 25 pesos(gr) diferentes, de esta manera se aprecia la trayectoria de los datos y se realiza el ajuste de línea. Adicional a esto, se hace una estimación del error estándar, como también el intervalo de predicción y el FIT del error. Para más claridad se realizan las respectivas gráficas y se muestra una taba con los datos antes mencionados, esto se evidencia al ejecutar el siguiente código.

```
close all
clear all
clc

%% Vector con los pesos (en gramos) aplicada al sensor

Masa = [0, 97.2, 154, 250, 463, 500, 622.8, 733, 750, 1000, 1250,1500,
1750, 2000 ...
            ,2250, 2500, 2750, 3000, 3250, 3500, 3750, 4000,
4500,4750,5000]';

%%  Vector con los voltajes emitidos por el sensor ante la aplicación de
cada peso
Voltaje = [0, 0.048, 0.052, 0.127, 0.149,0.178, 0.208, 0.253, 0.263,
0.334, 0.358, 0.413, 0.425, 0.447 ...
               ,0.441, 0.466, 0.481, 0.504, 0.515, 0.527, 0.535, 0.545,
0.553,0.563, 0.567]';
%% Se obtiene el Polinomio Característico para el sensor
[coefPoli,S] = polyfit(Masa, Voltaje, 3)

%% Se estima el intervalo de predicción de acuerdo a la estimación del
error estándar(delta)
[vol2,delta] = polyval(coefPoli, Masa,S);

%% Se gráfica las señales en cuestión y se muestra una La tabla de datos

plot(Masa, Voltaje, 'bo' )
hold on
plot (Masa, vol2, 'r-' )
plot (Masa, vol2 + 2 * delta, 'm--' , Masa, vol2-2 * delta, 'm--' )
title( 'Ajuste lineal de datos con intervalo de predicción del 95%' )
legend( 'Datos' , 'Ajuste lineal' , 'Intervalo de predicción del 95%' )
T = table(Masa,Voltaje,vol2,Voltaje-
vol2,'VariableNames',{'Masa','Voltage','Fit','FitError'})
```

7.2. Cálculo del Modelo Geométrico Directo (MGD)

En este apartado se encuentra el desarrollado un script, que su objetivo es encontrar el Modelo Geométrico Directo de una de las piernas del robot bípedo, para ello se efectúa la multiplicación sucesiva de izquierda a derecha de las 6 matrices de transformación, se hace

de manera simbólica, pero se puede calcular de manera numérica, pasando estas variables a formato numérico y asignando a cada una el valor del parámetro correspondiente.

```
close all
clear all
clc
%Creación de variables simbólicas, que representan los ángulos (t#),
% las longitudes(L#,R#) y altura del robot (hr).
syms t1 t2 t3 t4 t5 t6;
syms Lo L7 L9 L10 L11 L12 L13 R8 R12 hr;
%% Se cargan las matrices de transformación de cada articulación
TR =  [   0  0  0  0;
          0  1  0  0;
          0  0  1  hr
          0  0  0  1];
T1 = [ cos(t1) -sin(t1) 0 Lo;
       sin(t1) cos(t1)  0  0;
          0       0     1  L7
          0       0     0  1];
T2 = [ cos(t2) -sin(t2) 0 0;
            0    0   -1   0;
         sin(t2) cos(t2) 0 0;
          0      0      0  1];

T3 = [ cos(t3) -sin(t3) 0 L9;
            0    0   -1   -L10;
         sin(t3) -sin(t3) 0 0;
          0      0      0  1];
T4 = [ cos(t4) -sin(t4) 0 L11;
       sin(t4) cos(t4)  0  0;
          0       0     1  R8;
          0      0      0  1];
T5 = [ cos(t5) -sin(t5) 0 L12;
       sin(t5) cos(t5)  0  0;
          0       0     1  0;
          0      0      0  1];
T6 = [ cos(t6)  -sin(t6) 0   L13;
            0    0     1    R12;
       -sin(t6)  -cos(t6)  0  0;
          0      0      0    1];
%% Se efectúa multiplicaciones sucesivas de derecha a izquierda
OT1 = TR*T1;
OT2 = OT1*T2;
OT3 = OT2*T3;
OT4 = OT3*T4;
OT5 = OT4*T5;
%% Cálculo del MGD
OT6 = simplify(OT5*T6)
```

7.3. Lectura de sensores y cálculo del ZMP

En este apartado, se dan a conocer los códigos que se implementaron para la toma de datos dados por los sensores desde la ESP32, mediante la plataforma de Arduino hasta el tratamiento dado a estos datos para el cálculo del ZMP en la plataforma de Matlab. Para este proceso se requirió del paso de información por medio de Nodos ROS, conexión wifi y comunicación serial.

