Simulación e Implementación de Movimientos para Sistema Robótico Modular Considerando Diferentes Configuraciones

Paola Natalia Lancheros Guzmán

Laura Beatriz Sanabria Galvis

Tutor:

Ing. Ricardo Andrés Castillo Estepa, PhD.

Universidad Militar Nueva Granada

Facultad de Ingeniería

Programa de Ingeniería En Mecatrónica

Bogotá

2016

Simulación e Implementación de Movimientos para Sistema Robótico Modular Considerando Diferentes Configuraciones

Paola Natalia Lancheros Guzmán

Laura Beatriz Sanabria Galvis

Tutor:

PhD. Ricardo Andrés Catillo Estepa

Trabajo de Grado para Optar al Título de Ingeniero en Mecatrónica

Universidad Militar Nueva Granada

Facultad de Ingeniería

Programa de Ingeniería en Mecatrónica

Bogotá

2016

Bogotá (14, 10, 2016)

Nota de Aceptación

_____

_____

_____

_____

_____

_____

_____

Firma del presidente del Jurado

_____

Firma de Jurado

_____

Firma de Jurado

Agradecimientos

A nuestros padres y familia por brindarnos su amor, educación, formación personal y apoyo incondicional.

A Ricardo Castillo PhD. y a el Ing. David Gómez, por su tiempo, dedicación y valiosos aportes.

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

pág.

Lista de Tablas

Lista de Figuras

Lista de Anexos

Glosario

A

ABS: Llamado Acrilonitrilo Butadieno Estireno, es un termoplástico amorfo empleado para generar piezas mediante la impresión 3D. Es resistente al impacto, tiene alta tenacidad, es duro y rígido.

Algoritmo: Serie de operaciones organizadas que por medio de un procedimiento lógico dan solución a una problemática.

Amplitud de onda: Distancia que existe entre el punto medio de la onda y el punto más alto o pico.

Ápodos: Referente al grupo compuesto por sistemas robóticos que carecen de extremidades y se desplazan por medio de movimientos corporales.

Archivo Controlador: Referente al algoritmo empleado para la comunicación y generación de movimiento del sistema robótico modular.

Articulación: Unión entre dos componentes, en donde alguno de ellos tiene al menos un grado de libertad (movimiento).

Auto-Configurable: Referente a los sistemas robóticos modulares que mediante algoritmos son capaces de modificar su forma para ajustarse al entorno.

Autonomía: En robótica, es la capacidad de un robot para actuar ante circunstancias no contempladas, sin la intervención de un operario.

C

Cinemática: Rama de la mecánica que estudia el movimiento sin contemplar las causas que lo producen.

Coordinación: Capacidad de un sistema de ejecutar acciones simultáneamente y de forma organizada para lograr un objetivo.

E

Estabilidad: Capacidad de evitar que el robot se caiga en el momento de la marcha. (Gorrostieta & Vargas Soto, 2008).

F

Fase: Diferencia de tiempo transcurrido para que dos ondas sinusoidales alcancen un mismo punto de referencia. Se mide en unidades de ángulo.

Frecuencia: Determina la cantidad de veces que se repite un evento por unidad de tiempo.

G

Grados De Libertad DOF *(Degrees of Freedom)*: Número de movimientos independientes que es capaz de llevar a cabo una articulación.

H

Homogéneo: Referente a las unidades con características idénticas entre ellas.

L

Locomoción: Capacidad de desplazamiento que tiene un sistema.

M

Módulo: Dos segmentos iguales unidos por una articulación (Gonzalez Gómez, 2008).

O

Oscilador: Sistema que genera una onda periódica.

P

Parámetro: Variable perteneciente a una ecuación, cuyo valor es asignable.

S

Simulación: Representación de un sistema real en entorno virtual que permite observar su comportamiento en condiciones ideales.

T

Topología: Disposición en el espacio en la que están organizadas las unidades que conforman un sistema.

Resumen

MECABOT 3.0 es un sistema robótico modular desarrollado en la Universidad Militar Nueva Granada, compuesto por múltiples módulos homogéneos, con capacidad de formar diversas estructuras o configuraciones (oruga, serpiente, hexápodo y rueda) gracias a sus diferentes tipos de conexión, al reordenar sus partes para adaptarse al terreno en el que se desplaza. Este trabajo se enfoca en la locomoción en configuraciones tipo oruga y serpiente, mediante la simulación de MECABOT 3.0 en el software Webots, a partir de la generación de un código controlador basado en un modelo simplificado del control CPG (Central Pattern Generator), representado por generadores sinusoidales de frecuencia fija propuesto por (Gonzalez Gómez, 2008). Igualmente se realiza la construcción de los semi-módulos y mediante el software MATLAB® se realiza una interfaz gráfica que permite la manipulación del sistema. Para las pruebas de simulación e implementación se varían los parámetros del archivo controlador, el tipo de configuración y las conexiones entre los semi-módulos con el fin de comparar los resultados y obtener datos de velocidad como indicador de rendimiento.

Abstract

MECABOT 3.0 is a modular robotic system developed in the Militar Nueva Granada University, it is composed by multiple homogeneous modules, in capacity of forming diverse structures or configurations (caterpillar, snake, hexapod and wheel) due to its different connection types, by reordering its parts in order to adapt itself to the terrain where it moves. This project focuses on the locomotion in caterpillar and snake – type configurations, through MECABOT 3.0 simulation on Webots software, starting from the generation of a controller code based on a simplified control model of CPG (Central Pattern Generator), represented by sine fixed frequency generators proposed by (Gonzalez Gómez, 2008). *Likewise, the construction of the semi-modules is realized and by using software MATLAB® a graphic interface that allows the systems manipulation. For the simulation and implementation tests the controller parameters are varied, the configuration type and the connections between the semi-modules in order to compare and obtain velocity data as performance indicator.*

1 Introducción

La necesidad del ser humano de mejorar su calidad de vida lo ha llevado a indagar mecanismos que le ayuden a ejecutar tareas cotidianas, evitar riesgos y solucionar problemas e imprevistos; una de las soluciones encontradas, es la construcción de dispositivos denominados “Robots”, capaces de interactuar con su alrededor de forma automática. La robótica, rama de la tecnología destinada al estudio del diseño y construcción de los robots (Real Academia Española, 2016), ha ido expandiendo sus campos de acción abarcando aplicaciones desde la medicina hasta las exploraciones espaciales, por medio del desarrollo de robots especializados para estas tareas (Dowling, 1997).

Existen múltiples clasificaciones para un robot; algunas son según, su aplicación, su locomoción, su arquitectura o su posicionamiento; cuando se habla de posicionamiento el robot puede tener su base sujeta a una superficie denominándose robot fijo o industrial; por el contrario, si el robot es capaz de desplazar toda su estructura de un punto a otro dentro de su espacio de trabajo se denomina robot móvil, siendo su desplazamiento una gran ventaja frente a los robots fijos, ya que se puede movilizar hasta el lugar que se requiera.

Dependiendo del entorno en el que se encuentre el robot móvil, se necesita un tipo específico de mecanismo de desplazamiento, bien sea ruedas, piernas, sistemas voladores, nadadores o ápodos, entre otros (Dobra, 2014). Los sistemas robóticos ápodos se emplean para representar el movimiento de animales como la serpiente y la oruga; no cuentan con extremidades y se desplazan únicamente por oscilaciones de su cuerpo ofreciendo ventajas frente a los robots con piernas o ruedas como la estabilidad, la capacidad de adaptación a terrenos, buena tracción, tamaño y redundancia puesto que suelen emplear actuadores de movimientos simples accionados en secuencia (Dowling, 1997). Un diseño apto para la representación de esta redundancia es por medio de los sistemas robóticos modulares. Estos sistemas robóticos están conformados por múltiples módulos o bloques que mediante dispositivos de conexión, permiten la transferencia de fuerzas, torques, sistemas eléctricos y de comunicación; a su vez, pueden modificar la estructura que conforman al unirse para

adaptarse a una nueva condición del entorno o completar una tarea diferente presentando ventajas económicas y funcionales respecto a los robots de estructura fija, ya que al utilizar módulos relativamente sencillos, son fáciles de producir, ensamblar y reemplazar; además, son mucho más versátiles puesto que un único sistema se puede emplear en variedad de operaciones tales como exploración, inspección , medicina, reconocimiento en entornos peligrosos (Rus, Salemi, Shen, & Yim, 2007) (Liedke, Matthias, Winkler, & Wörn, 2013) etc.

El estudio realizado en este proyecto de grado se basa en el módulo MECABOT 3.0 desarrollado en la Universidad Militar Nueva Granada, con el que se conforma un sistema robótico modular en cadena, en el que los MECABOT 3.0 se encuentran conectados uno tras otro mediante una interfaz manual con el fin de lograr la coordinación de los movimientos de los módulos en forma tal que se consiga el desplazamiento deseado. Es por esto que, en este documento, se presenta una serie de simulaciones que permiten definir el comportamiento del sistema en condiciones ideales, a partir de un modelo matemático que especifica los parámetros que generan el movimiento del sistema robótico modular. Posteriormente, el modelo matemático se implementa en los módulos MECABOT 3.0 con el fin de demostrar su aplicabilidad en un entorno real a partir de la obtención de algunos parámetros de movimiento y las consideraciones de diseño en desarrollos futuros, y así sentar una base para la generación y exploración de más estudios que permitan expandir el conocimiento en el área de la robótica modular.

El documento está organizado de la siguiente forma: 1.Introducción, 2.Referentes Teóricos, 3.Antecedentes de Robótica Modular, 4. Simulación del Sistema Robótico Modular MECABOT 3.0, 5.Implementación Sistema Robótico Modular MECABOT 3.0, 6.Pruebas y Resultados, 7. Conclusiones y Recomendaciones a Trabajos Futuros, 8. Anexos.

1.1 Definición del Problema

Los robots de estructura fija presentan una serie de dificultades al ser empleados en entornos variables, puesto que no suelen estar diseñados para ajustarse a alteraciones en las condiciones de trabajo, haciendo necesario el uso de varios robots cada uno con capacidad de cumplir un número limitado de tareas, causando que la inversión en el proyecto aumente considerablemente; igualmente, al estar constituidos por sistemas complejos suelen requerir de mano de obra especializada para su reparación; tampoco permiten alterar considerablemente su morfología según la circunstancia lo demande, manejando un área de trabajo limitada. Esta serie de desventajas conllevan a buscar una alternativa que sea capaz de cumplir una amplia variedad de requisitos, sin consumir muchos recursos. Para dar solución a estos inconvenientes, se desarrolló un nuevo concepto: "Sistema Robótico Modular", bajo la idea de que a partir de bloques con capacidad de unirse unos con otros, se formara una estructura superior, que al ser reordenada mediante la modificación de sus conexiones facilitara su desempeño en diversas tareas, bien sea en la ejecución de su labor o en su desplazamiento. La capacidad de agrupación de un sistema robótico modular permite obtener diferentes tipos de locomociones tales como: oruga, serpiente, de rueda, carro, bípedo, cuadrúpedo, hexápodo etc.

Dada la importancia del desarrollo de estos sistemas robóticos modulares, la Universidad Militar Nueva Granada ha incursionado en este ámbito desde el año 2013 con el diseño del módulo MECABOT en el proyecto "Diseño y simulación de un robot modular reconfigurable" y MECABOT 3.0 en el proyecto "Rediseño e implementación de un módulo robótico para sistema colaborativo", con el fin de emplearlo como herramienta para operaciones de búsqueda y rescate. El desarrollo de MECABOT ha sido enfocado únicamente a su diseño sin ser puesto a prueba bajo simulación o implementación, y teniendo en cuenta que este sistema robótico modular está en capacidad de adoptar diferentes configuraciones, se hace necesario el desarrollo de un archivo controlador que genere una locomoción que se ajuste a cada configuración. Entonces ¿Cómo realizar la simulación e

implementación del movimiento del sistema robótico modular MECABOT 3.0 existente en la Universidad Militar Nueva Granada en configuraciones tipo oruga y serpiente?

1.2 Objetivo General

Simular e implementar movimientos para un sistema robótico modular considerando dos diferentes configuraciones

1.3 Objetivos Específicos

- Simular la locomoción tipo oruga para el sistema robótico modular MECABOT 3.0.
- Simular la locomoción tipo serpiente para el sistema robótico modular MECABOT 3.0.
- Implementar la configuración tipo oruga en el sistema robótico modular MECABOT 3.0.
- Implementar la configuración tipo serpiente en el sistema robótico modular MECABOT 3.0.
- Realizar Pruebas finales y obtener un indicador de rendimiento.

1.4 Justificación

La robótica fija plantea su enfoque en dar solución a problemas específicos, desarrollando para cada aplicación un robot con ciertas características bien delimitadas; pero, esto genera mayores costos debido a la falta de versatilidad de los mismos. Para solucionar este problema, es ideal construir un sistema robótico modular que se adapte fácilmente a múltiples condiciones y escenarios. La robótica modular, permite que, en base a módulos relativamente sencillos, se puedan construir diversas estructuras adaptables a una gran variedad de tareas. Los robots con funciones fijas presentan otra desventaja ya que, si se daña una pieza de su estructura, el robot queda inutilizable, lo que genera inconvenientes como la detención de la producción o la necesidad de comprar un nuevo robot, entre otros; al tener un sistema robótico modular homogéneo, los diferentes componentes son fácilmente reemplazables, ya que su estructura es idéntica, sencilla y su ensamble es simple, permitiendo

menor complejidad a la hora de realizar las reparaciones necesarias. Es por esto, que la investigación en el ámbito de la robótica modular es de gran importancia para suplir las necesidades de diversas aplicaciones en ambientes variables, que no puedan ser cumplidas por robots de estructura fija o por el ser humano. Los sistemas robóticos modulares pueden clasificarse según los arreglos geométricos en los que se unen los bloques que lo constituyen, pueden ser *Lattice* (enrejado), cadena, árbol, o móviles. Los sistemas robóticos modulares tipo cadena se conforman por la unión en serie de los módulos, tienen como ventajas la capacidad de desplazarse por terrenos no uniformes gracias a que pueden adoptar 5 modos de marcha: línea recta, trayectoria circular, rodar, rotación y desplazamiento lateral, éste se divide a su vez en normal, inclinado y tipo remero (Gonzalez Gómez, 2008) (Rus, Salemi, Shen, & Yim, 2007).

El desarrollo de este documento se enfoca en un estudio previo del movimiento que se genera con el sistema robótico modular para los movimientos en línea recta y desplazamiento lateral, de tal forma que se dimensionen las condiciones de funcionamiento y así lograr la simulación e implementación del movimiento de un sistema robótico modular MECABOT 3.0 bajo dos configuraciones: oruga y serpiente.

Las razones por las cuales se busca trabajar en el campo de la robótica modular son:

- Versatilidad: Su configuración es adaptable al medio donde se encuentre, su capacidad de reconfiguración facilita el desmonte y montaje de otra morfología que se adecue al ambiente y nuevas tareas que se requieran.
- Robustez: Facilidad de intercambio de piezas entre diferentes robots y reparación de piezas defectuosas.
- Economía: Reducción de costos al tener la posibilidad de hacer muchas copias de un módulo, donde la producción en masa reduce el costo. Posibilidad de construcción de máquinas complejas a partir de estos módulos, debido a la generalidad y reutilización de las piezas de un sistema. (Rus, Salemi, Shen, & Yim, 2007).
- Los sistemas robóticos modulares pueden ser auto-reconfigurables, auto-reparables.

- Los sistemas robóticos modulares pueden transformarse en diferentes configuraciones funcionales de acuerdo al entorno en el que se encuentren.
- Son utilizados en operaciones de búsqueda y rescate, como herramientas de carga o pueden dividirse en secciones pequeñas para ejecutar tareas en paralelo.
- Pueden emplearse en el campo industrial, aeroespacial, de exploración, investigación, tratamientos médicos, detección militar, entre otras (Cui, Tang, Wang, Zhao, & Zhu, 2013).

1.5 Metodología

1) Simulación de la locomoción tipo oruga para un sistema robótico modular.
   a) Revisión de la literatura existente sobre sistemas de locomoción tipo oruga para sistemas robóticos modulares.
   b) Desarrollo del modelo matemático para describir la locomoción en 1D en tiempo continuo.
   c) Desarrollo del modelo matemático para describir la locomoción en 1D en tiempo discreto.
   d) Revisión de la información disponible para el manejo del software de simulación Webots.
   e) Importación del diseño de los módulos al software Webots
   f) Ensamble virtual del sistema robótico modular en la configuración tipo oruga según las conexiones tipo pivote – pivote, pivote – centro, centro – centro.
   g) Recopilación de los datos obtenidos en la simulación del sistema robótico modular MECABOT 3.0 virtual en configuración tipo oruga según las conexiones tipo: pivote – pivote, pivote – centro, centro – centro, a partir de la variación de diversos parámetros en el archivo controlador de Webots.

2) Simulación de la locomoción tipo serpiente para un sistema robótico modular.
   a) Revisión de la literatura existente sobre sistemas de locomoción tipo serpiente para sistemas robóticos modulares.

b) Desarrollo del modelo matemático para describir la locomoción en 2D en tiempo continuo.
c) Desarrollo del modelo matemático para describir la locomoción en 2D en tiempo discreto.
d) Ensamble del sistema robótico modular MECABOT 3.0 virtual en la configuración tipo serpiente según las conexiones tipo pivote – pivote, pivote – centro, centro – centro.
e) Recopilación de los datos obtenidos en la simulación del sistema robótico modular MECABOT 3.0 virtual en configuración tipo serpiente según las conexiones tipo: pivote – pivote, pivote – centro, centro – centro, a partir de la variación de diversos parámetros en el archivo controlador de Webots.

3) Implementación la configuración tipo oruga en un sistema robótico modular.
   a) Construcción de los módulos mediante piezas impresas en 3D.
   b) Ensamble de las piezas impresas para construir los módulos.
   c) Ubicación de los componentes electromecánicos en los módulos.
   d) Desarrollo de una interfaz gráfica que comunique los módulos con el computador maestro.
   e) Caracterización de los motores que articulan los pivotes.
   f) Programación de las tarjetas electrónicas.
   g) Construcción el ensamble del sistema robótico modular en la configuración tipo oruga según las conexiones tipo pivote – pivote, pivote – centro, centro – centro.
4) Implementación de la configuración tipo serpiente en un sistema robótico modular.
   a) Construcción del ensamble del sistema robótico modular en la configuración tipo serpiente según las conexiones tipo pivote – pivote, pivote – centro, centro – centro.
5) Pruebas finales, algunos indicadores de rendimiento y trabajo futuro
   a) Recopilación de los datos obtenidos en la implementación del sistema robótico modular MECABOT 3.0 real en configuración tipo oruga según las conexiones tipo: pivote – pivote, pivote – centro, centro – centro, a partir de la variación de diversos parámetros en la interfaz gráfica.

b) Recopilación de los datos obtenidos en la implementación del sistema robótico modular MECABOT 3.0 real en configuración tipo serpiente según las conexiones tipo: pivote – pivote, pivote – centro, centro – centro, a partir de la variación de diversos parámetros en la interfaz gráfica.
c) Análisis y comparación de los datos obtenidos a partir de la implementación en el módulo MECABOT 3.0 real, con el fin de concluir un rango de funcionamiento adecuado, la conexión más conveniente para cada configuración y establecer nuevos criterios de diseño que permitan mejorar el funcionamiento de MECABOT 3.0 para una generación futura.

2 Referentes Teóricos

2.1 Marco Teórico

2.1.1 Robótica

El ser humano en su constante afán por el progreso, ha desarrollado máquinas independientes capaces de realizar labores complejas que faciliten su trabajo, conocidas hoy en día bajo el término de "robot". En la actualidad, se considera robot al "sistema electromecánico reprogramable que permite realizar diferentes tareas repetitivas que requieren un grado elevado de precisión" (Aranda, y otros, 2012) o bien a la "máquina o ingenio electrónico programable capaz de manipular objetos y realizar operaciones antes reservadas sólo a las personas" (Calles, Hernández, Ortiz, & Rodríguez, 2015). Así, la robótica se puede definir como una "técnica que aplica la informática al diseño y empleo de aparatos que, en sustitución de personas, realizan operaciones o trabajos, por lo general en instalaciones industriales" (Calles, Hernández, Ortiz, & Rodríguez, 2015); o como "un sistema autónomo que existe en el entorno físico, puede sensar su entorno y actuar sobre este para alcanzar algunas metas", entendiendo por autónomo como aquel robot que "actúa en base a sus propias decisiones, y no es controlado por el ser humano". (Mataric, 2007).

Teniendo en cuenta el concepto de robot, se puede comprender con mayor facilidad el término "robótica móvil". Un robot móvil se define como "Un dispositivo electromecánico capaz de desplazarse dentro de un espacio de trabajo con diferentes niveles de autonomía" (Aranda, Salgado, & Velasco, 2002).

2.1.2 Robótica Móvil

Un robot móvil, puede definirse como "una plataforma mecánica dotada de un sistema de locomoción, que le permite desplazarse en un determinado entorno de trabajo; de un sistema de planificación de tareas y trayectorias, que define de forma automática una secuencia de acciones que deben ser llevadas a cabo por el robot; de un sistema de percepción que le permite tener conocimiento del entorno por el que se mueve, y de un sistema de control y navegación, que, en

función de la tarea a realizar y de la información sensorial, dirija los movimientos del robot en cada instante, dotándolo de cierto grado de autonomía.” (Calonge & de la Fuente, 1999). Es decir, es un robot que puede desplazarse por el entorno en el que opera, gracias a unas trayectorias que pueden ser definidas previamente o determinadas por el robot mientras se desplaza.

La robótica móvil se desarrolló con el fin de extender el campo de uso de los robots hacia actividades que requieren un desplazamiento, por ejemplo, vigilancia, exploración, labores de búsqueda y rescate, entre otros. Para cumplir con estas labores, se han desarrollado varios mecanismos que generan la movilidad del robot, entre ellos ruedas y piernas. Un ejemplo de esto es el robot Pioneer, es la representación de investigación más popular en robótica móvil. Como se puede observar en la Figura 2-1 esta constituido por un controlador integrado, servomotores de 500 pasos, ruedas de 19cm,cuepo de aluminio resistente, 8 sensores de ultrasonido orientados hacia adelante. (Mobile Robots, 2016)

Figura 2-1 Robot Pioneer
Tomado de: (PIONEER, 2015)

Otra representación es el robot ASIMO (Advanced Step in Innovate Mobility) como se puede apreciar en la Figura 2-2 se trata de un robot humanoide de la compañía japonesa Honda. Este pretende incentivar la investigación en el campo de la movilidad para crear un robot que interactúe con las personas y las ayude a hacer su vida más fácil enfocándose en personas que carecen de movilidad completa en sus cuerpos. (Honda, 2016)

Figura 2-2 Robot ASIMO
Tomado de: (Honda, 2016)

2.1.3 Arquitectura de Robots Móviles Modulares

El concepto de modularidad se define como "la característica de un sistema de poder ser construido a partir de un conjunto de componentes estandarizados intercambiables". (Yim, 1994)

La robótica modular tiene como objetivo el ensamble de módulos simples en una gran variedad de configuraciones para adaptarse a diferentes aplicaciones, su diseño va enfocado a la generalidad y optimización en un rango de aplicaciones. La locomoción depende de la estructura y su configuración elegida de tal modo que cumpla con las restricciones del medio en el que está inmerso como obstáculos, estabilidad y las tareas que debe cumplir. En el presente trabajo se utiliza la configuración tipo oruga, una de las más usadas para el desplazamiento en espacios reducidos y la configuración tipo serpiente, con el fin de que el robot pueda evadir obstáculos. (Brandt, Christensen, & Hautop Lund, 2007).

Los robots modulares pueden ser clasificados según su arquitectura por su disposición geométrica en cada unidad, así:

- ***Arquitectura móvil:*** Están compuestos de unidades que emplean su entorno para ejecutar maniobras de desplazamiento y conformar cadenas o *lattice* (mallas) complejas.
- ***Arquitectura Lattice (malla):*** Los módulos están conectados por un patrón tridimensional (malla). El control y movimiento se pueden realizar en paralelo y la reconfiguración es sencilla ya que se tiene un número limitado de ubicaciones para el desplazamiento de los módulos. La simulación computacional se puede escalar con facilidad para representar sistemas más complejos.
- ***Arquitectura en cadena:*** La interconexión es en cadena o árbol, es aquella en las que los módulos están conectados en serie, es muy versátil ya que pueden llegar a casi cualquier punto en el espacio, el movimiento y control de los módulos es secuencial; sin embargo, son computacionalmente más difíciles de simular, analizar y controlar. Dentro de esta categoría se encuentra topología 1D, 2D y 3D, los cuales pueden ser apreciados en la Figura 2-3 (Gonzalez Gómez, 2008).

