Simulación e implementación de locomoción tipo bípeda utilizando sistema robótico modular MECABOT

Jhonatan Stiven Olarte Vega

Tutor:

Ing. Ricardo Andrés Castillo Estepa, PhD.

Universidad Militar Nueva Granada

Facultad de Ingeniería

Programa de ingeniería en Mecatrónica

Bogotá

2018

Simulación e implementación de locomoción tipo bípeda utilizando sistema robótico modular MECABOT

Jhonatan Stiven Olarte Vega

Tutor:

Ing. Ricardo Andrés Castillo Estepa, PhD.

Universidad Militar Nueva Granada

Facultad de Ingeniería

Programa de ingeniería en Mecatrónica

Bogotá

2018

Bogotá (-, -, -)

Nota de Aceptación

_____

_____

_____

_____

_____

_____

_____

Firma del presidente del Jurado

_____

Firma de Jurado

Agradecimientos

A mis padres y mi hermana por la paciencia y comprensión que han tenido, siempre apoyándome para completar esta carrera tan bonita, a mi novia por su compañía y sus motivaciones incondicionales, a mis amigos más cercanos por hacerme una mejor persona.

A Ricardo Castillo PhD por sus conocimientos, por su paciencia y por su guía durante este tiempo de realización de esta tesis. A la ingeniera Vanessa Cruz por sus conocimientos, experiencias e ideas.

A la Universidad Militar Nueva Granada, al programa de Ingeniería en Mecatrónica, a los docentes y al personal de los laboratorios por la formación integral brindada durante el transcurso del pregrado.

Contenido

Índice de figuras

Índice de tablas

Índice de Anexos

Glosario

A

Amplitud: Distancia entre el punto más alejado de una onda sinusoidal a su punto de equilibrio.

B

Bias: Diferencia de fase entre ondas sinusoidales o de una onda sinusoidal a partir del tiempo cero.

Bioinspirado: Basado en la naturaleza.

Bípedo: Locomoción en dos patas.

C

Cinemática: Estudio del movimiento sin tener en cuenta sus causas (fuerzas, pares) .

D

DOF: Abreviación de grados de libertad.

G

Generadores sinusoidales: Central Pattern Generators simplificados.

Grados de libertad: Movimiento rotacional o traslacional en el plano tridimensional permitido por una articulación.

M

MECABOT: Robot modular creado por Davinci (UMNG)

Módulo: Estructuras simples generalmente provistas de sistemas independientes (microcontorlador, actuadores, etc.) .

O

Offset: Desplazamiento, positivo o negativo, del punto de equilibrio de una onda sinusoidal.

S

Semi módulo: En el Mecabot, se refiere a un pivote unido a un cuerpo. Dos semi módulos conforman un módulo.

W

Webots: Simulador robótico.

Resumen

Este trabajo se enfoca en analizar los parámetros de la caminata humana en una perspectiva 2D, para realizar un modelo mecánico con el sistema robótico modular Mecabot de la Universidad Militar Nueva Granada. Es la primera arquitectura bípeda de 6 desarrolladas para el proyecto Mecabot (rueda, serpiente, oruga, hexápodo, cuadrúpeda y bípeda) , está arquitectura descansa sobre una estructura diseñada con el objetivo de mantener el equilibrio del robot, se utilizaron generadores sinusoidales con la función de evitar picos de corriente en los motores. Para validar los controladores primero se hizo la simulación en el software Webots®, la selección del software se realizó debido a sus características del entorno de simulación, como torque y velocidad de cada uno de los motores simulados.

Los resultados obtenidos miden la velocidad de la arquitectura en función de la variación de la frecuencia de movimiento, luego se realizó la respectiva implementación de la arquitectura bípeda realizando las pruebas en diferentes superficies, esto debido porque el sistema robótico modular Mecabot, va a ser un robot de exploración y rescate, el cual se acoplará en la arquitectura más óptima para el terreno en donde se encuentre.

Abstract

This work focuses on analyzing the parameters of the human walk in a 2D perspective, to make a mechanical model with the modular robotic system Mecabot of the Military University Nueva Granada. It is the first bipedal architecture of 6 developed for the Mecabot project (wheel, snake, caterpillar, hexapod, quadrupeda and bipedal) , this architecture rests on a structure designed to maintain the balance of the robot, sinusoidal generators were used to prevent power spikes in the engines. To validate the drivers first the simulation was done in the software Webots®, the selection of the software was made due to its characteristics of the simulation environment, such as torque and speed of each of the simulated engines.

The results obtained measure the speed of the architecture in function of the variation of the frequency of movement, then the respective implementation of the bipedal architecture was carried out performing the tests on different surfaces, This is because the modular robotic system Mecabot is going to be an exploration and rescue robot, which will be coupled in the most optimal architecture for the terrain where it is located.

1. Introducción

Nuestra sociedad vive en constantes transformaciones, por los avances tecnológicos gracias a las investigaciones que se llevan a cabo día a día, para dar solución a un problema o simplemente mejorar la calidad de vida. La sociedad cada día globalizada, avanza imparable hacia lo que es un futuro incierto, debido a los grandes avances científicos, tecnológicos y de muchos otros ámbitos que se realizan.

Este trabajo se fundamentó en la aplicación de la robótica, la cual es una ciencia o rama de la tecnología, que estudia el diseño y construcción de máquinas capaces de desempeñar tareas realizadas por el ser humano o que requieren del uso de inteligencia.

Las ciencias y tecnologías de las que deriva podrían ser: el álgebra, los autómatas programables, las máquinas de estados, la mecánica o la informática (Macchiavello, 2008).

La robótica es fundamental en la construcción de “artefactos”, que tratan de materializar el deseo humano de crear seres semejantes a nosotros que nos facilite el trabajo, en las diferentes áreas como; construcción, oficina, hogar, agricultura, seguridad, medicina y demás actividades económicas, turísticas y de esparcimiento del ser humano. El ingeniero español Leonardo Torres Quevedo (que construyó el primer mando a distancia para su torpedo automóvil mediante telegrafía sin hilodrecista automático, el primer transbordador aéreo y otros muchos ingenios) acuñó el término “automática” en relación con la teoría de la automatización de tareas tradicionalmente asociadas a los seres humanos.

Como podemos apreciar recientemente en el evento Consumers Electronic Show de Las Vegas (CES 2019), con más de 3 mil empresas con nuevos juguetes para los amantes de la tecnología, la innovación y las sorpresas como son:

En la movilidad automotriz, las cuatro ruedas y un motor están mandadas a recoger en el futuro. Mercedes Benz, Kia, Hyundai y BMW presentaron sus vehículos autónomos, libres de gasolina y de errores. Por un lado, BMW hizo una alianza con Intel para dotar con Inteligencia Artificial a sus máquinas para que sean capaces de decidir por sí solas. Entre tanto Kia y Mercedes presentaron sus carros autónomos que, aunque están diseñados para una persona, prometen ir a más de 200 kilómetros por hora, por último, Hyundai se llevó todas las miradas cuando presentó su vehículo autónomo de

emergencia bautizado ‘Elevate’. Este auto puede modificar su chasis en brazos robóticos para escalar montañas y atravesar caminos que serían imposibles para una camioneta 4X4.

En lo científico sus integrantes acordaron de renovar las pruebas de orina para detectar drogas, embarazo y análisis de la glucosa. En vez de ir a un centro especializado a esperar durante una hora por sus exámenes, ya podrá comprar una ‘TestCard’ de cuatro dólares que analizará su orina y luego podrá escanear con su celular para conocer los resultados en tan solo minutos.

En el hogar, el teatro en casa gozaba de buena vida en los hogares del mundo gracias a que las pantallas crecían y sus salidas de audio se quedaban en el olvido. Ahora, Samsung presentó a ‘The Wall’, un televisor de 219 pulgadas -de ahí su nombre- que cuenta con la nueva tecnología de audio inalámbrico WiSA (Altavoz y Audio Inalámbrico), esta tecnología le brinda a los usuarios un sonido envolvente y de alta calidad a través de 5 altavoces de Klipsch. Lo mejor de todo es que no necesitan cables o receptores, ya que cuentan con un chip que es compatible con cualquier salida de sonido. Lo único regular que se puede destacar de esta tecnología es que los usuarios tendrán que encontrar una toma eléctrica para conectar estos WiSA.

La gastronomía llegó al evento más importante de tecnología por primera vez en el pabellón FoodTech, en el que miles de inversores encontraron máquinas interesantes como el Selffee, una impresora portátil para imprimir su cara en galletas. Algo similar ocurre con Ripples que pinta su cara en la espuma de su latte. Bearrobotics es una compañía encargada de mantener su cerveza fría sin importar a donde vaya. Algunos usuarios en redes sociales la bautizaron como el R2D2 de nuestro tiempo, Con la moda de volver a lo artesanal y lo ancestral llegó al CES 2019. Bartesian. Esta máquina le permite hacer su propia cerveza de acuerdo a su gusto. Como si se tratara de una máquina de café, esta tecnología cuenta con ‘pods’ de sabores para su cerveza. Tiene un costo de 229 dólares.

Con todos estos avances la tecnología busca mejorar la calidad de las personas en ciertos campos específicos, sin dejar de lado el cuidado por el medio ambiente, a su vez obliga a la humanidad en capacitarse para buscar nuevas formas de generar ingresos económicos, ya que la tendencia de las empresas es sistematizar la mano de obra, lo que genera mayores ganancias, pero mayor inversión.

1.1. Planteamiento del problema

Una problemática actual son las limitaciones que presentan los robots, ya que estos solo cumplen tareas limitadas y se necesitan varios para desempañar diferentes tareas, la robótica modular entra en este aspecto al ofrecer un número muy variado de aplicaciones, gracias a su cambio de arquitectura y adaptabilidad a diferentes entornos, una de sus aplicaciones se puede ver como el ingreso en espacios hostiles para el ser humano para desempeñar tareas de rescate.

La Universidad Militar Nueva Granada con su grupo de investigación DAVICNI han desarrollado un robot modular llamado MECABOT, que ha pasado desde simulación a implementación en diferentes versiones, la última versión de este robot el MECABOT 5.0, este robot tiene la ventaja de contar ya con diferentes arquitecturas de locomoción como son hexápodo, serpiente, rueda, oruga, con el fin de ampliar este tipo de conocimientos de desarrollo enfocados al robot modular MECABOT ¿Se puede simular e implementar la arquitectura de locomoción bípeda con el sistema robótico modular MECABOT?

1.2. Objetivo general

Simular e implementar el tipo de locomoción bípeda en el sistema robótico modular MECABOT

1.3. Objetivos específicos

- Diseñar la arquitectura tipo bípeda haciendo uso del software de simulación 3D para robótica Webots y los módulos del sistema robótico modular MECABOT.

- Ensamblar la arquitectura tipo bípeda en un entorno virtual en el software de simulación 3D para robótica Webots.
- Elaborar los movimientos básicos de la locomoción tipo bípeda, en un entorno virtual en el software de simulación 3D para robótica Webots

- Implementar la locomoción tipo bípeda utilizando los módulos físicos del sistema robótico modular MECABOT.

- Realizar pruebas de funcionamiento y evaluar los resultados obtenidos.

1.4. Delimitación

La siguiente propuesta pretende enfocarse únicamente a la simulación e implementación del tipo de locomoción bípeda con un sistema de robótica modular MECABOT 5.0 de la Universidad Militar Nueva Granada, luego se procederá a los análisis del funcionamiento de la locomoción tipo bípeda con el robot modular, estos resultados servirán para hacer una realimentación para futuros proyectos con el MECABOT.

Se resalta que las modificaciones al diseño mecánico y electrónico de los módulos individuales MECABOT, empleados en el ensamble del tipo de locomoción bípeda no son objetivos de esta propuesta de trabajo de grado.

1.5. Justificación

La robótica tiene diversos campos de acción, uno de los campos más explotados a lo largo de la historia es la robótica de manipuladores y móvil de estructura fija, los cuales cumplen funciones específicas de manera efectiva, sin embargo, si se desea realizar una función diferente, se debe adaptar estos robots para cumplir esta nueva función, pero en algunos casos no es posible adaptarlos o los costos de estas adaptaciones son muy elevados.

La robótica modular surge como una rama de la robótica que permite dar una solución económica y funcional al problema mencionado anteriormente, los robots modulares son sistemas robustos, económicos, adaptables y programables para realizas distintas tareas de acuerdo a las necesidades del usuario, adicionalmente el costo mantenimiento o reparación de los robots modulares es menor respecto a otros tipos, ya que en algunos casos solo es necesario reemplazar parcial o totalmente uno o más módulos, esto lo hace llamativos para los desarrollos en comunidades universitarias, industriales y de software-hardware libre

El grupo de investigación DAVINCI de la Universidad Militar Nueva Granada, ha diseñado, simulado e implementado el MECABOT, un sistema robótico modular multipropósito, se han implementado diversas arquitecturas en distintas versiones, con el fin de lograr un proceso de investigación y mejoramiento continuo de este sistema. A futuro se planea utilizar el sistema robótico modular MECABOT en operaciones de rescate y búsqueda, en ambientes donde por diversas condiciones no pueda acceder fácilmente o presente riesgo para el personal humano.

1.6.Metodología

Para el estudio de la arquitectura bípeda de METCABOT 5., nos remitiremos a la definición de la robótica, utilizando una investigación descriptiva, que detallaremos a continuación:

El propósito del investigador es describir situaciones y eventos. Esto es, decir cómo es y se manifiesta determinado fenómeno. Los estudios descriptivos buscan especificar las propiedades importantes de personas, grupos, comunidades o cualquier otro fenómeno que sea sometido a análisis miden o evalúan diversos aspectos, dimensiones o componentes del fenómeno o fenómenos a investigar. Desde el punto de vista científico, describir es medir. Esto es, en un estudio descriptivo se selecciona una serie de cuestiones y se mide cada una de ellas independientemente, para así y valga la redundancia-- describir lo que se investiga.

Revisaremos la filología que trata la robótica, para poder pasar al diseño de la arquitectura bípeda, para poder obtener las recomendaciones para futuros trabajos relacionados con el tema.

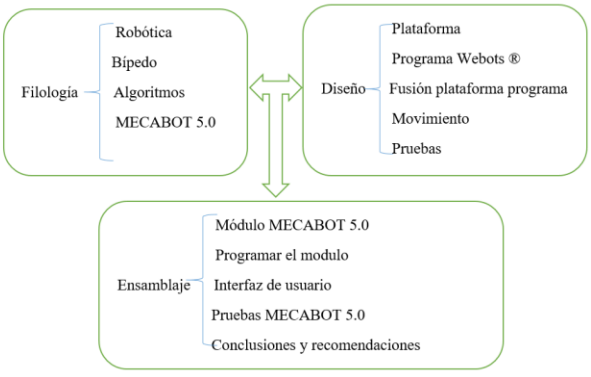


Figura 1-1 Metodología

Tomado de [7]

2. Marco Referencial

2.1.Robótica

La robótica es la rama de la tecnología, que se enfoca en el desarrollo de diseños e implementación de diferentes mecanismos autónomos, con el fin de ayudar a la ser humano en diferentes tareas con mayor facilidad, logrando una mayor efectividad en el manejo de tiempo y recursos para dichas tareas [1].

El termino robot fue usado por primera vez por el autor checoslovaco Karel Capek en su obra de teatro R.U.R. (Robots Universales de Rossum) en 1921. Esta termino proviene de la palabra checa robota la cual significa “trabajo forzado”[2].

Los robots al principio del siglo XX eran observados con miedo, ya que era muy popular la idea de que se revelaran y terminaran asesinado al ser humano, esta idea se fortaleció gracias a la ciencia ficción, la cual tiene varios ejemplos en que las maquinas inteligentes se revelan y tratan de asesinar al ser humano, aun en estos años del siglo XXI, un número de personas creen que puede llegar a ser realidad. Con la llegada de la segunda guerra mundial se requirieron muchas tareas automatizadas, esto llevo al desarrollo en masa de los robots, en especial en la industria automotriz la cual los utilizaba para tareas específicas como, soldadura, pintura, trasporte etc. [2]

Actualmente con el avance en el área informática y electrónica, existen robots más diversos enfocados para realizar actividades más complicadas, peligrosas y desagradables para el ser humano actividades como, exploración, manipulación de sustancias de alto riego, rescate, alturas etc. [3].

2.2.Robot Móvil

Los robots móviles se describen como maquinas con un sistema de locomoción para desplazarse fácilmente por su zona de trabajo, estos surgen a la necesidad de tener el mismo robot en diferentes zonas, con el tiempo los robots móviles comienzan a tener diferentes tareas mas complicadas como, rescate, exploración, búsqueda etc.