Etapa 1: Adquisición de datos emitidos por los sensores de presión

El siguiente código fue desarrollado en la plataforma de Arduino ID, este se encarga de la adquisición de los datos emitidos por los sensores de presión hacia la ESP32, y el posterior envió de esos datos hacia el nodo publicador ROS creado en Python 2.7.17 a través del puerto serial. El código se presenta a continuación.

```
/*
Se definen las variables que albergan los números de los pines GPIO de
la tarjeta ESP32, por los cuales se adquieren los datos emitidos por los
sensores.
*/
#define sen1Pin  35
#define sen2Pin  36
#define sen3Pin  39
#define sen4Pin  34
#define sen5Pin  32
#define sen6Pin  33
#define sen7Pin  25
#define sen8Pin  26
// Creamos una cadena que guarda los datos de los sensores
String DatosSensores= " ";
// Creamos un Array de 8 posiciones para la lectura de los datos
float DatSen[8];

void setup() {
 // Se definen los pines GPIO como entrada
  pinMode(sen1Pin, INPUT);
  pinMode(sen2Pin, INPUT);
  pinMode(sen3Pin, INPUT);
  pinMode(sen4Pin, INPUT);
  pinMode(sen5Pin, INPUT);
  pinMode(sen6Pin, INPUT);
  pinMode(sen7Pin, INPUT);
  pinMode(sen8Pin, INPUT);

  Serial.begin(115200);
}
void loop() {
// Leemos cada dato emitido por los sensores y escalizamos de 0-5
voltios
```

```
DatSen[0] = map(analogRead(sen1Pin), 0, 4095, 0,5);
DatSen[1] = map(analogRead(sen1Pin), 0, 4095, 0,5);
DatSen[2] = map(analogRead(sen1Pin), 0, 4095, 0,5);
DatSen[3] = map(analogRead(sen1Pin), 0, 4095, 0,5);
DatSen[4] = map(analogRead(sen1Pin), 0, 4095, 0,5);
DatSen[5] = map(analogRead(sen1Pin), 0, 4095, 0,5);
DatSen[6] = map(analogRead(sen1Pin), 0, 4095, 0,5);
DatSen[7] = map(analogRead(sen1Pin), 0, 4095, 0,5);
// Se concatena cada dato obtenido en una cadena
for (int i = 0; i < 8; i = i + 1) {

    DatosSensores = "";
    DatosSensores += DatSen[0];
    DatosSensores += ";";
    DatosSensores += DatSen[1];
    DatosSensores += ";";
    DatosSensores +=DatSen[2];
    DatosSensores += ";";
    DatosSensores += DatSen[3];
    DatosSensores += ";";
    DatosSensores += DatSen[4];
    DatosSensores += ";";
    DatosSensores += DatSen[5];
    DatosSensores += ";";
    DatosSensores += DatSen[6];
    DatosSensores += ";";
    DatosSensores += DatSen[7];

   }

   // Se envía los datos por el Puerto serial al Nodo ROS creado en
PYTHON
   Serial.println(DatosSensores);
   delay (300);

}
```

Etapa2: Recepción de los datos en Python y envió hacia Matlab

Este archivo es desarrollado en Python 2.7.17 y contiene el nodo publicador de Ros, el cual debe ser ejecutado en una terminal, luego de haber subido el programa de adquisición de datos en la ESP32 (etapa1_Anexo 3), este nodo se encarga de acceder al puerto serial a través de la función leerarduino(), también establece conexión con Rosbridge para enviar los datos de los sensores hacia la plataforma de Matlab a través del tópico “/Sensores”.

```
1 # coding=utf-8
2 import time
```

```
3 import serial
4 import roslibpy
5
6 #Se obtiene los valores enviados de la ESP32 y se los decodifica en
  una cadena
7
8 def leerarduino(puerto, velocidad):
9     try:
10        ser = serial.Serial(puerto, velocidad)
11        ser.close()
12        ser.open()
13        m = ser.readline()
14        sensores = m.decode()
15        ser.flush()
16        return sensores
17    except KeyboardInterrupt:
18        print('interrupted!')
19
20 #Se establece la conexión con Rosbridge
21 client = roslibpy.Ros(host='192.168.100.11', port=9090)
22 client.run()
23 print(client.get_params)
24
25 # Se especifica el tópico en el cual se va a publicar en este caso
  sensores
26 talker = roslibpy.Topic(client, '/Sensores', 'std_msgs/String')
27
28 while client.is_connected:
29     #Conexión por el puerto serial
30     sensores = leerarduino('/dev/ttyUSB0', 115200)
31     #Publica el mensaje en el tópico
32     talker.publish(roslibpy.Message({'data': sensores}))
```

```
33      #Se muestra lo recibido en la terminal
34      print(sensores)
35      time.sleep(1.1)
36      talker.unadvertise()
37
    client.terminate()
```

Es preciso decir que no se puede abrir en simultaneo la herramienta Monitor serial de Arduino y mostrar la lectura en la terminal en Python, ya que las dos plataformas ocuparían el puerto serial y por ende existiría un conflicto generando a un error que interrumpe la comunicación o él envió erróneo de datos.

Etapa 3: Recepción, manipulación y cálculo del ZMP deseado

Este programa es creado en la plataforma de Matlab y es ejecutado desde una computadora remota, por ende, se realiza la conexión con Rosbridge. Para la obtención de la cadena enviada desde el nodo publicador creado en Python (etapa2), se crea un nodo suscriptor llamado “Tsensores” el cual accede al tópico “/Sensores” y almacena la información existente en este.