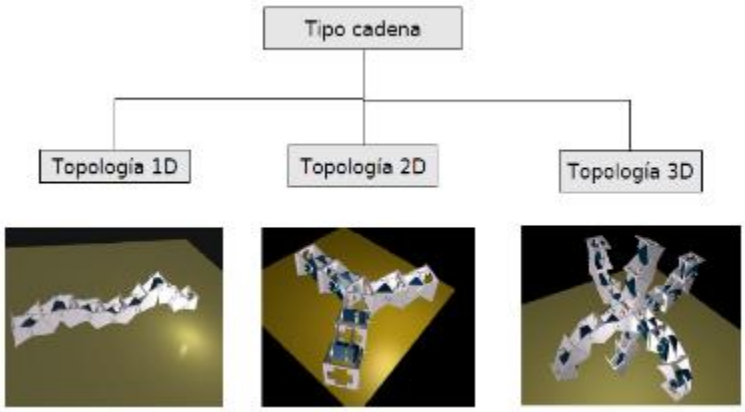


Figura 2-3 Topologías para robot tipo cadena
(Gonzalez Gómez, 2008)

2.1.4 Arquitectura Tipo Cadena Topología 1D

Dentro de la topología unidimensional se encuentran dos clases: Los ápodos y los ápodos autopropulsados como se ilustra en la Figura 2-4.

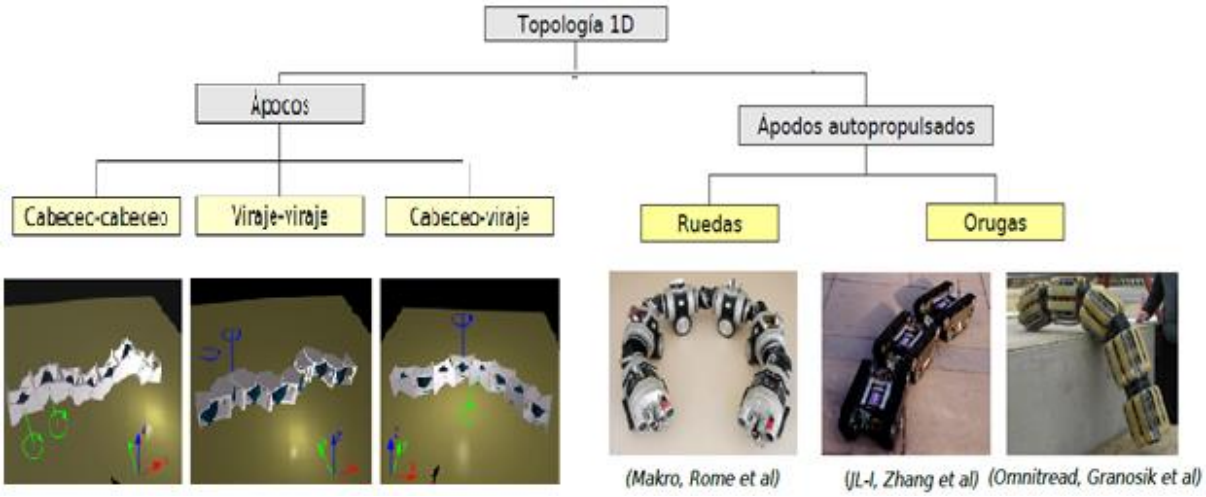


Figura 2-4 División de topología 1D para robots Tipo cadena
Tomado de: (Gonzalez Gómez, 2008)

El término ápodo hace referencia a que son carentes de patas y el desplazamiento terrestre se realiza por medio del deslizamiento de su cuerpo contra la superficie. Los robots ápodos son inspirados en la naturaleza, en patrones de movimiento como los de gusanos y serpientes que tienen una estructura segmentada que permite que su cuerpo adopte formas ondulatorias y de esta manera cambie su forma fácilmente, adaptándose al terreno por el que se desplazan. Los módulos para los robots ápodos por si solos no logran avanzar, sólo pueden hacerlo al estar unidos y formar una cadena coordinada. (Méndez-Polanco & Muñoz-Meléndez, 2008). Estos se clasifican según su conexión en: conexión viraje-viraje, conexión cabeceo- viraje, conexión cabeceo-cabeceo.

Para el caso de los ápodos autopropulsados, las partes que conforman el robot tienen ruedas u orugas para realizar el desplazamiento como se ilustra en la Figura 2-4, no son robots bio-inspirados, se trata de la unión en cadena de módulos similares. Los módulos pueden desplazarse como unidades autónomas ya que disponen de sistemas propulsivos (Rus, Salemi, Shen, & Yim, 2007).

Como se mencionó anteriormente, dependiendo de la conexión de los módulos que conforman la cadena, se da lugar a diferentes configuraciones:

- Viraje-Viraje: Los módulos están conectados de tal forma que todos giran en el plano xy. Su desplazamiento se da en forma lateral, perpendicular a la longitud de la cadena. Esta

conexión genera el movimiento apoyando su cuerpo en el suelo completamente; emplea ruedas pasivas para reducir la fricción.

- Cabeceo-Cabeceo: Los módulos están conectados de tal forma que todos giran en el plano xz, generando la configuración tipo oruga. Su desplazamiento se da hacia adelante o atrás.
- Cabeceo-Viraje: Esta configuración resulta de alternar el mismo número de módulos de cabeceo y de viraje, generando locomoción de dos dimensiones y formando una curva en tres dimensiones. Para este caso, el módulo de viraje está rotado 90° con respecto al módulo de cabeceo (referencia). Genera diferentes tipos de movimiento como desplazamiento lateral, trepar, rodar, entre otras.

El control para las tres arquitecturas mencionadas anteriormente puede ser centralizado o distribuido entre módulos, y el movimiento puede ser ejecutado en serie o paralelo.

Para el estudio de los sistemas robóticos modulares tipo cadena, se realiza el diseño e implementación de las configuraciones tipo serpiente y oruga, en un sistema robótico modular ápodo tipo cadena de M módulos, estos son isomorfos y se caracterizan según los siguientes parámetros:

- Angulo de doblaje (φ_i): ángulo que se forma entre dos segmentos unidos por una articulación y está determinado por el rango del servomotor [-90°, 90°].
- Longitud (L): Se define como la distancia entre los extremos del módulo, cuando el ángulo de doblaje es de 0°. El valor de cada segmento, es de L/2.
- Número de módulos (M): Cantidad de módulos que conforman el sistema.
- Longitud del robot (l): es la distancia total de la unión de los M módulos, entonces $l = ML$.

2.1.5 Concepto Módulo

Antes de comenzar a hacer cualquier análisis matemático relativo a la robótica modular, se debe entender la conformación de un módulo. Un módulo son dos segmentos iguales unidos por una articulación (Gonzalez Gómez, 2008), así pues, la longitud de este es la suma del par de segmentos (segmento derecho y segmento izquierdo).

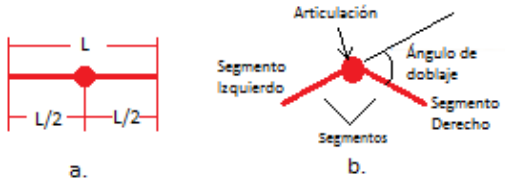


Figura 2-5 Representación módulo.
a. En reposo. b. En movimiento

En la Figura 2-5 se pueden observar algunos parámetros con los que se trabajan como lo son la longitud del módulo y el ángulo de doblaje al comenzar la locomoción.

2.1.6 Parámetros Modelo Cinemático

Para describir el movimiento en las diferentes configuraciones se toman como parámetros los siguientes:

- Orientación $\vec{\sigma}$: Permite conocer la orientación del robot en un instante con respecto a un eje de referencia.
- Velocidad lineal media (Ec. 2-1) $\underset{V}{\rightarrow}$:

$$\underset{V}{\rightarrow}=\frac{\overrightarrow{\Delta r}}{T}=\underset{\Delta r}{\rightarrow}*f$$

Ec. 2-1

Para este documento, se realiza el estudio de un sistema robótico modular donde se analiza un único tipo de módulo (M), que se desplaza en una superficie homogénea, en la que el robot tendrá una velocidad media constante durante un periodo de tiempo T que se repite cíclicamente. Esta consideración es en base a la investigación realizada por (Rus, Salemi, Shen, & Yim, 2007) sobre el movimiento en un robot modular reconfigurable.

2.2 Locomoción en 1D

A continuación, se presentan algunos de los enfoques existentes para la generación de la locomoción de un sistema robótico modular en cadena, haciendo énfasis en el modelo propuesto por Gómez (Gonzalez Gómez, 2008) en el que se basa este proyecto de grado. Se realiza una breve explicación matemática de los generadores sinusoidales, sus parámetros y variables para el grupo

cabeceo-cabeceo (configuración tipo Oruga), tanto en tiempo continuo como en tiempo discreto. Se definen consideraciones, restricciones y criterio de estabilidad de locomoción con el fin de obtener las ecuaciones que rigen el movimiento del sistema robótico modular en configuración tipo Oruga.

2.2.1 Consideraciones

Antes de iniciar con el análisis matemático de la locomoción en configuración tipo oruga o serpiente, se hace necesario definir que el sistema robótico modular a trabajar es MECABOT 3.0 II (Pardo Puentes, 2015) presentado en la sección 3.1.7. Su diseño mecánico y electrónico se explica en detalle en la sección 5 Implementación Sistema Robótico Modular MECABOT 3.0.

Para el desarrollo de esta sección se tendrá en cuenta que:

- MECABOT 3.0 se emplea para conformar un sistema robótico modular 1D en cadena.
- MECABOT 3.0 cuenta con un cuerpo y un pivote articulado
- La articulación de MECABOT 3.0 está dada por dos servomotores ubicados a los costados del pivote.
- MECABOT 3.0 cuenta con un acople magnético mediante imanes.

2.2.2 Generadores Sinusoidales

Para lograr que el sistema robótico modular tipo cadena se desplace únicamente por el giro en sus articulaciones, se deben calcular las funciones y parámetros que determinan los ángulos necesarios para que el robot coordine los M módulos.

Existen diversos enfoques que permiten la obtención de estas funciones tales como el clásico, tablas de control de la marcha (Gait Control Tables), y bio-inspirado. El enfoque clásico se basa en la cinemática inversa que busca hallar, a partir de la posición del efector final, el conjunto de ángulos que deben adoptar las articulaciones del robot (Castellanos, Delgado, & Giraldo, 2006), pero tiene como desventaja que el modelo matemático varía según la configuración

del robot, es decir, requiere una gran capacidad de procesamiento para las diferentes ecuaciones que genera. Por otro lado, las tablas de control de marcha se componen de una serie de elementos que describen secuencias del comportamiento de los grados de libertad, para producir un paso en la marcha (Yim, 1994) , de esta manera el controlador procesa la secuencia y envía a los actuadores a las respectivas posiciones, este tipo de controlador demanda bajo coste computacional, pero tiene como desventaja que, para ajustar un nuevo movimiento se hace necesario volver a computar la tabla (Gonzalez Gómez, 2008). Por último, se tiene el enfoque bio-inspirado, basado en circuitos neuronales de animales vertebrados e invertebrados denominados CPG (Central Pattern Generators) que se encargan de producir estímulos rítmicos para acciones como la respiración y la marcha, sin necesidad de recibir estímulos externos (Bucher & Marder, 2001); este enfoque busca obtener un modelo matemático que mediante una aproximación al CPG, permita el control de la locomoción de los sistemas robóticos. Los modelos de control basados en CPG se han empleado un gran número de aplicaciones como cuadrúpedos (Fukuoka, Hada, Kimura, & Takase, 2003), hexápodos (Arai, Inagaki, & Yuasa, 2003), nadadores (Ijspeert & Crespi, 2008), terrestres (Dario, Menciassi, Sfakiotakis, la Spin, & Tsakiris, 2005), entre otros ya que presenta como ventaja que actúa directamente en las articulaciones, lo que produce un movimiento más armonioso y eficiente (Crespi & Ijspeert, 2007). Es por esto, que se emplea un modelo matemático de control basado en el CPG como controlador para los módulos del sistema robótico a trabajar.

Para lograr la locomoción del sistema robótico modular en cadena MECABOT 3.0 se requiere de un modelo matemático que permita obtener la locomoción de las configuraciones tipo serpiente y oruga deseadas, para esto se emplea el modelo matemático bio-inspirado presentado por Gómez; este modelo plantea que “En estado estacionario, (las redes CPG) se comportan como osciladores de frecuencia fija lo que permite sustituirlas por un modelo simplificado formado por generadores sinusoidales. La ventaja es que son extremadamente sencillos de implementar y se requieren muy pocos recursos para su realización.” (Gonzalez Gómez, 2008)

Gómez describe matemáticamente los osciladores de frecuencia fija gracias al desarrollo por series de Fourier (Ec. 2-2).

Las series de Fourier tienen la forma:

$$\varphi_i(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos\left(\frac{2\pi n}{T} t\right) + b_n \operatorname{sen}\left(\frac{2\pi n}{T} t\right) \right)$$

Ec. 2-2

Se realiza una aproximación de la serie con el fin de limitar el problema a las funciones sinusoidales de frecuencia fija (Ec. 2-3) (n = 1) se tiene que:

$$\varphi_i(t) \approx \frac{a_0}{2} + a_1 \cos\left(\frac{2\pi}{T} t\right) + b_1 \operatorname{sen}\left(\frac{2\pi}{T} t\right)$$

Ec. 2-3

Realizando el cambio de variables (Ec. 2-4)

$$a_1 = A_i \operatorname{sen}(\psi_i),\ b_1 = A_i \cos(\psi_i)\ , O_i = \frac{a_0}{2}$$

Ec. 2-4

Donde (Tabla 2-1)

Símbolo	Variable	Unidad	Símbolo	Variable	Unidad
i	Generador	-	ψ_i	Fase	rad
A_i	Amplitud	-	O_i	Compensación	rad
T	Periodo	s	t	Tiempo	s

Tabla 2-1 Variables serie de Fourier

Se tiene (Ec. 2-5)

$$\varphi_i(t) = O_i + A_i \operatorname{sen}(\psi_i) \cos\left(\frac{2\pi}{T} t\right) + A_i \cos(\psi_i) \operatorname{sen}\left(\frac{2\pi}{T} t\right)$$

Ec. 2-5

Según la identidad del seno de la suma de dos ángulos, se obtiene de forma simplificada (Ec. 2-6):

$$\varphi_i(t) = A_i \operatorname{sen}\left(\frac{2\pi}{T} t + \psi_i\right) + O_i$$

Ec. 2-6

La coordinación del movimiento del sistema robótico, no se ve afectado por el valor de la frecuencia, por tanto, se reemplaza por el valor de ϕ y se estudia la posición de la articulación en función de la fase. La fase varía linealmente con el tiempo según la relación (Ec. 2-7)

$$\phi(t) = \frac{2\pi}{T}t$$

Ec. 2-7

Por tanto (Ec. 2-8),

$$\varphi_i(\phi) = A_i sen(\phi + \psi_i) + O_i, i \epsilon \{1 \dots M\}$$

Ec. 2-8

El ángulo de doblaje de cada módulo se encuentra en $\varphi_i \epsilon [O_i - A_i, O_i + A_i]$. Debido a las limitaciones mecánicas de los servomotores (3.1.7), el valor de ángulo máximo posible es de 180°

$$|O_i| + A_i \leq 90°$$

Ec. 2-9

2.2.2.1 Generadores Sinusoidales para Grupo Cabeceo-Cabeceo

Para esta configuración se tiene en cuenta que lo conforman M generadores sinusoidales con amplitud A y diferencia entre fases $\Delta\varphi$ de un par de módulos consecutivos, la ecuación que describe esta onda es Ec. 2-10:

$$\varphi_i(\phi) = A\,\text{sen}(\phi + (i-1)\Delta\phi + \psi_i), i \epsilon \{1 \dots M\}$$

Ec. 2-10

La anterior ecuación se obtiene de la Ec. 2-7 tomando en cuenta las siguientes restricciones:

- Para lograr simetría de las oscilaciones frente al origen, todos los generadores tienen un offset $O_i = 0$ y la misma amplitud A.
- Las fases de todos los módulos se expresan en función de la fase del primer módulo ψ_1. Al tener limitado el estudio a un sistema en estado estacionario, el valor de la fase inicial se omite.
- Para obtener un movimiento armónico en el conjunto de módulos, la diferencia de fase $\Delta\phi$ es constante.

En el presente capítulo se analiza el modelo matemático de la curva serpentinoide 2D que da origen a la locomoción en una dimensión de la configuración Cabeceo-Cabeceo, definiendo sus parámetros, características y restricciones, al considerar el sistema en tiempo continuo y discreto.

2.2.2.2 Curva Serpentinoide 2D

El sistema robótico modular MECABOT 3.0 se desplaza en una dimensión en configuración tipo oruga, por tanto, se modela con una curva continua s, que indica la distancia entre los extremos del robot. Estas curvas vienen descritas por el ángulo de doblaje $\theta(s)$ cuando es continuo, y φ_i cuando es discreto.

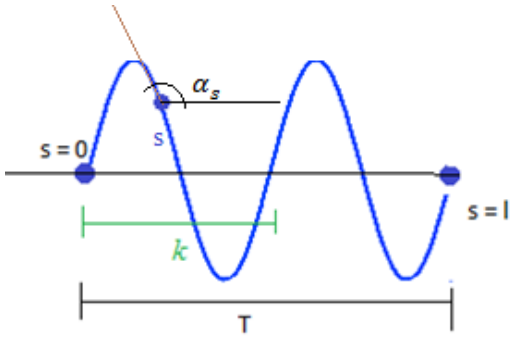


Figura 2-6 Parámetros de la curva serpentinoide

La curva serpentinoide (Figura 2-6) describe una onda sinusoidal que varía a lo largo del eje corporal de un sistema robótico modular tipo oruga o serpiente y se representa por (Ec. 2-11)

$$\theta(s) = A\ sen\left(\frac{2\pi k}{l}s\right)$$

Ec. 2-11

Donde $\theta(s)$ es el ángulo de doblaje o el ángulo que se forma al unir dos tangentes a la curva serpentinoide con periodo T y una ondulación *k*, de longitud l que pasan por dos puntos a una distancia ds. Por otro lado, cuando una curva serpentinoide es dependiente del ángulo de doblaje y además de la fase, se denomina onda serpentinoide y se describe así (Ec. 2-12):

$$\theta(s,\phi) = A\ sen\ \left(\phi + \frac{2\pi k}{l}s\right)$$

Ec. 2-12

El ángulo que se forma desde una línea tangente al punto *s* con respecto al eje *x* se representa por α_s donde, según el estudio sobre los robots biológicamente inspirados realizado por Hirose (Hirose, 1993) , la ecuación de α_s (Ec. 2-13)

$$\alpha_s = \alpha\ cos\left(\frac{2\pi k}{l}s\right)$$

Ec. 2-13

Por otro lado $\propto$ es el ángulo de serpenteo que se forma entre la línea tangencial en el punto inicial de la curva (s=0) con respecto al eje x.

El valor máximo del ángulo de serpenteo es 120° ya que, de superarse, el robot presentaría colisión entre los módulos. Las ondulaciones k y la altura son inversamente proporcionales.

Se tienen como parámetros de la curva serpentinoide $\propto, k\ y\ l$ pero al ser l una constante la curva se define por $\propto y\ k$ como se observa en la Figura 2-7.

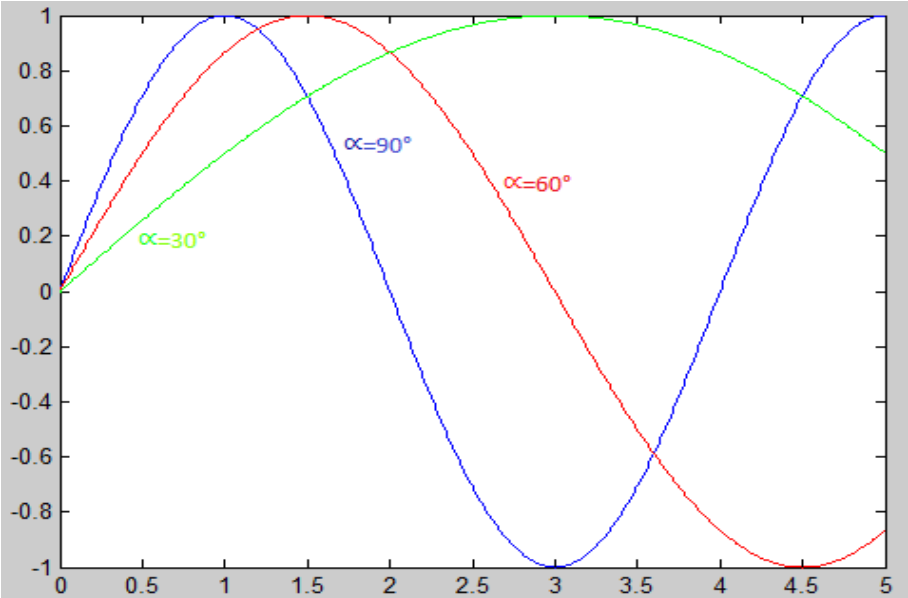


Figura 2-7 Curva serpentinoide para diferentes ángulos de doblaje

La curva serpentinoide se caracteriza únicamente por los pares $(\propto, k)$ por tanto, es posible generar un sistema coordenado que permita definir la forma del robot, al igual que la región en la que es estáticamente estable, es decir en la que existe movimiento.

2.2.3 Sistema En Tiempo Continuo

La locomoción de un sistema robótico modular en configuraciones tipo oruga y serpiente consiste en la propagación de ondas sinusoidales a lo largo del cuerpo, estas ondas propician el movimiento únicamente si se tiene una adecuada coordinación entre los módulos contemplando los parámetros de ciclo que dan lugar a la marcha. Dependiendo del sentido de propagación de la

onda, se determina si el robot avanza (cola a cabeza) o si retrocede (cabeza - cola). La distancia que se desplaza, se determina según la altura de la onda propagada, a mayor altura h de la onda, mayor desplazamiento se obtiene.

Para determinar correctamente una onda serpentinoide se deben tener en cuenta los siguientes parámetros (Tabla 2-2)

Símbolo	**Variable**	**Rango**	**Unidad**
α	Ángulo de serpenteo	$\alpha \in [0°, 120°]$	rad
k	Número de ondulaciones	$k \geq 1$	-
l	Longitud	-	m
ϕ	Fase	-	rad
h	Altura	-	m
w	Anchura	-	m

Tabla 2-2 Parámetros de la onda serpentinoide

Los anteriores parámetros definen los valores en los que el movimiento es estáticamente estable, el desplazamiento es nulo o el movimiento es muy brusco. Se dice que el robot es estable en una fase ϕ si existen al menos dos puntos de contacto con la superficie $k \geq 2$ que, al unirlos con un segmento de recta, se atraviese el centro de gravedad del robot (Figura 2-8). Si el robot es estable en todas las fases ϕ, entonces la locomoción es estáticamente estable. Sin embargo, en la transición entre los puntos de la región estable pueden aparecer puntos en los que $k = 1$ por tanto, cae en la región inestable, provocando movimientos bruscos en el robot, esta transición se puede suavizar variando el ángulo α.

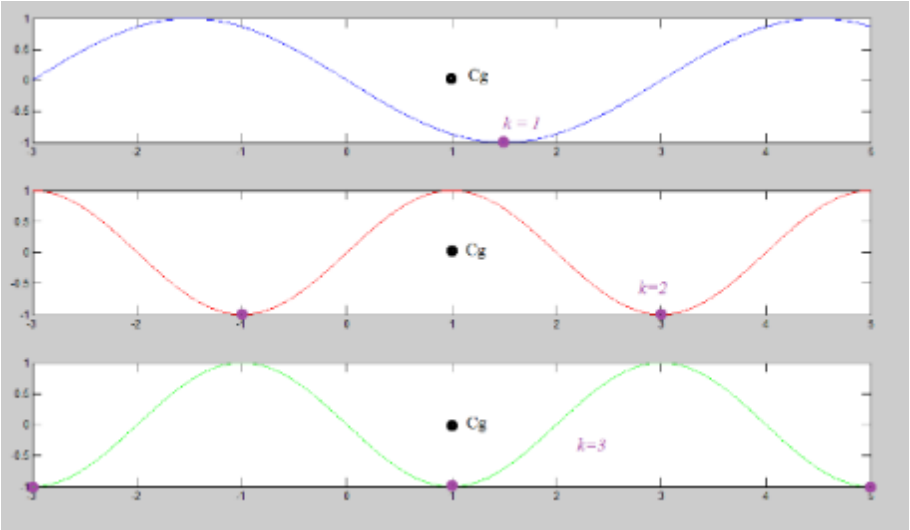


Figura 2-8 Puntos de contacto en onda serpentinoide

La estabilidad en la locomoción mejora al aumentar el valor del parámetro *k*, ya que este proporciona un mayor número de puntos medios de contacto con la superficie y por tanto hace que la altura del robot disminuya.

