UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Comparación de dos algoritmos de control de marcha de un robot bípedo de 12 GDL

TESIS

Que para obtener el título de

Ingeniero Mecatrónico

P R E S E N T A

Eduardo Ontiveros García

DIRECTOR DE TESIS

Dr. Edmundo Gabriel Rocha Cózatl

Ciudad Universitaria, Cd. Mx., 2018

"…La batalla de la vida no siempre la gana el hombre más fuerte,

o el más ligero, porque tarde o temprano, el hombre que gana,

es aquél que cree poder hacerlo."

-Rudyard Kipling-

AGRADECIMIENTOS

A mi mamá, quien ha sido mi mayor sostén y me ha brindado su infinito amor.

A mi papá, quien ha sido el mejor maestro y siempre ha estado a mi lado.

A Alma Delia García Crescencio, por todo el tiempo y la dedicación aportados a este documento.

A mi hermano, por su constante guía y ejemplo.

A Karla Arista quién, además de ser la mejor amiga, realizó invaluables intervenciones.

A mi familia, por estar siempre a mi lado, dándome el aliento necesario.

A mis amigos Francisco y Edgar, quienes nunca me dejaron solo y me ayudaron cada vez que lo necesité.

A Aracely Galicia, por ayudarme a ser una mejor persona y brindarme su incondicional apoyo.

Al Dr. Edmundo Rocha, por la orientación dada cada vez que surgió alguna duda.

A mis sinodales; Rigel Gámez, Noé Martínez, Eduardo Hernández y Nelly Peña; quienes sugirieron oportunas correcciones.

A la Facultad de Ingeniería, que me ha otorgado la oportunidad de formarme como ingeniero y ha sido mi segunda casa.

Al apoyo otorgado por el programa PAPIIT IN1139: "Planeación y control de marcha de un robot bípedo".

Índice de contenido

Índice de figuras

Índice de tablas

I. GLOSARIO

AZR – *Allowable ZMP Region* (región de ZMP permisible)

CoM – Centro de masa

CoP – Centro de presión

DC – *Direct Current* (corriente directa)

FSR – *Force Sensing Resistor* (sensor resistivo de fuerza)

GDL – Grado de libertad

IMU – *Inertial Medition Unit* (unidad de medición inercial)

PID – Proporcional integral derivativo

SS – Soporte simple

SD – Soporte doble

SSI – Soporte simple sobre pie izquierdo

SSD – Soporte simple sobre pie derecho

SDA – Soporte doble ambos pies misma altura

SDD – Soporte doble con el pie derecho enfrente

SDI – Soporte doble con el pie izquierdo enfrente

Xzmp – Coordenada X del ZMP

Yzmp – Coordenada Y del ZMP

ZMP – *Zero Moment Point* (punto de momento cero)

II. RESUMEN

El presente escrito tiene como objetivo principal el desarrollo, implementación y prueba de dos diferentes algoritmos de control durante la ejecución de marcha de un robot bípedo de 12 GDL.

La importancia de esta investigación en el área de la robótica, radica en la aplicación de los criterios de ZMP y los ángulos de la cadera en el desarrollo de una caminata dinámicamente estable en el robot. El controlador consiste en una serie de controles PID que generan una compensación en ángulo para ciertas articulaciones.

Para el desarrollo de esta investigación se utilizaron los datos obtenidos de la planeación de trayectorias de un trabajo previo (Merino Morales, 2016), con los cuales se adquirieron datos experimentales que permitieron establecer un punto de partida con la finalidad de comparar los resultados obtenidos.

A continuación, se desglosan los capítulos del presente documento con la finalidad de introducir al lector en las temáticas a tratar en cada apartado:

Capítulo 1. Introducción

Se describe la arquitectura del robot bípedo y la nomenclatura de los motores y eslabones que se utilizan en el trabajo. Por otro lado, se aborda el punto de partida del proyecto de investigación del robot bípedo para esclarecer la problemática a resolver; posteriormente, se estipulan los objetivos generales y específicos a desarrollar para finalmente, abordar el estado del arte con el propósito de conocer alternativas de vanguardia.

Capítulo 2. Antecedentes

Esta sección consta de dos partes; en la primera se profundizan conceptos necesarios para la comprensión de este documento, entre los cuales se encuentran: IMU, pitch, roll y ZMP. En la segunda parte se mencionan las aportaciones desarrolladas en diferentes trabajos previos de investigación relacionados al proyecto del robot bípedo, fundamentales en el desarrollo del presente escrito. En dichas contribuciones destacan Fernanda Merino Morales y Saddan Hernández.

Capítulo 3. Elementos auxiliares del sistema de lazo cerrado del robot bípedo.

En este apartado se desarrollan los componentes necesarios para la aplicación de un sistema de control con realimentación, desde la creación de la interfaz gráfica y cambio de microcontrolador, hasta la medición de las variables del robot en lazo abierto, con las cuales se elaboró una trayectoria de referencia de control a través de métodos numéricos. Asimismo, se diseñan algoritmos que ayudan a evitar errores observados

en la caminata, como el aterrizaje incorrecto del pie y una baja elevación durante los pasos.

Capítulo 4. Sistema de control de estabilidad de marcha

En este capítulo se define la estructura del controlador con la finalidad de establecer una conexión entre los elementos de control para realizar un sistema de lazo cerrado. Por otro lado, se define la estrategia de control al realizar un análisis de la postura y articulaciones del robot. Posteriormente, se detalla la obtención de las constantes utilizadas en el controlador por medio de un método heurístico.

Capítulo 5. Análisis de resultados

En esta sección, se detallan las pruebas efectuadas y se realizan las observaciones para comprobar y comparar la funcionalidad del sistema de lazo cerrado, tanto de los ángulos de la cadera, como del ZMP.

Capítulo 6. Conclusiones y trabajo a futuro

En esta parte, se da cuenta al lector de las resoluciones finales de cada sistema de control, comparando los objetivos del presente trabajo con los resultados obtenidos. Finalmente, se establece un precedente para futuras investigaciones relacionadas con el robot bípedo.

1. Introducción

1.1. Arquitectura del robot bípedo Lynx Scout

Desarrollado por Lynxmotion y distribuido por Dynamo Electronics, el Robot Lynx Scout® es un bípedo que cuenta con 12 GDL [1], con seis motores en cada una de sus piernas hecho de aluminio negro anodizado, aunque algunos de sus componentes están fabricados con policarbonato lexán mediante corte laser y modelos de inyección de componentes [2].

El bípedo scout cuenta con seis eslabones por pierna, unidas mediante otro eslabón central denominado cadera, estos trece eslabones se relacionan entre sí a través de juntas rotacionales impulsadas por servomotores (véase la figura 1.1).

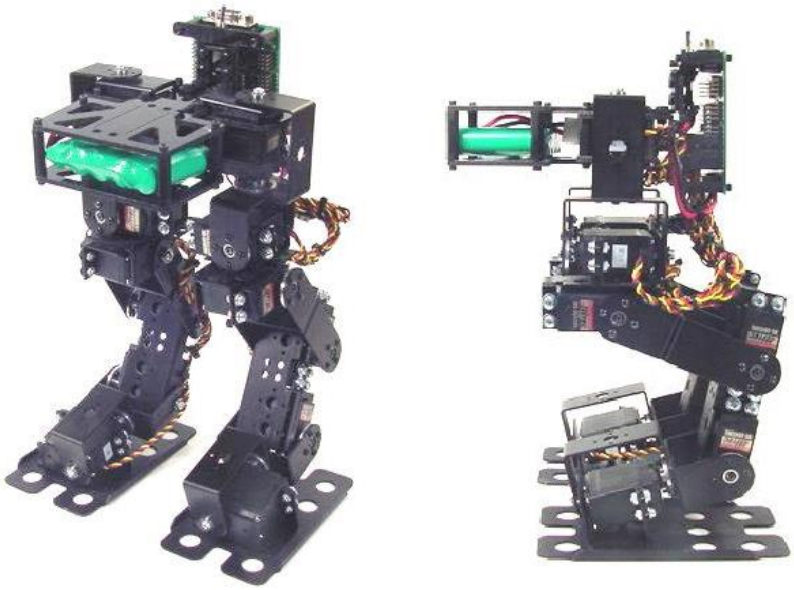


Figura 1.1. Vista isométrica y lateral del robot bípedo Scout [2]

La figura 1.2 muestra un modelo asistido por computadora del robot. En él se pueden identificar a los servomotores en color rojo, la cadera en amarillo, en marrón los tobillos y, finalmente, los eslabones que forman la estructura de la rodilla en azul.

El eslabón de la cadera se nombró con la letra B, los de las piernas se identifican con la nomenclatura “ni”; donde “n” indica el número del eslabón partiendo desde arriba empezando con n=1 y finalizando abajo con n=6, si la pierna es izquierda entonces i=1, en caso contrario i=2.

La rotación de los ejes de los motores se identifica por medio de los ángulos $\theta_{ni,}$ que obedece a la nomenclatura antes descrita.

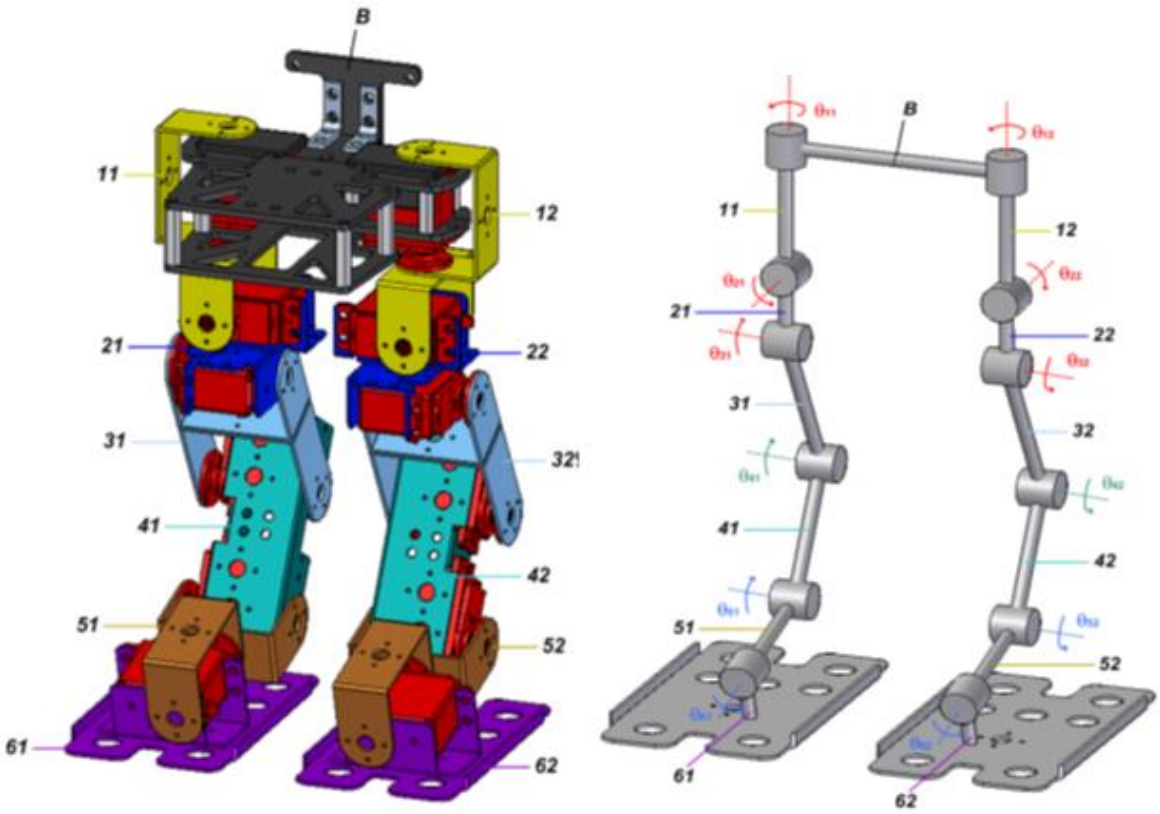


Figura 1.2. Diseño asistido por computadora del robot bípedo y su modelo simplificado [1].

En la tabla A se pueden ver las características físicas principales del robot.

Tabla A. Características técnicas del robot bípedo Scout [2]

Altura	*30 cm*
Peso	*1.1 kg*
Fuente de voltaje	*5 Volts*
Actuadores	*Servomotores DC*
GDL	*12*

1.2. Evolución y estado actual del proyecto

En 2008, el Departamento de Mecatrónica de la Facultad de Ingeniería de la Universidad Nacional Autónoma de México, adquirió un robot bípedo Scout, a partir de entonces se han desarrollado diferentes tesis, proyectos de investigación y artículos utilizándolo como plataforma de aprendizaje.

Asimismo, con sustento en el Programa de Apoyo a Proyectos de Innovación Tecnológica (PAPIIT), “Planeación de trayectorias y control de marcha de un robot bípedo de 12 grados de libertad”, se ha logrado el desarrollo de modelos cinemáticos, dinámicos [1], así como de trayectorias de marcha desde diferentes enfoques del robot bípedo Scout.

Los avances en el proyecto han permitido la generación de trayectorias de marcha del robot bípedo a partir de un análisis de la cinemática inversa en el programa Mathematica [3], esos datos se guardaron en un archivo separado por tabulaciones y, posteriormente, se hizo posible su envío a un microcontrolador por medio de una interfaz gráfica.

Por otro lado, también se ha hecho posible el desarrollo de una caminata estable del robot sin realimentación en una superficie completamente horizontal y sin perturbaciones. Se han implementado nuevos actuadores y con mejores características tales como la cantidad de carga que soportan y el material de sus engranes para generar el par necesario para desplazar peso adicional o desarrollar elementos de control [4].

Se aumentó el número de los sensores de fuerza en la planta de los pies con el fin de calcular variables de control con mayor precisión. Igualmente, se implementó una nueva unidad de medición inercial para obtener lecturas más precisas y de manera más rápida, con el fin de comparar los parámetros obtenidos con los teóricos (o proyectados) y evaluar la caminata del robot [4].

Además, con ayuda de alumnos de estancias académicas se desarrolló un algoritmo para medir el ZMP en cada pie, de manera que no se consuman demasiados recursos en el microcontrolador y puedan reservarse para cualquier otra actividad deseada.

Recientemente se ha trabajado en un proyecto de controlador difuso que mantiene la cadera en posición horizontal cuando la postura del robot es estática o durante la marcha [5].

1.3. Planteamiento del problema

En un entorno real donde se planea que el robot camine, por ejemplo, sobre una superficie irregular o con cambios de pendiente, no se cumplen las condiciones ideales para realizar una caminata sin realimentación; por lo que resulta indispensable diseñar y desarrollar los elementos de un sistema de control con el fin de aumentar la robustez del sistema y así garantizar la estabilidad de la caminata.

El sistema de realimentación puede ser diseñado a través de estrategias de control basadas en las variables medidas a través de los sensores: el ZMP y la posición angular de la cadera. Dicho lo anterior, es necesario implementar ambas estrategias de control y comparar la robustez obtenida en diferentes condiciones que afecten la estabilidad del robot.

De igual manera, al ejecutarse una caminata por el robot bípedo Scout, no se obtiene una marcha fluida y semejante a la del ser humano, debido principalmente a que la comunicación entre la interfaz gráfica y el microcontrolador no ha permitido ejecutar los ángulos de manera más rápida, característica indispensable al implementar un sistema realimentado.

1.4. Objetivos generales

El presente trabajo se enfocará principalmente en:

- Diseñar un sistema de adquisición de datos de los sensores que funcione mientras se desarrolla la caminata del robot
- Aumentar la velocidad de transmisión de los ángulos de cada servo
- Mejorar la capacidad del robot bípedo de caminar ante perturbaciones

- Comparar los criterios de ZMP y ángulos de cadera
- Establecer un punto de partida para la aplicación de principios de control más avanzados que permitan una reacción más rápida y amortiguada, lo que se verá reflejado en una caminata más estable

1.5. Objetivos específicos

De manera más detallada y con el fin de delimitar lo que el presente trabajo busca desarrollar, los objetivos específicos son:

- Sustituir el microcontrolador actual
- Desarrollar una nueva interfaz de envío y adquisición de datos provenientes del robot: ZMP y ángulos de la cadera
- Desarrollar un sistema que calcule variables de control y corrija la postura de acuerdo a un valor deseado en diferentes posiciones estáticas
- Crear un sistema de referencia a seguir a lo largo de la trayectoria del robot bípedo
- Mantener la cadera del robot de manera horizontal durante la caminata, mediante un sistema de control y probar dicha condición como criterio de estabilidad
- Implementar un sistema realimentado utilizando el ZMP como parámetro de control y comprobar su funcionamiento

1.6. Estado del arte

1.6.1. Robots bípedos

En los últimos años los avances en tecnología han cambiado radicalmente nuestra forma de vida y los inventos actuales han facilitado las actividades diarias. Los llamados smartphones ahora nos permiten hacer una videollamada con cualquier persona al otro lado del mundo; algo que antes sólo se podía ver en una película de ficción, hoy es una realidad. Del mismo modo, el desarrollo en nuevos microcontroladores, tecnologías de procesamiento de datos y descubrimiento de materiales, han permitido que las investigaciones en robótica obtengan mejores resultados.

En los primeros años del desarrollo de la robótica, su principal objetivo fue el diseño y construcción de robots industriales, aquellos cuya función era optimizar procesos automáticos en líneas de producción. Sin embargo, recientemente se ha notado un creciente interés en los llamados robots de servicio (véase figura 1.3). Debido a que son los encargados de ayudar a los seres humanos, por lo general mediante la realización de un trabajo sucio, opaco, distante, peligroso o repetitivo, como las tareas domésticas. Por lo general son autónomos y/o gestionados por un sistema de control integrado en él mismo, con opciones de anulación manuales. El término "robot de servicio" no está completamente definido a la fecha. La Federación Internacional de Robótica (IFR) ha propuesto una definición tentativa: "un robot de servicio es un robot que opera semi o totalmente autónomo para realizar servicios útiles para el bienestar de los seres humanos y equipos, con exclusión de las operaciones de fabricación” [6].

En los últimos años, se ha buscado que los robots de servicio tengan una mayor semejanza con los seres humanos, por lo que se ha pretendido la implementación de dos extremidades superiores para interactuar con su entorno y dos piernas para desplazarse. El interés de que un

robot realice ambas actividades representa un verdadero reto, en el cual diferentes investigaciones científicas han reunido sus esfuerzos y logrado excelentes resultados.

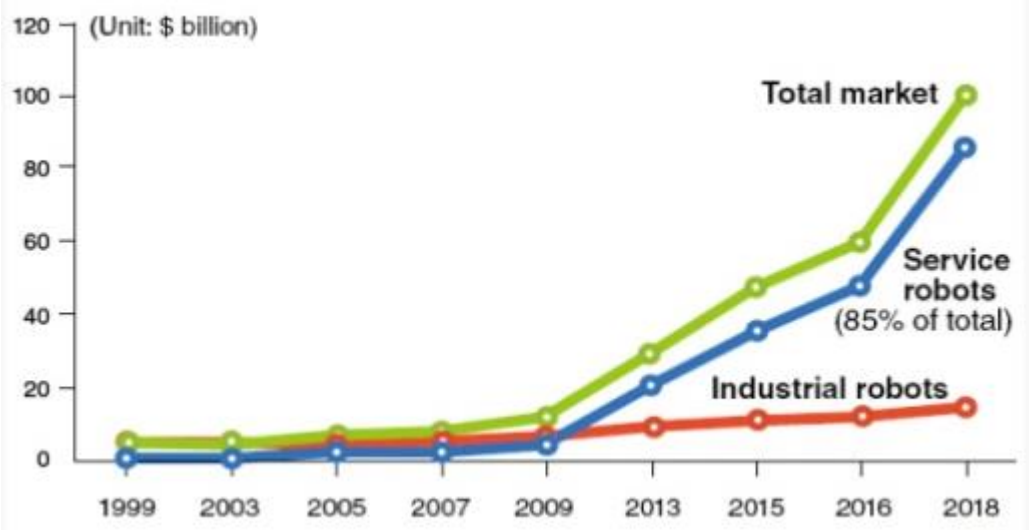


Figura 1.3. Demanda total de robots en el mundo, industriales y de servicio [7]

Es importante mencionar que el desplazamiento de los robots de servicio varía desde robots que tienen múltiples extremidades hasta los que usan ruedas para su movimiento; este trabajo, sin embargo, se centra en los robots bípedos con sistemas de control de marcha; me refiero, por ejemplo, a los robots Nao, ASIMO y Atlas.

1.6.1.1. Robot Nao

Nao es un robot humanoide desarrollado por la empresa Softbank Robotics. Es totalmente programable e interacciona de manera natural, habla, ve y escucha; además, puede ser utilizado para promocionar algún producto en un evento y hasta para realizar coreografías [8]. Sus 25 GDL le permiten caminar y levantarse en caso de caída, dar media vuelta e incluso patear un balón (véase figura 1.4).

A la fecha, el robot Nao ha ganado gran popularidad en el mundo de la robótica, desde el 15 de agosto de 2007 cuando sustituyó al perro robot Aibo de Sony como el prototipo Standar de la Robocup en la *Robot Soccer World Cup* [10]. Muestra de ello lo encontramos en la figura 1.5 donde es posible observar un partido de la Standar Platform League en la Robocup de Nagoya, 2017. Por su parte, la Universidad de Tokio y los laboratorios de Harvard y Stanford en Estados Unidos, han adquirido más de 200 robots Nao para llevar a cabo diferentes investigaciones. Las principales características del robot Nao pueden ser apreciadas en la tabla B.

Figura 1.4. Robot Nao pateando un balón [9]

Figura 1.5. Partido de la Standard Platform League en la Robocup de Nagoya, 2017.

Tabla B. Características técnicas del Robot Nao [11]

Altura	***58 cm***
Peso	*5.4 kg*
Carga extra	*Hasta 600 g*
Fuente de voltaje	*Bateria*
Actuadores	*Brush corriente directa*
GDL	*25*
Sensores	*2 Cámaras, 4 micrófonos, 9 sensores táctiles, sensor inercial, telémetro, 2 sensores ultrasónicos y 8 sensores de presión.*

Debido a la popularidad de Nao, existe una gran cantidad de investigaciones sobre él, entre ellos destaca el trabajo realizado por Juan José Alcaraz Jiménez, *Control Retroalimentado Robusto de Marcha Bípeda Omnidireccional en el Robot Humanoide Nao* [12]. En él se habla sobre el modelo parametrizado mesa-carro, el uso de los ángulos de la cadera, el centro de masa y el ZMP como criterio de estabilidad al momento del diseño de la caminata. Además, se menciona un mecanismo para asegurar el contacto de la planta de pie a través del uso de los sensores de fuerza, esto para realizar una correcta medición del ZMP.

En el documento se establece el uso de criterios de un sistema de control de balance y estabilidad de la marcha del robot, como el ZMP y los ángulos de la cadera que, por medio de la cinemática inversa, promueven una caminata más estable [12].

1.6.1.2. Robot ASIMO

En 1946 el ingeniero Sōichirō Honda funda en Hamamatsu, Japón, el Instituto Honda de Investigación Técnica, en el que se dedicaba a adaptar e instalar en bicicletas pequeños motores utilizados en la Segunda Guerra Mundial [13].

A partir de entonces, Honda no sólo se dedicó al desarrollo de motores para bicicleta, sino también de automóviles, podadoras, motocicletas, además de participar en el apoyo a la investigación y el desarrollo de nuevas tecnologías [14]. Pero la inmersión de Honda en el área de los robots humanoides comienza en 1986. Su meta a cumplir era desarrollar robots que pudieran coexistir y colaborar con los seres humanos, con el fin de sumar valor a la sociedad [15]. Ese mismo año se inició el desarrollo de la colección *Series E*, prototipos bípedos experimentales y de investigación.

En la tabla C se pueden apreciar la evolución física y tecnológica de los robots que fueron parte de la *Series E*.

Tabla C. Evolución de robots de Series E [16]

Modelo	Año	Peso (kg)	Altura (cm)	GDL	Avance tecnológico
E0	1986	16.5	101.3	6	Primer robot, caminaba en línea recta, 5 segundos cada paso
E1	1987	72	128.8	12	Caminaba a 0.25 km/h, primero en tener 12 GDL
E2	1989	67.7	132	12	Alcanzaba los 1.2 km/h, se aproximaba más a imitar la caminata humana
E3	1991	86	136.3	12	Podía caminara a 3 km/h la velocidad de una caminata humana promedio
E4	1991	150	159.5	12	Se alargó la rodilla 40 cm para alcanzar una velocidad de 4.7 km/h
E5	1992	150	170	12	Primer modelo de locomoción autónoma
E6	1993	150	174.3	12	Control autónomo de balanceo mientras camina por escaleras, pendientes o al pisar un obstáculo

Su evolución estética se puede ver en la figura 1.6.

Figura 1.6. Evolución estética de Series E [17]

Después del prototipo E6 comenzó el desarrollo de la serie P, un nuevo paso en la historia de los robots bípedos. Con 1.9 metros de altura y un peso de 175 kg, P1 fue el primer prototipo humanoide de Honda que tenía tronco, brazos y cabeza, lo cual le permitía manipular objetos, tomar manijas y accionar interruptores. Su investigación empezó en 1993 y finalizó en 1997.

Figura 1.7. Evolución estética de Series P [17]

En 1996, Honda anunció el primer robot humanoide inalámbrico de nombre P2, el cual sustituyó a P1 al tener un aspecto mucho más atractivo y ser más pequeño con 180 cm de altura. P2 era capaz de mover objetos de varios kilogramos y de empujar un pequeño carro mientras caminaba. Un año después, al buscar un robot más ligero y pequeño, se desarrolló el P3 con 160 cm de altura y proporciones más aproximadas a las humanas [15]. Como se puede apreciar

en la figura 1.7 Los robots de la colección *Series P* fueron cada vez más pequeños y parecidos a un ser humano.

El siguiente objetivo de Honda fue desarrollar un robot con la capacidad de participar en un entorno humano y ser más realista. Como resultado, en el año 2000 se anunció el robot ASIMO (Advanced Step in Innovative MObility); éste tuvo menores dimensiones que su antecesor (véase figura 1.8) y su tecnología le permitió tener una caminata más suave. En su momento, ASIMO fue considerado el robot bípedo más avanzado [18].

En noviembre de 2011, Honda agrupó un equipo de investigación y desarrollo, bajo el nombre de Honda Robotics, y presentó una nueva actualización de ASIMO [13] con nuevas características (véase tabla D).

Tabla D. Características técnicas de robot ASIMO [19]

Altura	***1.3 metros***
Peso	*50 kg*
Carga extra	*Hasta 1 kg, 10 kg empujando un carro*
Fuente de voltaje	*Batería recargable (Li-ION) 51.8 V*
Actuadores	*Servomotores con reductor armónico de velocidad y driver.*
GDL	*57 en total, 6 en cada pierna.*
Sensores	*Sensor de área del pie, giroscopio y acelerómetro, 3 sensores ultrasónicos, 2 cámaras y sensor laser.*

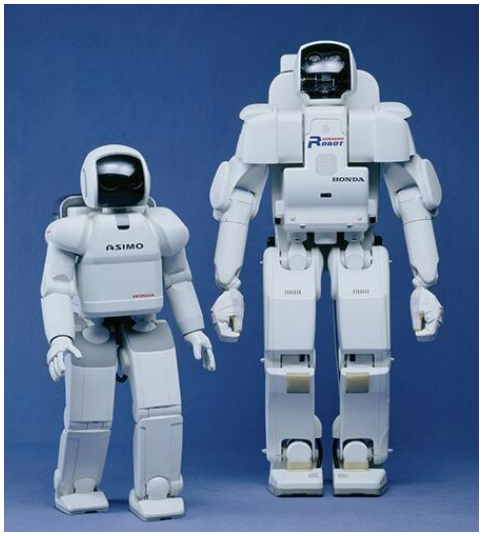


Figura 1.8. Robot ASIMO a la izquierda y su antecesor P3 a la derecha [20]

En la actualidad, ASIMO es capaz de caminar, correr a 6 km/h o en reversa, saltar en una o dos piernas, mantener el equilibrio en superficies irregulares, bailar, subir escaleras, manipular

objetos, empujar carros de supermercado, reconocer rostros de personas y sus voces, incluso, cuando hablan al mismo tiempo [21].

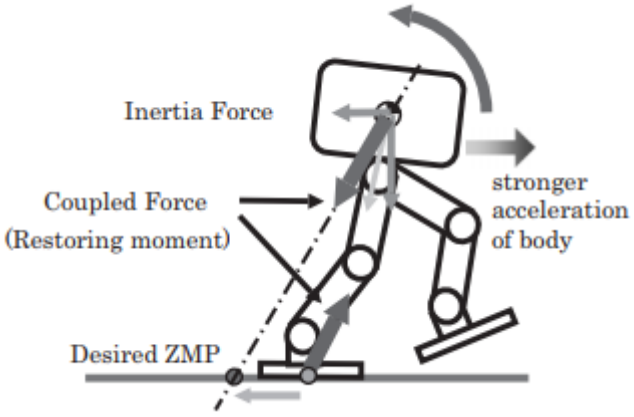


Figura 1.9. Aplicación de un ZMP deseado como parámetro de control en ASIMO [15]

ASIMO es la culminación de dos décadas de investigación en robótica humanoide que no sólo ha incentivado a jóvenes a estudiar ciencias y matemáticas [16], también se ha utilizado como modelo experimental. De igual manera, ha motivado importantes investigaciones sobre locomoción bípeda, tal es el caso del trabajo realizado por Toru Takenaka, *The control system for the Honda humanid robot* [15], publicado por la Universidad de Oxford en 2006, en el que se habla sobre la estabilidad de la marcha del robot por medio del ZMP y el Centro de Presión (CoP). Takenaka trata también el tema del aterrizaje del pie y menciona un ZMP deseado como referencia que, al igualarse, asegura una caminata estable (véase figura 1.9).

1.6.1.3. Robot Atlas

La empresa estadounidense Boston Dynamics ha realizado en los últimos años avances importantes en el desarrollo de los robots de servicio con sistemas de control de posición y orientación complejos. Los proyectos desarrollados por esta empresa se han convertido en una referencia para casi cualquier tipo de robot, desde los móviles que utilizan llantas para desplazarse, hasta los cuadrúpedos.

Uno de los proyectos más destacados en el área de la locomoción bípeda de Boston Dynamics es el robot Atlas, concebido como el último de la línea de robots humanoides que puede coordinar los movimientos de los brazos, el torso y las piernas para lograr el control móvil de todo el cuerpo [22]. Incluso tiene la capacidad de *"…manipular objetos en su entorno y viajar en terrenos difíciles…"* según la propia descripción de Boston Dynamics [22].

La estructura de Atlas aprovecha la impresión 3D para hacerlo compacto con una alta relación resistencia/peso. Las cualidades más importantes del robot Atlas son que puede mantener su equilibrio aun cuando es empujado o empuja algún objeto y que, en caso de caída, puede levantarse. Sus características pueden advertirse en la tabla E.

Tabla E. Características técnicas de robot Atlas [22]

Altura	*1.5 metros*
Peso	*75 kg*
Carga extra	*Hasta 11 kg*
Fuente de voltaje	*Batería*
Actuadores	*Hidráulicos*
GDL	*28*
Sensores	*LiDAR y visión estéreo*

El robot Atlas, además, es capaz de realizar saltos de hasta 1.2 metros y giros de 180 grados, manipular objetos, bajar escaleras (véase figura 1.10) y hacer backflips en el aire, según puede constatarse en un video subido a la plataforma YouTube en noviembre de 2017 [23].

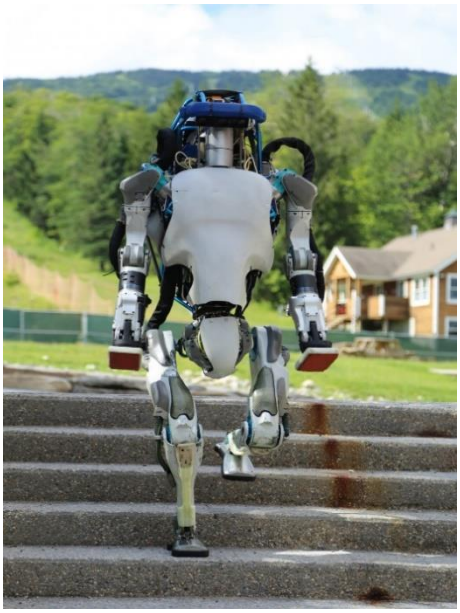


Figura 1.10. Robot Atlas bajando escaleras [22]

Un artículo publicado en 2016 propone un sistema de control, para el robot Atlas, basado en el modelado de su cinemática inversa y dinámica inversa, utilizando controladores de alto nivel y usando como parámetros el CoP y el Centro de Masa (CoM) [24].

1.6.1.4. Robot HUBO

El Instituto Avanzado de Ciencia y Tecnología de Corea del Sur (KAIST, por sus siglas en inglés) empezó la investigación de la robótica bípeda en el año 2000, liderados por el profesor Oh Jun-ho, con la meta de desarrollar un robot humanoide que pudiera vivir con los seres humanos y ayudarlos [25].

Ese mismo año, se diseñó el KHR-0, un robot bípedo sin torso utilizado como plataforma experimental para la determinación de patrones básicos de caminata y el entendimiento de la locomoción bípeda [26].

Basado en el prototipo KHR-0, en 2002 se desarrolló el KHR-1 (véase figura 1.1), un primer robot humanoide de KAIST cuyas características eran su capacidad de caminar, girar y balancearse por sí mismo, el uso de sensores de fuerza e inclinación y su carencia de carcasa, manos y cabeza. Su configuración fue muy semejante a la usada en las posteriores versiones, excluyendo al sistema operativo [26].

El prototipo KHR-2, desarrollado en 2003, se caracterizó por la implementación de sensores, batería, conexión inalámbrica, carcasa, manos y cabeza (véase figura 1.12). Utilizaba el sistema operativo Windows XP con RTX para un control en tiempo real [26].

El progreso en las investigaciones en el KAIST, dio como resultado en 2004 el tercer prototipo de la serie de robots KHR. HUBO, el nombre de la versión KHR-3, contaba con una mayor rigidez mecánica y mejor capacidad de reducción en el engranaje de las articulaciones. Podía bailar, mover los dedos, subir escaleras y poseía una apariencia de un peleador de tae kwon do (véase figura 1.13). Actualmente, HUBO es el robot más famoso de Corea del Sur [26].

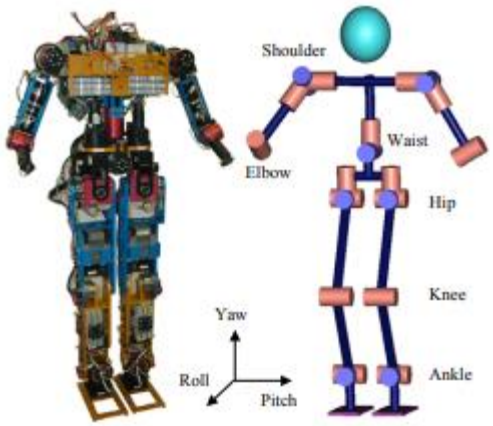


Figura 1.11. Fotografía y estructura de KHR-1 [27]

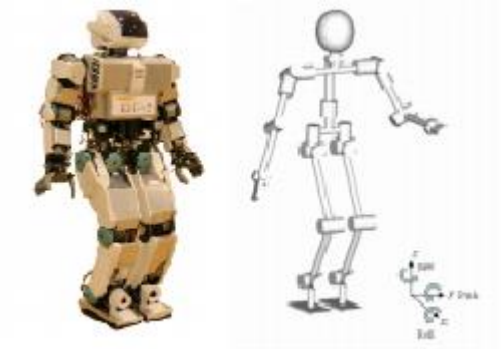


Figura 1.12. Fotografía y estructura de KHR-2 [28]

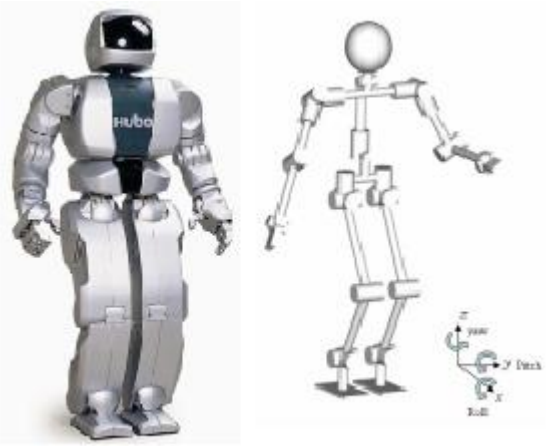


Figura 1.13. Fotografía y estructura de HUBO [29]

En 2005, con la ayuda de Hanson Robotics, se creó una versión del modelo KHR-3, denominada Albert HUBO el cual, además de contar con materiales especiales que asemejaban una piel artificial, podía caminar y generar una gama completa de expresiones faciales, como risa, tristeza, sorpresa, enojo, etc., con la ayuda de 28 motores [26] (véase figura 1.14).

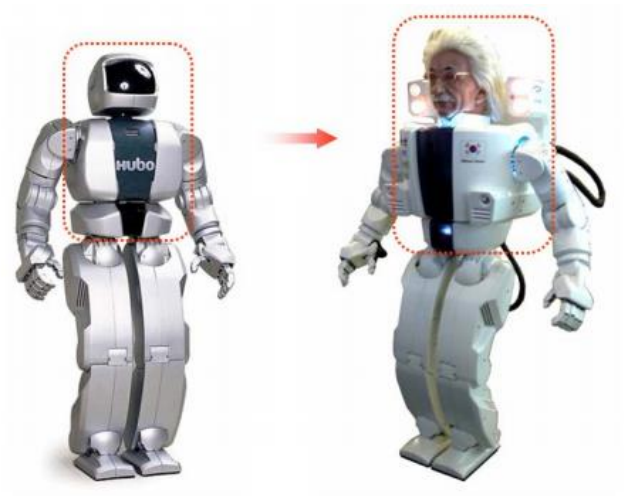


Figura 1.14. Modificaciones de KHR-3 para Albert HUBO [30]

En 2009 se desarrolló una siguiente versión del robot HUBO, el prototipo KHR-4, mejor conocido como HUBO 2 (véase figura 1.15), este tenía como propósito ser el robot más ligero de tamaño humano en el mundo, para lo cual se cambiaron los materiales y la configuración de los brazos, haciéndolo más ligero que sus antecesores y permitiéndole correr a 3.6 km/h [26].

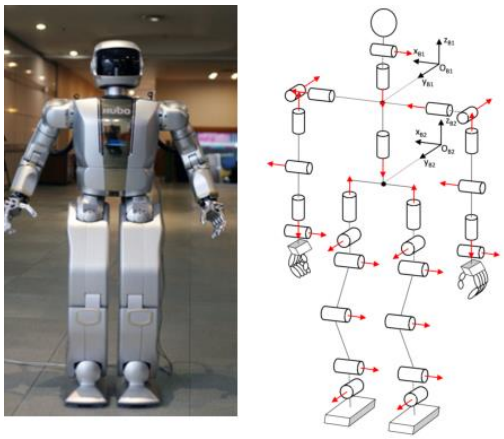


Figura 1.15. Fotografía y estructura de HUBO2 [31]

Finalmente, en 2015 se reveló la última versión del robot HUBO, la cual ganó la edición del mismo año del DARPA Robotics Challenge en Pomona, California. El DRC- HUBO+ cuenta con mayor estabilidad y velocidad, al implementar un sistema adaptable de ruedas.

Al analizar el terreno a través de un sistema de sensores e inteligencia artificial, DRC- HUBO+ decide que método de desplazamiento es más eficiente. Durante la competencia fue capaz de conducir un vehículo, salir del mismo, girar una manija y abrir una puerta o una válvula giratoria (véase figura 1.16), insertar un enchufe y superar un terreno accidentado [32].

Figura 1.16. DRC-HUBO+ en DARPA Robotics Challenge [32]

Las características de la evolución de la Serie de los robots KHR desde el modelo KHR-0 hasta DRC-HUBO+ puede apreciarse en la siguiente tabla:

Tabla F. Evolución de robots de serie KHR [26] [33]

	KHR-0	KHR-1	KHR-2	HUBO	Albert HUBO	HUBO 2	DRC-HUBO+
Peso (kg)	29	48	56	55	57	45	80
Altura (cm)	110	119	120	125	125	125	175
Velocidad de marcha (km/h)	-	-	1.0	1.25	1.25	1.5	1.8
GDL	-	21	41	41	66	40	32

A pesar de que ASIMO, Nao, Atlas y HUBO son robots bípedos, difieren en medidas, actuadores, sensores y algoritmos de control. Las investigaciones que se han realizado en cada uno de ellos han sido, sin embargo, un punto de partida para poder desarrollar el presente trabajo.

En el caso del robot Nao, el método para la generación de sus trayectorias es el mismo que se utilizó en el trabajo de investigación del robot bípedo Scout, además de que cuentan con dimensiones similares.

Por otro lado, ASIMO emplea un sistema de aterrizaje de la planta del pie para la implementación de un sistema de control que usa como parámetro el ZMP, un método con características similares al utilizado para el desarrollo del cumplimiento de los objetivos de esta tesis.

Por su parte, Atlas representa uno de los robots con mejor caminata bípeda dinámicamente hablando, lo cual lo vuelve un referente para el desarrollo, creación y mejora de cualquier sistema de control de los robots bípedos.

Finalmente HUBO, al ser un proyecto desarrollado por el KAIST, cuenta con documentación de fácil acceso sobre la tecnología empleada en su sistema de control, así como los métodos de generación de sus trayectorias.

A diferencia de los cuatro robots anteriormente mencionados, que utilizan el movimiento del torso como acción de control, el robot bípedo Scout al no contar con éste, complejiza el desarrollo de un algoritmo de estabilización de marcha, para el cual utilizará únicamente sus 12 GDL.

1.6.2. ZMP como parámetro de control

El primero en utilizar el ZMP como parámetro de control de caminata fue Miomir Vukobratović en su trabajo *On The Stability of Anthropomorphic Systems* [34], donde analiza la obtención del ZMP por medio de sensores (véase figura 1.17) y su importancia en el diseño de la caminata de los robots bípedos.

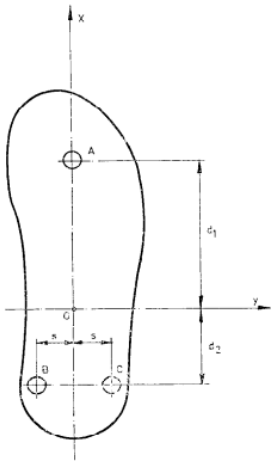


Figura 1.17. Posición de componentes en medición del ZMP en 1972 [34]

En su obra, Vukobratovic también establece los principios fundamentales para la aplicación del ZMP en la estabilidad durante el desarrollo de la caminata y propone una región de seguridad donde este garantiza la correcta dinámica de los robots bípedos; esta región recibe el nombre de polígono de soporte (véase figura 1.18).

En 1990 Ching-Long Shih publicó un artículo denominado *Trajectory Synthesis and Physical Admissibility for a Biped Robot During the Single-Support Phase* [35], donde hace un análisis sobre el comportamiento del robot durante el soporte simple. Analiza también diferentes trayectorias de ZMP producidas por diferentes parámetros de control, como el centro de gravedad (COG) y la altura de la cadera, reconociendo que, algunas trayectorias, vuelven más estable la caminata que otras (véase figura 1.19).

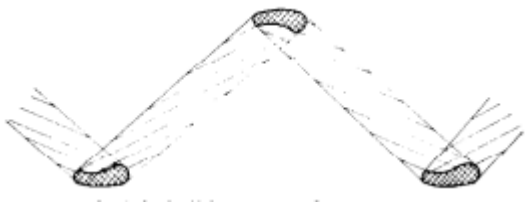


Figura 1.18. Boceto de Vukobratovic sobre polígonos de soporte [34]

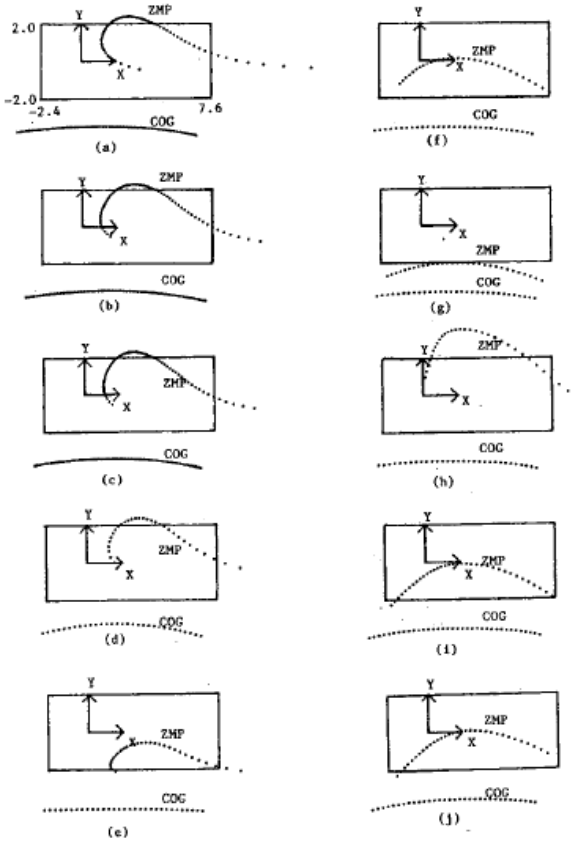


Figura 1.19. Diseño de trayectorias del ZMP durante la fase de soporte simple en 1990 [35]

En 1998 Jong H. Park y Yong K. Rhee propusieron en *ZMP Trajectory Generation for Reducing Trunk Motions of Biped Robots* [36] un método para reducir el movimiento del tronco de un robot bípedo de 7 GDL a través de la generación de trayectorias deseadas de ZMP determinadas por lógica difusa (véase figura 1.20). Con dicha implementación, lograron realizar un control parcial del ZMP a través de la articulación del tronco que modifica únicamente una de sus coordenadas.

Poco tiempo después, un equipo de la Universidad de Tokio propuso en el año 2000, un sistema de control utilizando como base experimental al robot humanoide bípedo h5 [37], que posee un total 30 GDL -6 GDL en cada pierna- y 12 sensores de fuerza en las plantas de sus dos pies para realizar mediciones del ZMP.

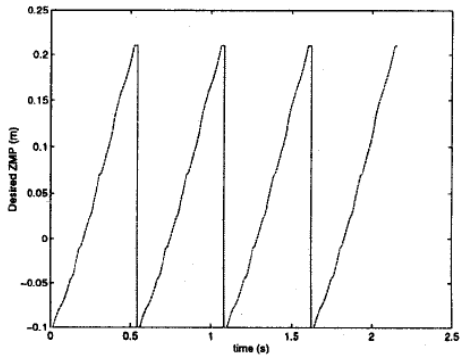


Figura 1.20. Coordenada X del ZMP deseado de un robot de 7 GDL durante la marcha [36]

El sistema de control propuesto por la Universidad de Tokio, empleaba las dos coordenadas del ZMP (X e Y) para realizar una marcha estable utilizando el torso para compensar los errores calculados. En la figura 1.21 se pueden apreciar los resultados obtenidos, a la izquierda el control del ZMP aplicado en X, a la derecha en la coordenada Y.

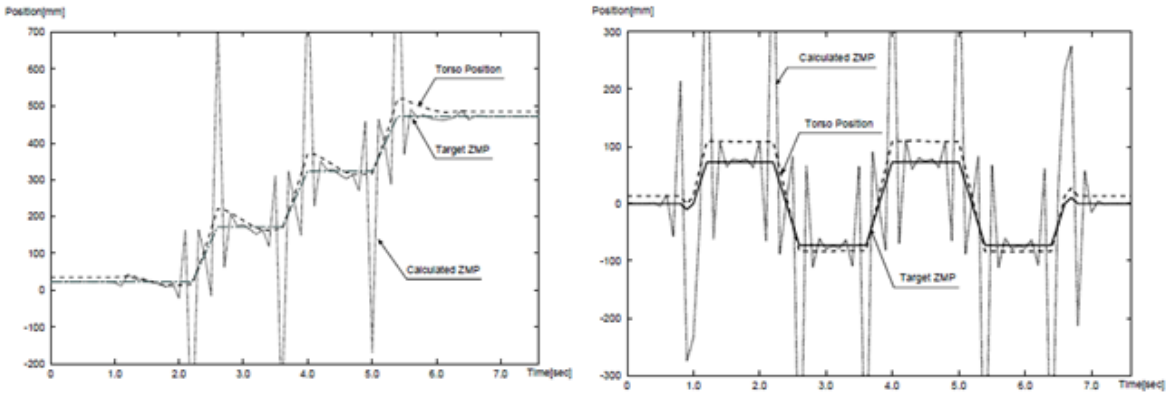


Figura 1.21. Resultado de aplicación de un sistema de control de ZMP en robot h5 [37]

En 2009 Kemalettin Erbatur y Kurtse Okan propusieron en *Natural ZMP Trajectories for Biped Robot Reference Generation* [38], un método de generación de trayectorias de ZMP de referencia que vuelve más natural el movimiento del robot. Ese ZMP de referencia lo crearon a través de series de Fourier y la aplicación de un modelo de péndulo invertido lineal en un robot de 12 GDL, lo que les permitió descubrir que modificando el ZMP de referencia, se tiene un ciclo de caminata hasta 30% más eficiente energéticamente hablando.

Basándose en el trabajo anterior, en 2014 Hyeok-Ki Shin y Byung Kook Kim desarrollaron en *Energy-Efficient Gait Planning and Control for Biped Robots Utilizing the Allowable ZMP Region* [39] un control del ZMP a través de una zona denominada AZR (Allowable ZMP región o, por su traducción, región de ZMP permisible) que en realidad se trata de un área dentro del conjunto de polígonos de soporte que se generan durante la caminata (véase figura 1.22).

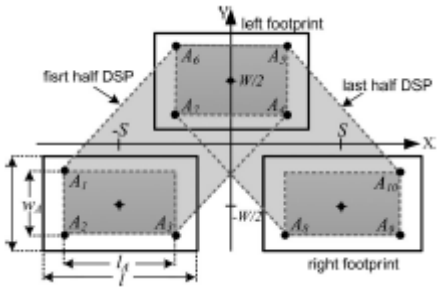


Figura 1.22. Ejemplo de AZR dentro de polígonos de soporte [39]

Para el análisis cinemático implementaron un modelo de péndulo invertido en 3-D del robot bípedo Darwin que cuenta con 20 GDL, de los cuales 12 corresponden a sus piernas.

Para el diseño del ZMP de referencia consideraron el movimiento del centro de masa y se ajustó al AZR, teniendo como resultado una caminata estable con un seguimiento del ZMP medido al deseado donde el ZMP calculado está representado en rojo mientras el deseado se puede apreciar en azul (véase figura 1.23). La aplicación de este sistema de control tuvo como resultado una reducción en el consumo energético de los motores.

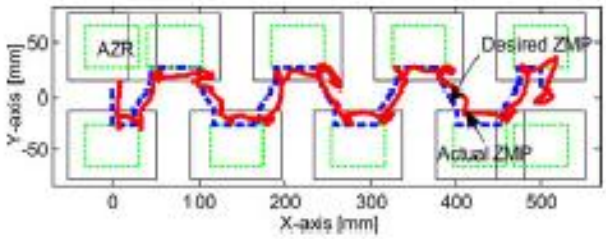


Figura 1.23. Ejemplo de trayectoria ZMP calculada y ZMP de referencia [39]

Recientemente, se encuentra en desarrollo un artículo de nombre *Human Motion Analysis and its Application to Walking Stabilization with COG and ZMP* [40], donde se realiza un ejercicio comparativo entre tres métodos de obtención del ZMP; en el primero se utilizan sensores de fuerza para detectar el ZMP, en el segundo se emplea un método convencional de momento de torsión de un motor y, en el último, se analiza el uso de un observador de torque de reacción (véase figura 1.24). Dentro de los resultados obtenidos en dicho experimento, cabe señalar que se estableció una mejor aproximación mediante el uso de sensores de fuerza.

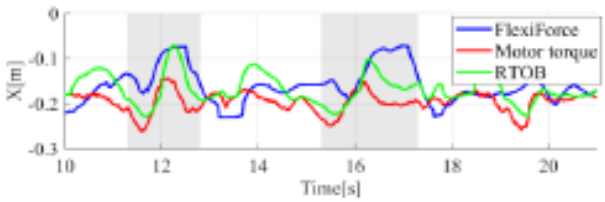


Figura 1.24. Resultado de obtención de ZMP por diferentes métodos [40]

Utilizando como parámetros el ZMP y el centro de gravedad del mecanismo, Seonghye realizó pruebas de control de estabilidad de caminata en un mecanismo de 4 GDL analizando el

comportamiento del robot en una caminata sin perturbaciones. También realizó una prueba añadiendo una carga extra de 3 kg. El control, sin embargo, sólo se efectuó en la coordenada X del ZMP (véase figura 1.25).

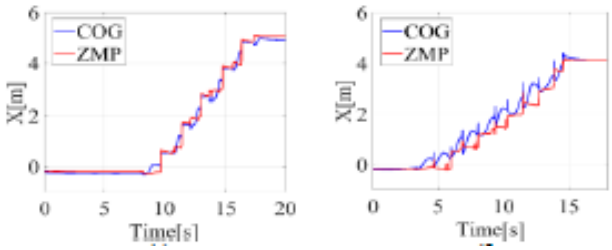


Figura 1.25. Aplicación de sistema de control de ZMP en un mecanismo de 4 GDL [40]

1.6.3. Sensores inerciales en robótica bípeda

Los sensores inerciales son los encargados de medir las componentes de aceleración y velocidades angulares de un sistema, con el fin de analizar su movimiento. A causa de su bajo costo, reducido tamaño y peso, actualmente se pueden encontrar estos sensores en sistemas biomédicos, sistemas de activación de bolsas de aire, monitoreo de máquinas y vibraciones, aplicaciones militares, sistemas autónomos de navegación y guía. Debido a esto, en años recientes la investigación del uso de estos sensores en otros campos de la tecnología se ha ampliado.