Como sabemos para el cálculo del ZMP es necesario trabajar con números, es por esto que se convierte dicha cadena en formato de número doublé, de esta manera se procede a escalizar los datos dados por los sensores, por medio de la ecuación general de un polinomio de orden 3 y los coeficientes obtenidos en el (Anexo 1). Una vez realizado el proceso anterior y con los parámetros de distancias se calcula el punto ZMP, finalmente para tener una vista más detallada del comportamiento del ZMP ante movimientos del robot bípedo, se gráfica: el polígono de soporte, el COP y el punto ZMP deseado.

```
clear all
close all
clc
%% Se establece la comunicación con el WebSockets-ROS
  rosshutdown;
  while strcmp(estarosonline('localhost',11311),"Satisfactorio")
  rosinit('localhost',11311);

 try
  TSensores = rossubscriber('/Sensores');
  scandata = receive(TSensores,2); %Recibe hasta dos mensajes de ROS
  Valorsensores=strsplit(scandata.Data,';');%Divide el mensaje hasta
donde encuentre un ;
  Valorsensores=str2double(Valorsensores) %Convierte la cadena a numero
(double)
```

```
  %% A continuación se definen los parámetros necesarios para la
obtención del ZMP
% Distancias en X
l1=-8.45;
l2=-4.95;
l3=-8.45;
l4=-4.95;
l5=4.95;
l6=8.45;
l7=4.95;
l8=8.45;
% Distancias en Y
L1=6.05;
L2=6.05;
L3=-6.05;
L4=-6.05;
L5=6.05;
L6=6.05;
L7=-6.05;
L8=-6.05;
%% Se definen los elementos necesarios para establecer el polígono de
soporte
L = 15.7;    % Longitud de los pies
A = 6.4;     % Ancho de los pies
S = 6.9;     % Separación entre los pies del robot

x_der = [S/2 (S/2 + A) (S/2 + A) S/2];
y_der = [-L/2 -L/2 L/2  L/2];

x_izq = [-S/2 (-S/2 - A) (-S/2 - A)  -S/2];
y_izq = [-L/2 -L/2  L/2  L/2];
%% Se realiza la escalización De los datos emitidos por los sensores
%(de acuerdo al polinomio de caracterización)
  vs1=0.0102*(Valorsensores(1,1)^3)+0,0004*(Valorsensores(1,1)^2);
  vs2=0.0102*(Valorsensores(1,2)^3)+0,0004*(Valorsensores(1,2)^2);
  vs3=0.0102*(Valorsensores(1,3)^3)+0,0004*(Valorsensores(1,3)^2);
  vs4=0.0102*(Valorsensores(1,4)^3)+0,0004*(Valorsensores(1,4)^2);
  vs5=0.0102*(Valorsensores(1,5)^3)+0,0004*(Valorsensores(1,5)^2);
  vs6=0.0102*(Valorsensores(1,6)^3)+0,0004*(Valorsensores(1,6)^2);
  vs7=0.0102*(Valorsensores(1,7)^3)+0,0004*(Valorsensores(1,7)^2);
  vs8=0.0102*(Valorsensores(1,8)^3)+0,0004*(Valorsensores(1,8)^2);

 %% Se encuentra la coordenada del ZMP en X e Y

 Xcop=((vs1*l1)+(vs2*l2)+(vs3*l3)+(vs4*l4)+(vs5*l5)+(vs6*l6)+(vs7*l7)+(
 vs8*l8))/(vs1+vs2+vs3+vs4+vs5+vs6+vs7+vs8);
 Ycop=((vs1*L1)+(vs2*L2)+(vs3*L3)+(vs4*L4)+(vs5*L5)+(vs6*L6)+(vs7*L7)
 +(vs8*L8))/(vs1+vs2+vs3+vs4+vs5+vs6+vs7+vs8);

  ext = 1.15;
```

```
  %%---- Se realizan las gráficas correspondientes de COP, ZMP deseado y
el polígono de soporte
  figure(1)
  axh = axes('Parent',gcf);
  cla(axh)
  patch(x_der,y_der,'r')
  alpha(0.15);
  hold on
  patch(x_izq,y_izq,'b')
  alpha(0.15);
  xlabel('X');
  ylabel('Y');
  drawnow
  axis([(-S/2 - A)*ext  (S/2 + A)*ext  (-L/2)*ext (L/2)*ext]);
  title('ZMP EN PUNTO DE GENUFLEXIÓN');
  plot(0,0,'b+');
  plot(Xcop,Ycop,'*');
  txt = '\leftarrow';
  text(Xcop,Ycop,txt,'FontSize',14)
  legend('Pie Derecho','Pie Izquierdo','COP','ZMP Deseado');
  grid on;
  xlabel('X');
  ylabel('Y');
  drawnow

 end
  pause(6)
end
```