Estos robots cuentan con diferentes niveles de autonomía, este nivel está determinado por la capacidad del robot para percibir el ambiente de trabajo mediante sensores y poder modificar su

comportamiento en consecuencia con los obstáculos encontrados, existen una variedad de robots móviles entre las cuales entran los robots con ruedas con todas sus variaciones y los robots con patas. [4]

2.3.Robots con patas

En la actualidad, el diseño de sistemas de locomoción de robots caminantes se ha derivado del estudio de sistemas biológicos, especialmente de animales terrestres comunes, en los cuales se puede observar un sistema de locomoción basado en un conjunto de eslabones y articulaciones denominadas patas.[5]

La mayoría de robots caminantes están destinados a la exploración de terrenos irregulares inaccesibles para el ser humano. Entre las principales aplicaciones de estas máquinas están la detección de minas personales, transportación de equipos y herramientas de construcción, trabajos de forestación, limpieza de fachadas, entre otros.[5]

2.4.Robots Bípedos

Los robots con patas presentan una desventaja que es la velocidad que alcanzan a comparación de los robots con ruedas, pero ganan versatilidad y pueden desplazarse en terrenos irregulares, una ventaja que tienen los robots bípedos es que, al contar con una caminata humana, hace que la interacción de los humanos con estos robots sea mejor, al ser semejantes.[6]

2.4.1. YABIRO

Es un robot que tiene 14 grados de libertad de los cuales 6 corresponde a cada una de las piernas y 2 para el torso, para el movimiento presenta un generador de patrones de movimiento, su objetivo es tener un sistema robusto tanto en el referente mecánico, como en el electrónico, los actuadores incorporados en el YABIRO son servomotores.[7]

Figura 2-1 Plataforma del robot Yabiro

Tomado de [7]

2.4.2. ASIMO

ASIMO (Advanced Step in Innovative Mobility) es un robot humanoide, desarrollado por la empresa japonesa HONDA en el año 2000 y en el año 2011 se presentó su última versión, ASIMO presenta una gran variedad de sensores para evitar obstáculos, también tiene un sistema de filtrado para reconocer la voz de una persona en un entorno ruidoso.[8]

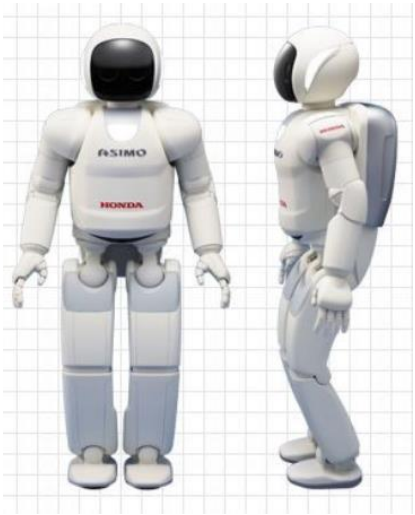


Figura 2-2 Robot Honda ASIMO

Tomado de:[8]

2.4.3. SONY QRIO

QRIO es un prototipo desarrollado por la empresa japonesa SONY lanzado en el 2003, fue el seguimiento del desarrollo del SONY AIBO, no se vendió ninguna unidad porque no había superado varias fases de desarrollo, al final el proyecto fue cancelado.[9]

Figura 2-3 SONY QRIO

Tomado de:[9]

2.4.4. Humanoid Robot HRP-2

Es un robot bípedo HRP fue desarrollado por el ministerio de economía, comercio e industria japonés, en la versión 2 se incorpora muchas más funciones, ya que puede hacer frente a superficies irregulares, caminar por pistas estrechos y poder levantarse después de una caída. [10]

Figura 2-4 Humanoid Robot HRP-2

Tomado de:[10]

2.5.Robótica Modular

El termino robótica modula ser refiere a una familia de sistemas robóticos compuestos por pequeñas unidades interconectadas llamados módulos, que se unen a través de acoples. Los módulos son diseñados con arquitecturas aparentemente simples, a pesar de ser independientes, están dotados se sistemas para interactuar entre ellos y acoplarse para formar una arquitectura más especializada para tareas específicas.[11]

La modularidad brinda al sistema robótico ventajas funcionales y económicas respecto a los robots convencionales. Un robot modular es capaz de adaptarse a diferentes situaciones, cambiando el posicionamiento de sus módulos para construir una arquitectura diferente, también se pueden incluir módulos especiales como, por ejemplo, pinzas, sensores, baterías y otros. Gracias a que son módulos, cuando uno presente una falla, la solución es el cambio de ese modulo economizando el mantenimiento a comparación de un robot especializado.[11]

2.6.Mecabot

En el año 2013 la universidad Militar Nueva Granada comenzó su investigación en el área de robots modulares, dando como resultado el desarrollo de los módulos Mecabot, en su primera generación contaba con cinco servomotores controlados por un PIC24HJ12HJ12GP201 de Microchip, Driver Micro Maestro 6 de Pololu, comunicación RF Transceptor +0Bm, sensores de proximidad y cuatro baterías de litio.[12] [13] [14]

Figura 2-5 Diferencias entre el Mecabot 1.0 y 2.0

Módulo del Mecabot 2.0 (negro) en comparación al módulo del Mecabot 1.0 (rojo).Tomado de:[15]

El Mecabot 1.0 tenía deferentes grados de libertad, como el movimiento rotacional a los extremos y contaba con un acople mecánico, pero los motores utilizados no poseían las características necesarias para efectuar movimientos en este modelo, por sus dimensiones y peso, por lo tanto, se desarrollaron diferentes variantes de este, dando pie al Mecabot 3.0, Mecabot 4.0. y Mecabot 3.0 [12]

2.6.1. Arquitectura Serpiente y Oruga

La arquitectura tipo rueda se implementó con los módulos modulares Mecabot 3.0 fueron las primeras arquitecturas implementadas, utilizaban generadores sinusoidales para el control del movimiento y se llevó a la practica con movimientos de avance. [15]

Figura 2-6 Arquitectura tipo oruga

Arquitectura oruga con el Mecabot 3.0 Tomado de:[15]

2.6.2. Arquitectura Hexápodo

La arquitectura tipo rueda se implementó con los módulos modulares Mecabot 4.0, esta es una de las arquitecturas propuestas para el programa del Mecabot, esta arquitectura tiene como

ventaja que fue la primera arquitectura con patas del programa, como resultado lo hace muy versátil para terrenos irregulares, a comparación de las arquitecturas anterior a ellas.[12]

Figura 2-7 Arquitectura tipo hexapodo

Arquitectura hexápodo con el Mecabot 4.0 Tomado de:[12]

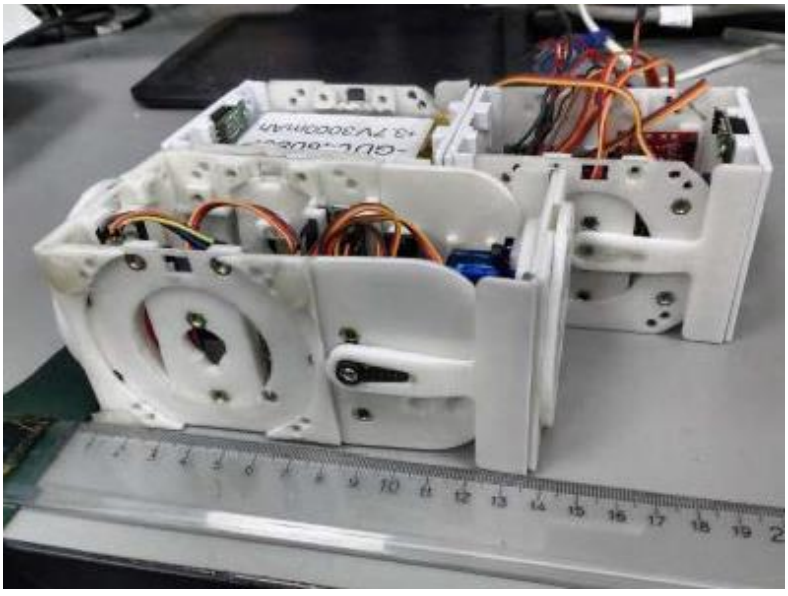


Figura 2-8 Diferencias entre el Mecabot 3.0 y 4.0

Módulo del Mecabot 3.0 (frente) en comparación al semi-módulo del Mecabot 4.0 (atras). Tomado de:[14]

2.6.3. Arquitectura Rueda

La arquitectura tipo rueda se implementó con los módulos modulares Mecabot 4.0 la cual presenta movimientos hacia adelante y hacia atrás, también conto con comunicación con los módulos X-bee, utilizo patrones de movimiento por captura de frames [14]

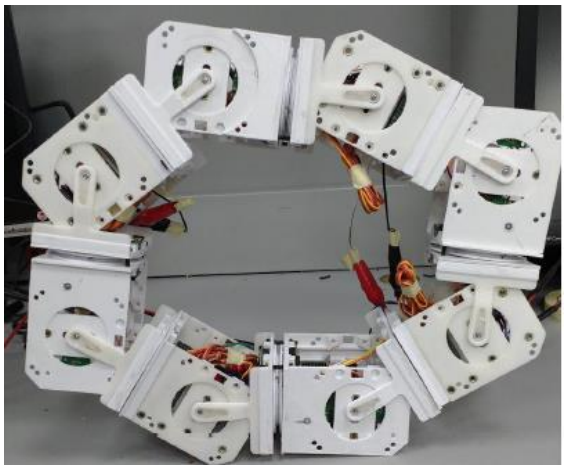


Figura 2-9 Arquitectura tipo rueda

Estructura con alimentación de fuente y uniones aseguradas con tornillos. Tomado de:[14]

2.6.4. Arquitectura Salamandra

La última versión de Mecabot desarrollada es la 5.0 en donde se ha desarrollado la arquitectura de salamandra, este nuevo modelo de Mecabot es más grande que su contraparte el Mecabot 5.0, pero sus motores poseen más torque, esta configuración es la que a alcanzado las velocidades más altas de desplazamiento.

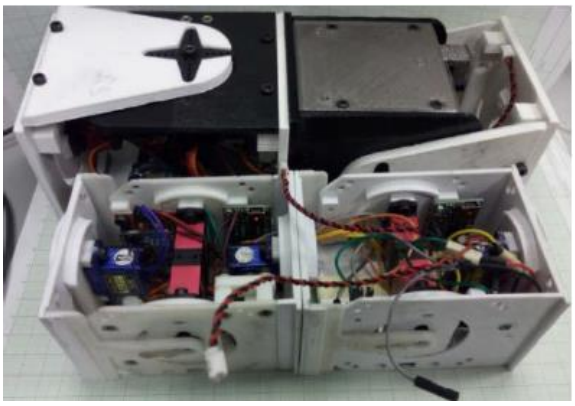


Figura 2-10 Diferencias entre el Mecabot 4.0 y 5.0

Semi-módulo del Mecabot 4.0 (frente) en comparación al semi-módulo del Mecabot 5.0 (atras).Tomado de:[16]

Figura 2-11 Arquitectura tipo cuadrúpeda salamandra

Mecabot 5.0 con diferente posiciones en su movimiento recto .Tomado de:[16]

2.7.Caminata humana principio de locomoción

La caminata humana es una actividad cotidiana, este movimiento ha sido estudiado muchas veces para hallar un modelo matemático donde se describa su comportamiento, el análisis de esta acción es importante para diferentes áreas, como la son la médica para prótesis y rehabilitación, en la computación para crear animaciones en diferentes simulaciones o juegos y en la robótica para exploración y creación de humanoides.[17] [18] [4]

Para el análisis de la marcha humana se puede analizar en diferentes planos, para este caso el plano sagital es el que se va a utilizar, ya que es un plano paralelo al movimiento cuando existe un desplazamiento en línea recta, los planos de un ser humano se pueden ver en la Figura 2-12[17]

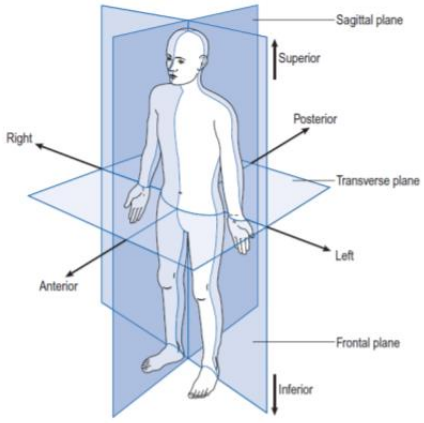


Figura 2-12 Planos del ser humano.

Tomado de:[19]

2.7.1. Etapas de balanceo

La fase de balanceo de la pierna se asemeja al movimiento de un péndulo. La fase de balanceo se divide en las etapas de:

- Aceleración: Se inicia en el momento de despegue del pie, con una aceleración máxima y velocidad inicial cero.
- Balanceo medio: La pierna pasa a la pierna que realiza el apoyo simple, en este instante ocurre la velocidad máxima de la pierna.
- Deceleración: Después del balanceo medio, comienza la deceleración, por ende, la disminución de velocidad de la pierna.

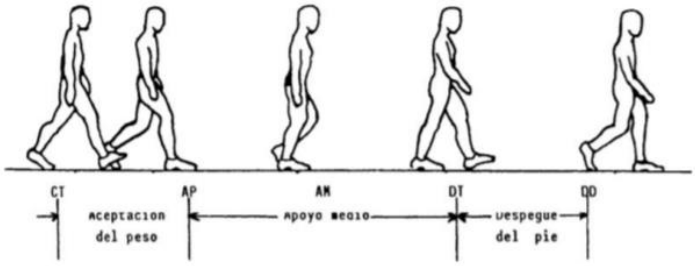


Figura 2-13 Ciclo de caminata humana

Tomado de:[19]

Los movimientos de los eslabones en el plano sagital son de extensión y flexión, los cuales son medidos desde la articulación de la cadera y la rodilla. En la Figura 2-14se aprecia la extensión y flexión para las articulaciones de la cadera y rodilla.

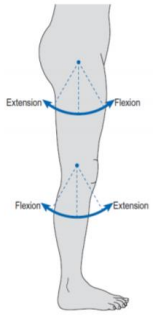


Figura 2-14 Eslabones de la pierna humana simplificados

Tomado de:[19]

2.7.2. Intervalo 1

La caminata humana se puede separar por tres secuencias diferentes en donde se hace uso de los diferentes eslabones y articulaciones de la pierna, el primer intervalo hace referencia de cuando el talón tiene contacto con la superficie y se desplaza hasta el punto medio de la marcha, el segundo intervalo comienza cuando se el talón está en el punto medio y hasta el levantamiento del mismo sobre la superficie y el tercero hace referencia a la etapa de balanceo, es decir cuando el talón está en el aire y se desplaza al punto de contacto.[19]

El movimiento que tiene la rodilla en el primer intervalo inicia completamente en extensión, luego la rodilla comienza a flexionarse y continua hasta cuando la planta del pie este completamente plana con la superficie de contacto, como se muestra en la siguiente Figura 2-15

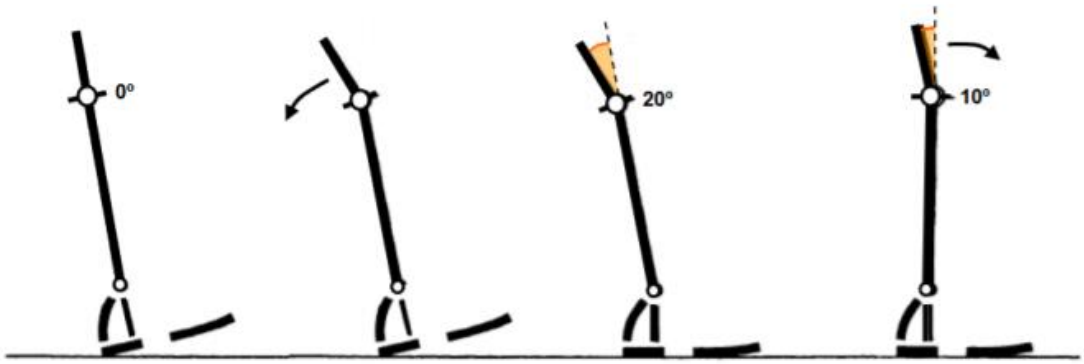


Figura 2-15 Ángulos de movimiento de la rodilla

Tomado de:[20]

La cadera en este intervalo presenta una flexión y cuando el pie se encuentra con la superficie de contacto y la flexión comienza a disminuir hasta cuando llega el talón al punto medio la flexión llega a los 0°, como se muestra en la Figura 2-16.

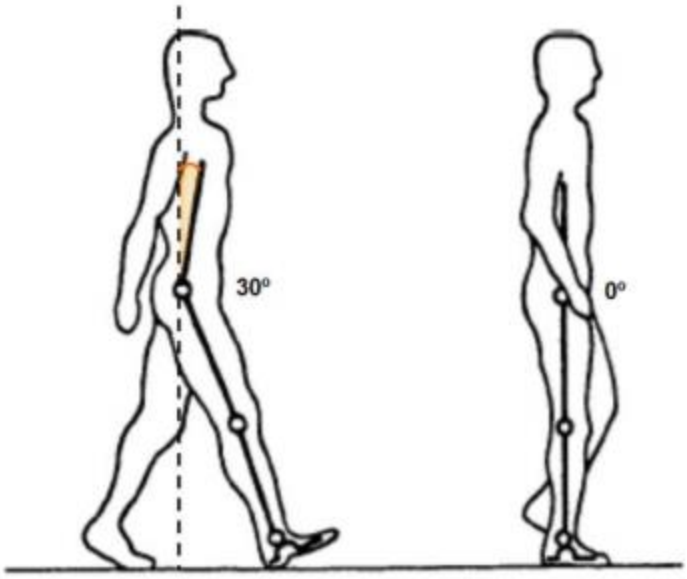


Figura 2-16 Ángulos de movimiento de la cadera

Tomado de:[20]

2.7.3. Intervalo 2

La rodilla comienza con un ángulo de flexión, esta disminuye casi a cero grados antes de que el talón se levante sobre la superficie de apoyo y cuando se despliega el talón y los dedos aumenta la flexión de la rodilla, como lo muestra en la Figura 2-17

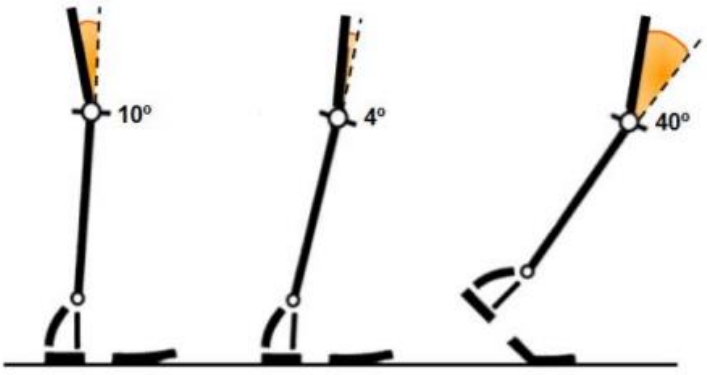


Figura 2-17 Ángulos de movimiento de la rodilla

Tomado de:[20]

La cadera comienza en una posición neutra la cual comienza a tener un proceso de extensión, cuando se llega al momento cuando el talón y los dedos se levanta de la superficie de contacto, la cadera alcanza un ángulo de extensión máximo, como se muestra en la Figura 2-18

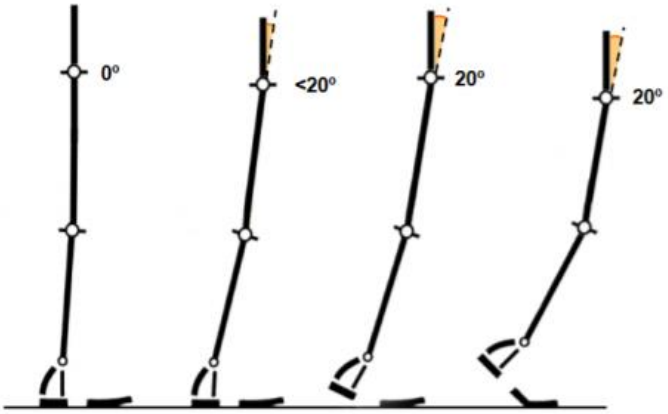


Figura 2-18 Ángulos de movimiento de la cadera

Tomado de:[20]

2.7.4. Intervalo 3

La rodilla en este intervalo tiene su punto máximo de flexión de aproximadamente 65°, mientras que la cadera parte de un punto neutral y comienza a flexionarse, como se aprecia en la Figura 2-19

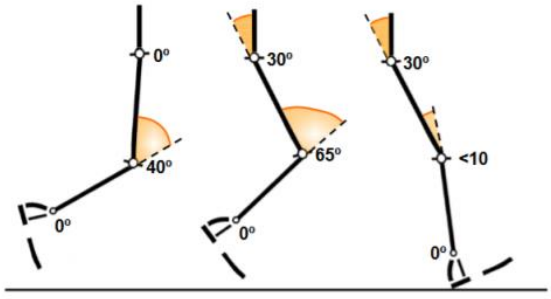


Figura 2-19 Ángulos de la rodilla y la cadera

Tomado de:[20]

El proceso de la caminata humana con los intervalos anteriormente mencionados, se representa gráficamente mostrando el siguiente resultado.