Una vez se obtiene (α, k) en los valores en donde el movimiento es estáticamente estable es posible calcular el desplazamiento o paso (Figura 2-9) mediante la ecuación Ec. 2-14.

$$\Delta x = (l - w)\,\frac{1}{k}$$

Ec. 2-14

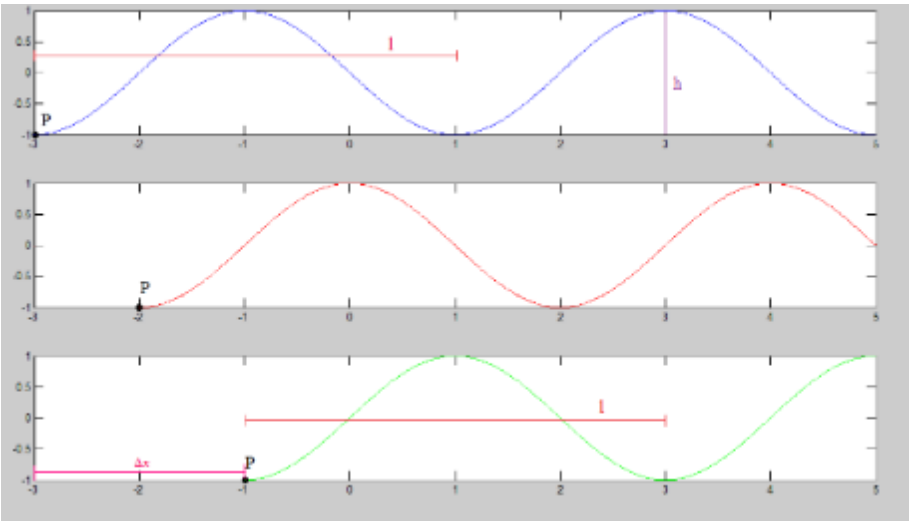


Figura 2-9 Parámetros espacio de formas

2.2.4 Sistema En Tiempo Discreto

Los robots modulares reales son sistemas discretos constituidos por múltiples módulos *M*, que, al unirse conforman el sistema robótico y permiten su movimiento. Para el sistema robótico modular MECABOT 3.0, se parte del desarrollo matemático en tiempo continúo realizado en la sección anterior (2.2.3) teniendo en consideración la distancia *d* que separa los diferentes módulos y empleando los mismos parámetros que describen la locomoción en tiempo continuo (número de ondulaciones y ángulo de doblaje). Para determinar la onda generadora del movimiento del sistema se emplea un nuevo parámetro M_u (Ec. 2-15):

$$M_u = \frac{M}{k}$$

Ec. 2-15

Mientras mayor sea el parámetro M_u, la curva formada por los módulos es más cercana a la curva serpentinoide del modelo continuo. Así, la locomoción del robot discreto queda definida bajo la siguiente Tabla 2-3:

Símbolo	Variable	Unidad
$\Delta\phi$	Variación de fase	rad
M	Número de módulos	-
M_u	Número de módulos por ondulación	-
k	Número de ondulaciones	s^{-1}
A	Amplitud de los generadores	-
α	Ángulo de serpenteo	rad
d	distancia	m

Tabla 2-3 Parámetros locomoción del robot en tiempo discreto

2.2.4.1 Consideraciones

A diferencia del modelo continuo, en el modelo discreto se debe considerar la distancia existente entre las articulaciones Ec. 2-16.

$$|\Delta\phi| = \frac{2\pi k}{M}$$

Ec. 2-16

Al reemplazar M_u en la ecuación Ec. 2-17.

$$|\Delta\phi| = \frac{2\pi}{M_u}$$

Ec. 2-17

En grados Ec. 2-18

$$|\Delta\phi| = \frac{360}{M_u}$$

Ec. 2-18

Debido a esta discretización, se generan nuevas restricciones.

Por definición $|\Delta\phi|$, puede estar comprendido entre -180 y 180 grados, cuando las articulaciones están opuestas en fase, su ángulo de doblaje es $\phi_i = -\phi_{i+1}$ como se aprecia en la Tabla 2-4, todas las articulaciones tienen el mismo valor absoluto de ángulo, pero signos opuestos.

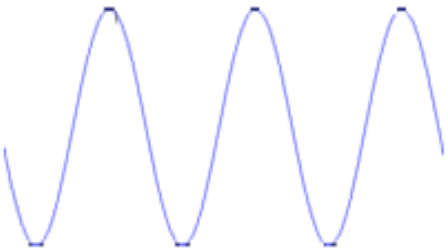
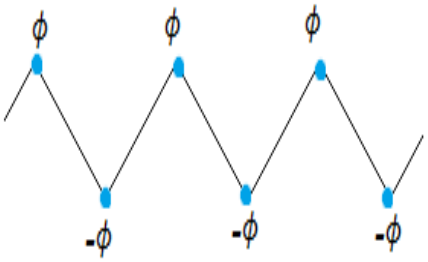
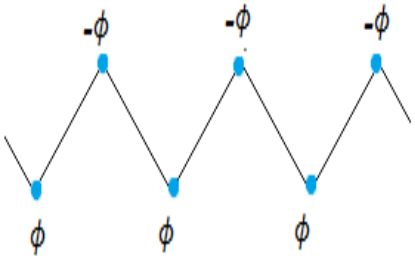
Variación de Fase $\|\Delta\phi\| = 180; M = 6; k = 3$		
	$\boldsymbol{\phi_1}$	ϕ_2
Continuo		
Discreto		

Tabla 2-4 Ángulo de fase en tiempo continuo y discreto

El valor máximo de M_u está determinado por ul número de módulos que componga el sistema, cuando se tiene k = 1, esta cantidad de módulos, se ve limitada por el torque máximo de los servomotores empleados.

El valor máximo de k está dado por Ec. 2-19:

$$k_{max} = \frac{M}{M_{u_{min}}} = \frac{M}{2}$$

Ec. 2-19

Cuanto mayor sea k, menor cantidad de módulos existe por ondulación y el error de discretización es mayor.

Se define la posición de una articulación como Ec. 2-20:

$$s = d_0 + (i-1)d; 1 < i < M$$

Ec. 2-20

Y la unión entre articulaciones Ec. 2-21:

$$s = (i-1)d; 1 < i < M$$

Ec. 2-21

El ángulo de los segmentos de la curva está dado por Ec. 2-22:

$$\alpha_s = \alpha\, cos\left(\frac{2\pi k}{L}s\right) \therefore s = (i-1)d; L = Md$$

Ec. 2-22

Reemplazando se obtiene la Ec. 2-23:

$$\alpha_i = \alpha\, cos\left(\frac{2\pi k}{M}(i-1)\right)$$

Ec. 2-23

El ángulo con el eje x queda expresado como Ec. 2-24:

$$\alpha_i(\phi) = \alpha\, cos\left(\phi + \frac{2\pi k}{M}(i-1)\right)$$

Ec. 2-24

Ángulo de doblaje Ec. 2-25:

$$\varphi_i = \theta(s) = A\, sen\left(\frac{2\pi k}{L}s\right)$$

Ec. 2-25

Reemplazando *s* se obtiene la Ec. 2-26:

$$\varphi_i = \theta(s)|_{s=\mathrm{d_0+(i-1)d}} = A\, sen\left(\frac{2\pi k}{M}((i-1) + \frac{d_0}{d})\right)$$

Ec. 2-26

Siendo A Ec. 2-27:

$$A = 2\alpha\, sen\left(\frac{\pi k}{M}\right)$$

Ec. 2-27

Entonces se obtiene Ec. 2-28

$$\varphi_i(\phi) = 2\alpha\, sen\left(\frac{\pi k}{M}\right)\, sen\left(\phi + \frac{2\pi k}{M}((i-1) + \frac{d_0}{d})\right)$$

Ec. 2-28

2.2.4.2 Criterio de estabilidad para locomoción en 1D

La estabilidad es una característica importante en la locomoción de los robots, este término se refiere a la capacidad de evitar que el robot se caiga en el momento que esté en marcha. (Gorrostieta & Vargas Soto, 2008) Para establecer la estabilidad del sistema, se requiere conocer los puntos de apoyo que tiene el robot en cada instante, estos puntos dependen del número de ondulaciones que describen la curva serpentinoide, y de la cantidad de módulos que conforman la cadena. Estos se relacionan según M_u o número de módulos por ondulación por la ecuación Ec. 2-29.

$$M_u = \frac{M}{k}$$

Ec. 2-29

Este parámetro permite definir la forma de las ondulaciones del robot y determinar si es estable.

El criterio de estabilidad para locomoción en 1D define que, en todo momento durante la propagación de la onda serpentinoide, deben existir al menos dos puntos de apoyo contra el suelo, es decir, que siempre que $k \geq 2$, se puede garantizar la estabilidad de la locomoción del robot durante todas las fases permitiendo que el centro de gravedad permanezca a la misma altura.

2.3 Locomoción en 2D

A continuación se definen los parámetros que describen la locomoción en 2D de un sistema robótico modular en arquitectura cadena, señalando los diferentes modos de caminar y profundizando en el desarrollo del modelo matemático en tiempo continuo y discreto, para el desplazamiento lateral principal basado en el modelo propuesto por (Gonzalez Gómez, 2008), el

cual se implementa para la locomoción en configuración tipo serpiente del presente proyecto de grado. Además, se especifican las restricciones y el criterio de estabilidad de locomoción en 2D para conseguir el desplazamiento lateral.

2.3.1 Parámetros de la locomoción en 2D

La locomoción en 2D de los sistemas robóticos modulares en cadena, presentan 8 movimientos posibles cuando tienen M articulaciones, estos movimientos se caracterizan según los grados de libertad del movimiento definidos según los parámetros característicos del generador de ondas sinusoidales ($\alpha_v, \alpha_h, k_v, k_h, \Phi_{vh}$), donde (Tabla 2-5):

Símbolo	Variable	Unidades
α_v	Ángulo de serpenteo vertical	rad
α_h	Ángulo de serpenteo horizontal	rad
k_v	Número de ondulaciones verticales	s^{-1}
k_h	Número de ondulaciones horizontales	s^{-1}
ϕ_{vh}	Diferencia de fase entre módulo vertical y horizontal.	rad

Tabla 2-5 Parámetros de onda sinusoidal en 3D

Se pueden encontrar 5 modos de caminar diferentes: movimiento en línea recta, trayectoria circular, rodar, desplazamiento lateral y rotación (Gonzalez Gómez, 2008).

- Línea recta: En este grupo entran las configuraciones de tipo cabeceo - cabeceo.
- Trayectoria circular: Permite realizar giros de un radio determinado cuando el ángulo de doblaje horizontal tiene un valor constante diferente de cero.
- Rodar: Los módulos rotan alrededor del eje corporal produciendo un desplazamiento perpendicular al eje longitudinal.
- Lateral: El robot se desplaza hacia uno de los lados manteniendo paralelo su eje longitudinal, puede ser:

  - Principal: El robot se desplaza al ondular el cuerpo (Movimiento tipo serpiente).
  - Inclinado: Los módulos forman un ángulo con respecto a su eje corporal.
  - Remero: El robot se curva formando una "U" que alterna el punto de apoyo entre el centro y los extremos para desplazarse.
- Rotación: Genera el cambio en orientación del eje longitudinal, puede ser en forma de "U" o "S".

Para este proyecto de grado, el sistema robótico modular se desplaza en dos dimensiones limitándose al desarrollo del movimiento del sistema en configuración tipo serpiente, es decir, desplazamiento lateral principal o normal, considerando una superficie homogénea y sin obstáculos mediante el uso del enfoque simplificado de osciladores sinusoidales de frecuencia fija presentados en la sección 2.2.2. Primero, empleando el modelo en tiempo continuo, suponiendo que el sistema consta de un único módulo M y posteriormente, en tiempo discreto teniendo en cuenta la distancia existente entre los módulos.

2.3.2 Modelo Matemático Desplazamiento Lateral Principal

2.3.2.1 Sistema en Tiempo Continuo

Según el modelo matemático basado en CPG presentado por Gómez (Gonzalez Gómez, 2008), el movimiento lateral principal o configuración tipo serpiente está definido por los parámetros $(\alpha_v, \alpha_h, k, \Phi_{vh})$ y permite que el robot se desplace hacia los lados sin variar la orientación de su eje corporal; emplea un sistema modular conformado por dos grupos de generadores independientes: horizontal o de viraje φ_{hi} y vertical o cabeceo φ_{vi}, que, al superponerse, forman ondas tridimensionales que varían de acuerdo a la fase entre ellas Φ_{vh}.

Estos grupos (Ec. 2-30, Ec. 2-31) se describen en base a la ecuación del generador sinusoidal (Ec. 2-8) como:

$$\varphi_{vi}(\phi) = A_v\, sen(\phi + (i-1)\Delta\phi_v), i \in \left\{1..\frac{M}{2}\right\}$$

Ec. 2-30

$$\varphi_{hi}(\phi) = A_h\, sen(\phi + (i-1)\Delta\phi_h + \Delta\phi_{vh}), i \in \left\{1..\frac{M}{2}\right\}$$

Ec. 2-31

Donde:

Símbolo	Variable	Símbolo	Variable
O_h	Offset horizontal	$\Delta\phi_h$	Diferencia de fase entre dos módulos horizontales consecutivos.
O_v	Offset vertical	$\Delta\phi_v$	Diferencia de fase entre dos módulos verticales consecutivos.
A_h	Amplitud horizontal	$\Delta\phi_{vh}$	Diferencia de fase entre módulo vertical y horizontal.
A_v	Amplitud vertical	Ψ_{hi}	Fase módulo horizontal.
φ_{hi}	Angulo de doblaje horizontal	Ψ_{vi}	Fase módulo vertical.
φ_{vi}	Angulo de doblaje vertical		

Tabla 2-6 Parámetros de la onda sinusoidal 3D en tiempo continuo

El resultado es una onda serpentinoide 3D que describe la forma del robot.

2.3.2.2 Onda Serpentinoide 3D

La curva serpentinoide 3D está descrita según las ecuaciones (Ec. 2-32, Ec. 2-33), en donde $\theta_h(s)$ y $\theta_v(s)$ son los ángulos de doblaje horizontal y vertical que varian sinusoidalmente con respecto a *s*:

$$\theta_h(s) = A_h\, sen\left(\frac{2\pi k_h}{l}s + \Psi_h\right)$$

Ec. 2-32

$$\theta_v(s) = A_v\, sen\left(\frac{2\pi k_v}{l}s + \Psi_v\right)$$

Ec. 2-33

Esta curva se forma por la superposición de dos ondas serpentinoides: la de cabeceo y la de viraje; cada una de estas tiene su propio ángulo de serpenteo y número de ondulaciones Figura 2-10.

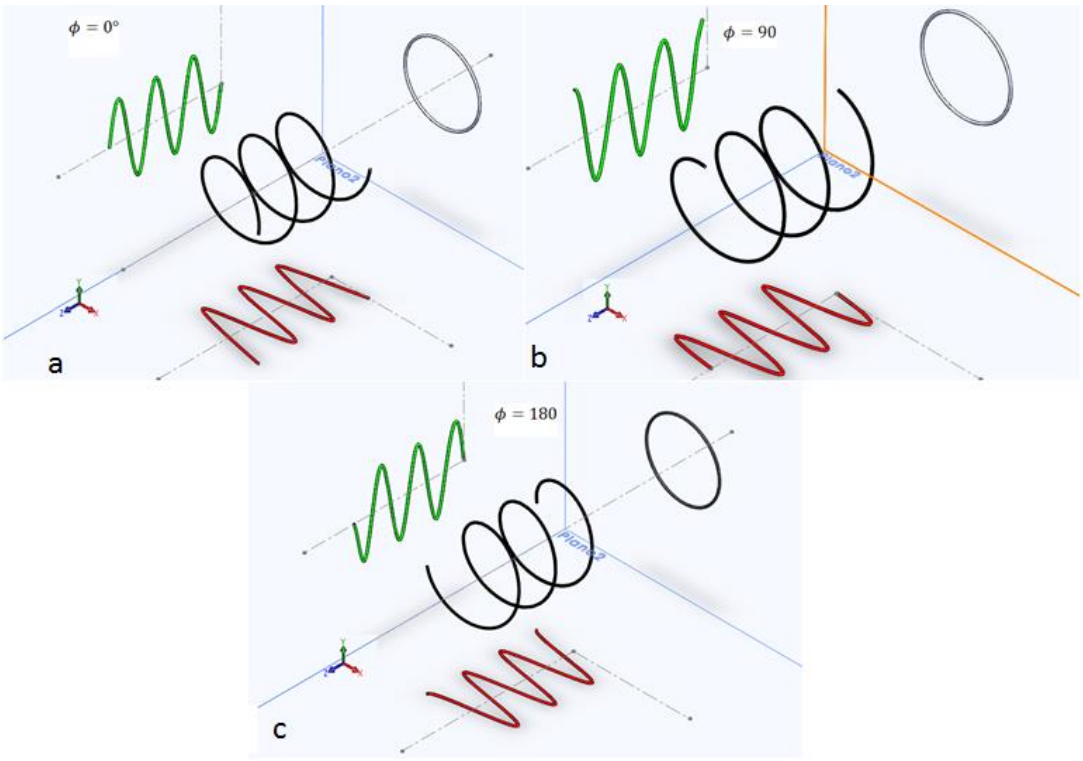


Figura 2-10 Ángulos de serpenteo k=3 a. 0° b. 90° c. 180°

Teniendo en cuenta la ecuación (Ec. 2-27) la expresión para la onda serpentinoide 3D se expresa como sigue (Ec. 2-34, Ec. 2-35):

$$\theta_v(s,\phi) = 2\alpha_v \, sen\left(\frac{2\pi k_v}{M}\right) sen\left(\phi + \frac{2\pi k_v}{l} s\right)$$

Ec. 2-34

$$\theta_h(s,\phi) = 2\alpha_h \, sen\left(\frac{2\pi k_h}{M}\right) sen\left(\phi + \frac{2\pi k_h}{l} s + \Delta\phi_{vh}\right)$$

Ec. 2-35

Donde:

$\theta_h(s)$: *Angulo de doblaje horizontal de un punto situado en s*

$\theta_v(s)$: *Angulo de doblaje vertical de un punto situado en s*

Las ondas serpentinoides 3D se dividen en isomorfas y no isomorfas.

- Isomorfas: Son aquellas en las que los parámetros k_vy $k_h$tienen el mismo valor, por tanto, los ángulos de doblaje de cabeceo y viraje son constantes.
- No isomorfas: Los parámetros k_vy k_h son diferentes entre sí.

La clasificación de las ondas isomorfas está dada por la proyección de la onda 3D en el plano *yz*, que dependiendo del ángulo de diferencia de fase horizontal-vertical da como resultado el trazo de las siguientes formas geométricas:

- Recta: Cuando $\Delta\phi_{vh} = 0°$ la onda se da sobre un plano y por tanto su proyección resulta en una línea recta formando un ángulo con el eje *y*.
- Circular: Cuando $\Delta\phi_{vh} = 90°$.
- Elíptica: Cuando $0 < \Delta\phi_{vh} < 90°$.

Por último, para definir la onda serpentinoide, es necesario determinar la relación $\frac{\alpha_v}{\alpha_h}$ que permite determinar la anchura y altura de la misma, es decir, si la relación es $\frac{\alpha_v}{\alpha_h} = 1$, la onda tiene la misma anchura que altura. Si $\alpha_v \ll \alpha_h$ se obtiene una onda plana, mientras que si $\alpha_v = 0$, se tiene una onda recta y su comportamiento se definiría como un robot tipo viraje - viraje.

2.3.2.3 Restricciones

- Los módulos de cabeceo en 2D manejan las mismas restricciones que los módulos de cabeceo en 1D encontradas en la sección 2.2.2.1.
- Los módulos de Viraje:
    - El offset O_h es nulo, para trayectorias rectas (oscilación simétrica).
    - Amplitud igual para todos los módulos.
    - La diferencia de fase entre dos módulos consecutivos es $\Psi_{hi} = (i-1)\Delta\phi_h + \Psi_{h1}$, donde Ψ_{hi} es la fase del módulo horizontal.
- El valor de fase para el módulo horizontal de referencia Ψ_{h1} , es igual a la suma entre la fase para el modulo vertical de referencia Ψ_{vi} y la diferencia de fase entre módulo vertical y horizontal $\Delta\phi_{vh}$ como sigue en la ecuación (Ec. 2-36).

$$\Psi_{h1} = \Psi_{v1} + \Delta\phi_{vh}$$

Ec. 2-36

2.3.2.4 Criterio de Estabilidad para locomoción 2D

El criterio de estabilidad para la locomoción en 2D empleado planteado por (Gonzalez Gómez, 2008) indica que deben existir al menos dos ondulaciones en el sistema robótico modular; igualmente, se emplea un principio válido para cualquier movimiento en 2D denominado principio de la onda plana, que se cumple cuando el ángulo de serpenteo vertical es mucho menor al ángulo de serpenteo horizontal $\alpha_v \ll \alpha_h$ y, por tanto, el centro de gravedad es más cercano al suelo, permitiendo que el movimiento tienda a ser estable y el sistema robótico modular no se derrumbe hacia un costado. Esto se ve representado por medio de la relación $\frac{\alpha_v}{\alpha_h}$ que para el caso de onda plana debe tender a cero.

$$\frac{\alpha_v}{\alpha_h} \to 0$$

Ec. 2-37

De esta manera las dimensiones del robot se pueden aproximar a la onda horizontal que recorre la cadena.

2.3.2.5 Sistema en Tiempo Discreto

Para poder implementar la curva serpentinoide 3D en el sistema robótico modular, es necesario discretizar teniendo en cuenta que está conformado por articulaciones de tipo cabeceo situadas en $s = (i-1)d + d_0$ y viraje en $s = (i-1)d + d_0 + \frac{d}{2}$ alternadamente. Los ángulos de doblaje se obtienen teniendo en consideración la ubicación de los puntos s en las articulaciones verticales y horizontales (Ec. 2-38, Ec. 2-39):

$$\varphi_{vi}(\phi) = \theta_v(\phi)|_{s=(i-1)d+d_0} = 2\,\alpha_v\, sen\left(\frac{2\pi k_v}{M}\right) sen\left(\phi + \frac{4\pi k_v}{M}\left(i - 1 + \frac{d_0}{d}\right)\right)$$

Ec. 2-38

$$\varphi_{hi}(\phi) = \theta_h(\phi)|_{s=(i-1)d+d_0+\frac{d}{2}} = 2\,\alpha_h\, sen\left(\frac{2\pi k_h}{M}\right) sen\left(\phi + \frac{4\pi k_h}{M}\left(i - 1 + \frac{d_0}{d} + \frac{1}{2}\right) + \Delta\phi_{vh}\right)$$

Ec. 2-39

Siguiendo el mismo principio de la ecuación Ec. 2-19 para estableces el número mínimo de módulos se tiene que, sí *M=2* entonces *k=1* y se obtiene una diferencia de fase de 180° lo que

no permite generar movimiento. Por esta razón deben existir al menos 3 módulos horizontales y al tener una configuración cabeceo-viraje se hace necesario tener el mismo número de módulos horizontales como verticales, por lo que la cadena debe tener un número mínimo total de módulos mayor o igual que 6.

Los valores de los parámetros de desplazamiento deben estar dentro de un rango para producir el desplazamiento lateral en el robot, estos rangos se establecen en la tabla

Símbolo	Variable	Rango
M	Número de módulos (horizontales y verticales)	$M \geq 6$
$\propto$	Ángulo de serpenteo	$\propto \leq \propto_{max} \leq 120$
$k = k_v = k_h$	Número de ondulaciones	$k \in [1, M/4]$

Tabla 2-7 Rango de valores de los parámetros para desplazamiento lateral principal

2.4 Conclusiones del capítulo

En este capítulo se presentó el concepto del modelo matemático empleado en este proyecto, se estudiaron las series de Fourier como representación del control basado en CPG, posteriormente se realizó el cambio de variables en base a la curva serpentinoide 2D para obtener el análisis matemático correspondiente a un sistema robótico modular en configuración tipo oruga (Cabeceo - Cabeceo), tanto en tiempo continuo, representando el sistema como un único módulo, como en tiempo discreto, tomando en consideración los múltiples módulos que conforman la cadena

Posteriormente se definieron las ecuaciones que describen el movimiento en configuración tipo serpiente (Cabeceo-Viraje), basadas en el modelo matemático desarrollado para la locomoción en 1D. Se toma en consideración que para esta configuración se tendrá un posicionamiento diferente de los módulos en la cadena, alternando módulos horizontales y verticales; por lo tanto, cada módulo según su orientación tendrá una ecuación distinta. Los módulos verticales ayudan a levantar la cadena de la superficie con una amplitud más pequeña que la de los módulos

horizontales y los módulos horizontales permiten el desplazamiento lateral de la cadena generando que el sistema cuente con dos grados de libertad.

Se delimitan los criterios de estabilidad para la locomoción, con el fin de asegurar que, durante su marcha el robot no va a tender a caer hacia un costado; y las restricciones y rangos en los que se deben encontrar los parámetros para conseguir el movimiento en configuración tipo oruga y tipo serpiente; por último, se obtienen las ecuaciones en tiempo discreto que son las que se utilizan en el sistema robótico modular MECABOT 3.0 tanto para la simulación como la implementación.

3 Antecedentes de Robótica Modular

En este capítulo se estudia la evolución de algunos sistemas robóticos modulares ápodos en los últimos 20 años con el fin de presentar el contexto histórico, los avances realizados en proyectos anteriores y la información que puedan aportar para MECABOT 3.0.

La robótica modular hace referencia a un grupo de sistemas robóticos compuestos por módulos pequeños, fácilmente interconectados. Éstos pueden ser de dos tipos, configurables, que cuentan con interfaces de conexión y auto-configurables, que son aquellos que basan su decisión en reglas geométricas para su accionamiento con módulos vecinos detectados, dando paso a la reconstrucción requerida (Rus, Salemi, Shen, & Yim, 2007).