En el caso de la robótica bípeda, en 2002 Gienger, M. estableció en su trabajo *Walking control of a biped robot based on inertial measurement* [41] la adquisición de parámetros de control de un robot Johnnie de 17 GDL mediante un sensor inercial y filtros, así como el uso de una estructura de aluminio que evita la interferencia electromagnética (véase figura 1.26).

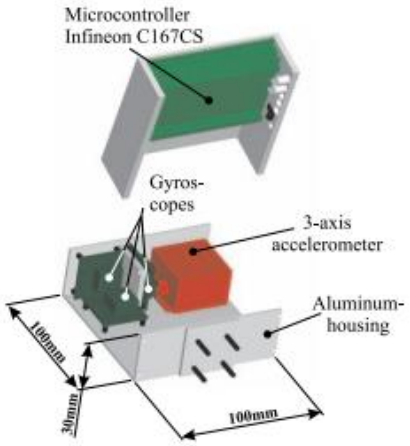


Figura 1.26. Unidad de medición inercial de robot Johnnie [41]

Los avances logrados por Gienger permitieron que Sebastian Loehmeier, en su investigación doctoral de 2010 implementara un sistema de medición inercial de alta precisión construido a partir de giroscopios de fibra óptica y acelerómetros MEMS para estimar la orientación y velocidad de la parte superior del cuerpo del robot bípedo LOLA (robot bípedo de 25 GDL) con respecto a un marco inercial (véase figura 1.27). El sistema se caracterizó por un ruido muy

bajo, insensibilidad a las vibraciones, gran ancho de banda, alta estabilidad y ausencia de sesgo a largo plazo [42].

Figura 1.27. Sistema de medición inercial de robot LOLA [42]

En 2006 se publicó una investigación realizada por el KAIST de nombre *Experimental realization of dynamic walking of the biped humanoid robot KHR-2 using zero moment point feedback and inertial measurement* [43], donde se detalla un sistema de predicción de movimiento utilizando un sensor inercial ubicado en el torso del robot bípedo KHR-2 de 41 GDL, el que permitió conocer la inclinación del robot mientras camina, realizar correcciones en los ángulos de sus motores y evitar su caída (véase figura 1.28). Como resultado, se obtuvo un conjunto de exitosos experimentos de caminata bípeda en el robot, manteniendo los ángulos de inclinación del torso básicamente horizontales (véase figura 1.29).

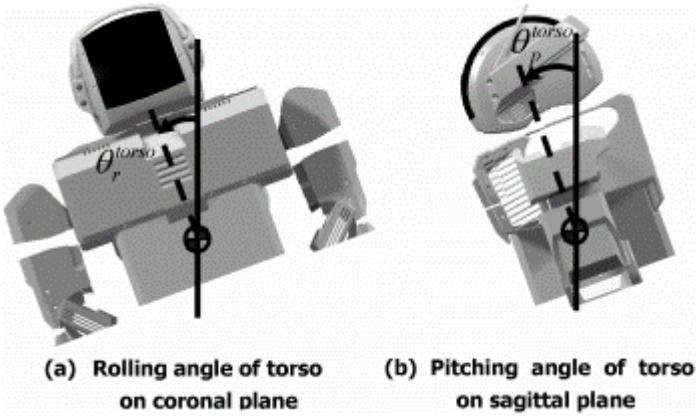


Figura 1.28. Medición inercial en robot KHR-2 [43]

En 2018 se publicó el artículo *Locomotion Control of 7-Linked Walking Robot Embedded with CPG Using Neural Oscillator space* [44], donde se desarrolló un sistema de estabilización de marcha capaz de realizar una caminata similar a la humana en un robot de 7 GDL, utilizando redes neuronales CPG (véase figura 1.30), las cuales tienen una importante participación en la generación y control de locomoción bípeda de sistemas biológicos como los seres humanos y animales.

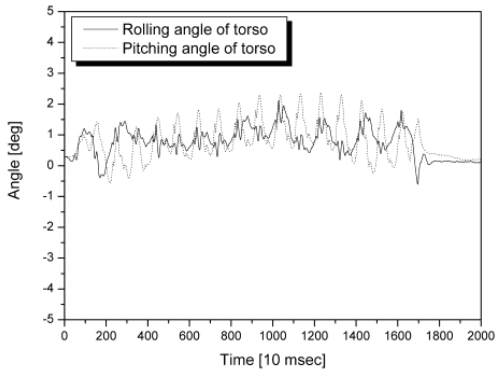


Figura 1.29. Ángulos de torso de KHR-2 durante caminata [43]

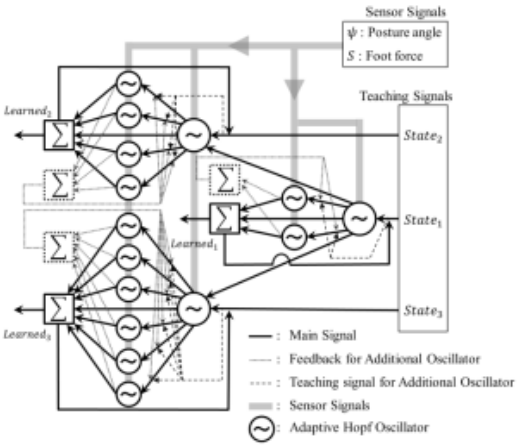


Figura 1.30. Estructura de red de controlador CPG con osciladores [44]

Las redes CPG del robot son alimentadas por los sensores de fuerza en el pie y por un sensor de postura, este último formado por un transductor inercial instalado en un microcontrolador (véase figura 1.31). El principal aporte de esta investigación es el uso de la red neuronal y la medición de la inclinación de la postura como método de aprendizaje del robot.

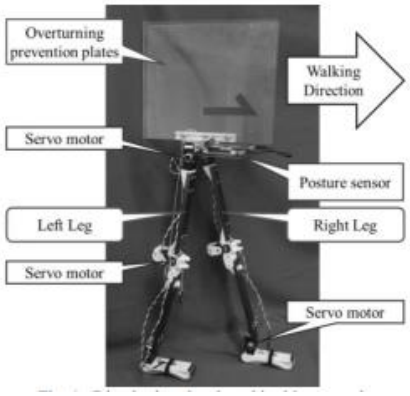


Figura 1.31. Robot de 7 GDL con sensor de postura [44]

2. Antecedentes

2.1. Conceptos básicos

2.1.1. Robótica y locomoción bípeda.

La robótica es una rama de la ingeniería mecatrónica que se encarga del estudio de los robots, así como de su diseño, construcción y operación.

El término robot proviene del vocablo checo *robota* que significa fuerza de trabajo, servidumbre o esclavitud, pero es hasta 1921 cuando el novelista y autor checo Karel Capek acuña el término moderno para nombrar como tal a las criaturas mecánicas protagonistas de su obra teatral R. U. R. (Rosum's Universal Robots) [45].

De conformidad a los fines del presente trabajo, y siguiendo la definición planteada por la Real Academia Española, un robot es una máquina o ingenio electrónico programable, apto para realizar operaciones antes reservadas sólo a los seres humanos. Un robot bípedo es, a su vez, aquél dispositivo mecánico programable capaz de realizar una locomoción bípeda o caminata semejante a la del ser humano, entendiendo por locomoción bípeda, a la capacidad de un sistema para moverse a través del espacio gracias a que posee dos extremidades inferiores que hacen posible su desplazamiento (véase figura 2.1).

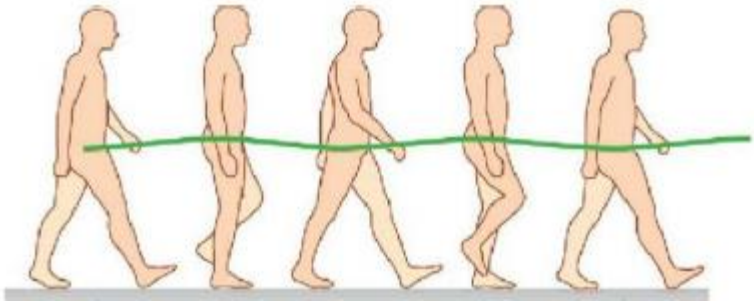


Figura 2.1. Locomoción bípeda en caminata humana [46]

Un ciclo de caminata comprende cuatro fases periódicas [47], a saber:

1. Fase de Soporte Doble (SD): Se caracteriza por el contacto de ambos pies, compartiendo el peso total del cuerpo.
2. Fase de Pre-balanceo: Los dedos del pie anterior del robot tienen contacto con el suelo mientras su talón se encuentra levantado, por lo que aún se considera SD, es la etapa de transición entre soporte doble y soporte simple.
3. Fase de Soporte Simple (SS): Es el término que se utiliza cuando el robot únicamente cuenta con una superficie de apoyo en contacto con el piso. Sucede cuando el robot sólo está apoyado en un pie.
4. Fase Post-balanceo: Se diferencia por ser la etapa de aterrizaje del pie después del SS.

2.1.2. ZMP y polígono de soporte

En física, un momento se define como la cantidad de movimiento angular con respecto a un sistema de referencia [48]; en otras palabras, es una fuerza que produce un giro en torno a un

punto o un eje. El momento se obtiene a partir del producto vectorial de la distancia vectorial respecto al punto de aplicación de la fuerza por el vector de ésta.

Su ecuación está representada de la siguiente manera:

$$\vec{M} = \vec{r}\, x\, \vec{F}$$

Donde *r* es el vector de la posición de la aplicación de la fuerza *F*.

Por ejemplo, en el punto medio de un balancín se genera un momento cuando alguien se sube en aquél, entre mayor sea la masa de la persona, mayor será la fuerza y el momento. Lo mismo sucede con la distancia respecto al punto central, mientras más alejado sea el punto de aplicación de la fuerza, el momento se incrementará. Como se puede advertir en la figura 2.2 la persona de la izquierda genera el mismo momento que la persona de la derecha que tiene menor masa, pero está sentada a una mayor distancia del centro.

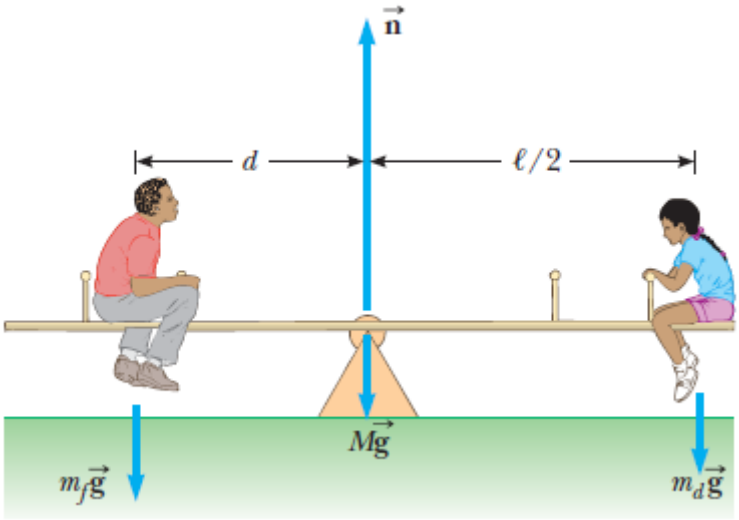


Figura 2.2. Sistema de momentos en equilibrio [49]

El ZMP (Zero-Moment Point) o por su traducción al español, punto de momento cero, se define como el punto del suelo respecto al cual el momento creado por las fuerzas inerciales y gravitacionales no tiene componente en los ejes horizontales [34]. El término fue utilizado por primera vez en enero de 1968 por Miomir Vukobratović en el Tercer Congreso de la Unión de Mecánica Teórica y Aplicada en Moscú. El ZMP puede ser calculado siempre que exista un contacto de un sistema con el piso (véase figura 2.3).

El ZMP al asociarse al CoP de la reacción del piso con el pie [51], se puede definir como el punto donde las fuerzas reactivas en la planta del pie se pueden representar con una resultante en dirección perpendicular al plano de la planta del pie [47]. Por tanto, se puede dar un estimado del ZMP a través de la medición de la fuerza ejercida sobre las plantas de los pies.

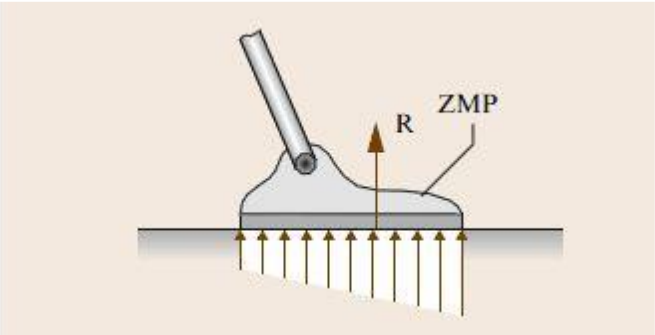


Figura 2.3. Vista lateral de ubicación de ZMP en suela de un robot bípedo [50]

La fuerza resultante en el CoP se obtiene al sumar todas las fuerzas que actúan sobre la planta del pie, por lo que la ecuación asociada a tal fuerza es:

$$F_{RN} = \sum_{i=1}^{n} F_{Ni}$$

Para calcular las coordenadas del CoP, se toman en cuenta los momentos generados por cada una de las fuerzas, por lo que el cálculo del CoP resulta:

$$P_{CoP} = \frac{\sum_{i=1}^{n} F_{Ni} P_i}{\sum_{i=1}^{n} F_{Ni}}$$

Así, la ecuación que representa el ZMP asociado a la planta del pie es:

$$P_{ZMP} = \frac{\sum_{i=1}^{n} F_{Ni} P_i}{F_{RN}}$$

Dónde: *i* es el número de fuerzas reactivas en la planta del pie; P_i es el conjunto de vectores de posición asociados a cada una de las fuerzas; F_{RN} es la fuerza resultante y P_{ZMP} es el vector asociado al ZMP. Esta ecuación será utilizada más adelante para obtener el ZMP del robot bípedo.

El polígono de soporte, por otro lado, se define como el área formada por el límite del conjunto mínimo convexo que contiene un número determinado de puntos de contacto con el suelo [47]. En otras palabras, es la región definida por los límites externos de las superficies de un cuerpo que están en contacto con el piso. En el caso del robot bípedo, si éste se encuentra en SS, el polígono de soporte será la silueta del pie en el piso, por el contrario, si está en SD, éste se formará por los márgenes de ambos pies. En la figura 2.4 se pueden apreciar en gris ejemplos de polígonos de soporte durante la caminata de un sistema bípedo.

La importancia de ambos términos en la investigación de la locomoción bípeda radica en que tienen una estrecha relación con la estabilidad estática y dinámica de los robots bípedos. De esta manera, cualquier robot logrará una marcha dinámicamente estable si el ZMP se encuentra dentro del polígono de soporte en cada momento de la caminata [47].

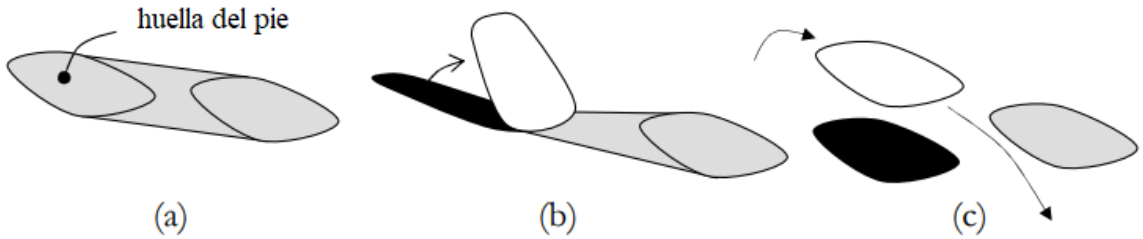


Figura 2.4. Ejemplos de polígonos de soporte [47]

(a) SD (b) SD en pre-balanceo (c) SS

2.1.3. Transductores, sensores y actuadores

Se denomina transductor a todo dispositivo que transforma una señal de una magnitud física en una señal correspondiente, pero de otra magnitud diferente; es decir, un dispositivo que convierte un tipo de energía en otro. No obstante, en la práctica, se consideran transductores generalmente a los que ofrecen o reciben una señal eléctrica. Esto se debe al interés de este tipo de señales en la mayoría de procesos de medida, ya que actualmente los sistemas electrónicos son los principales encargados de controlar procesos y variables [52].

Se puede clasificar a los transductores en dos categorías: de entrada (señal física/señal eléctrica) y salida (señal eléctrica/acción). La tendencia actual, particularmente en robótica, es emplear el término *sensor* para designar el transductor de entrada, y el término *actuador* para designar el transductor de salida. Los primeros pretenden la obtención de información, mientras que los segundos buscan la conversión de energía [52].

Por ejemplo, un micrófono es un sensor acústico que convierte la energía acústica (vibraciones sonoras: oscilaciones en la presión del aire) en energía eléctrica (variaciones de potencial eléctrico). Un altavoz es, por otro lado, un actuador acústico que transforma variaciones de potencial eléctrico en vibraciones sonoras.

2.1.4. Ángulos de Euler

De acuerdo con el teorema de rotación de Euler *"*una rotación arbitraria de un sistema puede ser descrita por solamente tres parámetros" [53], por lo que, un método común para especificar la orientación angular de un sistema de coordenadas respecto a otro es el uso de los tres ángulos de Euler que, mediante giros, definen el nuevo estado de un sistema de coordenadas a otro. La secuencia de rotaciones usada para representar la orientación de un objeto es primero *yaw*, luego *pitch* y finalmente *roll* [1] (véase figura 2.5).

[1] Las interpretaciones al español de *pitch, roll* y *yaw* son: alabeo, cabeceo y guiñada, respectivamente.

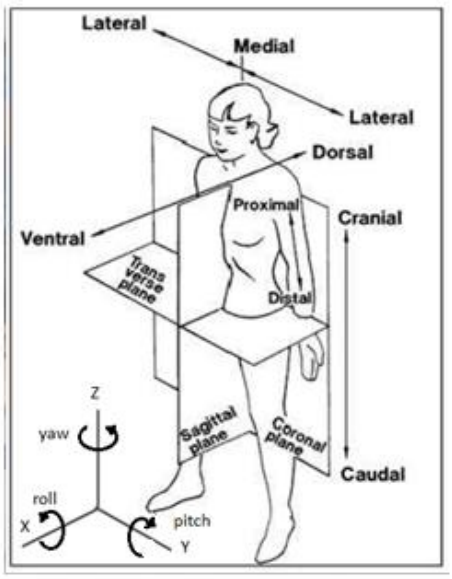


Figura 2.5. Sistema de referencia con las direcciones de giro de los ángulos de Euler[2]

2.1.5. Servomotores

Un servomotor, o simplemente servo, es un dispositivo que tiene un eje de rendimiento controlado. Este puede ser llevado a posiciones angulares específicas al enviar una señal codificada y mientras ésta exista, el servo mantendrá la posición angular del engranaje. Cuando la señal codificada cambia, modificará la posición angular.

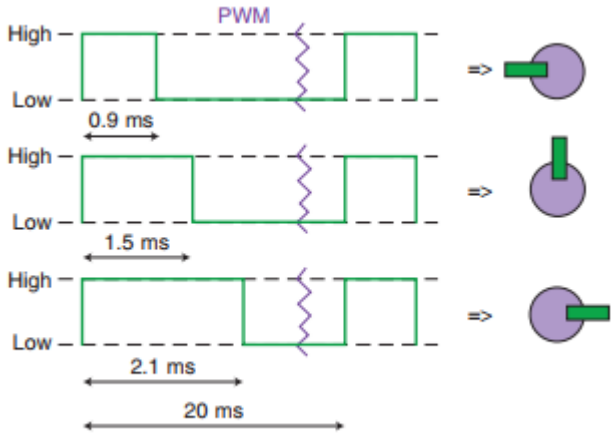


Figura 2.6. Ejemplo de funcionamiento de servo a través de PWM [55]

El motor del servo tiene circuitos de control y un potenciómetro (una resistencia variable) que está conectado al eje central del servomotor y permite al circuito de control supervisar el ángulo actual del servo. Si el eje está en el ángulo correcto, entonces el motor estará fijo, pero si el circuito detecta que el ángulo no es el correcto, el motor girará en la dirección adecuada hasta corregirlo. El eje del servo es capaz de llegar aproximadamente a 180 grados, aunque en

[2] Esquema basado en [85]

algunos casos varía hasta los 210, según el fabricante. En general, un servo normal se usa para controlar un movimiento angular de entre 0 y 180 grados [54]

Comúnmente los servos se controlan por medio de PWM (*Pulse-Width Modulation*), que es la modulación del ancho de pulso de una onda cuadrada que normalmente va de 0 a 5 Volts; al modificar el PWM se cambia la posición del motor. En la figura 2.6 es posible notar que al aumentar el tiempo de la señal PWM en el estado *HIGH* el servo aumenta su giro, para llegar a 180 grados la parte activa del PWM debe durar 2.1 ms.

2.1.6. GDL

Se denominan grados de libertad (GDL) de un sistema, al total de parámetros de entrada que se deben controlar de forma independientemente a fin de llevarlo a una posición en particular [56].

En la figura 2.7 se ejemplifica un sistema biela-manivela de un solo grado de libertad, en el que, para saber la posición total del sistema, sólo se necesita determinar un parámetro; en este caso, el ángulo θ, que representa el giro de la manivela. Conociendo este parámetro se puede deducir la posición de la biela y el pistón.

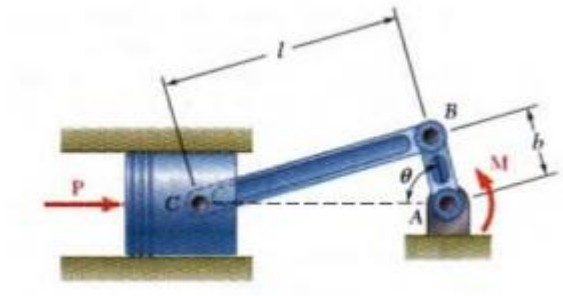


Figura 2.7. Sistema biela-manivela de 1 GDL [48]

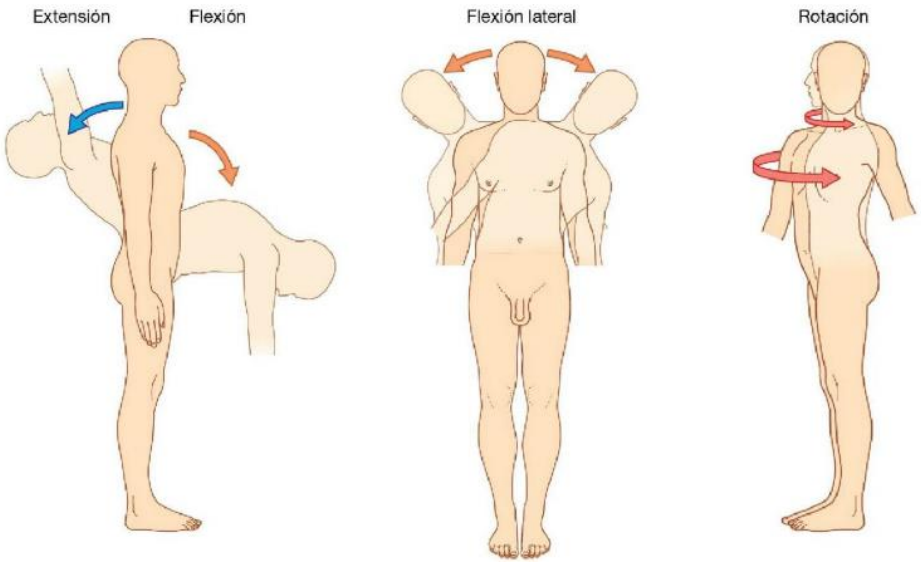


Figura 2.8. Tronco humano como sistema de 3 GDL [46]

Otro ejemplo es el tronco humano, el cual se puede representar como un sistema de 3 GDL de acuerdo con los ángulos de Euler (véase la figura 2.8); en él, uno de los grados de libertad le permite agacharse (*pitch*), otro hace posible una flexión latera (*roll*) y el último es el encargado de su movimiento de rotación (*yaw*).

2.1.7. Microcontrolador

Un controlador es un dispositivo programable que se utiliza para la regulación, supervisión e intervención de procesos mediante señales de entrada, de salida e instrucciones previamente definidas.

Por ejemplo, el controlador de un refrigerador cuenta con un sensor que mide la temperatura frecuentemente y, al ésta exceder los límites predeterminados, hace que el controlador envíe una señal al sistema de enfriamiento con el fin de modificarla y mantenerla dentro de un rango deseado.

Un microcontrolador, a su vez, es un circuito que incorpora los elementos que conforman un controlador con una alta escala de integración, por lo que su tamaño suele representarse con milímetros (véase figura 2.9). Su composición comúnmente es de la siguiente manera:

- Un procesador o unidad central de procesamiento, necesaria para ejecutar las instrucciones
- Memorias RAM y ROM para conservar datos y las instrucciones del programa.
- Líneas de entrada y salida, útiles para interactuar con el exterior (sensores, actuadores, otros controladores o dispositivos electrónicos, internet, etc.)
- Generador de pulsos de reloj, fundamental para sincronizar el funcionamiento del sistema
- Módulos útiles para el control de periféricos como puertos seriales, conversores analógico-digital, generadores de señales PWM, entre muchos otros más que varían dependiendo el dispositivo

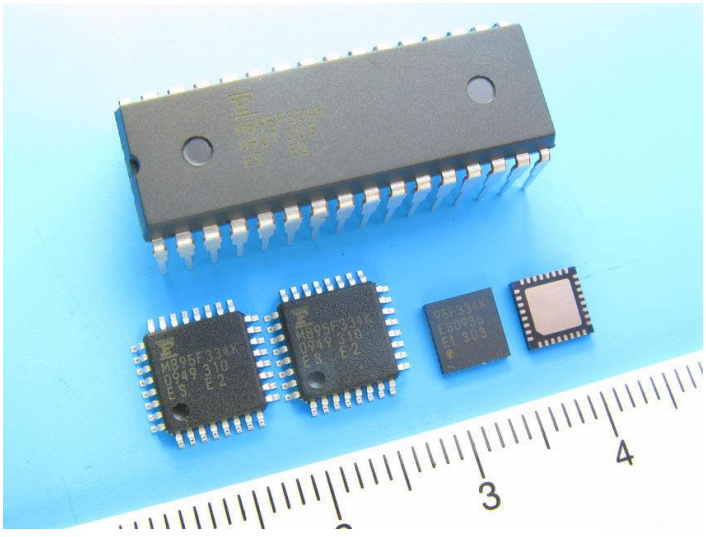


Figura 2.9. Ejemplos de microcontroladores en diferentes presentaciones [57]

Los microcontroladores son, actualmente, los circuitos de control por excelencia, por lo que pueden encontrarse en cualquier dispositivo digital moderno: teclados de computadoras, televisores, teléfonos celulares, hornos de microondas, reproductor de discos Blu-ray, entre otros [58].

Por lo general, la instalación de un microcontrolador es de forma permanente, sin embargo, en ocasiones se necesita probar el funcionamiento de un modelo y realizar varias pruebas en él para modificarlo si es necesario; por este motivo, se diseñó un circuito impreso para el diseño y aplicación rápida de sistemas digitales y analógicos al que se le denominó tarjeta de desarrollo. De inmediato estas se convirtieron en un elemento útil en cuanto al mejoramiento de procesos de diseño al permitir una disminución en el tiempo de validación de prototipos y ofrecer una solución y producto final [59].

Las tarjetas de desarrollo se componen, principalmente, de un microcontrolador, una entrada de diferencia de potencial para su operación y pines I/O con una nomenclatura para su fácil conexión. Además, suelen tener diferentes periféricos que facilitan su uso, tales como botones de reinicio, indicadores LED de funcionamiento y conexión de cable USB para una comunicación serial, entre otros que varían dependiendo de cada tarjeta. La principal desventaja de ellas continúa siendo que, al contener en su estructura un microcontrolador, es de mayores dimensiones que éste, en la figura 2.10, se puede apreciar el ejemplo de una tarjeta de desarrollo con su respectivo microcontrolador.

Figura 2.10. Tarjeta Arduino UNO basado en el microcontrolador ATmega328P [60]

2.2. Trabajo previo

2.2.1. Obtención de ángulos de trayectoria de caminata

La generación de ángulos de trayectoria del robot bípedo Scout se ha trabajado desde diferentes perspectivas. Una de ellas y la más reciente dentro del grupo de trabajo es la presentada en la tesis *Planeación de trayectorias de un robot bípedo con un modelo parametrizado carro mesa* [3] realizada por Fernanda Merino Morales.

Debido a la complejidad del modelo cinemático completo, se optó por utilizar modelos simplificados, por ejemplo, el sistema carro mesa propone ver al robot como un carro de masa

M corriendo en dirección sagital y lateral en una mesa con masa despreciable (véase la figura 2.11). Las ecuaciones resultantes de la aplicación de este método pueden describir la dinámica de cualquiera de los componentes. La idea es que el carro tiene que acelerar a cierta velocidad hacia el extremo exterior de la mesa para evitar que gire respecto a la base. De modo que, la ubicación del ZMP depende de la posición y aceleración del carro [12].

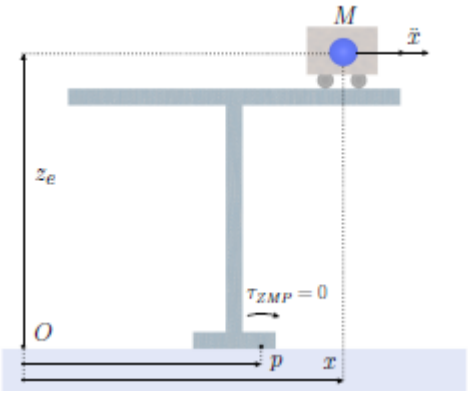


Figura 2.11. Representación de modelo simplificado carro-mesa [12]

La altura del centro de masa (CoM) permanece constante en el modelo antes descrito, pero, si la cadera tuviera la libertad de moverse hacia arriba y hacia abajo, se produciría un movimiento con mayor semejanza a la caminata humana; debido a esta necesidad, Fernanda Merino utilizó el uso de un modelo parametrizado [61] que permite una aceleración en la altura del centro de masa del bípedo Scout como se puede ver en la figura 2.12.

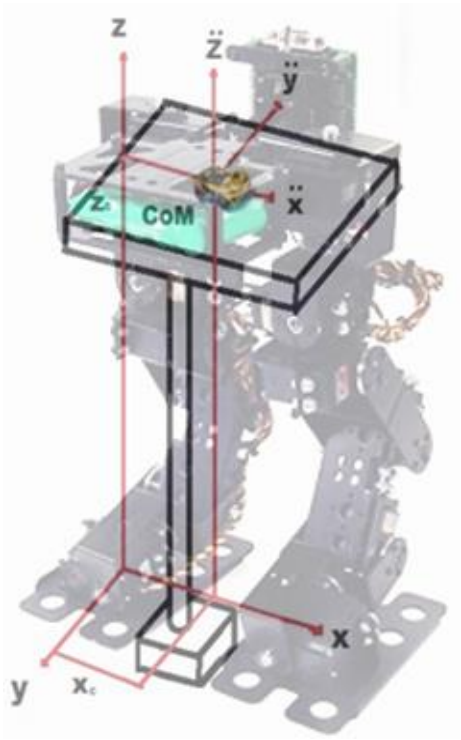


Figura 2.12. Ejemplo de aplicación del sistema mesa-carro en el robot bípedo

El uso del modelo parametrizado mesa-carro ofrece diferentes ventajas que son importantes destacar, por ejemplo: se puede variar la altura del CoM para ampliar y reducir la longitud de paso, según convenga; se reduce el impacto de la planta del pie del robot que podría producir una caída e, incluso, se pueden generar patrones para que logre subir escaleras [3].

Para el diseño de la caminata en la tesis antes mencionada, se propuso una trayectoria en los ejes XY del ZMP del bípedo (véase figura 2.13) que idealmente debe seguir para tener una marcha estable.

Los parámetros teóricos utilizados para la planeación de la trayectoria fueron los siguientes: número de muestras generadas = 1080; longitud de paso = 0.08 [m] y ancho de paso = 0.0598 [m], g= 9.81 m/s^2 y tiempo de muestreo = 5 ms [3].

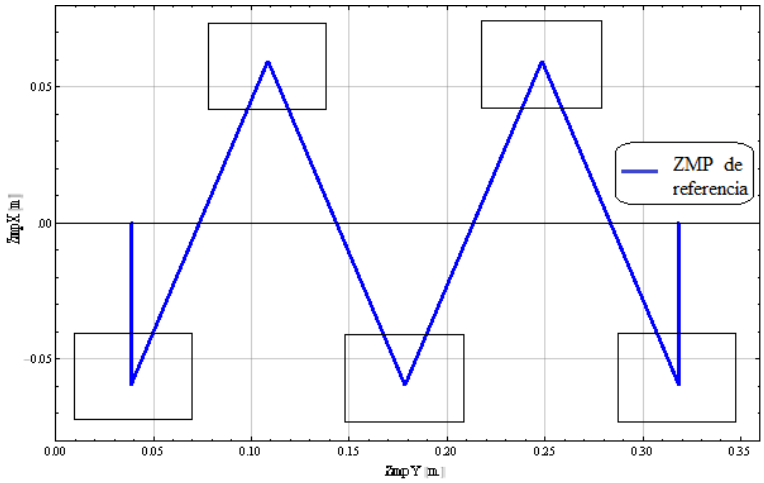


Figura 2.13. ZMP propuesto para diseño de caminata de robot bípedo Scout [3]

La trayectoria que se planeó de esta manera es la posición del centro de masa la cual, en conjunto con las trayectorias de los pies, se introdujo en el modelo cinemático inverso para obtener las trayectorias de los eslabones del robot. Como resultado se obtuvieron los ángulos que cada motor del robot seguirá a lo largo de la caminata, los cuales están representados en las siguientes figuras[3]:

[3] Elaboración propia con base en los resultados obtenidos en el trabajo de experimentación de Fernanda Merino

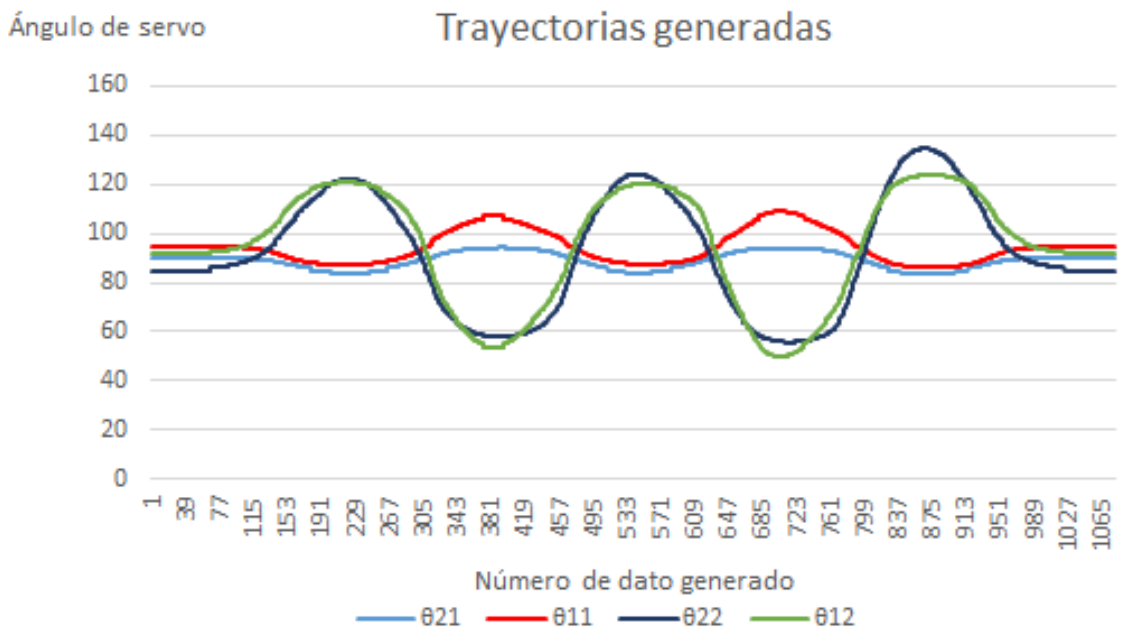


Figura 2.14. Trayectoria de ángulos de servomotores de las articulaciones 21, 11, 22 y 12.

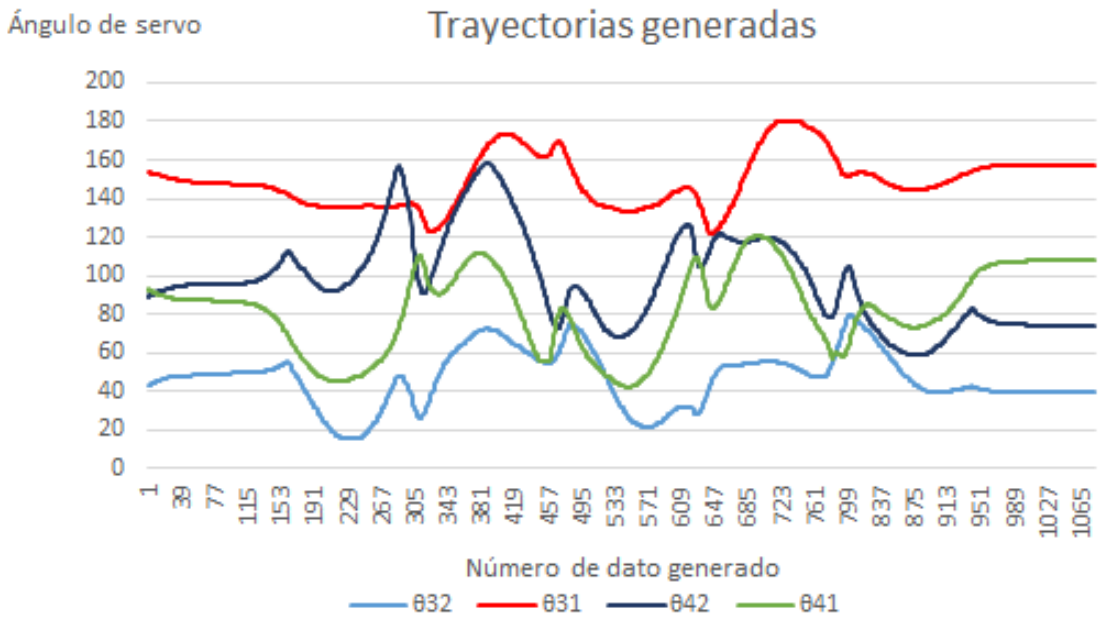


Figura 2.15. Trayectoria de ángulos de servomotores de las articulaciones 32, 31, 42 y 41

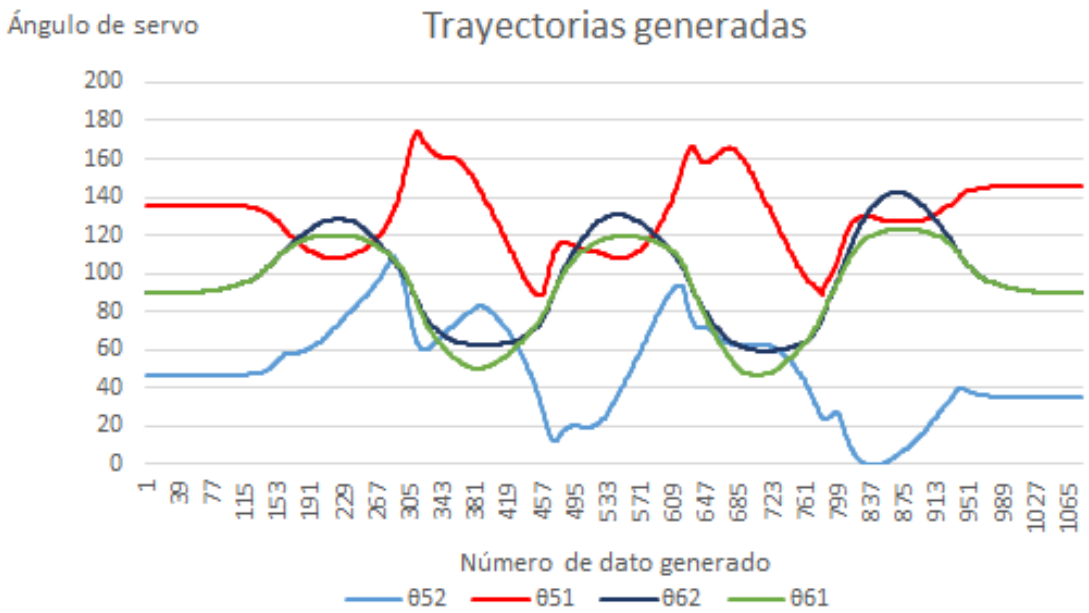


Figura 2.16. Trayectoria de ángulos de servomotores de las articulaciones 52, 51, 62 y 61

A lo largo de la trayectoria diseñada (véase figura 2.17) es posible identificar las siguientes fases: soporte simple sobre pie izquierdo (SSI); soporte simple sobre pie derecho (SSD); soporte doble donde el pie derecho se encuentra enfrente (SDD); soporte doble donde el pie izquierdo se encuentra enfrente (SDI) y soporte doble donde ambos pies se localizan a la misma altura y en contacto con el piso (SDA). El robot inicia y termina con la fase SDA y, a lo largo de la trayectoria, en total realiza dos pasos con el pie izquierdo y tres con el derecho, no precisamente en ese orden.

- Se inicia el ciclo de marcha
- El número de pasos dados con el pie derecho es igual a cero
- El robot se coloca en SDA
- Se coloca sobre su pie izquierdo
- Se verifica si ya se han dado dos pasos con el pie derecho
- Si no, el robot bípedo se coloca en SDD
- Posteriormente se para sobre su pie derecho
- Da un paso para colocarse en SDI
- Aumenta en uno el número de pasos dados por el pie derecho
- Si el robot ya ha realizado dos pasos con el pie derecho, se coloca en posición SDA
- Se finaliza el ciclo de caminata

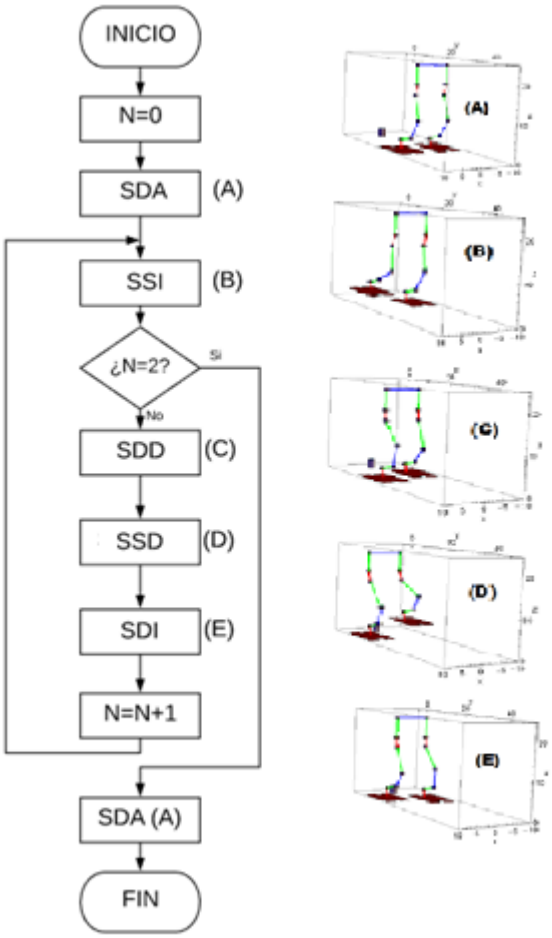


Figura 2.17. Diagrama de flujo de caminata diseñada con representación simulada y algoritmo Diagrama basado en [83]

Utilizando la notación del diagrama un ciclo de caminata podría ser expresado con la secuencia [(A), (B), (C), (D), (E), (B), (C), (D), (E), (B), (A)].

2.2.2. Servomotores HS-5645MG

No importa que tan rápido sea un sistema de control. Si los actuadores son incapaces de reaccionar a una velocidad adecuada, el robot puede caer ante una perturbación.

Teniendo conocimiento de esto, el equipo de trabajo decidió emplear el servomotor modelo HS-5645MG de la empresa surcoreana Hitec (véase figura 2.18) para hacer posible el desplazamiento del robot a través de las trayectorias calculadas o permanecer en posiciones estáticas [4]. Lo anterior fue posible debido a que sus dimensiones son compatibles con la estructura del robot y su tren de engranes de metal permiten soportar un momento angular de 12.1 kg/cm, característica necesaria si el robot se requiere sostener en un pie.

Figura 2.18. Servomotor HS-5645MG [62]

Las principales características del servomotor modelo HS-5645MG se pueden apreciar en la siguiente tabla:

Tabla G. Características técnicas de servomotor HS-5645MG [62]

Especificación	Rango de valores
Diferencia de potencial de operación (Volts)	4.8 – 6.0
Velocidad (segundos para recorrer 60°)	0.23 – 0.18
Torque máximo (kg/cm)	10.3 – 12.1
Corriente de operación sin carga (mA)	450
Máximo consumo de corriente (mA)	2400
Peso (gramos)	60
Dimensiones (mm)	40.6 x 19.8 x 37.8

2.2.3. Sensores del robot bípedo

La medición de las variables de control del robot es indispensable si se pretende realizar un control realimentado ya que son el medio por el que el robot se relaciona con su entorno. Sin información del exterior el robot sería incapaz de reaccionar ante una perturbación que lo haga caer. Por lo anterior, es importante conocer las características de cada sensor para realizar una medición adecuada.

El robot bípedo Scout cuenta con dos tipos de sensores previamente seleccionados [4], que permiten obtener los parámetros de control (ángulos de la cadera y ZMP).

Un dispositivo que nos permite medir los ángulos de Euler de un sistema es la unidad de medición inercial (IMU por sus siglas en inglés) que se utiliza principalmente en sistemas de navegación. Su principio de funcionamiento se basa en el uso de un acelerómetro, que se encargan de medir la aceleración inercial, mientras que un giroscopio mide la rotación angular. Ambos sensores tienen tres GDL para realizar aproximaciones en tres ejes [63]. Actualmente

las IMU también utilizan un magnetómetro para obtener una orientación con respecto al campo magnético terrestre y aumentar la exactitud de la medición, algo parecido a una brújula.

Para medir la orientación de la cadera el sensor UM7-LT (véase figura 2.19) permite conocer los ángulos pitch y roll. A diferencia de una IMU típica, que sólo proporciona lecturas de sensor sin procesar, el UM7-LT presenta un microcontrolador incorporado que combina los datos del sensor utilizando un filtro de Kalman ampliado que se utiliza para obtener una lectura de los ángulos con mayor precisión y generar estimaciones de orientación 500 veces por segundo. Además, el UM7-LT tiene la capacidad de realizar comunicación serial con cualquier microcontrolador y un magnetómetro, que debe ser calibrado si se requiere conocer el ángulo *yaw* del sistema [64]. Las características anteriores permitieron al equipo de trabajo considerar el sensor UM7- LT como la IMU que el robot utilizará en las mediciones de los ángulos de la cadera [4].

Figura 2.19. Sensor UM7-LT [64]

Además del UM7-LT, el robot cuenta con ocho sensores resistivos de fuerza (FSR por sus siglas en inglés) en la planta de cada pie. Los FSR son dispositivos hechos con película de polímero y presentan una disminución en su resistencia eléctrica cuando aumenta la magnitud de una fuerza aplicada en su superficie activa [65]; básicamente se trata de sensores que posibilitan medir una fuerza a través de una señal eléctrica.

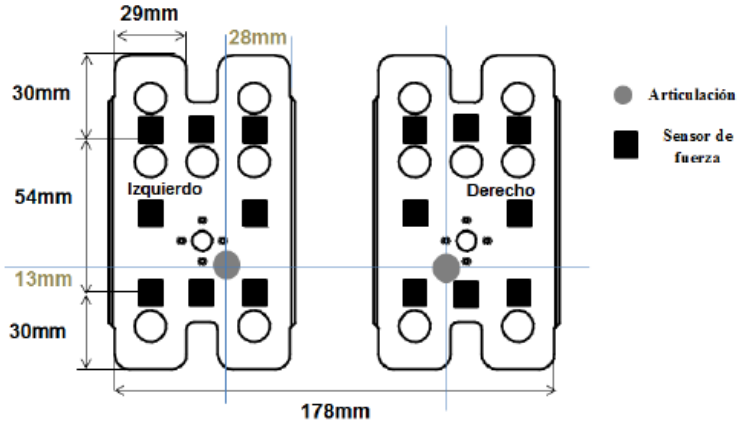


Figura 2.20. Sistema de arreglos de sensores de fuerza en la planta de los pies del robot bípedo [4]

Para analizar las fuerzas dentro del polígono formado y calcular el ZMP, Saddan acomodó en una plantilla antiderrapante dieciséis sensores resistivos de fuerza de manera simétrica y con una referencia en la articulación del tobillo del pie izquierdo (véase figura 2.20).

El bajo costo y propiedades físicas del modelo FlexiForce A201 (véase tabla H) permitieron a Saddan Hernández considerarlo como una opción adecuada para emplearse en la medición de fuerzas de reacción en los pies respecto al piso. El FSR que se comenta es, además, flexible, delgado y, mediante un circuito de acondicionamiento sencillo, permite obtener mediciones aproximadas (véase figura 2.21).

Tabla H. Características técnicas de sensor FlexiForce A201 [67]

Especificación	Valor
Grosor	0.203 mm
Largo	152 mm
Ancho	14 mm
Diámetro de área de Sensado	9.53 mm
Conector	3 pines cuadrado macho
Temperatura de operación	-40°C – 60°C
Tiempo de respuesta	<5 µs

Figura 2.21. Sensor FlexiForce A201 [66]

3. Elementos auxiliares del sistema de lazo cerrado del robot bípedo.

3.1. Interfaz gráfica

Una interfaz gráfica se define como un medio que, mediante dispositivos y conexiones, permiten la interacción del usuario con la computadora y sus contenidos [68].

En el caso del robot bípedo es necesario el desarrollo de un software con una interfaz gráfica que permita enviar los ángulos generados al robot de manera eficiente. A pesar de que un humano podría enviar cada dato manualmente, se tardaría demasiado en comparación con un programa informático; lo que a una persona le toma segundos, a una computadora, microsegundos.

La interfaz gráfica del robot bípedo programada en Labview era muy útil en cuanto a simulaciones, programación gráfica y facilidad de modificación, sin embargo, no ha permitido un envío de ángulos eficiente y una recepción de datos simultánea. Por esta razón, se ha buscado otro lenguaje de programación para generar una nueva interfaz.

Existen gran cantidad de lenguajes de programación, cada uno con sus ventajas y desventajas; sin embargo, para los propósitos de este trabajo es necesario encontrar uno con la capacidad para enviar datos, graficar y con abundante documentación que permita su fácil comprensión.

C# (leído “C Sharp”) es un lenguaje de programación desarrollado por Microsoft para la plataforma .NET, adecuando todas sus estructuras a las características y capacidades de dicha plataforma. Al ser posterior a C++ y Java, que actualmente son los lenguajes más conocidos que permiten una programación orientada a objetos, C# mejora y combina la mayoría de las características más importantes de ambos lenguajes [69]. Las características de C# lo convierten así, en un lenguaje idóneo para el proyecto.

Algunas de las ventajas que pueden destacarse del lenguaje de programación C# son las siguientes:

- Es un lenguaje simple en cuanto a sintaxis [70]
- Existe una gran cantidad de información disponible en internet [69]
- Está orientado a objetos
- Se trata de un lenguaje de programación de propósito general
- Se puede hacer una fácil migración del programa a un nuevo lenguaje, especialmente si el programador está familiarizado con C, C++ y Java
- Cuenta con aplicaciones multi-hilo simplificadas
- Posee una buena compatibilidad con sistemas embebidos (por ejemplo, microcontroladores) [71]

Microsoft Visual Studio es, a su vez, un entorno de desarrollo integrado que soporta diversos lenguajes de programación, entre ellos, C#. También es un software de libre descarga y con vasta información en sitios de internet, por lo que resulta igualmente adecuado para los fines planteados en este trabajo de experimentación.

En la figura 3.1 se observa la interfaz realizada en C# ejecutándose y, con el número correspondiente a sus elementos, se señalan cada una de sus características:

1. Botones que permiten la comunicación con el robot: pausa en la marcha, rebobinado de datos y salida del programa.
2. Casillas de configuración de envío: velocidad de transferencia de datos y elección del microcontrolador del robot bípedo.
3. Casilla de trayectoria de servos: donde se escribirá la ruta del archivo de texto que contiene los ángulos de marcha previamente generados.
4. Casilla de establecimiento de retraso: en esta casilla se propone una pausa entre cada dato de salida, con el fin de que se envíen a cierta velocidad; el sistema espera un tiempo establecido para mandar el próximo dato lo que permite que cada ángulo llegue separado del próximo y evitar errores de comunicación.
5. Interruptor loop: hace posible la repetición de ciclos de marcha, de modo que el robot siga caminando hasta que el usuario lo desee.
6. Indicadores del estado del robot: muestran el total de ciclos que se han repetido, el número de datos enviados y cuál se está enviando al momento.
7. Caja de texto de archivo de registro: en ella se establece la ruta del documento de texto donde se guardan los datos provenientes del microcontrolador que envía por comunicación serial para su posterior análisis.
8. Gráfica: muestra las variables provenientes del robot bípedo al momento de la marcha.
9. Interruptor de sistema de control: activa y desactiva el algoritmo de control.
10. Pestañas de ayuda e información sobre el programa.
11. Selección de variable control a analizar.