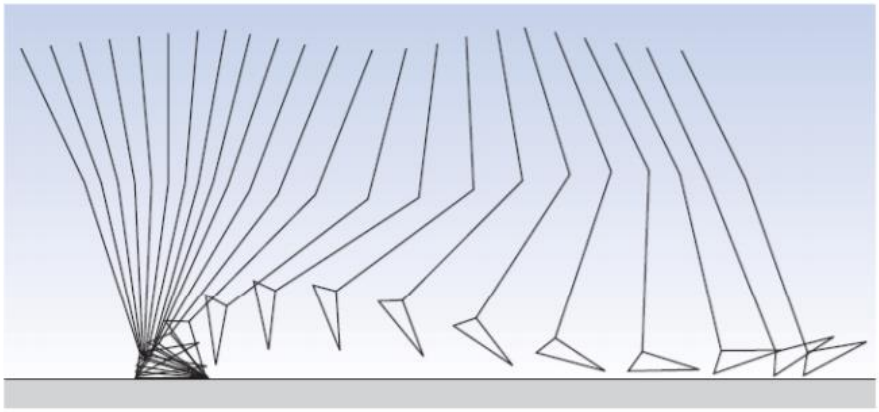


Figura 2-20 Simulación de movimiento

Tomado de:[19]

3. Desarrollo de la arquitectura bípeda utilizando Webots®

3.1. Clases de acoples entre módulos del Mecabot 5.0

Los módulos robóticos Mecabot 5.0 para poder formar diferentes arquitecturas, presentan acoples magnéticos, sin embargo, para esta aplicación en específico se usan tornillos para tener una mayor rigidez en la arquitectura. Los acoples posibles con el Mecabot 5.0 son cara-pivote, pivote-pivote, cara-cara y a caras laterales del módulo. [16]

3.2. Propuesta de arquitectura bípeda utilizando los módulos robóticos Mecabot 5.0

Existen diferentes maneras de implementar arquitecturas bípedas con robótica modular, en este caso se busca una arquitectura bioinspirada en la marcha humana, teniendo unas complicaciones como lo son los grados de libertad, para evitar que la arquitectura final sea muy grande y pesada se opta por simplificar unas articulaciones.

Para el diseño de la arquitectura bípeda se pretende imitar a la locomoción humana, para este proceso se hace una aproximación de los grados de libertad que tienen la pierna humana teniendo como resultado el siguiente esquema como se muestra en la Figura 3-1.

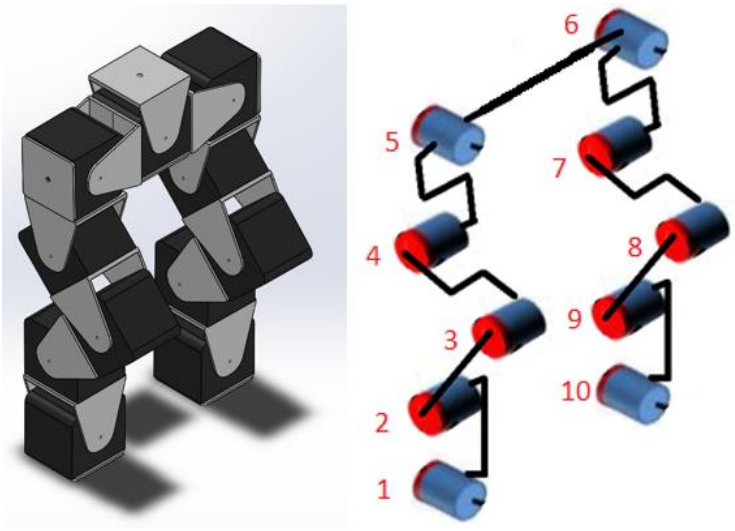


Figura 3-1 Robot bioinspirado para el Mecabot 5.0

Tomado de: elaboración propia.

El problema con esta arquitectura es el grande número de módulos que presenta, provocando que los motores no puedan mover el peso total de toda la arquitectura, entonces se opta por buscar un modelo donde el número de módulos sea el menor, se llega al siguiente modelo, donde cuenta con tres motores por pierna (cadera, rodilla y talón) respectivamente, como se muestra en la Figura 3-2, se adiciona un motor extra para el giro del sistema ubicado al final del sistema.

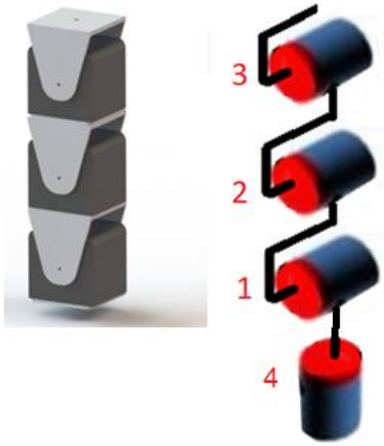


Figura 3-2 Pierna con Mecabot 5.0

Tomado de: elaboración propia.

El problema con este diseño es que al implementar la segunda pierna al momento de dar el paso la estructura pierde equilibrio ya que su centro de masa se encuentra entre las dos piernas y no hacia la pierna de apoyo como pasa con la caminata humana normalmente, una de las soluciones que proponen es el uso de una caminadora de las que usan los bebes para aprender a caminar, con esta herramienta ayuda a que la arquitectura no pierda el equilibrio mientas una de las piernas está en el aire.

Ilustración 3-1 Caminadora de la arquitectura

Tomado de: elaboración propia.

Una vez solucionado el problema del equilibrio se observan los eslabones principales para tener una marcha bioinspirada, estos serían tres los cuales son: el primero como el fémur, el segundo como la tibia – peroné y el tercero como punto de contacto con la superficie de contacto el cual es el pie.

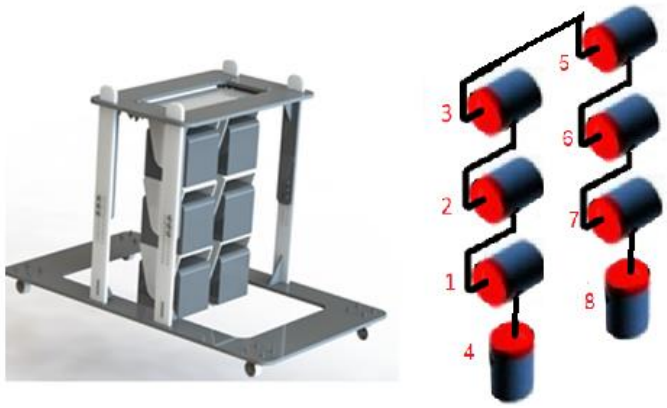


Figura 3-3 Arquitectura bípeda con caminadora

Tomado de: elaboración propia.

La estructura de la caminadora cuenta con dos maneras para graduar la altura, esta se ajusta para modificar el punto en que el pie toca con la supervise de contacto, esto conlleva a que se desplace más distancia, y por ende consuma más potencia los motores, la atura se ajusta a los costados de la estructura la cual se puede graduar cada tres milímetros por muesca y la otra es en la parte superior la cual cuenta con cuatro tornillos los cuales tienen un ajuste mucho más fino, funcionando como tornillos sin fin.

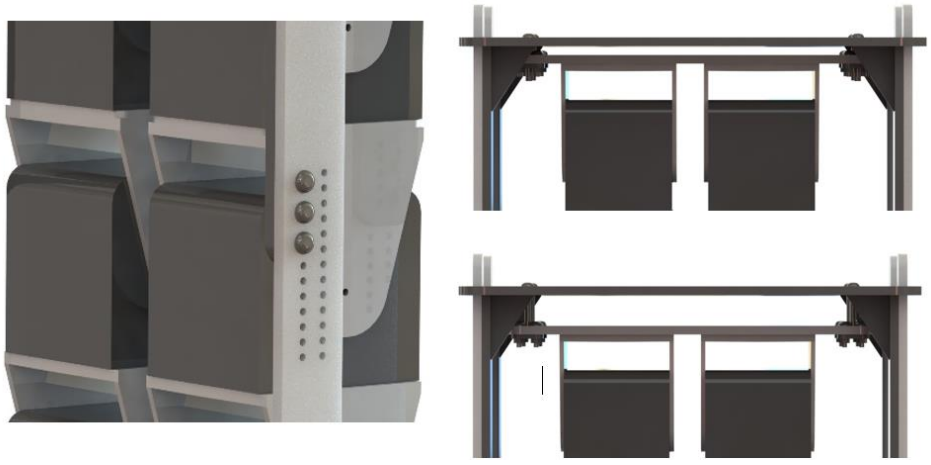


Figura 3-4 Mecanismo de ajuste de altura

Sistema lateral(izquierda) y el sistema de ajuste fino(derecha) Tomado de: elaboración propia.

Se puede observar en la Figura 3-4 los tornillos laterales de la estructura y también el sistema de ajuste fino como se muestra en la zona de la derecha.

3.3. Cinemática de las piernas

Los grados de libertad utilizados por la arquitectura se muestran en Figura 3-3, ya que las dos piernas son iguales, su cinemática es la misma, en sistema coordenado cero y tres corresponde al módulo superior y a la superficie de contacto, se encuentra un motor que gira la rueda de la superficie de contacto, pero se omite para calcular la cinemática.

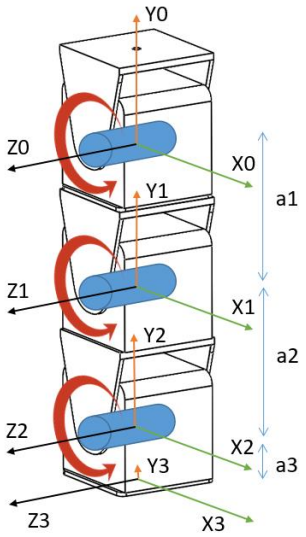


Figura 3-5 Mecanismo de ajuste de altura

Sistema lateral(izquierda) y el sistema de ajuste fino(derecha) Tomado de: elaboración propia.

Articulación	$\boldsymbol{\theta_i}$	$\boldsymbol{d_i}$	$\boldsymbol{a_i}$	$\boldsymbol{\alpha_i}$
1	θ_1	**0**	$\boldsymbol{a_1}$	**0**
2	θ_2	**0**	a_2	**0**
3	θ_3	**0**	a_3	**0**

Tabla 3-1 Parametros Denavit Hartenberg columna

Tomado de: elaboración propia.

Matriz e transformación:

$$A_0^3 = A_0^1 A_1^2 A_2^3 \qquad (1)$$

Donde:

$$A_0^1 = \begin{bmatrix} C\theta_1 & -S\theta_1 & 0 & a_1C\theta_1 \\ S\theta_1 & C\theta_1 & 0 & a_1S\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} A_1^2 = \begin{bmatrix} C\theta_2 & -S\theta_2 & 0 & a_2C\theta_2 \\ S\theta_2 & C\theta_2 & 0 & a_2S\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} A_2^3 = \begin{bmatrix} C\theta_3 & -S\theta_3 & 0 & a_3C\theta_3 \\ S\theta_3 & C\theta_3 & 0 & a_3S\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_0^3 = \begin{bmatrix} -C\theta_3(S\theta_{12} - C\theta_{12}) & S\theta_3(S\theta_{12} - C\theta_{12}) \\ C\theta_3(C\theta_1S\theta_2 - S\theta_1C\theta_2) & -S\theta_3(C\theta_1S\theta_2 - S\theta_1C\theta_2) \\ S\theta_3 & C\theta_3 \\ 0 & 0 \end{bmatrix} \dots$$

$$\dots \begin{matrix} C\theta_1S\theta_2 - S\theta_1C\theta_2 & C\theta_1\big(a_1 + C\theta_2(a_2 + a_3C\theta_3)\big) - S\theta_{12}(a_2 + a_3C\theta_3) \\ S\theta_{12} - C\theta_{12} & C\theta_1S\theta_2(a_2 + a_3C\theta_3) + S\theta_1\big(a_1 + C\theta_2(a_2 + a_3C\theta_3)\big) \\ 0 & a_3S\theta_3 \\ 0 & 1 \end{matrix} \Bigg] \quad (2)$$

3.4.Perfiles de movimiento de las piernas.

Se obtienen los resultados de las gráficas mencionadas anteriormente y para un mejor manejo de las señales se manejan en forma de sumatoria de senos, esto ayuda para modificar la amplitud y la frecuencia de cada una de las señales.

La sumatoria de senos ayuda para que en el microcontrolador se maneje un control descentralizado y sin tablas de datos, haciendo que el costo computacional sea menor y el movimiento de los motores sean más suaves.[16] [21]

$$Perfil = \sum A_n \sin(B_n t + C_n) \quad (3)$$

Los perfiles de movimiento son tres, uno para cada articulación las cuales son cadera, rodilla y tobillo, como se mencionó anteriormente una vez tomando los datos de la gráfica, se procede a generar las sumatorias de senos con MATLAB® con la función “Curve Fitting Tool”, con esta herramienta se hallan tres funciones, tanto la pierna de la derecha e izquierda presentan los mismos perfiles de movimiento, pero con un desfase de trecientos sesenta grados, para este desfase se hace de la siguiente manera.

$$Perfil = \sum A_n \sin\left(B_n t + C_n + \frac{Desfase * B_n}{\pi}\right) \quad (4)$$

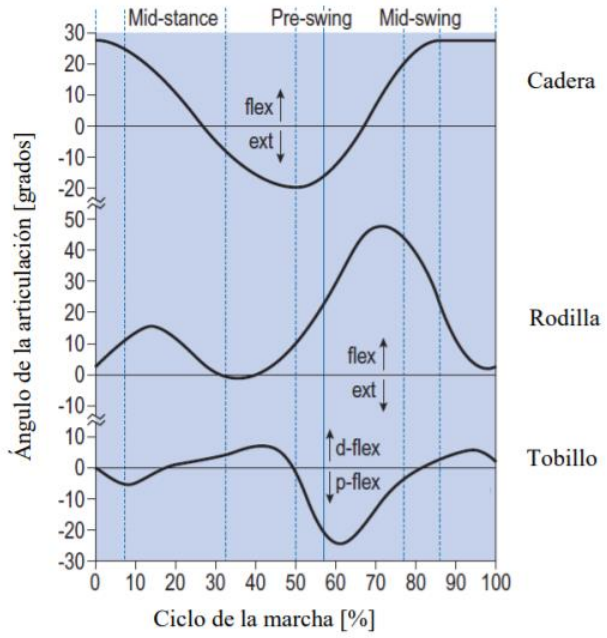


Figura 3-6 Perfiles de movimiento

Tomado de:[19]

3.5.Ajuste de coordinación locomoción recta y hacia atrás

Para una buena coordinación entre las dos piernas es necesario cumplir la ecuación (5), para cada uno de los motores.

$$E = E_1 + E_2 + E_3 + E_4 + E_5 \quad (5)$$

Los caracteres E_n corresponden a las siguientes expresiones

$$E_1 = A_{[i]} * \sin(\pi * F * t + bias_{[i]}) \quad (6)$$

$$E_2 = \frac{A_{[i]}}{2} * \sin(2\pi * F * t + bias_{[i]}) \quad (7)$$

$$E_3 = \frac{A_{[i]}}{4} * \sin(3\pi * F * t + bias_{[i]}) \quad (8)$$

$$E_4 = \frac{A_{[i]}}{8} * \sin\left(4\pi * F * t + bias_{[i]}\right) \quad (9)$$

$$E_5 = \frac{A_{[i]}}{16} * \sin\left(5\pi * F * t + bias_{[i]}\right) \quad (10)$$

Cada una de las piernas presenta un desfase de 180° por lo cual para el desfase se hace el siguiente calculo.

$$Offset_{[i]} = bias_{[i]} + \frac{Constante * F_{[i]}}{\pi} \quad (11)$$

Con este desfase independientemente de la frecuencia, el offset se mantiene y no se pierde la relación de la señal, en Figura 3-7 se observa el desfase de las señales sinusoidales a 180°.

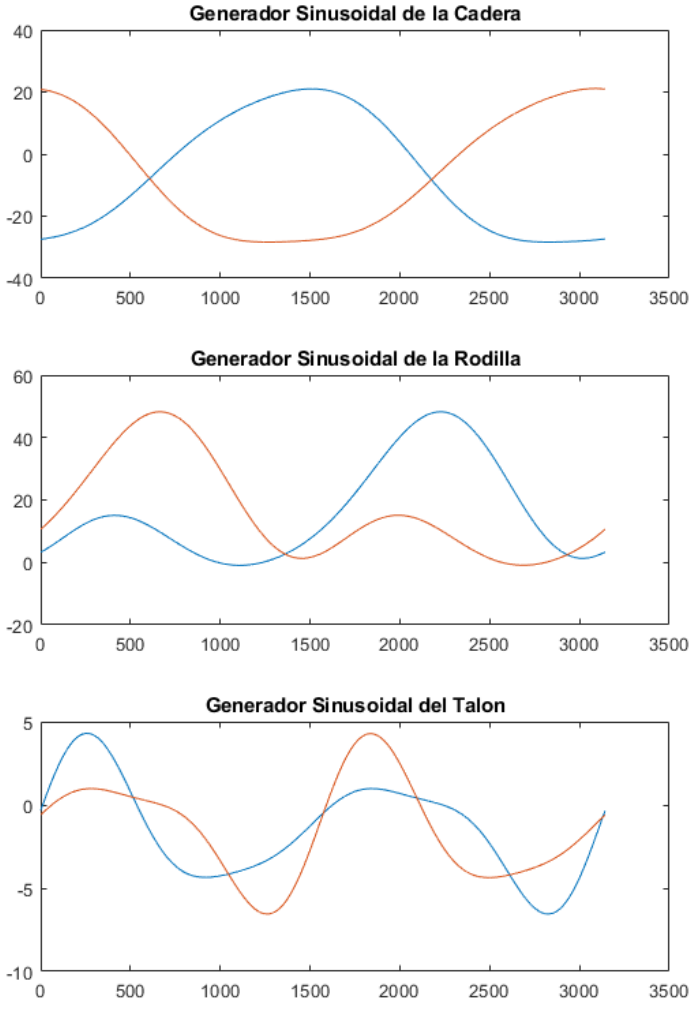


Figura 3-7 Perfiles de movimiento desfasados

Señales sin desfase(Azul) Señal desfasada(Naranja)Tomado de: elaboración propia.