7.4. Generación de trayectorias articulares

A continuación, se presenta el código implementado en la plataforma de Matlab para la generación de trayectorias articulares por el método de ondas sinusoidales acopladas. En este se realiza la inicialización de variables comunes y de variables independientes para cada articulación, las primeras son parámetros usados para la generación de todas las trayectorias y poseen el mismo valor para el cálculo de cada una de estas, las cuales son: el tiempo de ciclo, pasos, muestras, tiempo y frecuencia. Por otro lado, tenemos las variables independientes: desplazamiento, amplitud y fase, estas variables se asignan a cada articulación y sus valores pueden diferir para el cálculo de cada trayectoria articular. Adicionalmente, una vez establecidos los parámetros de cada variable, se calcula y grafican las trayectorias de referencia generadas para cada articulación. Es preciso decir que el código fue creado como una función, ya que será usado en el Anexo 5 y Anexo 6.

```
// Se crea la función que retorna la matriz de señales de referencias
function Matriz = F_GeneracionTrayectorias()
%% Variables Comunes
```

```
TiempoCiclo=20;
Pasos=0.2;
Muestras=TiempoCiclo/Pasos;
Tiempo = (0:Pasos:TiempoCiclo);
Frecuencia=1/TiempoCiclo;
%% ARTICULACION1: Cadera izquierda(M1)
Desplazamiento_1 = (pi/180*60;
Amplitud_1 = 0;
Fase_1 = 0;
%% ARTICULACION 2: Cadera izquierda(M2)
Desplazamiento_2 = (pi/180*60;
Amplitud_2 = 0;
Fase_2= 0;
%% ARTICULACION 3: Cadera izquierda(M3)
Desplazamiento_3 = (pi/180*42.5;
Amplitud_3 = (pi/180)*15;
Fase_3 = 0;
%% ARTICULACION 4 (Rodilla)
Desplazamiento_4 = (pi/180)*102.5;
Amplitud_4 = (pi/180)*7;
Fase_4 = 0;
%% ARTICULACION 5
Desplazamiento_5 = (pi/180)*70;
Amplitud_5 =(pi/180)*7;
Fase_5 = 0;
%%   ARTICULACION 6
Desplazamiento_6 = (pi/180)*60;
Amplitud_6 = 0;
Fase_6 = 0;
%%   ARTICULACION 7
Desplazamiento_7 = (pi/180)*60;
Amplitud_7 = 0;
Fase_7 = 0;
%%   ARTICULACION 8
Desplazamiento_8 = (pi/180)*60;
Amplitud_8 =0;
Fase_8 = 0;
%%   ARTICULACION 9
Desplazamiento_9 = (pi/180)*122.5;
Amplitud_9 = (pi/180)*15;
Fase_9 = 0;
%%   ARTICULACION 10
Desplazamiento_10 = (pi/180)*62.5;
Amplitud_10 = (pi/180)*7;
Fase_10 = 0;
%% ARTICULACION 11
Desplazamiento_11 = (pi/180)*50;
Amplitud_11 = (pi/180)*7;
Fase_11 = 0;
%%   ARTICULACION 12
```

```
Desplazamiento_12 = (pi/180)*60;
Amplitud_12 = 0;
Fase_12 = 0;
%% Se generan las trayectorias articulares sinusoidales, para cada
articulación
% Trayectorias para articulaciones estáticas
Referencia_1= Desplazamiento_1; % Para articulación1
Referencia_2= Desplazamiento_2;
Referencia_6=Desplazamiento_6;
Referencia_7=Desplazamiento_7;
Referencia_8=Desplazamiento_8;
Referencia_12= Desplazamiento_12;
% Trayectorias para articulaciones que presentan movimiento
Referencia_3=Desplazamiento_3 + (Amplitud_3 * sin((2*pi*Frecuencia *
Tiempo) + Fase_3));
Referencia_4=Desplazamiento_4 + (Amplitud_4 * sin((2*pi*Frecuencia *
Tiempo) + Fase_4));
Referencia_5=Desplazamiento_5 + (Amplitud_5 * sin((2*pi*Frecuencia *
Tiempo) + Fase_5));
Referencia_9= Desplazamiento_9 + (Amplitud_9 * sin((2*pi*Frecuencia *
Tiempo) + Fase_9));
Referencia_10= Desplazamiento_10 + (Amplitud_10 * sin((2*pi*Frecuencia *
Tiempo) + Fase_10));
Referencia_11= Desplazamiento_11+ (Amplitud_11* sin((2*pi*Frecuencia *
Tiempo) + Fase_11));

Referencia_1=Referencia_1*ones(1,(Muestras+1));
Referencia_2=Referencia_2*ones(1,(Muestras+1));
Referencia_6=Referencia_6*ones(1,(Muestras+1));
Referencia_7=Referencia_7*ones(1,(Muestras+1));
Referencia_8=Referencia_8*ones(1,(Muestras+1));
Referencia_12=Referencia_12*ones(1,(Muestras+1));
%% Se guardan las trayectorias generadas en una matriz
Matriz = [Referencia_1;Referencia_2;
Referencia_3;Referencia_4;Referencia_5;...
Referencia_6;Referencia_7;Referencia_8;
Referencia_9;Referencia_10;Referencia_11;...
Referencia_12];
%% Graficamos las señales de referencia para cada articulación
(descomentar líneas)
% Para mirar la gráfica descomentar las líneas que se encuentran
enseguida
% [R,D] = size(Matriz);
% Time = linspace(0,TiempoCiclo,D);
% figure(1);
% subplot(3,2,1);
% plot(Time,Matriz(6,:)); title('ARTICULACION 6');
% subplot(3,2,2);
% plot(Time,Matriz(12,:)); title('ARTICULACION 12');
% subplot(3,2,3);
```

```
% plot(Time,Matriz(5,:)); title('ARTICULACION 5');
% subplot(3,2,4);
% plot(Time,Matriz(11,:)); title('ARTICULACION 11');
% subplot(3,2,5);
% plot(Time,Matriz(2,:)); title('ARTICULACION 2');
% subplot(3,2,4);
% plot(Time,Matriz(8,:)); title('ARTICULACION 8');
% subplot(3,2,5);
% plot(Time,Matriz(1,:)); title('ARTICULACION 1');
% subplot(3,2,6);
% plot(Time,Matriz(7,:)); title('ARTICULACION 7');
% plot(Time,Matriz(4,:)); title('ARTICULACION 4');
% subplot(3,2,6);
% plot(Time,Matriz(10,:)); title('ARTICULACION 10');
%
% figure(2);
% subplot(3,2,1);
% plot(Time,Matriz(3,:)); title('ARTICULACION 3');
% subplot(3,2,2);
% plot(Time,Matriz(9,:)); title('ARTICULACION 9');
% subplot(3,2,3);
% plot(Time,Matriz(2,:)); title('ARTICULACION 2');
% subplot(3,2,4);
% plot(Time,Matriz(8,:)); title('ARTICULACION 8');
% subplot(3,2,5);
% plot(Time,Matriz(1,:)); title('ARTICULACION 1');
% subplot(3,2,6);
% plot(Time,Matriz(7,:)); title('ARTICULACION 7');
end
```

7.5. Punto de genuflexión y trayectorias de movimiento articular

Este anexo da a conocer las respectivas etapas para lograr que el robot permanezca en posición de genuflexión o siga una trayectoria de movimiento articular unidireccional, para esto se establece la comunicación a través de nodos ROS (desarrollados en las plataformas de Matlab y Python), comunicación por el puerto serial y conexión wifi, con la finalidad de hacer el paso de información entre las tarjetas para llevar la orden hasta el robot bípedo.