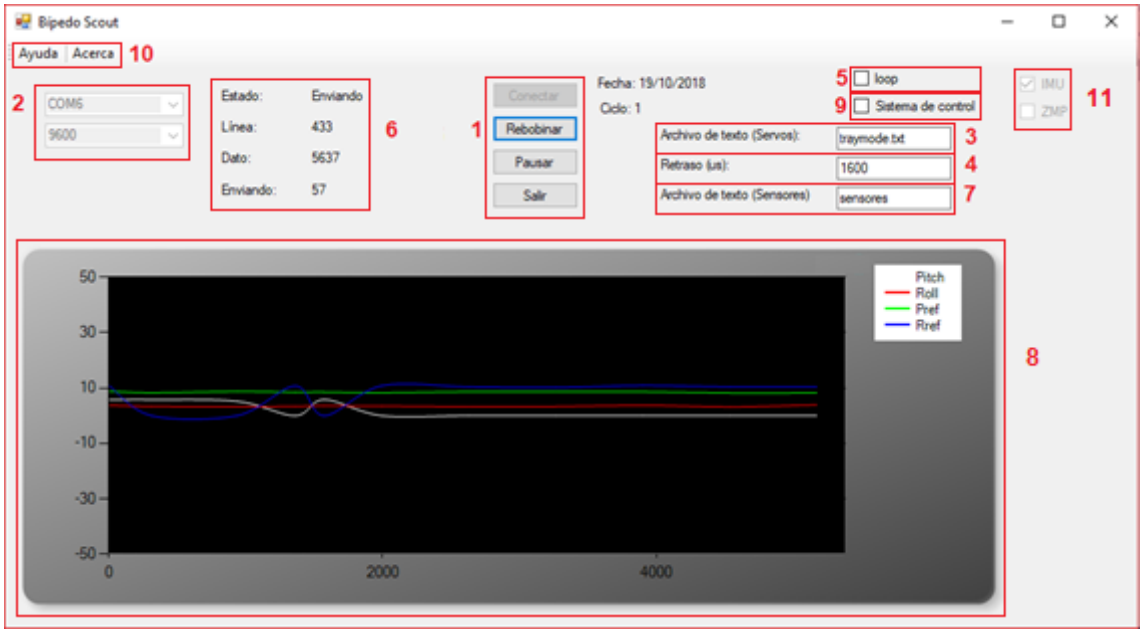


Figura 3.1. Interfaz en C# ejecutándose

3.2. Tarjeta de desarrollo TM4C123G LaunchPad

La comunicación entre el microcontrolador y la computadora es implícitamente bilateral, por lo que, al aumentar la velocidad de envío de datos desde la interfaz en la computadora, se debe aumentar la velocidad de comunicación del controlador. En el mercado existe una amplia

cantidad de tarjetas de desarrollo que pueden sustituir a la actual Arduino Mega (véase figura 3.2).

Figura 3.2. Vista frontal de tarjeta Arduino Mega [72]

La tarjeta de desarrollo Tiva™ C Series TM4C123G LaunchPad de la empresa Texas Instruments está diseñada a partir del microcontrolador TM4C123GH6PM (véase figura 3.3) que posee un oscilador cinco veces más rápido que el del microcontrolador actual y su programación puede realizarse en el software *Energia*, un entorno similar al utilizado en la tarjeta Arduino que permite migrar de manera sencilla los programas antes realizados y trabajar con base en ellos.

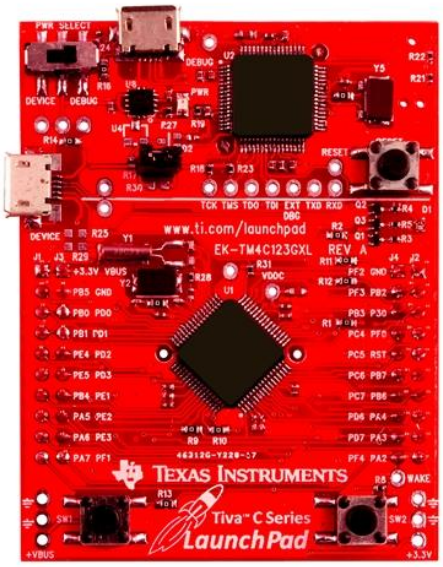


Figura 3.3. Vista frontal de tarjeta TM4C123G LaunchPad [73]

Las principales características y diferencias entre el Arduino Mega 2560 y la tarjeta TM4C123G, pueden encontrarse en la siguiente tabla:

Tabla I. Comparación técnica entre tarjetas Tiva y Arduino [60] [74]

TARJETA	ARDUINO MEGA 2560	TM4C123G LaunchPad
Diferencia de potencial de operación	5V	3.3 - 5 V
Diferencia de potencial de alimentación	7-12V	4.75 - 5.25V
Pines de entrada y salida digitales	54 (15 PWM)	35 (16 PWM)
Pines de entrada analógicos	16	12
Memoria flash	256 KB	256 KB
EEPROM	4 KB	2 KB
Velocidad de oscilador	16 MHz	80 MHz
Puertos seriales	4	8

Si se señalan las desventajas, aunque en la Tarjeta Tiva TM4C123G se pueden guardar menos valores de manera permanente debido a la baja capacidad de almacenamiento de su memoria EEPROM, tal característica resulta poco relevante para los fines de este proyecto, ya que al momento de la caminata todos los datos se guardan en la memoria RAM. La tarjeta también cuenta con una inferior cantidad de entradas analógicas que limitan la cantidad de sensores que podemos leer al mismo tiempo, lo cual se soluciona utilizando un multiplexor analógico.

A pesar de que sólo posee 35 pines I/O digitales, son más que suficientes para mover los servos del robot y realizar la caminata, al mismo tiempo que se revisan sus variables de control.

Al considerar todas estas características, se optó por implementar la tarjeta Tiva TM4C123G que, al comunicarse con la interfaz en C#, tiene como resultado un sistema que realiza un ciclo de marcha de 40 segundos, al mismo tiempo que grafica datos enviados desde el microcontrolador y los guarda en un documento de texto separado por tabulaciones.

3.3. Algoritmo de identificación de fases de caminata.

El algoritmo de control debe comportarse de manera diferente durante cada fase de la caminata pues, no es lo mismo el análisis dinámico del robot en SS que en SD. Resulta necesario, por tanto, emplear una metodología de obtención de fases de caminata y para ello, el uso de los FSR es indispensable, ya que ofrecen información de las áreas de contacto de las plantas de los pies. De esa suerte, si los sensores de algún pie no están activos, significa que este se encuentra levantado y el robot en SS. Por otro lado, si los sensores de ambos pies están en contacto con el piso, se deduce que el bípedo se encuentra en SD. Teniendo en cuenta las premisas antes mencionadas, y con ayuda del diagrama de flujo de las fases de caminata, se logró determinar la siguiente tabla:

Tabla J. Características de fases cinemáticas de caminata

Fase cinemática	Sensores pie izquierdo activos	Sensores pie derecho activos	Característica
SDA	SI	SI	Es la primera y última fase cuando ya se han dado cinco pasos
SSI	SI	NO	Siempre va después de la fase SDI o SDA
SDD	SI	SI	Sucede después de SSI excepto cuando el robot ya ha dado cuatro pasos
SSD	NO	SI	Siempre se encuentra después de SDD
SDI	SI	SI	Es la fase posterior a SSD

Finalmente, se procedió a aplicar un algoritmo que, tomando las características de cada fase cinemática de la caminata, dio como resultado una diferenciación automática en una u otra de ellas mientras el robot está realizando un ciclo de caminata (véase figura 3.4).

Figura 3.4. Fuerzas de reacción en cada pie durante un ciclo de caminata

Donde:

(1) SDA en los intervalos [0,190] y [919,1080]
(2) SSI en los intervalos [191,286], [506,606] y [848,919]
(3) SDD en los intervalos [287,330] y [607,675]
(4) SSD en los intervalos [331,444] y [676,786]
(5) SDI en los intervalos [445,505] y [787,847]

De acuerdo a los datos obtenidos en la experimentación se encontró que en el intervalo [506,606] se produce una pequeña perturbación en la fuerza medida del pie derecho cuando ésta, se supone, debería ser siempre nula (debido a que es la fase SSI). Lo anterior se debe a que los ángulos de trayectoria en la marcha no alzan suficientemente el pie y en ocasiones se produce un ligero roce de los FSR con la superficie del suelo. En el tema 3.4 se desarrollará un sistema que evite este tipo de situaciones.

3.4. Sistema de elevación de pie

Durante la fase SSD, el robot debe de tener el pie izquierdo levantado hasta que llegue el momento de volver a apoyarlo para pasar a la fase SDI; sin embargo, como se muestra en la sección anterior, su pie puede llegar a rozar el suelo en momentos en que no es conveniente que lo haga, debido a que se podrían producir fuerzas de reacción que alterarían la estabilidad de su marcha. Además, si se planea hacer una diferenciación de las fases de la caminata, estas deben ser totalmente distintivas.

Con objeto de evitar la perturbación, se utilizó un esquema simplificado de los eslabones de la rodilla izquierda (véase figura 3.5) para analizar cómo el robot levanta el pie y diseñar un desplazamiento vertical del mismo sin producir un desplazamiento horizontal con respecto a la cadera.

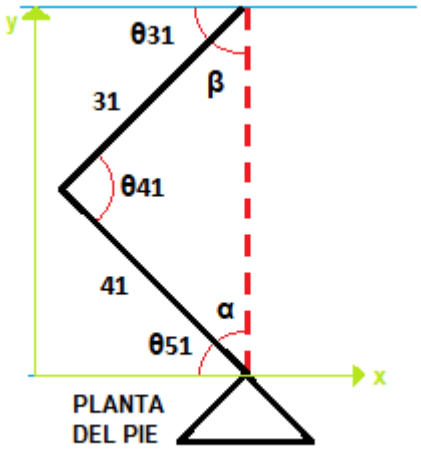


Figura 3.5. Representación cinemática de los eslabones de la rodilla

Como resultado del análisis y con base en conocimientos en trigonometría se puede deducir que:

$$\theta_{51} + \alpha = 90° \ldots (1)$$

$$\theta_{31} + \beta = 90° \ldots (2)$$

$$\alpha + \theta_{41} + \beta = 180° \ldots (3)$$

Resolviendo el sistema de ecuaciones formado por (1), (2) y (3) se obtiene la ecuación:

$$\theta_{51} + \theta_{31} = \theta_{41} \ldots (4)$$

Por otra parte, considerando que el cambio de posición en el eje horizontal del motor 51 respecto a la cadera es igual a cero, se tiene que:

$$\Delta x_{51} = 0 \ldots (5)$$

$$\Delta x_{51} = L31cos\theta_{31} - L41cos\theta_{51} \ldots (6)$$

Igualando las ecuaciones (5) y (6) se tiene que:

$$L31cos\theta_{31} = L41cos\theta_{51} \ldots (7)$$

Donde L31 y L41 son las longitudes de los eslabones 31 y 41 respectivamente, y haciendo la consideración de que tienen la misma longitud, se obtiene la siguiente ecuación:

$$\theta_{31} = \theta_{51} \ldots (8)$$

Finalmente, de la sustitución de la ecuación (8) en (4) se tiene que:

$$2\theta_{31} = \theta_{41} \ldots (9)$$

La ecuación (9) determina que para elevar el pie es necesario que los ángulos θ_{51} y θ_{31} tengan el mismo cambio en su posición, mientras que el ángulo θ_{41} debe ser la suma de ambos para que no exista movimiento en el eje horizontal. Esto tendrá como resultado una elevación vertical del pie que responde a la siguiente relación:

$$EVP = (L41 + L31)sen\theta_{31} \ldots (10)$$

Donde L31=L41= 5 cm.

3.5. Comprobación experimental de parámetros de caminata

Una caminata en condiciones ideales debería dar como resultado que los parámetros de diseño medidos –los ángulos de Euler de la cadera y el ZMP- sean igual a los teóricos, por lo que, entre más alejados se encuentren de lo programado, mayor probabilidad existirá de que el robot tenga una caída. Por esta razón, medir los valores en una superficie horizontal y sin perturbaciones permitió conocer el punto de partida en la aplicación de un sistema de control y comparar los resultados una vez implementado.

3.5.1. Medición de ángulos de cadera durante marcha

En el caso de los ángulos de la cadera se utilizó la IMU durante un ciclo de marcha en una superficie horizontal y un entorno con mínimas perturbaciones. Al realizar el análisis de resultados se encontró que los ángulos de la cadera durante la caminata presentan una variación diferente a cero (véase figura 3.6); dicho valor es la referencia de la caminata debido al diseño basado en el modelo parametrizado carro-mesa.

Figura 3.6. Ángulos medidos pitch y roll de la cadera durante marcha

3.5.2. Algoritmo de obtención de ZMP experimental

El cálculo del ZMP puede realizarse rigurosamente tomando en cuenta las masas de sus eslabones, sus centros de masa, matrices de inercia y rotación, entre otros parámetros [50]. Sin embargo, este ejercicio es costoso computacionalmente hablando, especialmente si se requiere un procesamiento adicional, como lo es la aplicación de un sistema de lazo cerrado.

No obstante, tal como se mencionó en el tema 2.1.1. *ZMP y polígono de soporte*, basta con conocer el CoP para obtener un estimado del ZMP.

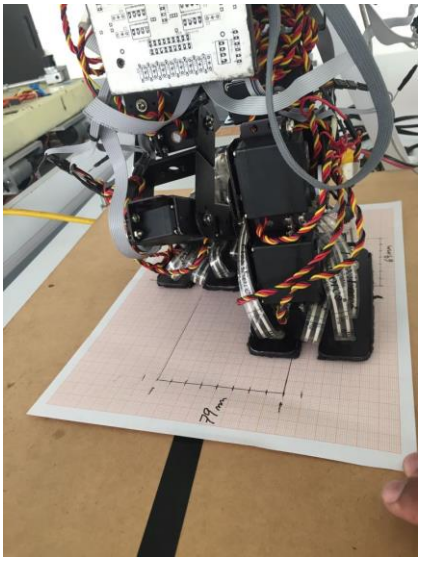


Figura 3.7. Medición experimental de longitud y ancho de paso

Dos aspectos importantes que se necesita conocer, y medir experimentalmente, son el ancho y longitud del paso ya que de ellos depende el correcto cálculo del ZMP. Para su medición se realizaron varios ciclos de marcha sobre una superficie lisa graduada que dieron por resultado los valores siguientes: longitud de paso = 69 mm y ancho de paso = 79 mm (véase figura 3.7) utilizados a su vez como las distancias entre los puntos de origen de cada pie (Xeo e Yeo).

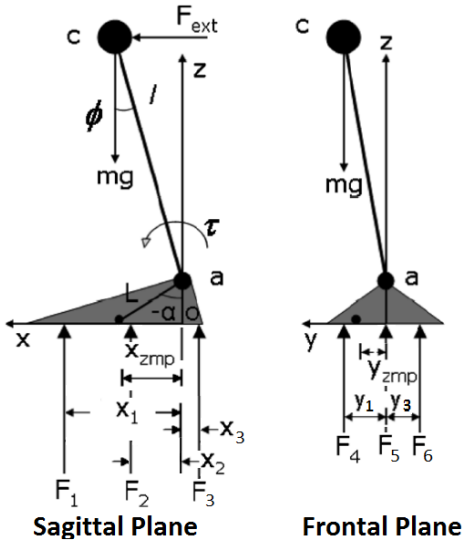


Figura 3.8. Análisis de cálculo de ZMP en planos sagital y frontal [4]

Para la obtención experimental del ZMP de cada pie a través de los FSR se propuso, a su vez, analizar el cálculo desde el plano sagital y frontal del robot según puede observarse en la figura 3.8. El ZMP del pie izquierdo en los ejes X e Y se calculó respecto al origen (la articulación del tobillo) de la siguiente manera:

$$\Sigma M_x = 0 \dots (11)$$

$$\Sigma M_y = 0 \dots (12)$$

Desarrollando la suma de momentos en las ecuaciones (11) y (12) se tiene:

$$F_1(x_1 - X_{ZMP}) - F_2(X_{ZMP} - x_2) - F_3(X_{ZMP} + x_3) = 0 \dots (13)$$

$$-F_4(Y_{ZMP} + y_1) + F_5(y_2 - Y_{ZMP}) + F_6(y_3 - Y_{ZMP}) = 0 \dots (14)$$

Donde:

*M*y es el momento en el eje Y

*M*x es el momento en el eje X

Fi es un componente de la fuerza total de reacción de la planta del robot bípedo

Xi es la distancia de aplicación de la fuerza Fi en el eje X respecto al origen

Yi es la distancia de aplicación de la fuerza Fi en el eje Y respecto al origen

Xzmpi es la coordenada en X del ZMP del pie izquierdo

Yzmpi es la coordenada en Y del ZMP del pie izquierdo

Llegado a este punto es importante considerar que la fase SD involucra al pie derecho y las fuerzas de reacción correspondientes. Por esta razón, las ecuaciones (13) y (14) se reformulan de la siguiente manera:

$$F_1(x_1 - X_{ZMP}) - F_2(X_{ZMP} - x_2) - F_3(X_{ZMP} + x_3) + F_7(x_4 - X_{ZMP}) - F_8(X_{ZMP} - x_5) - F_9(X_{ZMP} + x_6) = 0 \dots (15)$$

$$-F_4(Y_{ZMP} + y_1) + F_5(y_2 - Y_{ZMP}) + F_6(y_3 - Y_{ZMP}) + F_{10}(y_4 - Y_{ZMP}) + F_{11}(y_5 - Y_{ZMP}) + F_{12}(y_6 - Y_{ZMP}) = 0 \dots (16)$$

Despejando las coordenadas del ZMP de (15) y (16) se obtienen las siguientes ecuaciones:

$$X_{ZMP} = \frac{F_1x_1 + F_2x_2 + F_3x_3 + F_7x_4 + F_8x_5 + F_9x_6}{F_1 + F_2 + F_3 + F_7 + F_8 + F_9} \dots (17)$$

$$Y_{ZMP} = \frac{F_4y_1 + F_5y_2 + F_6y_3 + F_{10}y_4 + F_{11}y_5 + F_{12}y_6}{F_4 + F_5 + F_6 + F_{10} + F_{11} + F_{12}} \dots (18)$$

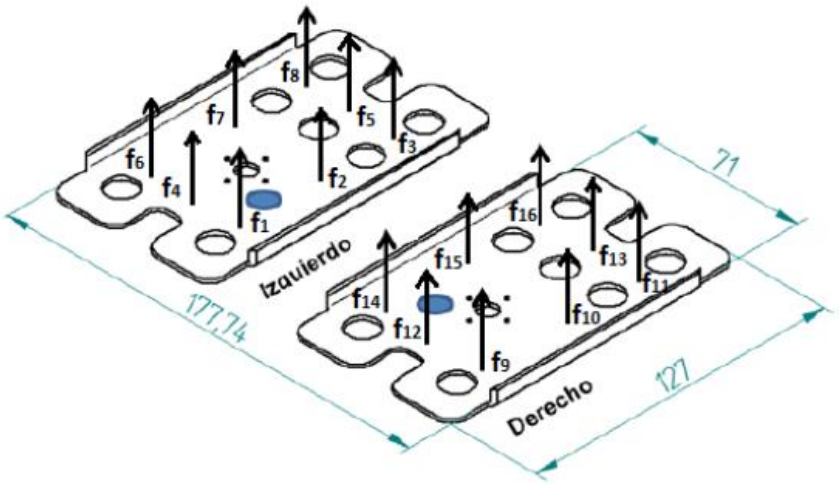


Figura 3.9. Representación de fuerzas de reacción en las plantas del pie del robot bípedo [4]

Debido a que cada fuerza F_i se mide por tres sensores de fuerza (véase figura 3.9), se tiene que:

$$F_1 = f_8 + f_5 + f_3$$
$$F_2 = f_7 + f_2$$
$$F_3 = f_6 + f_4 + f_1$$
$$F_4 = f_1 + f_2 + f_3$$
$$F_5 = f_4 + f_5$$
$$F_6 = f_6 + f_7 + f_8$$
$$F_7 = f_{16} + f_{13} + f_{11}$$
$$F_8 = f_{15} + f_{10}$$
$$F_9 = f_{14} + f_{12} + f_9$$

$$F_{10} = f_{14}+f_{15}+f_{16}$$
$$F_{11} = f_{12}+f_{13}$$
$$F_{12} = f_{9}+f_{10}+f_{11}$$

Cada f_i representa la fuerza medida por cada sensor.

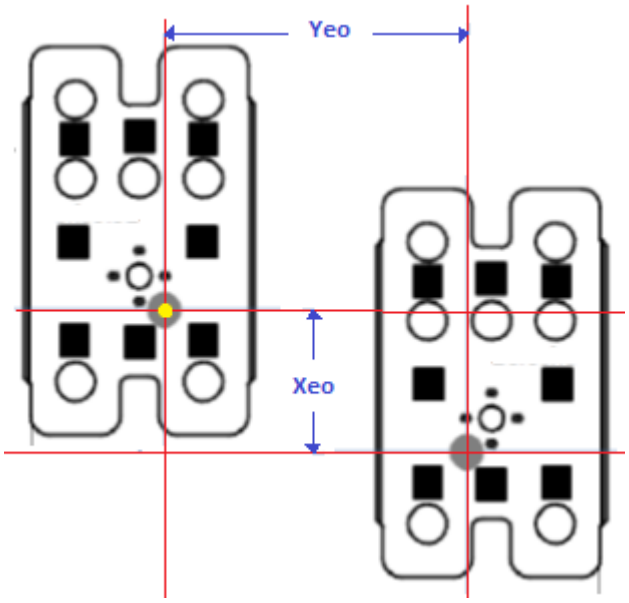


Figura 3.10. Representación de distancias Xeo e Yeo[4]

Como se vio en el tema anterior, las distancias de cada sensor respecto al origen son las siguientes:

$$x_1 = 54\ mm$$
$$x_2 = 20\ mm$$
$$x_3 = -13\ mm$$
$$y_1 = 13.5\ mm$$
$$y_2 = -7.5\ mm$$
$$y_3 = -28.5\ mm$$
$$x_4 = Xeo + 54\ mm$$
$$x_5 = Xeo + 20\ mm$$
$$x_6 = Xeo - 13\ mm$$
$$y_4 = Yeo - 13.5\ mm$$
$$y_5 = Yeo + 7.5\ mm$$
$$y_6 = Yeo + 28.5\ mm$$

[4] Esquema basado en [4]

Donde:

Xeo es la distancia entre la articulación θ_{62} y la referencia θ_{61} en el eje X, esta varía dependiendo la longitud de paso (véase figura 3.10) y en cuál fase cinemática de caminata se encuentre el robot (véase tabla K).

Yeo es la distancia entre las articulaciones θ_{62} y θ_{61} en el eje Y, la cual se puede considerar como el ancho de paso por lo que esta será igual a 79 mm en cualquier fase de soporte doble de la marcha.

Tabla K. Distancia entre centros de referencia de los pies del robot bípedo en eje X

Fase cinemática	SDA	SSI	SDD	SSD	SDI
Distancia Xeo [mm]	0	N/A	69	69	-69

Teniendo en cuenta las longitudes de paso y las fórmulas para obtener las coordenadas del ZMP, se procedió a realizar un algoritmo para calcularlo; éste, sin embrago, tiende a variar mucho entre una medida y otra debido a que, tan sólo una pequeña fuerza (la vibración de los motores, por ejemplo), puede mover las coordenadas calculadas. Teniendo en cuenta lo anterior, se decidió tomar un conjunto de ZMP calculado consecutivamente y realizar un promedio; de esta manera, se obtuvo una señal con menos ruido y un poco más confiable.

De los resultados del cálculo de los ZMP promediados durante la marcha, se puede apreciar que la distancia recorrida por la coordenada Xzmp alcanza los 350 [mm] mientras que, en la referencia ideal propuesta por Fernanda Merino, tan sólo se alcanzan 320 [mm] (véase figura 3.11). Por otro lado, las coordenadas mínimas y máximas de la coordenada Y del ZMP coinciden con las planteadas en el diseño de la caminata (véase figura 3.12).

Figura 3.11. Coordenada X de ZMP experimental durante caminata

Figura 3.12. Coordenada Y de ZMP experimental durante caminata

Al graficar la coordenada X contra la coordenada Y del ZMP, se obtuvo un esquema similar al ZMP propuesto por Fernanda Merino. La comparación de la figura 3.13 con la figura 2.13 permite comparar los resultados experimentales con la propuesta teórica.

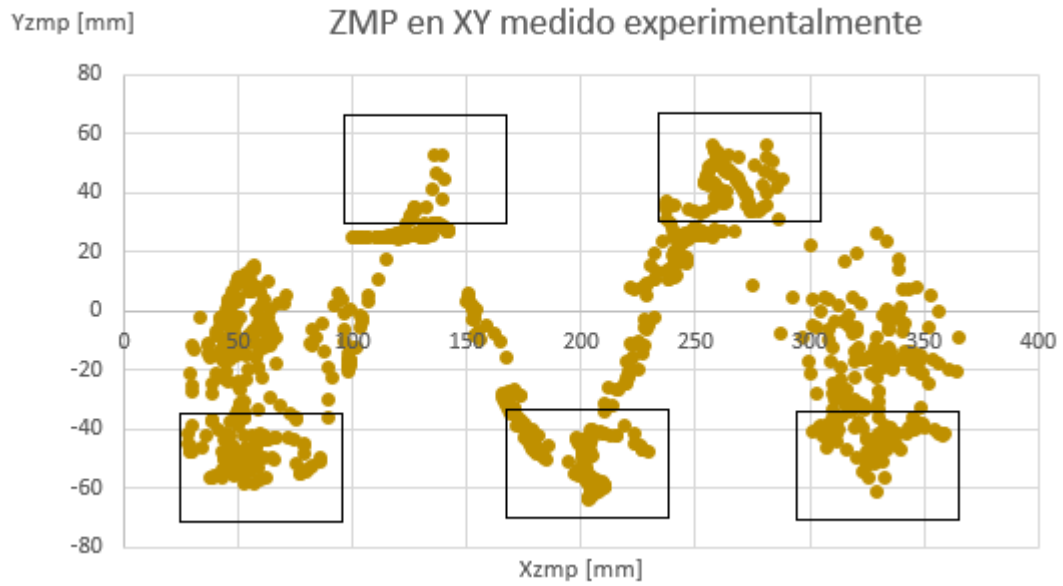


Figura 3.13. ZMP experimental en ejes XY

3.6. Diseño de referencia de control de ZMP

Considerando las mediciones experimentales del ZMP, la longitud y el ancho del paso, se determinó modificar la referencia de control del ZMP, es decir, la trayectoria de valores de ZMP que, si el robot siguiera "idealmente", derivaría en una caminata estable.

El primer paso para realizar esta referencia, fue conocer cómo se comporta estadísticamente el ZMP del robot durante una caminata. Para ello, se promediaron quince muestras de caminatas, se realizó un análisis de cada una de sus fases y se representó su comportamiento por medio de un polinomio de segundo grado a través de regresiones lineales por mínimos cuadrados [75].

Por ejemplo, cuando el robot se encuentra en soporte doble, con el pie izquierdo enfrente por segunda vez [SDI (2)], se realizaron las dos ecuaciones de segundo grado que representan el comportamiento en X e Y del ZMP. En las figuras 3.14 y 3.15 es posible apreciar las coordenadas X e Y del ZMP medido y las aproximadas por regresión lineal, en donde se observa una variación al inicio de la gráfica (intervalo [1,13]), la cual se relaciona con el impacto de la planta del pie al momento del cambio de SS a SD.

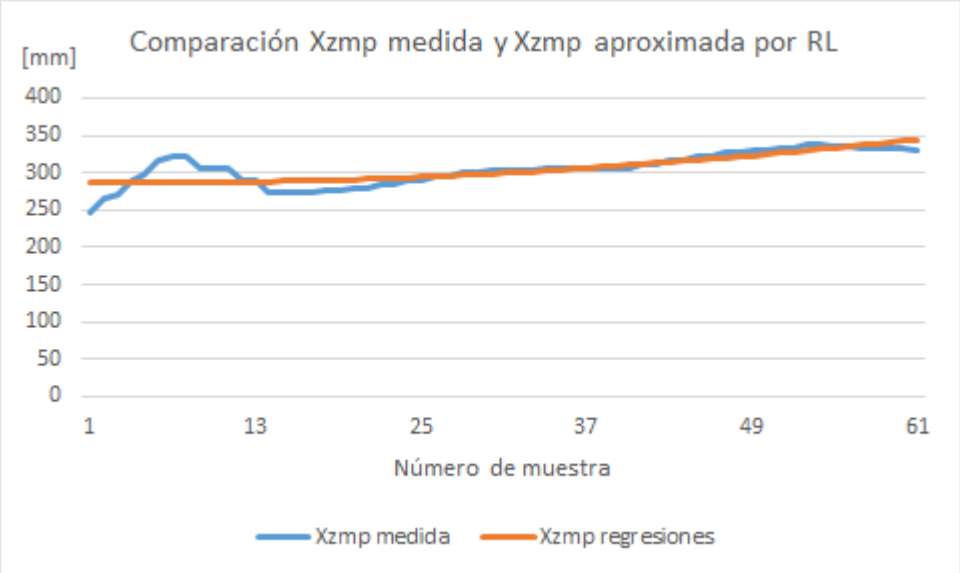


Figura 3.14. Xzmp medido y Xzmp aproximado por regresión lineal en fase SDI (2)

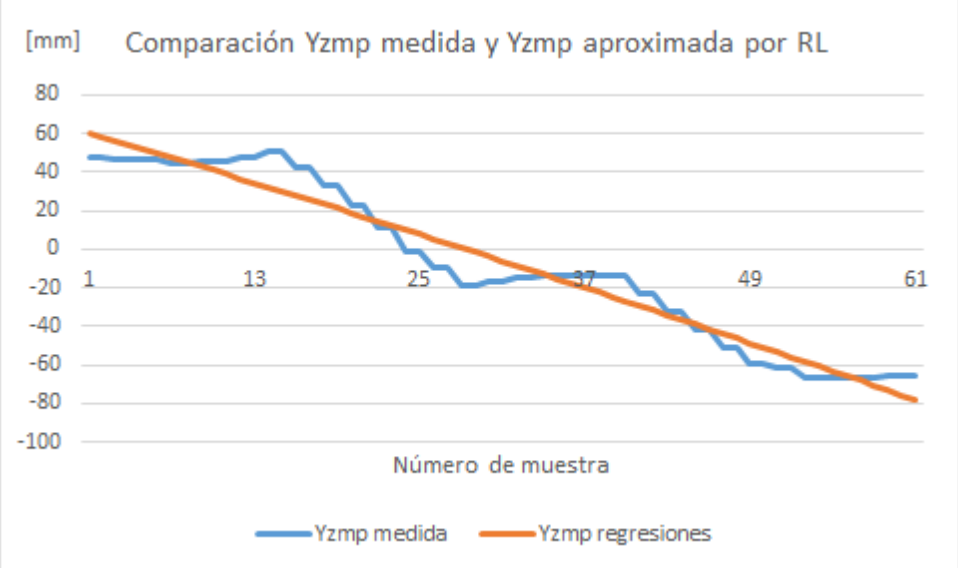


Figura 3.15. Yzmp medido e Yzmp aproximado por regresión lineal en fase SDI (2)

Posteriormente, se realizó una gráfica en los ejes XY para tener una noción más certera sobre el comportamiento del bípedo en el plano, en donde se puede apreciar que los ZMP medidos más alejados de la curva realizada por regresión lineal promueven que ésta se incline más hacia ellos (véase figura 3.16).

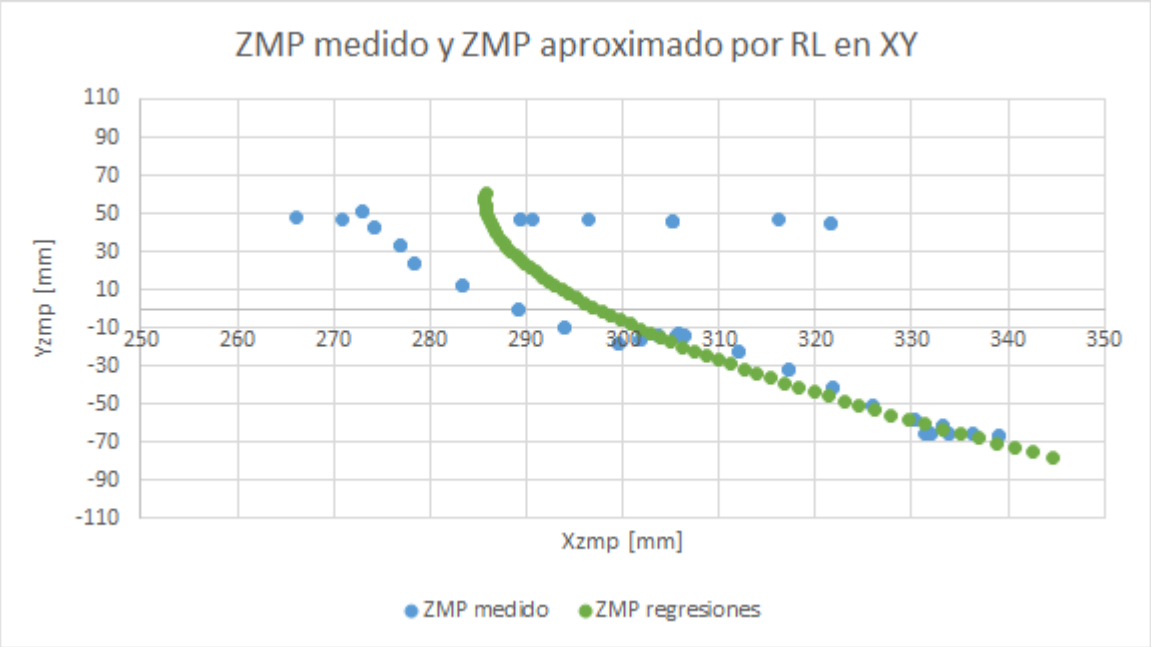


Figura 3.16.ZMP medido y ZMP aproximado por regresión lineal en XY durante SDI (2)

Una vez visualizado el comportamiento de las ecuaciones de los ZMP aproximados por regresión lineal en SDI (2), se realizó el mismo método para todas las fases con la finalidad de obtener un conjunto de ecuaciones que representen el comportamiento de las coordenadas del ZMP del robot en cada fase cinemática durante toda la marcha. En las figuras 3.17 y 3.18 se pueden apreciar algunos “picos” durante el cambio de fase, estos representan cambios bruscos en el valor del Xzmp, mientras que los valores correspondientes a la coordenada Yzmp se encuentran en el rango de -80 mm a 60 mm ubicándola en los límites del polígono de soporte.

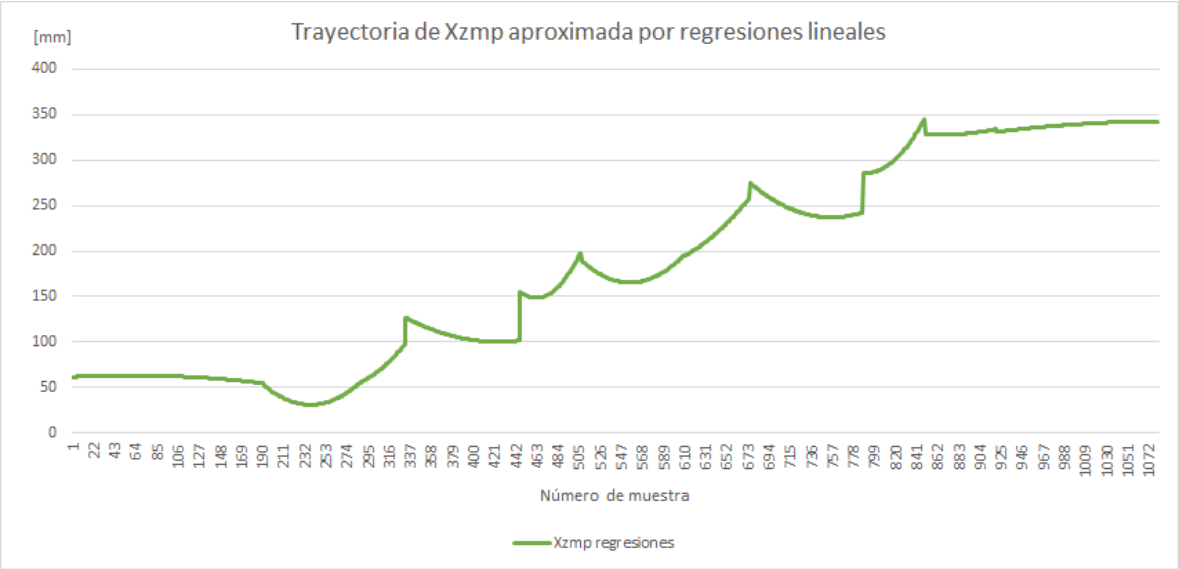


Figura 3.17. Xzmp aproximada por regresiones lineales durante la marcha

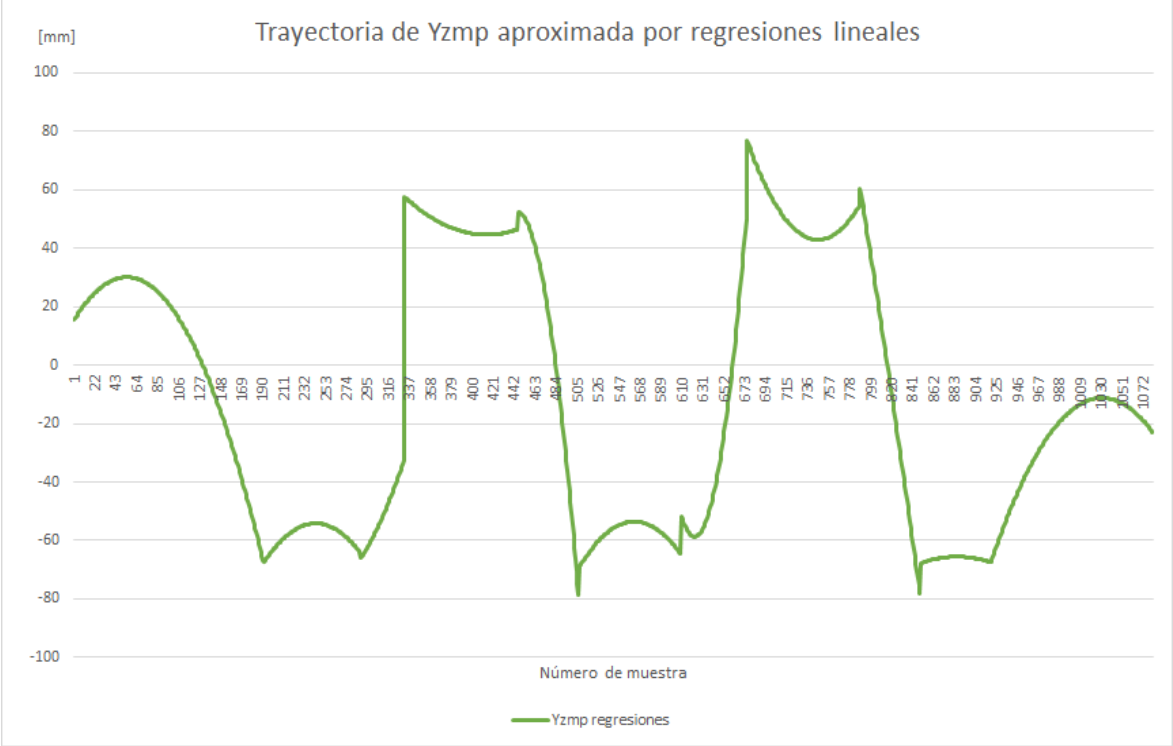


Figura 3.18. Yzmp aproximada por regresiones lineales durante la marcha

De igual manera, se realizó el análisis en el plano XY para apreciar cómo tiende a comportarse el robot durante la caminata (véase figura 3.19) en dónde se percibió una falta de continuidad entre los trazos. En círculos rojos se marcó, además, la existencia de un comportamiento inusual que pudiera interpretarse como un tipo de rebote en el ZMP al momento del pre-balanceo y post-balanceo en SS.

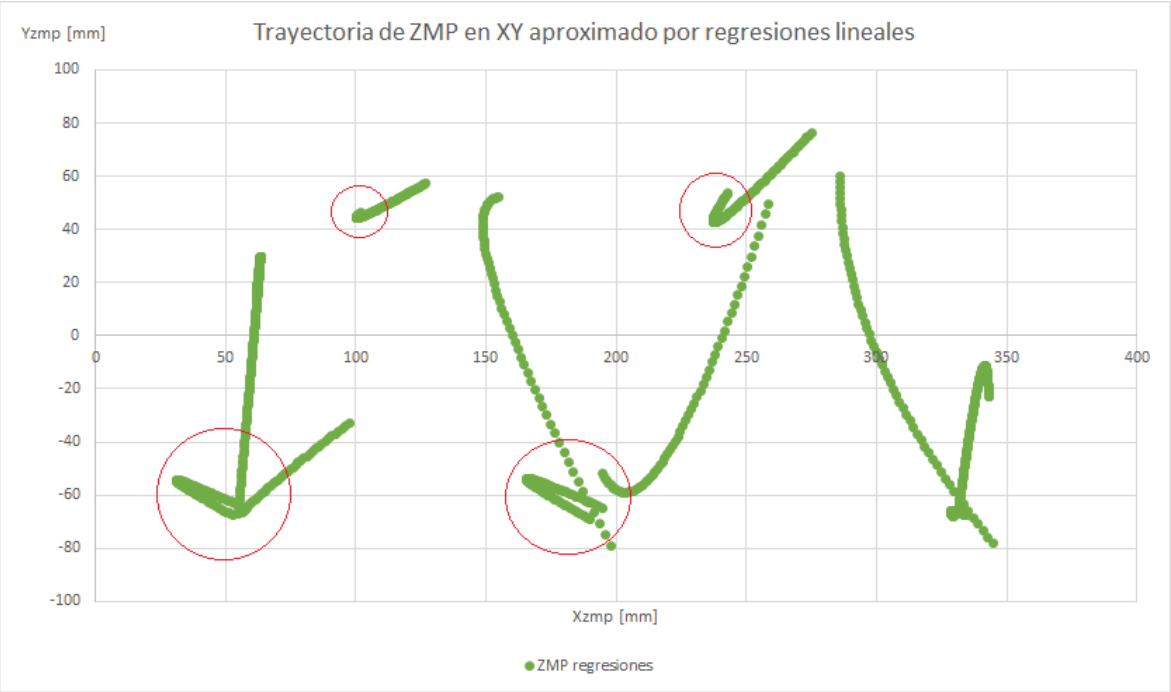


Figura 3.19. ZMP aproximado por regresiones lineales en el plano XY durante la marcha

De la trayectoria aproximada por regresiones lineales del ZMP se puede concluir que, en ocasiones, se acerca al borde del polígono de soporte, lo que se traduce en un aumento en la inestabilidad del sistema; además, tiende a tener saltos muy grandes en los valores Xzmp e Yzmp, lo cual quiere decir que, si se ocupa como referencia, el robot tendría que seguir un valor ideal que cambiaría bruscamente y esto significaría una perturbación innecesaria. Finalmente, la coordenada Yzmp empieza en 15.6 mm, pero al final de la trayectoria termina en un valor diferente (-22.8 mm), situación indeseada si lo que se pretende es repetir la caminata indefinidamente, debido a que la referencia cambiaría repentinamente al iniciar un nuevo ciclo de marcha. Lo anterior permite descartar el uso del ZMP aproximado por este método como referencia durante el recorrido del robot, aunque ofrece información fundamental a considerar en el diseño de la trayectoria de control del ZMP.

El ZMP aproximado por regresión lineal en realidad señala el comportamiento previsto del robot; visto de otra manera, el robot indica por dónde suele pasar su ZMP y su comportamiento en la fase de pre-balanceo y post-balanceo en SS.

Posteriormente, de acuerdo con la información obtenida por las regresiones lineales, se realizó una trayectoria de ZMP de referencia propuesta, es decir, se le brindó al robot un conjunto de valores, por el cual debía pasar el ZMP y así, asegurar su estabilidad. Para la creación de tal referencia se tomaron en cuenta todas las desventajas de la trayectoria por regresiones lineales. Asimismo, se consideró el comportamiento mostrado del robot. Por ejemplo, en las fases de SS el ZMP del robot tiende a ir hacia atrás antes de dar el paso para la fase de SD.

Para el diseño de esta referencia se realizó una serie de diagramas donde se propuso un conjunto de valores de ZMP basados en los resultados de las regresiones lineales para generar, en cada fase de la caminata, dos polinomios de segundo grado dados tres puntos cartesianos.

En la figura 3.20, se pueden observar las suelas del robot representadas por rectángulos azules. La curva azul simboliza la trayectoria del ZMP aproximada por regresión lineal. Como se puede apreciar, su inicio y fin se encuentran cerca del borde del polígono de soporte. La línea roja está formada por valores propuestos del ZMP que se encuentran en la zona central del polígono de soporte.

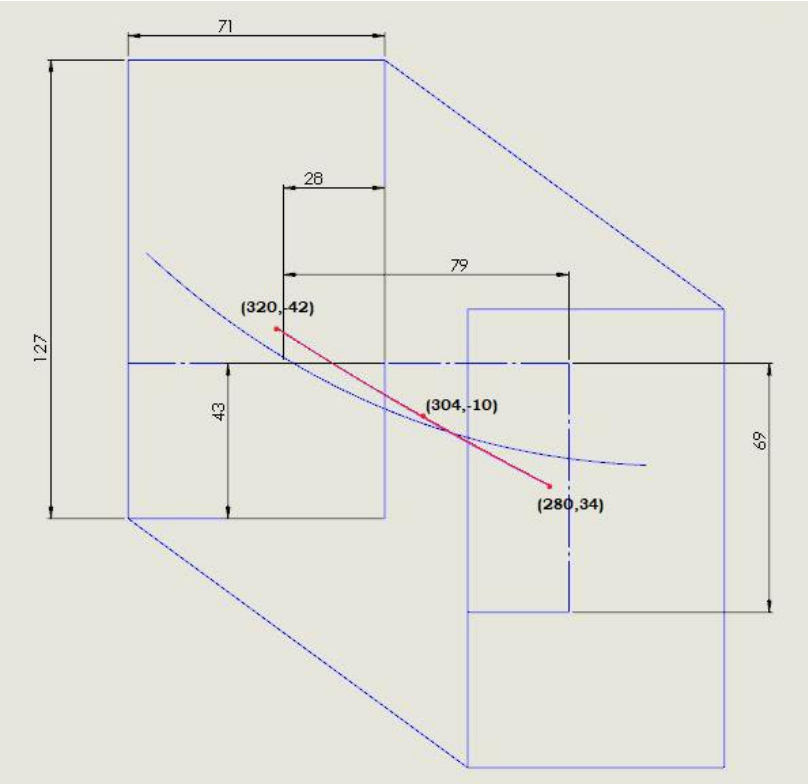


Figura 3.20. Propuesta de trayectoria de ZMP en SDI (2)

La ecuación de los ZMP de referencia resulta de la siguiente forma:

$$Xzmp\ (z) = a(z)^2 + bz + c \ldots \text{(19)}$$

$$Yzmp\ (z) = d(z)^2 + ez + f \ldots \text{(20)}$$

Donde *a, b, c, d, e* y *f* son coeficientes calculados una vez propuestos tres valores de coordenadas (X,Y) del ZMP en sus respectivos tiempos discretos *z.*

En el caso de la fase SDI (2) se propusieron las coordenadas [280,34], [304,-10] y [324,-42] para las muestras [787], [817] y [847] respectivamente, debido a que se encuentran dentro del polígono de soporte. A diferencia de los resultados mostrados en la regresión lineal en dicha fase, la coordenada Yzmp tiene valores más alejados del borde del polígono de soporte. Despejando las ecuaciones (19) y (20), y sustituyendo las coordenadas del ZMP, se calcularon los coeficientes *a, b, c, d, e y f* y se logró tener la ecuación de trayectoria de ZMP de referencia durante la fase SDI (2), con límites más seguros y, por lo tanto, una mayor confiabilidad (véase figura 3.21).

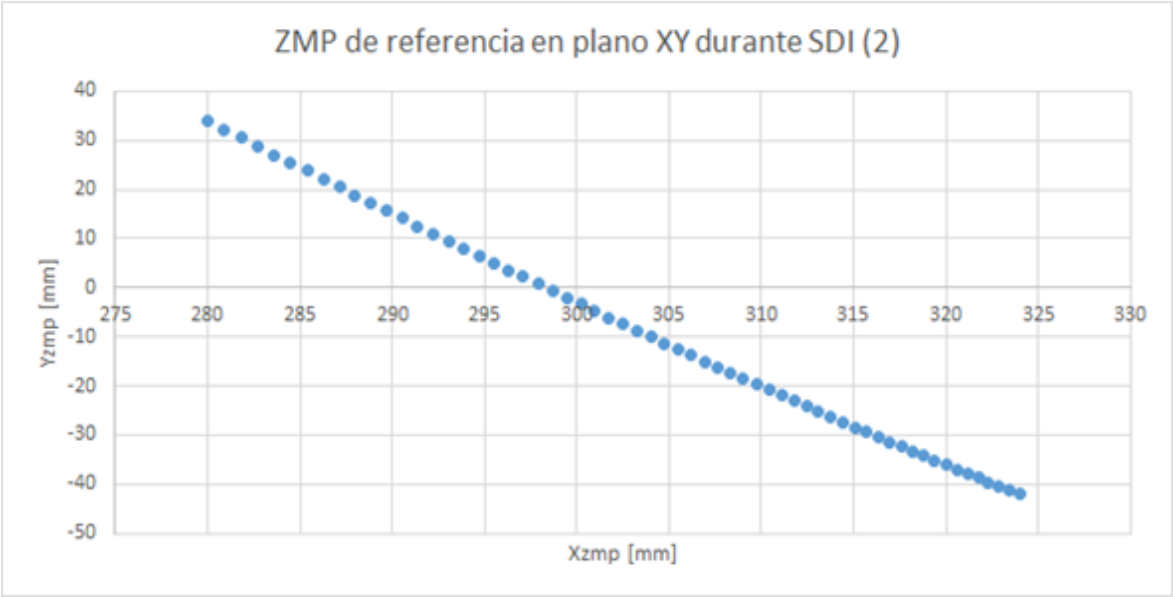


Figura 3.21. Coordenadas de ZMP de referencia en XY durante SDI (2)

A continuación, se detallarán los parámetros considerados en el diseño de las trayectorias del ZMP en cada una de las fases cinemáticas.

Es importante recordar en este punto que dichas fases son:

1. SDA en los intervalos [0,190] y [919,1080].
2. SSI en los intervalos [191,286], [506,606] y [848,919].
3. SDD en los intervalos [287,330] y [607,675].
4. SSD en los intervalos [331,444] y [676,786].
5. SDI en los intervalos [445,505] y [787,847].

Sin embargo, en este apartado se reorganizarán las fases de acuerdo con su orden al momento de realizar la caminata quedando de la siguiente manera:

1. SDA (1). Soporte doble ambos pies a la misma altura por primera vez.
2. SSI (1). Soporte simple sobre pie izquierdo por primera vez.
3. SDD (1). Soporte doble pie derecho al frente por primera vez.
4. SSD (1). Soporte simple sobre pie derecho por primera vez.
5. SDI (1). Soporte doble pie izquierdo al frente por primera vez.
6. SSI (2). Soporte simple sobre pie izquierdo por segunda vez.
7. SDD (2). Soporte doble pie derecho al frente por segunda vez.
8. SSD (2). Soporte simple sobre pie derecho por segunda vez.
9. SDI (2). Soporte doble pie izquierdo al frente por segunda vez.
10. SSI (3). Soporte simple sobre pie izquierdo por tercera vez.
11. SDA (2). Soporte doble ambos pies a la misma altura por segunda vez.

3.6.1. Diseño de ZMP referencial en SDA (1)

La fase SDA (1) es el inicio de la caminata. En ella el ZMP del robot comienza en el centro para inclinarse gradualmente a la izquierda. El robot también presenta un ligero apoyo hacia atrás

observado en la regresión lineal. Se propusieron los puntos [58,0], [48,-17] y [53,-33] en las muestras [0], [133] y [190] obteniéndose la siguiente curva:

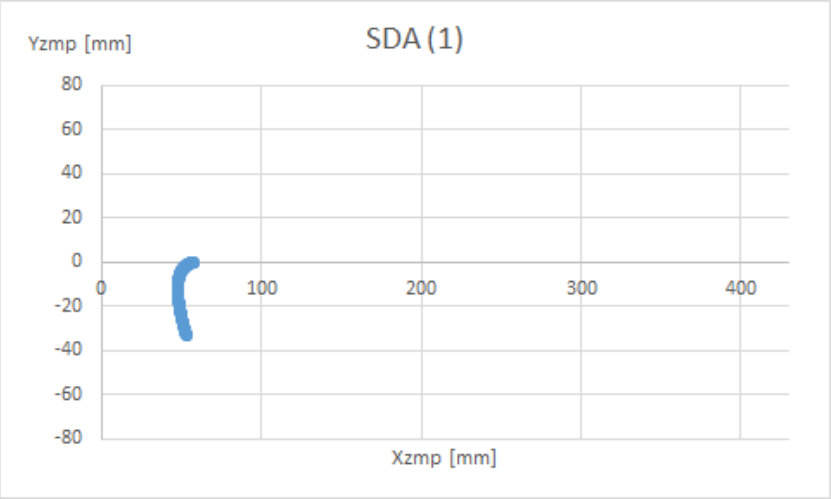


Figura 3.22. ZMP de referencia en XY durante SDA (1)

3.6.2. Diseño de ZMP referencial en SSI (1)

En la primera fase de SSI se consideraron los rebotes observados en la regresión lineal y, para mantener el ZMP dentro de una zona segura, se utilizó el último punto de SDA (1) como el inicio de la trayectoria en esta fase ([53,-33] en 191). Para su diseño se utilizaron, además, las coordenadas [48,-53] y [68,-35] en las muestras [240] y [286] (véase figura 3.23).