3.6.Ajuste de coordinación locomoción en giros

Para efectuar giros el sistema tiene que estar con los dos puntos de contacto con la superficie, mientras que reproducen movimientos contrarios, es decir, mientras la pierna izquierda avanza la pierna derecha retrocede al mismo tiempo, adicionalmente las ruedas de contacto realizan un giro coordinado en la misma orientación.

Figura 3-8 Rueda del módulo rotando

Tomado de: elaboración propia.

El sistema de giro, por cada paso dado gira un Angulo determinado, sin importar la frecuencia de entrada, el Angulo de giro depende de la fricción de la superficie de contacto y de la altura del bípedo.

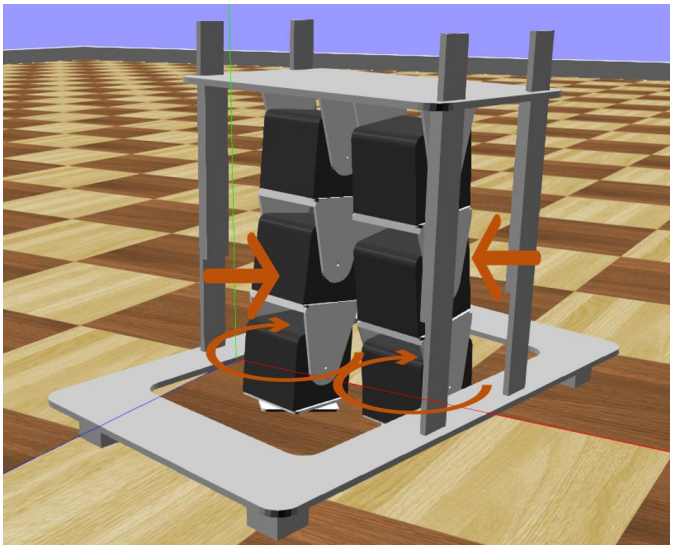


Figura 3-9 Giro del robot

Tomado de: elaboración propia.

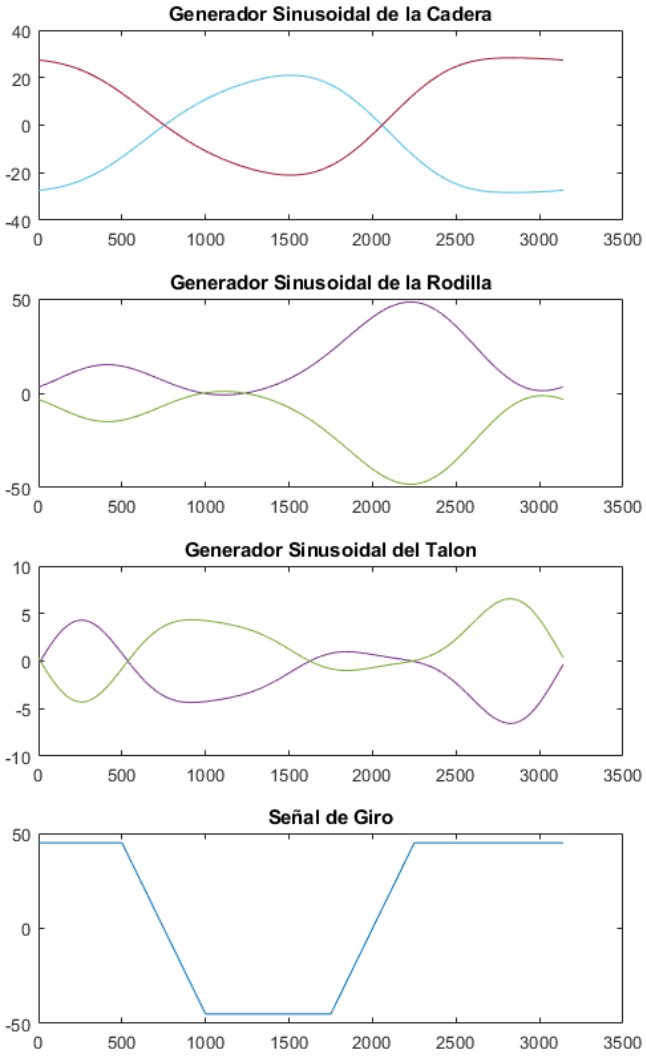


Figura 3-10 Perfiles de movimiento invertidos y señal de giro

Señales sin desfase(Azul) Señal invertida(Naranja)Tomado de: elaboración propia.

4. Simulación de la arquitectura utilizando software Webots®

4.1. Webots®

Webots ® es un software que permite desarrollar modelos, programar y simular un robot bajo diferentes ambientes. Es creado en 1996 con el apoyo del EPFL. Maneja distintos lenguajes: C, C++, Java, Python, Matlab y ROS, además permite la comunicación TCP/IP con otras interfaces de simulación o control.[22]

Adicional a las clases de nodos asociados a la creación del entorno existen otros tipos que permiten el modelo, creación, simulación y control del robot. A continuación, se listan los principales nodos y conceptos asociados para realizar una simulación en Webots ®:[23]

Actuator: Estos nodos reciben los comandos de un *Controller*, son llamados a través de este por medio del nombre asignado al actuador en sus propiedades.

Basic Time Step: Es un campo asociado al nodo *WorldInfo*, define el incremento discreto de tiempo de la simulación. Se da en milisegundos.

Controller: Recurso asociado al *Robot*. Puede ser programado en C, C++, Python, Java o Matlab. Se adiciona a través de Wizards > New Robot Controller

Controller time step: Incremento discreto de tiempo dentro del *control loop*. Se cambia a través del *wb_robot_step* en el archivo del *Controller*.

Device: Término asociado a un *Actuator* o un *Sensor*.

Dynamic/ Kinematic solid: A diferencia de un *Kinematic solid*, el *Dynamic Solid* tiene asociado un nodo *Physics* que permite movimiento bajo ciertas condiciones de peso, fricción, etc.

Robot: Nodo utilizado para definir un robot predeterminado o con modelo propio. Asociado a este están: *Controller*, actuadores, sensores, *Physics*, entre otros.

Sensor: Estos nodos reciben los comandos de un *Controller*, son llamados a través de este por medio del nombre asignado al sensor en sus propiedades.

Solid: Nodo que define un elemento cualquiera el cual puede tener asociado los nodos *Shape*, *Transform*, actuadores, sensores, etc. También puede estar asociado a un nodo *Robot.*

4.2.Modelamiento de la arquitectura en CAD

El modelamiento se realiza en solidworks® y luego se exporta a webots®, todos los ensamblajes se realizan con unas piezas simplificadas, esto con el fin de que la simulación no esté muy cargada y funcione de manera óptima, para la importación de objetos 3D en webots® se hace una simplificación de la caminadora y de los módulos robóticos Mecabot 5.0, esto se hace para que la simulación en el entorno virtual utilice menos recursos computacionales, en la Figura 4-1 se observa las tres piezas simplificadas del Mecabot 5.0 los cuales son pivote(blanco), rueda(blanco) y cuerpo(negro).

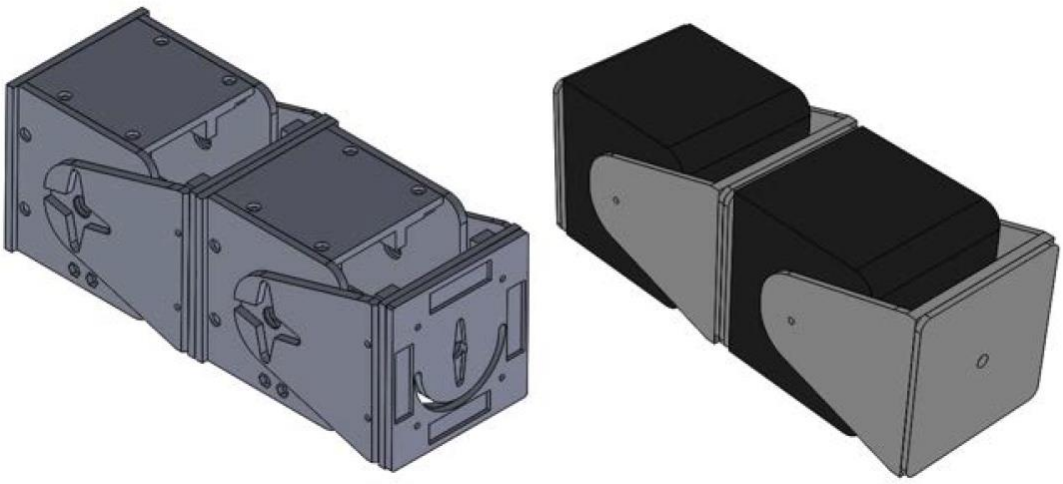


Figura 4-1 Simplificación del CAD para la simulación en Webots ®

Sema-módulo Mecabot 5.0 completo (izquierda) y Sema-módulo Mecabot 5.0 simplificado (derecha)
Tomado de: [16]

Para manejar los componentes del Mecabot 5.0 es necesario exportar las piezas de solidworks® en formato VRML97, una vez con las piezas con formato VRML97 se importan en webots® y se procede ubicar las piezas en una posición específica, para este paso se utilizan tres grupos principales el primero se compone del cuerpo con la rueda, el segundo se compone del cuerpo, la rueda y el pivote y por último la estructura que sostiene a los módulos acoplados,

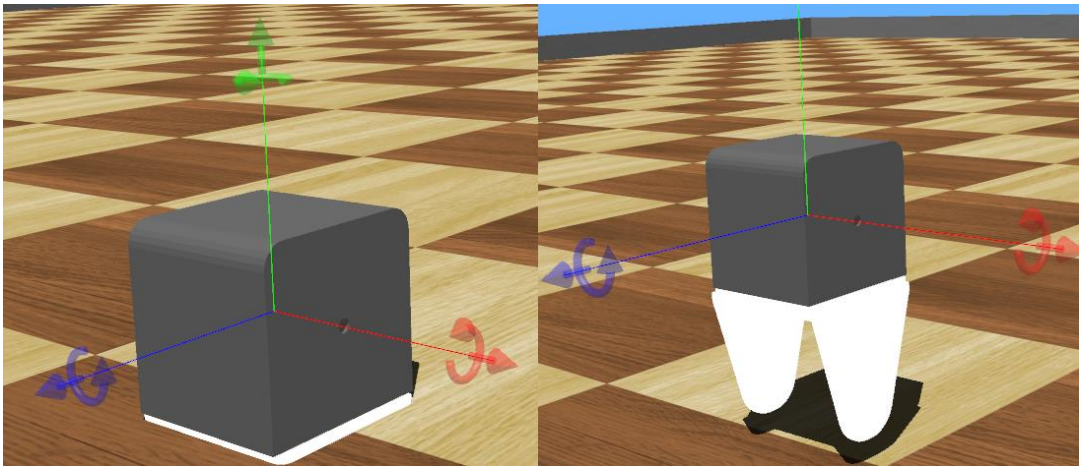


Figura 4-2 Solidos principales de los semi-módulos del Mecabot en Webots ®

Cuerpo y la rueda (izquierda) y el sistema de ajuste fino(izquierda) Tomado de: elaboración propia

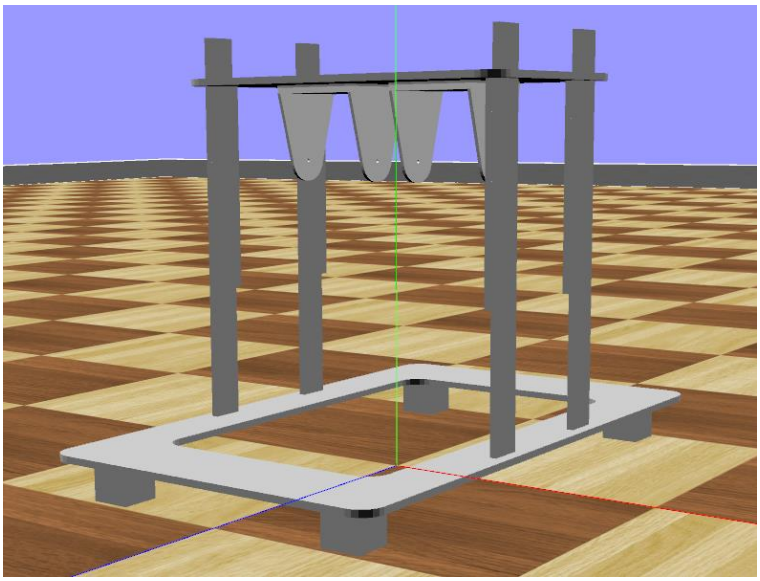


Figura 4-3 Caminadora simplificada con los pivotes en Webots ®

Tomado de: elaboración propia

Teniendo los grupos se procede a acoplar, con *HingeJoint*, para este procedimiento se tiene define un sólido para albergar al *HingeJoint*, esto se hace para ajustar las coordenadas del eje de giro, se agrega los parámetros correspondientes y se define como un motor rotacional, se agrega un sólido en el *endPoint* en donde se ubicará el siguiente grupo.

Figura 4-4 Jerarquía de hijos para un HingeJoint

Esta resaltado en amarillo, las coordenadas del eje de giro del HingeJoint Tomado de: elaboración propia

Una vez acoplados todos los grupos y definidos todos los *HingeJoint*, se especifican los *boundingObject* y las *Physics* de los elementos correspondientes.

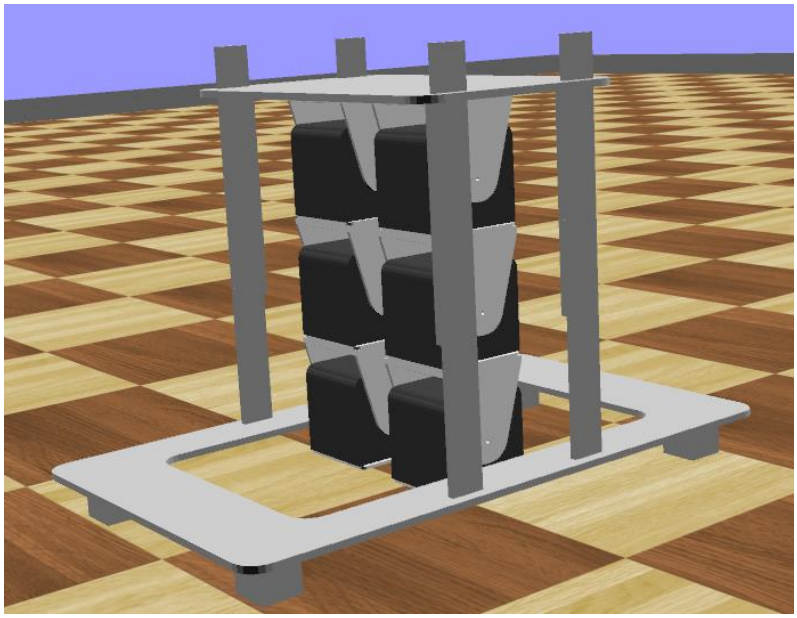


Figura 4-5 Arquitectura bípeda con el Mecabot 5.0 en Webots ®

Tomado de: elaboración propia

4.3.Simulación de la locomoción

La simulación se realiza con el software de webots® y se prueba las secuencias mencionadas anteriormente, otra de las maneras de manipular el mecanismo es ajustando la altura de la caminadora esta modifica el punto en que el eslabón 3 y la superficie presentan el contacto. Las pruebas que se realizaron fueron las siguientes

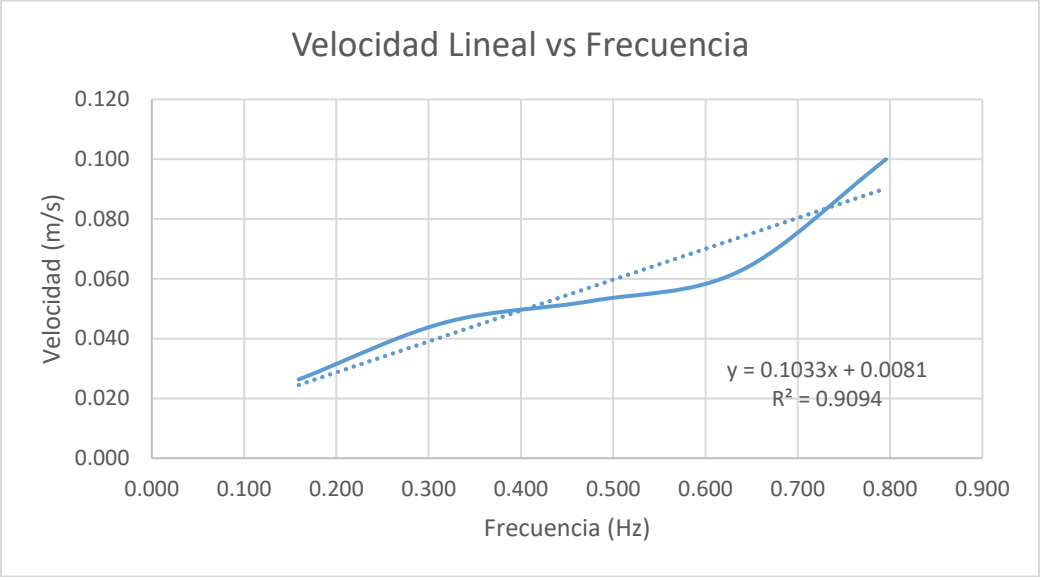


Figura 4-6 Relación de velocidad lineal y frecuencia

Tomado de: elaboración propia

Se procede a variar la frecuencia de los generadores sinusoidales para observar un comportamiento relacionado con la velocidad lineal del sistema, el resultado observado de la simulación permite relacionar como aumenta la velocidad del sistema con respecto a la frecuencia.

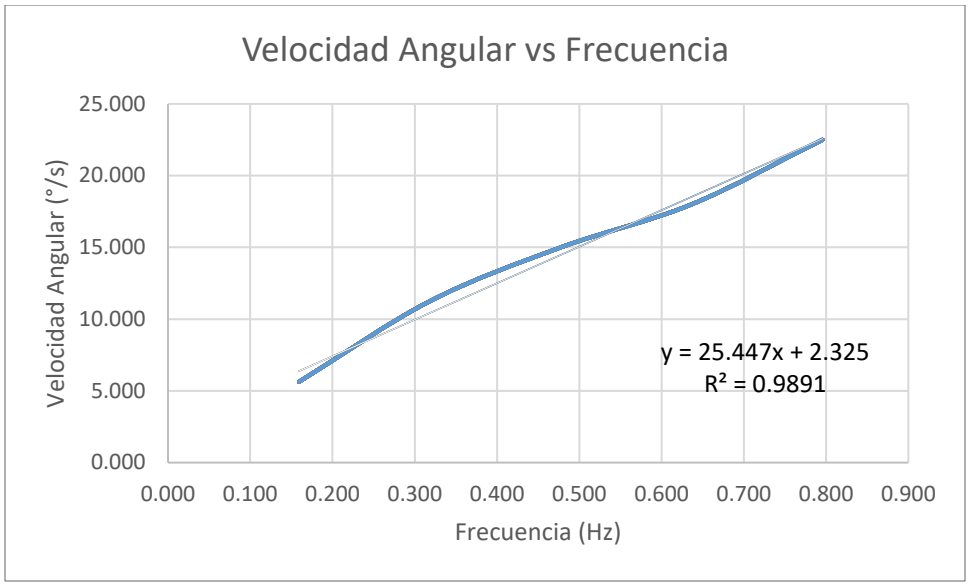


Figura 4-7 Relación de velocidad angular y frecuencia

Tomado de: elaboración propia

Como se observa en la Figura 4-7, la velocidad angular también aumenta con respecto a la frecuencia de los generadores sinusoidales, dando como resultado una relación directa entre la velocidad angular y la frecuencia de los generadores.