Etapa 1: Envió de las trayectorias articulares

En esta etapa se da a conocer el código creado en la plataforma de Matlab, para realizar el envió de las trayectorias articulares por medio de wifi y nodos ROS. Estas trayectorias articulares llevan al robot a la posición de genuflexión o realizar dos pasos de caminata, en la primera el robot se encuentra erguido y estable, para el cálculo de estas trayectorias, en el Anexo 4, se deben especificar el valor de las variables independientes respecto a la Tabla 4 (Parámetros de Trayectorias de Marcha Articulares mediante Ondas Sinusoidales Acopladas para el robot propuesto), como se evidencia, para obtener esta posición solo es

necesario tener el valor adecuado de la variable desplazamiento, ya que se encarga de mantener a la articulación en una posición angular estática. Por otro lado, para generar una trayectoria simulando la caminata humana, no es necesario realizar cambios en el Anexo 4. Como podemos notar, la diferencia entre las dos situaciones es el valor de la amplitud de la señal sinusoidal, a la cual se le asigna un valor de cero si se quiere un movimiento estático o una amplitud diferente de cero si se desea movimiento articular, hay que tener en cuenta que el valor de la amplitud se debe establecer rigurosamente, ya que el robot puede perder estabilidad.

El paso siguiente luego de realizar lo especificado en el párrafo anterior, es el envió de las señales hacia la plataforma de Python, para ello se debe hacer la comunicación con rosbridge, en seguida se adquieren y se almacenan las trayectorias articulares, para luego ser manipuladas y generar una cadena de datos, la cual se comparte por el nodo publicador "PubMotores" a través del tópico llamado "Tmotores" en forma de mensaje. Este nodo estará enviando las cadenas hasta que se interrumpa la conexión pulsando Ctrl + c en el command window. El código se presenta a continuación.

```
clear all
close all
clc

%% Se inicializa la comunicación con Rosbridge
rosshutdown;
while strcmp(estarosonline('localhost',11311),"Satisfactorio")
rosinit('localhost',11311);

try
%%  Se obtiene las trayectorias articulares
Matriz = F_GeneracionTrayectorias();
%%  Pasamos las señales articulares obtenidas en radianes a grados
MatrizDat = (180/pi).*Matriz;
MatrizRef = floor(MatrizDat)
[R,D] = size(MatrizRef);
% Recorremos la matriz que contiene las trayectorias articulares, para
% obtener el primer dato de cada trayectoria
for i = 1:1:D
    for j = 1:1:R
        con = i;
        % Se almacenan en una vector fila cada datos obtenido
        VecRef (1,j) = MatrizRef(j,i);
    end
% Creamos un terminador, el cual indica el fin de cada cadena de datos
enviados hacia el nodo Suscriptor creado en Python
    Terminador = "T";
    Salida = VecRef;
%Se crea la cadena de datos para él envió de los grados a los que
se posiciona cada articulación
```

```
Salida=strcat(num2str(Salida(1,1)),';',num2str(Salida(1,2)),';',num2str(
Salida(1,3)),';',num2str(Salida(1,4)),';'...

,num2str(Salida(1,5)),';',num2str(Salida(1,6)),';',num2str(Salida(1,7)),
';',num2str(Salida(1,8)),';',...
                num2str(Salida(1,9)),
';',num2str(Salida(1,10)),';',num2str(Salida(1,11)),';',num2str(Salida(1
,12)),';',Terminador)
    pause(0.7)
% Se crea el Nodo publicador llamado "PubMotores" y se define el tópico
"TMotores",
   %por el cual se está pasando las trayectorias articulares.
PubMotores = rospublisher('/TMotores', 'std_msgs/String')
esfcmsg = rosmessage(PubMotores);
esfcmsg.Data =Salida;
send(PubMotores,esfcmsg)

 j = 1;
end
end
 pause(0.5)
end
```

Etapa 2: Recepción de las trayectorias en la Raspberry pi y envió de las mismas hacia la tarjeta ESP32

En esta etapa se da a conocer el nodo suscriptor desarrollado en Pyhon, el cual recibe las trayectorias articulares enviadas por el tópico “Tmotores” desde el nodo publicador de la plataforma de Matlab (Anexo 5_Etapa1), y envía las respectivas trayectorias por el puerto serial, para ello, se declaran las librerías requeridas para trabajar con ROS y el puerto serial, luego se establece la conexión con rosbridge para tener acceso al nodo publicador, en seguida se crea el nodo suscriptor llamado “listener” para acceder al mensaje que contiene las trayectorias desde el tópico “Tmotores”, finalmente se realiza la conexión con el puerto serial y se envía las trayectorias por este.

```
1 #encoding=utf-8
2 from __future__ import print_function
3 import serial
4 import time
5 import roslibpy
6 from roslibpy.core import Message
7
```

```
# Se realiza la conexión con el Rosbridge y se inicializa el cliente
client = roslibpy.Ros(host='192.168.100.11', port=9090)
client.run()
#Creaión del nodo suscriptor
listener = roslibpy.Topic(client, '/TMotores', 'std_msgs/String')
listener.subscribe(lambda message: mensajeobtenido(message['data']))
# Se crea establece la comunicación por el puerto serial
arduino = serial.Serial(port='/dev/ttyUSB0', baudrate=115200,
timeout=0.5)
# Realizamos él envió de las trayectorias articulares por el puerto
serial
def write_read(x):
    arduino.write(str(x).encode('utf-8'))
    time.sleep(0.9)
    data = arduino.readline()
    return data
# Se obtiene y muestra las trayectorias articulares enviadas desde
Matlab
def mensajeobtenido(datos):
    val = write_read(datos)
    time.sleep(0.9)
    print(val)

try:
    while True:
        pass
except KeyboardInterrupt:
    client.terminate()
```