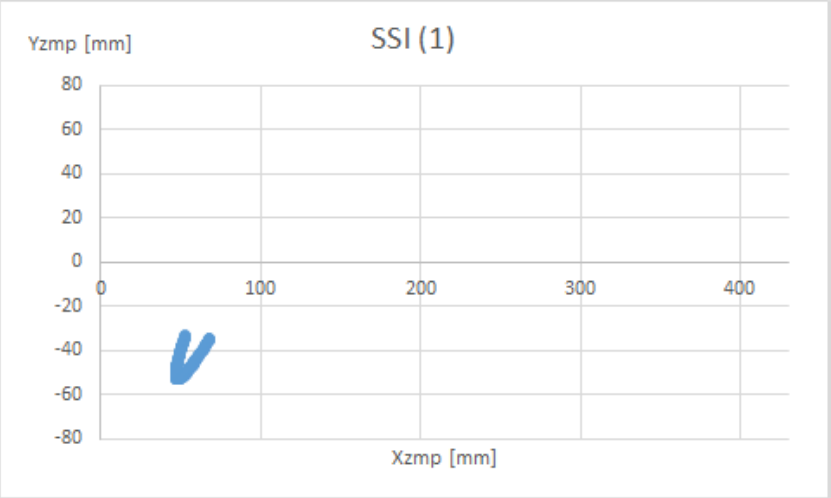


Figura 3.23. ZMP de referencia en XY durante SSI (1)

3.6.3. Diseño de ZMP referencial en SDD (1)

La fase SDD (1) tiene como fin conducir al robot hacia adelante y de izquierda a derecha. Tomando esto en consideración, se ocupó el punto final anterior [68,-35] y los puntos [89,-4] y [117, 34] en las muestras [287], [310] y [330], respectivamente, obteniendo una línea ligeramente curva (véase figura 3.24).

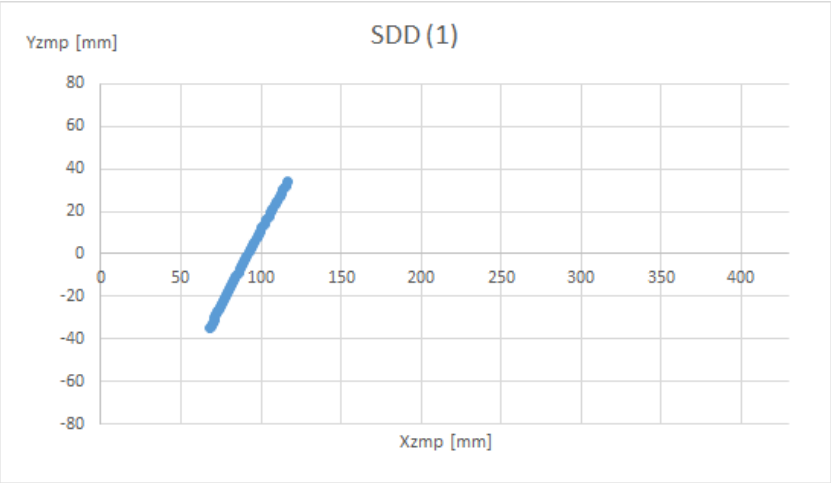


Figura 3.24. ZMP de referencia en XY durante SDD (1)

3.6.4. Diseño de ZMP referencial en SSD (1)

En la fase SSD (1) la regresión lineal mostró que el robot tiende a desplazarse hacia la derecha antes de regresar a la izquierda para avanzar a la fase de SDI (1) por lo que, además de utilizar el punto [117,34] en la muestra [331] como punto inicial para establecer una continuidad con la fase anterior, se propusieron los puntos [114,50] y [142,34] en las muestras [361] y [444], obteniendo una curva que lleva el ZMP del robot hacia atrás mientras se encuentra sobre su pie derecho (véase figura 3.25).

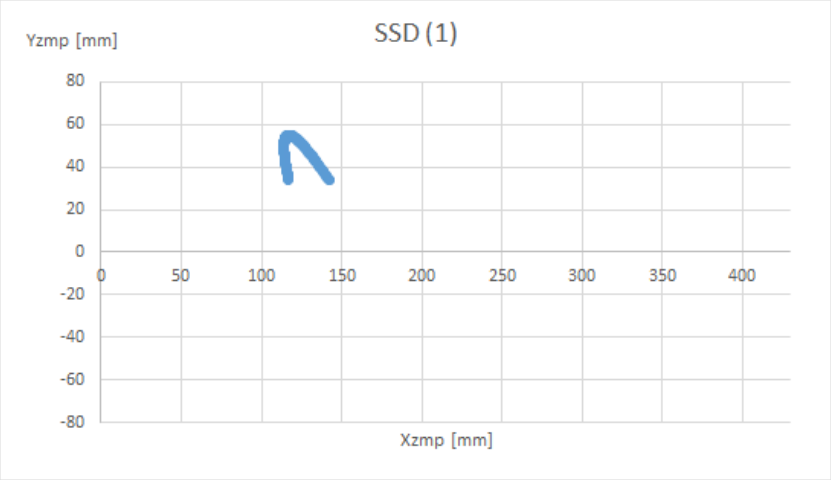


Figura 3.25. ZMP de referencia en XY durante SSD (1)

3.6.5. Diseño de ZMP referencial en SDI (1)

La fase SDI (1) tiene como objetivo dirigir el ZMP del robot hacia adelante y de derecha a izquierda, por lo que se utilizaron las coordenadas [166,-10] y [208, -45] en las muestras [475] y [505]. Además del punto inicial [142,34] en la muestra [445], que establece la conexión con la fase anterior, se obtiene una línea ligeramente curva que lleva al robot hacia el segundo soporte sobre el pie izquierdo [SSI (2)] (véase figura 3.26).

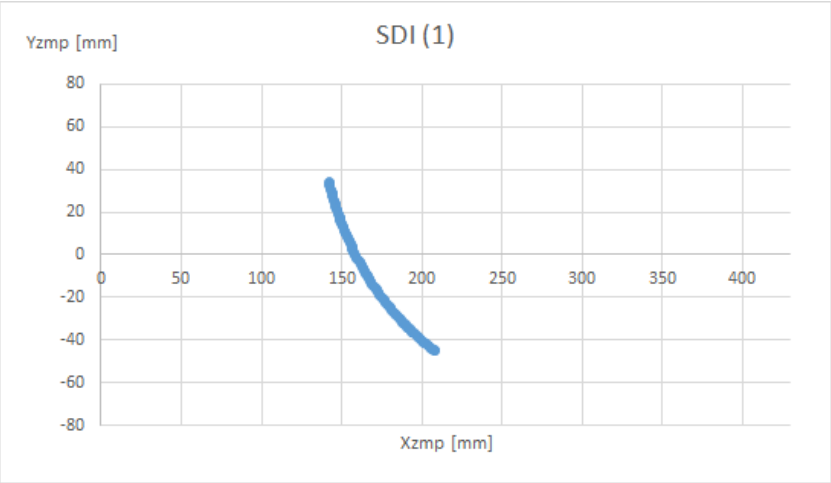


Figura 3.26. ZMP de referencia en XY durante SDI (1)

3.6.6. Diseño de ZMP referencial en SSI (2)

Para el diseño de la trayectoria del ZMP de referencia, en la fase SSI (2), se tomaron en cuenta las mismas consideraciones que en SSI (1); la diferencia en este caso es que la inercia del robot lo lleva hacia adelante, debido a que viene de la posición SDI (1). Así, además de considerar el punto inicial [208,-45] en la muestra [506], se tomaron en cuenta los puntos [191,-53] y [206,-35] en [555] y [606]. La gráfica resultante es semejante a la obtenida en SSI (1) con un desfase en la coordenada X igual a dos veces la longitud de paso (véase figura 3.27).

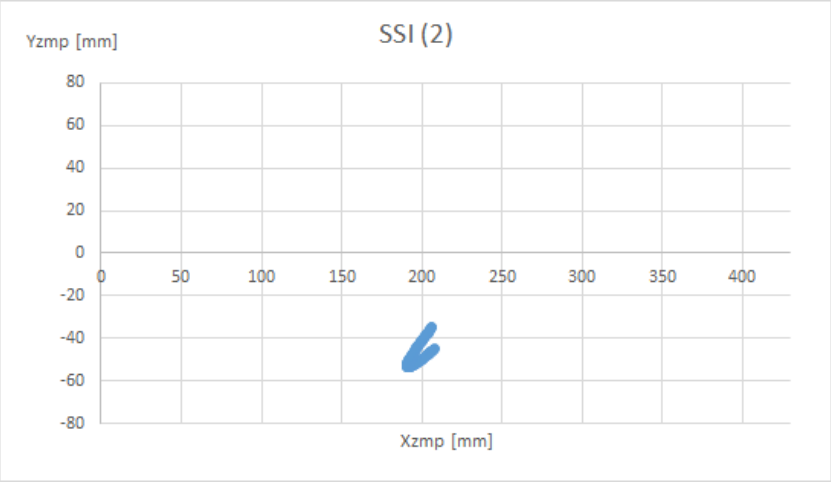


Figura 3.27. ZMP de referencia en XY durante SSI (2)

3.6.7. Diseño de ZMP referencial en SDD (2)

Al seguir la misma metodología en el diseño de la referencia del ZMP, que la utilizada en SDD (1), se obtuvo una gráfica muy semejante al emplear los puntos [206,35], [226,0] y [255,34] en los números de muestra [607], [638] y [675], respectivamente (véase figura 3.28).

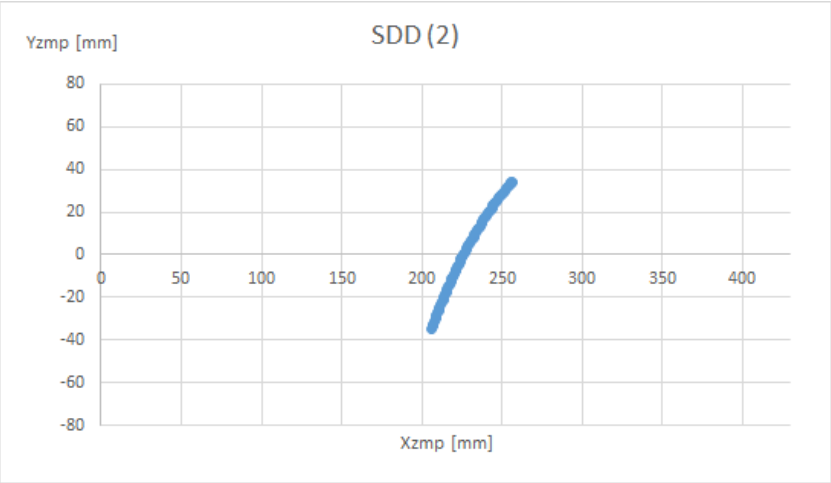


Figura 3.28. ZMP de referencia en XY durante SDD (2)

3.6.8. Diseño de ZMP referencial en SSD (2)

La trayectoria utilizada en SSD (2) es similar a la utilizada en SSD (1), con el respectivo desfase de dos veces la longitud de paso en el eje X. Para obtenerla, se utilizaron los puntos [255,34] (debido a que es el fin de la gráfica anterior) [254,55] y [280,34] en [676], [724] y [786] en ese orden (véase figura 3.29).

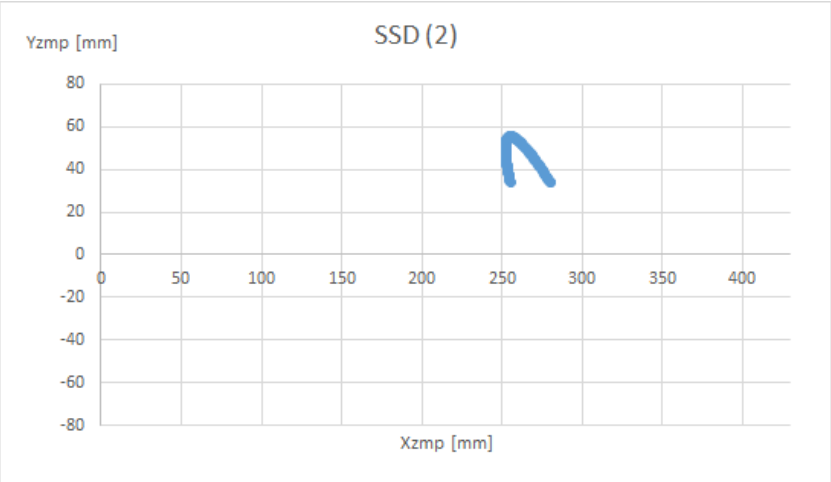


Figura 3.29. ZMP de referencia en XY durante SSD (2)

3.6.9. Diseño de ZMP referencial en SDI (2)

La fase SDI (2) (anteriormente analizada) se retoma en este apartado con el fin de establecer la conexión con las fases adyacentes. Las coordenadas utilizadas fueron los puntos [280,34], [304, -10] y [324,42] en las muestras [787], [817] y [847], su trayectoria, vista desde otra perspectiva, se puede apreciar en la siguiente figura:

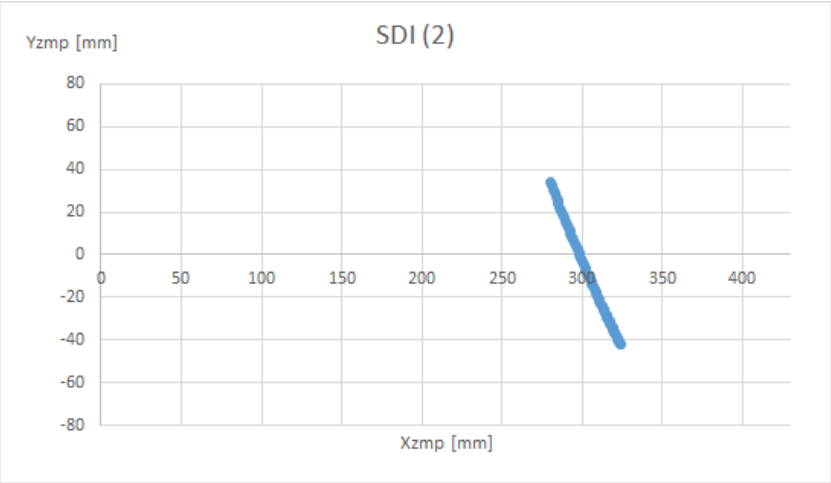


Figura 3.30. ZMP de referencia en XY durante SDI (2)

3.6.10. Diseño de ZMP referencial en SSI (3)

La regresión lineal en SSI (3) ofreció menos información que en las fases cinemáticas anteriores por lo que, para tener una conexión con la fase anterior, se consideró el comportamiento de las fases SSI (1) y SSI (2) y un punto inicial [324,-42] en la muestra [848].

Como resultado se obtuvo una curva que prepara al robot para la fase final. Dicha gráfica está formada, además, por los valores [329,-45] y [339,-35] en [878] y [919] (véase figura 3.31).

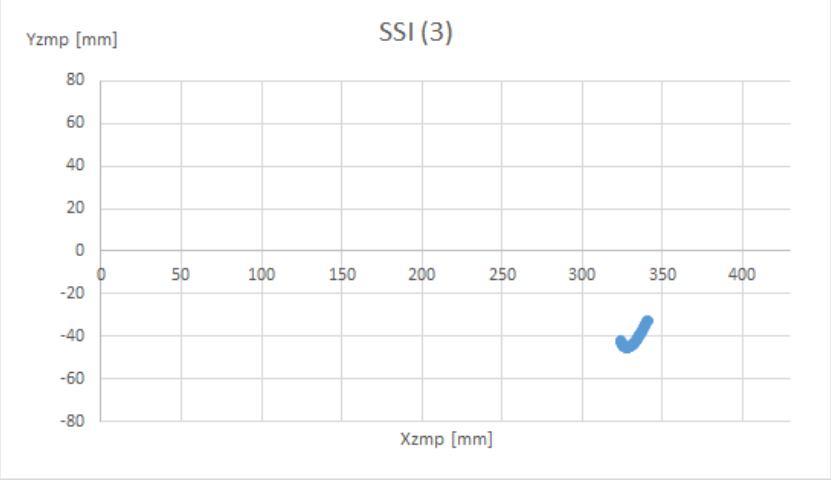


Figura 3.31. ZMP de referencia en XY durante SSI (3)

3.6.11. Diseño de ZMP referencial en SDA (2)

La trayectoria de referencia del ZMP en SDA (2) fue diseñada con los valores inmediatos de SSI (3) y SDA (1) como valor inicial y final, formados por las coordenadas [339,-35] y [334,0] en las muestras [920] y [1080], respectivamente.

Para el valor intermedio, se consideró el comportamiento de la referencia en SDA (1) y se propuso un valor en X ubicado 9 mm atrás y, en Y, un valor que se encuentre entre las dos coordenadas en Y antes mencionadas. Así se planteó el punto [330,-19] en el número de

muestra [981] obteniendo la última parte de la referencia de ZMP durante la marcha (véase figura 3.32).

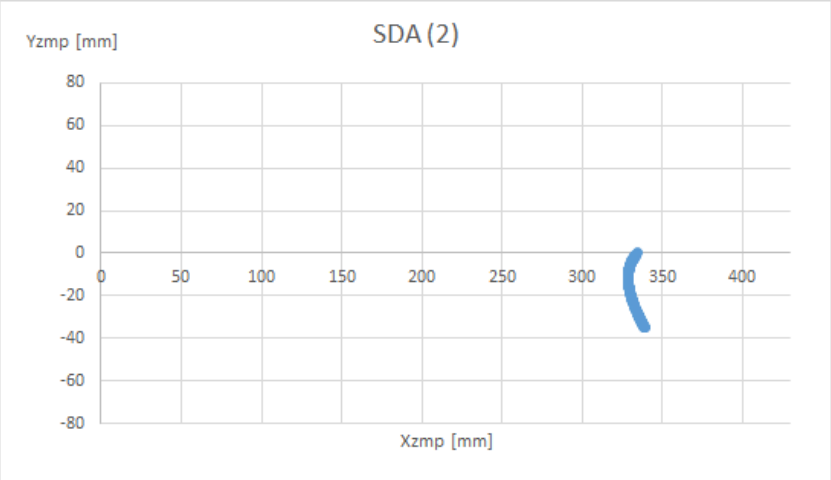


Figura 3.32. ZMP de referencia en XY durante SDA (2)

Una vez analizadas cada una de las trayectorias generadas, se procedió a unirlas para formar una trayectoria completa del ZMP referencial, la cual permite tener un parámetro de control en cada momento de la marcha. Cabe destacar que la gráfica de Xzmp propuesta se diferencia de la obtenida por regresiones lineales, por la atenuación en los cambios bruscos y por la disminución de los picos en la gráfica al tomar en cuenta los efectos pre-balanceo y post-balance (véase figura 3.33).

En el caso de la coordenada Yzmp, además es apreciable el inicio y fin de la trayectoria en Yzmp =0 (véase figura 3.34).

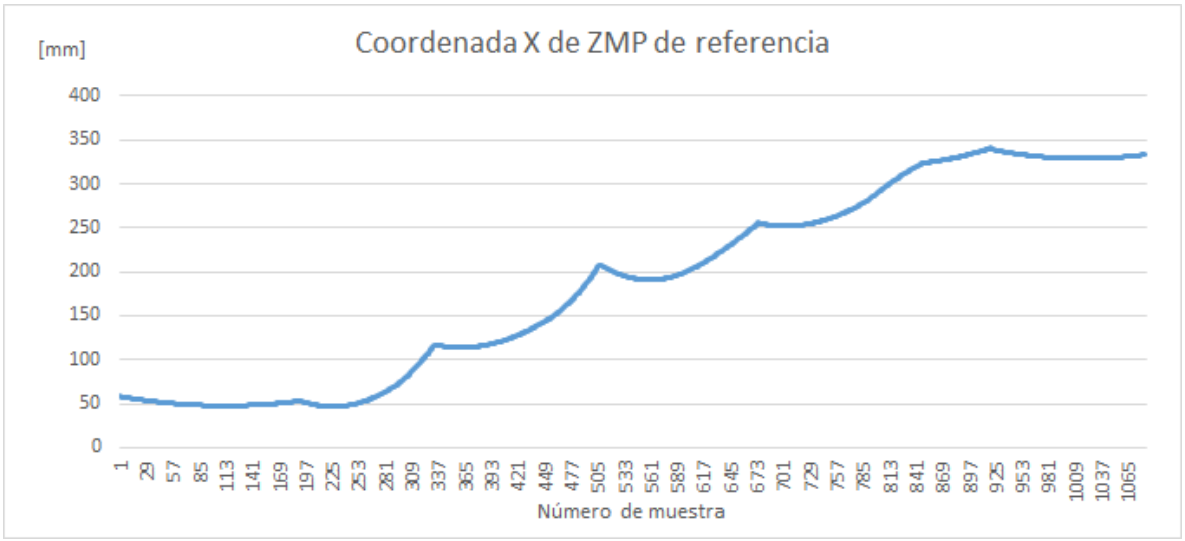


Figura 3.33. Xzmp de referencia durante la marcha

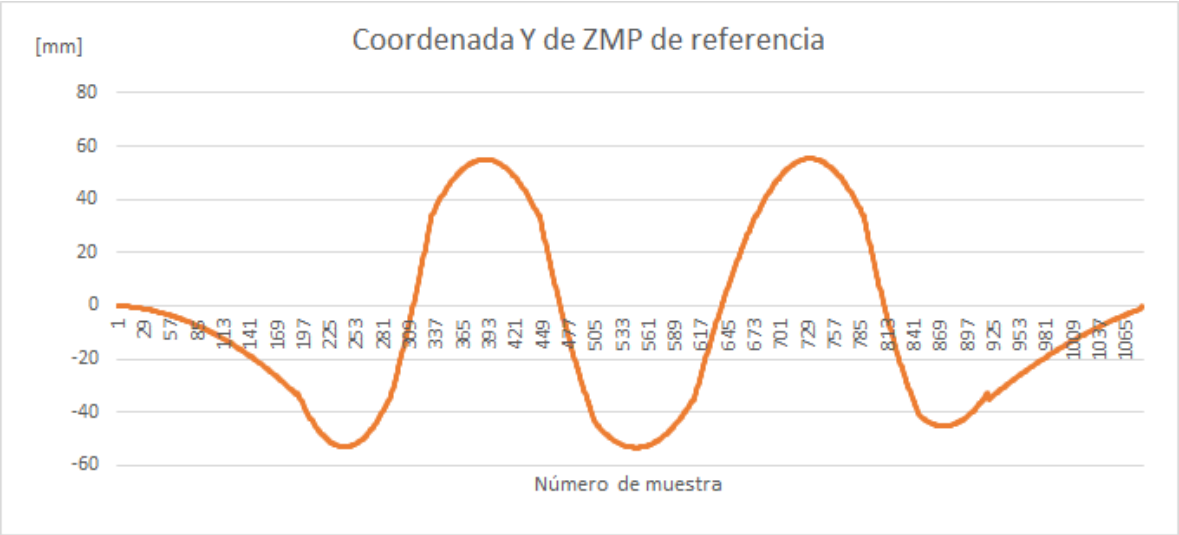


Figura 3.34. Yzmp de referencia durante la marcha

Al considerar que el ZMP es una señal que tiende a variar mucho con cualquier perturbación, se optó por generar una zona de tolerancia alrededor de la referencia, es decir, considerar que el robot está en una posición estable incluso si el sistema de control detecta que los ZMP medidos no son iguales al ZMP de referencia, pero se encuentran dentro de dicha zona. El límite superior de la zona de tolerancia referido es igual a la referencia más 20 mm, el límite inferior, a su vez, tiene 20 mm menos que el valor ideal.

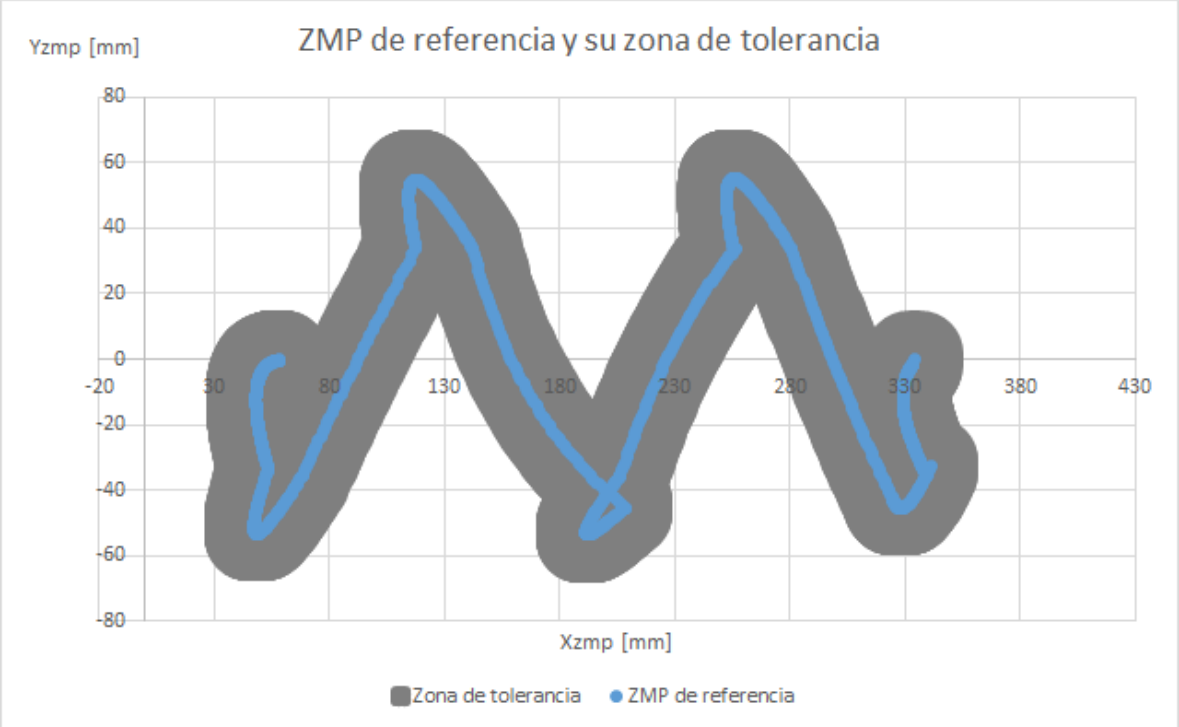


Figura 3.35. Conjunto de puntos de ZMP permisibles durante la marcha

Con el empleo de los parámetros descritos, se obtuvo un valor de referencia, rodeado de una zona de tolerancia, donde el ZMP se considera estable (véase figura 3.35), es decir, si el robot mantiene su ZMP dentro de esta zona, su marcha tenderá a ser estable y, por lo tanto, tendrá un mínimo riesgo de caída.

En términos generales, el ZMP de referencia permitirá establecer un parámetro de control en cada momento de la trayectoria de manera continua, con cambios suaves y minimizando el riesgo de caída del robot al mantenerse en zonas centrales del polígono de soporte.

3.7. Sistema de aterrizaje de planta del pie

Cuando el robot tiende a caerse, una de las acciones que debe realizar es asegurar el contacto de las plantas de sus pies con el suelo, esto permite un correcto cálculo del ZMP ya que es conveniente que todos los sensores de fuerza estén en contacto con el piso [76]. Para asegurar este escenario se propuso realizar un sistema que evite que la planta del pie sólo esté apoyada en su orilla y pueda aprovechar toda la superficie, a esto se le llama aterrizaje del pie.

Para el diseño de este sistema se utilizaron los FSR que, al estar en contacto con la superficie, producen una lectura de potencial eléctrico en el microcontrolador proporcional a la fuerza, en caso contrario, producen una diferencia de potencial menor a 0.1 V. Esta información es suficiente para conocer cuáles zonas del pie están en contacto con el suelo. En la figura 3.36 se simula una situación donde el robot bípedo tiene ambos pies en el suelo, donde los sensores marcados en rojo suponen estar en contacto con el piso

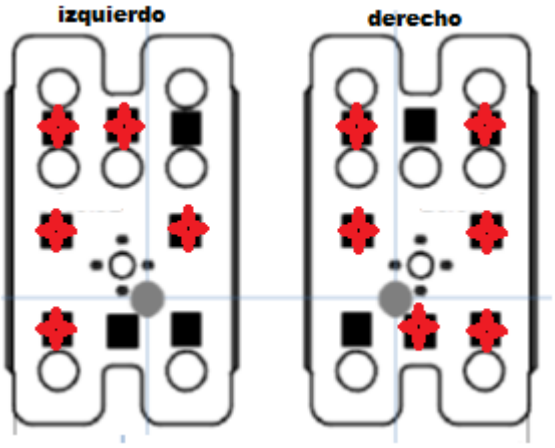


Figura 3.36. Ejemplo de contacto de FSR de los pies en una superficie irregular[5]

En ocasiones, a pesar de que la planta tenga un pleno contacto con el piso, se puede dar que no todos los sensores estén percibiendo fuerzas, esto se debe a que la superficie puede no ser completamente plana y que algunos sensores queden suspendidos.

Debido a lo anterior, se propuso ver a los sensores como un arreglo de filas. De esta manera, al activarse un sensor perteneciente a una fila se consideró que toda ésta se encuentra en contacto con el suelo. Debido a que la planta es una superficie plana, basta con que dos filas

[5] Esquema basado en [4]

estén en contacto para considerar que el robot tiene pleno soporte; así, existirá polígono de soporte y se podrá calcular el ZMP de manera adecuada.

Para la corrección de la postura del pie se propuso una gradual rotación de los servos de los tobillos (θ_{62} y θ_{61}), hasta que al menos dos sensores de diferentes filas estén activos, de esta manera, el peso del robot produce que los sensores que se encuentran levantados entren en contacto con el piso. Como resultado se logró una adecuada posición del pie respecto al piso y una medición más acertada del ZMP.

En la parte superior de la figura 3.37, se aprecia que el pie a la izquierda del observador tiene aproximadamente dos tercios de su superficie levantada. En la imagen inferior se puede apreciar el resultado del sistema de aterrizaje.

Figura 3.37. Aplicación de sistema de aterrizaje

3.8. Sistema de paro en caso de caída

Cuando a un robot, estando en caminata, se le aplica una fuerza demasiado grande caerá y será incapaz de levantarse. Si, además, el sistema de control está activo una vez que el robot ha caído, se generará un comportamiento errante e indeseado cuando el controlador intente corregir los ángulos de los servos sin lograr que las variables del control alcancen la referencia propuesta.

Considerando lo anterior, se ha propuesto implementar un sistema que detecte cuándo el robot ha sufrido una caída y detenga las acciones de control y el ciclo de caminata, mientras envía una señal a la interfaz para que detenga el envío de ángulos e informe al usuario del estado del robot.

El principal problema, sin embargo, es lograr establecer en qué punto se considera que el robot ha caído y cómo el sistema reconocerá automáticamente tal situación. Para solucionarlo, se ha considerado analizar un esquema simulando el estado antes y durante una caída, notando que, conociendo los ángulos de la cadera y las lecturas de los FSR, se puede deducir en cuáles condiciones el robot está volcando.

En la figura 3.38 es visible en el lado izquierdo el robot en posición estable, con todos sus FSR en contacto con el piso y un ángulo de cadera menor o igual a 45. A la derecha, se aprecia un robot en situación de caída, con sus FSR sin contacto y, por lo tanto, sin lectura mientras el ángulo de su cadera es mayor a 45 grados.

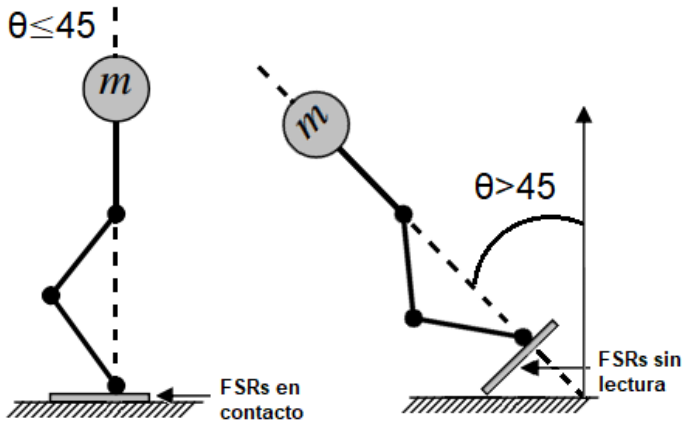


Figura 3.38. Representación de caída del robot[6]

Con ayuda de las anteriores consideraciones se desarrolló un algoritmo que se activa cuando la IMU mida un valor mayor a 45 grados en cualquiera de sus ángulos o cuando los FSR de ambos pies no tengan lecturas, produciendo que el robot adopte una posición fija mientras envía un código por el puerto serial a la interfaz para interrumpir la comunicación.

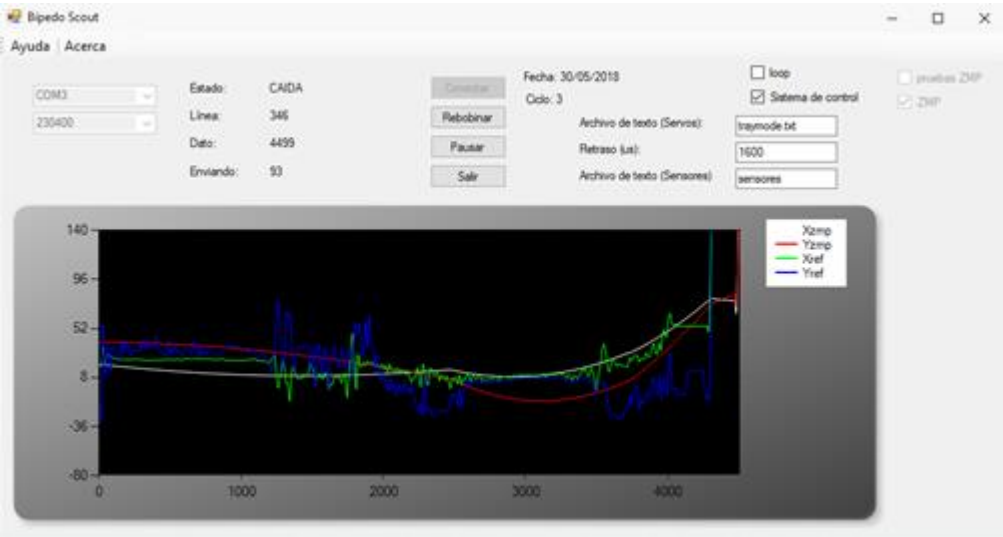


Figura 3.39. Interfaz gráfica detectando caída del robot

En la figura 3.39 se observa la palabra “CAIDA” en el indicador del estado del robot, al tiempo que las lecturas del robot se disparan, indicando que el robot ha volcado.

[6] Esquema basado en [84]

4. Sistema de control de estabilidad de marcha

4.1. Sistemas de control

Los sistemas de control tienen una estrecha relación con el avance de la ciencia y la ingeniería, a grado tal que se han convertido en una parte fundamental en el desarrollo de los sistemas robóticos, de fabricación, de operaciones industriales, de regulación de temperatura, presión, humedad, flujo, etcétera [77], pues ha crecido la necesidad de realizar procesos de una manera más automática y precisa.

Por ejemplo, el sistema de control de un horno eléctrico tiene un termómetro que se encarga de medir la temperatura en su interior, y cuando el controlador detecta una discrepancia (error) entre la referencia o temperatura deseada por el usuario y el valor medido, activa o desactiva, según se necesita, el calefactor hasta que el horno adquiere la temperatura requerida [77].

En esos términos, un sistema de control se define como un conjunto de elementos que actúan conjuntamente para regular la salida del mismo, según lo establece una referencia determinada [78].

Los sistemas de control son diseñados para realizar una tarea con ciertos parámetros de comportamiento. Cuando se inicia un proceso de diseño de control, los más comunes a considerar son los siguientes:

- Estabilidad: Si un sistema de control lineal e invariante en el tiempo es sometido a una perturbación y vuelve a un estado de equilibrio, se le considera estable. Si, por el contrario, el sistema devuelve una salida en forma de oscilaciones que continúan de forma indefinida, se le conoce como un sistema críticamente estable. Finalmente, si la salida del sistema diverge desde una condición inicial establecida, se le considera inestable [77]
- Tiempo de asentamiento: Es el tiempo que se requiere para que el sistema alcance una respuesta dentro de un rango alrededor del valor deseado con una variación de 2% a 5%
- Estado permanente: Se define como el comportamiento del sistema cuando se le ha aplicado una entrada y se ha establecido la salida después del tiempo de asentamiento [77]
- Error en estado permanente: Es la diferencia resultante entre la salida y la entrada de un sistema cuando ha transcurrido el tiempo de asentamiento. Este error representa la precisión del sistema, por lo que, al momento de diseñar un sistema de control, se pretende que éste sea cero [78]
- Sobrepaso máximo: Es el valor mayor del pico de la curva de respuesta, comúnmente se relaciona de manera directa con la estabilidad del sistema
- Tiempo de subida: Es el tiempo necesario para que la respuesta pase del 0% al 100% de su valor final
- Tiempo pico: Es el tiempo requerido para que la respuesta alcance el primer pico de sobrepaso [77]

En la siguiente figura se pueden apreciar los parámetros de comportamiento en la respuesta de un sistema de control estable:

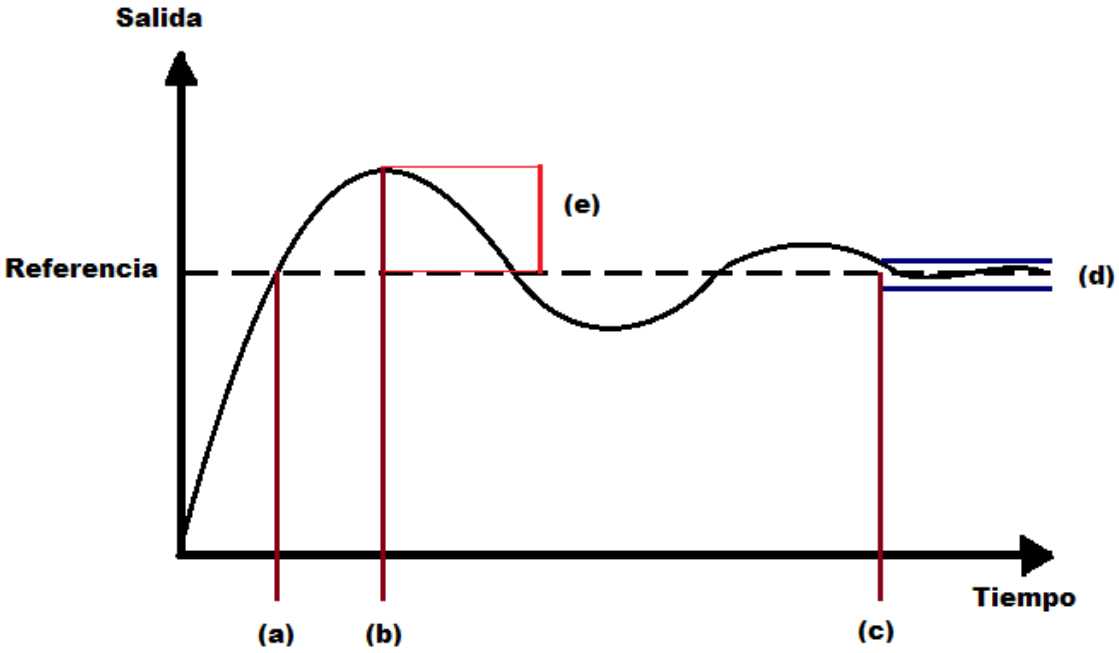


Figura 4.1. Ejemplo de respuesta de un sistema de control estable

(a) Tiempo de subida, (b) tiempo pico, (c) tiempo de asentamiento, (d) respuesta en estado permanente y (e) sobrepaso.

4.1.1. Sistema de lazo abierto y lazo cerrado

Cuando la salida de un sistema no tiene efecto sobre la acción de control se denomina sistema de control de lazo abierto pues la salida no se compara con la entrada de referencia y no existe realimentación (véase figura 4.2). A cada entrada de referencia le corresponde una condición de operación fija.

Ante la presencia de perturbaciones los sistemas en lazo abierto pueden no realizar la tarea designada, ya que su precisión depende de la calibración. En la práctica, el control en lazo abierto únicamente se utiliza si se conoce la relación entrada/salida y cuando no hay perturbaciones internas ni externas [77]. Los sistemas de control que funcionan con una base de tiempo son de lazo abierto, por ejemplo, a un horno de microondas se le puede programar cierta cantidad de tiempo; sin embargo, la variable de control, en este caso la temperatura de los alimentos, no es medida y no realimenta el sistema.

Figura 4.2. Ejemplo de diagrama de bloques de sistema de lazo abierto

Los sistemas de lazo cerrado, por su parte, se realimentan con la señal de error de actuación, que es la diferencia entre la señal de entrada y una función de la señal de salida que, generalmente, es la variable a controlar. El término de lazo cerrado siempre indica el uso de un control realimentado que puede ser la propia señal de salida o una función de la señal de salida y sus derivadas y/o integrales con el fin de reducir el error y llevar la variable de control a un valor deseado [77] (véase figura 4.3).

El uso de la realimentación vuelve la respuesta del sistema relativamente insensible a las perturbaciones externas, lo que produce una salida deseada, diferente a lo que puede suceder en el lazo abierto.

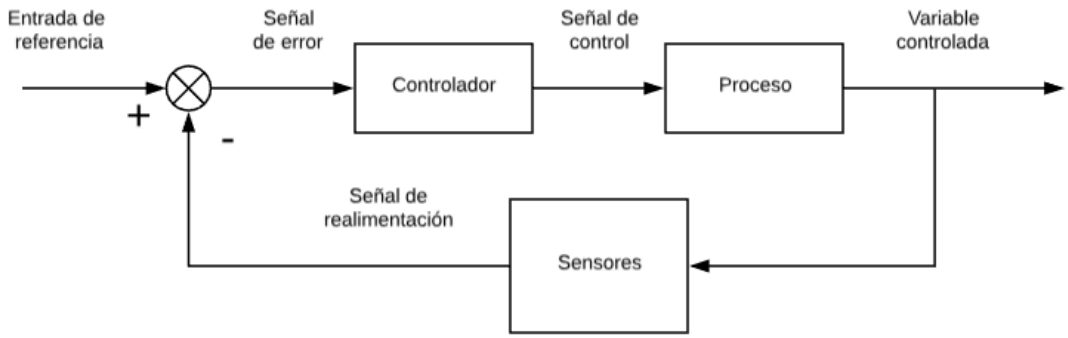


Figura 4.3. Ejemplo de diagrama de bloques de un sistema de lazo cerrado

4.2. Estructura del controlador

Para permitir un análisis ordenado del sistema y lograr una acción de control adecuada, es necesario estipular la configuración del sistema de lazo cerrado, de modo que se puedan unir, de manera adecuada, cada uno de sus elementos.

Al efecto, se propuso como estrategia de control el mantener los ángulos pitch y roll de la cadera en cero y conservar el ZMP del robot dentro del polígono de soporte, con el fin de evitar riesgos de caída durante la caminata. Como resultado, se obtuvieron dos diagramas de bloques que resultan fundamentales al momento de configurar los sistemas de realimentación.

En la figura 4.4 se puede ver el diagrama de bloques del sistema de lazo cerrado correspondiente al sistema de control de los ángulos pitch y roll.

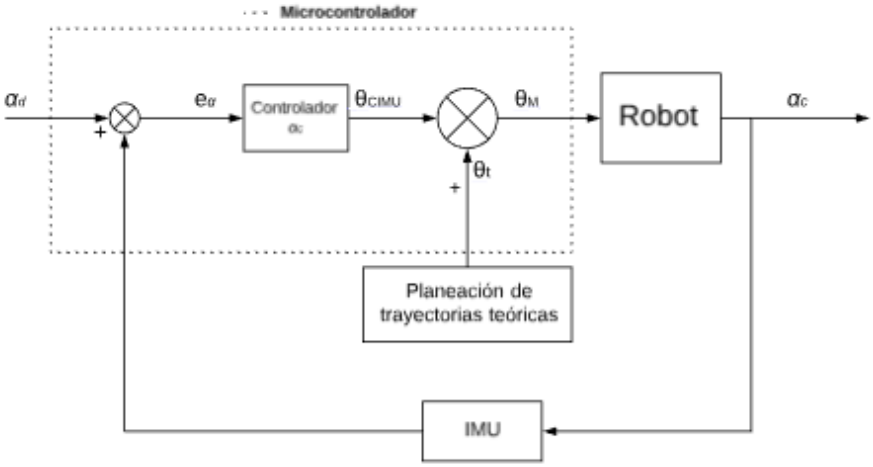


Figura 4.4. Diagrama de bloques del sistema de control de ángulos de la cadera del robot bípedo

Donde:

α_d son los ángulos deseados de la cadera (pitch y roll).

α_c son los ángulos medidos de la cadera (pitch y roll).

e_α es el error entre los ángulos de la cadera deseados y los medidos.

θ_{CIMU} es el conjunto de ángulos de corrección relacionado con e_α.

θt es el conjunto de ángulos teóricos obtenidos a partir del modelo simplificado carro-mesa y enviados a través de la interfaz gráfica al robot.

θ_M son los ángulos que se envían a cada uno de los motores del robot bípedo.

Atendiendo al diagrama, los ángulos teóricos de una caminata son enviados desde la interfaz gráfica al microcontrolador, esto produce en el robot una postura con sus ángulos de cadera correspondientes, los cuales son medidos por la IMU. Más adelante, estos datos son comparados con valores deseados, lo cual da como resultado un error que permite al microcontrolador calcular ángulos de corrección de la cadera y sumarlos a los ángulos teóricos antes mencionados y, de esta manera, lograr que la cadera permanezca en posición horizontal durante la caminata.

En el caso del sistema de control del ZMP, el microcontrolador calcula el ZMP por medio de los FSR, por lo que el diagrama de bloques de este sistema resulta de la siguiente manera:

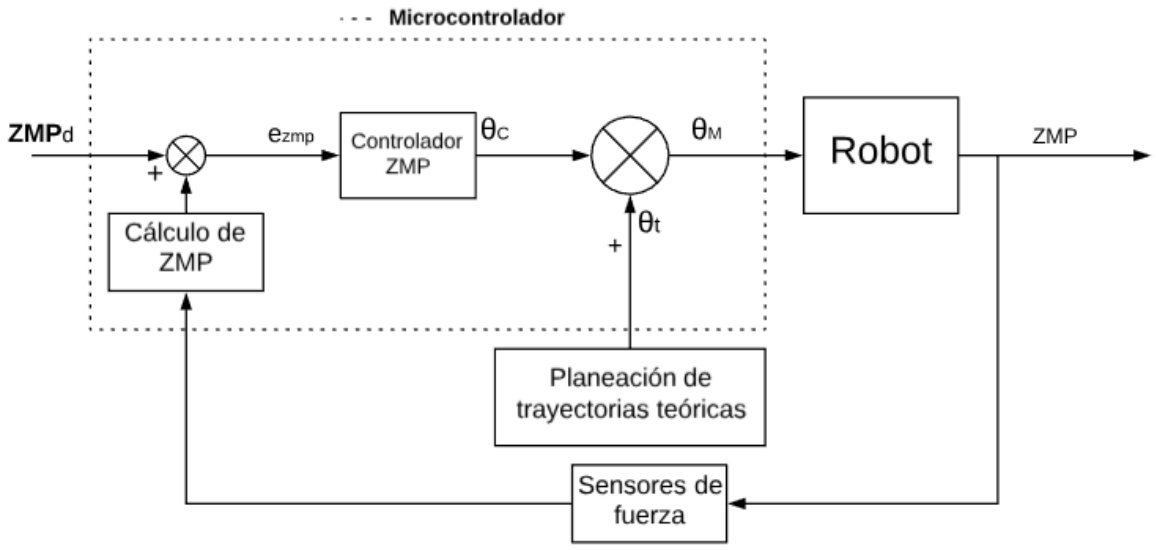


Figura 4.5. Diagrama de bloques del sistema de control del ZMP del robot bípedo

Donde:

ZMP_d son las coordenadas deseadas del ZMP (X e Y).

e_{zmp} es el error entre las coordenadas del ZMP y las calculadas.

θ_C es el conjunto de ángulos de corrección relacionado con e_{zmp}.

θt es el conjunto de ángulos teóricos obtenidos a partir del modelo simplificado carro-mesa y enviados a través de la interfaz gráfica al robot.

θ_M son los ángulos que se envían a cada uno de los motores del robot bípedo.

En este caso, el microcontrolador recibe los ángulos teóricos de un ciclo de marcha, los cuales son enviados desde la interfaz gráfica, esto produce en el robot una postura con su respectivo ZMP, valor que es calculado a través de los datos provenientes de los FSR. Posteriormente, se mide el error entre las coordenadas del ZMP (X, Y) y los valores de referencia y se calculan los ángulos de corrección que permitan al robot eliminar dicho error, adoptando así una nueva posición más estable. Este ciclo se repite una y otra vez durante toda la marcha.

4.3. Análisis de control de postura.

Puesto que algunos servos tienen mayor impacto, por ejemplo, en el cálculo de la coordenada X del ZMP o en el ángulo pitch de la cadera, resulta indispensable conocer la dirección del giro de cada motor y su efecto en las variables de control con el fin de diseñar una acción de control adecuada. Para ello, el método empleado en este escrito consistió en variar el ángulo de un servo en una fase determinada de la caminata y analizar qué variables se perturbaron más por este cambio. Así, al variar el servomotor 32, de 45 a 75 grados durante la fase de SSD, se afectaron principalmente las variables pitch y Xzmp, mientras que roll y la coordenada Yzmp permanecieron prácticamente inalteradas.

La figura 4.6 ilustra la relación entre del ángulo θ32 y el ángulo pitch mientras el ángulo roll se mantiene prácticamente invariante. Por otro lado, la figura 4.7 muestra la correspondencia de θ32 y la coordenada Xzmp, mientras Yzmp no es afectada.

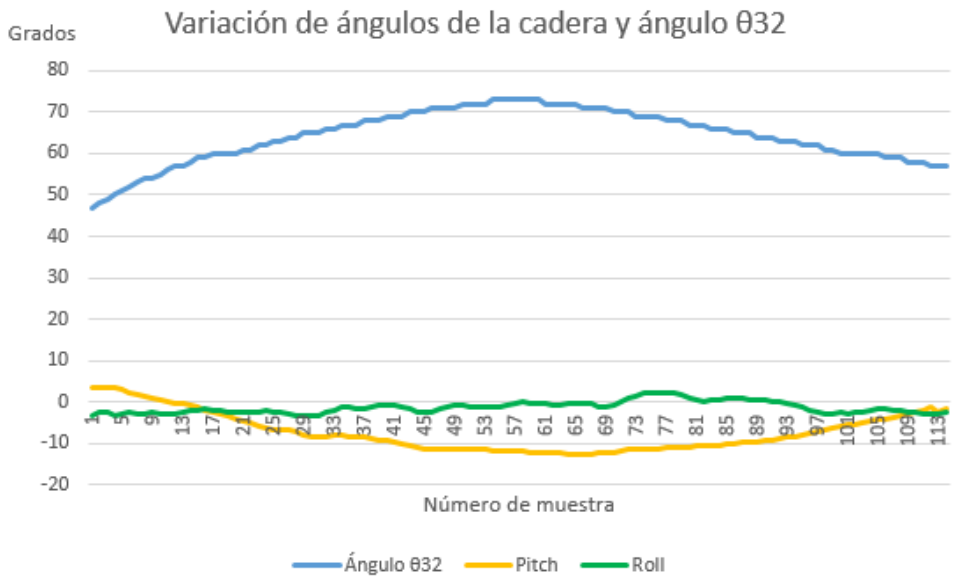


Figura 4.6. Efecto del giro del ángulo del motor 32 en los ángulos pitch y roll en SSD

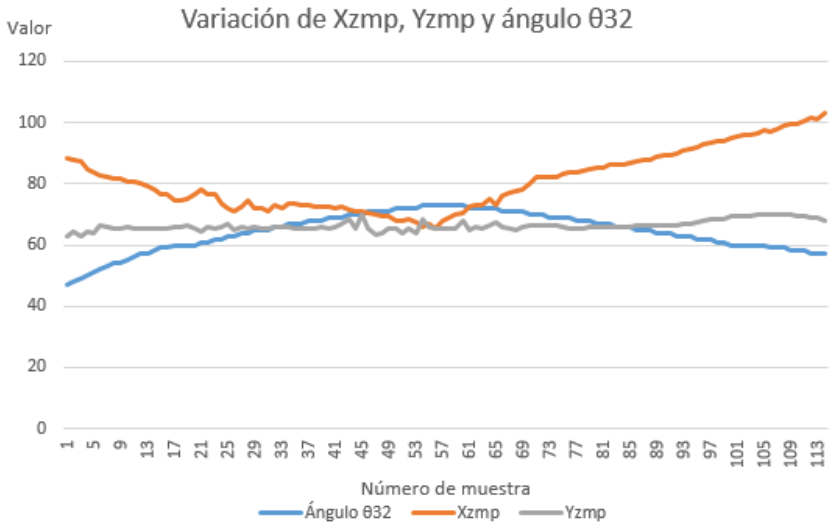


Figura 4.7. Efecto del giro del ángulo del motor 32 en las coordenadas del ZMP en SSD

Al aplicar el método antes mencionado, en cada una de las fases, se obtuvo una relación entre los motores y su impacto en las variables de control (véase tabla L).