5. Ensamble de la arquitectura bípeda con los módulos robóticos Mecabot 5.0

5.1. Mecabot 5.0

Para la implementación de la arquitectura bípeda se opta para que el sistema posea dos controladores, cada controlador manejara una extremidad. Los módulos pueden disponer de ruedas sin imanes que permiten ser acopladas a otros módulos mediante tornillos o por un motor común, cuando las ruedas presentan imanes basta con unirlas a otro modulo las mismas ruedas.[16]

5.2. Componentes electrónicos en el Mecabot 5.0

Los componentes electrónicos del Mecabot 5.0 se describen a continuación de una manera detallada y como se configuraron cada uno de ellos.

5.2.1. Xbee S2

Los módulos Xbee son basados en el estándar ZigBee de redes Mesh, el cual permite baja latencia, alto tráfico de datos y buena sincronización. La tarjeta utilizada por el Mecabot 5.0 es la serie 2 ZB de la familia Xbee.[24]

3.3V @ 40mA
250kbps Max data rate
2mW output (+3dBm)
400ft (120m) range
Built-in antenna
Fully FCC certified
6 10-bit ADC input pins
8 digital IO pins
128-bit encryption
Local or over-air configuration
AT or API command set

Figura 5-1 Especificaciones Xbee XB24-Z7WIT

Tomado de: [25]

La Xbee se programa atreves del programa XCTU®, en este programa se configura parámetros de la Xbee como la velocidad de la comunicación y cuales tarjetas van a ser las transmisoras y las receptoras de los datos. La comunicación empleada en la arquitectura bípeda es muy similar a la arquitectura cuadrúpeda tipo salamandra, en donde se configuran los dispositivos en configuración transparente (AT), lo cual permite comunicación serial punto multipunto.[16] [26]

La trama de bits que utiliza la arquitectura bípeda se muestra en la Tabla 5-1 en donde la modalidad indica la función especificada por el usuario, las cuales son, M: manual, S: secuencia de movimiento y R sincronización de los módulos.

Bit	0	1	2	3	4	5
	Modalidad	Complemento				

Tabla 5-1 Trama de datos comunicación para arquitectura bípeda

Tomado de: [16]

La modalidad (M) permite el manejo por separado de cada motor de la arquitectura bípeda, esto con el fin de verificación y calibración de los motores.

Bit	0	1	2	3	4	5
	M	ID Módulo	ID Motor	Signo	Ángulo	

Tabla 5-2 Trama de datos para modalidad manual (M)

Tomado de: [16]

La modalidad (S) significa los generadores sinusoidales y el movimiento hace referencia a "U" movimiento hacia adelante, "D" movimiento hacia atrás, "L" giro hacia la izquierda y "R" giro hacia la derecha

Bit	0	1	2	3	4	5
	S	0	0	Movimiento	Frecuencia	

Tabla 5-3 Trama de datos para modalidad generadores sinusoidales (S)

Tomado de: elaboración propia

El reset se ejecuta con el fin de sincronizar los semi-módulos del Mecabot, con el fin de que el offset de los generadores sinusoidales de cada semi-modulo este sincronizado.

Bit	0	1	2	3	4	5
	R	0	0	set/reset	0	0

Tabla 5-4 Trama de datos para modalidad Reset para sincronizar los modulos (R)

Tomado de: elaboración propia

Para la comunicación de los semi-módulos del Mecabot se nombra cada uno de los módulos respectivamente, como se muestra en la Figura 5-2, en donde el router es el computador y los receptores son los dos semi-módulos del Mecabot 5.0.

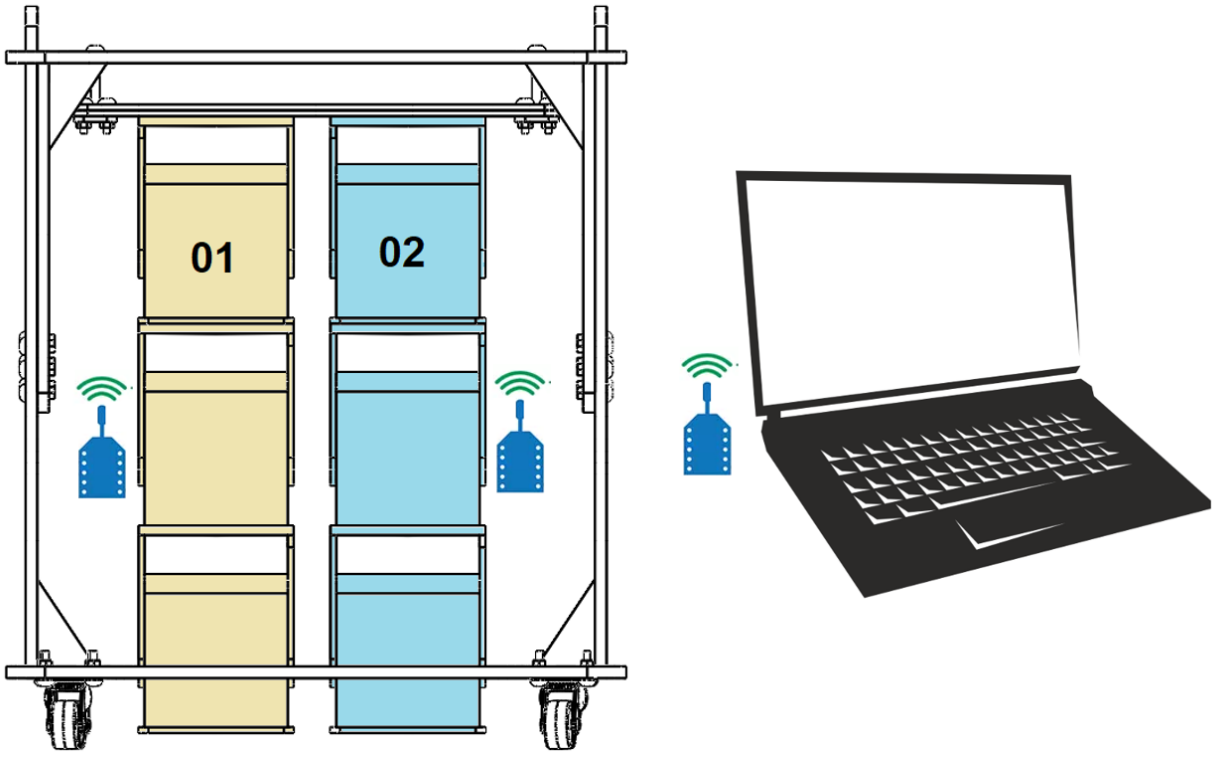


Figura 5-2 Comunicación serial Xbee computador Mecabot 5.0

Tomado de: elaboración propia

5.2.2. Teensy

Los semi-módulos del Mecabot 5.0 manejan como cerebro la versión 3.2 de la teensy, Esta tarjeta es muy versátil, ya que el software Teensyduino es compatible con Arduino, el cual hace el proceso de inicialización de registros más simple.

Dimensiones: 35 mm x 18 mm
Procesador: 32 bits ARM Cortex- M4 48 MHz CPU
128K de memoria Flash, 16K RAM, EEPROM 2K
14 entradas analógicas de alta resolución (13 bits utilizable, hardware de 16 bits)
34 digital I / O Pins (10 compartido con analógico)
10 salidas PWM
7 temporizadores de intervalos / retrasos, separadas de PWM
3 UARTs (puertos serie)
SPI, I2C, I2S, IR modulador
I2S (para la interfaz de audio de alta calidad)

Figura 5-3 Especificaciones Teensy 3.2

Tomado de: [27]

Cuando se usa por primera vez una tarjeta teensy 3.2, es necesario verificar que el *Windows Serial Installer* este correctamente instalado, una vez este proceso hecho, se carga un programa vació con el programa de arduino, sin seleccionar ningún puerto COM, luego se observara como puerto COM la teensy para trabajar sin ningún problema con ella.

5.2.3. Motores

Los motores utilizados para los semi-módulos del Mecabot 5.0 consisten en un servo-motor y un micro-servo-motor

Peso: 15,8 g
Dimensiones: 23 x 12 x 29 mm
Torque @ 4.8V: 3,1 kg / cm
Torque @ 6.0V: 3,9 kg / cm
@ 4.8V Velocidad: 0.16 sec / 60 ° sin carga
@ 6.0V Velocidad: 0.13 sec / 60 ° sin carga
Engranaje: Metal gear rodamiento / bola

Figura 5-4 Especificaciones Servomotor Power HD 1810MG

Tomado de: [28]

Torque(4.8V): 15.5 kg-cm (215.3 oz/in)
Torque(6.0V): 17.0 kg-cm (236.1 oz/in)
Velocidad: 0.16 sec (4.8V) | 0.14 sec (6.0V)
Rango de operación: 4.8 ~ 6.0 DC Volts
Peso: 63.0 g (2.22 oz)
Tipo de rodamiento: rodamiento de bola x 2
Tipo de Motor ：Motor DC
Temperatura de operación: -20°C~60°C
Frecuencia de trabajo: 1520μs / 50hz
Dimensiones: 40.7 x 20.5 x 39.5 mm

Figura 5-5 Especificaciones Servomotor HD 1501MG

Tomado de: [29]

5.2.4. Driver Pololu

El driver Pololu también se utilizó para el Mecabot 4.0, Debido a que se redujeron los motores utilizados en la versión 5.0 se usa solo un driver por modulo y no dos como en el modelo pasado, para utilizar de manera correcta el driver se tiene que instalar la librería *PololuMaestro* en arduino e instalar el software *Maestro Control Center* de pololu, una vez en esta interfaz se ajustan parámetros como los de comunicación, límites de target entre otros.

Dimensiones: 2.16 cm x 3.05 cm
Peso: 4.8 g
Canales: 6
Baud: 300-200000 bps
Voltaje mínimo de operación: 5 V
Voltaje máximo de operación: 16 V

Figura 5-6 Especificaciones del Driver Micro Maestro Pololu 6 canales

Tomado de: [30]

5.2.5. Regulador Pololu y Protección Circuit Module

Los reguladores implementados en el mecabot 5.0 son dos, el D24V22FS y el BEC 5V/6V – 5A, el segundo regulador se usa para alimentar los motores, ya que con 6V de alimentación se tiene el máximo torque de los motores.

Dimensiones: 0.7" x 0.7" x 0.31"
Peso: 2.3 g
Voltaje mínimo de operación: 5.3 V
Voltaje máximo de operación: 36 V
Corriente de salida continua: 2.5 A
Voltaje de salida: 5 V
Protección contra voltaje reverso

Figura 5-7 Especificaciones del Regulador D24V22F5 Pololu

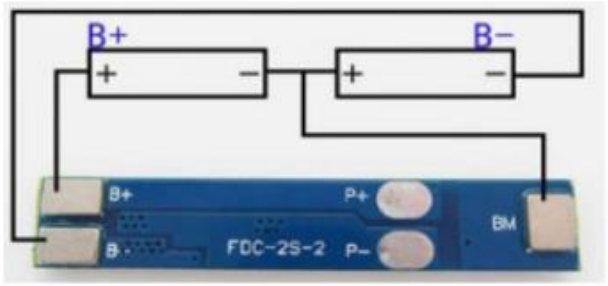
Tomado de: [31]

Dimensiones 42mm x 11mm x 24mm
Voltaje de operación 8V - 26V
Voltaje de salida: 5V - 6V
Corriente de salida: 5A

Figura 5-8 Especificaciones regulador switching BEC 5V/6V – 5A

Tomado de: [32]

los semi-módulos mecabot 5.0 manejan dos baterías las cuales están conectadas en serie, esta conexión se realiza bajo un circuito con protección, el cual se encarga de evitar daños por cortocircuito, sobrecarga y sobre descarga principalmente, las especificaciones se encuentran en la Figura 5-9, para el manejo de la tarjeta se tiene que realizar una precarga.



Dimensiones: 36 x 6 x 2 mm
Voltaje de carga máximo: 8V
Corriente de carga: 3A
Corriente de descarga: 3A
Detección de sobre carga por celda máximo: 4.38V
Detección de sobre descarga por celda: 2.9 V
Temperatura de Trabajo: -40°C a 50°C

Figura 5-9 Especificaciones PCB FDC-2S-2

Tomado de:[33] [34]

5.3.Procedimiento de ensamblaje de los módulos robóticos Mecabot 5.0

La arquitectura bípeda maneja dos módulos robóticos, distribuidos con un controlador para cuatro motores por cada módulo, en la Figura 5-10 se observa la ubicación de cada uno de los componentes usados.

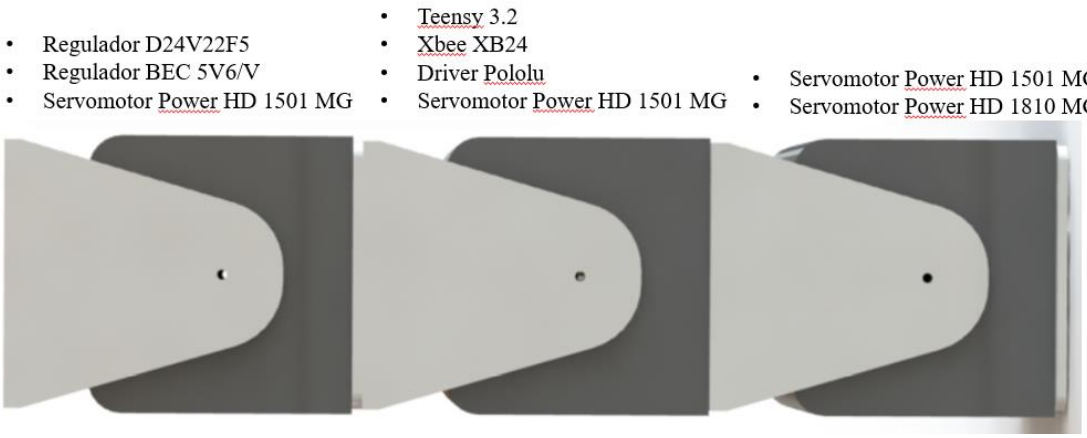


Figura 5-10 Distribución modulo arquitectura bípeda

Tomado de: elaboración propia

La conexión implementada en los módulos consiste, en dos puertos seriales de la teensy conectados al pololu maestro y a la xbee, correspondientemente, el pololu maestro conectado adicionalmente a cada uno de los motores instalados, el regulador D24V22F5 alimentando al pololu maestro, teensy y xbee, mientras que el regulador BEC 6V conectado al polulu maestro, pero como alimentación de los motores, este esquema se puede observar en la Figura 5-11.

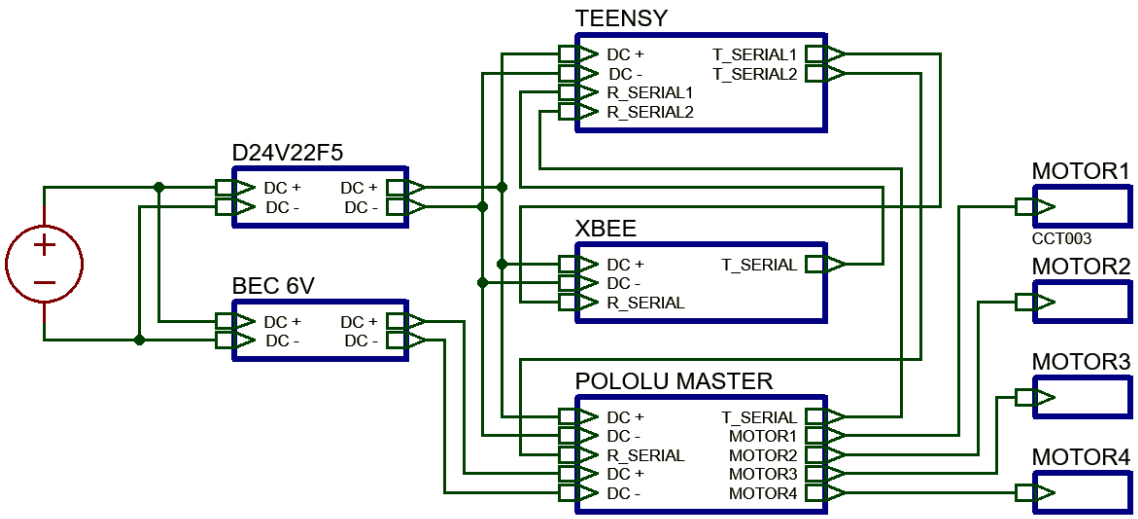


Figura 5-11 Plano eléctrico de la arquitectura bipeda

Tomado de: elaboración propia

Debido al alto amperaje que utiliza cada módulo entre 2.5A - 3.8A es necesario que el cable de alimentación de los motores sea de filamentos con un calibre mínimo de 16, esto para evitar sobrecalentamientos en los cables de alimentación y adicionalmente ruido que afecte el mecanismo. Para ensamblar la arquitectura bípeda primero se comenzó con la impresión de las piezas necesarias para el armado correspondiente de la misma. Luego se procede a la unión de ciertas piezas para asegurarlas dentro de la estructura, estas piezas sirven para mantener a la teensy, xbee y pololu master en un lugar fijo dentro de los módulos.