Etapa 3: Recepción de las trayectorias en la Esp32 y envió de las señales a las articulaciones del Robot Bípedo

En esta etapa se muestra el código empleado para enviar las trayectorias a cada articulación del robot. Estando en el entorno de Arduino, se debe incluir la biblioteca de ESP32.Servo.h

en la cabecera del código, esta librería permite controlar los servomotores por los puertos PWM de la placa ESP32, luego se crean doce objetos de tipo servo, después de esto, se definen doce variables en las cuales se les asigna el número del pin GPIO (PWM) donde se conectan los servomotores. Adicionalmente, se inicializan variables Pos# y Cad#, la primera se encarga de guardar el número de la posición donde se encuentra el limitador ";" y la segunda es donde se guardan los datos que se enviaran a los servomotores. Estando en el Void setup(), se conecta la variable servo al número del pin GPIO con el comando attach(). Para finalizar en el Void loop se define un ciclo while que se estará ejecutando mientras existan datos para leer en el puerto serial, al interior del ciclo se obtienen las cadenas enviadas por el puerto serial hasta el terminador especificado en Matlab ("T"), se hace la respectiva separación de los datos para cada servomotor, y se convierten a formato de numérico Float para ser enviados a su respectiva articulación.

```
#include <ESP32Servo.h>
//Se crean objetos de tipo Servo, una para cada articulación
Servo myservo1;
Servo myservo2;
Servo myservo3;
Servo myservo4;
Servo myservo5;
Servo myservo6;
Servo myservo7;
Servo myservo8;
Servo myservo9;
Servo myservo10;
Servo myservo11;
Servo myservo12;
// Se definen las variables que albergan el pin GPO
// por el cual se envían las señales a cada articulación
#define Servo1Pin  25
#define Servo2Pin  26
#define Servo3Pin  27
#define Servo4Pin  14
#define Servo5Pin  13
#define Servo6Pin  12
#define Servo7Pin  23
#define Servo8Pin  22
#define Servo9Pin  1
#define Servo10Pin 3
#define Servo11Pin 21
#define Servo12Pin 19
// Se inicializan las variables necesarias para realizar la adquisición,
manipulación y envió de los datos
int
pos1=0,pos2=0,pos3=0,pos4=0,pos5=0,pos6=0,pos7=0,pos8=0,pos9=0,pos10=0,p
os11=0,pos12=0,pos13=0;
```

```
String cad = "", cad1= "", cad2= "",cad3= "",cad4= "",cad5= "",cad6 =
"",cad7 = "", cad8= "", cad9= "",cad10= "",cad11= "",cad12= "",cad13=
"";

void setup() {

//Asignamos el pin GPO a cada objeto de tipo Servo
myservo1.attach(Servo1Pin);
myservo2.attach(Servo2Pin);
myservo3.attach(Servo3Pin);
myservo4.attach(Servo4Pin);
myservo5.attach(Servo5Pin);
myservo6.attach(Servo6Pin);
myservo7.attach(Servo7Pin);
myservo8.attach(Servo8Pin);
myservo9.attach(Servo9Pin);
myservo10.attach(Servo10Pin);
myservo11.attach(Servo11Pin);
myservo12.attach(Servo12Pin);

Serial.begin(115200);
}
void loop() {
// Se verifica si existen datos para leer en el puerto serial
 while (!Serial.available());
// Lee la cadena que se encuentre en el puerto serial hasta
// donde encuentre el terminador (T) enviado desde Matlab
 cad = Serial.readStringUntil('T');
//Encuentra y guarda la posición donde está el ";"
 pos1  = cad.indexOf(';');
 pos2 = cad.indexOf(';',pos1+1);
 pos3 = cad.indexOf(';',pos2+1);
 pos4 = cad.indexOf(';',pos3+1);
 pos5 = cad.indexOf(';',pos4+1);
 pos6 = cad.indexOf(';',pos5+1);
 pos7 = cad.indexOf(';',pos6+1);
 pos8 = cad.indexOf(';',pos7+1);
 pos9 = cad.indexOf(';',pos8+1);
 pos10 = cad.indexOf(';',pos9+1);
 pos11 = cad.indexOf(';',pos10+1);
 pos12 = cad.indexOf(';',pos11+1);
 pos13 = cad.indexOf(';',pos12+1);
 // Guardamos cada dato de la cadena por separado
 cad1  = cad.substring(0,pos1);
 cad2  = cad.substring(pos1+1,pos2);
 cad3  = cad.substring(pos2+1,pos3);
 cad4  = cad.substring(pos3+1,pos4);
 cad5  = cad.substring(pos4+1,pos5);
 cad6  = cad.substring(pos5+1,pos6);
 cad7  = cad.substring(pos6+1,pos7);
```