Tabla L. . Influencia de los servos en las variables de control en diferentes fases cinemáticas

Segmento	Servo	SDA	SSI	SSD	SDI	SDD
CADERA	12	Yaw	-	Yaw	Yaw	Yaw
	11	Yaw	Yaw	-	Yaw	Yaw
	22	Roll, Y_{ZMP}	-	Roll, Y_{ZMP}	Roll, Y_{ZMP}	Roll, Y_{ZMP}
	21	Roll, Y_{ZMP}	Roll, Y_{ZMP}	-	Roll, Y_{ZMP}	Roll, Y_{ZMP}
	32	Pitch, Roll, X_{ZMP}, Y_{ZMP}	-	Pitch, X_{ZMP}	Pitch, Roll, X_{ZMP}, Y_{ZMP}	Pitch, Roll, X_{ZMP}, Y_{ZMP}
	31	Pitch, Roll, X_{ZMP}, Y_{ZMP}	Pitch, X_{ZMP}	-	Pitch, Roll, X_{ZMP}, Y_{ZMP}	Pitch, Roll, X_{ZMP}, Y_{ZMP}
RODILLAS	42	Pitch, Roll, X_{ZMP}, Y_{ZMP}	-	Pitch, X_{ZMP}	Pitch, Roll, X_{ZMP}, Y_{ZMP}	Pitch, Roll, X_{ZMP}, Y_{ZMP}
	41	Pitch, Roll, X_{ZMP}, Y_{ZMP}	Pitch, X_{ZMP}	-	Pitch, Roll, X_{ZMP}, Y_{ZMP}	Pitch, Roll, X_{ZMP}, Y_{ZMP}
TOBILLOS	52	Pitch, Roll, X_{ZMP}, Y_{ZMP}	-	Pitch, X_{ZMP}	Pitch, Roll, X_{ZMP}, Y_{ZMP}	Pitch, Roll, X_{ZMP}, Y_{ZMP}
	51	Pitch, Roll, X_{ZMP}, Y_{ZMP}	Pitch, X_{ZMP}	-	Pitch, Roll, X_{ZMP}, Y_{ZMP}	Pitch, Roll, X_{ZMP}, Y_{ZMP}
	62	Roll, Y_{ZMP}	-	Roll, Y_{ZMP}	Roll, Y_{ZMP}	Roll, Y_{ZMP}
	61	Roll, Y_{ZMP}	Roll, Y_{ZMP}	-	Roll, Y_{ZMP}	Roll, Y_{ZMP}

Como se observa en la tabla, si el cambio de giro de un motor modifica la variable pitch, en mayor o menor medida también modifica la coordenada X del ZMP. La misma relación se advierte entre el ángulo roll y la coordenada Y del ZMP.

En el caso de la fase de SD, hay una mayor intervención de los servos en el cambio de las variables de control. Por ejemplo, los motores de las rodillas modifican todos los parámetros de control, mientras que en SS únicamente cambian el ángulo pitch y la coordenada Xzmp. Lo anterior, se debe principalmente a que el soporte doble representa una cadena cinemática cerrada, es decir, que tiene dos contactos en el suelo. El análisis de la fase SDA, desde los planos sagital y coronal, dan más información al respecto.

4.3.1. Análisis de control en SDA desde plano sagital

Si el robot se encuentra en una plataforma inclinada se genera también una inclinación de su cadera y una desviación de la coordenada X del ZMP. Al ocurrir, el controlador corrige la postura hasta que el error entre los valores de referencia, y los valores medidos, son igual a cero.

En la figura 4.8, se puede ver a la izquierda el sistema en una posición indeseable, a la derecha se puede ver la postura corregida con las variables de control igualadas a la referencia.

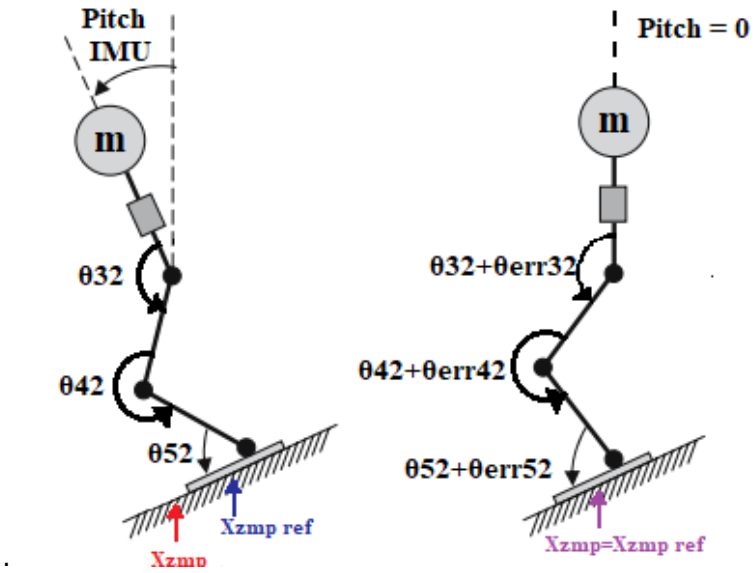


Figura 4.8. Corrección de postura de robot bípedo en plano sagital[7]

4.3.2. Análisis de control en SDA desde plano coronal

Estando el robot en la fase SDA, sobre una superficie inclinada, se modifica el ángulo Roll de la cadera y se desvía la coordenada Yzmp de su valor deseado. En este caso, la acción de control de la postura difiere de la analizada en el plano sagital ya que es necesario modificar las longitudes de los tobillos a la cadera mientras se corrigen los ángulos de los servomotores 62 y 61; ello permitirá mantener al ZMP dentro del polígono de soporte e igualar a cero el ángulo roll.

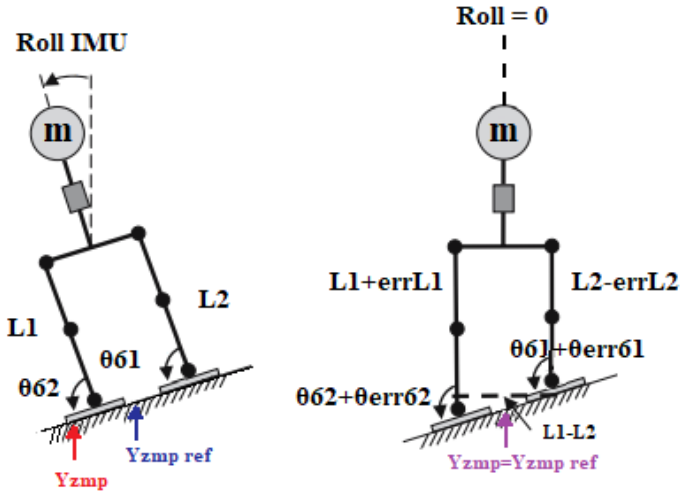


Figura 4.9. Corrección de postura de robot bípedo en plano coronal[8]

[7] Esquema basado en [84]
[8] Esquema basado en [84]

Lo anterior significa que, para controlar las variables roll e Yzmp en SDA, es necesario modificar los ángulos de los tobillos y estirar una pierna mientras se encoge la otra. El algoritmo empleado para modificar las longitudes de las piernas está basado en la ecuación (9) que fue utilizada para una elevación del pie en SS. Así, el controlador medirá la diferencia entre las variables calculadas y las de referencia para obtener un error que será el que varíe las longitudes de las piernas y modifique los ángulos de los tobillos para lograr una postura estable (véase figura 4.9).

4.4. Controladores PID

La familia de controladores PID pueden dividirse en sus tres diferentes acciones de control: proporcional (P), integral (I) y derivativa (D). La combinación de estos controladores da como resultado los llamados PI, PD y PID [79].

El controlador P entrega una salida que es directamente proporcional al error, es decir:

$$u(t) = K_p e(t)$$

Donde *Kp* es una ganancia proporcional ajustable, *u(t)* es la salida del controlador y *e(t)* es la señal del error.

El controlador P puede controlar cualquier sistema estable, sin embargo, se desempeña de manera limitada y suele presentar un error en estado permanente [79].

El controlador I permite obtener una salida que es igual al error acumulado multiplicado por una ganancia Ki, es decir:

$$u(t) = K_i \int_0^t e(\tau)d\tau$$

Donde la ganancia Ki es una constante de integración que indica la velocidad con la que se repite la acción proporcional.

Aunque es un modo de control lento, la principal aportación a los sistemas de control es que corrige el error en estado permanente hasta igualarlo a cero [79].

El controlador PI combina el controlador proporcional, permitiendo una rápida acción de control, y el integral que consiste en la constante corrección del error hasta igualarlo a cero. Se define mediante la ecuación:

$$u(t) = K_p e(t) + \frac{K_p}{Ti} \int_0^t e(\tau)d\tau$$

Donde Ti es el tiempo que ajusta la acción integral. Nótese que Kp entre Ti es igual a Ki [79].

El controlador PD logra unir una acción de control derivativa con la proporcional, lo que permite obtener un controlador que responda a la velocidad del cambio del error y corregirlo antes de que se incremente demasiado. Si bien el controlador PD no modifica de manera directa el error en estado permanente, sí añade un amortiguamiento al sistema que permite una mayor

ganancia en la acción proporcional y una mejor precisión en estado permanente. Su principal desventaja es la sensibilidad a las señales de ruido que pueden llegar a saturar el actuador.

La ecuación del controlador PD es:

$$u(t) = K_p e(t) + K_d \frac{de(t)}{dt}$$

Donde Kd es una constante de derivación que permite que la respuesta de la acción proporcional se multiplique, sin esperar a que el error lo haga.

La acción de control derivativa nunca se utiliza por sí sola, debido a que únicamente resulta eficaz durante la primera respuesta del controlador ante una perturbación [79].

El controlador PID ejerce las tres acciones de control (proporcional-integral-derivativa) y reúne sus ventajas para lograr un control más robusto. Con una alta velocidad de respuesta, un amortiguamiento añadido y un error nulo en estado permanente. Su ecuación característica se define como:

$$u(t) = K_p e(t) + K_d \frac{de(t)}{dt} + K_i \int_0^t e(\tau) d\tau$$

El principal desafío al implementar un controlador PID radica en la obtención de las ganancias Kp, Ki y Kd, para que el sistema se comporte de manera deseada. Uno de los métodos utilizados para conocer dichas ganancias se explica a continuación [79].

4.4.1. Procedimiento heurístico de ajuste de parámetros Kp, Ki y Kd.

El ajuste de las variables PID se realizó mediante un método heurístico de sintonización utilizado frecuentemente en la solución de diferentes problemas de control. Es un método sencillo que permite encontrar parámetros útiles del controlador PID a través de prueba y error, sin necesidad de conocer el modelo físico del sistema [80].

A pesar de no ser el método más eficiente, permite conocer un conjunto de ganancias Kp, Ki y Kd que logran controlar el sistema.

La técnica consiste en:

- Configurar el controlador PID como un controlador P, es decir, igualar las ganancias Kd y Ki a cero
- Incrementar la Kp hasta provocar que el sistema tenga oscilaciones autosustentables
- Establecer la Kp como la mitad del valor obtenido en el paso anterior.
- Aumentar Ki hasta lograr que el sistema elimine el error en estado permanente en un tiempo propuesto
- Aumentar Kd hasta que el sistema logre su recuperación de una perturbación brusca [80]

Este método se aplicó tomando en cuenta los efectos de aumentar cada ganancia de manera independiente de acuerdo con la tabla M. De esta manera, si se requiere disminuir el tiempo de subida, sin aumentar el tiempo de asentamiento, se deberá incrementar el valor de Kp.

Tabla M. Influencia de las ganancias del controlador PID en comportamiento del sistema [81]

Parámetro	Tiempo de subida	Sobrepaso	Tiempo de asentamiento	Error en estado permanente	Estabilidad
Kp	Disminuye	Incrementa	Cambia ligeramente	Disminuye	Se degrada
Ki	Disminuye	Incrementa	Incrementa	Se elimina	Se degrada
Kd	Cambia ligeramente	Disminuye	Disminuye	Teóricamente sin efecto	Mejora si Kd no es muy grande

4.5. Sistema de control de ángulos de cadera

El objetivo principal del sistema de control de los ángulos de la cadera es el de mantener la cadera horizontal, por lo que los ángulos pitch y roll deben ser iguales a cero. Lo anterior es posible gracias a un controlador PID que modifica su comportamiento de acuerdo con la fase cinemática en la que el robot se encuentra y por la activación y desactivación de acciones de control al inicio de cada fase cinemática. Al existir un ángulo diferente a cero, el controlador calcula un ángulo de corrección de acuerdo las fórmulas:

$$\theta c1i = Kp_i\,(e_{PITCH}) \pm Ki_i \sum_0^t e_{PITCH}(t)\Delta t \pm Kd_i(\Delta e_{PITCH})$$

$$\theta c1j = Kp_j\,(e_{ROLL}) \pm Ki_j \sum_0^t e_{ROLL}(t)\Delta t \pm Kd_j(\Delta e_{ROLL})$$

Donde:

θc1i es el ángulo de corrección del motor relacionado con el ángulo pitch de la cadera.

Kpi, Kii y *Kdi* son las ganancias del controlador PID de la variable pitch (proporcional, integral y derivativa), obtenidas para una específica fase y un determinado motor.

e_{PITCH} es el error de la variable pitch.

θc1j es el ángulo de corrección del motor relacionado con el ángulo roll de la cadera.

Kpj, Kij y *Kdj* son las ganancias del controlador PID de la variable roll (proporcional, integral y derivativa), obtenidas para una específica fase y un determinado motor.

e_{ROLL} es el error de la variable roll.

t es el tiempo de muestreo.

De esta manera, y considerando que hay servos que modifican el valor de pitch y roll de la cadera, los ángulos que los motores seguirán mientras el sistema de control esté activo. Se pueden representar con la siguiente ecuación:

$$\theta ni_{IMU} = \theta t_i + \theta c1i_i + \theta c1j_i$$

Donde:

θni es el ángulo de la articulación 'ni' que se obtiene del sistema de control de los ángulos de la cadera.

θti es el ángulo teórico de la articulación 'ni' generado a partir del modelo carro-mesa.

El conjunto de ganancias del controlador PID, que gobiernan el sistema de control en cada momento de la fase, son visibles en la tabla N.

Tabla N. Ganancias PID del sistema de control de los ángulos de la cadera.

θni	**SDA, SDD,SDI**		**SSI**		**SSD**	
IMU	Kpi,Kii,Kdi	Kpj,Kij,Kdj	Kpi,Kii,Kdi	Kpj,Kij,Kdj	Kpi,Kii,Kdi	Kpj,Kij,Kdj
32	0.4,0.02,0.3	0.045,0.0001,0.001	-	-	0.5,0.02,0.2	-
31	0.4,0.02,0.3	0.045,0.0001,0.001	0.45,0.01, 0.1	-	-	-
42	-	0.9, 0.0002, 0.002	-	-	-	-
41	-	0.9, 0.0002, 0.002	-	-	-	-
52	-	0.045,0.0001,0.001	-	-	-	-
51	-	0.045,0.0001,0.001	-	-	-	-
62	-	0.09,0.00025,0.15	-	-	-	0.17,0.0007,0.11
61	-	0.09,0.00025,0.15	-	0.25, 0.0007, 0.1	-	-

De la anterior tabla destaca que en el caso de las fases SSI y SSD es necesario únicamente un motor para controlar cada ángulo de la cadera.

Cada que se inicia una fase cinemática se calcula un conjunto de ángulos de corrección que va cambiando mientras la marcha se está ejecutando. Al finalizar dicha fase, estos ángulos son atenuados mientras unos nuevos ángulos de la siguiente fase son calculados.

A fin de explicar mejor el método anteriormente descrito, se hizo una prueba para conocer la influencia de cada fase durante la marcha sumando los ángulos de corrección de cada una de las juntas. En el resultado se observa, en el cambio de fase, un gradiente en la participación de las fases involucradas con el fin de que se tenga un control específico de cada postura y sus ángulos de la cadera, además, una breve participación cooperativa para evitar dejar al robot sin acción de control en el cambio de fase cinemática, cuando se activan y desactivan los sistemas específicos (véase figura 4.10).

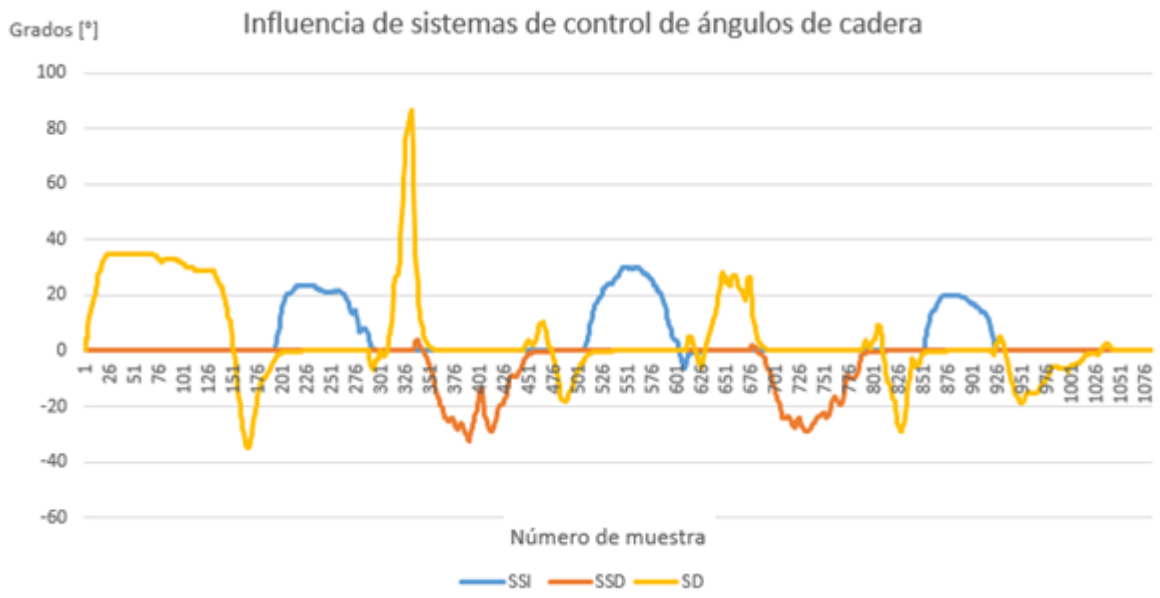


Figura 4.10. Influencia de sistemas de control específicos de cada fase en ángulos de corrección relacionados al control de ángulos pitch y roll durante la marcha

4.6. Sistema de control del ZMP

Con el propósito de promover una caminata estable, el sistema de control del ZMP tiene como objetivo medir fuerzas a través de los FSR para calcular el ZMP y compararlo con la referencia mencionada en el tema 3.6.

Una vez calculado el ZMP, el controlador comparará sus coordenadas (X, Y) con las coordenadas del ZMP de referencia. Si existe una discrepancia entre ellas, calculará un ángulo de corrección de conformidad a las siguientes formulas:

$$\theta c2x = Kp_x\left(e_{Xzmp}\right) \pm Ki_x \sum_{0}^{t} e_{Xzmp}(t)\Delta t \pm Kd_x(\Delta e_{Xzmp})$$

$$\theta c2y = Kp_y\left(e_{Yzmp}\right) \pm Ki_y \sum_{0}^{t} e_{Yzmp}(t)\Delta t \pm Kd_y(\Delta e_{Yzmp})$$

Donde:

θc2x y *θc2y* son los ángulos de corrección del servo con relación a las coordenadas del ZMP X e Y, respectivamente.

Kpx, Kix y Kdx son las ganancias del controlador PID relacionado con el control del Xzmp (proporcional, integral y derivativa), obtenidas para una específica fase y un determinado motor.

e_{Xzmp} es el error entre el Xzmp medido y el Xzmp referencial.

Kpy, Kiy y Kdy son las ganancias del controlador PID relacionado con el control del Yzmp (proporcional, integral y derivativa), obtenidas para una específica fase y un determinado motor.

e_{Yzmp} es el error entre el Yzmp medido y el Yzmp referencial.

t es el tiempo de muestreo.

En semejanza al sistema de control de los ángulos de la cadera, existen servos que intervienen en las dos coordenadas del ZMP en fases de soporte doble. La ecuación que representa a los ángulos totales de los servomotores es:

$$\theta ni_{ZMP} = \theta t_i + \theta c2x_i + \theta c2y_i$$

Donde:

θni_{ZMP} es el ángulo que, de la articulación 'ni', se obtiene del sistema de control del ZMP.

θti es el ángulo teórico de la articulación 'ni' generado a partir del modelo carro-mesa.

Para observar qué ganancias del controlador PID fueron sintonizadas para establecer un sistema de control funcional se desarrolló la siguiente tabla:

Tabla O. Ganancias PID del sistema de control del ZMP en SD.

θni ZMP	**SDA**		**SDD**		**SDI**	
	Kpx,Kix,Kdx	Kpy,Kiy,Kdy	Kpx,Kix,Kdx	Kpy,Kiy,Kdy	Kpx,Kix,Kdx	Kpy,Kiy,Kdy
32		0.005, 0.0005, 0.02		0.003, 0.0001, 0.0001		0.006, 0.0001, 0.0001
31		0.005, 0.0005, 0.02		0.003, 0.0001, 0.0001		0.006, 0.0001, 0.0001
42	0.05, 0.005, 0.1	0.01, 0.001, 0.04	0.01, 0.001, 0.01	0.006, 0.0002, 0.0002	0.025, 0.001, 0.01	0.012, 0.0002, 0.0002
41	0.05, 0.005, 0.1	0.09, 0.0002, 0.0002	0.01, 0.001, 0.01	0.006, 0.0002, 0.0002	0.025, 0.001, 0.01	0.012, 0.0002, 0.0002
52	0.05, 0.005, 0.1	0.005, 0.0005, 0.02	0.01, 0.001, 0.01	0.003, 0.0001, 0.0001	0.025, 0.001, 0.01	0.006, 0.0001, 0.0001
51	0.05, 0.005, 0.1	0.005, 0.0005, 0.02	0.01, 0.001, 0.01	0.003, 0.0001, 0.0001	0.025, 0.001, 0.01	0.006, 0.0001, 0.0001
62		0.03, 0.0025, 0.02		0.018, 0.00005, 0.0001		0.036, 0.0005, 0.0001
61		0.03, 0.0025, 0.02		0.018, 0.00005, 0.0001		0.036, 0.0005, 0.0001

Es importante destacar que, a pesar de la similitud del sistema de control del ZMP al de los ángulos de la cadera, fue necesario manejar diferentes valores de ganancias en cada fase de soporte doble (SDA, SDD, SDI), debido a que el cálculo del ZMP, como ya se ha mencionado, depende de la longitud de paso.

Debido a que el cálculo del ZMP da como resultado una señal con mucho ruido, es necesario utilizar ganancias (principalmente las derivativas) con valores pequeños en comparación a la ganancia proporcional.

En el caso de las fases SDD y SDI, se generan ganancias menores debido a que el sistema tiende a ser más estable gracias a la longitud de paso.

Finalmente, en las fases de SS solamente se utilizaron tres motores en el control de las variables Xzmp e Yzmp, por lo que las ganancias son mayores que en SD. Véase al respecto la tabla P.

Tabla P. Ganancias PID del sistema de control del ZMP en SS.

θni ZMP	**SSI**		**SSD**	
	Kpx,Kix,Kdx	Kpy,Kiy,Kdy	Kpx,Kix,Kdx	Kpy,Kiy,Kdy
42	0.1, 0.005, 0.01	-	-	-
41	-	-	0.1, 0.005, 0.01	-
52	0.1, 0.005, 0.01	-	-	-
51	-	-	0.1, 0.005, 0.01	-
62	-	0.06, 0.0003, 0.01	-	-
61	-	-	-	0.04, 0.0002, 0.01

El controlador también considera la fase cinemática de la caminata y activa o desactiva acciones de control para tener un algoritmo específico de cada etapa y lograr una estabilidad en cada momento de la caminata.

Con el fin de conocer la influencia de cada sistema de control de ZMP específico en cada fase, comprobar que se mantienen activos en su fase correspondiente y se desactivan gradualmente en el cambio de fase, se realizó una prueba sumando los ángulos de corrección de cada motor. En la figura 4.11 se puede apreciar que en los cambios de fase existe una conmutación entre

los sistemas de control específicos, de manera semejante al caso del sistema de control de ángulos de cadera.

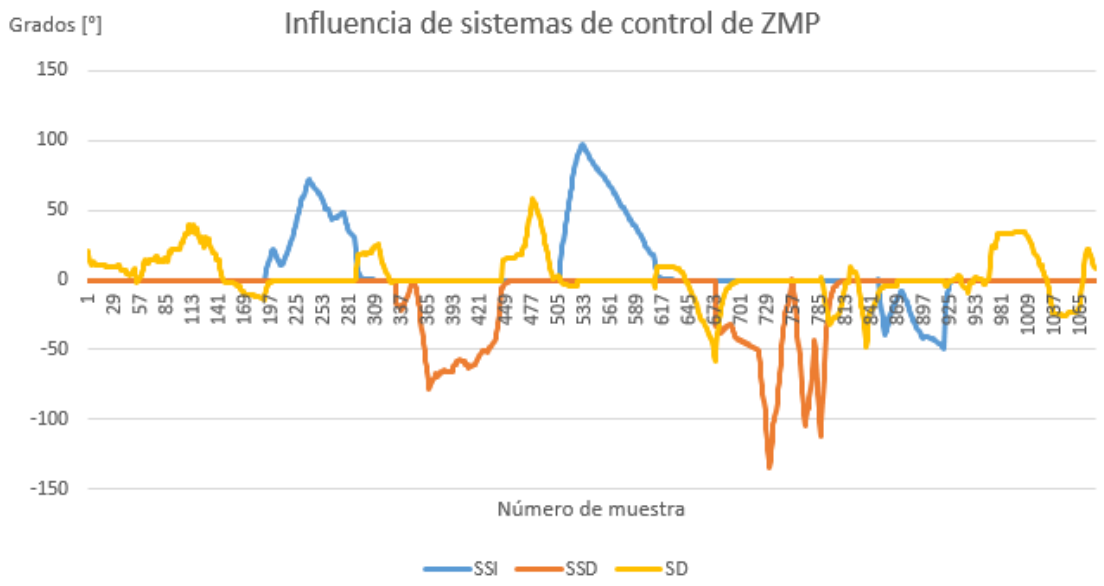


Figura 4.11. Influencia de sistemas de control específicos de cada fase en ángulos de corrección relacionados al control del ZMP durante la marcha

5. Análisis de resultados

5.1. Resultados del sistema de control de ángulos de cadera

Las pruebas ejecutadas, para comprobar el funcionamiento del sistema de control de ángulos pitch y roll de la cadera, se realizaron sobre una superficie de madera –elaborada por alumnos de servicio social- que puede variar la inclinación de su pendiente. El método de experimentación constó de cinco ensayos. En uno se colocó la base en posición horizontal y, en los cuatro restantes, se inclinó la base 5 grados en roll o 5 grados en pitch en ambos sentidos. Además, en algunas ocasiones, se generaron pequeñas perturbaciones al robot a través de ligeros empujones con la mano durante diferentes fases de la caminata.

Los cinco grados arriba señalados se eligieron considerando que el robot pudiera caminar en una rampa común para silla de ruedas cuya inclinación máxima es, en México, de 8% o 4.57 grados de acuerdo con la norma NOM-030-SSA3-2013, publicada en el Diario Oficial de la Federación el día 25 de julio de 2013 [82].

A continuación, se muestran los resultados obtenidos en cada ensayo y un análisis sobre lo observado.

5.1.1. Prueba en superficie horizontal

La prueba se realizó en el banco de pruebas sin ninguna inclinación, por lo que los ángulos medidos sin el sistema de control, fueron similares a los señalados en el tema 3.5.1 sobre **medición de ángulos de cadera durante marcha**. La prueba se ejecutó con dos perturbaciones en SDA mientras el sistema de control estaba activo, una en la muestra 50 y otra más en la muestra 100. La importancia de esta prueba radica en el análisis del comportamiento del sistema de lazo cerrado en condiciones ideales y compararlo con los datos de la caminata en lazo abierto.

En general, el sistema de control logró que el robot se posicionara por debajo de los 2 grados de diferencia respecto a la referencia, mientras que, sin el sistema de control, el error fue mayor (figuras 5.1 y 5.2). En términos generales, el ángulo roll no se vio afectado por las perturbaciones.

Figura 5.1. Ángulos pitch de la cadera durante la caminata sobre superficie horizontal

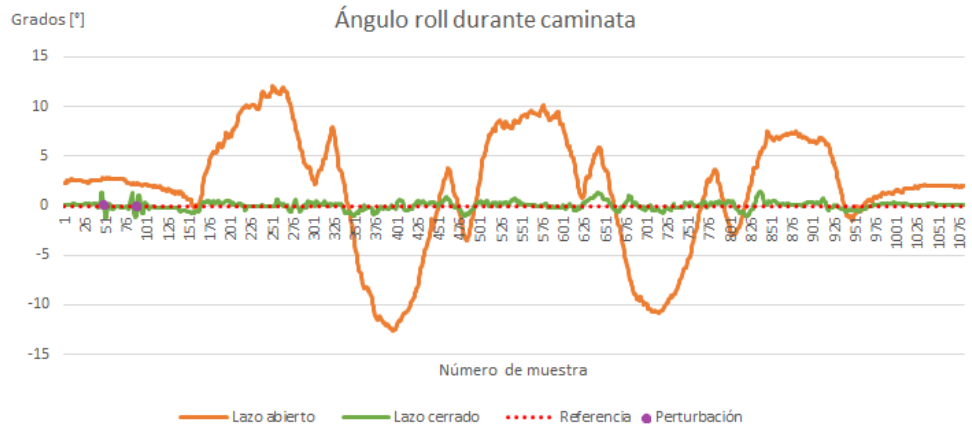


Figura 5.2. Ángulos roll de la cadera durante la caminata sobre superficie horizontal

5.1.2. Prueba con inclinación pitch sentido negativo.

La siguiente prueba se realizó aumentando la pendiente de la base de pruebas cinco grados en dirección de pitch, pero en sentido contrario, con el fin de inclinar al robot hacia enfrente, además, durante la fase SSI (1) se provocó una pequeña perturbación en el giro correspondiente a roll. Como resultado, se tuvo un comportamiento más alejado de la referencia, aunque dentro de lo esperado.

En la figura 5.3 se pueden apreciar dos aspectos. El primero es que el sistema de control modificó correctamente la postura del robot en el caso de la perturbación y, el ángulo pitch, se mantuvo dentro de un error de 2 grados en ambos sentidos. Por otro lado, la inclinación en pitch provocó un aumento en las mediciones de la IMU por lo cual dicho ángulo de la cadera superó los 11 grados.

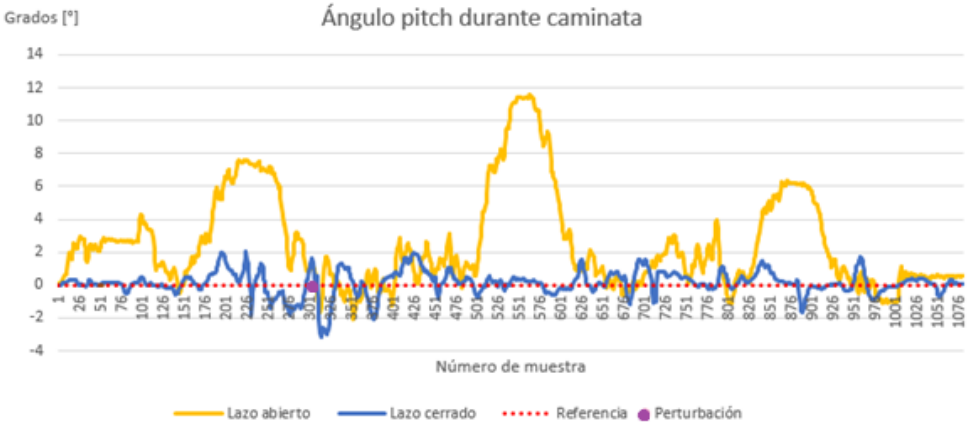


Figura 5.3. Ángulos pitch de la cadera durante la caminata sobre superficie inclinada menos cinco grados en dirección pitch

La perturbación en roll provocó que el robot alcanzará hasta los cuatro grados mientras el sistema de control se encontraba encendido; a pesar de esto, logró recuperarse satisfactoriamente y continuar la marcha. El ángulo en roll, sin sistema de control, se comportó

de manera similar al analizado en la superficie horizontal, al no ser afectado por la inclinación en pitch (véase figura 5.4).

Figura 5.4. Ángulos roll de la cadera durante la caminata sobre superficie inclinada menos cinco grados en dirección pitch

5.1.3. Prueba con inclinación pitch sentido positivo.

En el último experimento, donde se comprobó el funcionamiento del sistema de control en pitch, se incrementó la pendiente de la base cinco grados en sentido positivo y dirección de pitch, esto provocó que el bípedo se inclinara en sentido contrario de su marcha. El robot realizó un ciclo de caminata con el sistema de control activo mientras se generó una perturbación en la muestra 445, justo en el cambio de fase de SSD (1) a SDI (1), con el fin de corroborar la conmutación de los sistemas de control específicos de cada fase. Después se desactivó el sistema de control para repetir la prueba sin perturbación.

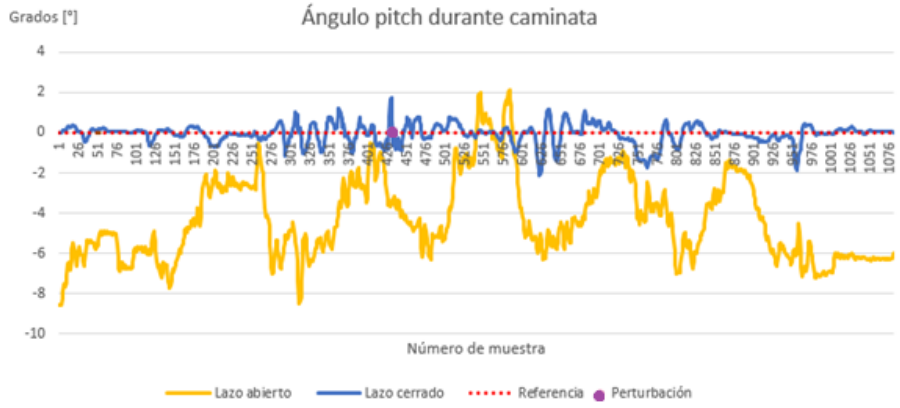


Figura 5.5. Ángulos pitch de la cadera durante la caminata sobre superficie inclinada cinco grados en dirección pitch

La perturbación no afectó en gran medida el desempeño esperado del sistema, el cual consiguió estabilizarse en breve. El análisis del robot, sin el sistema de control, nos devuelve una gráfica con un desfase en el ángulo pitch en sentido negativo (véase figura 5.5). Debido a que, al inclinar la base en sentido positivo, el robot tiende a inclinarse hacia atrás provocando este tipo de comportamiento.

Por su parte, el sistema de control en roll al no verse afectado por la inclinación, ni por la perturbación, tuvo una buena actuación al mantener la cadera con un ángulo roll menor a 2 grados, el sistema en lazo abierto, se comportó de manera similar al experimento anterior (véase figura 5.6).

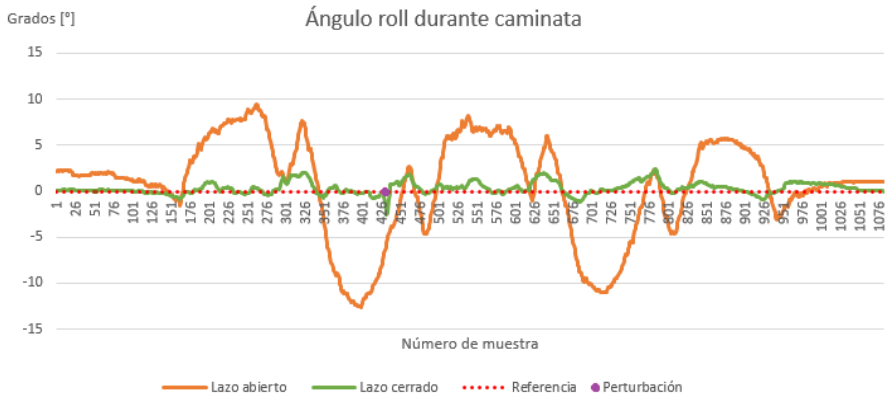


Figura 5.6. Ángulos roll de la cadera durante la caminata sobre superficie inclinada cinco grados en dirección pitch

5.1.4. Prueba con inclinación roll sentido negativo.

Siendo necesario comprobar el funcionamiento del sistema realimentado en ambas direcciones de giro, en la siguiente prueba se inclinó la base cinco grados en sentido negativo y aplicó una perturbación en la muestra 355, durante la fase SSD (1). Como resultado, el robot cayó mientras el sistema de lazo cerrado estaba activo. Por su parte, el sistema de lazo abierto falló, incluso, sin una perturbación.

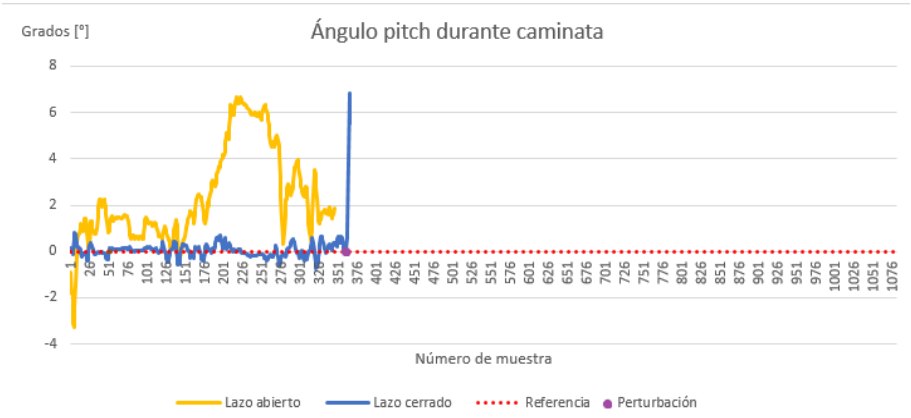


Figura 5.7. Ángulos pitch de la cadera durante la caminata sobre superficie inclinada menos cinco grados en dirección roll

El robot fue incapaz de reponerse de una perturbación, aún con el sistema de control activo (véase figura 5.7). La función de sistema de paro, en caso de caída, permitió detener la marcha y evitar que el robot llevara sus motores a sus ángulos máximos al intentar corregir la postura, lo cual pudo dañar los actuadores y la estructura del robot.

En la figura 5.8 se puede observar que la caída fue en la misma dirección que el robot tendía a volcar, esto significa que la robustez del sistema no fue la suficiente para compensar la perturbación aplicada.

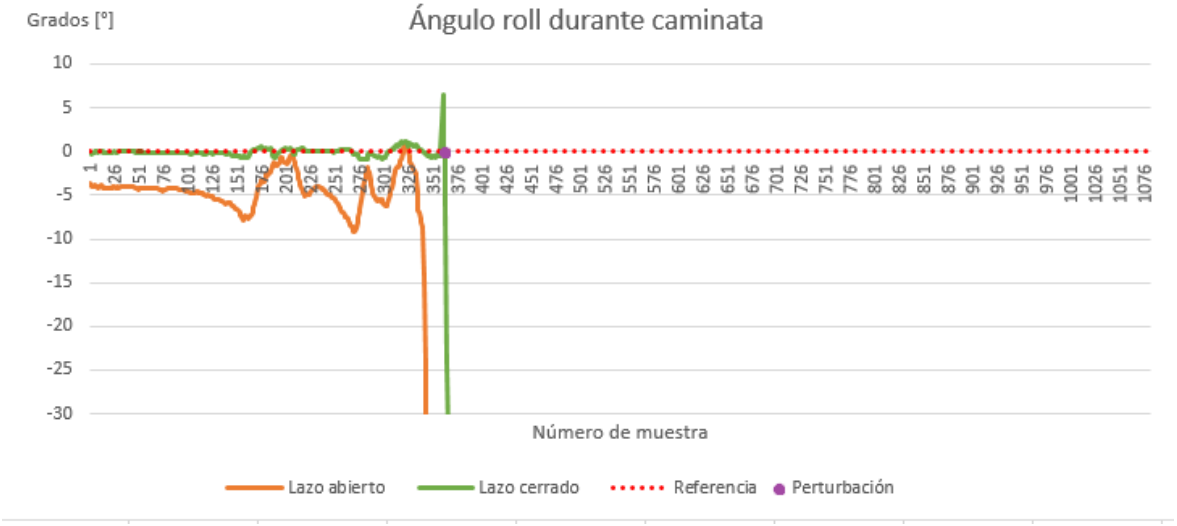


Figura 5.8. Ángulos roll de la cadera durante la caminata sobre superficie inclinada menos cinco grados en dirección roll

5.1.5. Prueba con inclinación roll sentido positivo.

Considerando los resultados de la prueba con inclinación roll en sentido negativo, se optó por realizar un ciclo de caminata con el sistema de lazo cerrado mientras se medía el ZMP para comprobar si en algún momento se generaba una zona de inestabilidad. La prueba se realizó con la base inclinada cuatro grados y medio en la dirección y sentido del giro roll sin perturbaciones; el robot nuevamente cayó después de un comportamiento aceptable hasta la fase SSD (1) (véase figura 5.9). En esta ocasión se optó por documentar la prueba en un video para su análisis posterior y, al revisarse, pudo observarse la causa probable de su caída.

Lo que se encontró fue que el sistema de control de ángulos de cadera se encarga solamente de mantener los ángulos pitch y roll cercanos a cero y, al intentar igualar el ángulo roll a cero, la coordenada Y del ZMP sale del polígono de soporte provocando una posición inestable del robot que posteriormente involucra una caída.

Figura 5.9. Ángulos roll de la cadera durante la caminata sobre superficie inclinada cinco grados en dirección roll

En la figura 5.10, la línea azul representa la inclinación de la cadera y la amarilla, una posición estimada de la coordenada Yzmp. En la imagen de la izquierda también se puede observar al robot con un ángulo roll diferente a cero mientras que, en la de la derecha, el momento de su caída mientras que el sistema de control corrigió la posición de la cadera, expulsando al Yzmp de la planta del pie y, por lo tanto, del polígono de soporte, creando una situación de inestabilidad dinámica.

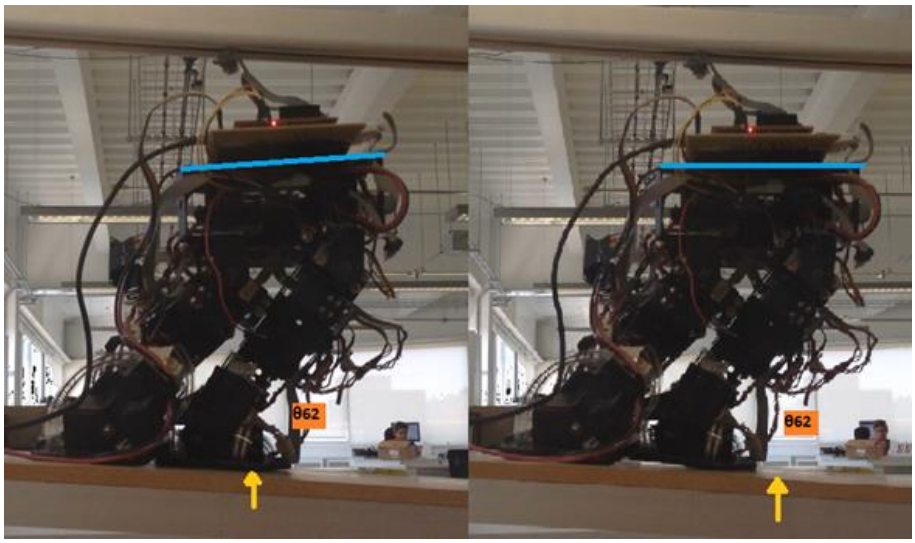


Figura 5.10. Imágenes del robot antes de caer

Al analizar las coordenadas del ZMP durante la caminata se confirmó que el ZMP se ubicaba fuera de la zona de tolerancia que se mencionó en el tema 3.6 sobre el **diseño de referencia de control de ZMP** lo que aumentó las probabilidades de una caída. En la figura 5.11 se pueden observar las coordenadas del ZMP durante la prueba y, dentro de un círculo verde, los calculados justo antes de la caída, los cuales están fuera de la zona de tolerancia.

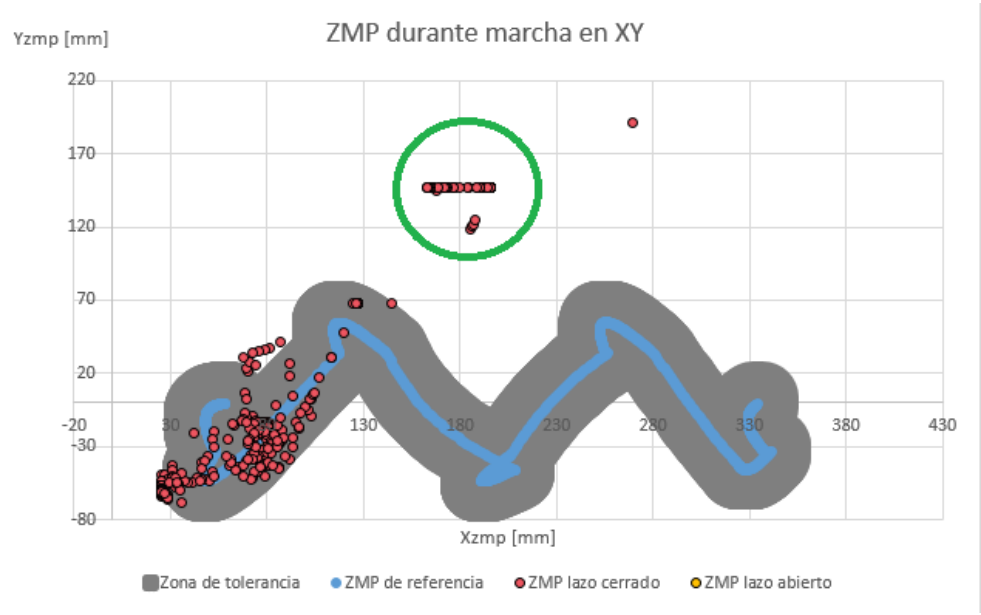


Figura 5.11. Ubicación del ZMP en XY durante marcha y caída del robot

5.2. Resultados del sistema de control del ZMP

Teniendo en consideración lo anterior, se efectuaron diferentes pruebas con el robot realizando caminatas sobre la base de madera. En cada una de ellas se modificó la pendiente para provocar un cambio en las coordenadas del ZMP y aumentar el riesgo de caída; el experimento se hizo activando y desactivando el sistema de control del ZMP. Finalmente, se hizo una prueba con el robot soportando una carga mientras realizaba el ciclo de marcha.

Se graficaron en cada prueba las coordenadas del ZMP con y sin control, contra número de muestra y también en el plano XY. Se hizo lo mismo con los errores absolutos, es decir, la distancia entre el ZMP medido y el límite más cercano de la zona de tolerancia -si el valor está dentro de la zona de tolerancia, el error absoluto se considera nulo-. Finalmente, se realizó una gráfica que muestra el error acumulado de cada una de las variables, con el fin de comparar la efectividad de la marcha en lazo cerrado contra el lazo abierto.

5.2.1. Prueba en superficie horizontal

La primera prueba se llevó a cabo en una superficie sin ningún tipo de inclinación ni perturbación a fin de corroborar el funcionamiento del sistema de lazo cerrado en condiciones ideales. El resultado fue que, de acuerdo a lo esperado, el valor de lazo abierto, no varía demasiado en comparación al medido en el lazo cerrado, sin embargo, sí se aprecia una mejoría en el comportamiento del robot mientras el sistema realimentado se encuentra activo. En el caso de Xzmp, el sistema de control mantuvo la variable dentro del rango deseado como se puede ver en la siguiente gráfica:

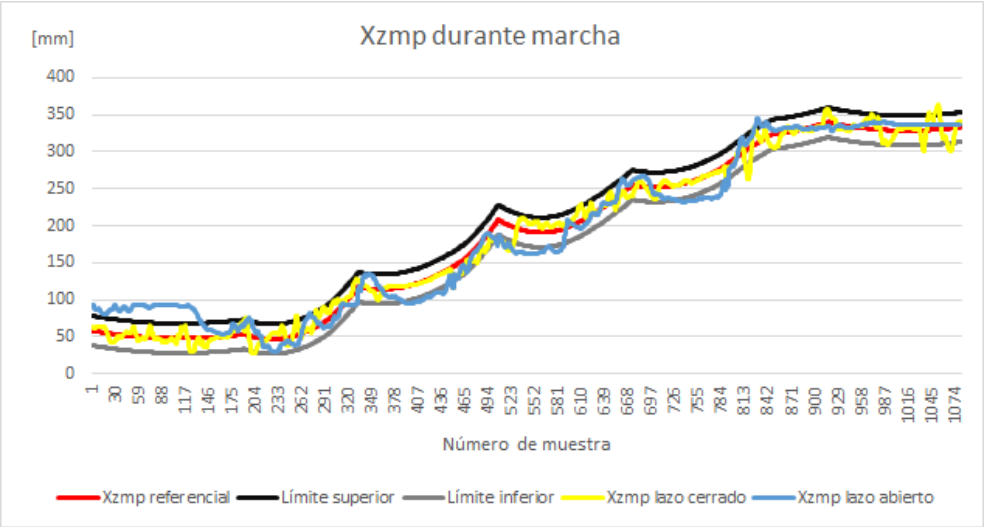


Figura 5.12. Coordenada X del ZMP durante caminata en superficie horizontal

En el caso de la coordenada Yzmp se puede observar en la figura 5.13, que el sistema en lazo abierto tiende a un comportamiento semejante al aproximado por regresiones lineales; por su parte, el sistema en lazo cerrado por lo general se mantuvo dentro del límite inferior y superior de tolerancia.

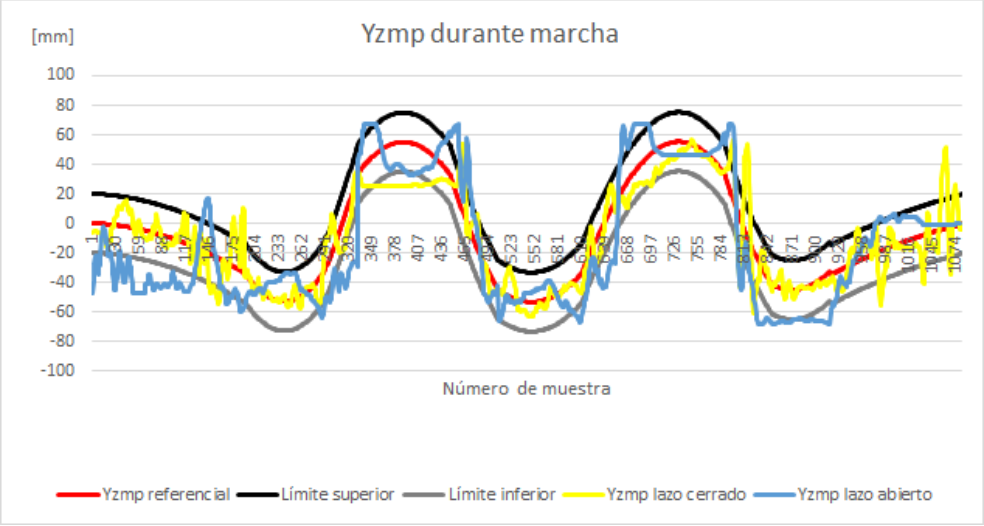


Figura 5.13. Coordenada Y del ZMP durante caminata en superficie horizontal

El análisis de la gráfica del ZMP de sus dos coordenadas permite conocer el comportamiento de su trayectoria en el plano donde camina. El sistema en lazo tuvo una mayor cantidad de mediciones fuera de la zona de seguridad (véase figura 5.14); por su parte, el sistema en lazo cerrado, solo al final de la marcha, presentó ciertas imprecisiones. Al estar en SD, sin embargo, esos puntos se encontrarían dentro del polígono de soporte, aunque más cercanos al pie derecho, lo cual representa una zona estable.

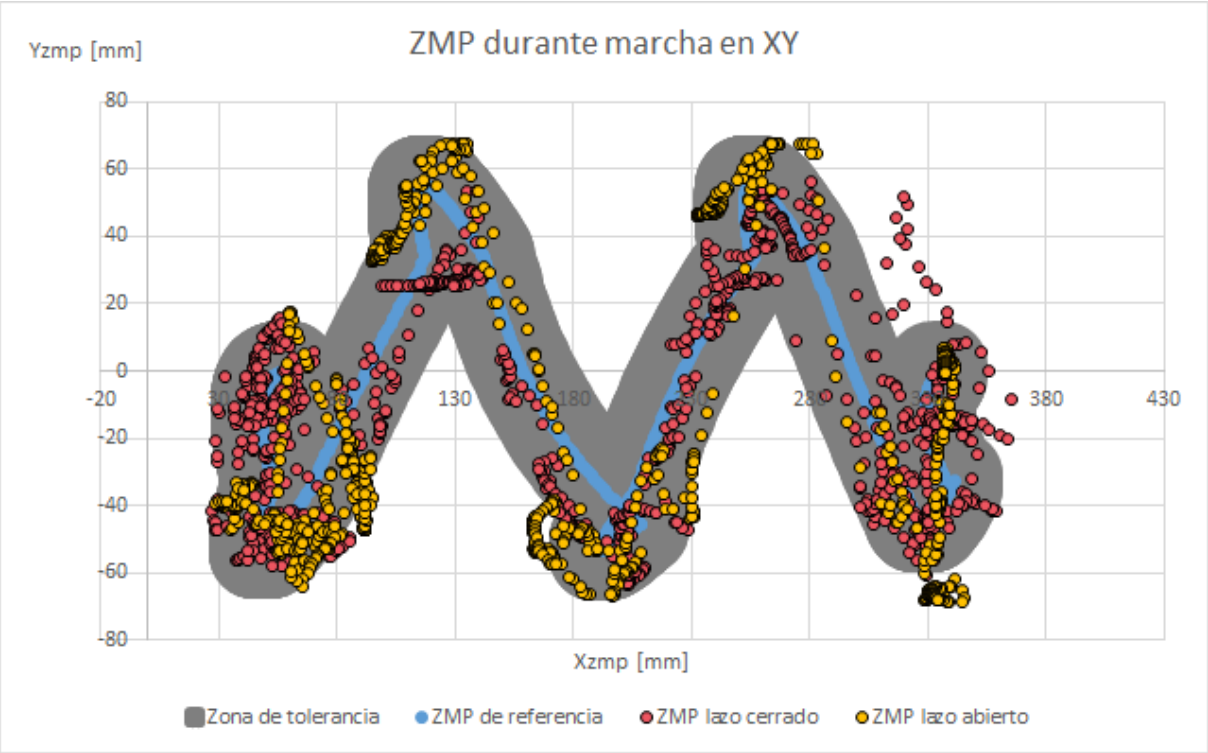


Figura 5.14. ZMP en XY durante caminata en superficie horizontal

Pese a que los comportamientos no presentaron una gran diferencia entre los valores medidos, el sistema realimentado muestra un menor error en comparación a la caminata sin control. En la figura 5.15, se puede ver un aumento en el error en lazo abierto al inicio de la caminata que en lazo cerrado se corrige; por su parte, la corrección en Yzmp del sistema con realimentación no tuvo el mejor desempeño durante la fase SSD (1), donde se aprecia un constante error (véase figura 5.16).