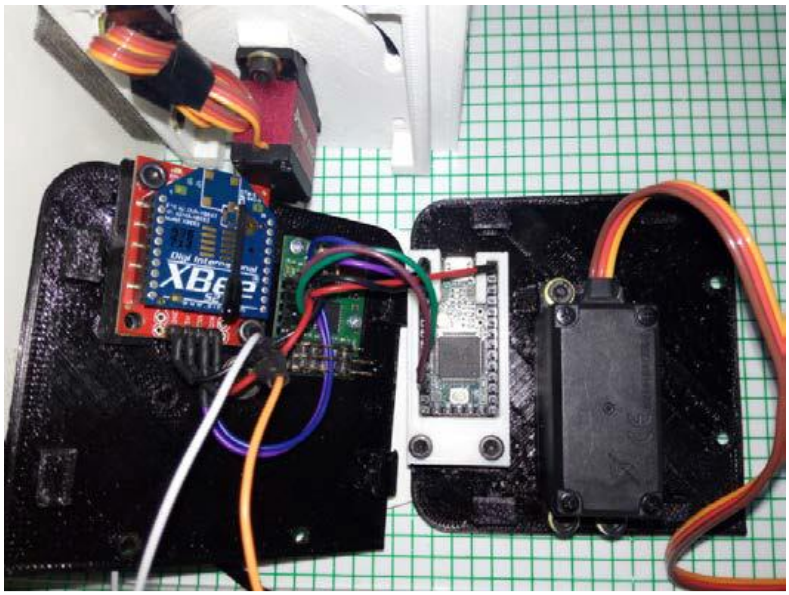


Figura 5-12 Fase de ensamble de elementos electrónicos del Mecabot 5.0

Tomado de: [16]

El Mecabot 5.0 emplea un sistema de baterías y protección de estas, estas se omiten debido a que las baterías entregan una corriente insuficiente para poder mover la arquitectura en una manera óptima para su desarrollo funcional.

5.4. Caracterización de los motores

Para la caracterización de los motores se utiliza el Maestro Control Center®, y modificar los parámetros de comunicación del puerto serial a 115200 de baud rate, para que la comunicación entre componentes este sincronizada, debido a oscilaciones que se generan cuando llega a los puntos de 2500, el rango para manejar los motores HD1501MG se reduce a 2200, esto no representa ningún problema, ya que no es necesario que los motores lleguen a ángulos rectos, Para los motores HD 1810MG es diferente porque presentaban oscilaciones entre valores de 600 y 2500, por lo tanto su rango se redujo a 800 y 2000

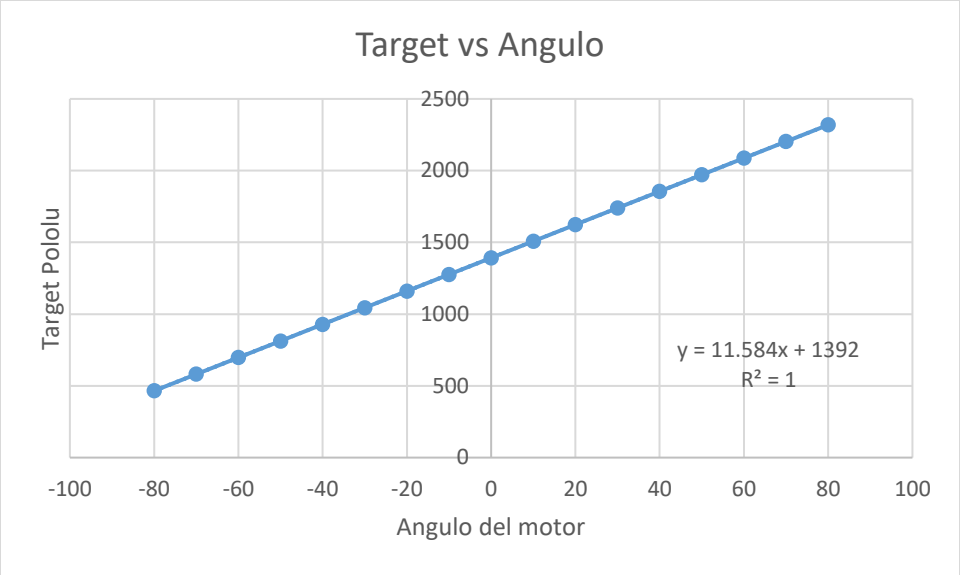


Figura 5-13 Caracterización del servomotor HD1501MG

Tomado de: elaboración propia

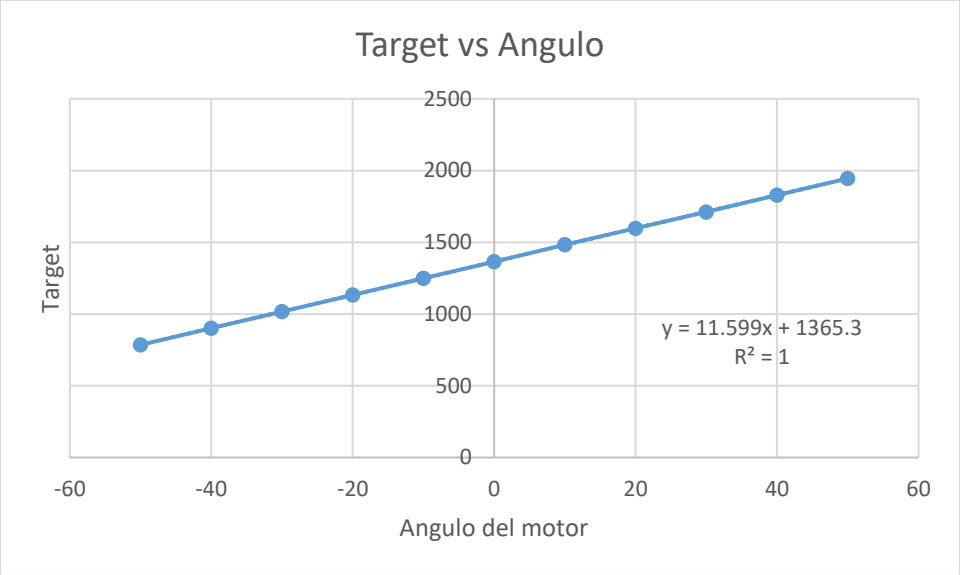


Figura 5-14 Caracterización del servomotor HD1810MG

Tomado de: elaboración propia

6. Pruebas de la arquitectura bípeda con los módulos robóticos mecabot 5.0

Debido a que el robot Mecabot 5.0 es pensado para operaciones de exploración, los movimientos son probados en distintas superficies, primeramente, en ambientes estructurados en el laboratorio, donde la superficie es totalmente plana, pero presenta diferentes grados de fricción (lisa o rugosa) y firmeza (blanda o dura). Posteriormente se prueba el robot en ambientes no estructurados fuera del laboratorio que presentan distintos grados de irregularidad: andén, pasto y rocoso, siendo el andén el terreno que menos irregularidades tiene.

6.1. Interfaz de usuario en MATLAB®

La interfaz de usuario desarrollada, contiene 4 paneles los cuales son, comunicación que se encarga de abrir o cerrar el puerto en donde esté conectado el módulo de la Xbee, en este caso 'COM6', el parámetro se varia dentro del código fuente, dos paneles de parámetros variables, un que está en la esquina superior derecha se encarga de variar los ángulos de 80° a 80° dependiendo del módulo y motor seleccionado, el que está en la esquina inferior izquierda, se encarga de enviar una frecuencia de operación y el movimiento deseado.

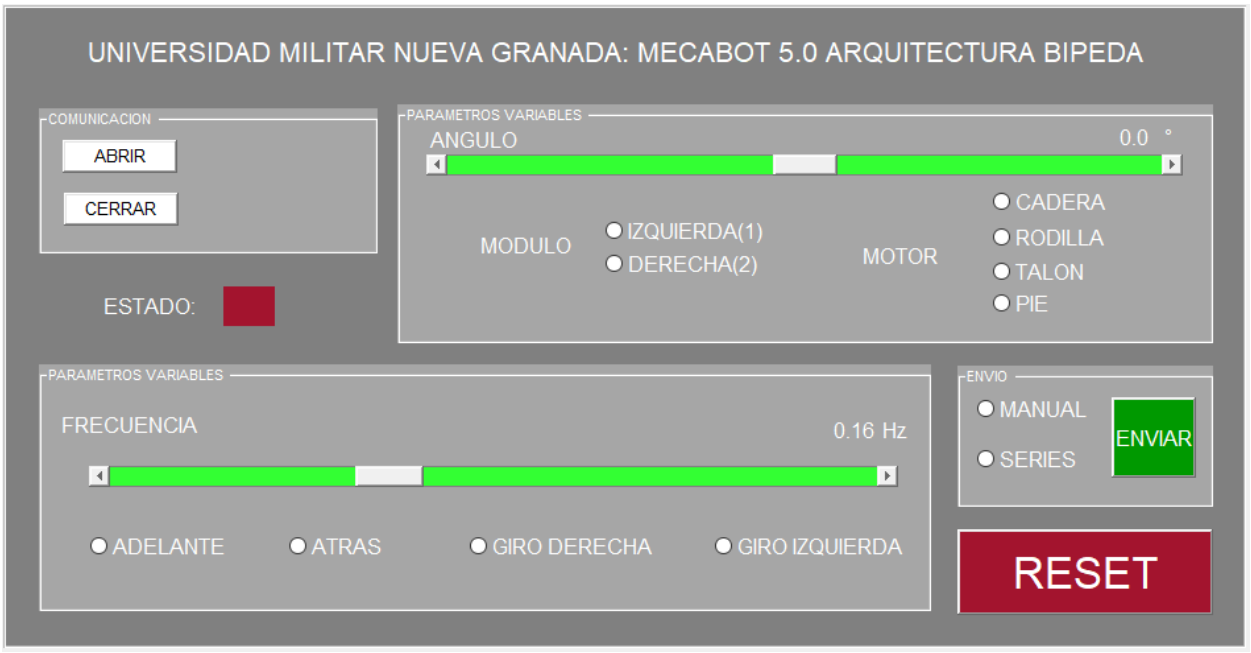


Figura 6-1 Interfaz de usuario en MATLAB®

Tomado de: elaboración propia

El botón reset sincroniza los módulos relacionados a la estructura, y una alerta visual de estado la cual cambia a verde, amarillo y rojo (abierto, espera y cerrado) , indicando el estado del puerto.[16]

6.2. Lógica del programa en la teensy

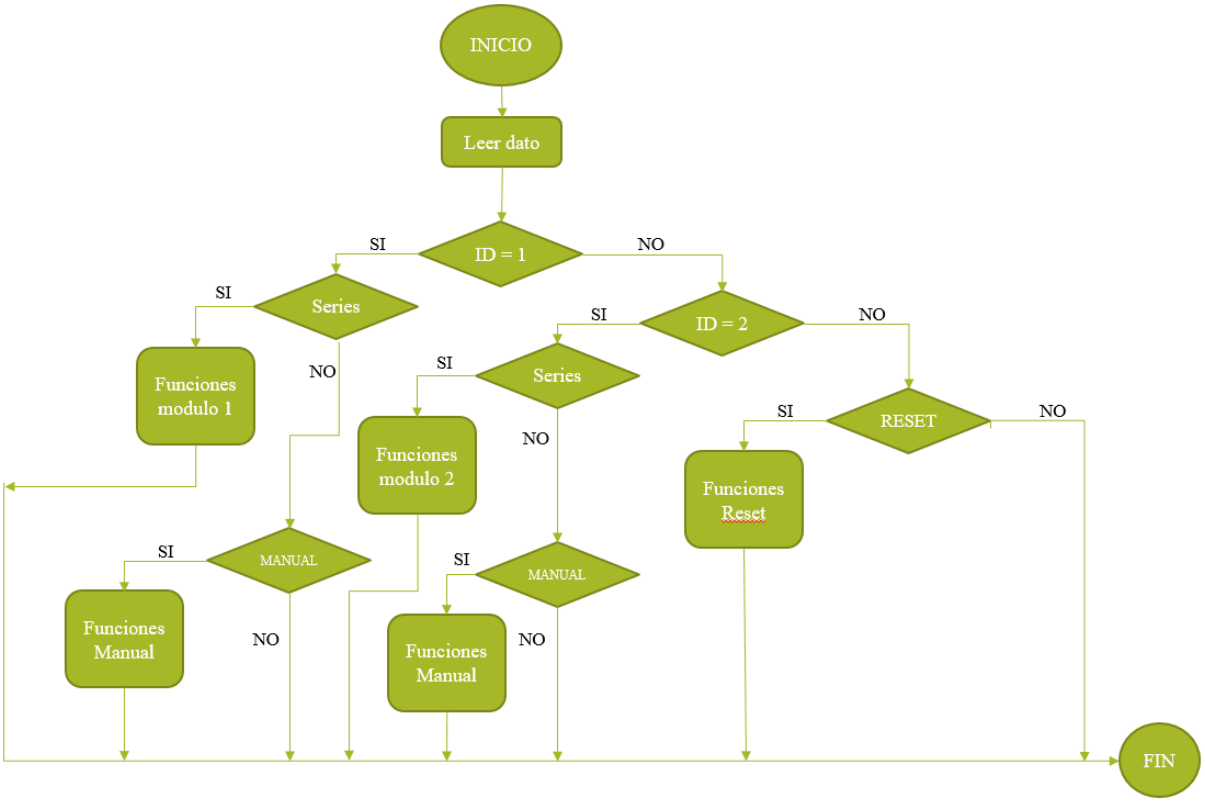


Figura 6-2 Diagrama de flujo

Tomado de: elaboración propia

6.3. Implementación movimientos lineales

Para la realización de las pruebas se tuvieron en cuenta diferentes superficies para observar el comportamiento del sistema en diferentes irregularidades del terreno. Como resultado para las pruebas en una superficie lisa con baldosas, los resultaron, mostraron una respuesta casi lineal del sistema, pero la arquitectura bípeda no posee fuerza suficiente para desplazarse con una frecuencia

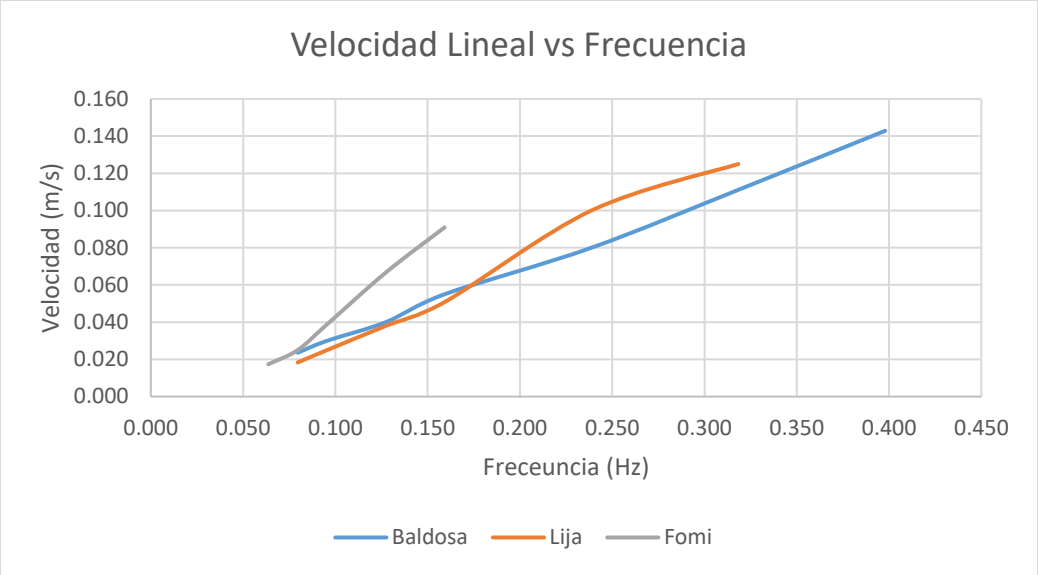


Figura 6-3 Resultados de velocidad lineal en superficies controladas

Tomado de: elaboración propia

Para los resultados en fomi, debido a la superficie blanda, las ruedas de la caminadora, presentaban una fricción muy alta, por lo tanto, no se pudo realizar pruebas con frecuencias altas en esta superficie, las pruebas realizadas en esta superficie se llevaron a cabo entre frecuencias de 60 mHz y 160 mHz.

Los resultados en lija, tienen una tendencia muy similar a la baldosa, pero por la fricción presentada, genera una velocidad ligeramente mayor, pero no puede llegar a la frecuencia de 400 mHz, que se puede alcanzar en la baldosa.

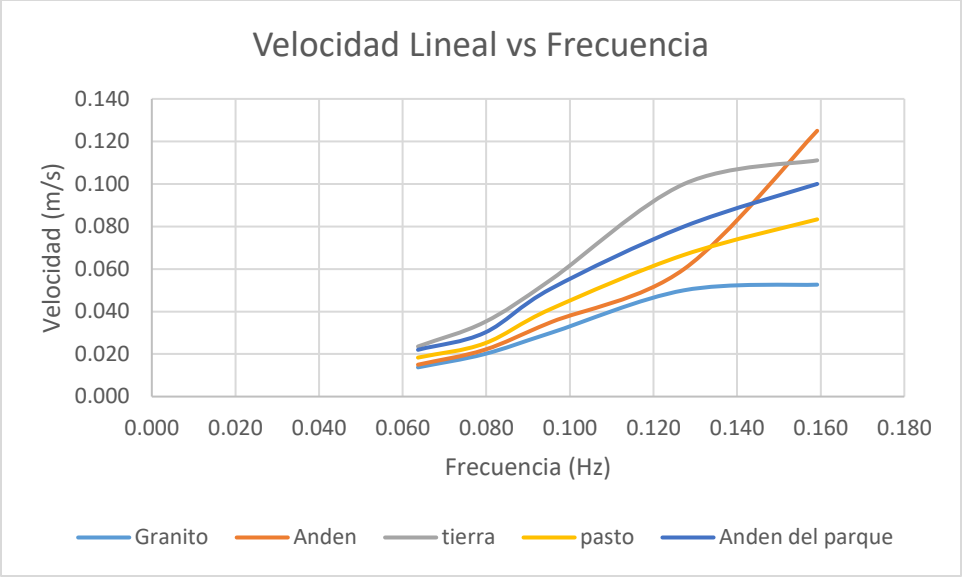


Figura 6-4 Resultados de velocidad lineal en superficies exteriores

Tomado de: elaboración propia

Para las pruebas exteriores, se tuvieron en cuenta diferentes superficies, como lo son, el pasto, anden, tierra, granito y anden de parque, se puede observar que en la tierra presenta una velocidad mayor, pero a frecuencias altas en el andén, la velocidad aumenta, no se realización pruebas con una frecuencia mayor debido a que se necesitaba más torque en los motores.

6.4.Implementación movimientos rotacionales

Con los resultados de la velocidad angular del sistema se encuentra otra respuesta que es una curva, pero al igual que el caso anterior, para frecuencias mayor a 0.4 Hz, entonces los datos se tomaron hasta esa frecuencia mencionada.