Un sistema robótico modular presenta ventajas económicas y funcionales comparadas con las de un robot de estructura fija ya que puede ser reconfigurado morfológicamente según las condiciones en las que se encuentre, por medio de módulos adicionales que se ajusten al nuevo arreglo morfológico dando la posibilidad de que se reduzcan los costos de inversión. (Althoff, Giusti, & Icer, 2016).

La investigación en robótica ha estado mayormente ligada a la idea de controlar ensamblajes completos compuestos por piezas modulares estáticas ya que, al estar especificados para una única tarea, su rendimiento es mayor que el de un robot modular, razón por la cual se ha dejado de lado el estudio de estos. Por ejemplo, a nivel Colombia se registra en bases virtuales como investigaciones más recientes en robótica modular los siguientes documentos:

- “Diseño y Simulación de un Algoritmo para el Control de un Robot Modular tipo Cadena” de la Universidad Nacional de Colombia en el año 2010. (García, 2010)
- “Preliminary Studies on Modular Snake Robots applied on de-mining Tasks” de la Pontificia Universidad Javeriana en el año 2011 (Hernandez, y otros, 2011)
- “A distributed model predictive control (D-MPC) for modular robots in chain configuration” de la Pontificia Universidad Javeriana en el año 2011 (Cortes, Linares, Melo, & Patino, 2011).

- “Central pattern generators and hormone inspired messages: A hybrid control strategy to implement motor primitives on chain type modular reconfigurable robots” de la Universidad Nacional de Colombia en el año 2011 (Melo, y otros, 2011).
- "Center of mass displacements using rolling gaits for modular robots on the outside of pipes" de la Pontificia Universidad Javeriana en el año 2011 (Paez, Melo, & Parra, 2011).
- “Indoor and outdoor parametrized gait execution with modular snake robots” de la Pontificia Universidad Javeriana en el año 2012 (Parra, Melo, & Paez, 2012).
- “Conceptual design of a modular snake origami robot” de la Pontificia Universidad Javeriana en el año 2013 (Granados, Melo, & Paez, 2013).
- “Diseño y simulación de un robot modular reconfigurable” de la Universidad Militar Nueva Granada en el año 2013 (Hurtado Erasso & Rubiano Montaña, 2013)

El último trabajo encontrado data del año 2013 lo que implica que, sí se ha continuado con el desarrollo de estos proyectos no se encuentra en las bases virtuales y por tanto no se puede evidenciar un avance significativo. Además de esto, las pocas referencias que se encuentran concernientes a investigación en robótica modular en Colombia, demuestran una carencia de interés e inversión en este campo. Sin embargo, la idea de su implementación es un proyecto bastante prometedor.

Por el lado de simulación, se tienen avances exitosos ya que por medio de algoritmos descentralizados y genéticos se ha logrado hacer el control para diversos robots. Al hablar de algoritmos descentralizados se hace referencia a la independencia algoritmo-plataforma, por medio de la toma de decisiones del módulo autónomo que, por diferentes métodos de aprendizaje, permiten síntesis automática de reglas de control. (Kirkner, 2007). Algunos avances en el campo de la robótica modular según su capacidad de configuración:

3.1 Sistemas Robóticos Modulares No Auto-Reconfigurables

3.1.1 POLYBOT G1 (1997)

Figura 3-1 Robot POLYBOT G1en configuración rueda y serpiente
Tomado de: (Duff, Roufas, & Yim, 2000)

Este sistema robótico modular utiliza tablas de control para la generación de movimiento, tiene un control centralizado en Bucle abierto con el cual consigue movimiento tipo serpiente, rueda y araña. Sin embargo, tiene limitaciones al no poder almacenar todos los posibles movimientos que aumentan a medida que se añaden más módulos. Esta generación (G1) no tiene la capacidad de conexión dinámica, su conexión es por medio de tornillos. Son cubos de 5cm de arista y tienen un grado de libertad por módulo, las piezas son hechas de plástico y cuentan con un servomotor RC ubicado en la parte inferior de la estructura, no tienen ningún sensor, la alimentación y electrónica está ubicada fuera del robot. La conexión en fase (módulos conectados con la misma orientación) y desfase (módulo rotado 90° con respecto al anterior) permite la construcción de robots ápodos (Duff, Roufas, & Yim, 2000).

3.1.2 DEFORMATRON (2006)

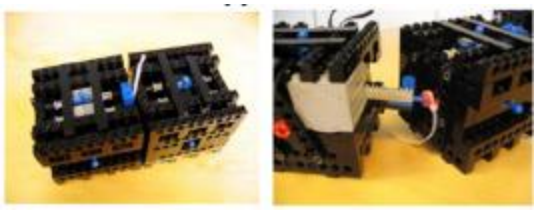


Figura 3-2 Módulos acoplados DEFORMATRON
Tomado de: (Stoy, 2006)

DEFORMATRON es un robot modular homogéneo tipo LEGO (ver Figura 3-2), en el que se busca representar la facilidad de manipulación de su entorno mediante estructuras semejantes a las biológicas como los tendones, huesos y músculos de los seres vivos. Los módulos constan de seis conectores extensibles centrados en cara que proporcionan conexiones rígidas o flexibles entre ellos. Su principal ventaja es que los bloques pueden ser combinados para producir estructuras con potencia de accionamiento proporcional al número de módulos interconectados, así como la facilidad de producir movimientos de traslación y rotación (Stoy, 2006).

3.1.3 CGZ-I (2008)

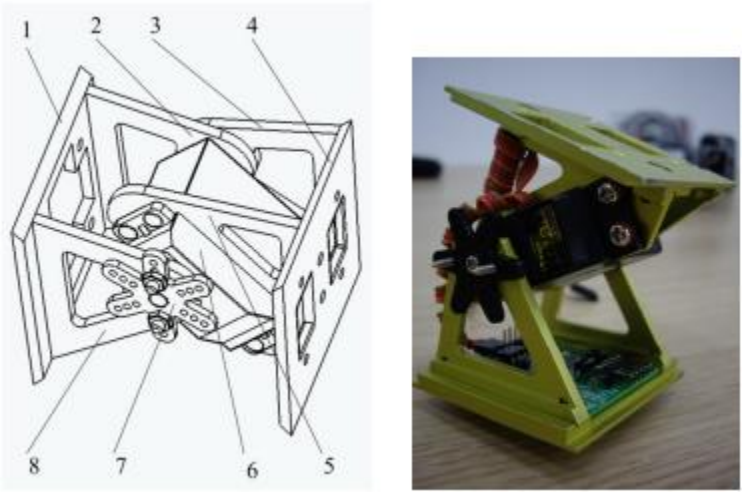


Figura 3-3 Diseño CGZ-I1.
Tomado de: (Cheng, Gonzalez-Gomez, Jianwei, Xie, & Zhang, 2008)

Robot modular de bajo costo, esta es la versión mejorada del robot modular Y1, cuenta con una estructura mecánica de fácil construcción, cuatro caras de conexión, un controlador eléctrico, algoritmo de control eficiente para activar sus articulaciones, conexión de forma manual y entorno

de programación amigable. Cada módulo posee un grado de libertad, al conectar en diferentes modos se activan las articulaciones por medio de un servomotor RC dotándolos con la capacidad de cambiar de forma en 2D, se controla por medio de un oscilador con parámetros tales como amplitud, frecuencia, fase y desplazamiento (Cheng, Gonzalez-Gomez, Jianwei, Xie, & Zhang, 2008).

3.1.4 3D – TRUNK (2009):

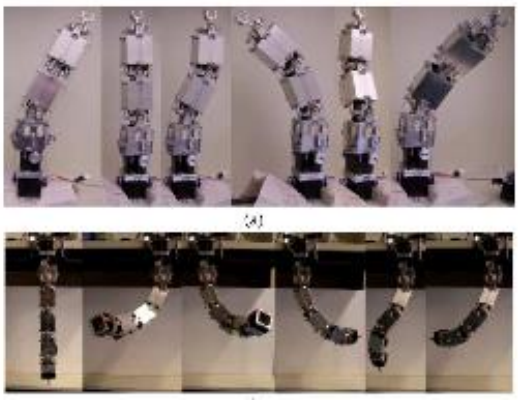


Figura 3-4 3D TRUNK
Tomado de: (KeJun & Wörgötter, 2009)

El 3D – TRUNK está basado en el funcionamiento de los músculos empleando mecanismos de tensión para realizar los movimientos. Como se observa en la Figura 3-4, consta de bloques unidos en serie por junturas de revolución, contienen motorreductores, tarjetas de control, unidades de energía y sistemas de comunicación. Como resultado se obtiene un robot de bajo peso, con gran facilidad para cambiar de forma (KeJun & Wörgötter, 2009).

3.1.5 MECABOT (2013)

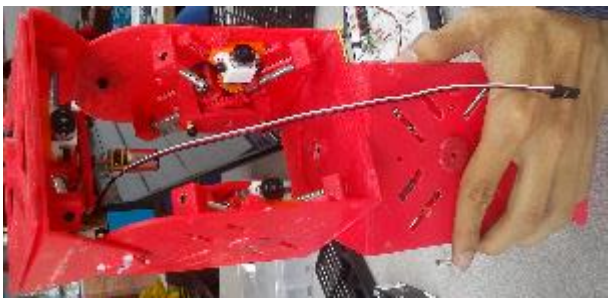


Figura 3-5 Sistema Robótico modular MECABOT
Tomado de: (Hurtado Erasso & Rubiano Montaña, 2013)

Sistema robótico modular desarrollado en la Universidad Militar Nueva Granada, su sistema de acople consiste en cuatro pestañas en forma cilíndrica ubicados en las ruedas, dos de ellas de recepción y las otras dos de inserción, esto con el fin de asegurar las piezas acopladas. En cuento al diseño electrónico, cuenta con un microcontrolador PIC24HJ12GP201 de Microchip, un Driver Micro Maestro 6 Pololu para controlar los 5 Micro servomotores que tiene cada módulo, sensores de distancia SHARP GP2Y0D810Z0F y un módulo de comunicación RF Transceptor +0dBm, utiliza 4 baterías de polímero de iones de litio. Presenta como desventaja su gran tamaño y sus motores que no tienen la capacidad para conseguir la locomoción deseada (Hurtado Erasso & Rubiano Montaña, 2013).

3.1.6 SEA SNAKE (2014)

Figura 3-6 SEA Snake
Tomado de: (Bilgen, y otros, 2014)

SEA Snake está conformado por 16 módulos como se ilustra en la Figura 3-6, cada uno con una juntura de 1DOF que permite una rotación de 180°, y dos módulos especiales correspondientes a la cabeza y cola del robot. Está diseñado en aluminio anodizado y cumple con los estándares IP66 que lo hace resistente a filtraciones de agua. También cuenta con conectividad a Ethernet, RS 485, encoders, sensores inerciales, sensores de voltaje, corriente y temperatura. Para su movimiento emplea motores sin escobillas (Bilgen, y otros, 2014).

3.1.7 MECABOT 3.0 (2014)

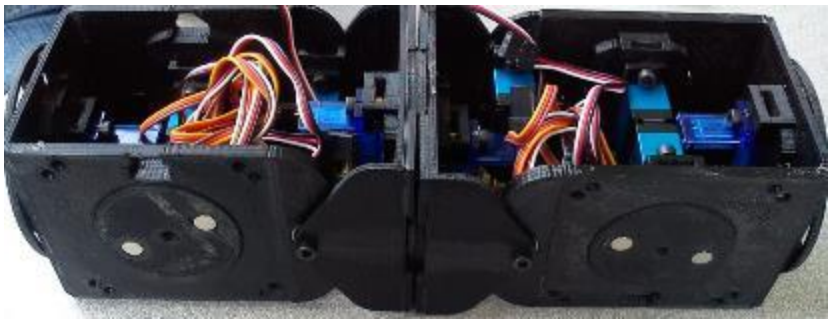


Figura 3-7 Sistema Robótico Modular MECABOT 3.0
Tomado de: (Pardo Puentes, 2015)

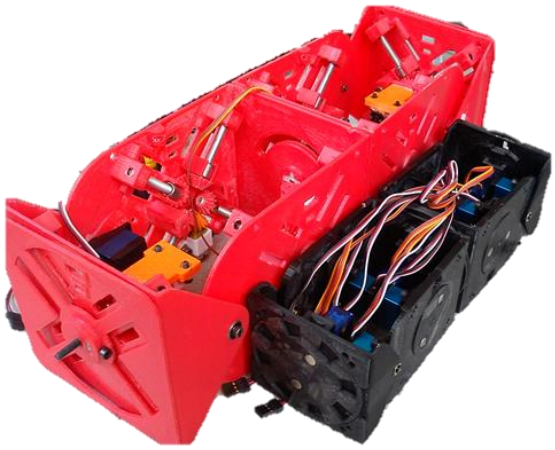


Figura 3-8 Comparación MECABOT vs. MECABOT 3.0
Tomado de: (Pardo Puentes, 2015)

Tercera generación del sistema robótico modular MECABOT desarrollado en la Universidad Militar Nueva Granada con el fin de ser empleado en operaciones de búsqueda y rescate urbano. A diferencia de su predecesor, implementa el concepto de "Unidad Modular Autónoma AMU", es decir, cada módulo puede ser empleado como un único robot móvil independiente. MECABOT 3.0 se enfoca en la reducción de tamaño y peso a casi la mitad de la primera versión, la estructura externa se produce en ABS gracias a la impresión 3D; cuenta con ruedas laterales que le aportan dos grados de libertad. El acople pasa de ser mecánico a magnético mediante imanes permanentes. En cuanto al sistema electrónico MECABOT 3.0 cuenta con Arduino nano para el procesamiento, comunicación por radiofrecuencia mediante XBee, sensores de aceleración y distancia, micro servomotores y alimentación por batería de litio.

3.2 Sistemas Robóticos Modulares Auto-Reconfigurables

3.2.1 M-TRAN I (1998)

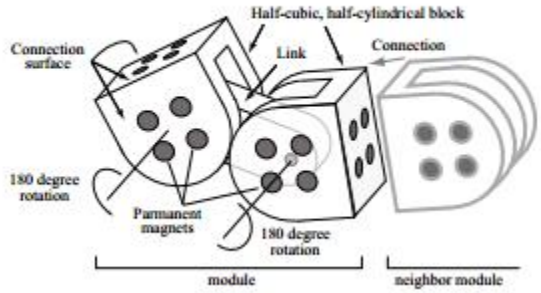


Figura 3-9 Módulo M-TRAN I
Tomado de: (Kamimura, Kokaji, Kurokawa, Tomita, & Yoshida, 2005)

El robot se compone de dos piezas semicilíndricas y un enlace como se observa en la Figura 3-9. Cada pieza semicilíndrica tiene imanes que dan conexión ortogonal a los módulos permitiendo una amplia variedad de configuraciones; cada bloque puede rotar alrededor de su eje ± 90 de forma independiente gracias a un motor ubicado dentro del enlace.

Es un robot modular auto-reconfigurable, controlado a través de un enlace por cable con el PC maestro, una simulación cinemática y un controlador centralizado basado en el concepto de CPG (Central Pattern Generator) para procesar una secuencia de movimientos primitivos, los módulos trabajan como esclavos bajo las órdenes enviadas desde el computador maestro por medio de una interfaz comunicación serial UART. Este proyecto se simuló como un grupo de módulos para reconfigurase en diferentes estructuras estáticas, permitiendo movimientos dinámicos como un sistema completo (Kamimura, Kokaji, Kurokawa, Tomita, & Yoshida, 2005).

3.2.2 POLYBOT G2 (2000)

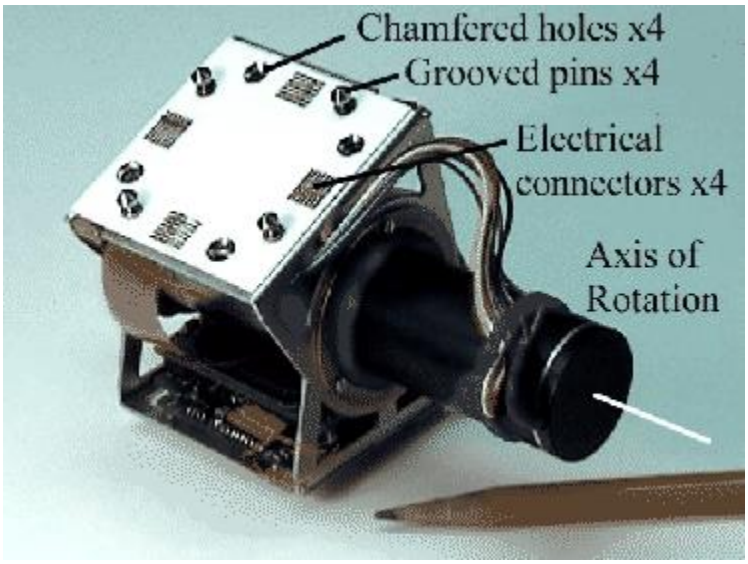


Figura 3-10 Diseño POLYBOT G2
Tomado de: (Duff, Roufas, & Yim, 2000)

Es la segunda generación de POLYBOT mencionado en la sección 3.1 son módulos auto-reconfigurables y tienen dos tipos de módulos: nodos y segmentos. Utilizan un motor de corriente continua sin escobillas en lugar se servomotores RC, con rango de giro es de -90° a 90°; una estructura metálica que lo hace más robusto, dos placas de conexión para unirse con más módulos por medio de ganchos y los conectores metálicos permiten alimentación y comunicación. Por otro lado, cada módulo tiene un Power-Pc 555 de Motorola y se comunican a través del BUS CAN.

El experimento implementado en estos módulos fue la auto-reconfiguración pasando de tipo rueda a tipo serpiente y finalmente el levantamiento del robot para formar una araña de cuatro patas (Duff, Roufas, & Yim, 2000).

3.2.3 PROTEO (2000)

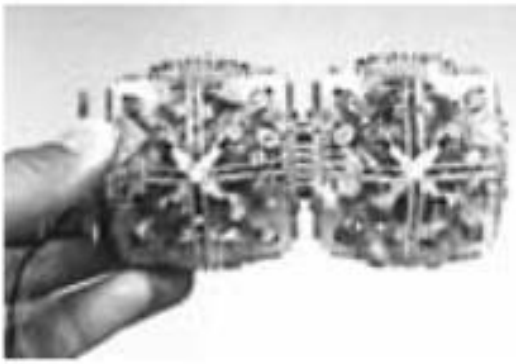


Figura 3-11 Módulo PROTEO
Tomado de: (Bojinov, Casal, & Hogg, 2000)

Proteo es un robot auto-reconfigurable desarrollado en Xerox PARC. Como se puede observar en la Figura 3-11 está conformado por múltiples robots modulares homogéneos con forma de dodecaedros rómbicos, cada módulo tiene doce caras de conexión idénticas y electroimanes para conectarse con el siguiente módulo. Es capaz de trasladarse gracias aristas rotacionales y de adaptarse para dar respuesta a cambios medioambientales (Bojinov, Casal, & Hogg, 2000).

3.2.4 CONRO (2002):

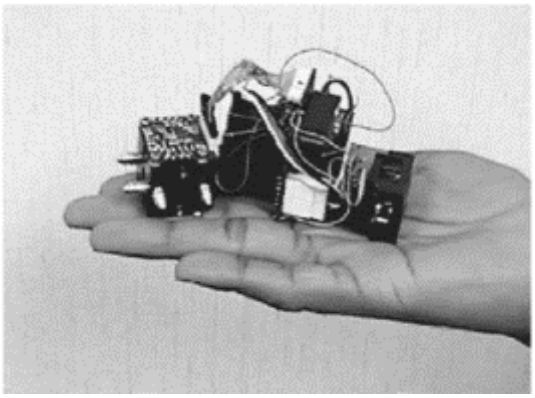


Figura 3-12 Proyecto CONRO
Tomado de: (Behar, Castano, & Will, 2002)

El proyecto CONRO se ilustra en la Figura 3-12, es un robot auto-reconfigurable, compuesto de módulos homogéneos con dos grados de libertad, mecanismo de cierre que permite la conexión de los módulos por medio de pines y agujeros alineados. Los módulos pueden adoptar

diferentes estructuras como hexápodo o serpiente. Consta de tres segmentos: Conector pasivo, cuerpo del módulo y conector activo. Cada módulo con procesador, fuente de energía, sensores, actuadores y sistemas de comunicación infrarrojo con el fin de garantizar la autosuficiencia del robot. Como resultado, los módulos son capaces de reconfigurarse con eficiencia gracias a la implementación de un control híbrido (distribuido y centralizado) (Behar, Castano, & Will, 2002).

3.2.5 M – TRAN II (2002):

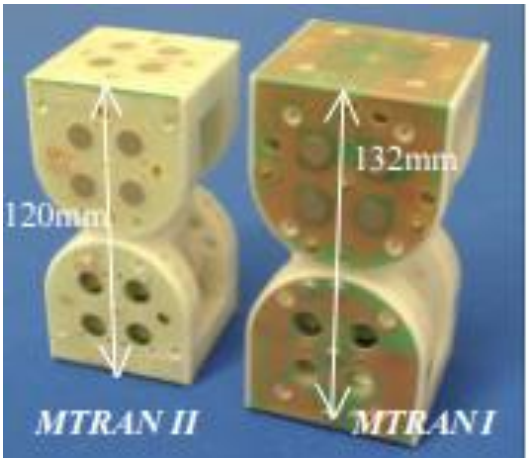


Figura 3-13 Comparación entre los dos prototipos
Tomado de: (Kamimura, Kokaji, Kurokawa, Tomita, & Yoshida, 2005)

Es el segundo prototipo del robot M-TRAN mencionado en la sección 3.2.1 La comparación entre los dos prototipos se puede observar en la Figura 3-13. Con respecto al primer prototipo, M-TRAN II cuenta con cambios como: reducción del tamaño, reducción del peso, mejoras de operación autónoma, menor consumo de energía, control más preciso de los motores, y mejor procesamiento de la información. M-TRAN II es autoalimentado, tiene su propio controlador y un bus de red para comunicarse con otros módulos. En cuanto al software se desarrolló un algoritmo de generación de patrones de locomoción (ALPG Automatic Locomotion Pattern Generation Software), y se implementó en el software Vortex (CM Labs Simulation, Inc.), un modelo matemático del robot y un controlador por redes neuronales basado en un modelo CPG (Central Pattern Generator). El controlador permite el movimiento de las junturas y transmite la dinámica de los movimientos a Vortex para lograr la simulación. Como resultado del estudio se obtuvo el algoritmo para la generación de movimientos estables para cualquier estructura modular mediante la red CPG y la simulación gracias al ALPG (Akiya Kamimura, 2005).

3.2.6 M-TRAN III (2005)

Figura 3-14 Comparación entre las tres generaciones de M-TRAN
M-TRAN I (izquierda), M-TRAN II (centro) M-TRAN III (derecha)
Tomado de: (Kamimura, Kokaji, Kurokawa, Tomita, & Yoshida, 2005)

Es el tercer prototipo de M-TRAN, es un sistema robótico auto-reconfigurable tipo híbrido. En comparación con el anterior prototipo M-TRAN II mencionado en la sección 3.2.6, tiene mejoras relativas a la velocidad y fiabilidad de la conexión. Este prototipo no utiliza conexión por medio de imanes como las anteriores versiones, sino que hace uso de una conexión mecánica que es mucho más rápida y requiere menor potencia. Cada módulo cuenta con cuatro microcomputadores: un maestro y tres esclavos. Todos los microcomputadores maestros ubicados en los módulos están conectados por bus CAN, para comunicase entre ellos y sincronizar sus movimientos. Cuenta con comunicación inalámbrica vía Bluetooth y sensores de proximidad infrarrojo (Kamimura, Kokaji, Kurokawa, Tomita, & Yoshida, 2005).

3.2.7 SUPERBOT (2006)

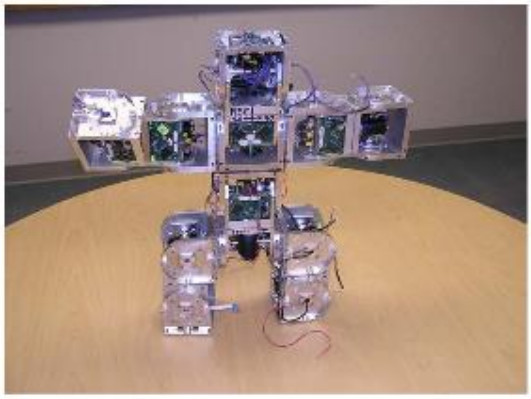


Figura 3-15 SUPERBOT
Tomado de: (Salemi, Moll, & Shen, 2006)

Los módulos SUPERBOT (ver Figura 3-15) constan de tres grados de movilidad que le brindan ventajas de movilidad, flexibilidad, auto-reconfiguración y eficiencia al momento de ejecutar tareas, son capaces de compartir energía y comunicarse por medio de LED infrarrojos. Puede trabajar en ambientes hostiles y cambiantes en el tiempo. Tiene la capacidad de mover y cargar cierto número de módulos vecinos y de reconocer variables de consideración en su entorno (Salemi, Moll, & Shen, 2006).