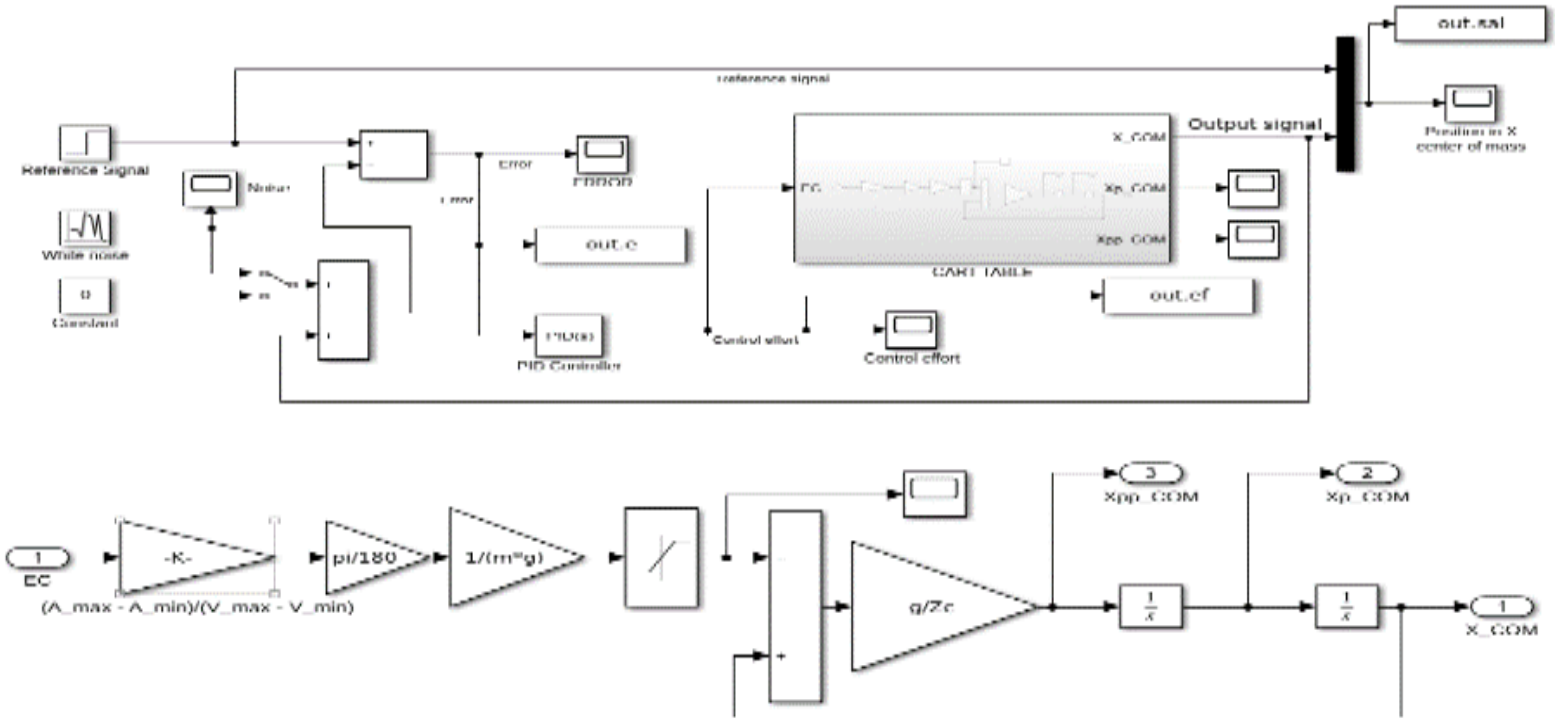
```
 cad8  = cad.substring(pos7+1,pos8);
 cad9  = cad.substring(pos8+1,pos9);
 cad10 = cad.substring(pos9+1,pos10);
 cad11 = cad.substring(pos10+1,pos11);
 cad12 = cad.substring(pos11+1,pos12);
 cad13 = cad.substring(pos12+1,pos13);
 // Mostramos en el Monitor Serial la cadena que se recibió
 Serial.println("Se recibio: "+cad1 +";"+ cad2+";"+ cad3+";"+ cad4+";"+
cad5+";"+cad6 +";"+ cad7+";"+ cad8+";"+ cad9+";"+ cad10+";"+ cad11+";"+
cad12);//"+";"+ cad13);
 // Convertimos cada fragmeno de la cadena que contienen los grados de
giro
 // para cada motor en Numero Float y le enviamos la señal a cada
articulación
 myservo1.write(cad1.toFloat());
 myservo2.write(cad2.toFloat());
 myservo3.write(cad3.toFloat());
 myservo4.write(cad4.toFloat());
 myservo5.write(cad5.toFloat());
 myservo6.write(cad6.toFloat());
 myservo7.write(cad7.toFloat());
 myservo8.write(cad8.toFloat());
 myservo9.write(cad9.toFloat());
 myservo10.write(cad10.toFloat());
 myservo11.write(cad11.toFloat());
 myservo12.write(cad12.toFloat());
 }
```

7.6. Simulación del Modelo Cart_Table en Simulink

El siguiente código define los parámetros del robot bípedo, empleados para la simulación del modelo Cart_Table en Simulink.

```
close all;
clear all;
clc;

m     = 2495;      % Masa del robot (gramos)
g     = -9.81;     % Gravedad
Zc    = 65.1;      % Longitud del suelo a la cadera
Xo    = 0.01;      % Posición inicial
Vo    = 0;         % Velocidad inicial
Lmax  = 8.5;     % Longitud borde frontal del pie
Lmin  = -7.3;    % Longitud borde trasero del pie
A_max = 120;     % Ángulo en grados máximo del servomotor
A_min = 0;       % Angulo en grados mínimo del servomotor
V_max = 255;     % valor digital máximo en Arduino para comandar la
posicion del servomotor
V_min = 0;       % valor digital mínimo en Arduino para comandar la
posición del servomotor
```



Cart_Table

Figura 38. Representación del Modelo Cart_Table en diagramas de bloque en Simulink

7.5 Diagrama de bloques simulink multibody

A continuación, se da una breve explicación de cada bloque utilizado para realizar la simulación del robot bípedo en simulink multibody.