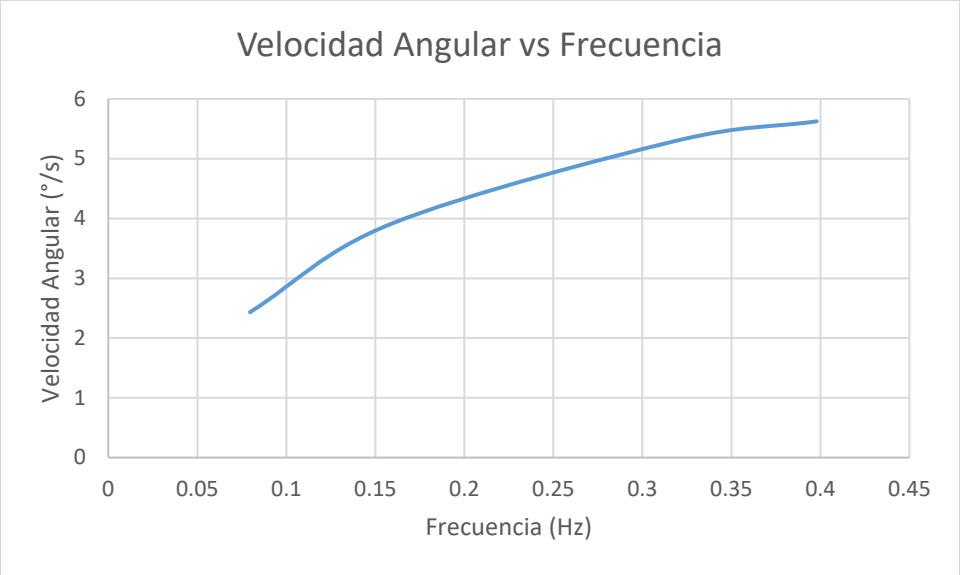


Figura 6-5 Resultados de velocidad Angular en el piso del laboratorio

Tomado de: elaboración propia

Se realizaron pruebas en otras superficies, pero por el torque de los motores, el sistema no podía girar, entonces solo se tiene en cuenta los resultados en baldosa, esto debido a que es una superficie uniforme y sin mucha fricción, a comparación de las otras superficies.

7. Conclusiones y recomendaciones a trabajos futuros

De acuerdo con las simulaciones hechas en el entorno de simulación webots, se diseña un controlador basado en generadores sinusoidales, el cual cuanta con funciones como avance, retroceso, giro a la izquierda y giro a la derecha para la arquitectura bípeda implementada con el Mecabot 5.0

Con las pruebas realizadas en el entorno virtual se observa que el comportamiento de la velocidad lineal y angular del sistema están relacionadas con la frecuencia del sistema, esto de una manera proporcional.

La arquitectura bípeda logra desplazarse en el movimiento de avance en un gran número de superficies, pero estas superficies tienen que tener una superficie muy uniforme, de lo contrario las ruedas de la caminadora se verán afectadas, haciendo que los motores tengan que hacer más fuerza de la que pueden proveer.

La arquitectura bípeda solo puede girar en una superficie completamente uniforme y sin mucha fricción del contrario el torque de los motores no será suficiente, para desempeñar el movimiento de la manera óptima.

Se comprueba la efectividad de los generadores sinusoidales en el control de los módulos 5.0, como versiones simplificadas, para esta tesis tres generadores, que aseguran un movimiento más suave comparados con los de la tabla de control.

De acuerdo con la investigación realizada de esta tesis, se sugiere los siguientes posibles trabajos futuros.

El desarrollo posterior de una cadera, la cual permita que la arquitectura bípeda, no necesite de la caminadora para efectuar sus movimientos, y un control de equilibrio para tener una locomoción mucho más eficiente.

La implementación de baterías mucho más capaces para alimentar a los motores en su máximo torque sin ningún problema, y el cambio de las aletas plásticas de los servomotores por unas metálicas, para evitar el daño prematuro de estas.

La posibilidad de continuar con el desarrollo de la caminata humana, para superar otros obstáculos como lo son, subir y bajar escaleras, trotar y correr, esto con el fin de tener de que la arquitectura bípeda puede ser muy versátil.

REFERENCIAS

[1] «Definicion», *ROBOTICA*, 01-nov-2006. .

[2] «¿Qué es Robot? - Su Definición, Concepto y Significado». .

[3] «Robot, cyborg y androide, qué son y cuáles son sus diferencias». [En línea]. Disponible en: https://hipertextual.com/2014/07/robot-cyborg-androide. [Accedido: 22-ago-2018].

[4] Javier González Jiménez, «Estimación de la Posición de un Robot Móvil». 2015.

[5] E. Calle, I. Ávila, J. Zambrano, «Diseño e Implementación de un Robot Móvil Cuadrúpedo». oct-2007.

[6] Galarza Palma Pablo, «Robot Bípedo-UIDE». 2014.

[7] Miguel Albero, Francisco Blanes, Gines Benet, Jose Simó, Pascual Perez, «YABIRO: Prototipo De Robot Bípedo Autónomo». 2003.

[8] HONDA, «Honda Robotics», 2000. .

[9] SONY, «Robot». .

[10] Kenji KANEKO, «Humanoid Robot HRP-2». 2004.

[11] Andrea Alejandra Vergara Pulgar, «Diseño Y Fabricación De Robots Modulares Blandos». 2015.

[12] M. C. Berdugo, «Simulación e Implementación de una Configuración de Robot Hexápodo Utilizando Sistemas de Robótica Modular para Evaluar su Locomoción», Universidad Militar Nueva Granada, 2017.

[13] R. Montaña, O. Gerardo, H. Erasso, y C. Andrés, «Diseño y simulación de un robot modular reconfigurable», Universidad Militar Nueva Granada, 2013.

[14] W. A. M. Rubiano y R. A. C. Estepa, «Simulación e Implementación de Sistema Robótico en Arquitectura Tipo Rueda utilizando Robotica Modular», Universidad Militar Nueva Granada, 2017.

[15] L. Guzmán, P. Natalia, S. Galvis, y L. Beatriz, «Simulación e Implementación de Movimientos para Sistema Robótico Modular Considerando Diferentes Configuraciones», Universidad Militar Nueva Granada, 2016.

[16] Vanessa Cruz Carbonell, «Simulación E Implementación De Arquitectura Cuadrúpeda Utilizando Sistema Robótico Modular Mecabot», Universidad Militar Nueva Granada, 2018.

[17] Leonardo E., Contreras Bravo1, y Luís F. Vargas, «Generación de modelos de caminata bípeda a través de diversas técnicas de modelamiento». .

[18] L. E. Contreras, J. A. Tristancho y L. F. Vargas, «Análisis Biomecánico de Marcha Humana a Través de Técnicas de Modelaje». 2012.

[19] Basilio Nikolas Tamburrino Cabrera, «Diseño Y Construcción De Una Pierna Exoesquelética Para La Asistencia De La Marcha», Universidad De Chile.

[20] F. Hernández, «Diseño y construcción de prototipo neumático de prótesis de pierna humana». 2008.

[21] Juan Gonzales, «Robóticamodular Y Locomoción: Aplicación A Robots Ápodos». 2008.

[22] «Webots: robot simulator». .

[23] «Webots documentation: Glossary». .

[24] «Que es Xbee». .

[25] «XBee 2mW Wire Antenna - Series 2 (ZigBee Mesh) - WRL-10414 - SparkFun Electronics». .

[26] «Videos». .

[27] «Teensy 3.2». .
[28] «Poder 1810MG metal Gear HD sin núcleo Digital Servo 3,9 kg / 16g / .13sec». .
[29] «Servo giro limitado HD-1501MG 17.0 kg-cm». .
[30] «Pololu - Micro Maestro 6-Channel USB Servo Controller (Assembled)». .
[31] «Pololu 5V, 2.5A Step-Down Voltage Regulator D24V22F5». .
[32] «Regulador Switching BEC 5V / 6V - 5A». .
[33] «Lithium Ion 2S 7.4V 3A Over Charge-Discharge Protection PCB». .
[34] «Aliexpress.com: Comprar 1 unids 2 s 3A 7.4 V/8.4 V 18650 batería de litio cargador Junta carga de la batería del li ion BMS sobre carga descarga módulo de protección de li-ion charger board fiable proveedores en EC Buying Ali Store». .

ANEXOS

Anexo A Tablas de recolección de pruebas simuladas en movimiento de avance

Frecuencia(Hz)	Velocidad(m/s)
0.159	0.026
0.318	0.045
0.477	0.053
0.637	0.063
0.796	0.100

Tabla Anexo 1 Relación frecuencia y velocidad lineal

Anexo B Tablas de recolección de pruebas simuladas en movimiento de giro

Frecuencia(Hz)	Velocidad(°/s)
0.159	5.625
0.318	11.250
0.477	15.000
0.637	18.000
0.796	22.500