3.2.8 ATRON (2007):

Figura 3-16 ATRON
Tomado de: (Brandt, Christensen, & Hautop Lund, 2007)

ATRON ilustrado en la Figura 3-16, es un sistema Auto-reconfigurable compuesto por 100 módulos robóticos interconectados de un alto grado de autonomía, capaces de tener 90° de rotación, se pueden conectar y desconectar de los módulos vecinos, comunicarse entre sí, sensar orientaciones relativas, distancia entre objetos y soportar hasta dos módulos (Brandt, Christensen, & Hautop Lund, 2007).

3.2.9 SINGO (2009):

Figura 3-17 SINGO
Tomado de: (Kova, Rubenstein, & Shen, 2009)

SINGO ilustrado en la Figura 3-17, se enfoca principalmente en optimizar las aplicaciones de auto-reconfiguración, auto-ensamble y auto-reparación de un robot modular. Cuenta con un mecanismo de unión flexible que se adapte a las necesidades de movimiento del robot. El diseño se basa en un sistema de dos conectores sin-género de tal forma que su conexión sea "universal" (Kova, Rubenstein, & Shen, 2009).

3.2.10 CoSMO (2013)

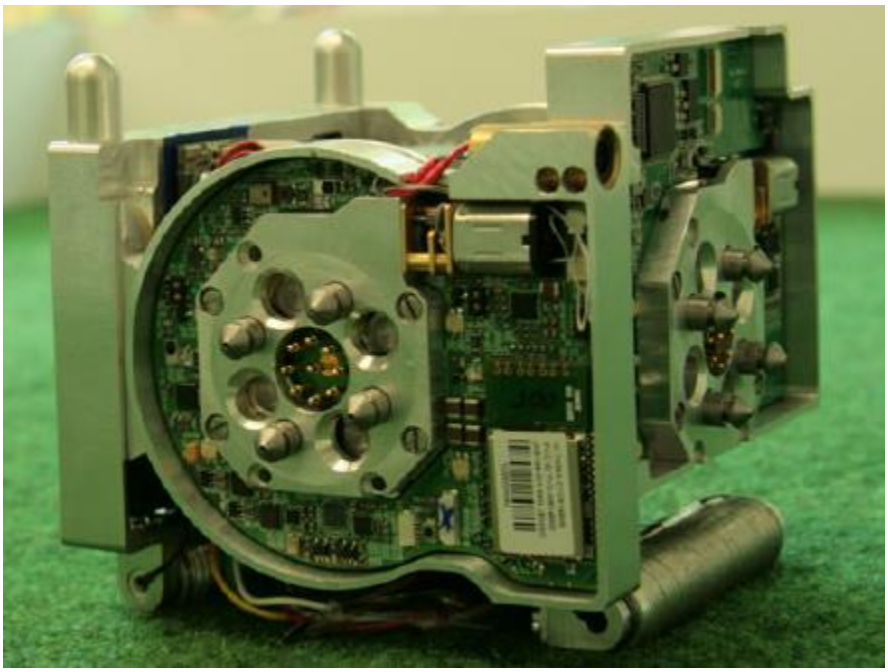


Figura 3-18 Primer prototipo de un módulo CoSMO
Tomado de: (Liedke, Matthias, Winkler, & Wörn, 2013)

Robot auto-reconfigurable, se compone de varios bloques uniformes y es desarrollado dentro de dos proyectos europeos REPLICATOR y SYMBRION. En comparación con otros sistemas robóticos modulares, este cuenta con alta capacidad computacional, exhibe una comunicación con un considerable ancho de banda para la comunicación entre los módulos conectados y los módulos comparten energía entre sí. Su diseño mecánico lo hace flexible para realizar diferentes tareas y adaptarse a cambios imprevistos en el medio ambiente. Sin embargo, el tamaño y masa de los módulos del robot son todavía comparables con otras plataformas. Las tarjetas de circuito impreso se encuentran en las paredes, protegidas por un marco de metal; en el interior se encuentra la batería y la unidad de accionamiento principal. Utilizan un motor DC sin escobillas que es el más adecuado para el intercambio de energía entre robots y dentro de este se ubican sensores Hall para la medición de posición y control (Liedke, Matthias, Winkler, & Wörn, 2013).

3.3 Conclusiones del capítulo

En este capítulo se presenta el estado del arte relativo a algunas de las investigaciones desarrolladas en el campo de la robótica modular, clasificados según su capacidad de auto-reconfiguración, destacando las principales características de diseño mecánico y diseño electrónico; modo de alimentación, modo de conexión, grados de libertad, comunicación entre módulos, sensores y control. Se evidencia la evolución de los sistemas robóticos modulares M-TRAN, POLYBOT y MECABOT con sus respectivas mejoras y ajustes y se mencionan problemas de diseño que presentan algunos de los proyectos. Lo anterior con el fin de tener un precedente de la evolución que ha tenido la robótica modular y poder tomarlo como referente a la hora de desarrollar el presente trabajo de grado.

4 Simulación del Sistema Robótico Modular MECABOT 3.0

4.1 Introducción

En este capítulo se presenta el software Webots desarrollado por Cyberbotics©, elegido para la simulación del sistema robótico modular MECABOT 3.0, sus características principales, herramientas y definiciones necesarias para la creación de la simulación. Luego, se muestra el diseño de los semi-módulos MECABOT 3.0 detallando las piezas y se describe el proceso de importación de la geometría al nodo *world* (mundo) de Webots desde SolidWorks.

La obtención de la velocidad se logra a partir del contador de tiempo de simulación del software, la ubicación de un marcador en el origen del sistema robótico modular y el uso de un piso métrico.

El análisis de velocidad con la configuración tipo oruga se realiza para las tres conexiones posibles: Pivote - Centro, centro-centro, Pivote - Pivote obteniendo las respectivas tablas y gráficas; posteriormente, se simula la configuración tipo serpiente para las conexiones Pivote - Centro, centro-centro, Pivote - Pivote con las que se obtienen las tablas y graficas con valores de velocidad que permiten concluir un rango de funcionamiento adecuado del sistema para su posterior implementación.

4.2 Webots

Webots es un simulador que proporciona un ambiente de prototipado rápido para crear mundos virtuales en 3D con propiedades físicas que permiten definir: Masa, coeficiente de fricción, densidad, entre otras. El usuario puede agregar objetos pasivos o activos dentro del nodo mundo (*world*); los objetos activos son llamados robots móviles y pueden tener diferentes formas de locomoción, según se requiera; además, es posible emplear sensores y actuadores, cámaras, motores, emisores, receptores, entre otros, que el software suministra para definir el comportamiento del robot. Dentro de Webots es posible realizar simulaciones de prototipaje de robots móviles, locomoción, comportamiento adaptativo etc. Para la definición del comportamiento del robot dentro del software, se emplean lenguajes como: C, C++, Java, Python

y MATLAB®; para mostrar el comportamiento deseado. Webots brinda varios ejemplos de modelos de robots y controladores para facilitar al usuario la comprensión del funcionamiento del programa. Para el desarrollo de la simulación del presente proyecto de grado, se emplea el lenguaje de código C puesto que la documentación provista por Webots se encuentra enfocada principalmente a éste.

Los principales conceptos con los que trabaja Webots se definen a continuación (Cyberbotics Ltd., 2014) (Cyberbotics Ltd., 2014):

World (Mundo): Es la descripción en 3D de las propiedades de los robots y su ambiente, es decir, contiene la información de cada objeto como: posición, orientación, geometría, apariencia, propiedades físicas, etc. Está organizado en estructuras jerárquicas en donde los objetos contienen otros objetos, por ejemplo, un robot puede contener dos articulaciones que a su vez contienen un motor, etc. El nodo *world* especifica en nombre del controlador utilizado por cada robot.

Controller (Controlador): Programa que controla el robot en un archivo *world* (.wrl) especifico, pueden ser escritos en cualquier lenguaje de programación soportado por Webots. AL iniciar la simulación, Webots lanza el controlador en procesos separados asociando el controlador a la simulación que este corriendo. Los lenguajes C y C++ únicamente necesitan ser compilados en ejecutables (.exe, para Windows), Python y MATLAB®™ requieren su propio intérprete y Java requiere tanto compilación (.class o .jar) como interprete (Java Virtual Machine).

Nodes (Nodos): Webots trabaja bajo una estructura de herencia como se mencionó anteriormente, es por esto que los nodos son elementos agregados al robot para ir dando forma, propiedades y funcionalidad al mismo, de esta forma nodo *Robot* por ejemplo deriva en nodos que indican su geometría, apariencia, propiedades, sensores, actuadores, articulaciones, entre otras. Algunos de estos se explican de forma general:

- *Group (*Grupo*)**:*** Este agrupa nodos Transform y contienen consecutivamente la forma de cada parte para al final formar un ensamble total.

- *Hingejoint (*Bisagra*)*: Es el nodo más usado en la categoría de *Joints*, este permite agregar al modelo, movimiento rotacional alrededor de un eje dado (un grado de libertad).
- *Joint (Articulación)*: Es usado para agregar uno o más grados de libertad entre nodos padres y nodos hijos que deben ser cuerpos sólidos. Los nodos que derivan de *Joint* permiten crear diferentes tipos de restricciones entre los enlaces de los nodos sólidos.
- *Mecanismo Def-Use:* Este mecanismo es usado en Webots para evitar tener redundancia en el archivo. Permite definir un nodo en un lugar y poder ser reusado en otra parte, esto con el fin de no tener duplicados de nodos idénticos y que puedan ser modificados todos los nodos al mismo tiempo desde DEF. Este mecanismo depende del orden en que vayan apareciendo los nodos en el archivo, por lo que el nodo DEF debe aparecer primero para poder ser usado en otras partes.
- *Physics (*Física*)***:** Permite especificar parámetros para el motor de simulación física. Este contiene datos de masa, centro de gravedad y masa de distribución; con los que se proporciona información para crear las propiedades físicas del cuerpo y simular fuerzas de manera real.
- *Robot:* La principal estructura de un modelo *Robot* es un árbol compuesto por nodos de solidos vinculados a través de nodos *Joints*.
- *Shape (*Forma*)*: Está compuesto por dos archivos; apariencia y geometría, estos son usados para crear objetos en el mundo. La apariencia contiene un nodo llamado *Appereance* en donde se especifican los atributos visuales como material y textura, que se aplican a la geometría. Por otro lado, en la geometría *Geometry*, se agrega el nodo que se requiera como: *Box (*Caja*), Capsule (*Envoltura*), Cone (*Cono*), Cylinder (*Cilindro*), ElevationGrid*

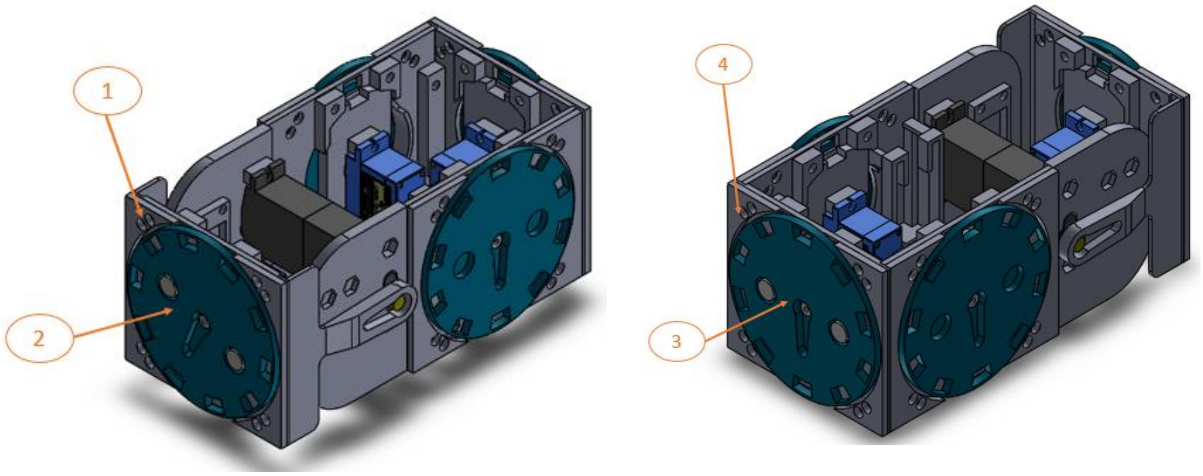
*(*Rejilla de elevación*), IndexedFaceSet (*Conjunto de caras indexada*), IndexedLineSet (*Conjunto de líneas indexadas*), Plane (*Plano*) o Sphere (*Esfera*).*

- *Transform (Transformada):* Este nodo agrupa los nodos que definen la coordinación del sistema tales como: Translación, rotación y escala.

4.3 Diseño Semi-Módulos MECABOT 3.0

Para realizar la simulación del sistema robótico modular, se requiere la construcción de virtual de MECABOT 3.0, en esta sección se muestran los componentes necesarios únicamente para la implementación en Webots, para la descripción en detalle de MECABOT 3.0 ver la sección 5 Implementación Sistema Robótico Modular MECABOT 3.0.

Los semi-módulos MECABOT 3.0 empleados para la importación a Webots son simplificados, con el fin de reducir el tiempo de simulación ya que, al tener todos los componentes y detalles mecánicos, el software WEBOTS tarda más tiempo en procesar toda la información en Figura 4-1 se aprecia los semi-módulos con todos los componentes.



1.Cara Pivote. 2.Rueda pivote.

3. Rueda centro. 4. Cara Centro.

Figura 4-1 Piezas principales de MECABOT 3.0

Para realizar la importación de una geometría desde el software de diseño SolidWorks al entorno de Webots es necesario seguir la secuencia de pasos descrita a continuación:

Diseñar la geometría deseada en SolidWorks y guardar el documento como extensión VRML (. wrl). En la Figura 4-2 se ilustra el diseño de una pieza del ensamble.

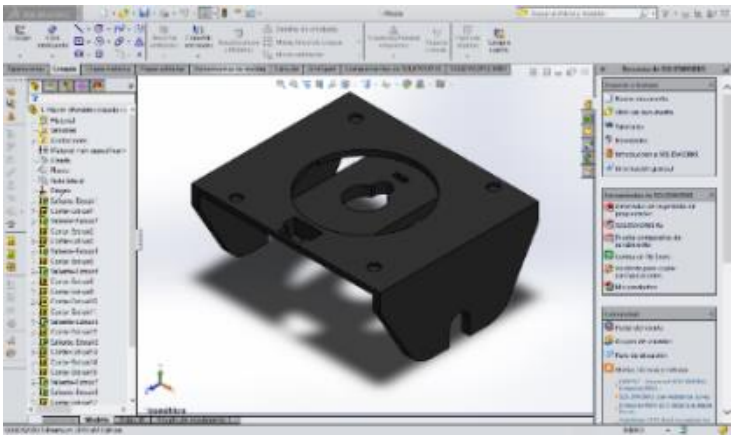


Figura 4-2 Diseño de geometría en SolidWorks

En el simulador Webots, en la opción *File (*Archivo*)*, seleccionar *Import* VRML 2.0 con el fin de importar la geometría, que aparece dentro de un nodo *Transform (*Transformada*)* en el árbol. Para poder ensamblar los componentes o simular el comportamiento de la geometría importada se debe añadir el nodo *Transform* generado a un nodo nuevo: *Robot*. Dentro del nuevo nodo *Robot*, se agrega como un *children (*Hijo*)* el nodo *Transform* resultante de la importación

- Como resultado, se observa la geometría deseada en el espacio de trabajo, como se ilustra en la Figura 4-3.

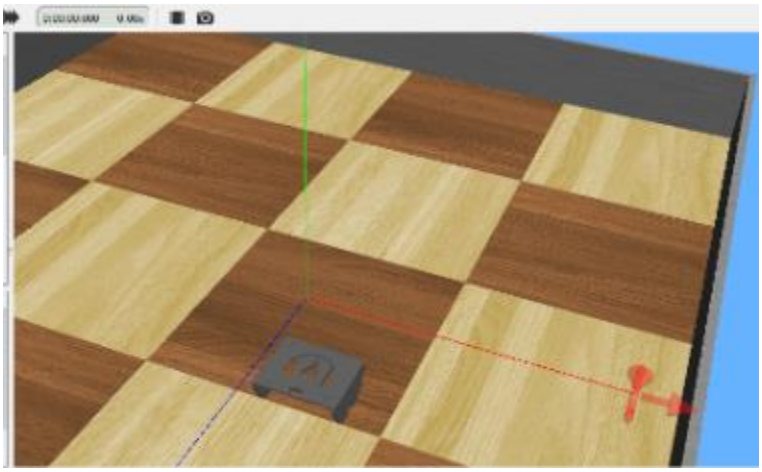


Figura 4-3 Geometría importada en Webots

Por ultimo al tener importadas todas las piezas se crean relaciones de posición y movimiento para la correcta simulación.

4.4 Conexiones de los Módulos en Configuración Tipo Oruga

Gracias al diseño de los módulos, es posible realizar 3 diferentes conexiones a partir de la unión entre ellos, para definir estas conexiones hay que tener presente el concepto de módulo visto en la sección 2.1.5. Estas conexiones son:

4.4.1 Conexión Pivote – Centro

Se logra a partir de la unión entre la cara del pivote de un semi-módulo, con la cara Centro del siguiente, teniendo como resultado la conformación de un módulo *M* como se ve en la Figura 4-4.

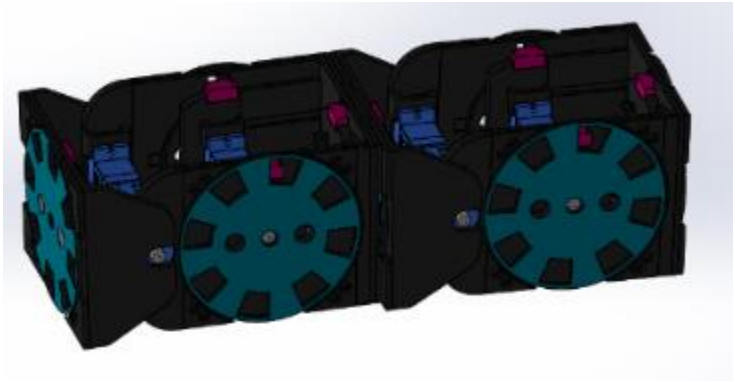


Figura 4-4 Módulo en configuración Pivote – Centro
posición vertical (configuración oruga)

En la Figura 4-5 se puede detallar la enumeración de los módulos en una cadena configuración tipo oruga, conexión Pivote Centro.

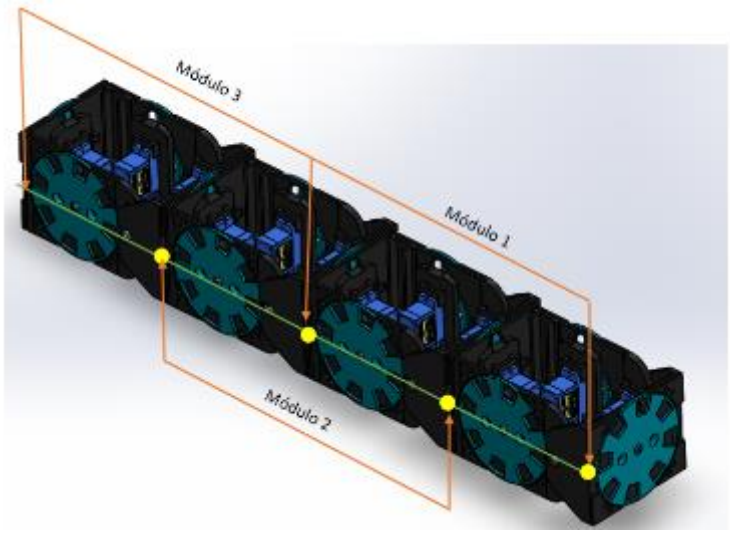


Figura 4-5 Enumeración de módulos Configuración oruga, conexión Pivote – Centro.

4.4.2 Conexión Pivote – Pivote

Se logra a partir de la conexión entre las dos caras Pivote, formando una módulo *M* como se ve en la Figura 4-6.

Figura 4-6 Módulo en Configuración Pivote – Pivote posición vertical (configuración oruga)

En la Figura 4-7 se evidencia la enumeración de los módulos, los círculos amarillos representan las articulaciones provistas por el pivote y las líneas verdes los segmentos, esta cadena está compuesta por dos módulos (*M=2*). Para este tipo de conexión un módulo está compuesto por dos semi-módulos, entonces, dependiendo del número de módulos que se requiera en la cadena, es necesaria 2 veces esta cantidad en semi-módulos.

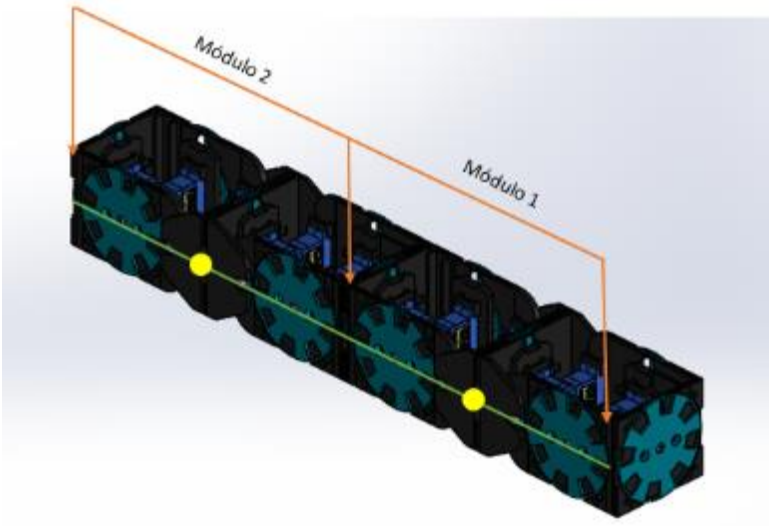


Figura 4-7 Enumeración de módulos Configuración oruga, conexión Pivote – Pivote.

4.4.3 Conexión Centro – Centro

Se logra a partir de la unión de dos caras Centro; al ser carente de articulación se conectan otro par de semi-módulos unidos por las caras Centro, formando así un módulo *M* en conexión Centro – Centro, como se ve en la Figura 4-8.

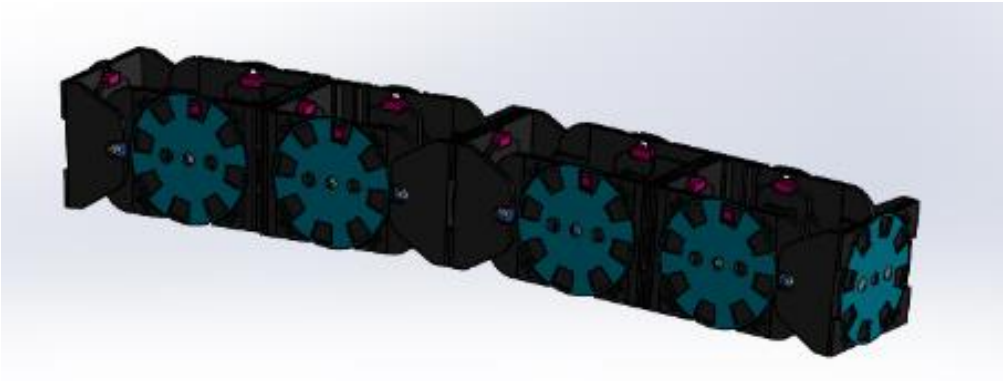


Figura 4-8 Módulo en conexión Centro-Centro posición vertical (configuración oruga)

Esta conexión es la única que consta de más de dos semi-módulos puesto que al unir únicamente dos caras centro, no se consigue tener una articulación que permita el movimiento.

En la Figura 4-9 se evidencia la enumeración de los módulos, de esta manera el módulo 1 comparte segmento con el módulo 2; de igual manera el módulo final (módulo 5) comparte un segmento con el módulo anterior (módulo 4) y en el resto de la cadena, cada módulo comparte sus dos segmentos con el anterior y el siguiente. Como resultado, este tipo de conexión necesita dos

semi- módulo más (uno al principio y otro al final) en comparación con la configuración Pivote – Pivote.

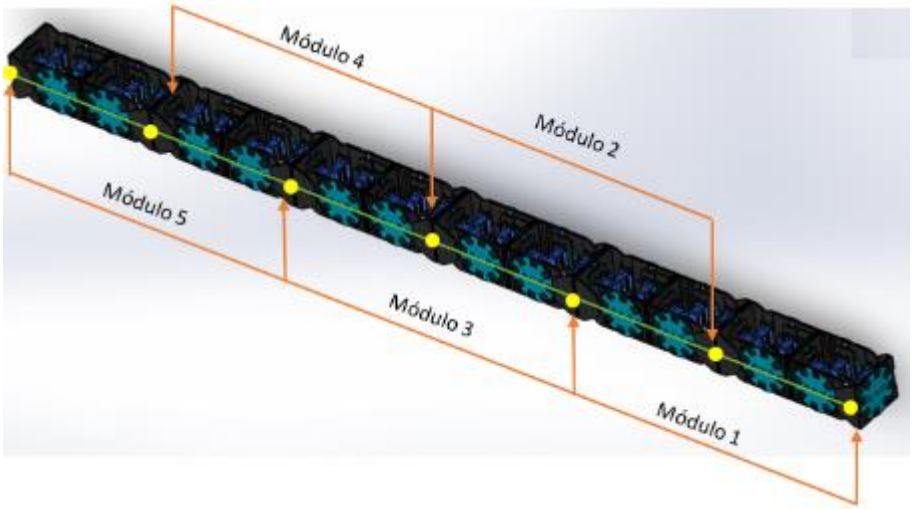


Figura 4-9 Enumeración de módulos Configuración oruga, conexión Centro – Centro

4.5 Conexiones de los Módulos en Configuración Tipo Serpiente

Gracias al diseño de los módulos, es posible realizar 3 diferentes conexiones a partir de la unión entre ellos, es necesario tener presente que, para la configuración tipo serpiente, los módulos alternan su posición ente vertical horizontal, siendo siempre el primer módulo vertical. Estas conexiones son:

4.5.1 Conexión Pivote – Centro

Se forma a partir de la unión entre la cara del pivote de un módulo *M* en posición horizontal y la cara Centro de otro módulo *M* en posición vertical, como se ve en la Figura 4-10.