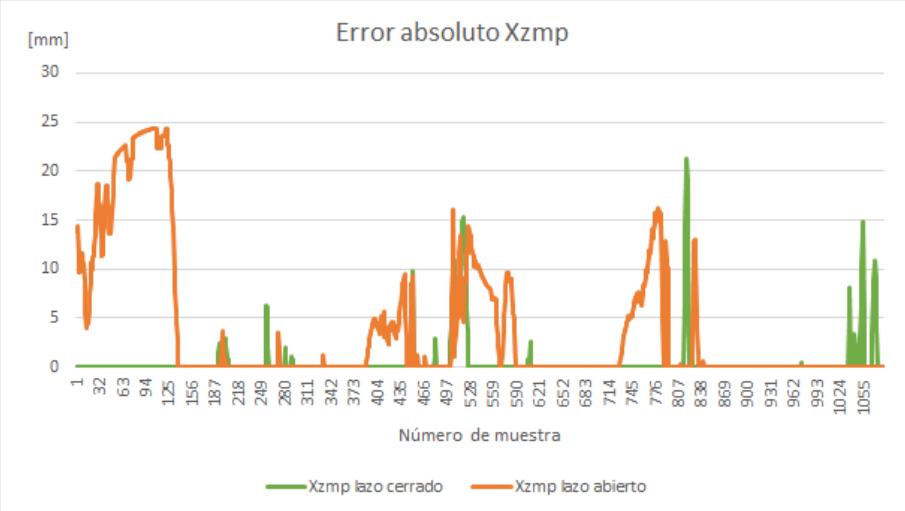


Figura 5.15. Error absoluto en Xzmp durante caminata en superficie horizontal

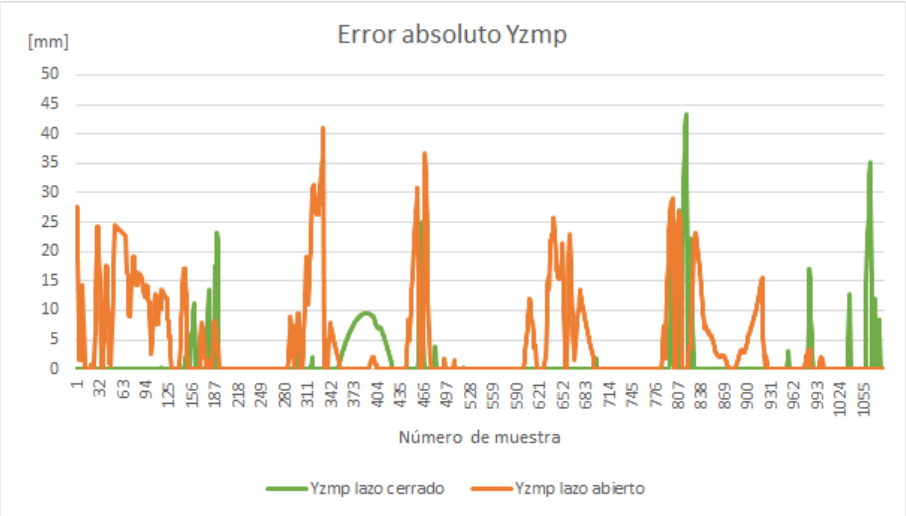


Figura 5.16. Error absoluto en Yzmp durante caminata en superficie horizontal

El análisis de un error acumulado permite hacer más evidente el funcionamiento del sistema de lazo cerrado; por ejemplo, al activarlo la coordenada Xzmp rondó los 500 mm a lo largo de 1080 muestras. En el caso del sistema en lazo abierto, este valor se eleva hasta los 4000 mm (véase figura 5.17).

Por otro lado, el error acumulado en Y, mientras el sistema en lazo cerrado se encontraba activo, superó los 1500 mm durante las 1080 muestras, mientras que su promedio fue de aproximadamente un milímetro de error en cada medida. Por su parte, el sistema en lazo abierto alcanzó los 5000 mm de error acumulado lo cual significa que, en promedio, tuvo un error de más de 4.6 mm durante la caminata (véase figura 5.18).

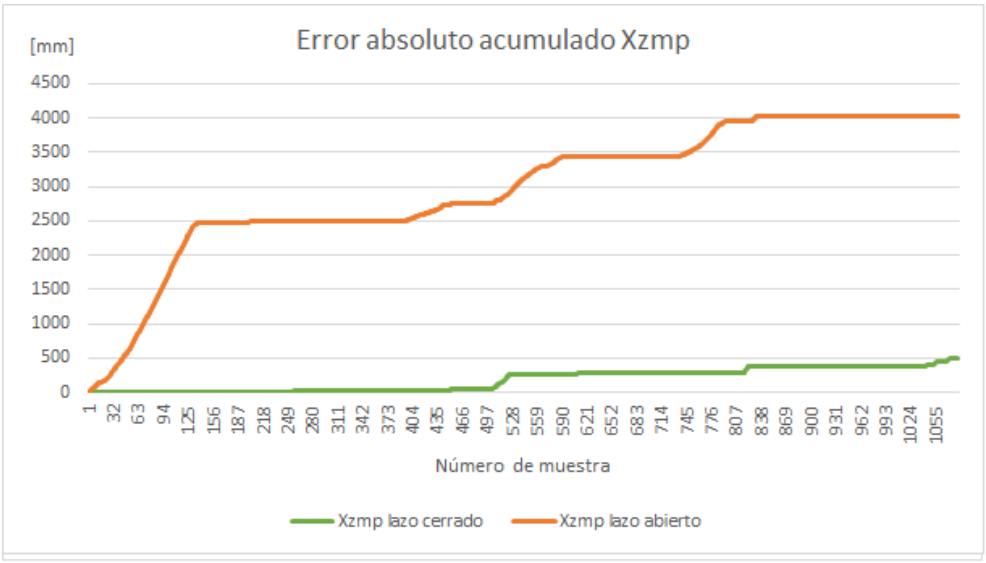


Figura 5.17. Error absoluto acumulado en Xzmp durante caminata en superficie horizontal

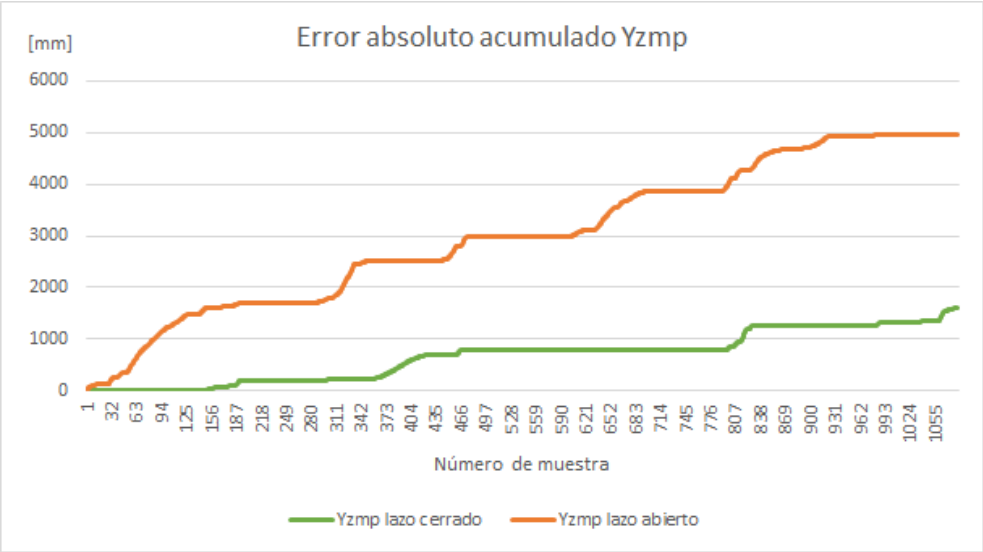


Figura 5.18. Error absoluto acumulado en Yzmp durante caminata en superficie horizontal

5.2.2. Prueba con inclinación pitch sentido negativo.

El experimento se desarrolló con la base de madera inclinada seis grados en dirección pitch y sentido negativo lo cual, en sentido estricto, provocaría que el robot tuviera una tendencia a volcar hacia enfrente ya que su peso se inclina en esa dirección. La variable a analizar es principalmente Xzmp, ya que es la más vulnerable a esta prueba, sin embargo, los resultados devolvieron un aceptable desempeño del sistema de control que mantuvo, en general, el valor del ZMP dentro del área de tolerancia, según se puede constatar en la figura 5.19.

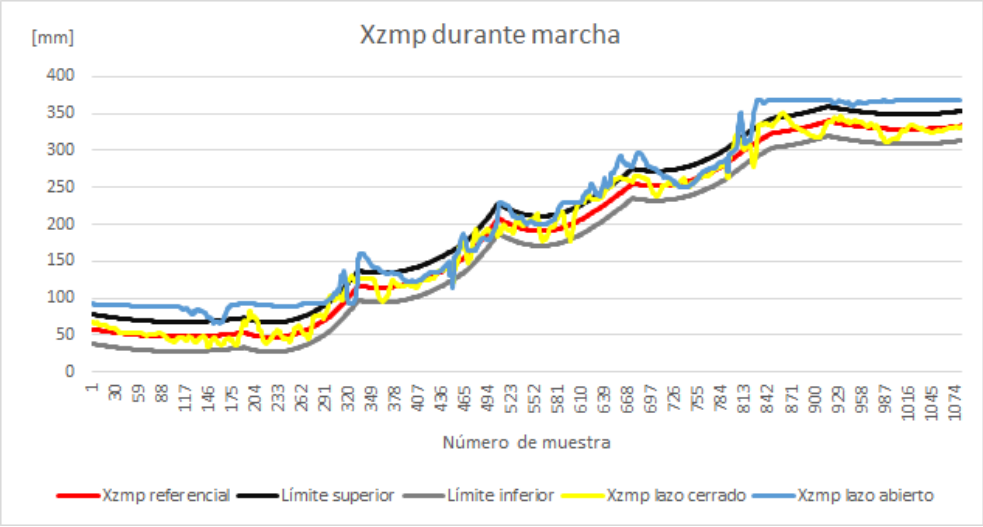


Figura 5.19. Coordenada X del ZMP durante caminata en superficie inclinada menos seis grados en dirección pitch

El sistema en lazo abierto se condujo de manera similar al visto en la prueba horizontal. Por otro lado, el sistema en lazo cerrado tuvo un mejor desempeño en esta prueba al mantenerse, en su mayoría, dentro de la zona de control (véase figura 5.20).

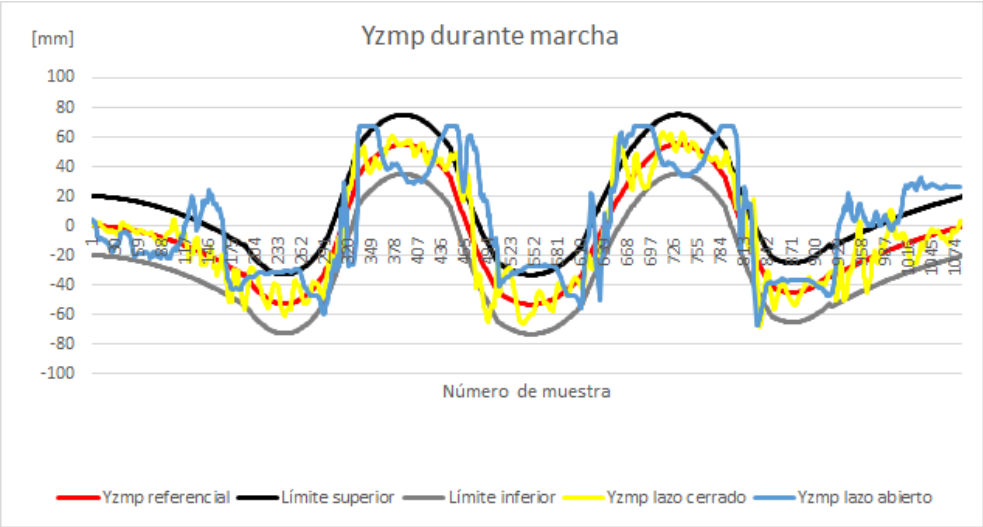


Figura 5.20. Coordenada Y del ZMP durante caminata en superficie inclinada menos seis grados en dirección pitch

El análisis del ZMP en XY muestra un notable desfase de los ZMP en lazo abierto, debido a la inclinación, mientras el sistema de control mantiene el ZMP con un comportamiento deseado (véase figura 5.21).

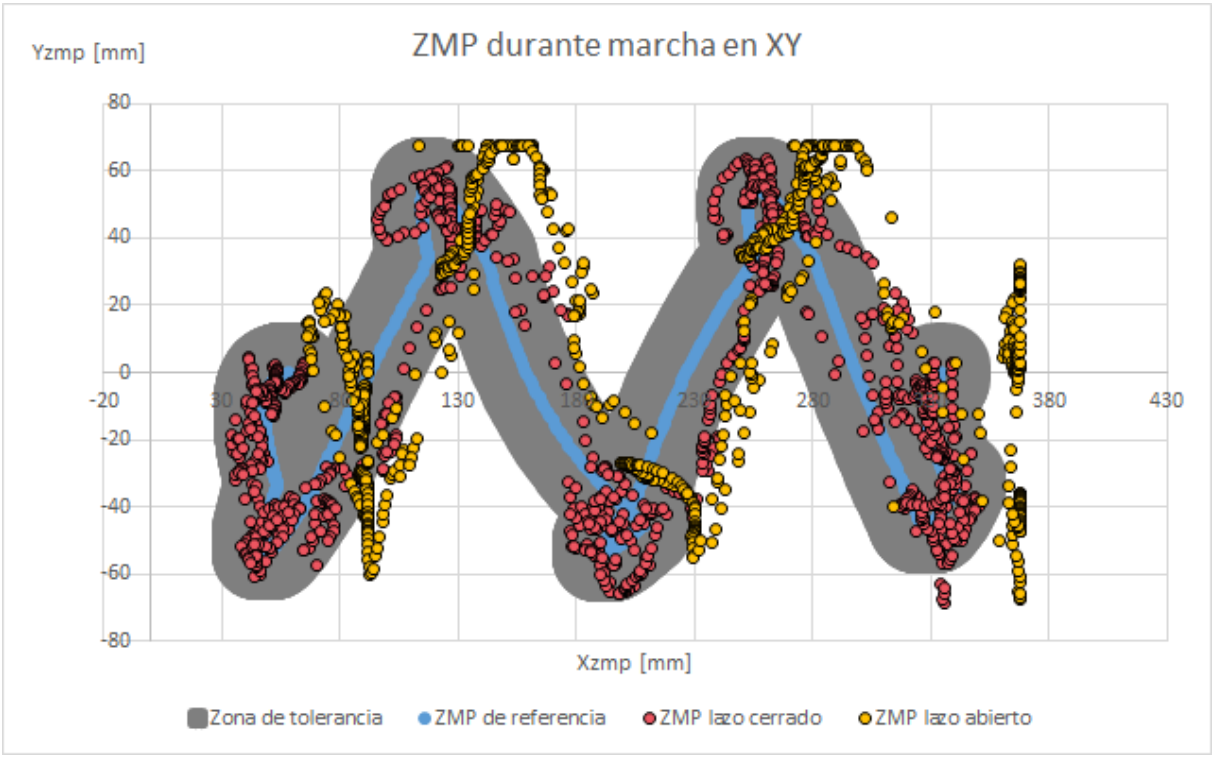


Figura 5.21. ZMP en XY durante caminata en superficie inclinada menos seis grados en dirección pitch

El sistema en lazo cerrado contó con una inferior cantidad de muestras con errores y de menor magnitud. En general, el sistema en lazo abierto tuvo imprecisiones que incluso mantuvieron al

Xzmp del robot por encima de los 30 mm, mientras que, durante el sistema con realimentación, se tuvieron picos de solamente 10 mm como se puede observar en la siguiente figura:

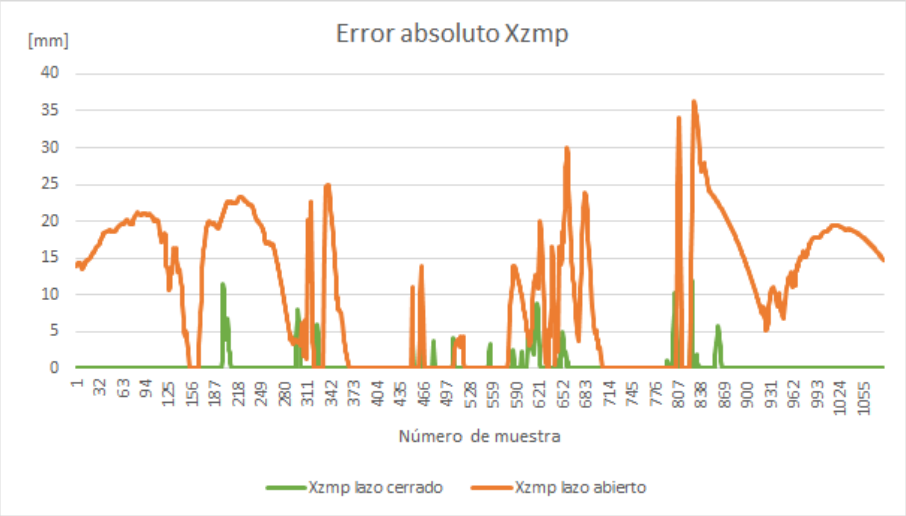


Figura 5.22. Error absoluto en Xzmp durante caminata en superficie inclinada menos seis grados en dirección pitch

En el caso de la coordenada Yzmp, es importante destacar que el sistema de lazo cerrado tuvo la mayor parte de sus errores en los cambios de fase lo cual significa que, la conmutación de los sistemas de control asociados a cada fase, no crean situaciones de riesgo a pesar de que no funcionan de manera óptima (véase figura 5.23).

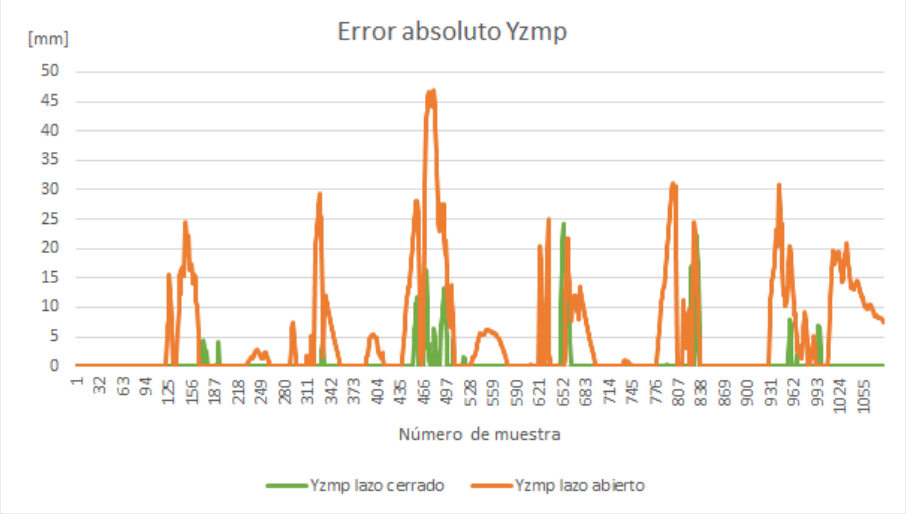


Figura 5.23. Error absoluto en Yzmp durante caminata en superficie inclinada menos seis grados en dirección pitch

En este experimento se notó una mayor contribución del sistema de control del ZMP al realizar un análisis del error acumulado en Xzmp y advertirse un incremento de acuerdo con la prueba anterior en lazo abierto. El error acumulado en lazo abierto superó los 11,000 mm debido al tipo de experimento mientras que, el sistema de control del ZMP, disminuyó este valor a menos de

los 500 mm durante la marcha (véase figura 5.24). En la coordenada Yzmp sólo se notó un ligero decremento en comparación a la prueba anterior, por lo que no amerita mayor atención.

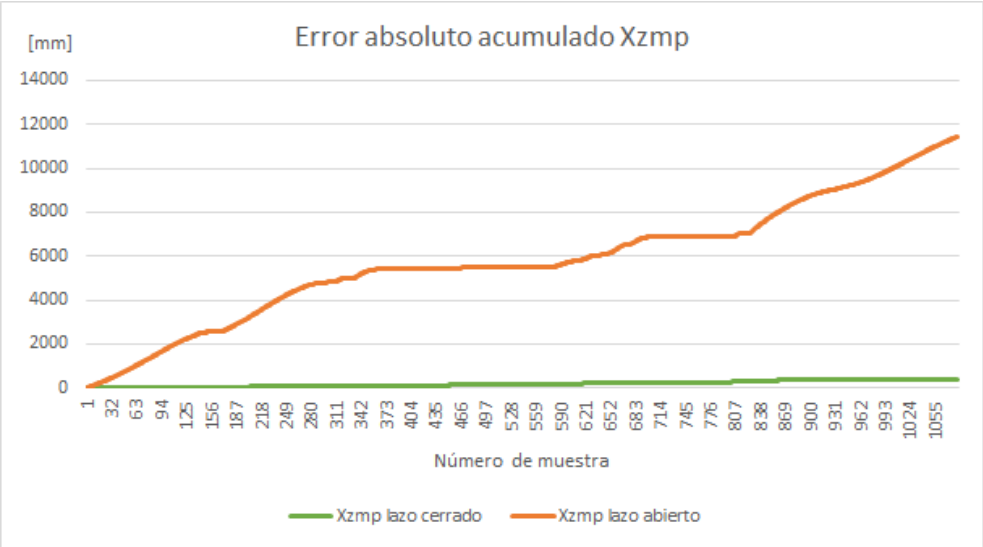


Figura 5.24. Error absoluto acumulado en Xzmp durante caminata en superficie inclinada menos seis grados en dirección pitch

5.2.3. Prueba con inclinación pitch sentido positivo.

Para esta prueba la base se inclinó seis grados en dirección y sentido del giro pitch, posición que genera una inclinación del robot hacia atrás y una mayor carga en los motores. Al superar esta prueba satisfactoriamente, se puede considerar que la variable pitch ha sido controlada, dentro de los objetivos de este escrito.

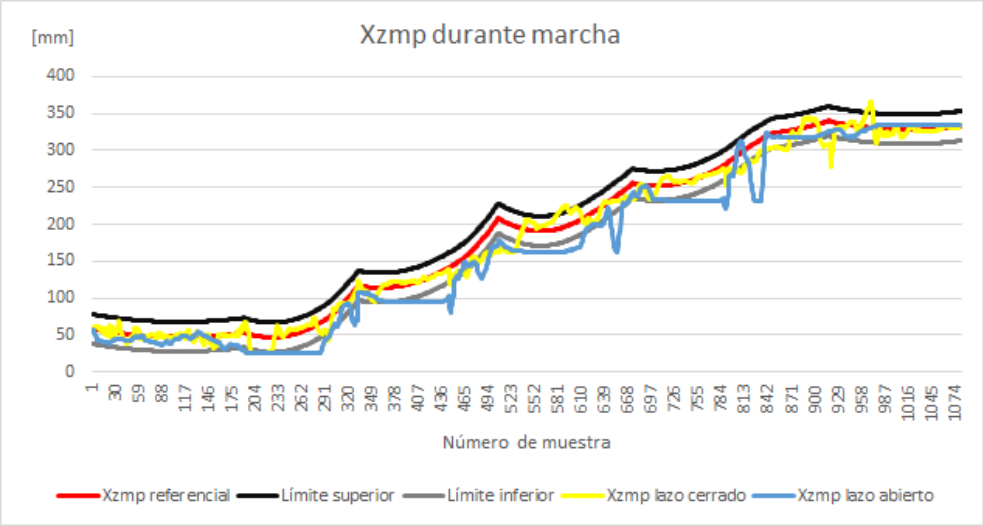


Figura 5.25. Coordenada X del ZMP durante caminata en superficie inclinada seis grados en dirección pitch

Al intentar subir la pendiente durante la marcha –con control- se percibe en el robot una mayor proximidad hacia el límite inferior de la zona de tolerancia en Xzmp, en comparación a las

pruebas anteriores (véase figura 5.25). A pesar de esto, el sistema de control tuvo una buena intervención de dicho valor al mantenerlo, en su mayoría, dentro de la zona de tolerancia.

En la figura 5.26, al igual que en las pruebas anteriores, se pueden observar algunos puntos fuera de la zona de seguridad, cuando el sistema de lazo cerrado se encuentra activo, especialmente cuando se realiza la conmutación entre sistemas de control.

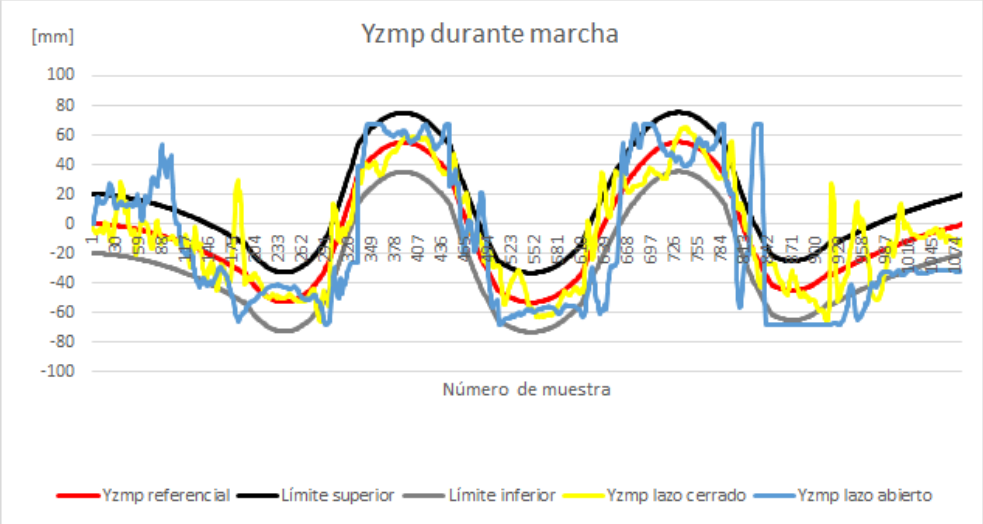


Figura 5.26. Coordenada Y del ZMP durante caminata en superficie inclinada seis grados en dirección pitch

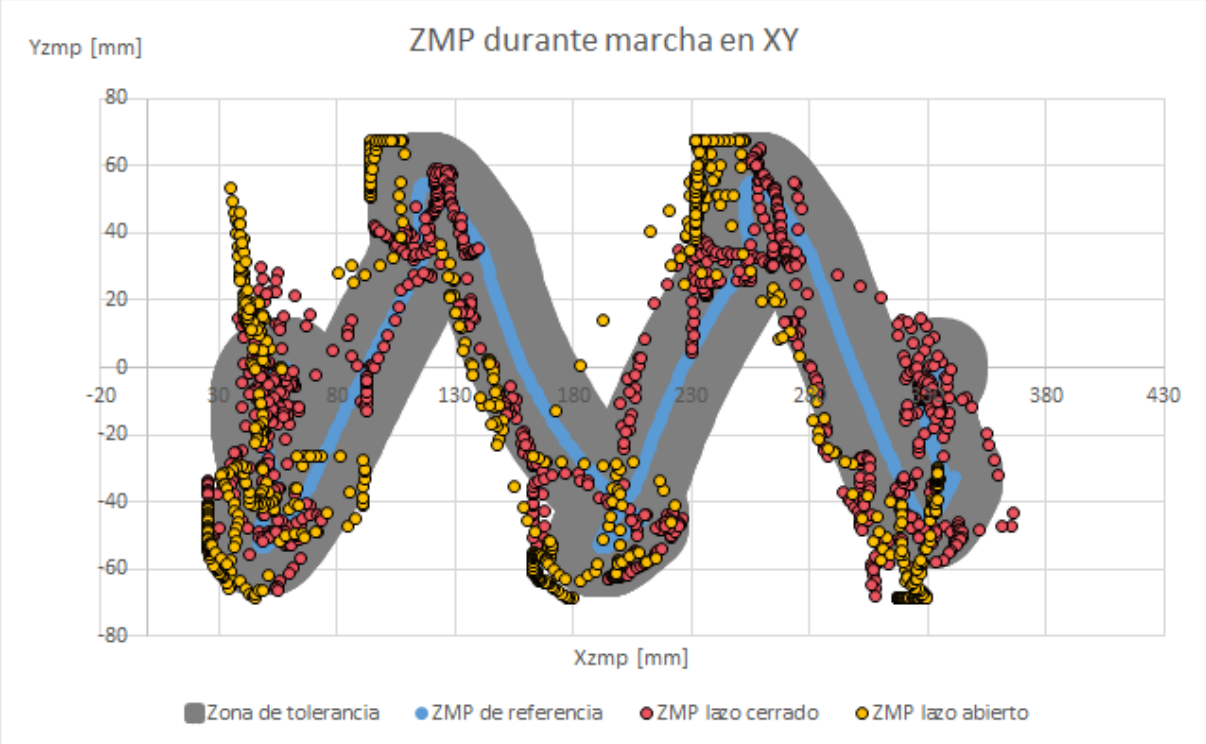


Figura 5.27. ZMP en XY durante caminata en superficie inclinada seis grados en dirección pitch

Como se puede ver en la figura 5.27, el robot tiene una mayor cantidad de puntos fuera de la zona de tolerancia en lazo abierto en comparación a las pruebas anteriores que, si bien disminuyen su estabilidad, no provocan su volcadura, además, se observa que los puntos en ambos sistemas presentan un desfase hacia atrás del robot (izquierda de la gráfica), aunque los ZMP del sistema en lazo cerrado se encuentran, por lo general, más cercanos al ZMP referencia.

En esta prueba los errores absolutos, en el sistema de lazo abierto, no fueron tan notables como el caso anterior debido a que las trayectorias en lazo abierto tienden a llevar al robot hacia adelante y, de esa manera, a compensar la inclinación en ciertas ocasiones (véase figura 5.28).

En el caso del valor Yzmp se obtuvieron resultados parecidos a las pruebas anteriores debido a que, nuevamente, no se perturbó el robot en ese sentido, por lo que no resulta relevante mostrar las gráficas.

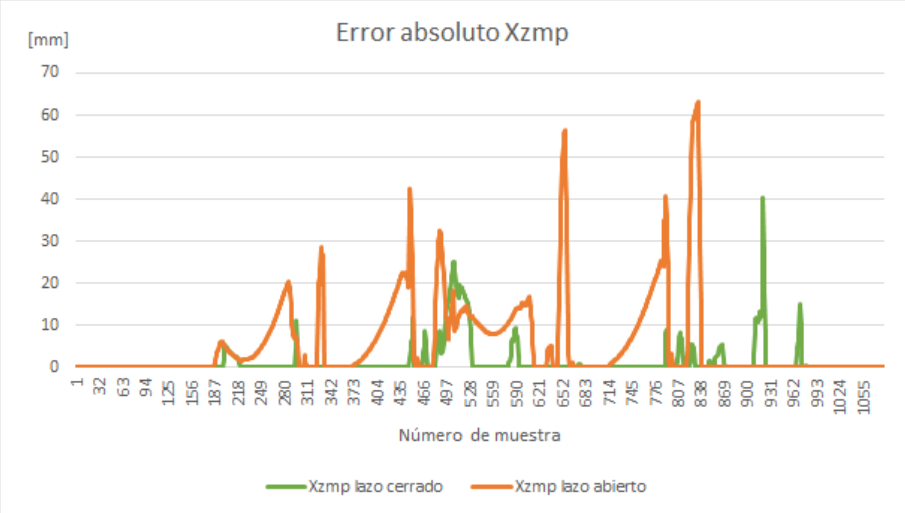


Figura 5.28. Error absoluto en Xzmp durante caminata en superficie inclinada seis grados en dirección pitch

El análisis del error acumulado en Xzmp en lazo abierto alcanzó los 6,000 mm. El sistema de control del ZMP, por su parte, disminuyó este valor a menos de 1500 mm; un valor un poco elevado considerando los resultados anteriores, pero razonable, de acuerdo con la naturaleza de la prueba (véase figura 5.29).

La diferencia es clara entre la capacidad de los sistemas de lazo cerrado y abierto para mantener al ZMP dentro de una zona segura.

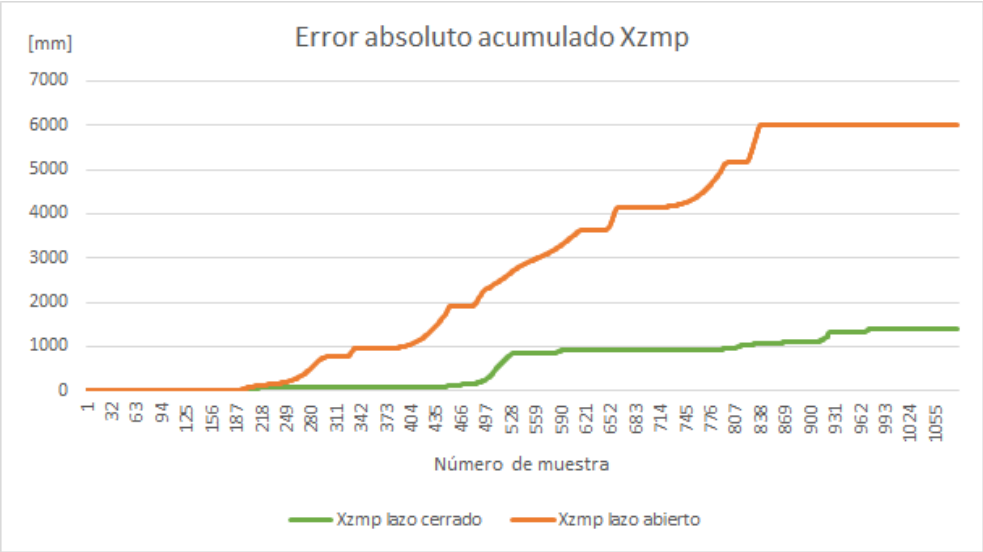


Figura 5.29. Error absoluto acumulado en Xzmp durante caminata en superficie inclinada seis grados en dirección pitch

5.2.4. Prueba con inclinación roll sentido negativo

Para probar la robustez del sistema de control del ZMP, afectando su coordenada Y, se inclinó la base cinco grados en dirección roll sentido negativo provocando que el robot se recargara más en su pie izquierdo y, por lo tanto, tuviera mayor riesgo de caer de ese lado.

Comparado con el sistema de control de ángulos de cadera, el sistema presentó una mejor respuesta a este tipo de desafío, teniendo un comportamiento un poco más variable, pero dentro de la zona de tolerancia. Es importante mencionar que en esta misma prueba el sistema de control de orientación de la cadera falló al volcar durante la marcha, sin embargo, el sistema de control de ZMP logró superarla exitosamente.

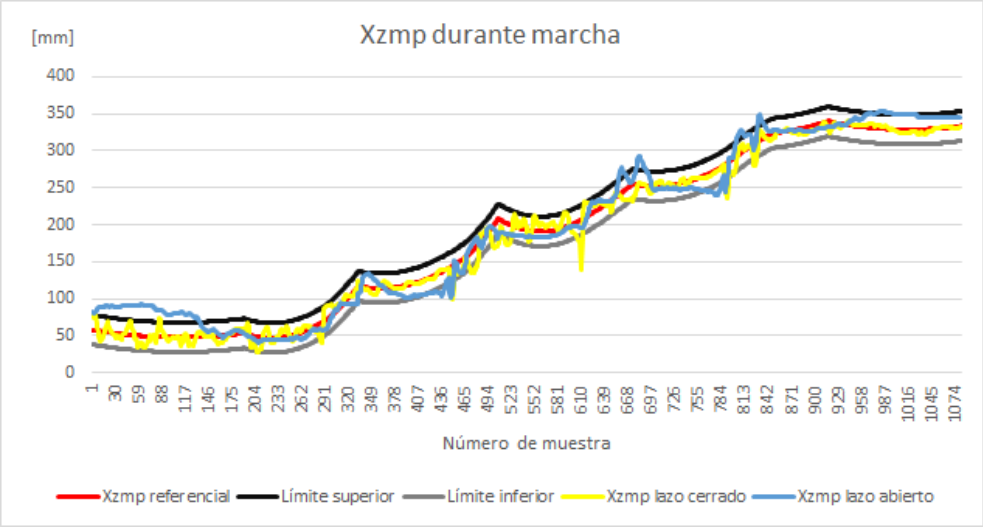


Figura 5.30. Coordenada X del ZMP durante caminata en superficie inclinada menos cinco grados en dirección roll

En el caso de la coordenada Xzmp, ésta no sufre mayor desviación al encontrarse, por lo general, dentro de los límites de seguridad, al no verse afectada por la prueba (véase figura 5.30).

En la figura 5.31 es posible discernir un comportamiento aceptable cuando el sistema de control calcula un punto fuera de la zona de seguridad y corrige la postura, mientras el sistema sin realimentación presenta una tendencia a mantener la coordenada Y del ZMP del robot por debajo del límite inferior de la zona de tolerancia, los sobresaltos en los cambios de fase son más notorios al conmutar el sistema de control.

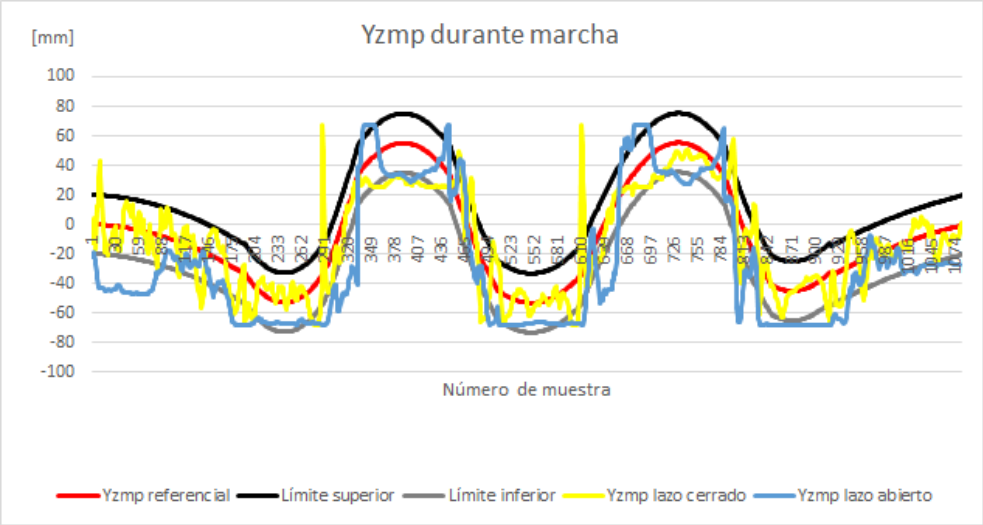


Figura 5.31. Coordenada Y del ZMP durante caminata en superficie inclinada menos cinco grados en dirección roll

La figura 5.32 muestra un conjunto de ZMP en lazo abierto que se encuentran fuera de la zona de tolerancia, a pesar de que esto vuelve al robot dinámicamente inestable, este no sufrió caída alguna. De igual manera, es posible notar una mayor cantidad de ZMP del sistema de lazo cerrado fuera de la zona de seguridad en comparación de las pruebas anteriores; sin embargo, tal situación de riesgo sigue siendo menos frecuente que lo visto durante la prueba en lazo abierto, cuyos puntos tienden a desfasarse a la izquierda del robot.

En esta prueba los errores absolutos son menores en la coordenada X que en la coordenada Y; algo predecible debido a la naturaleza del experimento.

En el caso del error absoluto en Xzmp existen dos aspectos a destacar, el primero es el constante error en lazo abierto al inicio de la caminata, parecido al visto en la prueba en horizontal. El segundo es la breve, pero gran perturbación de la señal de error en lazo cerrado, cuyo valor se eleva hasta los 45 mm y el sistema estabiliza inmediatamente en la muestra 610 aproximadamente (véase figura 5.33).

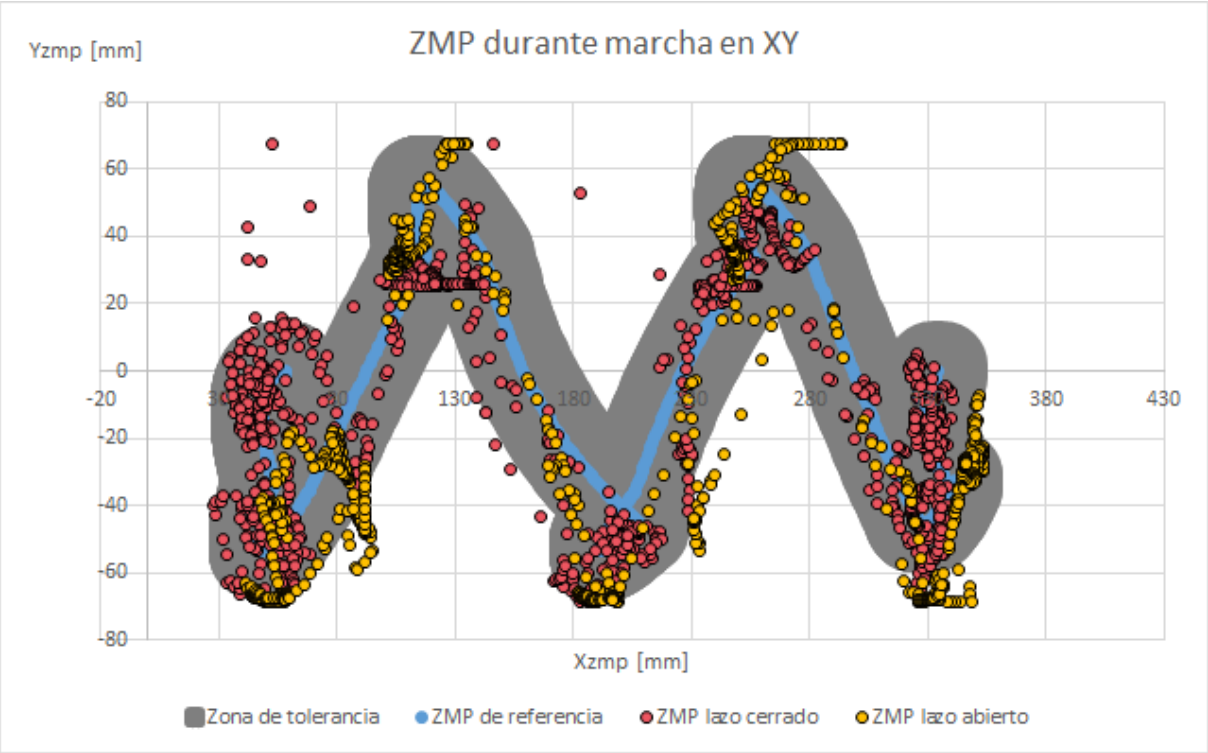


Figura 5.32. ZMP en XY durante caminata en superficie inclinada menos cinco grados en dirección roll

En la figura 5.34 es visible que el error en Yzmp durante la marcha, con el sistema de control en ZMP activo, tiene dos picos considerables, sin embargo, los anula de modo que el robot no tenga una caída; fuera de eso, tuvo un mejor desempeño que el sistema de lazo abierto.

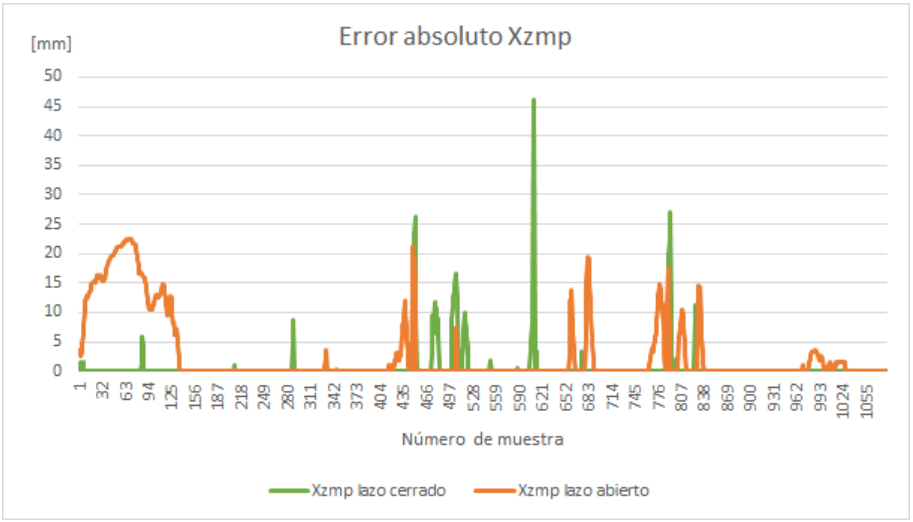


Figura 5.33. . Error absoluto en Xzmp durante caminata en superficie inclinada menos cinco grados en dirección roll

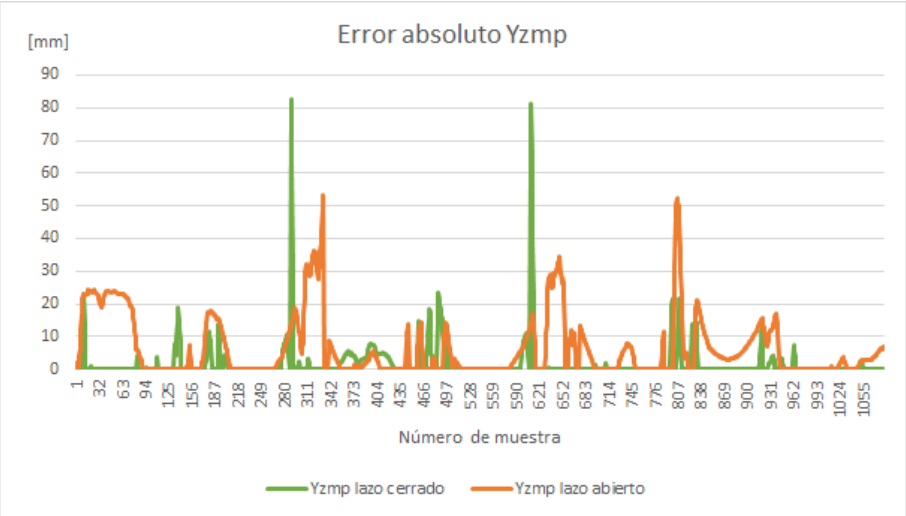


Figura 5.34. Error absoluto en Yzmp durante caminata en superficie inclinada menos cinco grados en dirección roll

A pesar de que en ocasiones el sistema de lazo cerrado presentó algunos picos en la señal del error, esta demuestra un mejor comportamiento del sistema de control contra el sistema de lazo abierto.

El error absoluto acumulado en la coordenada Xzmp es prácticamente idéntico a la prueba horizontal, por lo que no es necesario mayor detalle, por su parte, su equivalente en Yzmp tiene un aumento considerable en cuanto a las pruebas anteriormente realizadas superando los 6000 mm, en lazo abierto, y los 1800 mm, en lazo cerrado (véase figura 5.35).

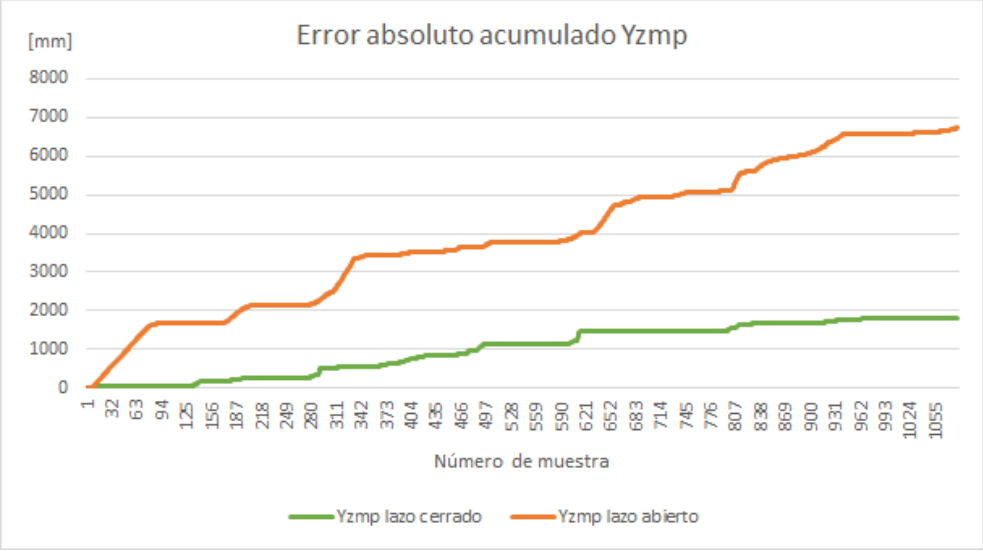


Figura 5.35. Error absoluto acumulado en Yzmp durante caminata en superficie inclinada menos cinco grados en dirección roll

5.2.5. Prueba con inclinación roll sentido positivo

La siguiente prueba consistió en aumentar la pendiente de la base cinco grados en dirección y sentido roll, provocando una inclinación hacia el lado derecho del robot y un incremento en la coordenada Yzmp; el éxito en esta prueba representa la capacidad del robot de caminar en cualquier dirección y pendiente (hasta 5 grados).

En la coordenada Xzmp es posible observar un aceptable rendimiento tanto en lazo abierto, como en lazo cerrado, con una notable mejoría en el sistema con realimentación (véase figura 5.36).

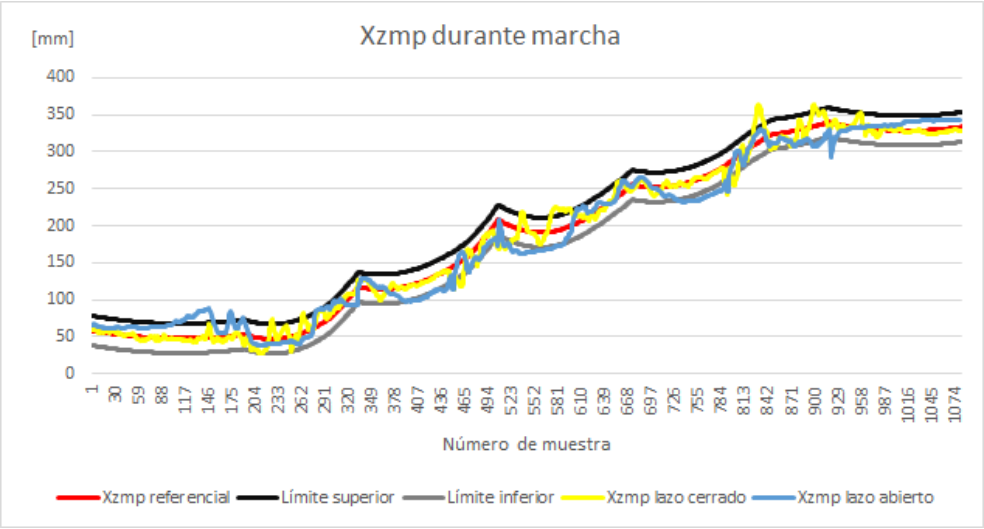


Figura 5.36. Coordenada X del ZMP durante caminata en superficie inclinada cinco grados en dirección roll

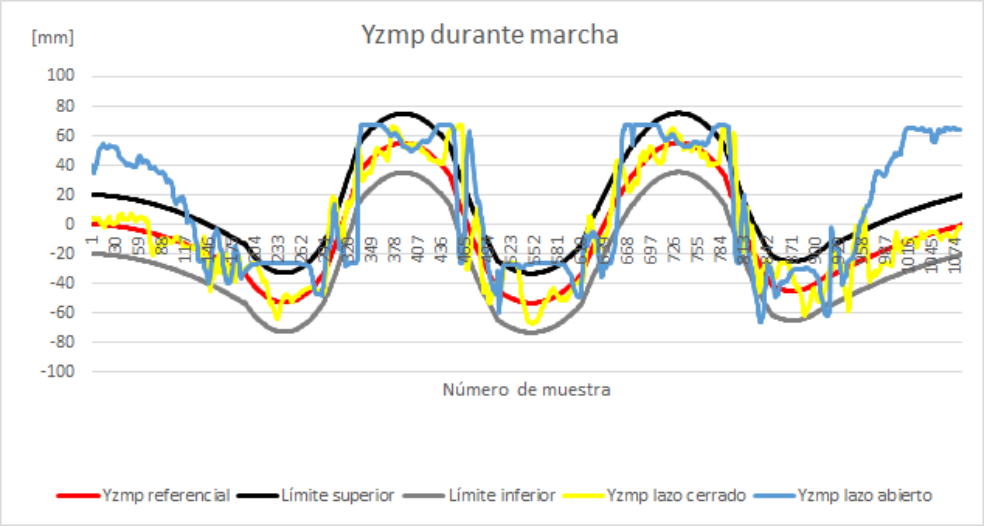


Figura 5.37. Coordenada Y del ZMP durante caminata en superficie inclinada menos cinco grados en dirección roll

En cambio, la coordenada Yzmp presenta un constante error al inicio y fin de la caminata, mientras que, el sistema en lazo cerrado, mantuvo el parámetro Yzmp dentro de los límites preestablecidos. Comparando con la misma prueba con el sistema de ángulos de cadera, se puede apreciar una mayor robustez en el sistema de ZMP, al lograr completar la prueba y controlar la variable de control, según se puede constatar en la figura 5.37.