Tabla Anexo 2 Relación frecuencia y velocidad angular

Anexo C Pseudo código de la teensy 3.2

```
#include <PololuMaestro.h>
MiniMaestro maestro(Serial2);

int ID = 2;

IntervalTimer Timer;
volatile float Time = 0;
volatile float Inicio = 0;

float EnvioC = 0;
float EnvioR = 0;
float EnvioT = 0;
float EnvioG = 0;
float Frecuency = 1;
float Angle = 0;
```

```
float var_inicio1;
float var_fin1;
float var_inicio2;
float var_fin2;
float inicio1 = 0.6;
float fin1 = 2.5;
float inicio2 = 5.3;
float fin2 = 5.5;
float limite_s = 45;
float limite_i = -45;
float m1;
float m2;
float b1;
float b2;

char bit0 = 0;
char bit1 = 0;
char bit2 = 0;
char bit3 = 0;
char bit4 = 0;
char bit5 = 0;

float sen_M1 = 0;
float sen_M2 = 0;
float sen_M3 = 0;
float sen_M4 = 0;

float valorBit1 = 0;
float valorBit2 = 0;
float valorBit3 = 0;
float valorBit4 = 0;
float valorBit5 = 0;

#define RESTART_ADDR       0xE000ED0C
#define READ_RESTART()     (*(volatile uint32_t *)RESTART_ADDR)
#define WRITE_RESTART(val) ((*(volatile uint32_t *)RESTART_ADDR) = (val))

float DC = 3.1416*3.076923077 + 0.153846154;

float AC[5] = {25.77,113.1,108,2.721,1.641};
float BC[5] = {1,0.01696,0.01756,2,3};
float CCZ[5] = {-1.224,-2.605,0.4996,1.588,3.04};
float CCD[5] = {(CCZ[0]+(DC*BC[0]/3.1416)), (CCZ[1]+(DC*BC[1]/3.1416)), (CCZ[2]+(DC*BC[2]/3.1416)),
(CCZ[3]+(DC*BC[3]/3.1416))};

float DR = DC;
float AR[5] = {44.4200,   16.7200,   13.6500,   28.1700,    3.6080};
float BR[5] = {0.0099,1,2,0.0125,3};
float CRZ[5] = { 0.9204,   -2.9230,   -0.7221,    3.8900,   -0.0606};
float CRD[5] = {(CRZ[0]+(DR*BR[0]/3.1416)), (CRZ[1]+(DR*BR[1]/3.1416)), (CRZ[2]+(DR*BR[2]/3.1416)),
(CRZ[3]+(DR*BR[3]/3.1416)), (CRZ[4]+(DR*BR[4]/3.1416))};


float DT = DC;
float AT[5] = {3.552,    7.948,    1.658,    0.795,   6.482};
float BT[5] = {2,0.01574,3,4,0.01755};
```

```
float CTZ[5] = {0.2696,    3.698 ,  0.05547,    0.1069,  0.4405};
float CTD[5] = {(CTZ[0]+(DT*BT[0]/3.1416)), (CTZ[1]+(DT*BT[1]/3.1416)), (CTZ[2]+(DT*BT[2]/3.1416)),
(CTZ[3]+(DT*BT[3]/3.1416)), (CTZ[4]+(DT*BT[4]/3.1416))};

int contador=0;

void setup() {

  Timer.begin(Timer_C, 1000);
  Serial1.begin(115200);
  Serial2.begin(115200);

}

void loop() {
  for(int k=0;k<6;k++){
    if(Serial1.available()>0  && contador==0){
      bit0 = Serial1.read();
      contador = 1;
    }
    if(Serial1.available()>0  && contador==1){
      bit1 = Serial1.read();
      contador = 2;
    }
    if(Serial1.available()>0  && contador==2){
      bit2 = Serial1.read();
      contador = 3;
    }
    if(Serial1.available()>0  && contador==3){
      bit3 = Serial1.read();
      contador = 4;
    }
    if(Serial1.available()>0  && contador==4){
      bit4 = Serial1.read();
      contador = 5;
    }
    if(Serial1.available()>0  && contador==5){
      bit5 = Serial1.read();
      contador = 0;
    }
  }
  if(bit3=='+'){
    valorBit3 = 1;
  }
  if(bit3=='-'){
    valorBit3 = -1;
  }
  valorBit1 = int(bit1) - 48;
  valorBit2 = int(bit2) - 48;
  valorBit4 = int(bit4) - 48;
  valorBit5 = int(bit5) - 48;
  if(bit0 == 'R'){
    if(bit3 == 'S'){
      Timer.begin(Timer_C, 1000);
      bit0 = 0;
    }else{
```

```
      WRITE_RESTART(0x5FA0004);
      bit0 = 0;
     }
     Inicio = 0;
   }
   if (ID == 1){
     if(bit0=='S'){

      Frecuency = valorBit4 + valorBit5 * 0.1;
      Inicio = 1;

      if(bit3=='U'){
        AVANCE_MODULO1(Frecuency);
      }
      if(bit3=='D'){
        RETROCESO_MODULO1(Frecuency);
      }
      if(bit3=='L'){
        DERECHA_MODULO1( Frecuency);
      }
      if(bit3=='R'){
        IZQUIERDA_MODULO1( Frecuency);
      }
     }
   }
   if (ID == 2){
     if(bit0=='S'){

      Frecuency = valorBit4 + valorBit5 * 0.1;
      Inicio = 1;

      if(bit3=='U'){
        AVANCE_MODULO2(Frecuency);
      }
      if(bit3=='D'){
        RETROCESO_MODULO2(Frecuency);
      }
      if(bit3=='L'){
        DERECHA_MODULO2( Frecuency);
      }
      if(bit3=='R'){
        IZQUERDA_MODULO2( Frecuency);
      }
     }
   }
   if(bit0=='M'){
     if (valorBit1 == ID){
      Angle = valorBit3 * (valorBit4*10 + valorBit5);
      MANUAL(Angle,valorBit2);
      Inicio = 1;
     }
   }
}

void MANUAL(float Angle,float valorBit2) {
  if (valorBit2 == 1){
```

```
   EnvioC = (11.806 * Angle) + 1398.2;
   EnvioC = EnvioC * 4;
   maestro.setTarget(0, EnvioC);
  }
  if (valorBit2 == 2){
   EnvioR = (11.806 * Angle) + 1398.2;
   EnvioR = EnvioR * 4;
   maestro.setTarget(1, EnvioR);
  }
  if (valorBit2 == 3){
   EnvioT = (11.806 * Angle) + 1398.2;
   EnvioT = EnvioT * 4;
   maestro.setTarget(2, EnvioT);
  }
  if (valorBit2 == 4){
   EnvioG = (11.806 * Angle) + 1398.2;
   EnvioG = EnvioG * 4;
   maestro.setTarget(3, EnvioG);
  }
}

void AVANCE_MODULO1(float Frecuency){

  //MOTOR 1

  sen_M1 = AC[0]*sin(BC[0]*Time*Frecuency+CCZ[0]) + AC[1]*sin(BC[1]*Time*Frecuency+CCZ[1]) +
AC[2]*sin(BC[2]*Time*Frecuency+CCZ[2]) + AC[3]*sin(BC[3]*Time*Frecuency+CCZ[3]) +
AC[4]*sin(BC[4]*Time*Frecuency+CCZ[4]);

  //MOTOR 2

  sen_M2 = AR[0]*sin(BR[0]*Time*Frecuency+CRZ[0]) + AR[1]*sin(BR[1]*Time*Frecuency+CRZ[1]) +
AR[2]*sin(BR[2]*Time*Frecuency+CRZ[2]) + AR[3]*sin(BR[3]*Time*Frecuency+CRZ[3]) +
AR[4]*sin(BR[4]*Time*Frecuency+CRZ[4]);

  //MOTOR 3

  sen_M3 = AT[0]*sin(BT[0]*Time*Frecuency+CTZ[0]) + AT[1]*sin(BT[1]*Time*Frecuency+CTZ[1]) +
AT[2]*sin(BT[2]*Time*Frecuency+CTZ[2]) + AT[3]*sin(BT[3]*Time*Frecuency+CTZ[3]) +
AT[4]*sin(BT[4]*Time*Frecuency+CTZ[4]);

  EnvioC = (11.806 * sen_M1) + 1398.2;
  EnvioC = EnvioC * 4;
  maestro.setTarget(0, EnvioC);
  EnvioR = (11.806 * sen_M2) + 1398.2;
  EnvioR = EnvioR * 4;
  maestro.setTarget(1, EnvioR);
  EnvioT = (11.806 * sen_M3) + 1398.2;
  EnvioT = EnvioT * 4;
  maestro.setTarget(2, EnvioT);
  EnvioG = (11.806 * limite_s) + 1398.2;
  EnvioG = EnvioG * 4;
  maestro.setTarget(3, EnvioG);

}
```

```
void AVANCE_MODULO2(float Frecuency){

  //MOTOR 1

  sen_M1 = AC[0]*sin(BC[0]*Time*Frecuency+CCD[0]) + AC[1]*sin(BC[1]*Time*Frecuency+CCD[1]) +
AC[2]*sin(BC[2]*Time*Frecuency+CCD[2]) + AC[3]*sin(BC[3]*Time*Frecuency+CCD[3]) +
AC[4]*sin(BC[4]*Time*Frecuency+CCD[4]);

  //MOTOR 2

  sen_M2 = AR[0]*sin(BR[0]*Time*Frecuency+CRD[0]) + AR[1]*sin(BR[1]*Time*Frecuency+CRD[1]) +
AR[2]*sin(BR[2]*Time*Frecuency+CRD[2]) + AR[3]*sin(BR[3]*Time*Frecuency+CRD[3]) +
AR[4]*sin(BR[4]*Time*Frecuency+CRD[4]);

  //MOTOR 3

  sen_M3 = AT[0]*sin(BT[0]*Time*Frecuency+CTD[0]) + AT[1]*sin(BT[1]*Time*Frecuency+CTD[1]) +
AT[2]*sin(BT[2]*Time*Frecuency+CTD[2]) + AT[3]*sin(BT[3]*Time*Frecuency+CTD[3]) +
AT[4]*sin(BT[4]*Time*Frecuency+CTD[4]);

  EnvioC = (11.806 * sen_M1) + 1398.2;
  EnvioC = EnvioC * 4;
  maestro.setTarget(0, EnvioC);
  EnvioR = (11.806 * sen_M2) + 1398.2;
  EnvioR = EnvioR * 4;
  maestro.setTarget(1, EnvioR);
  EnvioT = (11.806 * sen_M3) + 1398.2;
  EnvioT = EnvioT * 4;
  maestro.setTarget(2, EnvioT);
  EnvioG = (11.806 * limite_s) + 1398.2;
  EnvioG = EnvioG * 4;
  maestro.setTarget(3, EnvioG);

}
void RETROCESO_MODULO1(float Frecuency){

  //MOTOR 1

  sen_M1 = AC[0]*sin(BC[0]*Time*Frecuency+CCZ[0]) + AC[1]*sin(BC[1]*Time*Frecuency+CCZ[1]) +
AC[2]*sin(BC[2]*Time*Frecuency+CCZ[2]) + AC[3]*sin(BC[3]*Time*Frecuency+CCZ[3]) +
AC[4]*sin(BC[4]*Time*Frecuency+CCZ[4]);
  sen_M1 = -sen_M1;

  //MOTOR 2

  sen_M2 = AR[0]*sin(BR[0]*Time*Frecuency+CRZ[0]) + AR[1]*sin(BR[1]*Time*Frecuency+CRZ[1]) +
AR[2]*sin(BR[2]*Time*Frecuency+CRZ[2]) + AR[3]*sin(BR[3]*Time*Frecuency+CRZ[3]) +
AR[4]*sin(BR[4]*Time*Frecuency+CRZ[4]);
  sen_M2 = -sen_M2;

  //MOTOR 3

  sen_M3 = AT[0]*sin(BT[0]*Time*Frecuency+CTZ[0]) + AT[1]*sin(BT[1]*Time*Frecuency+CTZ[1]) +
AT[2]*sin(BT[2]*Time*Frecuency+CTZ[2]) + AT[3]*sin(BT[3]*Time*Frecuency+CTZ[3]) +
AT[4]*sin(BT[4]*Time*Frecuency+CTZ[4]);
  sen_M3 = -sen_M3;
```

```
  EnvioC = (11.806 * sen_M1) + 1398.2;
  EnvioC = EnvioC * 4;
  maestro.setTarget(0, EnvioC);
  EnvioR = (11.806 * sen_M2) + 1398.2;
  EnvioR = EnvioR * 4;
  maestro.setTarget(1, EnvioR);
  EnvioT = (11.806 * sen_M3) + 1398.2;
  EnvioT = EnvioT * 4;
  maestro.setTarget(2, EnvioT);
  EnvioG = (11.806 * limite_s) + 1398.2;
  EnvioG = EnvioG * 4;
  maestro.setTarget(3, EnvioG);

}
void RETROCESO_MODULO2(float Frecuency){

  //MOTOR 1

  sen_M1 = AC[0]*sin(BC[0]*Time*Frecuency+CCD[0]) + AC[1]*sin(BC[1]*Time*Frecuency+CCD[1]) +
AC[2]*sin(BC[2]*Time*Frecuency+CCD[2]) + AC[3]*sin(BC[3]*Time*Frecuency+CCD[3]) +
AC[4]*sin(BC[4]*Time*Frecuency+CCD[4]);
  sen_M1 = -sen_M1;

  //MOTOR 2

  sen_M2 = AR[0]*sin(BR[0]*Time*Frecuency+CRD[0]) + AR[1]*sin(BR[1]*Time*Frecuency+CRD[1]) +
AR[2]*sin(BR[2]*Time*Frecuency+CRD[2]) + AR[3]*sin(BR[3]*Time*Frecuency+CRD[3]) +
AR[4]*sin(BR[4]*Time*Frecuency+CRD[4]);
  sen_M2 = -sen_M2;

  //MOTOR 3

  sen_M3 = AT[0]*sin(BT[0]*Time*Frecuency+CTD[0]) + AT[1]*sin(BT[1]*Time*Frecuency+CTD[1]) +
AT[2]*sin(BT[2]*Time*Frecuency+CTD[2]) + AT[3]*sin(BT[3]*Time*Frecuency+CTD[3]) +
AT[4]*sin(BT[4]*Time*Frecuency+CTD[4]);
  sen_M3 = -sen_M3;

  EnvioC = (11.806 * sen_M1) + 1398.2;
  EnvioC = EnvioC * 4;
  maestro.setTarget(0, EnvioC);
  EnvioR = (11.806 * sen_M2) + 1398.2;
  EnvioR = EnvioR * 4;
  maestro.setTarget(1, EnvioR);
  EnvioT = (11.806 * sen_M3) + 1398.2;
  EnvioT = EnvioT * 4;
  maestro.setTarget(2, EnvioT);
  EnvioG = (11.806 * limite_s) + 1398.2;
  EnvioG = EnvioG * 4;
  maestro.setTarget(3, EnvioG);

}
void DERECHA_MODULO1(float Frecuency){

  limite_s = 45;
  limite_i = -45;
```

```
 inicio1 = 0.6;
 fin1 = 2.5;
 inicio2 = 5.3;
 fin2 = 5.5;

 var_inicio1 = (inicio1/Frecuency);
 var_fin1 = (fin1/Frecuency);
 var_inicio2 = (inicio2/Frecuency);
 var_fin2 = (fin2/Frecuency);
 m1 = (limite_i - limite_s) / (var_fin1 - var_inicio1);
 b1 = limite_s - m1*var_inicio1;
 m2 = (limite_s - limite_i) / (var_fin2 - var_inicio2);
 b2 = limite_i - m2*var_inicio2;

 //MOTOR 1

 sen_M1 = AC[0]*sin(BC[0]*Time*Frecuency+CCZ[0]) + AC[1]*sin(BC[1]*Time*Frecuency+CCZ[1]) +
AC[2]*sin(BC[2]*Time*Frecuency+CCZ[2]) + AC[3]*sin(BC[3]*Time*Frecuency+CCZ[3]) +
AC[4]*sin(BC[4]*Time*Frecuency+CCZ[4]);

 //MOTOR 2

 sen_M2 = AR[0]*sin(BR[0]*Time*Frecuency+CRZ[0]) + AR[1]*sin(BR[1]*Time*Frecuency+CRZ[1]) +
AR[2]*sin(BR[2]*Time*Frecuency+CRZ[2]) + AR[3]*sin(BR[3]*Time*Frecuency+CRZ[3]) +
AR[4]*sin(BR[4]*Time*Frecuency+CRZ[4]);

 //MOTOR 3

 sen_M3 = AT[0]*sin(BT[0]*Time*Frecuency+CTZ[0]) + AT[1]*sin(BT[1]*Time*Frecuency+CTZ[1]) +
AT[2]*sin(BT[2]*Time*Frecuency+CTZ[2]) + AT[3]*sin(BT[3]*Time*Frecuency+CTZ[3]) +
AT[4]*sin(BT[4]*Time*Frecuency+CTZ[4]);

 //MOTOR 4

 sen_M1 = -sen_M1;
 sen_M2 = -sen_M2;
 sen_M3 = -sen_M3;

 if (Time >= var_inicio1 && Time<=var_fin1){
    sen_M4 = m1*Time + b1;
 }else{
    if (Time >= (var_inicio2) && Time <= (var_fin2)){
       sen_M4 = m2*Time + b2;
    }else{
       if (Time > var_fin1 && Time < (var_inicio2)){
          sen_M4 = limite_i;
       }else{
          sen_M4 = limite_s;
       }
    }
 }

 EnvioC = (11.806 * sen_M1) + 1398.2;
 EnvioC = EnvioC * 4;
 maestro.setTarget(0, EnvioC);
 EnvioR = (11.806 * sen_M2) + 1398.2;
```

```
  EnvioR = EnvioR * 4;
  maestro.setTarget(1, EnvioR);
  EnvioT = (11.806 * sen_M3) + 1398.2;
  EnvioT = EnvioT * 4;
  maestro.setTarget(2, EnvioT);
  EnvioG = (11.806 * sen_M4) + 1398.2;
  EnvioG = EnvioG * 4;
  maestro.setTarget(3, EnvioG);

}
void DERECHA_MODULO2(float Frecuency){

  limite_s = 45;
  limite_i = -45;
  inicio1 = 0.6;
  fin1 = 2.5;
  inicio2 = 5.3;
  fin2 = 5.5;

  var_inicio1 = (inicio1/Frecuency);
  var_fin1 = (fin1/Frecuency);
  var_inicio2 = (inicio2/Frecuency);
  var_fin2 = (fin2/Frecuency);
  m1 = (limite_i - limite_s) / (var_fin1 - var_inicio1);
  b1 = limite_s - m1*var_inicio1;
  m2 = (limite_s - limite_i) / (var_fin2 - var_inicio2);
  b2 = limite_i - m2*var_inicio2;

  //MOTOR 1

  sen_M1 = AC[0]*sin(BC[0]*Time*Frecuency+CCZ[0]) + AC[1]*sin(BC[1]*Time*Frecuency+CCZ[1]) +
AC[2]*sin(BC[2]*Time*Frecuency+CCZ[2]) + AC[3]*sin(BC[3]*Time*Frecuency+CCZ[3]) +
AC[4]*sin(BC[4]*Time*Frecuency+CCZ[4]);

  //MOTOR 2

  sen_M2 = AR[0]*sin(BR[0]*Time*Frecuency+CRZ[0]) + AR[1]*sin(BR[1]*Time*Frecuency+CRZ[1]) +
AR[2]*sin(BR[2]*Time*Frecuency+CRZ[2]) + AR[3]*sin(BR[3]*Time*Frecuency+CRZ[3]) +
AR[4]*sin(BR[4]*Time*Frecuency+CRZ[4]);

  //MOTOR 3

  sen_M3 = AT[0]*sin(BT[0]*Time*Frecuency+CTZ[0]) + AT[1]*sin(BT[1]*Time*Frecuency+CTZ[1]) +
AT[2]*sin(BT[2]*Time*Frecuency+CTZ[2]) + AT[3]*sin(BT[3]*Time*Frecuency+CTZ[3]) +
AT[4]*sin(BT[4]*Time*Frecuency+CTZ[4]);

  //MOTOR 4

  if (Time >= var_inicio1 && Time<=var_fin1){
     sen_M4 = m1*Time + b1;
  }else{
     if (Time >= (var_inicio2) && Time <= (var_fin2)){
        sen_M4 = m2*Time + b2;
     }else{
        if (Time > var_fin1 && Time < (var_inicio2)){
           sen_M4 = limite_i;
```

```
        }else{
           sen_M4 = limite_s;
        }
     }
  }

  EnvioC = (11.806 * sen_M1) + 1398.2;
  EnvioC = EnvioC * 4;
  maestro.setTarget(0, EnvioC);
  EnvioR = (11.806 * sen_M2) + 1398.2;
  EnvioR = EnvioR * 4;
  maestro.setTarget(1, EnvioR);
  EnvioT = (11.806 * sen_M3) + 1398.2;
  EnvioT = EnvioT * 4;
  maestro.setTarget(2, EnvioT);
  EnvioG = (11.806 * sen_M4) + 1398.2;
  EnvioG = EnvioG * 4;
  maestro.setTarget(3, EnvioG);

}

void IZQUIERDA_MODULO1(float Frecuency ){

  limite_s = -45;
  limite_i = 45;
  inicio1 = 0.6 + 3.1416;
  fin1 = 2.5 + 3.1416;
  inicio2 = 5.3 + 3.1416;
  fin2 = 5.5 + 3.1416;

  var_inicio1 = (inicio1/Frecuency);
  var_fin1 = (fin1/Frecuency);
  var_inicio2 = (inicio2/Frecuency);
  var_fin2 = (fin2/Frecuency);
  m1 = (limite_i - limite_s) / (var_fin1 - var_inicio1);
  b1 = limite_s - m1*var_inicio1;
  m2 = (limite_s - limite_i) / (var_fin2 - var_inicio2);
  b2 = limite_i - m2*var_inicio2;

  //MOTOR 1

  sen_M1 = AC[0]*sin(BC[0]*Time*Frecuency+CCD[0]) + AC[1]*sin(BC[1]*Time*Frecuency+CCD[1]) +
AC[2]*sin(BC[2]*Time*Frecuency+CCD[2]) + AC[3]*sin(BC[3]*Time*Frecuency+CCD[3]) +
AC[4]*sin(BC[4]*Time*Frecuency+CCD[4]);

  //MOTOR 2

  sen_M2 = AR[0]*sin(BR[0]*Time*Frecuency+CRD[0]) + AR[1]*sin(BR[1]*Time*Frecuency+CRD[1]) +
AR[2]*sin(BR[2]*Time*Frecuency+CRD[2]) + AR[3]*sin(BR[3]*Time*Frecuency+CRD[3]) +
AR[4]*sin(BR[4]*Time*Frecuency+CRD[4]);

  //MOTOR 3

  sen_M3 = AT[0]*sin(BT[0]*Time*Frecuency+CTD[0]) + AT[1]*sin(BT[1]*Time*Frecuency+CTD[1]) +
AT[2]*sin(BT[2]*Time*Frecuency+CTD[2]) + AT[3]*sin(BT[3]*Time*Frecuency+CTD[3]) +
AT[4]*sin(BT[4]*Time*Frecuency+CTD[4]);
```

```
  //MOTOR 4
  if (Time >= var_inicio1 && Time<=var_fin1){
     sen_M4 = m1*Time + b1;
  }else{
     if (Time >= (var_inicio2) && Time <= (var_fin2)){
        sen_M4 = m2*Time + b2;
     }else{
        if (Time > var_fin1 && Time < (var_inicio2)){
           sen_M4 = limite_i;
        }else{
           sen_M4 = limite_s;
        }
     }
  }

  EnvioC = (11.806 * sen_M1) + 1398.2;
  EnvioC = EnvioC * 4;
  maestro.setTarget(0, EnvioC);
  EnvioR = (11.806 * sen_M2) + 1398.2;
  EnvioR = EnvioR * 4;
  maestro.setTarget(1, EnvioR);
  EnvioT = (11.806 * sen_M3) + 1398.2;
  EnvioT = EnvioT * 4;
  maestro.setTarget(2, EnvioT);
  EnvioG = (11.806 * sen_M4) + 1398.2;
  EnvioG = EnvioG * 4;
  maestro.setTarget(3, EnvioG);

}
void IZQUIERDA_MODULO2(float Frecuency){

  limite_s = -45;
  limite_i = 45;
  inicio1 = 0.6 + 3.1416;
  fin1 = 2.5 + 3.1416;
  inicio2 = 5.3 + 3.1416;
  fin2 = 5.5 + 3.1416;

  var_inicio1 = (inicio1/Frecuency);
  var_fin1 = (fin1/Frecuency);
  var_inicio2 = (inicio2/Frecuency);
  var_fin2 = (fin2/Frecuency);
  m1 = (limite_i - limite_s) / (var_fin1 - var_inicio1);
  b1 = limite_s - m1*var_inicio1;
  m2 = (limite_s - limite_i) / (var_fin2 - var_inicio2);
  b2 = limite_i - m2*var_inicio2;

  //MOTOR 1

  sen_M1 = AC[0]*sin(BC[0]*Time*Frecuency+CCD[0]) + AC[1]*sin(BC[1]*Time*Frecuency+CCD[1]) +
AC[2]*sin(BC[2]*Time*Frecuency+CCD[2]) + AC[3]*sin(BC[3]*Time*Frecuency+CCD[3]) +
AC[4]*sin(BC[4]*Time*Frecuency+CCD[4]);

  //MOTOR 2
```

```
  sen_M2 = AR[0]*sin(BR[0]*Time*Frecuency+CRD[0]) + AR[1]*sin(BR[1]*Time*Frecuency+CRD[1]) +
AR[2]*sin(BR[2]*Time*Frecuency+CRD[2]) + AR[3]*sin(BR[3]*Time*Frecuency+CRD[3]) +
AR[4]*sin(BR[4]*Time*Frecuency+CRD[4]);

  //MOTOR 3

  sen_M3 = AT[0]*sin(BT[0]*Time*Frecuency+CTD[0]) + AT[1]*sin(BT[1]*Time*Frecuency+CTD[1]) +
AT[2]*sin(BT[2]*Time*Frecuency+CTD[2]) + AT[3]*sin(BT[3]*Time*Frecuency+CTD[3]) +
AT[4]*sin(BT[4]*Time*Frecuency+CTD[4]);

  sen_M1 = -sen_M1;
  sen_M2 = -sen_M2;
  sen_M3 = -sen_M3;

  //MOTOR 4

  if (Time >= var_inicio1 && Time<=var_fin1){
     sen_M4 = m1*Time + b1;
  }else{
     if (Time >= (var_inicio2) && Time <= (var_fin2)){
        sen_M4 = m2*Time + b2;
     }else{
        if (Time > var_fin1 && Time < (var_inicio2)){
           sen_M4 = limite_i;
        }else{
           sen_M4 = limite_s;
        }
     }
  }

  EnvioC = (11.806 * sen_M1) + 1398.2;
  EnvioC = EnvioC * 4;
  maestro.setTarget(0, EnvioC);
  EnvioR = (11.806 * sen_M2) + 1398.2;
  EnvioR = EnvioR * 4;
  maestro.setTarget(1, EnvioR);
  EnvioT = (11.806 * sen_M3) + 1398.2;
  EnvioT = EnvioT * 4;
  maestro.setTarget(2, EnvioT);
  EnvioG = (11.806 * sen_M4) + 1398.2;
  EnvioG = EnvioG * 4;
  maestro.setTarget(3, EnvioG);

}
void Timer_C(){
  if (Inicio == 1){
    Time = Time + 0.001;
    if (Time >= (6.2836/Frecuency)){
      Time = 0;
    }
  }
}
```

Anexo D Tablas de recolección de pruebas en movimiento de avance en el laboratorio

Frecuencia(Hz)	Velocidad(m/s)		
	Baldosa	Lija	Fomi
0.064	x	x	0.017
0.080	0.024	0.018	0.025
0.095	0.030	0.025	0.039
0.127	0.040	0.038	0.067
0.159	0.055	0.051	0.091
0.239	0.080	0.100	x
0.318	0.111	0.125	x
0.398	0.143	x	x

Tabla Anexo 3 Relación frecuencia y velocidad lineal

Anexo E Tablas de recolección de pruebas en movimiento de avance en el exterior

Frecuencia(Hz)	Velocidad (m/s)				
	Granito	Anden	tierra	pasto	Anden del parque
0.064	0.014	0.015	0.024	0.018	0.022
0.080	0.020	0.022	0.035	0.025	0.030
0.095	0.030	0.035	0.055	0.041	0.051
0.127	0.050	0.060	0.100	0.067	0.080
0.159	0.053	0.125	0.111	0.083	0.100

Tabla Anexo 4 Relación frecuencia y velocidad lineal

Anexo F Tablas de recolección de pruebas en movimiento de giro

Frecuencia(Hz)	Velocidad(°/s)
0.159	5.625
0.318	11.250
0.477	15.000
0.637	18.000
0.796	22.500

Tabla Anexo 5 Relación frecuencia y velocidad angular