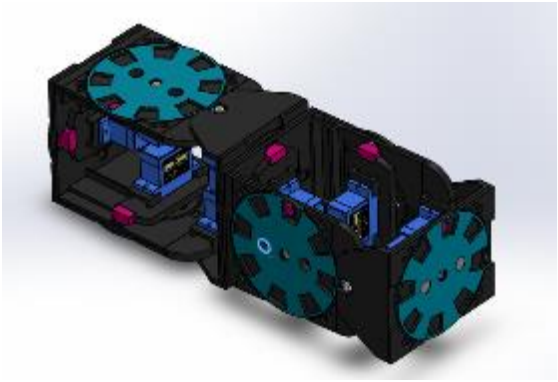


Figura 4-10 Cadena de dos módulos conexión Pivote - Centro configuración serpiente.

En la Figura 4-11 evidencia la manera de enumerar los módulos, los círculos amarillos representan las articulaciones provistas por el pivote y las líneas verdes los segmentos. En este tipo de conexión las articulaciones se encuentran más seguidas con respecto a las otras conexiones y comparten segmentos entre un módulo y otro; razón por la cual se alternan entre módulos verticales y horizontales, como se muestra.

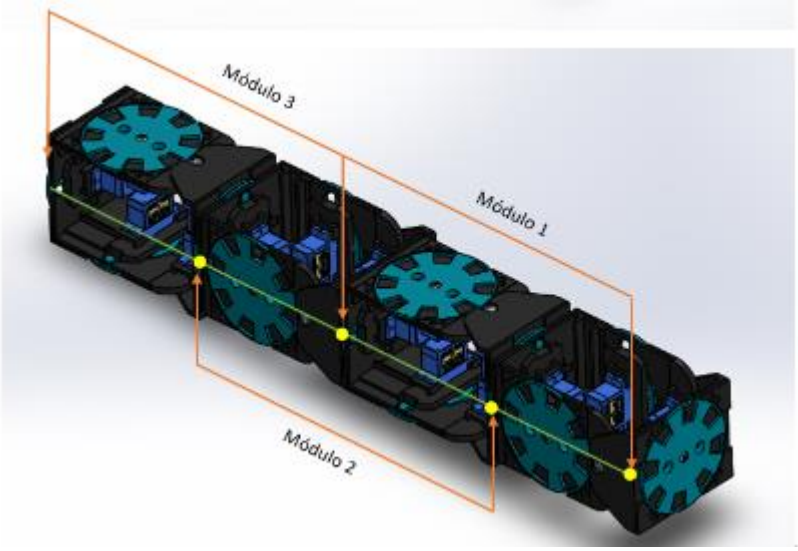


Figura 4-11 Enumeración de módulos Configuración serpiente, conexión Pivote – Centro

4.5.2 Conexión Pivote – Pivote

Se logra a partir de la unión entre un módulo en conexión Pivote - Pivote en posición vertical (ver Figura 4-6) y otro módulo en la misma conexión, pero en posición horizontal, formando una cadena en configuración tipo serpiente conformada por 2 módulos *M*, como se evidencia en la Figura 4-12.

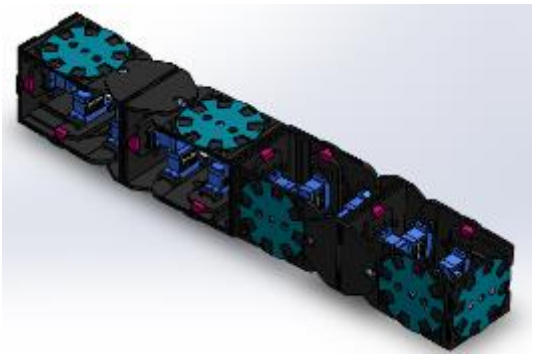


Figura 4-12 Conexión Pivote - Pivote en configuración serpiente.

La enumeración de los módulos y posicionamiento se hace más evidente en la Figura 4-13 los círculos amarillos representan las articulaciones provistas por el pivote y las líneas verdes los segmentos, la cadena tiene dos módulos (*M=2).*

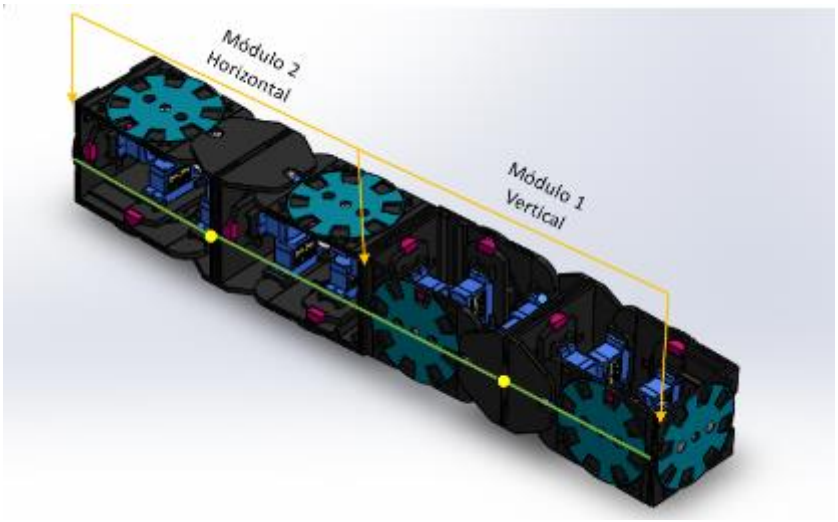


Figura 4-13 Enumeración de módulos Configuración serpiente conexión Pivote – Pivote.

4.5.3 Conexión Centro – Centro

Se logra a partir de la unión entre un módulo en conexión Centro - Centro en posición vertical (ver Figura 4-8) y otro módulo en la misma conexión, pero en posición horizontal, formando una cadena en configuración tipo serpiente, como se evidencia en la Figura 4-14.

Figura 4-14 Conexión Centro - Centro en configuración serpiente.

En la Figura 4-15 se muestra en detalle la enumeración de los módulos; en este caso la cadena está compuesta por tres módulos (*M=3*). Los círculos amarillos representan las articulaciones provistas por el pivote y las líneas verdes los segmentos.

- El módulo 1 está ubicado en posición vertical.
- El módulo 2 comparte segmento con el módulo1 y módulo 3.
- El módulo 3 se encuentra ubicado en posición horizontal.

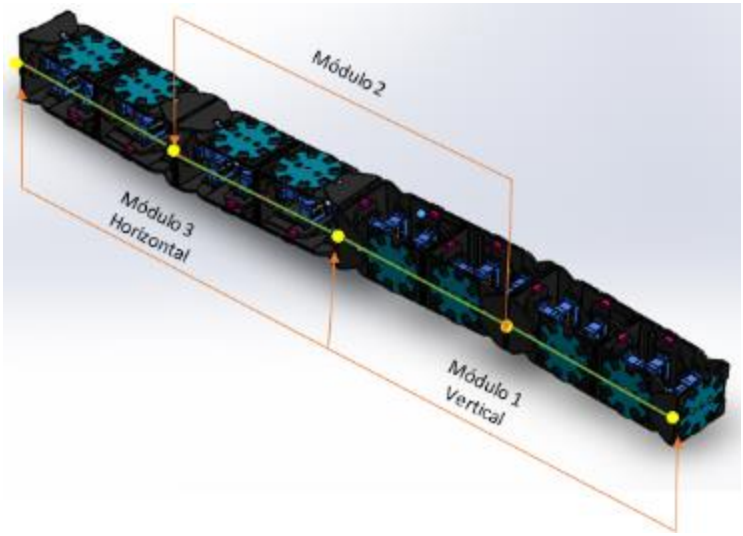


Figura 4-15 Enumeración de módulos Configuración serpiente conexión Centro – Centro.

4.6 Archivo Controlador

Teniendo importada la geometría del sistema robótico modular MECABOT 3.0 en el nodo *world* de Webots, se procede a desarrollar el archivo controlador que se encarga de definir su

comportamiento y permiten obtener los datos de velocidad requeridos para su posterior análisis. Este archivo controlador está escrito en lenguaje de código C.

4.6.1 Archivo Controlador Configuración Tipo Oruga

La Figura 4-16 explica el desarrollo del código en C para la configuración tipo oruga en el software Webots. Para iniciar se deben definir los nodos, identificar los motores, las variables de control del tiempo y las variables definidas por el usuario.

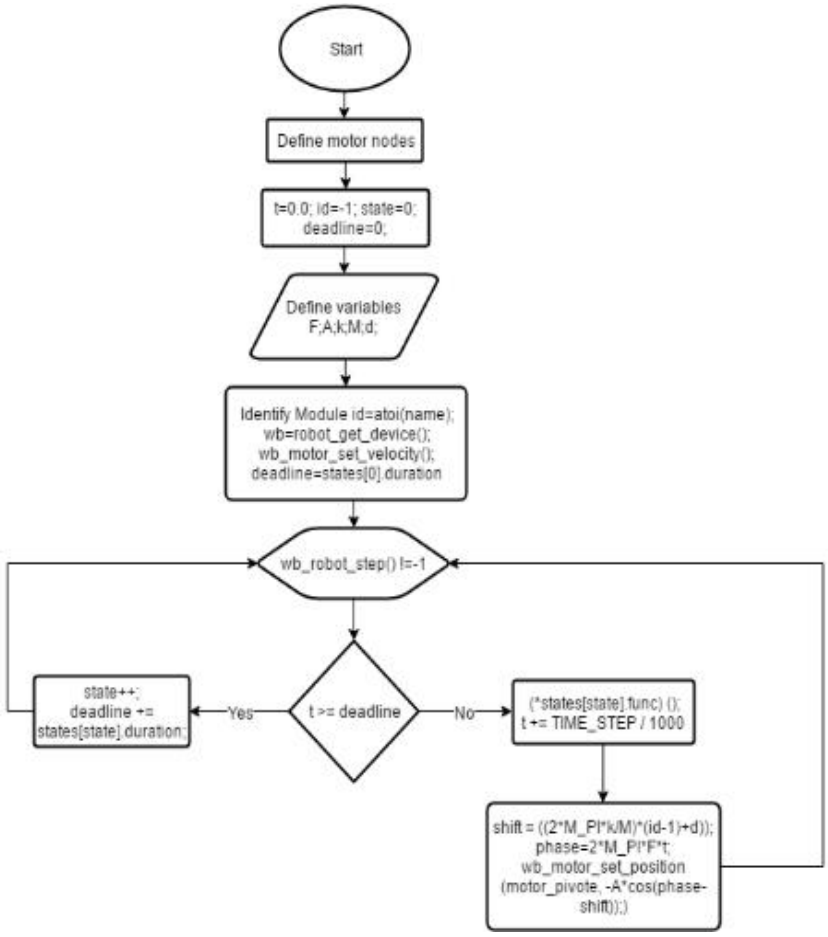


Figura 4-16 Diagrama de flujo controlador tipo oruga

En la Tabla 4-1 se definen las variables utilizadas en el controlador.

Símbolo	**Variable**	**Unidades**
t	Tiempo	s
id	Identificador módulo	-
$state$	Estado	-
$deadline$	Límite de tiempo	s
f	Frecuencia	s^{-1}

A	Amplitud	-
k	Número de ondulaciones	s^{-1}
M	Número de módulos	-
d	Distancia	m

Tabla 4-1 Variables archivo controlador tipo oruga

4.6.2 Archivo Controlador Configuración Tipo Serpiente

La Figura 4-17 describe el desarrollo del código en C para la configuración tipo serpiente en el software Webots, se debe tener en cuenta la disposición de los módulos en la cadena (horiontal o vertical) para determinar que ecuación define su movimiento.

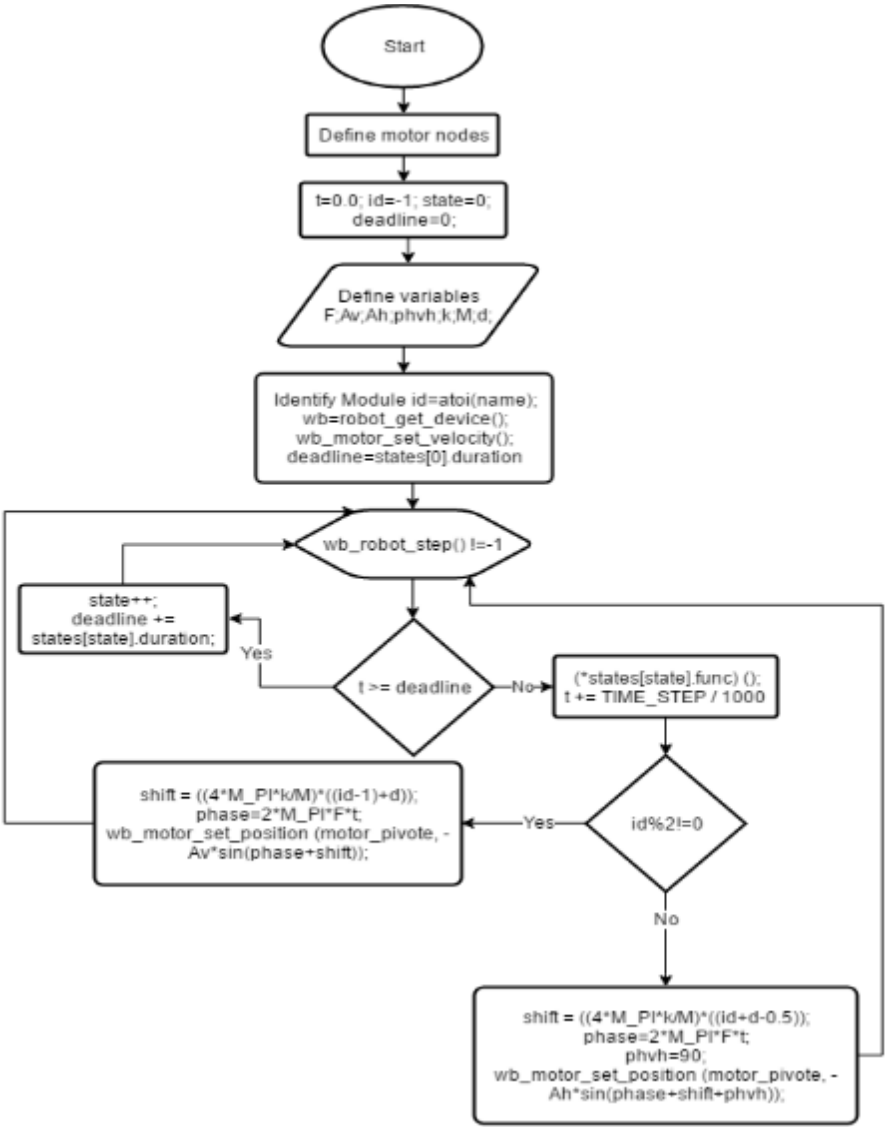


Figura 4-17 Diagrama de flujo controlador tipo serpiente

En la Tabla 2-1 se definen las variables utilizadas en el controlador.

Símbolo	Variable	Unidades
A_v	Amplitud vertical	-
A_h	Amplitud horizontal	-
$phvh$	Diferencia de fase entre horizontal y vertical	rad/s

Tabla 4-2 Variables archivo controlador tipo serpiente

4.7 Simulación en Configuración Tipo Oruga

Para esta configuración se utiliza el controlador de la sección 4.6.1, con la variación de los parámetros: f, A, M y k para obtener datos de velocidad. A continuación, se presentan los resultados para cada una de las conexiones:

Para el análisis de velocidad del sistema robótico modular MECABOT 3.0 en configuración tipo oruga, se da prioridad a la estabilidad de la locomoción de la cadena con el fin de evitar que esta se derrumbe sobre un costado, por tanto, siguiendo el criterio de estabilidad para la locomoción presentado en la sección 2.2.4.2 se emplea la Ec. 2-29 y se determina que para lograr el valor $k \geq 2$ se necesita un mínimo de 4 y 6 módulos para $k = 2$ y $k = 3$ respectivamente.

4.7.1 Conexión Pivote – Centro

Para esta configuración se mantiene constante A y k. Al simular se pudo observar que para que el robot avance, M debe ser par. Al ser ésta la más corta se hicieron varias pruebas con diferente número de módulos: con $k = 2,\ 4 \leq M \leq 10$ y con $k = 3,\ 6 \leq M \leq 12$, frecuencia $0.5 \leq f \leq 2$, amplitud $0.1 \leq A \leq 0.5$.

4.7.1.1 Simulación k=2

En la Tabla 4-3 se observa de color azul los segmentos y en verde las articulaciones que conforman el módulo para la variación en amplitud para k = 2 de la onda sinusoidal.

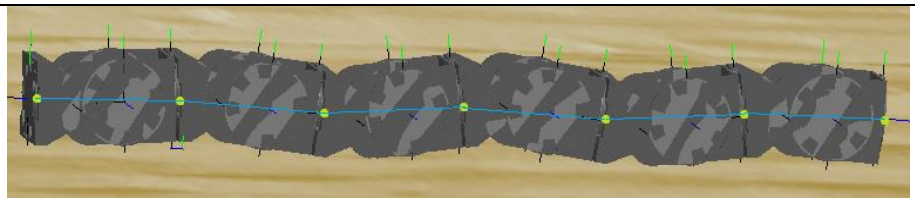
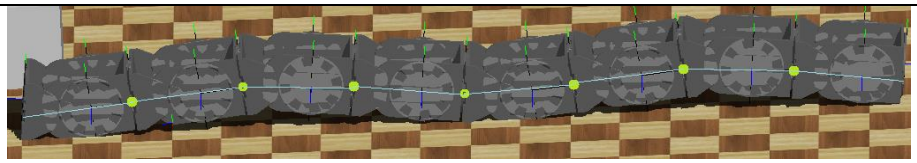
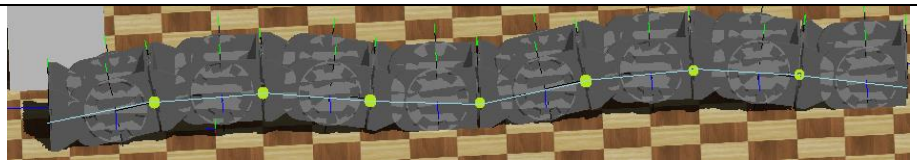
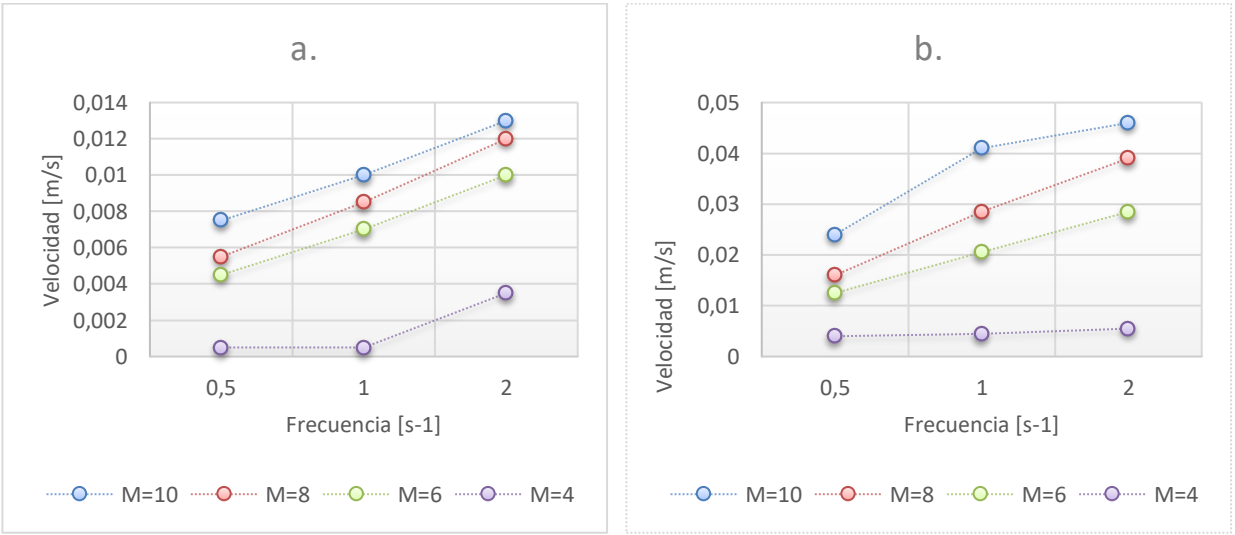
Amplitud	***k* = 2**
0,1	
0,3	
0,5	

Tabla 4-3 Ondas conexión Pivote – Centro con variación de amplitud y k =2

Los valores de velocidad obtenidos se muestran en la Tabla Apéndice A 8-1, muestra que entre mayor sea el número de módulos mayor es la velocidad. Para un análisis visual más sencillo de los valores obtenidos en la Tabla Apéndice A 8-1 se presentan los gráficos de la Figura 4-18.



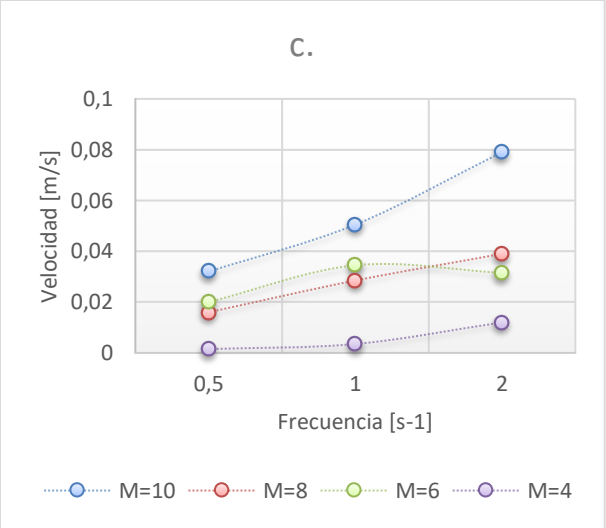


Figura 4-18 Gráficos velocidades configuración tipo Oruga Pivote - Centro k=2 a. A=0.1 b. A=0.3 c. A=0.5

En la Figura 4-18a. Existe una relación directamente proporcional entre f, V y M, a excepción de la cadena más corta $M = 4$ en donde la velocidad es despreciable $V \approx 0{,}0015\,\frac{m}{s}$. En la Figura 4-18b. Para $M = 8\ y\ M = 6$ se mantiene un comportamiento lineal, en el caso de $M = 4$ se tiene V casi nula $V \approx 0.0046\,\frac{m}{s}$; por el contrario $M = 10$ presenta la mayor velocidad $V = 0.046\,\frac{m}{s}$, pero la magnitud del incremento disminuye después de $f = 0.1$. En la Figura 4-18c. Se observa un incremento en la velocidad respecto a f y M a excepción de $M = 6$.

Para $k = 2$ se observa una velocidad máxima de $V = 0.079\ m/s$ con $M = 10, f = 2, A = 0.5$; una velocidad mínima de $V = 0.0005\ m/s$ con $M = 4, f = 0.5\ f = 1\ , A = 0.1$. Con $M = 4$ se obtienen velocidades muy bajas $V \approx 0.0015$ que no se ven afectadas por la variación de amplitud y frecuencia. En el caso de $M = 6$ con $A = 0.5, f = 2$ el incremento en velocidad no se mantiene directamente proporcional con respecto a la frecuencia, situación que no se presenta en los otros casos.

4.7.1.2 Simulación k=3

En la Tabla 4-4 se observa de color azul los segmentos y en verde las articulaciones que conforman el módulo para la variación en amplitud para *k* = 3 de la onda sinusoidal.