En la figura 5.38 es posible apreciar una tendencia en los ZMP, durante el sistema de lazo abierto, a desplazarse a la derecha del bípedo (hacia arriba en la gráfica), especialmente al inicio y fin de la caminata cuando el robot se encuentra en SDA. En cuanto al sistema de control realimentado, se puede ver casi la totalidad de sus ZMP dentro de la zona de tolerancia. Se puede decir que el sistema de control compensó de manera ambos ejes y mantuvo el ZMP dentro del rango establecido y, por lo tanto, el robot se mantuvo dinámicamente estable.

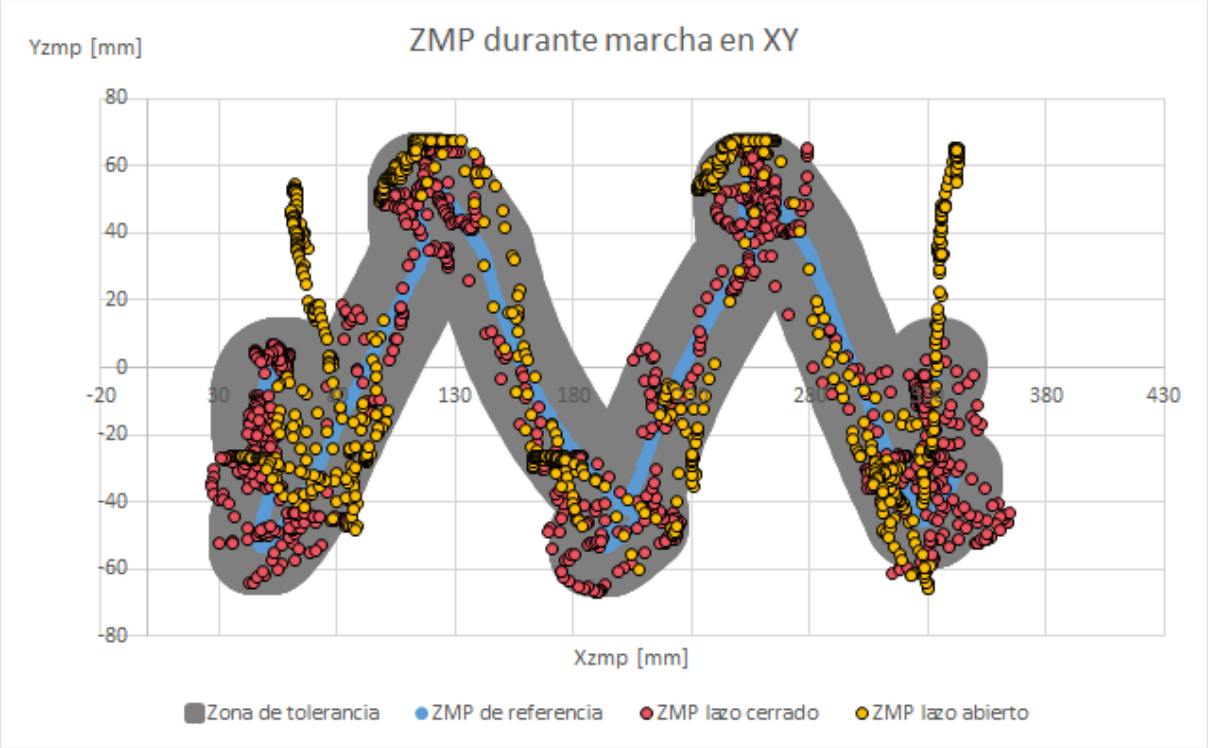


Figura 5.38. ZMP en XY durante caminata en superficie inclinada cinco grados en dirección roll

En la figura 5.39 es posible encontrar una semejanza en los valores de error entre ambos sistemas; sin embargo, se observa una menor cantidad de errores en Xzmp mientras el sistema de control se encuentra activo, aunque la diferencia puede no ser tan clara.

La coordenada Yzmp del robot durante la caminata, sin sistema de control, tuvo una gran cantidad de valores fuera de la zona de tolerancia por lo que, en la figura 5.40, se ve una mayor actividad de la señal de error respecto a su semejante en lazo cerrado.

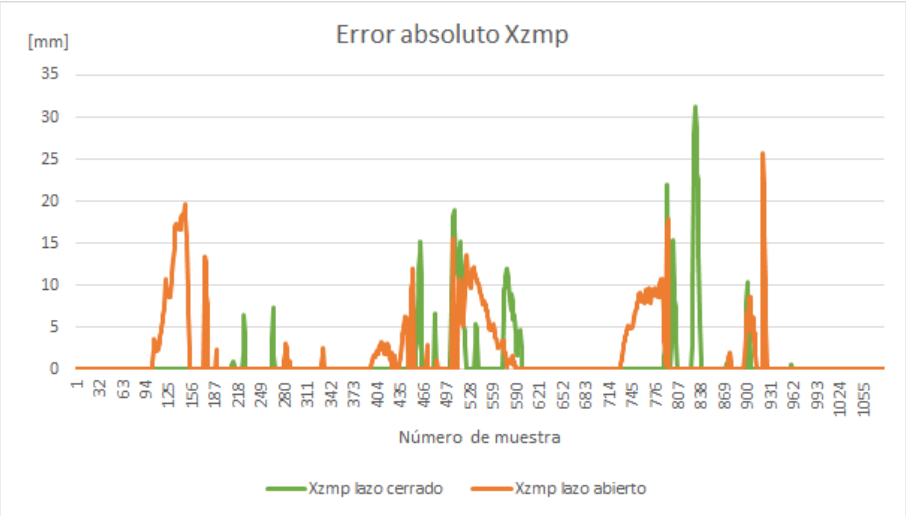


Figura 5.39. Error absoluto en Xzmp durante caminata en superficie inclinada menos cinco grados en dirección roll

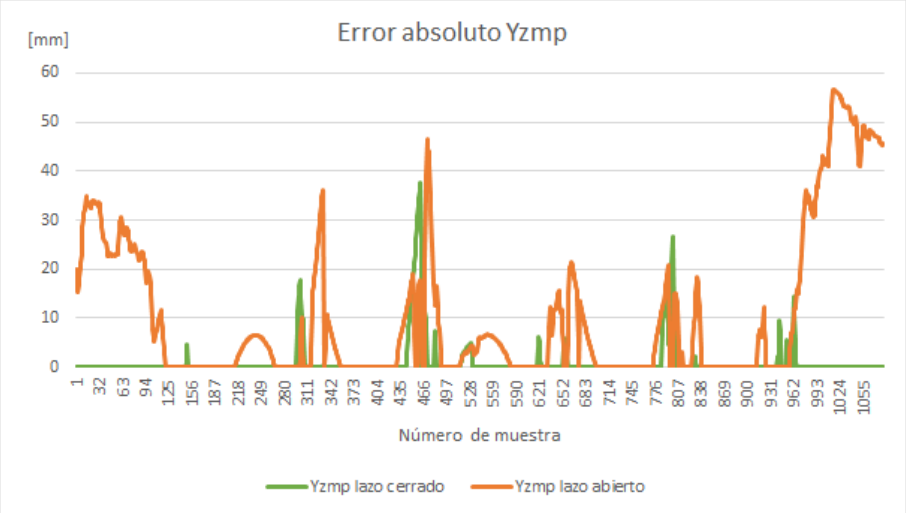


Figura 5.40. Error absoluto en Yzmp durante caminata en superficie inclinada menos cinco grados en dirección roll

El análisis del error absoluto acumulado, a su vez, permitió comparar desde una perspectiva diferente la discrepancia en Xzmp que aparentó ser similar entre los sistemas de lazo cerrado y abierto y evidenció el mejor desempeño del sistema realimentado (véase figura 5.41).

En el caso de su desempeño en Yzmp, en la figura 5.42, es posible ver un valor bastante elevado en el error acumulado en lazo abierto, superando los 11,000 [mm], lo que representa una desviación de la zona de tolerancia de aproximadamente 10 [mm] en cada momento de la caminata. Comparándolo con el promedio de un milímetro alcanzado en lazo cerrado, es innegable una mejoría en la estabilidad de la marcha frente al sistema sin control.

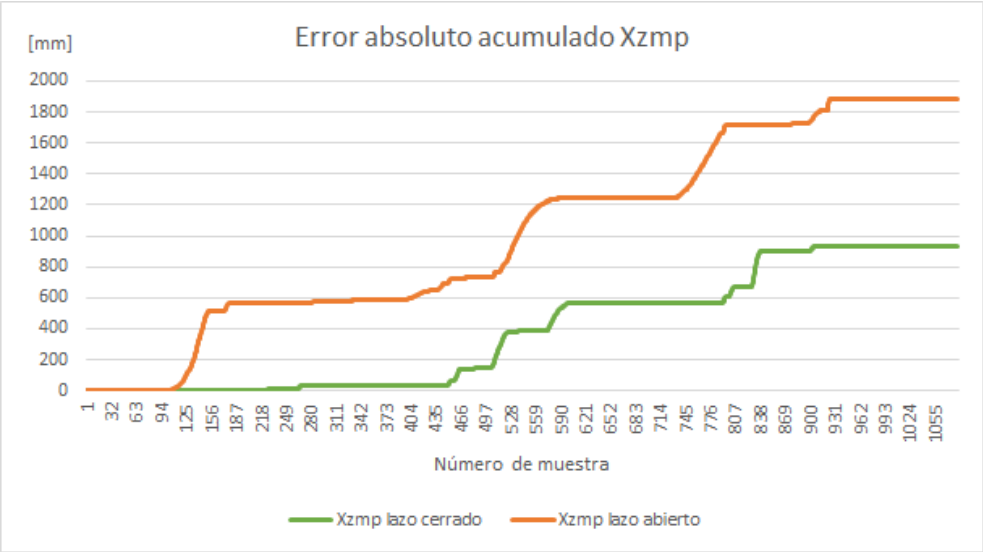


Figura 5.41. Error absoluto acumulado en Xzmp durante caminata en superficie inclinada cinco grados en dirección roll

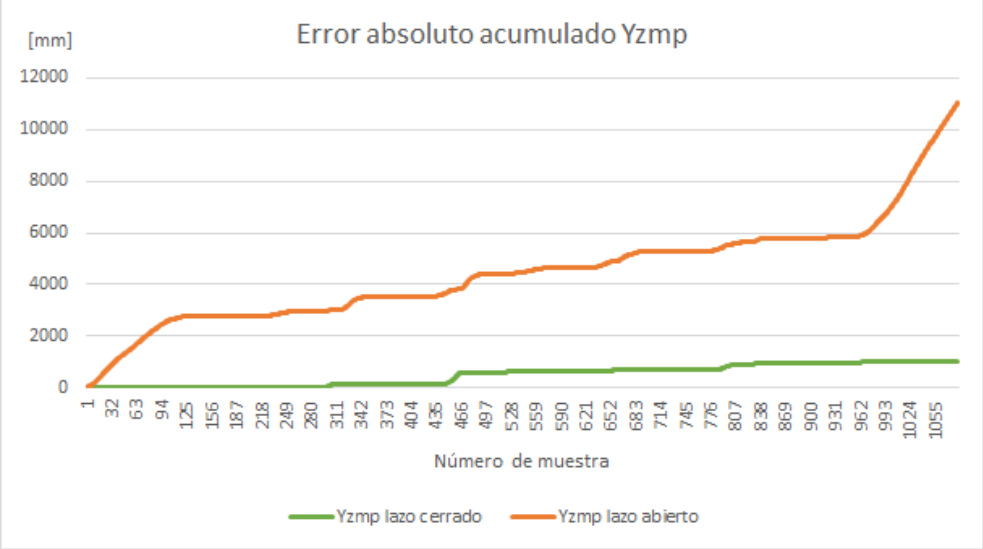


Figura 5.42. Error absoluto acumulado en Yzmp durante caminata en superficie inclinada cinco grados en dirección roll

5.2.6. Prueba en superficie horizontal con carga.

La prueba final consistió en producirle un cambio en la coordenada Xzmp llevándola hacia el límite inferior de la zona de tolerancia (véase figura 5.43). Para ello, al robot se le colocó una masa de acero de 150 gramos en la parte posterior. Durante la marcha, con el sistema de lazo abierto, el robot cayó en la muestra 330, después de terminar la fase SSI (1). En el sistema de lazo cerrado, a pesar de tener un error en ese mismo periodo, permitió que el robot se recuperara y continuara su marcha hasta el final.

En la figura 5.44 es posible notar que la coordenada Yzmp con realimentación sólo presentó ligeras perturbaciones, semejante a la prueba horizontal; por su parte, el sistema en lazo abierto deja de tener registro después de su caída.

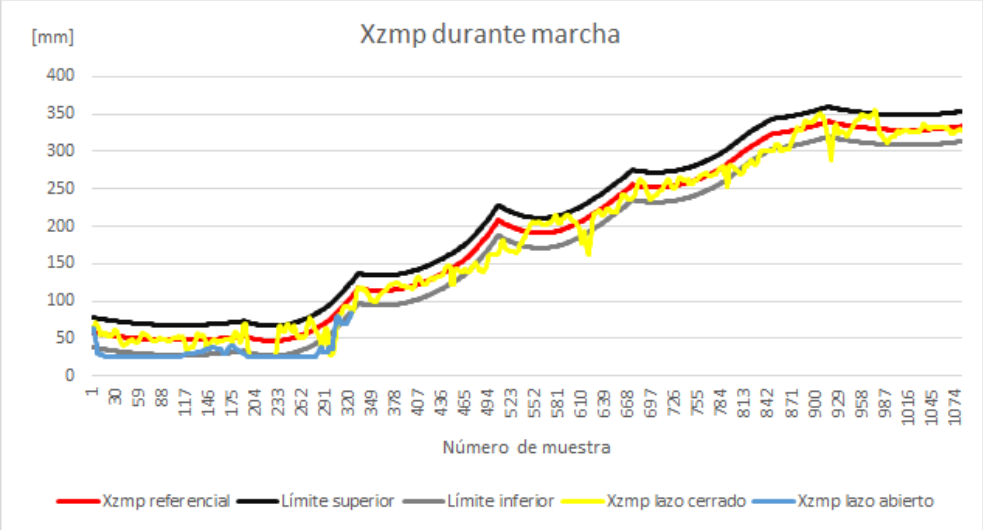


Figura 5.43. Coordenada X del ZMP durante caminata con carga

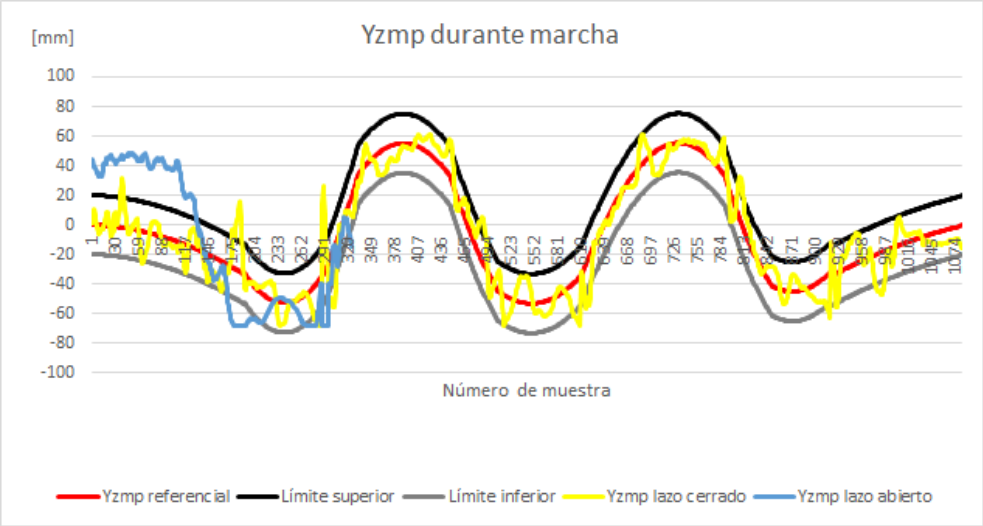


Figura 5.44. Coordenada Y del ZMP durante caminata con carga

En la figura 5.45 se encerró un conjunto de puntos amarillos dentro del círculo morado, los que representan los últimos ZMP medidos antes de la caída y, como se logra apreciar, se encuentran fuera de la zona de tolerancia, lo que aumentó la inestabilidad del robot. En cambio, el sistema realimentado, pese a que tuvo una mayor cantidad de puntos fuera de la zona de tolerancia, logró completar un ciclo de caminata sin caer.

La caminata con carga representó para el robot una constante corrección de sus ángulos; ello, pese a que durante la marcha únicamente se aprecian errores en algunos momentos,

especialmente en las fases de SS. Se concluye entonces que, en esta prueba, el sistema de control del ZMP tuvo un buen desempeño.

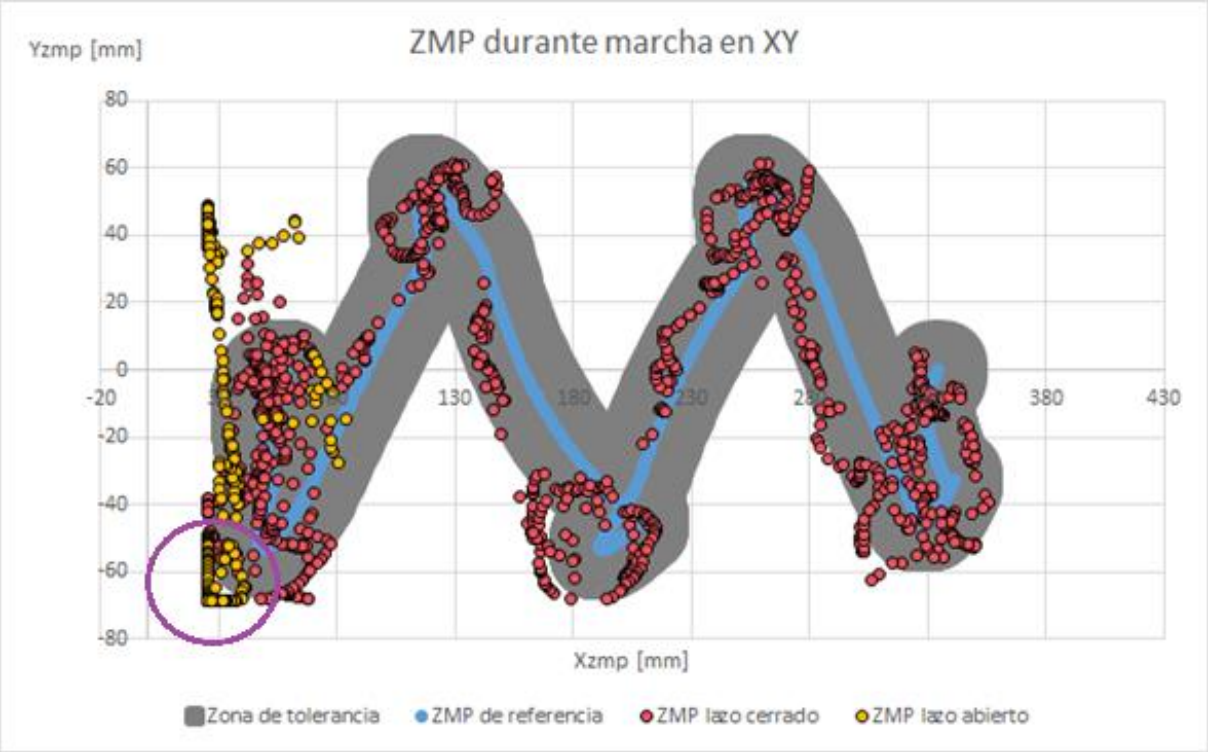


Figura 5.45. ZMP en XY durante caminata con carga

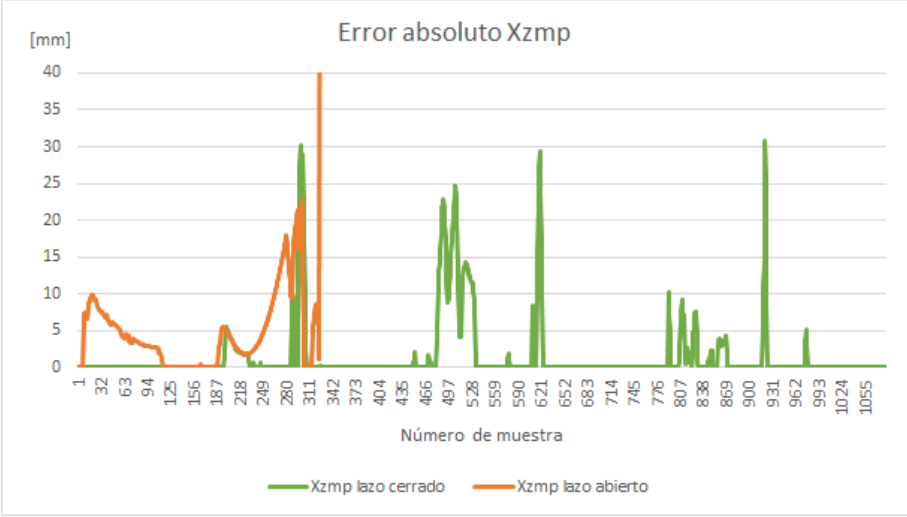


Figura 5.46. Error absoluto en Xzmp durante caminata con carga

En cuanto al error absoluto en la coordenada Xzmp en lazo abierto, al producirse la caída el robot deja de tener polígono de soporte por lo que el error en lazo abierto tiende al infinito. Al implementarse la realimentación, el robot continuó con la marcha. En este punto es importante

destacar la nula actividad del error en Xzmp durante el inicio y final de la caminata (véase figura 5.46).

De igual manera que en el Xzmp, en el caso de lazo abierto, el error absoluto en Yzmp se eleva drásticamente al producirse la caída. La participación del lazo cerrado, por su parte, logró un correcto desempeño al mantener al robot dentro de la zona, salvo por algunas imprecisiones (véase figura 5.47).

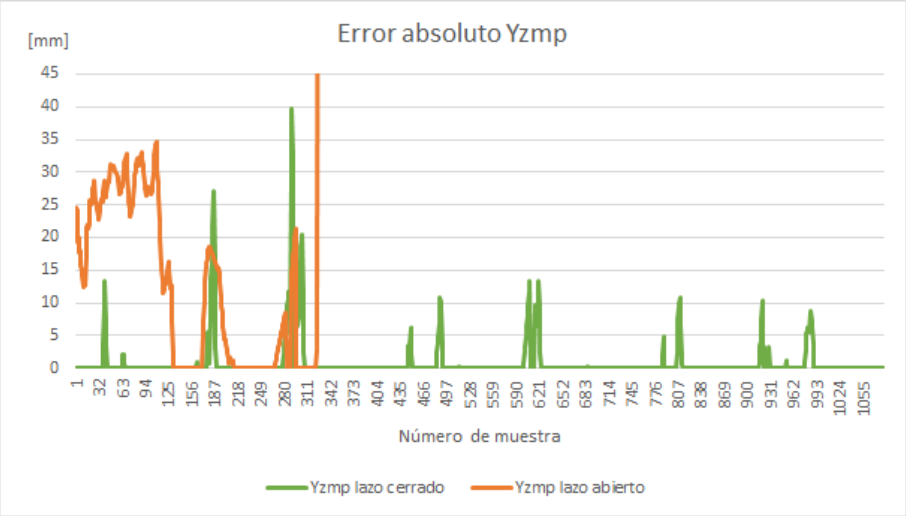


Figura 5.47. Error absoluto en Yzmp durante caminata con carga

En el caso del análisis del error absoluto acumulado, no podemos hacer una comparación entre el lazo abierto y el lazo cerrado durante toda la marcha, sin embargo, el comportamiento de ambos sistemas antes de que el robot cayera muestra un mejor ejercicio de la caminata por parte del sistema realimentado.

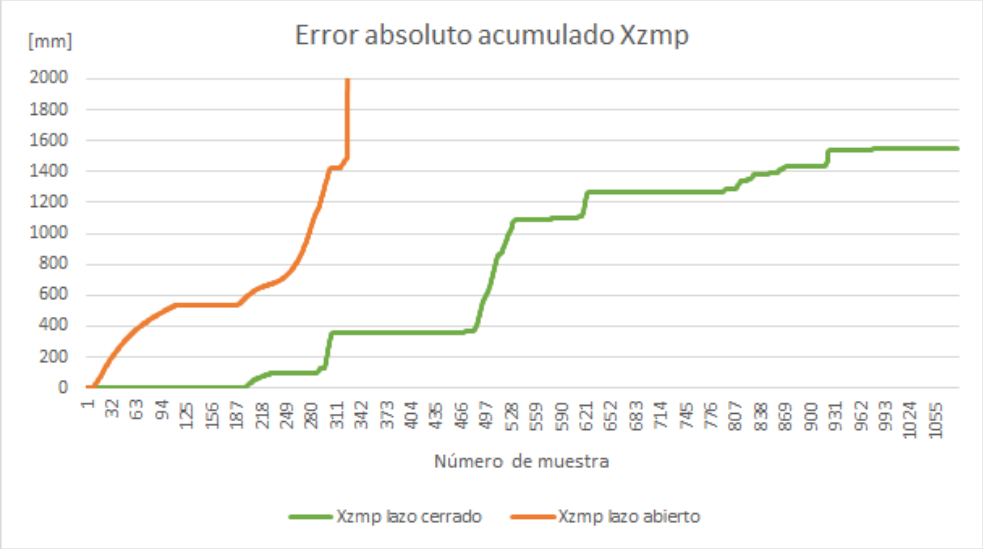


Figura 5.48. Error absoluto acumulado en Xzmp durante caminata con carga

Antes de la caída en lazo abierto, se nota que el error acumulado en Xzmp es menor en lazo cerrado, cuyo último valor supera los 1500 mm (véase figura 5.48).

Al momento de la caída, el sistema en lazo cerrado, tenía un error acumulado en Yzmp cercano a los 4000 mm, un valor superior incluso al acumulado en lazo cerrado al terminar la caminata (véase figura 5.49).

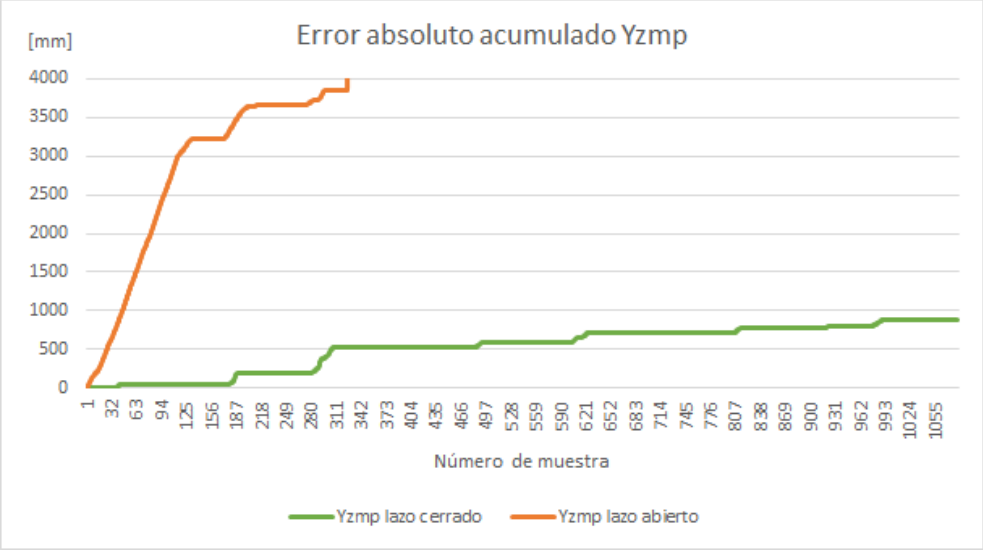


Figura 5.49. Error absoluto acumulado en Yzmp durante caminata con carga

A modo de conclusión, se realizó un análisis de los ZMP calculados durante las pruebas antes mencionadas, con el fin de medir cuántos puntos se encuentran fuera de la zona de tolerancia, mientras el sistema de lazo cerrado se encuentra activo y así compararlo con las muestras tomadas en lazo abierto.

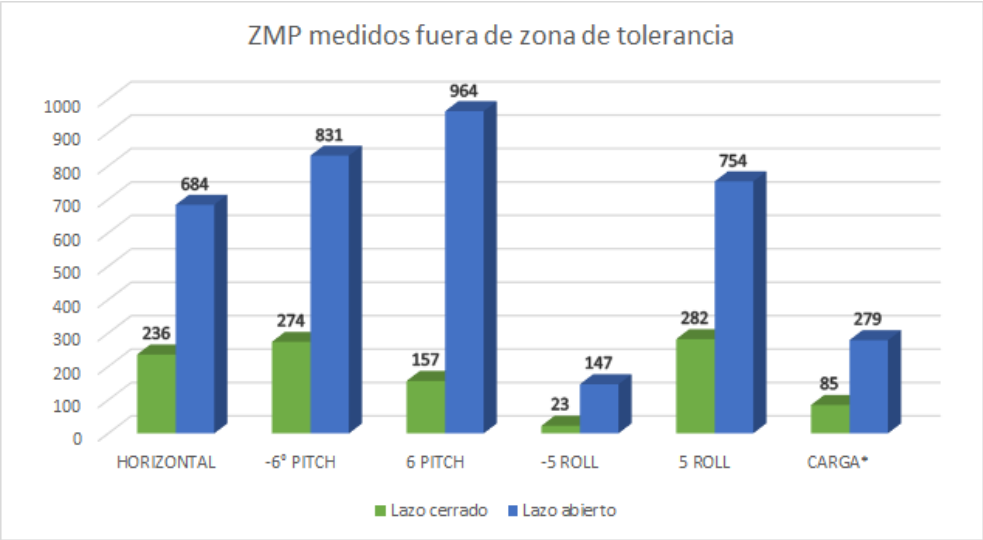


Figura 5.50. ZMP medidos fuera de zona de tolerancia en lazo abierto y cerrado

*En el caso de la prueba con carga, se hace la comparación hasta el número de muestra 324 que es cuando el robot en lazo abierto volcó.

Contrastando los resultados obtenidos, se confirma una mejoría al implementar el sistema de control del ZMP. Al apreciar la figura 5.50 se reafirma, además, que existe una mayor cantidad de puntos en una zona con más inestabilidad cuando el sistema de control no se encuentra en funcionamiento en cada una de las pruebas realizadas.

6. Conclusiones y trabajo a futuro

6.1. Conclusiones

Se lograron implementar sistemas de lazo cerrado de los sistemas de los ángulos pitch y roll de la cadera y las coordenadas del ZMP, respectivamente, y comparar sus acciones de control. El control de los ángulos de la cadera tuvo una buena respuesta ante perturbaciones en pitch, mientras que en roll, suele fallar debido a que la posición horizontal de la cadera no garantiza la estabilidad de la caminata. Por su parte, el sistema de control del ZMP permitió mantenerlo dentro del polígono de soporte logrando una caminata robusta aún con cambios de pendiente. El robot, incluso, fue capaz de realizar un ciclo de caminata con carga extra.

Cabe mencionar que el cambio en el microcontrolador y la interfaz mejoraron notablemente la capacidad de envío de datos permitiendo la implementación de un sistema realimentado.

Además, las consideraciones realizadas en el ancho y longitud de paso, además de los sistemas de aterrizaje, identificación automática de las fases de la caminata y el levantamiento de los pies, permitieron realizar una medición del ZMP aproximada a la real, lo que posibilitó la implementación del sistema de control.

Finalmente, la trayectoria del ZMP de referencia permitió establecer un parámetro de control continuo que minimizó el riesgo de caída. De igual manera, la sintonización de las ganancias de los controladores PID otorgó al robot la suficiente robustez para soportar diversas perturbaciones.

6.2. Trabajo a futuro

Los aportes presentados en esta investigación establecen un punto de partida para la implementación de un sistema de control más robusto. Si bien el método heurístico permitió un comportamiento aceptable del robot ante perturbaciones, la implementación de mejores algoritmos de obtención de las ganancias del controlador PID (Lógica difusa, redes neuronales, método Ziegler-Nichols, etcétera), representará un aumento en la calidad de la respuesta del sistema, entendiendo esto como una disminución del sobrepaso, un menor tiempo de subida y de asentamiento.

A pesar de que la medición de la longitud y ancho de paso resultaron útiles en la implementación del sistema de control, es necesaria una lectura más precisa y en tiempo real durante la caminata, por lo que resulta indispensable implementar un sistema que realice tal tarea.

La trayectoria del ZMP propuesta por medio de ecuaciones de segundo grado resultó ser una referencia continua y adecuada hasta cierto punto; sin embargo, al ser un conjunto de parámetros propuestos, ésta aún puede ser mejorada.

Finalmente, se pretende unir el sistema realimentado de los ángulos pitch y roll con el criterio de control del ZMP y el polígono de soporte, de acuerdo con el diagrama de bloques visto en la figura 6.1.

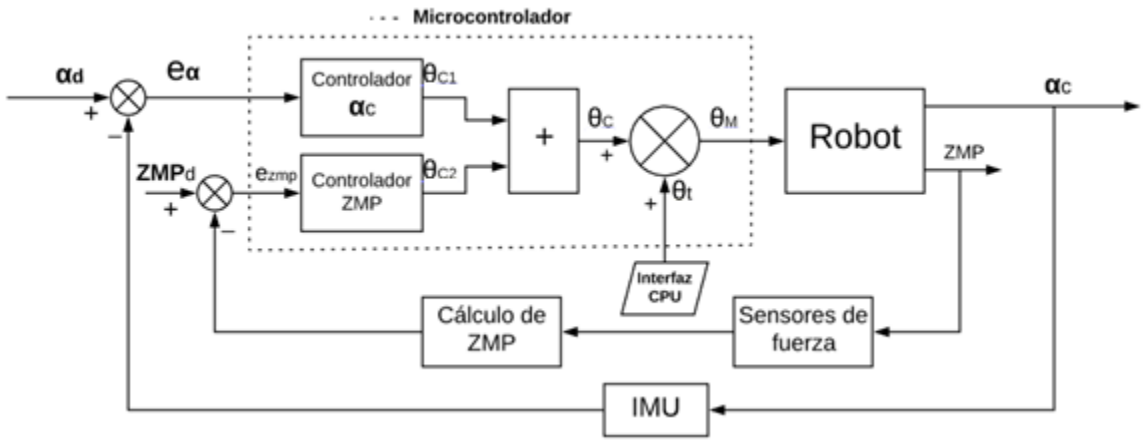


Figura 6.1. Diagrama de bloques de sistema de control de ZMP, pitch y roll

Donde:

α_d son los ángulos deseados de la cadera (pitch y roll).

α_c son los ángulos medidos de la cadera (pitch y roll).

ZMP es el punto de momento cero producido en el robot.

e_α es el error entre los ángulos de la cadera deseados y los medidos.

ZMP_d es el ZMP deseado.

e_{ZMP} es el error entre el ZMP medido y el ZMP referencial.

θ_{C1} es el conjunto de ángulos de corrección para los ángulos de la cadera.

θ_{C2} es el conjunto de ángulos de corrección para el ZMP del robot.

θ_C es el conjunto de ángulos de corrección a los motores del robot.

θt es el conjunto de ángulos teóricos obtenidos a partir de la cinemática inversa y enviados a través de la interfaz gráfica al robot.

θ_M son los ángulos que se envían a cada uno de los motores del robot bípedo.

REFERENCIAS

[1] O. NARVAÉZ AROCHE, Modelos cinemático y dinámico de un robot bípedo de doce grados de libertad internos, Facultad de Ingeniería, Universidad Nacional Autónoma de México, 2010.

[2] Dynamo Electronics, «Robot Bípedo Scout Lynxmotion - Dynamo Electronics,» [En línea]. Available: https://www.dynamoelectronics.com/plataformas/631-robot-bipedo-scout-lynxmotion.html. [Último acceso: 31 Enero 2018].

[3] M. F. MERINO MORALES, Planeación de trayectorias de un robot bípedo con un modelo parametrizado carro mesa, Facultad de Ingeniería, Universidad Nacional Autónoma de México, 2016.

[4] S. HERNÁNDEZ MÁRQUEZ, Renovación y mejoramiento de actuadores e instrumentación de un robot bípedo, Facultad de Ingeniería, Universidad Nacional Autónoma de México, 2016.

[5] A. SÁNCHEZ ORTEGA, Controladores difusos para la postura y marcha de un robot bípedo de 12 GDL, Facultad de Ingeniería, Universidad Nacional Autónoma de México, 2017.

[6] International Federation of Robotics, «Service Robots,» IFR, [En línea]. Available: https://ifr.org/service-robots. [Último acceso: 1 Noviembre 2017].

[7] C. BERNABEU, «1,4 millones de robots serán instalados desde ahora hasta el 2019,» InfoPLC, 3 Febrero 2017. [En línea]. Available: http://www.infoplc.net/plus-plus/mercado/noticias-mercado/item/103943-1,4-millones-robots-2019. [Último acceso: 20 Noviembre 2017].

[8] AliveRobots, «Robot NAO,» [En línea]. Available: https://aliverobots.com/nao/. [Último acceso: 4 Noviembre 2017].

[9] SPL, «RoboCup Standard Platform League,» [En línea]. Available: http://spl.robocup.org/. [Último acceso: 28 Octubre 2017].

[10] D. MELANSON, «Nao robot replaces AIBO in RoboCup Standard Platform League,» Engadget, 16 Agosto 2007. [En línea]. Available: https://www.engadget.com/2007/08/16/nao-robot-replaces-aibo-in-robocup-standard-platform-league/. [Último acceso: 12 Diciembre 2017].

[11] Aldebaran, «NAO Documentation,» [En línea]. Available: http://doc.aldebaran.com/2-1/home_nao.html. [Último acceso: 15 Enero 2018].

[12] J. J. ALCARAZ JIMÉNEZ, Control retroalimentario robusto de marcha bípeda omnidireccional en el robot humanoide Nao, Tesis Doctoral. Universidad de Murcia, 2014.

[13] HONDA, «Historia-HONDA,» [En línea]. Available: https://www.honda.mx/historia/. [Último acceso: 11 Marzo 2018].

[14] HONDA, «Acerca de Honda,» [En línea]. Available: https://www.honda.mx/acerca/. [Último acceso: 11 Marzo 2018].

[15] T. TAKENAKA, «The control system for the Honda humanoid robot.,» *Age and ageing, 2006,* vol. 35, pp. 24-26, 2006.

[16] HONDA, «Honda Worldwide ASIMO History,» [En línea]. Available: https://world.honda.com/ASIMO/history/. [Último acceso: 15 Marzo 2018].

[17] Honda Motor Europe España, «https://www.flickr.com/photos/hondaes/5131594048,» Flickr, 21 Abril 2009. [En línea]. Available: https://www.flickr.com/photos/hondaes/5131594048. [Último acceso: 19 Marzo 2018].

[18] C. IRVINE, «New Generation ASIMO, Honda's Humanoid Robot, Makes U.S. Debut at Consumer Electronics Show, Performances Produced by Martin Brinkerhoff Associates January 8-11, 2007, Las Vegas Convention Center,» Cision PRWeb, 20 Diciembre 2006. [En línea]. Available: http://www.prweb.com/releases/2006/12/prweb492912.htm. [Último acceso: 11 Marzo 2018].

[19] HONDA, «Inside ASIMO Robotics by Honda,» [En línea]. Available: http://asimo.honda.com/inside-asimo/. [Último acceso: 2018 Marzo 13].

[20] ASIMO, «Twitter ASIMO,» 12 Junio 2014. [En línea]. Available: https://twitter.com/asimo/status/477124637348732928. [Último acceso: 15 Marzo 2018].

[21] HONDA, «Asimo,» [En línea]. Available: https://www.honda.mx/asimo/. [Último acceso: 11 Marzo 2018].

[22] Boston Dynamics, «Atlas,» [En línea]. Available: https://www.bostondynamics.com/atlas. [Último acceso: 10 Febrero 2018].

[23] Boston Dynamics, «What's new Atlas?,» Youtube, 16 Noviembre 2017. [En línea]. Available: https://www.youtube.com/watch?v=fRj34o4hN4I. [Último acceso: 2018 Febrero 20].

[24] S. KUINDERSMA, «Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot.,» *Autonomous Robots,* vol. 40, nº 3, pp. 429-455, 2016.

[25] J.-Y. KIM, I.-W. PARK y J.-H. OH, «Design and walking control of the humanoid robot, KHR-2 (KAIST Humanoid Robot-2),» *International Conference on Control Robotics Society,* pp. 1539-1543, 2004.

[26] J.-W. HEO, I.-H. LEE y J.-H. OH, «Development of humanoid robots in HUBO Laboratory, KAIST,» *Journal of the Robotics Society of Japan,* vol. 30, nº 4, pp. 367-371, 2012.

[27] J.-H. KIM, «Development of a humanoid biped walking robot platform KHR-1-initial design and its performance evaluation,» *feedback,* vol. 1, p. 12, 2002.

[28] J.-Y. KIM, «System design and dynamic walking of humanoid robot KHR-2,» *Robotics and Automation,* pp. 1431-1436, 2005.

[29] I.-W. PARK, «Mechanical design of humanoid robot platform KHR-3 (KAIST humanoid robot 3: HUBO),» *Humanoid Robots, 2005 5th IEEE-RAS International Conference on. IEEE,* pp. 321-326, 2005.

[30] J.-H. OH, «Design of android type humanoid robot Albert HUBO,» *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on. IEEE,* pp. 1428-1433, 2006.

[31] M. A. ALI, H. A. PARK y C. G. LEE, «Closed-form inverse kinematic joint solution for humanoid robots,» *En Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on. IEEE,* pp. 704-709, 2010.

[32] E. GUIZZO y E. ACKERMAN, «The hard lessons of DARPA's robotics challenge [News],» *IEEE Spectrum,* vol. 52, nº 8, pp. 11-13, 2015.

[33] HUBO Lab, «DRC-HUBO+,» [En línea]. Available: http://hubolab.kaist.ac.kr/p_drchuboplus. [Último acceso: 2018 Septiembre 20].

[34] M. VUKOBRATOVIĆ y J. STEPANENKO, «On the stability of anthropomorphic systems,» *Mathematical biosciences, 1972, vol. 15, no 1-2, p. 1-37,* vol. 15, nº 1-2, pp. 1-37, 1972.

[35] C.-L. SHIH, «Trajectory synthesis and physical admissibility for a biped robot during the single-support phase,» *Robotics and Automation, 1990. Proceedings, 1990 IEEE International Conference on. IEEE,* pp. 1646-1652, 1990.

[36] J. H. PARK y Y. K. RHEE, «ZMP trajectory generation for reduced trunk motions of biped robots,» *En Intelligent Robots and Systems, 1998. Proceedings. 1998 IEEE/RSJ International Conference on. IEEE,* pp. 90-95, 1998.

[37] S. KAGAMI, «A fast generation method of a dynamically stable humanoid robot trajectory with enhanced zmp constraint,» *Proceedings of IEEE International Conference on Humanoid Robotics (Humanoid2000),* 2000.

[38] K. ERBATUR y O. KURT, « Natural ZMP trajectories for biped robot reference generation,» *IEEE Transactions on Industrial Electronics,* vol. 56, nº 3, pp. 835-845, 2009.

[39] H.-K. SHIN y B. K. KIM, «Energy-efficient gait planning and control for biped robots utilizing the allowable ZMP region,» *IEEE Transactions on Robotics,* vol. 30, nº 4, pp. 986-993, 2014.

[40] S. KIM, «Human motion analysis and its application to walking stabilization with COG and ZMP,» *IEEE Transactions on Industrial Informatics,* 2018.

[41] M. GIENGER, «Walking control of a biped robot based on inertial measurement,» *Proc. Int. Workshop. on Humanoid and Human Friendly Robotics. Tsukuba, 2002,* 2002.

[42] S. LOHMEIER, Design and realization of a humanoid robot for fast and autonomous bipedal locomotion, Fakultät für Maschinenwesen, Technischen Universität München., 2010.

[43] J.-Y. KIM, I.-W. PARK y J.-H. OH, «Experimental realization of dynamic walking of the biped humanoid robot KHR-2 using zero moment point feedback and inertial measurement,» *Advanced Robotics,* vol. 20, nº 6, pp. 707-736, 2006.

[44] H. SUZUKI, J. H. LEE y S. OKAMOTO, «Locomotion Control of 7-Linked Walking Robot Embedded with CPG Using Neural Oscillator space,» *Proceedings of the International MultiConference of Engineers and Computer Scientists.,* 2018.

[45] A. O. BATURONE, Robótica: manipuladores y robots móviles, Marcombo, 2005, p. 2.

[46] R. DRAKE, Gray's Anatomy for Students E-Book, Elsevier Health Sciences, 2015.

[47] M. H. P. DEKKER, Zero-moment point method for stable biped walking, Eindhoven University of Technology, 2009.

[48] F. P. BEER, Mecánica vectorial para ingenieros, McGraw-Hill, 1990.

[49] L. A. Medina Orozco, «ESTÁTICA | FÍSICA MECÁNICA,» Blogspot, [En línea]. Available: http://fisicamecanicaluismedina.blogspot.com/2015/03/la-estatica.html. [Último acceso: 15 Septiembre 2018].

[50] B. SICILIANO y O. KHATIB, Springer handbook of robotics, Springer, 2016.

[51] P. SARDAIN y G. BESSONNET, «Forces acting on a biped robot. center of pressure-zero moment point,» *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans,* vol. 34, nº 5, pp. 630-637, 2004.

[52] R. P. ARENY, Sensores y acondicionadores de señal, Marcombo, 2004.

[53] E. W. WEISSTEIN, «Euler's Rotation Theorem,» MathWorld-A Wolfram Web, [En línea]. Available: http://mathworld.wolfram.com/EulersRotationTheorem.html. [Último acceso: 10 Abril 2018].

[54] G. LÓPEZ y S. MARGNI, Motores y Sensores en Robótica, Uruguay, 2003, p. 17.

[55] N. PINCKNEY, «Pulse-width modulation for microcontroller servo control,» *IEEE potentials,* vol. 25, p. 29, 2006.

[56] A. CASTILLO, Apuntes para la materia de cinemática de las máquinas, S.L.P.: Universidad Autónoma de San Luis Potosí, 2005, p. 28.

[57] JHDEEDAR, «The World on the Hand of Microcontroller : 11 Steps,» Autodesk, Inc, [En línea]. Available: https://www.instructables.com/id/The-World-on-the-Hand-of-Microcontroller/. [Último acceso: 25 Febrero 2018].

[58] L. HERRERA, «Viviendas inteligentes (domótica),» *Ingeniería e Investigación,* vol. 25, nº 2, pp. 47-52, 2005.

[59] G. GONZÁLEZ, «Diseño e implementación de una Tarjeta de Desarrollo con profundización en desarrollo de aplicación de Touch Sensing,» *LACCEI 2013,* p. 2, 2013.

[60] ARDUINO, «Arduino Uno Rev3,» [En línea]. Available: https://store.arduino.cc/usa/arduino-uno-rev3. [Último acceso: 12 Febrero 2018].

[61] S. KAJITA, «Biped walking pattern generation by using preview control of zero-moment point,» *ICRA,* pp. 1620-1626, 2003.

[62] HITEC RCD, «HS-5645MG High Torque, Metal Gear Digital Sport Servo,» [En línea]. Available: https://hitecrcd.com/products/servos/sport-servos/digital-sport-servos/hs-5645mg/product. [Último acceso: 13 Enero 2018].

[63] N. AHMAD, «Reviews on various inertial measurement unit (IMU) sensor applications,» *International Journal of Signal Processing Systems,* vol. 1, nº 2, pp. 256-262, 2013.

[64] POLOLU, «Pololu - UM7-LT Orientation Sensor,» [En línea]. Available: https://www.pololu.com/product/2740. [Último acceso: 15 Enero 2018].

[65] L. CARRIÓN, D. OCHOA y J. A. VALVERDE, «SENSOR DE FUERZA RESISTIVO (FSR),» National Instruments, 2009. [En línea]. Available: http://www.datalights.com.ec/site2/images/stories/robotica/nap/nap_fsr.pdf. [Último acceso: 1 Febrero 2018].

[66] HETPRO, «Sensor Flexiforce A201,» [En línea]. Available: https://hetpro-store.com/sensor-flexiforce-a201. [Último acceso: 15 Diciembre 2017].

[67] TEKSCAN, «Small Force Sensing Resistor | FlexiForce A201 Sensor,» [En línea]. Available: https://www.tekscan.com/products-solutions/force-sensors/a201. [Último acceso: 15 Diciembre 2017].

[68] F. ZAMBRANO, «La usabilidad, entre la tecnología y la pedagogía.,» *Tema del mes,* p. 5, 2007.

[69] Y. C. LÓPEZ, O. P. RODRÍGUEZ y R. C. ROLDÁN, Iniciación a la programación en C#: un enfoque práctico, Delta Publicaciones, 2007.

[70] N. PEÑA, «microsoftNET-2.pptx,» Google Drive, [En línea]. Available: https://drive.google.com/file/d/0B3KikGnUAmmOckdzMXBOTDJESWs/view. [Último acceso: 15 Diciembre 2017].

[71] ECMA, «Standard ECMA-334,» Diciembre 2017. [En línea]. Available: https://www.ecma-international.org/publications/standards/Ecma-334.htm. [Último acceso: 28 Diciembre 2017].

[72] ARDUINO, «Arduino Mega 2560 Rev3,» [En línea]. Available: https://store.arduino.cc/usa/arduino-mega-2560-rev3. [Último acceso: 26 Diciembre 2017].

[73] TEXAS INSTRUMENTS, «EK-TM4C123GXL | Evaluations Modules & Boards | TI Store,» [En línea]. Available: https://store.ti.com/Tiva-C-LaunchPad.aspx. [Último acceso: 26 Diciembre 2017].

[74] TEXAS INSTRUMENTS, «SW-EK-TM4C123GXL Tiva (TM) C Series LauncPad Evaluation Board Software | TI.com,» [En línea]. Available: http://www.ti.com/tool/sw-ek-tm4c123gxl. [Último acceso: 26 Diciembre 2017].

[75] J. L. DEVORE, Probabilidad y estadística para ingeniería y ciencias, Séptima ed., Cengage Learning Editores, 2008, pp. 519-528.

[76] M. VUKOBRATOVIĆ y B. BOROVAC, «Zero-moment point—thirty five years of its life,» *International journal of humanoid robotics,* vol. 1, nº 1, pp. 157-173, 2004.

[77] K. OGATA, Ingeniería de control moderna, Pearson Educación, 2010.

[78] W. BOLTON, Ingeniería de control, Marcombo, 2001.

[79] V. MAZZONE, Controladores pid, Universidad Nacional de Quilmes, 2002.

[80] G. J. COSTA, «Tuning a PID controller,» *Power Transmission Engineering,* vol. 5, nº 2, pp. 26-31, 2011.

[81] K. H. ANG, G. CHONG y Y. LI, «PID control system analysis, design, and technology,» *IEEE transactions on control systems technology,* vol. 13, nº 4, pp. 559-576, 2005.

[82] DIARIO OFICIAL DE LA FEDERACIÓN, «DOF,» SEGOB, 12 Septiembre 2013. [En línea]. Available: http://www.dof.gob.mx/nota_detalle.php?codigo=5313974&fecha=12/09/2013. [Último acceso: 27 Julio 2018].

[83] U. JIMÉNEZ RIOJA, Equilibrio dinámico de un robot utilizando el modelo simplificado carro-mesa, Facultad de Ingeniería, Universidad Nacional Autónoma de México, 2014.

[84] J.-Y. KIM, I.-W. PARK y J.-H. OH, «Walking Control Algorithm of Biped Humanoid Robot on Uneven and Inclined Floor,» *J Intell Robot Syst,* vol. 48, pp. 457-484, 2007.

[85] National Aeronautics and Space Administration, «ANTHROPOMETRY AND BIOMECHANICS,» [En línea]. Available: https://msis.jsc.nasa.gov/sections/section03.htm. [Último acceso: 20 Enero 2018].