Solver Configuration: Este bloque es requerido en todos los modelos representados en simscape, en el cual se especifican los parámetros del solucionador para lograr la simulación del sistema representado.

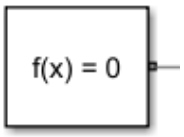


Figura 40. Bloque solver configuration

Mechanism Configuration: Este bloque proporciona parámetros mecánicos y de simulación al modelo creado en Multibody, los parámetros incluyen la gravedad y un delta de linealización para calcular derivadas parciales numéricas durante la linealización del sistema. En la figura 7.3 se especifican estas características para modelo propuesto.

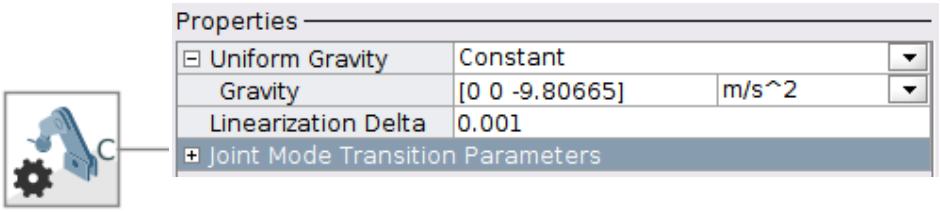


Figura 39. Bloque y propiedades de mechanism configuration

World Frame: Este elemento es esencial en la construcción del sistema de bloques que representan al robot propuesto, se conoce como el marco de referencia global, donde sus sus ejes son ortogonales y están dispuestos de acuerdo con la regla de la mano derecha. Apartir de este se definen los demás marcos del sistema.

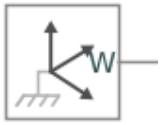


Figura 41. Bloque world frame

Brick Solid: Permite la creación de cuerpos rígidos para la representación de eslabones propios del robot, en este bloque se especifica la masa, dimensión, tipo de material, entre otras características. Es preciso decir que estos eslabones fueron creados a partir de la Tabla 1, donde se especifican las longitudes de los componentes del robot. A continuación, se muestra la creación del eslabón que representa la cadera del robot.

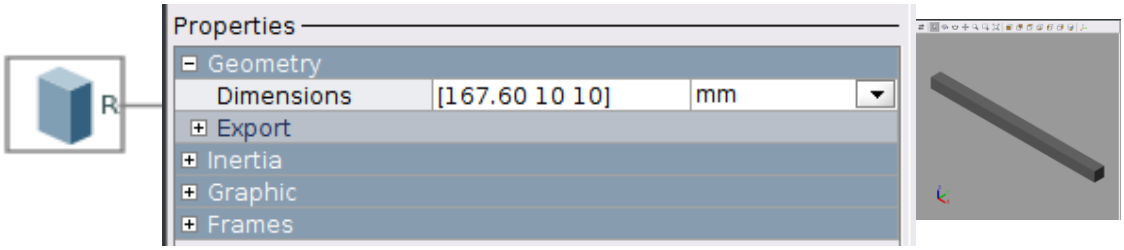


Figura 42. Eslabón cadera en Multibody

Transform Sensor: Se encarga de indicar la posición cartesiana y orientación respecto al eje X,Y e Z que toman los pies del robot.

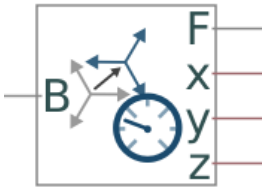


Figura 43. Bloque transform sensor

Spherical Solid: Este bloque es utilizado para la representación de las articulaciones del robot, dando una mejor perspectiva del movimiento de cada eslabón del robot.

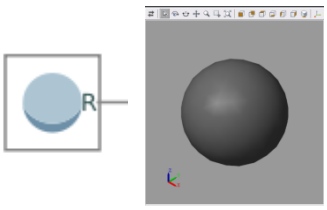


Figura 44. Bloque spherical solid

Rigid Transform: Es usado para realizas las transformaciones o rotaciones de los ejes de cada componente del modelo, teniendo en cuenta sobre que eje se llevara a cabo cada una de estas, en la Figura 7.8 se muestra una traslación sobre el eje -Z de 13.945 mm.

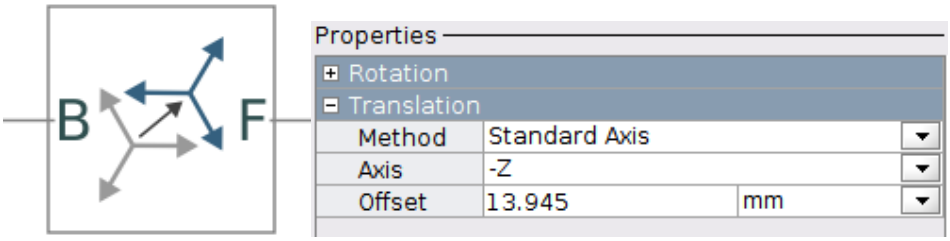


Figura 45. Bloque rigid transform.

Revolute Joint: Representa una articulación rotoide, en esta se especifica los límites de giro que tendrá la articulación, el ángulo, la velocidad, entre otras características. Este bloque puede ser actuado de manera interna o por una señal externa.

Figura 46. Bloque revolutejoin.