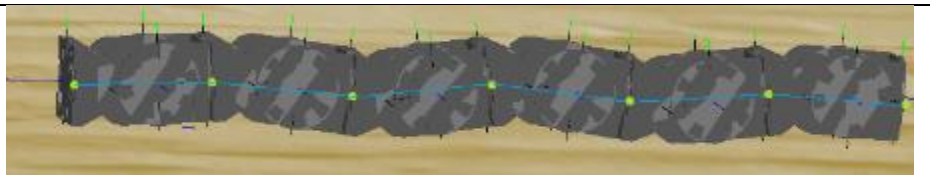
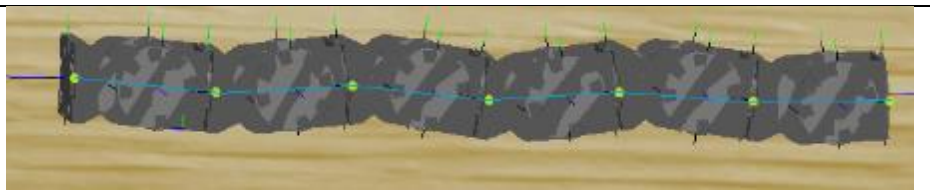
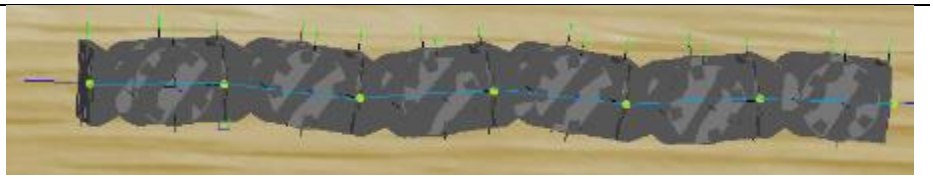
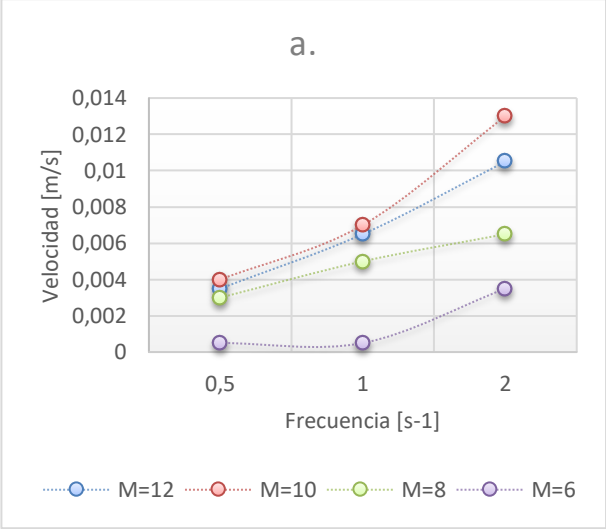
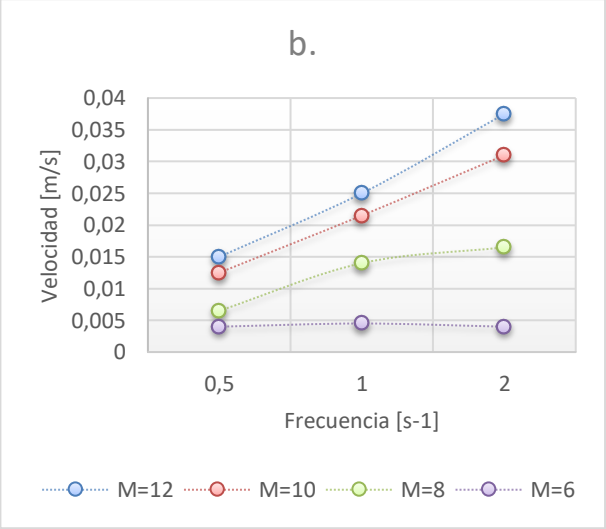
Amplitud	**k = 3**
0,1	
0,3	
0,5	

Tabla 4-4 Ondas conexión Pivote – Centro con variación de amplitud y k = 3

Los resultados de velocidad se observan en la Tabla Apéndice A 8-2, se puede observar que entre mayor número de módulos tenga el robot, mayor velocidad tiene.





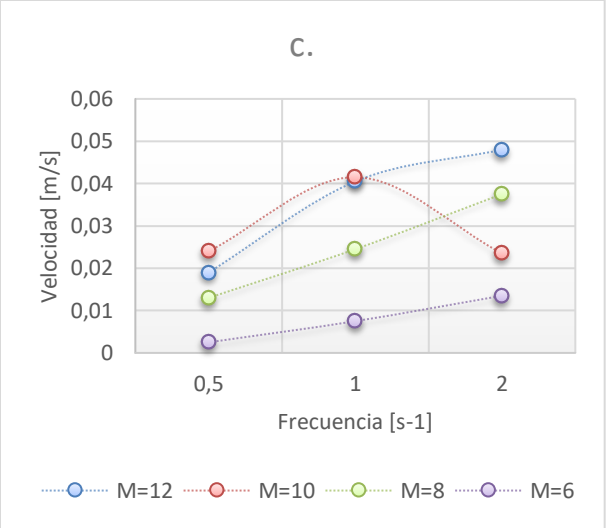


Figura 4-19 Gráficas velocidades configuración tipo Oruga, Pivote - Centro k=3; a. A=0.1. b. A=0.3 c. A=0.5

En la Figura 4-19a. Se obtienen velocidades muy bajas, con incrementos mínimos al variar el número de módulos. En la Figura 4-19b. Las velocidades son mayores con respecto a la Figura 4-19 a. siendo la velocidad máxima en $M = 12, f = 2, V = 0.0375\ m/s$. En la Figura 4-19c. Se observa incremento de V al aumentar f, sin embargo, al tener $M = 10$ y $f > 1$ se presenta una disminución considerable de la velocidad, lo que la hace menos viable.

4.7.1.3 Análisis de Simulación

Para $k = 3$ la velocidad máxima es $V = 0.048$ con $M = 12, f = 2, A = 0.5$; una velocidad minima de $V = 0.0005$ con $M = 6, f = 0.5, A = 0.1$. Con $M = 6$ se obtienen velocidades muy bajas $V \approx 0.0015 \frac{m}{s}$ con respecto a las cadenas más largas. En el caso de $M = 10$ con $A = 0.5, f = 2$ no hay incremento en la velocidad en todos los puntos con respecto a la frecuencia, situación que no se presenta en los otros casos. Al comparar los datos obtenidos entre $k = 2$ y $k = 3$, se determina que la velocidad máxima es $V = 0.079\ m/s$ con $k = 2$; sin embargo $k = 3$, es más estable.

4.7.2 Conexión Pivote – Pivote

Se mantiene constante k y M. Se observa que sólo funciona al utilizar un número de módulos impar puesto que de esta manera se alternan los módulos de los extremos para entrar en contacto con el piso y así permitir el avance de la oruga. Se simuló $k = 2$ con $M = 5\ y\ M = 7$;

$k = 3$ con $M = 7$ y $M = 9$. Se tomó en cuenta que más módulos serian inapropiados para la aplicación ya que al final quedaría una cadena demasiado extensa.

4.7.2.1 Simulación k = 2

En la Tabla 4-5 se observa de color azul los segmentos y en verde las articulaciones que conforman el módulo.

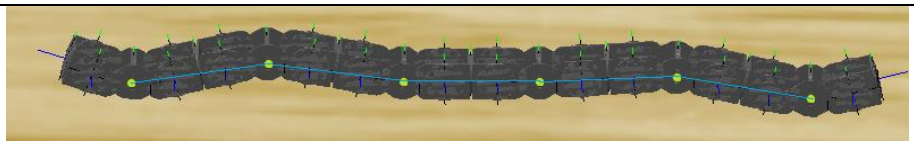
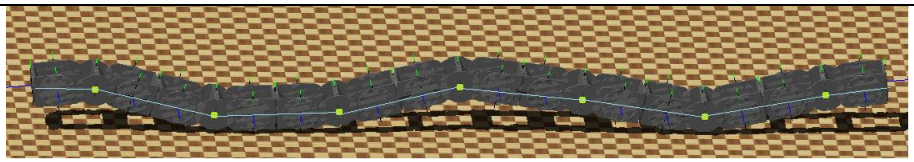
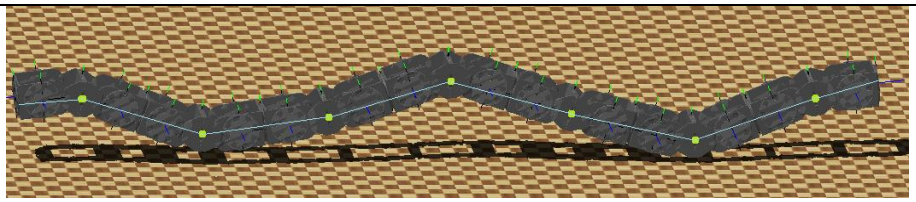
Amplitud	$\boldsymbol{k = 2}$
0,1	
0,3	
0,5	

Tabla 4-5 Ondas conexión Pivote – Pivote con variación de amplitud y k =2

Los valores de velocidad obtenidos para $k = 2$ se muestran en la Tabla Apéndice A 8-3.

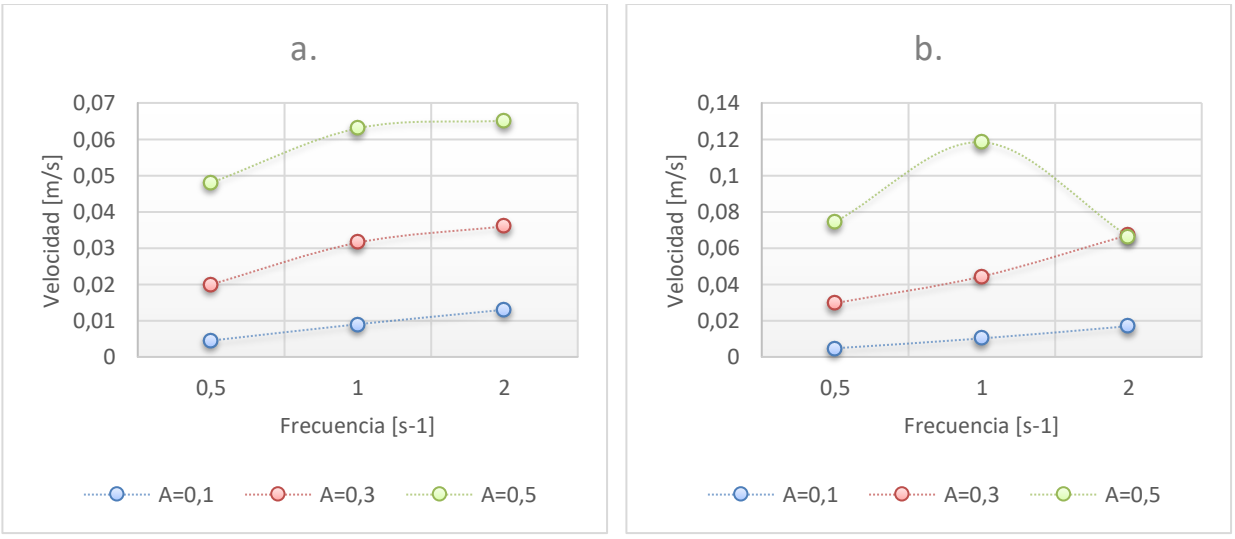


Figura 4-20 Gráficos velocidades configuración tipo Oruga Pivote - Pivote k=2
a. M=5 b. M=7

Figura 4-20a. Se observa un incremento de la velocidad con respecto a la frecuencia. Figura 4-20b. Cuando $A = 0.5$ la cadena deja de ser estable cuando $f > 1$. La velocidad máxima $V = 0.11835\, m/s$ con $A = 0.5, f = 1$ y $M = 7$; la velocidad mínima $V = 0.0047$ con $A = 0.1, f = 0.5$ y $M = 5$ y $M = 7$.

4.7.2.2 Simulación k = 3

Para $k = 3$ el número mínimo de módulos es M = 6. Para esta conexión es necesario que el número de módulo sea impar, por esto se simula con M = 7. En Tabla 4-6 se observa de color azul los segmentos y en verde las articulaciones que conforman el módulo para $k = 3$.

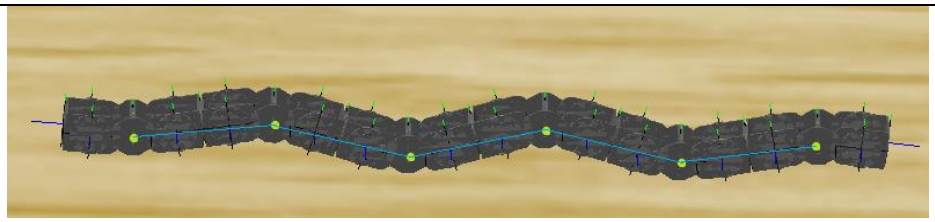
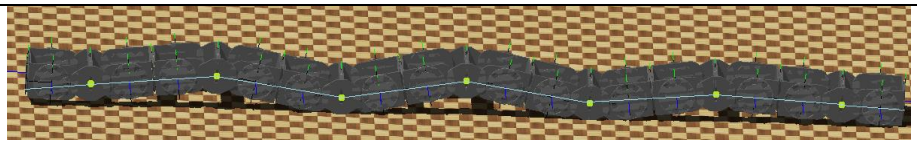
Amplitud	**k = 3**
0,1	
0,3	
0,5	

Tabla 4-6 Ondas conexión Pivote – Pivote con variación de amplitud y k =3

Las velocidades obtenidas para $k = 3$ se observan en la Tabla Apéndice A 8-4.

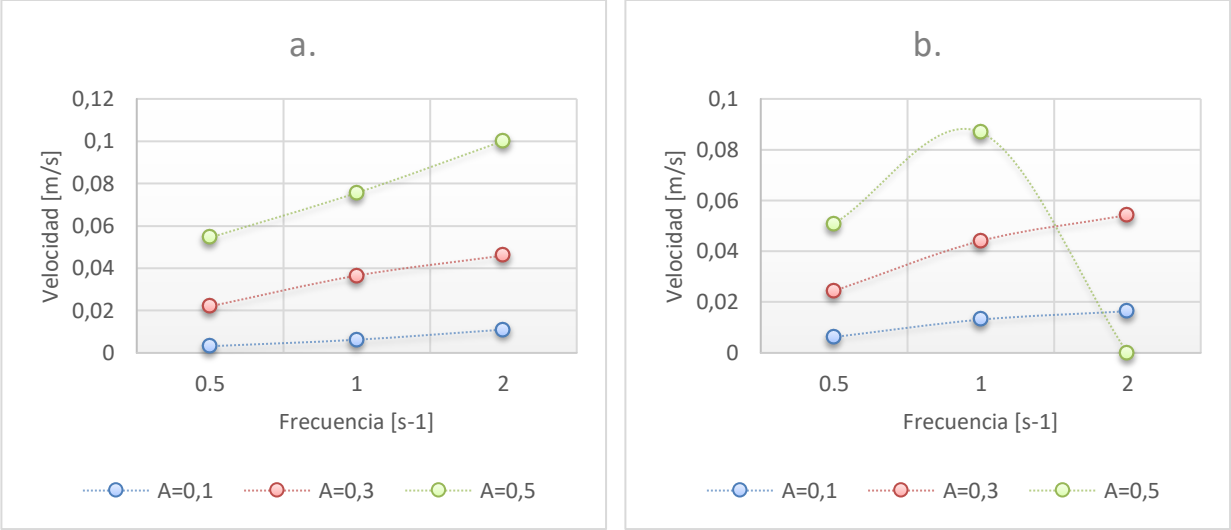


Figura 4-21 Gráficos velocidades configuración tipo Oruga Pivote - Pivote k=3
a. M=7 b. M=9

En la Figura 4-21a. se observa que incrementa la velocidad al aumentar la frecuencia y la amplitud. En la Figura 4-21b. se observa que para $f > 1$ en $A = 0.5$ se pierde la estabilidad del robot.

4.7.2.3 Análisis de simulación

La máxima velocidad es $V = 0.1\, m/s$ se da en $M = 7, f = 2\,, A = 0.5$; la mínima velocidad $V = 0.003$ en $M = 7, f = 0.5, A = 0.1$. Haciendo una comparación entre $k = 2$ y $k = 3$, se presenta mayor velocidad en $k = 2$, pero $k = 3$ es más estable. Se evidencia que a mayor número de módulos, menor es el rango de A que permite tener una locomoción estable.

4.7.3 Conexión Centro – Centro

Para esta configuración se tomaron en cuenta las mismas consideraciones que en el caso Pivote - Pivote.

4.7.3.1 Simulación k=2

En la Figura 4-7 se observa de color azul los segmentos y en verde las articulaciones que conforman el módulo para la variación de amplitudes para k = 2.

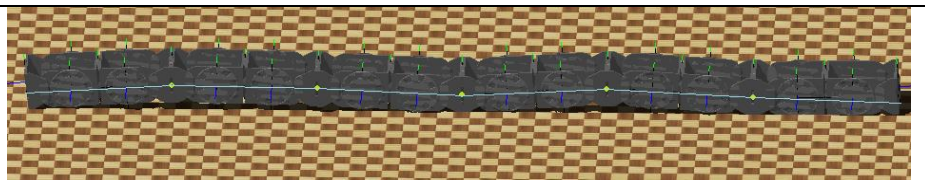
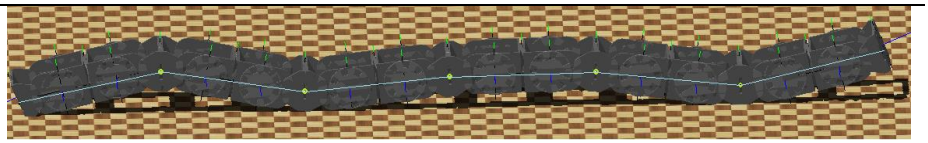
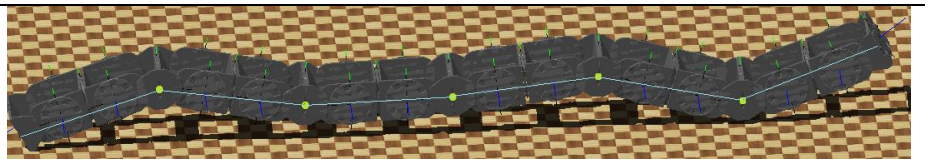
Amplitud	$k = 2$
0,1	
0,3	
0,5	

Tabla 4-7 Ondas conexión Centro – Centro con variación de amplitud y k =2

En la Tabla Apéndice A 8-5 se recolectan los datos para $k = 2$

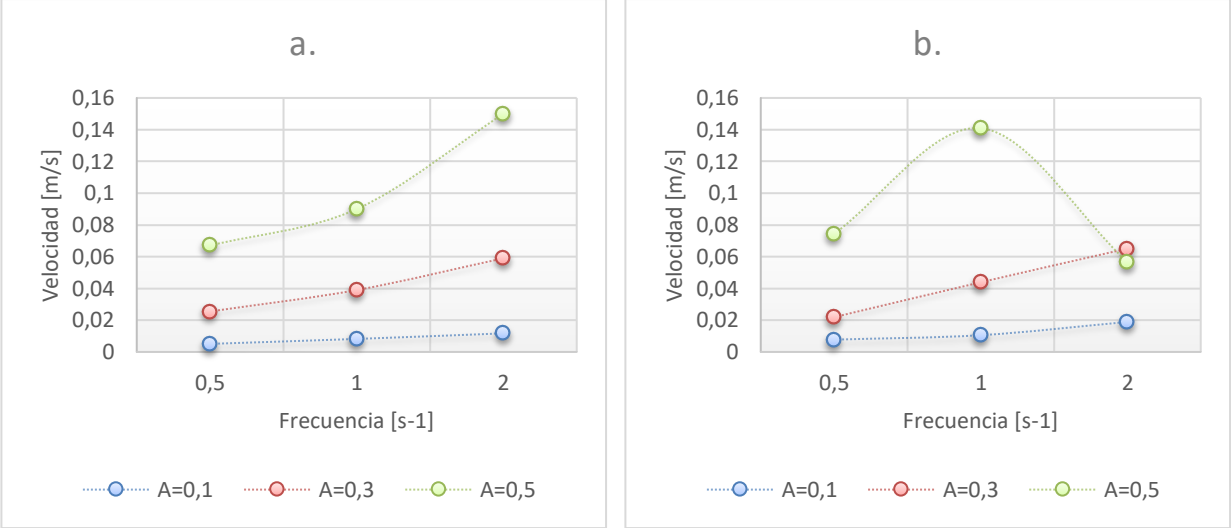


Figura 4-22 Gráficos velocidades configuración tipo Oruga Centro – Centro k=2, a. M=5 b. M=7

En la Figura 4-22a se observa que, al aumentar A, mayor es la velocidad. En la Figura 4-22b se observa que para $f > 1$ en $A = 0.5$ se pierde la estabilidad del robot.

4.7.3.2 Simulación k=3

En la Tabla 4-8 se observa de color azul los segmentos y en verde las articulaciones que conforman el módulo.

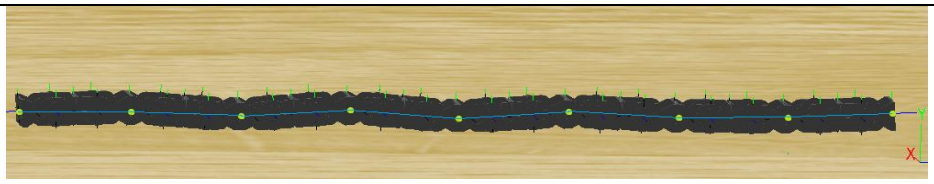
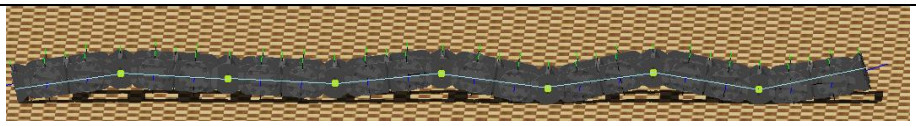
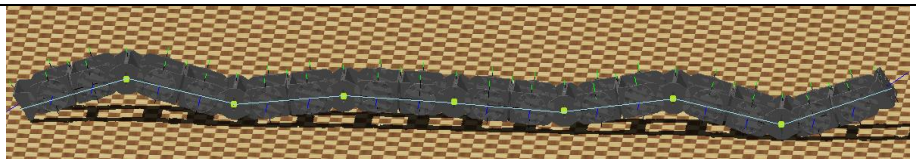
Amplitud	$\boldsymbol{k = 3}$
0,1	
0,3	
0,5	

Tabla 4-8 Ondas conexión Centro – Centro con variación de amplitud y k =3

En la Tabla Apéndice A 8-6 se recolectan los datos para $k = 3$

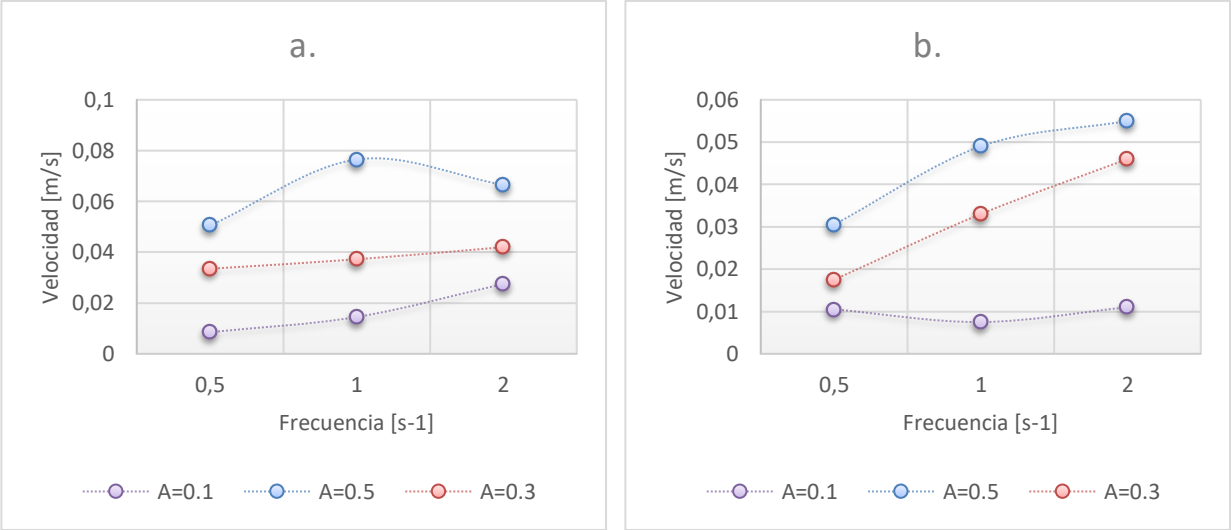


Figura 4-23 Gráficos velocidades configuración tipo Oruga Centro - Centro k=3.
a. M=7 b. M=9

En la Figura 4-23a. se observa un incremento en la velocidad respecto a f y A a excepción de A= 0.5. En la Figura 4-23b. se observa que para $A = 0.3$ y $A = 0.5$, la velocidad incrementa al incrementar los otros parámetros. Por otro lado, en $A = 0.1$ la velocidad no presenta un comportamiento definido.

4.7.3.3 Análisis de Simulación

La máxima velocidad es $V = 0.076\ m/s$ se da en $M = 7, f = 1\ , A = 0.5$; la velocidad mínima $V = 0.0075$ en $M = 9, f = 1, A = 0.1$. Haciendo una comparación entre $k = 2$ y $k = 3$, se presenta mayor velocidad en $k = 2$, pero $k = 3$ es más estable. Se evidencia que a mayor número de módulos, menor es el rango de A que permite tener una locomoción estable.