APÉNDICES

A. Código en C ++ - Tiva C Series TM4C123G

```
#include <Servo.h> //Libreria Servos
//PROGRAMA DE SISTEMA DE CONTROL DE ZMP DE ROBOT BÍPEDO DE 12 GDL
//PROGRAMA CREADO POR EDUARDO ONTIVEROS GARCÍA
int numerosBinarios [16][4] =
{
  {1, 1, 1, 1}, //ch1
  {1, 1, 1, 0}, //ch2
  {1, 1, 0, 1}, //ch3
  {1, 1, 0, 0}, //ch4
  {1, 0, 1, 1}, //ch5
  {1, 0, 1, 0}, //ch6
  {1, 0, 0, 1}, //ch7
  {1, 0, 0, 0}, //ch8
  {0, 1, 1, 1}, //ch9
  {0, 1, 1, 0}, //ch10
  {0, 1, 0, 1}, //ch11
  {0, 1, 0, 0}, //ch12
  {0, 0, 1, 1}, //ch13
  {0, 0, 1, 0}, //ch14
  {0, 0, 0, 1}, //ch15
  {0, 0, 0, 0} //ch16
};

//Polinomio de lectura de FSR (Hernández, 2016)
float conversionIzquierdo[9][5] =     {
  { +0.1762, -0.9540, +1.8915, +0.3617, +0.0052},
  { +0.1849, -0.9479, +1.7859, +0.3678, +0.0058},
  { +0.1297, -0.5123, +0.6333, +1.8661, +0.0095},
  { +0.2463, -1.3114, +2.6863, -0.1932, +0.0026},
  { +0.0000, +0.0000, +0.0000, +0.0000, +0.0000},
  { +0.2463, -1.3114, +2.6863, -0.1932, +0.0026},
  { +0.1849, -0.9479, +1.7859, +0.3678, +0.0058},
  { +0.1297, -0.5123, +0.6333, +1.8661, +0.0095},
  { +0.1849, -0.9479, +1.7859, +0.3678, +0.0058},
};
float conversionDerecho[9][5] =     {
  { +0.1762, -0.9540, +1.8915, +0.3617, +0.0052},
  { +0.1849, -0.9479, +1.7859, +0.3678, +0.0058},
  { +0.1297, -0.5123, +0.6333, +1.8661, +0.0095},
  { +0.2463, -1.3114, +2.6863, -0.1932, +0.0026},
  { +0.0000, +0.0000, +0.0000, +0.0000, +0.0000},
  { +0.2463, -1.3114, +2.6863, -0.1932, +0.0026},
  { +0.1849, -0.9479, +1.7859, +0.3678, +0.0058},
  { +0.1297, -0.5123, +0.6333, +1.8661, +0.0095},
  { +0.1849, -0.9479, +1.7859, +0.3678, +0.0058},
};

//VARIABLES ZMP
const int bit1_sen = 3 , bit2_sen = 4, bit3_sen = 8, bit4_sen = 9, EN_sen =
10, an = 2 ;
//Pines que enviarán el número en binario al multiplexor.
//Polinomios de referencia ZMP
```

```
52
53     float referenciaXzmp[11][3] = {
54       { +0.00086, -0.19059, +20.1897},//1.  SDA 1
55       { +0.00550, -0.38231, +15.3768},//2.  SSI 1
56       { +0.00723, +0.78099, +29.2118},//3.  SDD 1
57       { +0.00386, -0.22325, +10.2194},//4.  SSD 1
58       { +0.00879, +0.55517, -34.5640},//5.  SDI 1
59       { +0.00631, -0.66961, +32.6633},//6.  SSI 2
60       { +0.00221, +0.62113, +29.3767},//7.  SDD 2
61       { +0.00387, -0.21786, +10.2140},//4.  SSD 2
62       { -0.00304, +0.92184, -34.9188},//5.  SDI 2
63       { +0.00162, +0.12216, +9.87262},//10. SSI 3
64       { +0.00111, -0.21639, +25.2153},//11. SDA 2
65     };
66
67
68     float referenciaYzmp[11][3] = {
69       { -0.00084, -0.01644, +40.0173},//1.  SDA 1
70       { +0.00824, -0.82850, +7.82022},//2.  SSI 1
71       { +0.00723, +1.23554, +3.75723},//3.  SDD 1
72       { -0.00660, +0.75758, -5.75100},//4.  SSD 1
73       { +0.00647, -1.71780, +75.7113},//5.  SDI 1
74       { +0.00501, -0.41730, -4.58770},//6.  SSI 2
75       { -0.00400, +1.33207, +3.67197},//7.  SDD 2
76       { -0.00680, +0.77551, -5.76870},//4.  SSD 2
77       { +0.00808, -1.76780, +75.7597},//5.  SDI 2
78       { +0.00587, -0.28530, -1.72060},//10. SSI 3
79       { -0.00050, +0.29257, +4.70792},//11. SDA 2
80     };
81     //Polinomios de ZMP de referencia
82    float ax = referenciaXzmp[0][0], bx = referenciaXzmp[0][1], cx =
      referenciaXzmp[0][2];
83    float ay = referenciaYzmp[0][0], by = referenciaYzmp[0][1], cy =
      referenciaYzmp[0][2];
84     int arb = 0;
85     //Inicialización de valores
86     float Xref = cx, Yref = cy;
87    float lecturaTensionDerecho[9] = {0, 0, 0, 0, 0, 0, 0, 0, 0}; //Vector de
      lecturas de los sensores pie derecho.
88     float lecturaTensionIzquierdo[9] = {0, 0, 0, 0, 0, 0, 0, 0, 0};
89    float fuerzaDerecho[9] = {0, 0, 0, 0, 0, 0, 0, 0, 0};     //Vector de fuerzas
      pie derecho.
90    float fuerzaIzquierdo[9] = {0, 0, 0, 0, 0, 0, 0, 0, 0};   //Vector de fuerzas
      pie izquierdo.
91    float matrizDerecho[3][3] = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}}; //Matriz de
      fuerzas pie derecho.
92    float matrizIzquierdo[3][3] = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}}; //Matriz de
      fuerzas pie izquierdo.
93     float numeradorXd = 0, numeradorYd = 0, denominadord = 0;
94     float numeradorXi = 0, numeradorYi = 0, denominadori = 0;
95    float flexi, Xzmpi , Yzmpi, Xzmpd, Yzmpd, Yzmp, Xzmp, Xprom, Yprom, Xiprom,
      Yiprom, Xdprom, Ydprom;
96     //Valores auxiliares en promedio de ZMP
97     int f = 0, pf = 7;
98     float Xs[7];
```

```
99    float Ys[7];
100   float Xis[7];
101   float Yis[7];
102   float Xds[7];
103   float Yds[7];
104   //Distancias entre pies
105   float DECY = 79, DECX = 0;
106   //Ubicación de FSR
107   float posicionYi[3] = {13.5, -7.5, -28.5};
108   float posicionYd[3] = {28.5, 7.5, -13.5};
109   float posicionX[3] = { -13, 20, 54};
110   //Valor de alzar el pie
111   float alzar = 0;
112   //variables de ciclo for
113   int i, j, k;
114   //VARIABLES PROGRAMA SERVOS
115   int tiempo = 0;
116   int valor;
117   //Variable de lectura de ángulos
118   char sensor[20];
119   byte cont = 0, caida = 0;
120   int cs = 1, minim = 900, maxim = 2100;
121   Servo s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11, s12;
122
123
124   int linea = 0;
125   int P = 1;
126   // Servos en posición incial
127   float ang[12] = {90, 95, 85, 92, 43, 154, 89, 93, 46, 135, 100, 80};
128   //errores específicos de cada fase
129   float errssi[12] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
130   float errssd[12] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
131   float errsd[12] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
132   //Ángulos de corrección
133   float angcor[12] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
134   //Errores de salida del controlador
135   float errkp = 0, errdp = 0, errip = 0, errkr = 0, errdr = 0, errir = 0;
136   //Ganancias
137   float kix = 0.0, kdx = 0.0, kpx = 0.0, kiy = 0.0, kpy = 0.0, kdy = 0.0;
138   float kpx2 = 0.0, kix2 = 0.0, kpy2 = 0.0, kiy2 = 0.0;
139  float kp1 = 0.4, ki1 = 0.002, kd1 = 0.3, kp2 = 0.1, ki2 = 0.00025, kd2 =
     0.015;
140   float errant1 = 0, errant2 = 0;
141   unsigned long antes;
142   //Tiempo de muestreo y dt
143   int muestreo = 100, dt = 0;
144   byte PieI[3] = {0, 0, 0};
145   byte PieD[3] = {0, 0, 0};
146   float sumas[3] = {0, 0, 0};
147   //variables auxiliares
148   byte control = 0, estaux = 1, piecor = 0;
149   //Control ZMP
150  float errpx = 0, errpy = 0, errix = 0, erriy = 0, errdx = 0, errdy = 0,
     errantx = 0, erranty = 0;
151   //contador de pasos
```

```
byte paso = 0;


float fuerzaI = 0, fuerzaD = 0;
void setup() {

  Serial.begin(230400); //Iniciamos el serial
  //SERVOS
  s1.attach(40, minim, maxim); //PF_2
  s2.attach(39, minim, maxim);
  s3.attach(38, minim, maxim);
  s4.attach(37, minim, maxim);
  s5.attach(36, minim, maxim);
  s6.attach(35, minim, maxim);
  s7.attach(34, minim, maxim);
  s8.attach(31, minim, maxim);
  s9.attach(11, minim, maxim);
  s10.attach(12, minim, maxim);
  s11.attach(13, minim, maxim);
  s12.attach(14, minim, maxim);
  actualizar_servos();
  //Conexión con multiplexor
  pinMode(bit1_sen, OUTPUT);
  pinMode(bit2_sen, OUTPUT);
  pinMode(bit3_sen, OUTPUT);
  pinMode(bit4_sen, OUTPUT);
  pinMode(EN_sen, OUTPUT);
  digitalWrite(EN_sen, 0);
  pinMode(PF_1, OUTPUT);
}

void loop() {
  //Medidia de dt
  unsigned long ahora = millis();

  // Lectura de ángulos de entrada
  if (Serial.available() > 0) {
    cont = 0;
    memset(sensor, 0, sizeof(sensor));
    while (Serial.available() > 0) {
      delayMicroseconds(200);
      sensor[cont] = Serial.read();
      cont++;
    }
    if (cont == 1){
      valor = (sensor[0] - '0');
    }
    else if (cont == 2) {
      valor = ((sensor[0] - '0') * 10 + (sensor[1] - '0'));
    }
    else if (cont == 3) {
      valor = (sensor[0] - '0') * 100 + (sensor[1] - '0') * 10 + (sensor[2] -
      '0');
    }
    //caso de ángulos de motores
```

```
206         if (valor < 200) {
207           ang[cs - 1] = valor;
208 
209           if (linea == 331 || linea == 674) {
210             estaux = 2;
211             analogWrite(PF_1, 200);
212           }
213 
214           else if (linea == 190 || linea == 506 || linea == 848) {
215             analogWrite(PF_1, 0);
216             estaux = 0;
217           }
218           else if (linea < 2 || linea == 287 || linea == 445 || linea == 608 ||
        linea == 787 || linea == 919) {
219 
220               if (estaux == 0 || estaux == 2) {
221                 paso++;
222                 //Cambio de longitud de paso para medición de ZMP
223                 switch (paso) {
224                   case 1:
225                     DECX = 69;
226                     break;
227                   case 2:
228                     DECX = -69;
229                     break;
230                   case 3:
231                     DECX = 69;
232                     break;
233                   case 4:
234                     DECX = -69;
235                     break;
236                   case 5:
237                     DECX = 0;
238                     break;
239                   default:
240                     break;
241 
242                 }
243               }
244               estaux = 1;
245               analogWrite(PF_1, 50);
246           }
247           //Siguiente servo
248           cs++;
249         }
250         else { //Casos de envío de alertas
251           if (valor == 315) { //activación/desactivación de sistema de control
252             if (control == 0) {
253               control = 1;
254             }
255             else {
256               control = 0;
257             }
258           }
259           else { //en caso de caída
```

```
260           if (valor == 310) {
261             paso = 0;
262             estaux = 1;
263             errix = 0;
264             erriy = 0;
265             errpx = 0;
266             errpy = 0;
267             float ang[12] = {90, 95, 85, 92, 43, 154, 89, 93, 46, 135, 90, 90};
268             actualizar_servos();
269             linea = 0;
270             arb = 0;
271             P = 1;
272           }
273           //Trayectoria de ZMP de referencia
274           if (linea == 0 || linea == 190 ||  linea == 287 || linea == 331 ||
          linea == 445 ||
275               linea == 506  || linea == 608  || linea == 674 || linea == 787 ||
          linea == 848
276               || linea == 919 ) {
277             ax = referenciaXzmp[arb][0];
278             bx = referenciaXzmp[arb][1];
279             cx = referenciaXzmp[arb][2];
280             ay = referenciaYzmp[arb][0];
281             by = referenciaYzmp[arb][1];
282             cy = referenciaYzmp[arb][2];
283             arb++;
284             P = 1;
285           }
286           linea++;
287           Xref = P * P * ax + P * bx + cx;
288           Yref = P * P * ay + P * by + cy;
289           P++;
290           if (caida == 0) {
291             Serial.print(Xref);
292             Serial.print("\t");
293             Serial.print(Yref);
294             Serial.print("\t");
295             Serial.print(Xzmp);
296             Serial.print("\t");
297             Serial.println(Yzmp); }
298           else { //Alerta de caída a interfaz
299             Serial.print("231");
300             Serial.print("\t");
301             Serial.print("231");
302             Serial.print("\t");
303             Serial.print("231");
304             Serial.print("\t");
305             Serial.println("231");
306           }
307           cs = 1;
308         }
309       }
310     }
311     angulos();
312     dt = ahora - antes;
```

```
313     if (dt >= muestreo)
314     {
315       calculo_ZMP();
316       //SISTEMAS DE CONTROL ESPECÍFICOS DE CADA FASE
317       if (control == 1) {
318         Controlador_ZMP();
319         if (estaux == 0) {
320             kpx = 0.04;
321             kix = 0.0001;
322             kdx = 0.005;
323             kpy = 0.04;
324             kiy = 0.0002;
325             kdy = 0.01;
326             alzar = 0.08;
327           corpie();
328           errssi[9] = errssi[9] + errpx * kpx + errix * kix - errdx * kdx;
329           errssi[7] = errssi[7] + errpx * kpx + errix * kix - errdx * kdx;
330             if (P < 10) {
331               errssi[10] = 4 * alzar * P ;
332               errssi[2] = 20 * alzar * P ;
333               errssi[8] = 6.5 * alzar * P;
334               errssi[6] = 3 * alzar * P;
335               errssi[4] = 0.5 * alzar * P;
336               errssi[11] =  8 * alzar * P;
337           }
338           errssi[11] = errssi[11] - errpy * kpy - erriy * kiy + errdy * kdy;
339           //Anular otros sistemas
340           errsd[4] = errsd[4] / 1.35;
341           errsd[5] = errsd[5] / 1.35;
342           errsd[6] = errsd[6] / 1.35;
343           errsd[7] = errsd[7] / 1.35;
344           errsd[8] = errsd[8] / 1.35;
345           errsd[9] = errsd[9] / 1.35;
346           errsd[10] = errsd[10] / 1.35;
347           errsd[11] = errsd[11] / 1.35;
348         }
349         if (estaux == 1) { //SD
350           if (paso == 0) { //SDA
351             kpx = 0.05;     kix = 0.005;    kdx = 0.1;
352             kpx2 = 0.005;   kix2 = 0.0005;
353             kpy = 0.005;    kiy = 0.0005;    kdy = 0.02;
354             kpy2 = 0.0001;  kiy2 = 0.0001;
355           }
356           if (paso == 1 || paso == 3) { //SDD
357             if (P > 4) {
358               kpx = 0.01;   kix = 0.0001;    kdx = -0.01;
359               kpx2 = 0.01;  kix2 = 0.000;
360               kpy = 0.003;    kiy = 0.00001;    kdy = 0.0001;
361               kpy2 = 0.002;  kiy2 = 0.000;
362             }
363             else {
364               corpie();
365               anular();
366             }
367           }
```

```
368           if (paso == 2 || paso == 4) { //SDI
369             if (P > 4) {
370               kpx = 0.025;   kix = 0.0001;    kdx = 0.01;
371               kpx2 = 0.006;  kix2 = 0.000;
372               kpy = 0.006;    kiy = 0.0001;    kdy = 0.00;
373               kpy2 = 0.003;  kiy2 = 0.000;
374             }
375             else{
376               corpie();
377               anular();
378               errsd[3] = -P * 1;
379             }
380           }
381           else if (paso == 5){
382             kpx = 0.05;     kix = 0.005;    kdx = 0.1;
383             kpx2 = 0.005;   kix2 = 0.0005;
384             kpy = 0.005;    kiy = 0.0005;    kdy = 0.02;
385             kpy2 = 0.0001;  kiy2 = 0.0001;
386             if (P < 4) {
387               corpie();
388               anular();
389             }
390           }
391             if (errpy * errpy > 100) {
392               errsd[4] = errsd[4] - kpy * errpy;
393               errsd[6] = errsd[6] - 2 * kpy * errpy;
394               errsd[8] = errsd[8] - kpy * errpy;
395               errsd[5] = errsd[5] - kpy * errpy;
396               errsd[7] = errsd[7] - 2 * kpy * errpy;
397               errsd[9] = errsd[9] - kpy * errpy;
398               errsd[10] = errsd[10] - 6 * kpy * errpy  - 5 * kiy * erriy - kdy
                  * errdy ;
399               errsd[11] = errsd[11] - 6 * kpy * errpy  - 5 * kiy * erriy - kdy
                  * errdy ;
400               corpie();
401             }
402             if (errpx * errpx > 40) {
403               errsd[8] = errsd[8] - kpx * errpx - kix * errix - kdx * errdx ;
404               errsd[6] = errsd[6] - kpx * errpx - kix * errix - kdx * errdx ;
405               errsd[9] = errsd[9] + kpx * errpx + kix * errix + kdx * errdx ;
406               errsd[7] = errsd[7] + kpx * errpx + kix * errix + kdx * errdx ;
407             }
408
409           //Otros sistemas
410           for (j = 2; j < 12 ; j++) {
411             errssi[j] = errssi[j] / 1.4;
412             errssd[j] = errssd[j] / 1.2;
413           }
414         }
415         if (estaux == 2) { //SSD
416             if (P > 15) {
417               kpx = 0.12;
418               kix = 0.005;
419               kdx = 0.01;
420               kpy = 0.06;
```

```
421               kiy = 0.0003;
422               kdy = 0.01;
423               alzar = 0.2;
424             }
425               if (P <= 4){
426                 anular();
427             }
428           errssd[8] = errssd[8] - kpx * errpx - kix * errix + kdx * errdx;
429           errssd[6] = errssd[6] - kpx * errpx - kix * errix + kdx * errdx;
430           errssd[10] = errssd[10] - kpy * errpy - kiy * erriy + kdy * errdy;
431           //Otros sistemas
432           if (paso == 1 || paso == 3) {
433             for (j = 2; j < 12 ; j++) {
434               errsd[j] = errsd[j] / 1.15;
435             }
436           }
437           else {
438             for (j = 2; j < 12 ; j++) {
439               errsd[j] = errsd[j] / 1.08;
440             }
441           }
442         }
443       }
444       antes = ahora;
445     }
446     actualizar_servos();
447   }
448   void actualizar_servos(){
449     sumas[0] = 0;
450     sumas[1] = 0;
451     sumas[2] = 0;
452     for (i = 0; i < 12 ; i++) {
453       sumas[0] = sumas[0] + errssi[i];
454       sumas[1] = sumas[1] + errssd[i];
455       sumas[2] = sumas[2] + errsd[i];
456       if (control == 1) {
457         angcor[i] = ang[i] + errssi[i] + errssd[i] + errsd[i];
458       }
459       else {
460         angcor[i] = ang[i];
461         errsd[i] = 0;
462         errssd[i] = 0;
463         errssi[i] = 0;
464       }
465       if (angcor[i] > 180) {
466         angcor[i] = 180;
467       }
468       if (angcor[i] < 0) {
469         angcor[i] = 0;
470       }
471     }
472     s1.write(angcor[0]);
473     s2.write(angcor[1]);
474     s3.write(angcor[2]);
475     s4.write(angcor[3]);
```

```
476     s5.write(angcor[4]);
477     s6.write(angcor[5]);
478     s7.write(angcor[6]);
479     s8.write(angcor[7]);
480     s9.write(angcor[8]);
481     s10.write(angcor[9]);
482     s11.write(angcor[10]);
483     s12.write(angcor[11]);
484   }
485   void enviarBinario(boolean a, boolean b, boolean c, boolean d){
486     //CONFIGURACIÓN DE MULTIPLEXOR
487     digitalWrite(bit1_sen, d);
488     digitalWrite(bit2_sen, c);
489     digitalWrite(bit3_sen, b);
490     digitalWrite(bit4_sen, a);
491   }
492   float tensionFuerza(float tension, float a, float b, float c, float d, float
      e) {
493     //TRANSFORMACIÓN DE FUERZA A DIFERENCIA DE POTENCIAL
494     float Fuerza;
495     if (tension > 0) {
496       Fuerza = +a * tension * tension * tension * tension
497                + b * tension * tension * tension
498                + c * tension * tension
499                + d * tension
500                + e;
501     }
502     else {
503       Fuerza = 0;
504     }
505     return Fuerza;
506   }
507
508   void calculo_ZMP() {
509     k = 0;
510     for ( i = 0 ; i < 3 ; i++){
511       PieD[i] = 0;
512       PieI[i] = 0;
513     }
514     numeradorYi = 0;
515     numeradorXi = 0;
516     denominadori = 0;
517     numeradorYd = 0;
518     numeradorXd = 0;
519     denominadord = 0;
520   //Lectura de sensores
521     for ( j = 0 ; j < 16 ; j++){
522       enviarBinario(numerosBinarios[j][0], numerosBinarios[j][1],
        numerosBinarios[j][2], numerosBinarios[j][3]); //envía el respectivo canal
        seleccionado
523       flexi = analogRead(an);
524       Pie(j + 1, flexi);//85
525       if (flexi < 130) {
526         flexi = 0;
527       }
```

```
528         if (j < 8){
529           lecturaTensionIzquierdo[k] = flexi * (3.3 / 4095);
530         }
531         else {
532           lecturaTensionDerecho[k - 9] = flexi * (3.3 / 4095);
533         }
534         if (k == 3 || k == 12) {
535           k++;
536         };
537         k++;
538       }
539       fuerzaI = 0;
540       fuerzaD = 0;
541       for (i = 0; i < 9 ; i++) {
542         fuerzaIzquierdo[i] =
          tensionFuerza(lecturaTensionIzquierdo[i], conversionIzquierdo[i][0],
          conversionIzquierdo[i][1],conversionIzquierdo[i][2],
          conversionIzquierdo[i][3],conversionIzquierdo[i][4]);
543         fuerzaDerecho[i] = tensionFuerza(lecturaTensionDerecho[i],
          conversionDerecho[i][0], conversionDerecho[i][1], conversionDerecho[i][2],
          conversionDerecho[i][3], conversionDerecho[i][4]);
544         //Medición de fuerza en pie izquierdo
545         fuerzaI = fuerzaI + fuerzaIzquierdo[i];
546         //Medición de fuerza en pie derecho
547         fuerzaD = fuerzaD + fuerzaDerecho[i];
548       }
549       if (fuerzaI + fuerzaD == 0) {
550         caida = 1;
551         control = 0;
552       }
553       else {
554         caida = 0;
555       }
556       for (i = 0; i < 3; i++)  {
557         for (j = 0; j < 3; j++)  {
558           matrizIzquierdo[i][j] = fuerzaIzquierdo[i * 3 + j];
559           matrizDerecho[i][j] = fuerzaDerecho[i * 3 + j];
560         }
561       }
562       for ( i = 0; i < 3; i++) {
563         for ( j = 0; j < 3; j++) {
564           numeradorYi = numeradorYi + posicionYi[j] * matrizIzquierdo[j][i];
565           numeradorXi = numeradorXi + posicionX[j] * matrizIzquierdo[i][j];
566           denominadori = denominadori + matrizIzquierdo[i][j];
567
568           numeradorYd = numeradorYd + posicionYd[j] * matrizDerecho[j][i];
569           numeradorXd = numeradorXd + posicionX[j] * matrizDerecho[i][j];
570           denominadord = denominadord + matrizDerecho[i][j];
571         }
572       }
573       if (denominadori > 0.1) {
574         Xzmpi = numeradorXi / denominadori;
575         Yzmpi = numeradorYi / denominadori;
576       }
577       if (denominadord > 0.1) {
```

```
578       Xzmpd = numeradorXd / denominadord;
579       Yzmpd = numeradorYd / denominadord;
580     }
581     if (numeradorXd == 0 || numeradorYd == 0) {
582       Xzmpd = 0;
583       Yzmpd = 0;
584     }
585     if (numeradorXi == 0 || numeradorYi == 0) {
586       Xzmpi = 0;
587       Yzmpi = 0;
588     }
589     if ((denominadori + denominadord) * (denominadori + denominadord) > 0.01) {
590       Xzmp = (numeradorXi + numeradorXd + denominadord * DECX) / (denominadori
        + denominadord);
591       Yzmp = (numeradorYi + numeradorYd + denominadord * DECY) / (denominadori
        + denominadord);
592     }
593     //PROMEDIO DE ZMP
594     Xs[f] = Xzmp;
595     Ys[f] = Yzmp;
596     Xis[f] = Xzmpi;
597     Yis[f] = Yzmpi;
598     Xds[f] = Xzmpd;
599     Yds[f] = Yzmpd;
600     Xprom = 0;
601     Yprom = 0;
602     Xiprom = 0;
603     Yiprom = 0;
604     Xdprom = 0;
605     Ydprom = 0;
606     for ( i = 0; i < pf; i++) {
607       Xprom = Xprom + Xs[i];
608       Yprom = Yprom + Ys[i];
609       Xiprom = Xiprom + Xis[i];
610       Yiprom = Yiprom + Yis[i];
611       Xdprom = Xdprom + Xds[i];
612       Ydprom = Ydprom + Yds[i];
613     }
614     Xprom = Xprom / pf;
615     Yprom = Yprom / pf;
616     Xiprom = Xiprom / pf;
617     Yiprom = Yiprom / pf;
618     Xdprom = Xdprom / pf;
619     Ydprom = Ydprom / pf;
620     f++;
621     if (f == pf) {
622       f = 0;
623     }
624   }
625   void Controlador_ZMP()
626   {
627     if (estaux == 1) {
628       errpx = Xref - Xzmp;
629       errpy = Yref - Yzmp;
630     }
```

```
631     if (estaux == 0) {
632       errpx = Xref - Xzmpi;
633       errpy = Yref - Yzmpi;
634     }
635     if (estaux == 2) {
636       errpx = Xref - Xzmpd;
637       errpy = Yref - Yzmpd;
638     }
639     erriy = erriy + errpy;
640     errix = errix + errpx;
641 
642     errdx = errpx - errantx;
643     errdy = errpy - erranty;
644     errantx = errpx;
645     erranty = errpy;
646   }
647   void Pie(int n, float analogico)
648   { //LECTURA DE SENSORES PARA ATERRIZAJE DE PIE
649     if (analogico > 80) {
650       if (n == 1 || n == 2 || n == 3) {
651         PieI[2] = 1;
652       }
653       if (n == 4 || n == 5) {
654         PieI[1] = 1;
655       }
656       if (n == 6 || n == 7 || n == 8) {
657         PieI[0] = 1;
658       }
659       if (n == 9 || n == 10 || n == 11) {
660         PieD[2] = 1;
661       }
662       if (n == 12 || n == 13) {
663         PieD[1] = 1;
664       }
665       if (n == 14 || n == 15 || n == 16) {
666         PieD[0] = 1;
667       }
668     }
669   }
670   void anular() {
671     errpx = errpx / 2;
672     errpy = errpy / 2;
673     errix = 0;
674     erriy = 0;
675     errdy = 0;
676     errdx = 0;
677   }
678   void corpie() { //ATERRIZAJE DEL PIE
679     piecor = 0;
680     if (PieI[2] == 1) {
681       if (PieI[0] != 1 && PieI[1] != 1) {
682         switch (estaux) {
683           case 0:
684             errssi[11] -= 0.5;
685             break;
```

```
686           case 1:
687             if (paso == 1) {
688               errsd[11] -= 0.4;
689             }
690             errsd[11] -= 0.6;
691             break;
692           case 2:
693             errssd[11] -= 0.5;
694             break;
695           default:
696             break;
697         }
698         piecor = 1;
699       }
700     }
701     if (PieI[0] == 1) {
702       if (PieI[2] != 1 && PieI[1] != 1) {
703         switch (estaux) {
704           case 0:
705             errssi[11] += 0.5;
706             break;
707           case 1:
708             if (paso == 1) {
709               errsd[11] += 0.4;
710             }
711             errsd[11] += 0.5;
712             break;
713           case 2:
714             errssd[11] += 0.5;
715             break;
716           default:
717             break;
718         }
719         piecor = 2;
720       }
721     }
722     if (PieD[2] == 1) {
723       if (PieD[0] != 1 && PieD[1] != 1) {
724         switch (estaux) {
725           case 0:
726             errssi[10] -= 0.5;
727             break;
728           case 1:
729             if (paso == 1) {
730               errsd[10] -= 0.4;
731             }
732             errsd[10] -= 0.5;
733             break;
734           case 2:
735             errssd[10] -= 0.5;
736             break;
737           default:
738             break;
739         }
740         piecor = 3;
```

```
741       }
742     }
743     if (PieD[0] == 1) {
744       if (PieD[2] != 1 && PieD[1] != 1) {
745         switch (estaux) {
746           case 0:
747             errssi[10] += 0.5;
748             break;
749           case 1:
750             if (paso == 1) {
751               errsd[10] += 0.4;
752             }
753 
754             errsd[10] += 0.5;
755             break;
756           case 2:
757             errssd[10] += 0.6;
758             break;
759           default:
760             break;
761         }
762         piecor = 4;
763       }
764     }
765   }
```

B. Código en C # - Interfaz gráfica

```
1   using System;
2   using System.ComponentModel;
3   using System.IO;
4   using System.Windows.Forms;
5   using System.Diagnostics;
6   //PROGRAMA ENCARGADO DEL ENVÍO DE ÁNGULOS DE CAMINATA A UN ROBOT BÍPEDO, ASÍ
7   // COMO LA GRÁFICA DE SUS VARIABLES Y REGISTRO EN UN ARCHIVO DE TEXTO .TXT
8   //PROGRAMA CREADO POR EDUARDO ONTIVEROS GARCÍA
9   namespace bipedo_RxTx
10  {
11      public partial class Form1 : Form
12      {
13          StreamWriter SW; // OBJETO PARA ESCRITURA DE ARCHIVOS
14          StreamReader SR; //OBJETO PARA LECTURA DE ARCHIVOS
15          string Linea="", rutain="",rutaout="", cadena="";
                  //CADENAS AUXILIARES EN MANEJO DE ARCHIVOS
16          string[] texto;
                  //ARREGLO DONDE SE GUARDAN TODAS LOS ÁNGULOS DE CAMINATA
17           //VARIABLES AUXILIARES DE PROGRAMA
18          int dato = 0;
19          int indice = 0;
20          int largo = 0;
21          bool rebobinar = false;
22          int retraso = 5;
23          string sensores;
24          bool iterativo = false; //REPETICIÓN DE CAMINATA
25          bool parado = false;
26          byte ciclo = 1; // NÚMERO DE CICLOS DE CAMINATA REALIZADOS
27          public Form1()
28          {
29              InitializeComponent();
30              backgroundWorker1.WorkerReportsProgress = true; //HILOS
31              backgroundWorker1.WorkerSupportsCancellation = true;
32          }
33          private void NOP(long durationTicks) //DELAY (0.0000001 s)
34          {
35              var sw = Stopwatch.StartNew();
36              while (sw.ElapsedTicks < durationTicks)
37              {
38                  //gasta ciclos máquina
39              }
40          }
41
42          private void Form1_Load(object sender, EventArgs e)
43          {           //Se abre la aplicación
44              try //inicia búsqueda de puertos series y establece opciones de
                    velocidades en combobox
45              {
46                  string[] ports = System.IO.Ports.SerialPort.GetPortNames();
47                  string[] baud = new string[4] { "230400", "115200", "9600",
"14400" };
48                  comboBox1.Items.AddRange(ports);
49                  comboBox2.Items.AddRange(baud);
50                  comboBox1.Text = ports[0];
```

```
51                  comboBox2.Text = baud[0];
52                  DateTime dia = DateTime.Today; //establece la hora y día de
                 ejecución del programa para anexarlo al nombre del archivo generado
53                  string[] archivo1 = dia.ToString("d").Split('/');
54                  fecha.Text = archivo1[0] + "_" + archivo1[1] + "_" +
                    archivo1[2];
55                  string[] archivo2 = DateTime.Now.ToShortTimeString().Split('
                    ');
56                  Hora.Text = archivo2[0];
57              }
58              catch
59              { //Tratamiento de error
60                  MessageBox.Show("Error al abrir el puerto serie: ");
61              }
62          }
63
64          private void button4_Click(object sender, EventArgs e)
65          {
66              //botón de conexión al puerto Serie e inicio de pruebas
67              checkBox2.Enabled = false;
                  //deshabilita las opciones de configuración
68              checkBox3.Enabled = false;
69              if (checkBox3.Checked == true)
70              {
71                  //Configura la gráfica para medición de ZMP
72                  chart1.ChartAreas[0].AxisY.Minimum = -80;
73                  chart1.ChartAreas[0].AxisY.Maximum = 140;
74                  chart1.Series[0].Name = "Xzmp";
75                  chart1.Series[1].Name = "Yzmp";
76                  chart1.Series[2].Name = "Xref";
77                  chart1.Series[3].Name = "Yref";
78              }
79              else
80              {
81                  //Configura la gráfica para medición de ángulos pitch y roll
82                  chart1.ChartAreas[0].AxisY.Minimum = -50;
83                  chart1.ChartAreas[0].AxisY.Maximum = +50;
84                  chart1.Series[0].Name = "Pitch";
85                  chart1.Series[1].Name = "Roll";
86                  chart1.Series[2].Name = "Pref";
87                  chart1.Series[3].Name = "Rref";
88              }
89              rutain = textBox1.Text;
                  //Documento con ángulos de caminata en lazo abierto
90              DateTime dia = DateTime.Today;
91              string[] archivo1 = dia.ToString("d").Split('/');
92              fecha.Text = "Fecha: "+archivo1[0] + "/" + archivo1[1] + "/" +
                archivo1[2];
93              string[] archivo2 = DateTime.Now.ToShortTimeString().Split(' ');
94              Hora.Text ="";
95              string[] archivo3 = archivo2[0].Split(':');
96              rutaout = textBox3.Text + archivo1[0] + "_" + archivo1[1] + "_" +
                archivo1[2] +"_"+archivo3[0]+"_" +archivo3[1]+".txt";
                  //creación de ruta de archivo de texto con datos de la prueba a
                realizar con hora y fecha
```

```
97              if (!PuertoSerie.IsOpen)
98              {
99                  try
100                 {
101                     //Inicio de comunicación serial
102                     PuertoSerie.PortName = comboBox1.Text;
103                     PuertoSerie.BaudRate = Convert.ToInt32(comboBox2.Text);
104                     PuertoSerie.Open();
105                     button1.Enabled = true;
                        //habilita botones de envío de datos de caminata y pausa
106                     button2.Enabled = true;
107                     comboBox1.Enabled = false;
                        //deshabilita las configuraciones del puerto serie
108                     comboBox2.Enabled = false;
109                     button4.Enabled = false;
                        //deshabilita el botón de conectar
110                 }
111                 catch (Exception ex)
112                 {
                     //Tratamiento del error si no puede abrirse el puerto serial
113            MessageBox.Show("Error al abrir el puerto serie: " + ex.Message);
114                 }
115             }
116             SR = new StreamReader(rutain);
                  //Asignación de ruta de entrada al objeto de lectura de archivo
117             while (!SR.EndOfStream)
                  //lectura de todo el archivo para saber cuantas líneas tiene
118             {
119                 indice++;
120                 Linea = SR.ReadLine();
121             }
122             label2.Text = "Conectado"; //indicador de estado del programa
123             texto= new string[indice*13]; //indicador de cuantos datos tiene
        el archivo de texto (12 ángulos de servomotores más un indicador por
        línea)
124             SR = new StreamReader(rutain);
                  //Se vuelve a configurar el archivo de lectura al inicio
125             SW = new StreamWriter(rutaout, true);
                  //si el archivo donde se escribirá no existe lo crea y si
        existe, empieza a escribir al final
126             while (!SR.EndOfStream) //método para guardar todos los datos del
        archivo de texto en un arreglo de 14040 datos
127             {
128                 Linea = SR.ReadLine();
129                 string[] temporal = Linea.Split('\t');
130                 for (int k = 0; k < 12; k++) //12 ángulos
131                 {
132                     texto[largo * 13 + k] = temporal[k];
                  //texto [] arreglo de todos los ángulos
133                   }
134                   texto[largo * 13 + 12] = "300";
                  //Dato que indica el fin de la línea
135                   largo++; //avance de renglón
136             }
137
```

```
138             control.Enabled = true;
            //activación de casilla que indica el sistema de control
139         }
140
141         private void button1_Click(object sender, EventArgs e)
            //Envío de datos de caminata
142         {
143             label12.Text = "Ciclo: " + ciclo++;
                  //contador de ciclos de caminata realizados
144             foreach (var series in chart1.Series)
                  //graficador de variables de control
145             {
146                 series.Points.Clear();
147             }
148             retraso = Convert.ToInt32(textBox2.Text);
                  //establecimiento de retraso entre datos a enviar
149             label2.Text = "Enviando"; //estado
150             if (backgroundWorker1.IsBusy != true) //inicio de hilos
151             {
152                 // empieza la tarea simultánea.
153                 backgroundWorker1.RunWorkerAsync();
154             }
155             rebobinar = true; //se permite que se rebobinen los datos
156             button1.Text = "Rebobinar";
            //cambia función de botón 1 para rebobinar los datos de caminata
157             dato = 0;
158         }
159
160         private void button2_Click(object sender, EventArgs e)
            //botón de pausado de caminata y reanudación
161         {
162             if (parado == false)
163             {
164                 button2.Text = "Reanudar";
165                 parado = true; //pausa activada
166             }
167             else
168             {
169                 button2.Text = "Pausar";
170                 parado = false;
171             }
172         }
173
174         private void button3_Click(object sender, EventArgs e)
            //botón de paro de caminata
175             {
176             if (backgroundWorker1.WorkerSupportsCancellation == true)
177             {
178                 // Termina el hilo
179                 backgroundWorker1.CancelAsync();
180             }
181             try{
182                 //Cierra los archivos y el puerto serie
183                 NOP(retraso * 1000);
```

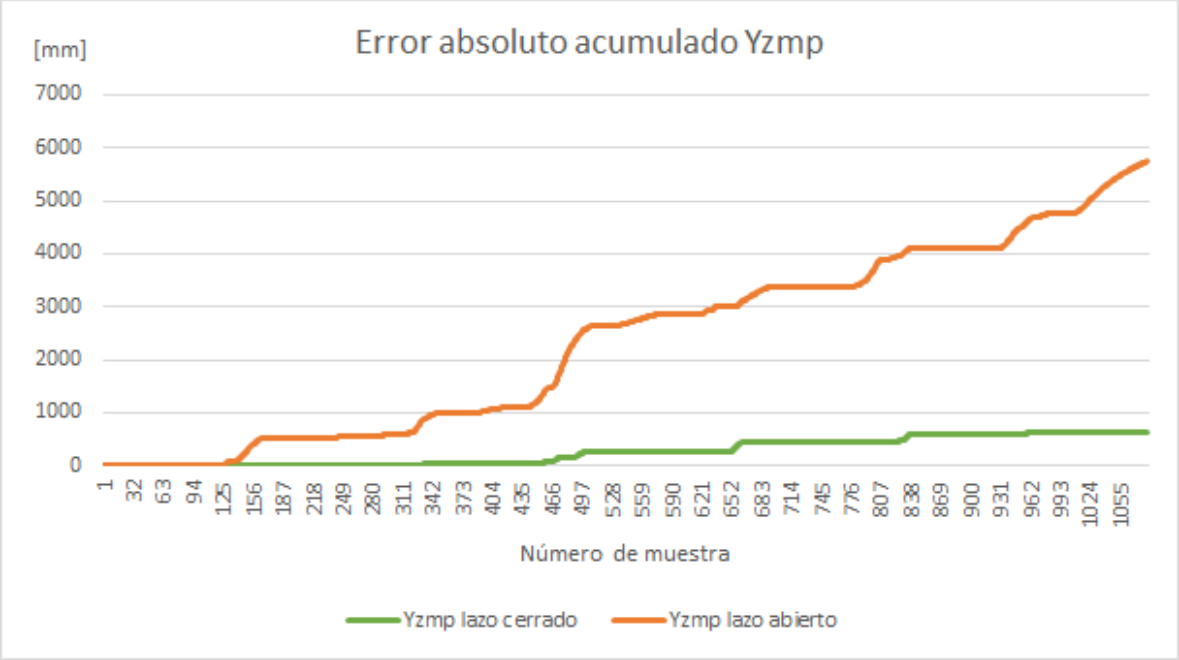
```
184                 PuertoSerie.Write("310");
                  //manda señal al robot que indica que la caminata se canceló
185                 NOP(retraso * 1000);
186                 PuertoSerie.Close();
187                 SR.Close();
188                 SW.Close();
189                 }
190             catch
191             {
192
193             }
194             //try{PuertoSerie2.Close();}catch{}
195             this.Close();
196         }
197
198         private void checkBox2_CheckedChanged(object sender, EventArgs e)
              //selección de modo ZMP
199         {
200             if (checkBox2.Checked == true)
201             {
202                 checkBox3.Checked = false;
203             }
204
205         }
206
207         private void checkBox3_CheckedChanged(object sender, EventArgs e)
              //Selección de modo IMU
208         {
209             if (checkBox3.Checked == true)
210             {
211                 checkBox2.Checked = false;
212             }
213         }
214
215         private void checkBox1_CheckedChanged(object sender, EventArgs e)
            //Activación de repetición de ciclos de caminata
216         {
217             if (checkBox1.Checked == true)
218             {
219                 iterativo = true;
220             }
221             else{
222                 iterativo=false; //desactivación
223             }
224         }
225
226         private void control_CheckedChanged(object sender, EventArgs e)
227         { //Activación de sistema de control
228             if (PuertoSerie.IsOpen)
229             {
230                 PuertoSerie.Write("315");
231             }
232         }
233
234         private void toolStripLabel1_Click(object sender, EventArgs e)
```

```
235         {
236             MessageBox.Show("Conecte el robot, seleccione su archivo de salida
        y el archivo con la trayectoria previamente cargada, presione conectar
        y enviar");
237         }
238
239         private void toolStripLabel2_Click(object sender, EventArgs e)
240         {
241             MessageBox.Show("Eduardo Ontiveros García, 2018. Derechos
        Reservados ");
242         }
243
244         //*****SECCIÓN HILOS*****
245         private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs
e)
246         {
247             BackgroundWorker worker = sender as BackgroundWorker;
                //DEFINICIÓN DE HILO PRINCIPAL
248             dato = 0;
249             for (int j = 0; j < texto.Length; j++)
        //recorrido de todo el arreglo de ángulos
250             {
251                 if (rebobinar == true)
252                 {
253                     j = 0;
254                     rebobinar = false;
                        //se rebobinan los datos al reiniciar el recorrido con j=0
255                     NOP(retraso * 10);
256                     PuertoSerie.Write("310");
257                     NOP(retraso * 10);
258                 }
259                 if (worker.CancellationPending == true)
              //interrupción del hilo
260                 {
261                     e.Cancel = true;
262                     break;
263                 }
264                 else
265                 {
266                     if (parado == false) //Si no esta en pausa
267                     {
268                         cadena = texto[j]; //toma el ángulo de motor a enviar
269                         PuertoSerie.Write(cadena); //y lo envía
270                         NOP(retraso * 10); //pausa de envío
271                         worker.ReportProgress(1);
272                         dato++; //aumenta el numero mostrado en el label
273
274                     }
275                     else
276                     { //en pausa no hace nada
277
278                         NOP(6000);
279                         worker.ReportProgress(1);//activa el hilo secundario
280                         j--;
    //contrarresta el aumento del for para continuar en el mismo ángulo
```

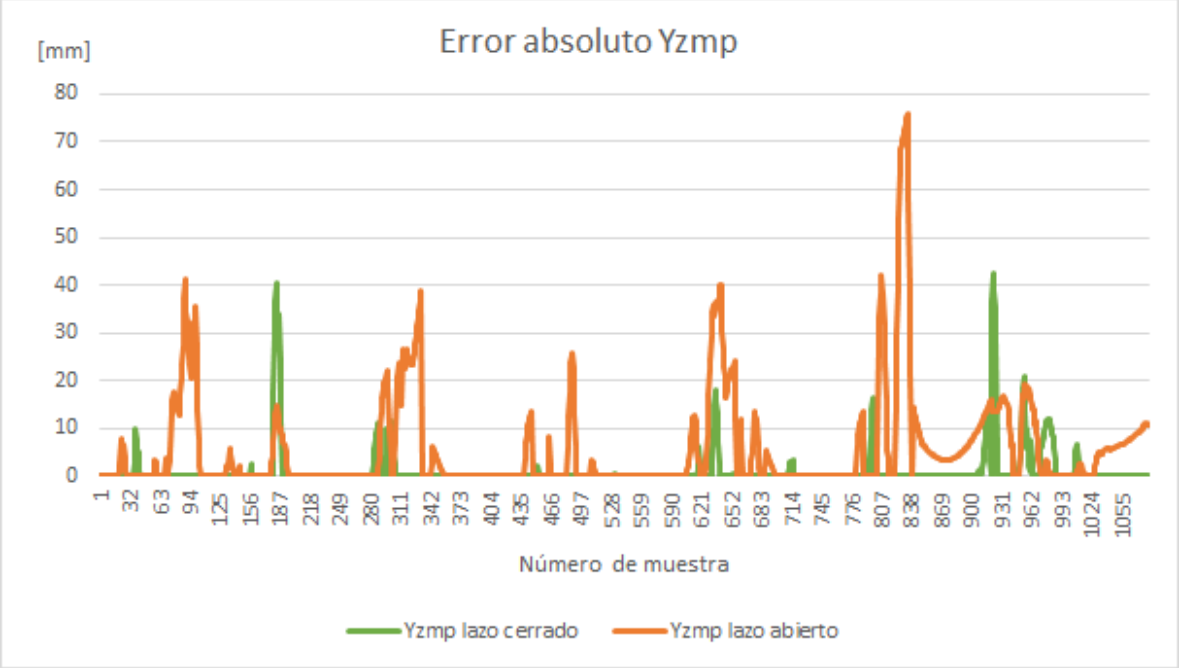
```
281                     }
282                 }
283             }
284         }
285 
286         private void backgroundWorker1_ProgressChanged(object sender,
ProgressChangedEventArgs e)
287         { //hilo secundario
288             try
289             {
290 
291                 sensores = PuertoSerie.ReadExisting();
                  //Lee los datos provenientes del robot
292                 if (sensores != "") //si hay datos que leer
293                 {
294                     string[] strgrafica = sensores.Split('\t'); //los separa
                  por las tabulaciones para obtener los cuatro datos
295                     SW.Write(sensores); //lo escribe en el documento de texto
296                     if (strgrafica.Length == 4)
                      //revisa que sean cuatro y los grafica
297                     {
298                         chart1.Series[0].Points.AddXY(dato, strgrafica[0]);
299                         chart1.Series[1].Points.AddXY(dato, strgrafica[1]);
300                         chart1.Series[2].Points.AddXY(dato, strgrafica[2]);
301                         chart1.Series[3].Points.AddXY(dato, strgrafica[3]);
302                         if (strgrafica[0] == "231" || strgrafica[1] == "231"
                    || strgrafica[2] == "231" || strgrafica[3] == "231")
303                         {//conjunto de datos que indican una caída
304 
305                             if (backgroundWorker1.WorkerSupportsCancellation
                              == true)
306                             {
307 
308                                 backgroundWorker1.CancelAsync();
                               //termina el hilo principal
309                             }
310                             label2.Text = "CAIDA"; //Estado del robot
311 
312                         }
313                         if (dato > 1500)
314                         {
315 
316                             chart1.ChartAreas[0].AxisX.Minimum = 0; //permite
                        que la gráfica pueda mostrar el ciclo de caminata completa
                        cuando se sobrepasan los 1500 datos
317                         }
318                     }
319 
320                 }
321             }
322 
323             catch (Exception ex)
324             { //tratamiento de error
325                 MessageBox.Show("Esto pasó: " + ex.Message);
326             }
```

```
327
328             //Se actualizan los indicadores
329             label1.Text = dato / 13 + "";//número de líneas enviadas
330             label8.Text = cadena; //Dato enviado
331             label6.Text = dato + ""; //número de datos enviados
332         }
333
334         private void backgroundWorker1_RunWorkerCompleted(object sender,
            RunWorkerCompletedEventArgs e)
335         {
336             if (e.Cancelled == true) //fin de caminata
337             {
338                 if (label2.Text != "CAIDA")
339                 {
340                     label2.Text = "Cancelado";
341                 }
342             }
343             else if (e.Error != null)
344             { //si existió un error lo muestra
345                 label2.Text = "Error: " + e.Error.Message;
346             }
347             else
348             {
349                 //si la caminata se realizó correctamente
350                 label2.Text = "Terminado!"; //indicador de que terminó
351                 //establece las condiciones iniciales del programa para
                    iniciar una nueva prueba
352                 button1.Text = "Enviar"; //
353                 chart1.ChartAreas[0].AxisX.Minimum = 0;
354                 NOP(retraso * 10);
355                 PuertoSerie.Write("310");
356                 NOP(retraso * 10);
357                 if (iterativo)
                  //si esta activado el modo de repetición de caminata
358                 {
359                     label12.Text = "Ciclo: " + ciclo++;
                        //aumenta en uno los ciclos realizados
360                     foreach (var series in chart1.Series)
361                     {
362                         series.Points.Clear();
363                     }
364
365                     rebobinar = true;
366                     if (backgroundWorker1.IsBusy != true)
                        //reinicia el ciclo de marcha
367                     {
368                         // Activa el hilo principal
369                         backgroundWorker1.RunWorkerAsync();
370                     }
371                 }
372             }
373         }
374
375     }
376 }
```

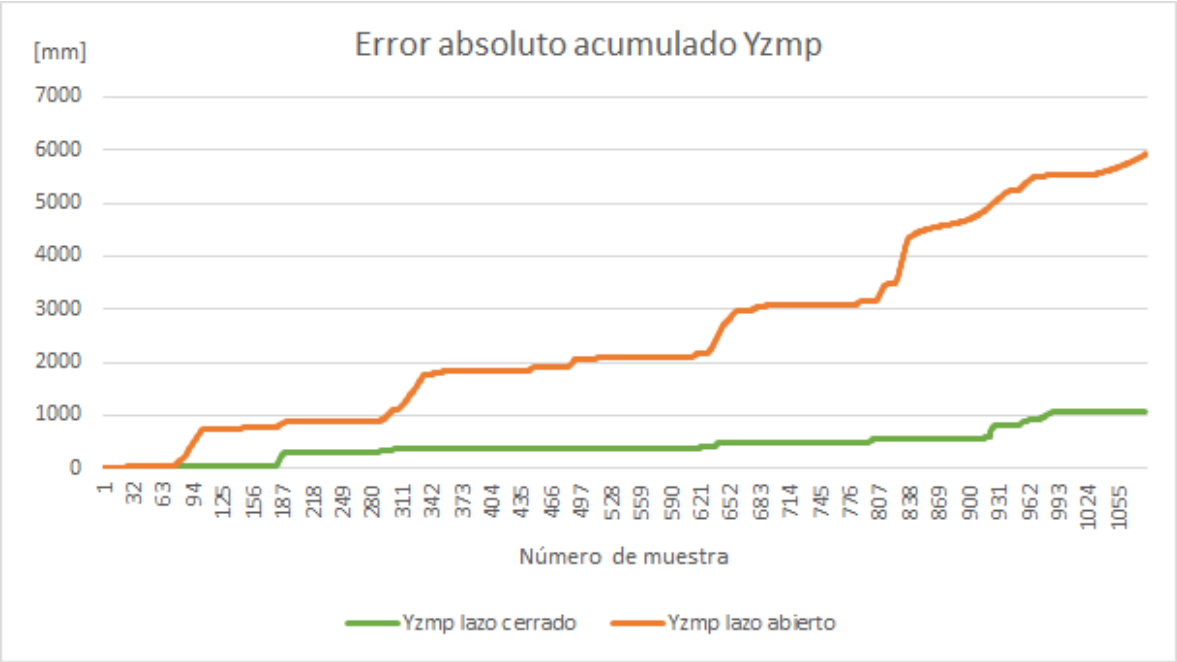
C. Gráficas complementarias



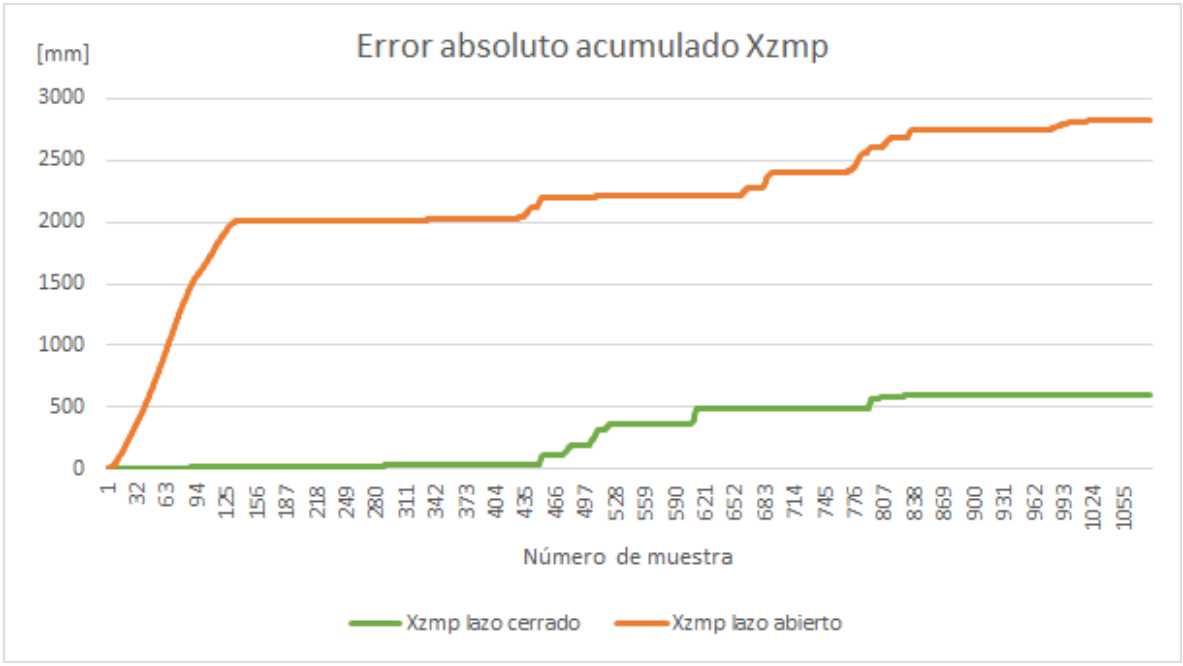
A. Error absoluto acumulado en Yzmp durante caminata en superficie inclinada menos seis grados en dirección pitch



B. Error absoluto en Yzmp durante caminata en superficie inclinada seis grados en dirección pitch



C. Error absoluto acumulado en Yzmp durante caminata en superficie inclinada seis grados en dirección pitch



D. Error absoluto acumulado en Xzmp durante caminata en superficie inclinada menos seis grados en dirección roll