4.8 Simulación en Configuración Tipo Serpiente

Gracias al diseño de los módulos, es posible realizar 3 diferentes conexiones a partir de la unión entre ellos, para definir estas conexiones hay que tener presente el concepto de módulo visto en la sección 2.1.5. Para esta configuración se utiliza el controlador de la Figura 4-17, con la variación de los parámetros: f, A_h, M y k, manteniendo constante $A_v = 0.1$ para obtener datos de velocidad. Es importante tener presente que, en la configuración tipo serpiente, los módulos se alternan entre verticales y horizontales, de esto depende la estructura de cada módulo para cada conexión. A continuación, se presentan los resultados para cada una de las conexiones:

4.8.1 Conexión Pivote – Centro

En la Figura 4-24 se observan líneas azules que representan los segmentos y puntos verdes que representan las articulaciones del sistema robótico modular, evidenciando tanto la componente vertical como la componente horizontal.

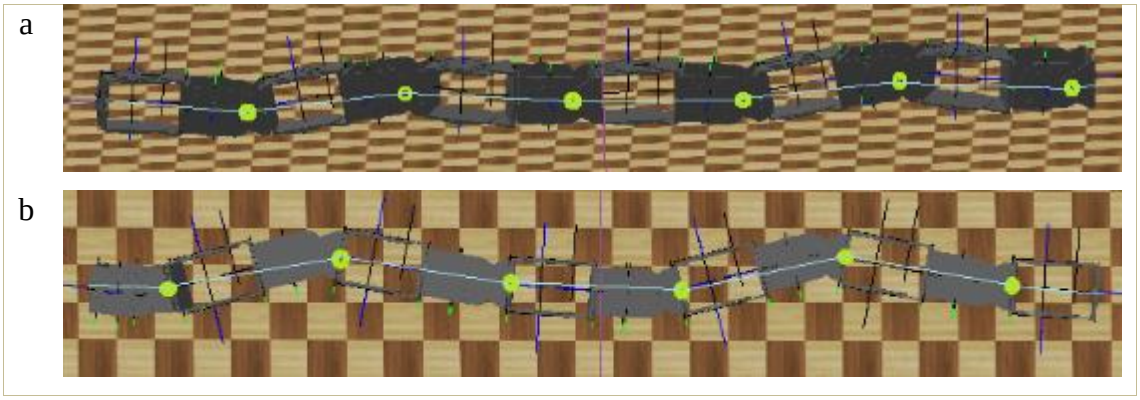


Figura 4-24 Ondas configuración tipo Serpiente conexión Pivote – Centro con k =2
a. vista frontal b. vista superior

Se mantiene constante $A_v = 0.1, k = 2, M = 12$, variando $0.3 \leq A_h \leq 0.7\ y\ 0.5 \leq f \leq 2$. Los valores de velocidad obtenidos se muestran en la Tabla Apéndice A 8-7.

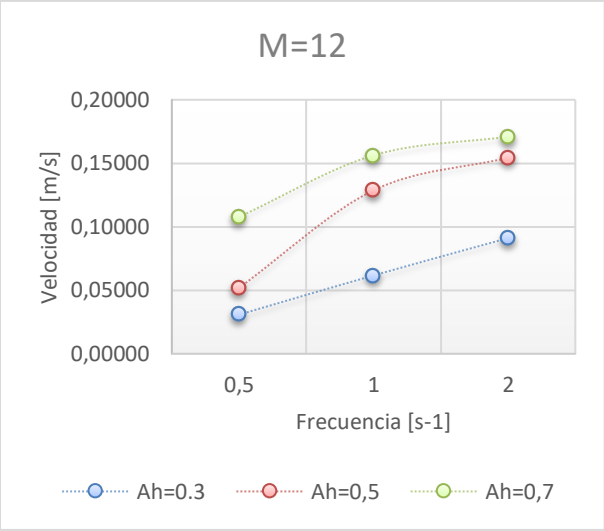


Figura 4-25 Gráfico velocidad configuración tipo Serpiente Pivote - Centro k=2

En la Figura 4-25 para todos los valores de A_h se tiene un incremento de la velocidad al aumentar la frecuencia. La velocidad máxima 0,171 m/s se da en $f = 2$y $A_h = 0.7$ y la mínima es 0.031 $f = 0.5$ y $A_h = 0.3$

4.8.2 Conexión Pivote – Pivote

En la Figura 4-26 se observan líneas azules que representan los segmentos y puntos verdes que representan las articulaciones del sistema robótico modular, evidenciando tanto la componente vertical como la componente horizontal.

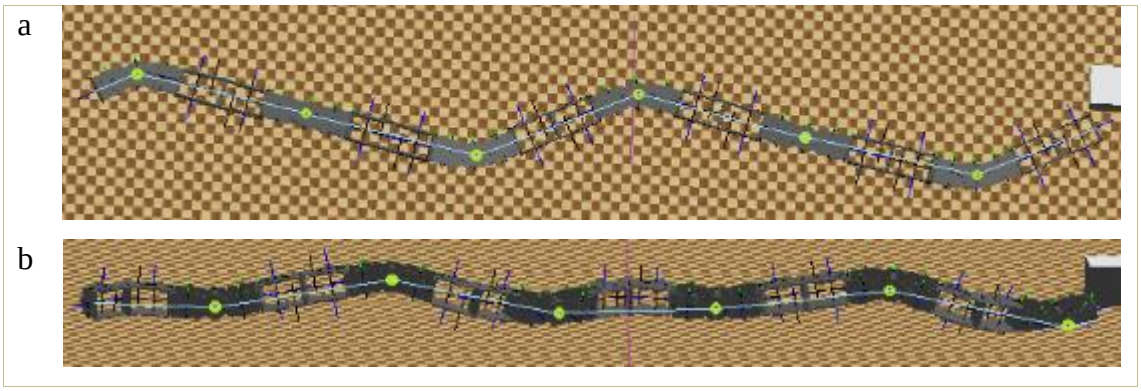


Figura 4-26 Ondas configuración tipo Serpiente conexión Pivote – Pivote con k =2
a. vista frontal b. vista superior

Se mantiene constante $A_v = 0.1,\ k = 2, M = 12,$ variando $0.1 \leq A_h \leq 0.5\ y\ 0.5 \leq f \leq 2$. Los valores de velocidad obtenidos se muestran en la Tabla Apéndice A 8-8 para $k = 2$

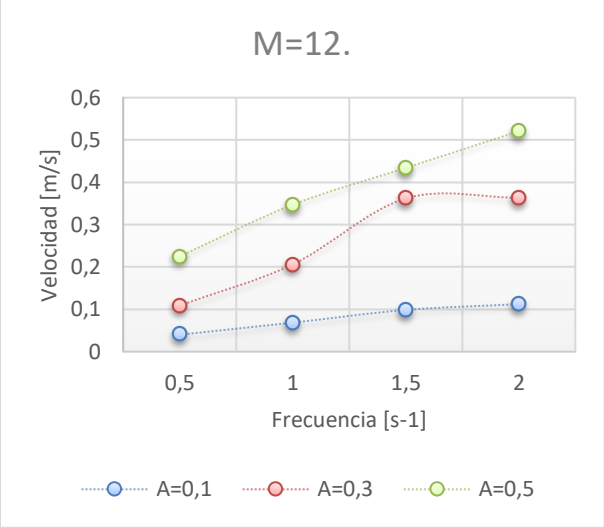


Figura 4-27 Gráfico velocidad configuración tipo Serpiente Pivote – Pivote k=2

En la Figura 4-27 para todos los valores de A_h se tiene un incremento de la velocidad al aumentar la frecuencia. La velocidad máxima 0.52 m/s se da en $f = 2$y $A_h = 0.5$ y la mínima es $0.04\ m/s\ f = 0.5$ y $A_h = 0.1$.

4.8.3 Conexión Centro – Centro

En la Figura 4-28 se observan líneas azules que representan los segmentos y puntos verdes que representan las articulaciones del sistema robótico modular, evidenciando tanto la componente vertical como la componente horizontal.

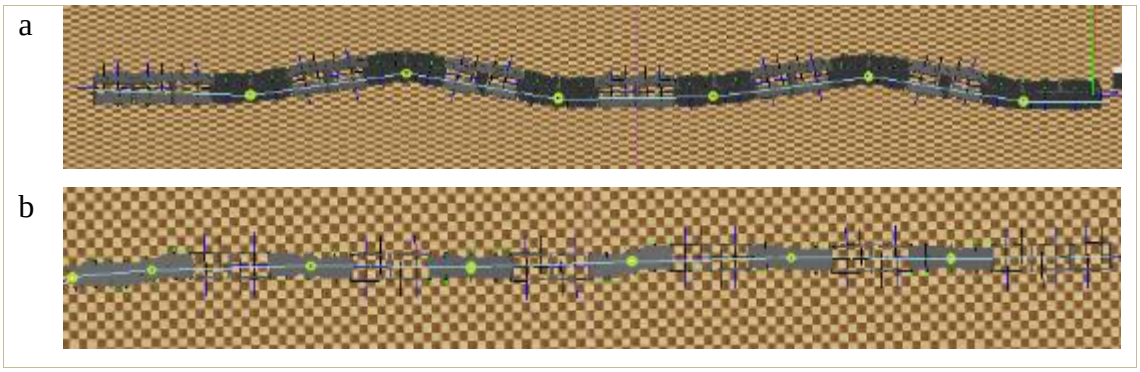


Figura 4-28 Ondas conexión Centro – Centro con k =2
a. vista frontal b. vista superior

Se mantiene constante $A_v = 0.1$, $k = 2, M = 12$, variando $0.3 \leq A_h \leq 0.7$ y $0.5 \leq f \leq 2$. Los valores de velocidad obtenidos se muestran en la Tabla Apéndice A 8-9.

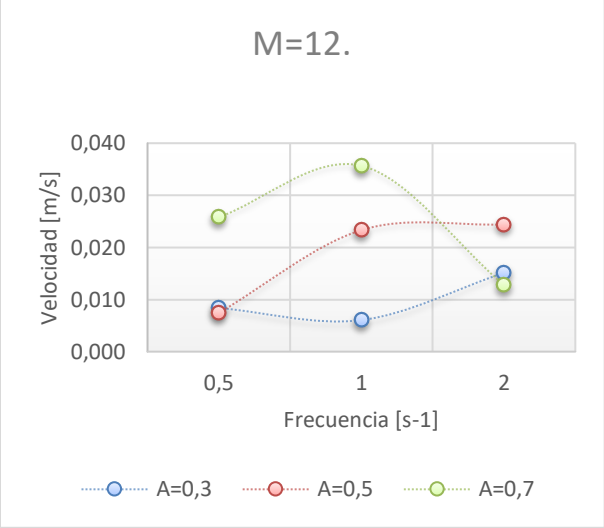


Figura 4-29 Gráfico velocidad configuración tipo Serpiente Centro – Centro k=2

En la Figura 4-29 se observa un comportamiento no definido en $A_h = 0.7$y $A_h = 0.3$, mientras que en $A_h = 0.5$ a partir de $f \geq 1$, la velocidad tiende a ser constante. La velocidad máxima es 0.036 m/s en $A_h = 0.7$y $f = 1$, mientras que la minima es de 0,006 m/s en $A_h = 0.3$ y $f = 1$.

4.9 Conclusiones del capítulo

En este capítulo se describió el software de simulación Webots y sus características principales, se presentó el diseño de MECABOT 3.0, las conexiones posibles y el controlador para configuraciones tipo oruga y serpiente; por último, se analizaron los datos obtenidos en la simulación dando como resultado las siguientes conclusiones:

La longitud minima de la cadena para cada tipo de conexión, es la que menor velocidad tiene sin importar el valor de A, f, k o la configuración. Para cualquier configuración, el valor de k que brinda mayor estabilidad es $k = 3$, ya que presenta 3 puntos de apoyo durante toda la propagación de la onda.

El rango de funcionamiento considerado inicialmente para la configuración tipo oruga era $0.1 \leq A \leq 0.5$ y $0.5 \leq f \leq 2$, asumiendo que con la mayor A y f se conseguiría la mayor velocidad, conservando la estabilidad de la locomoción en todos los casos; sin embargo, al realizar las pruebas se observa una disminución drástica en la velocidad, esto se puede justificar por las siguientes razones: al tener el mayor valor de amplitud y un valor muy pequeño de k con respecto a la longitud de la cadena, el largo de los módulos no permite el contacto con la superficie en los puntos medios, haciendo que todo el peso de la cadena recaiga sobre los módulos de los extremos y el desplazamiento sólo se dé por el que estos pivotes pueden proveer; igualmente afecta la estabilidad del robot provocando que este se derrumbe hacia un costado.

Para los dos valores de k probados tanto en la configuración oruga como en serpiente, la conexión que mejor funciona entre las tres, que requiere menor cantidad de semi-módulos, menor tiempo de simulación y mantiene un comportamiente más estable sin importar la variación de sus parámetros es la conexión Pivote – Centro; por el contrario, las conexiones Pivote – Pivote y Centro – Centro se componen de más semi-módulos, tardando más tiempo en simular.

La mayor velocidad obtenida se encuentra en la configuracion oruga con conexión Centro – Centro, con el mayor valor de A y f, debido a que un módulo tiene mayor longitud frente a las otras conexiones lo que provoca un mayor desplazamiento de toda la cadena.La mayor velocidad hallada para la configuración tipo serpiente se da en la conexión Pivote - Pivote, ya que la longitud de los módulos es mayor a la de Pivote - Centro generando mayor desplazamiento, sin alcanzar la longitud de Centro-Centro que debido a lo largo de su cadena es más lenta.

En la configuración tipo serpiente se observa que aunque el movimiento deseado es netamente lateral y sin alteración de la dirección del eje longitudinal de la cadena, se presenta un avance frontal pequeño, ya que se hace necesaria la generación de una onda vertical que permita la elevación de la cadena.

Al requerir la propagación de dos ondas en la cadena con configuración tipo serpiente, la componente A_h puede ser mayor con respecto a las limitaciones de la tipo oruga, sin embargo se

ve limitada por el diseño de los semi-módulos, obteniendo un rango de funcionamiento entre $0.1 \leq A_h \leq 0.7$, manteniendo A_v constante en 0.1.

5 Implementación Sistema Robótico Modular MECABOT 3.0

El sistema robótico modular MECABOT 3.0 es el tercer prototipo de MECABOT desarrollado en la Universidad Militar Nueva Granada. Este sistema robótico, está conformado por múltiples semi-módulos de tipo homogéneo. Su diseño fue resultado del desarrollo de proyectos "Diseño y simulación de un robot modular reconfigurable" y "Rediseño e implementación de un módulo robótico para sistema colaborativo", que para el presente trabajo de grado se toma como base para el estudio de la locomoción en configuraciones tipo oruga y serpiente por medio de la simulación e implementación.

A continuación, se describen las características mecánicas, electrónicas, ensamblaje y proceso de caracterización de los servomotores previos a la puesta en marcha, comunicación y procesamiento.

5.1 Modificaciones al Diseño Mecánico y Ensamble de MECABOT 3.0

En la Figura 5-1 y Figura 5-2 se pueden detallar la ubicación de las piezas enumeradas, que componen el semi-módulo MECABOT 3.0. En la Tabla 5-1 y Tabla 5-2 se definen los nombres de los componentes según el número. Las piezas son hechas en una impresora 3D con material ABS.

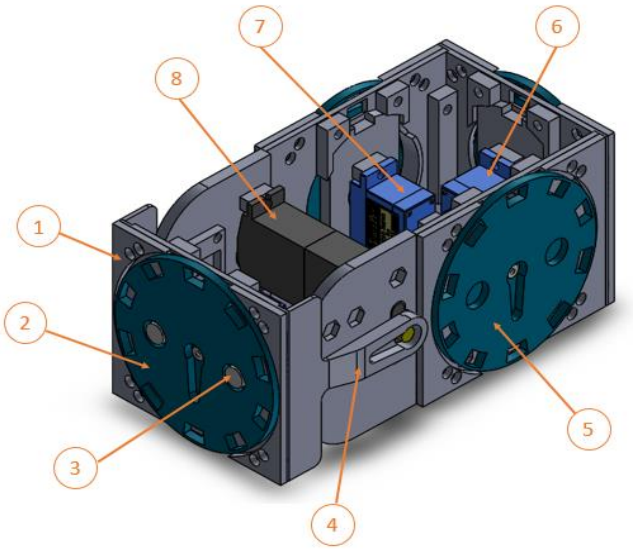


Figura 5-1 Enumeración componentes vista isométrica delantera semi-módulo MECABOT 3.0

Número de elemento	Componente
1	Cara pivote
2	Rueda Pivote
3	Imán
4	Pivote
5	Rueda lateral derecha.
6	Motor (6) cara Centro
7	Motor (4) rueda lateral izquierda
8	Motor (2) Pivote

Tabla 5-1 Componentes semi- módulo MECABOT 3.0 referentes a la Figura 5-1

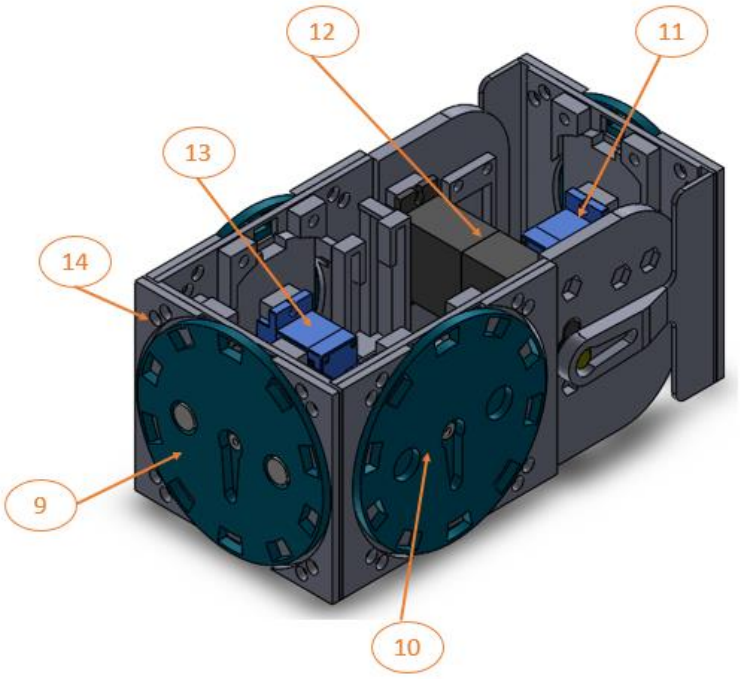


Figura 5-2 Enumeración componentes, vista isométrica posterior semi-módulo MECABOT 3.0.

Número de elemento	Componente
9	Rueda centro
10	Rueda lateral izquierda

11	Motor (5) rueda Pivote
12	Motor (1) Pivote
13	Motor (3) rueda lateral izquierda
14	Cara centro

Tabla 5-2 Componentes semi- módulo MECABOT 3.0 referentes a la Figura 5-2

El semi-módulo MECABOT 3.0 tiene como característica importante que actúa como una "*Unidad Modular Autónoma (AMU)"* lo que significa que cada semi-módulos puede moverse sin estar acoplado a otro, en Figura 5-3 se aprecia los 3 grados de libertad con los que cuenta el semi-módulo.

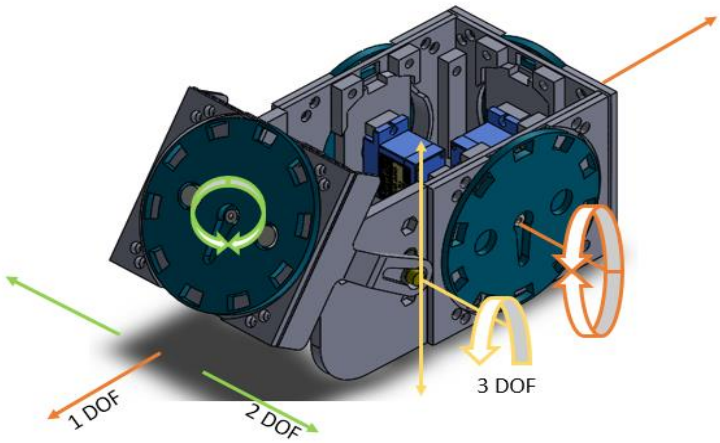


Figura 5-3 Grados de Libertad MECABOT 3.0

En cada rueda del Pivote y Centro se encuentran ubicados un par de imanes para la conexión con otro semi-módulo, uno con polaridad positiva y el otro con polaridad negativa.

Las ruedas ubicadas en las caras: pivote y centro permiten el giro (producido por los Micro-servomotores (5) y (6)) de estas, para el acople o desacople con otro semi-modulo por medio de imanes. Así entonces, si la polaridad coincide, los módulos se repelen y si son opuestos se conectan.

Las caras Pivote y Centro permiten formar diferentes conexiones como lo son: Pivote - Centro, Pivote - Pivote y Centro-Centro como se puede detallar en las secciones 4.4 y 4.5.

Las ruedas laterales permiten el desplazamiento hacia adelante y atrás (producido por los Micro-servomotores (3) y (4)), que son útiles para cumplir funciones cooperativas. Estas ruedas no son utilizadas para el presente trabajo de grado, debido a que el movimiento se hace únicamente por movimientos corporales proporcionados por la articulación del pivote.

El pivote actúa como articulación permitiendo la rotación (producida por los Micro-servomotores (1) y (2)) de un semi-módulo con respecto a otro. Con el pivote se consigue generar las configuraciones tipo cabeceo y viraje dependiendo del posicionamiento vertical u horizontal del módulo como se ve en la Figura 5-4, su estructura se modifica debido al tamaño de los motores y a que se desea aumentar el ángulo de giro.

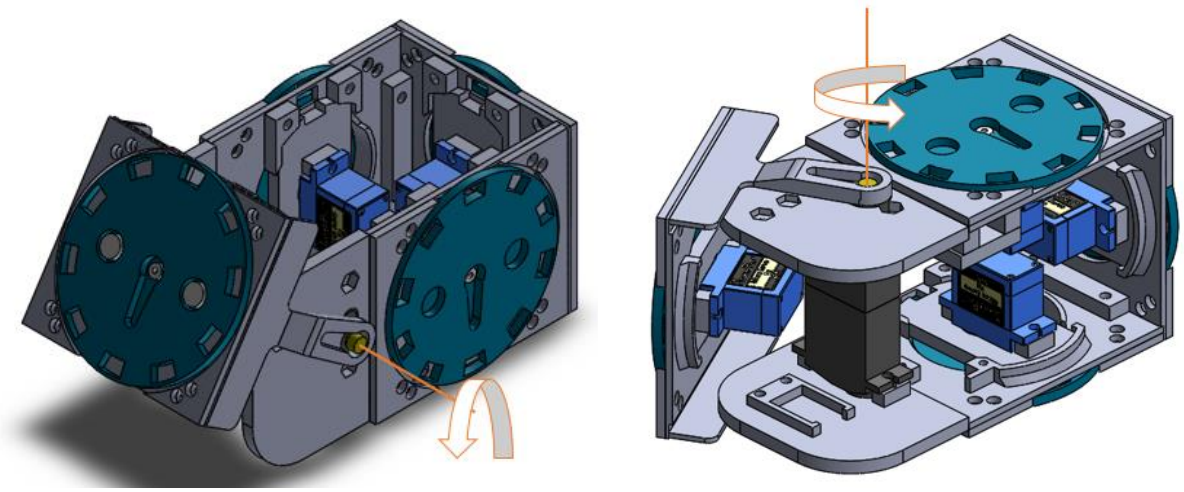


Figura 5-4 Configuraciones Semi-módulos MECABOT 3.0.
(Izq) configuración cabeceo (posición vertical) (Der)configuración viraje (posición horizontal).

Las piezas fueron construidas a través de una impresora 3D, el proceso de ensamblaje se detalla a continuación:

1. Lijado de las piezas para retirar el material sobrante que tienen al salir de la impresora, en la Figura 5-5 se evidencia el estado de las piezas sin lijar.

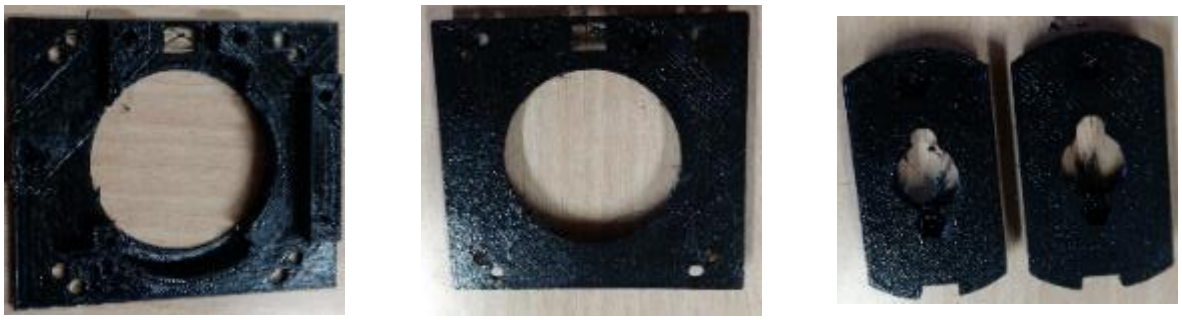


Figura 5-5 Estado de las piezas construidas por la impresora 3D.

2. Ubicación de tuercas para sujetar los elementos electrónicos por medio de tornillos Bristol M5 y los tornillos que sujetan la tarjeta *Teensy* que, al ser más largos, tuvieron que ser cortados (Figura 5-6).

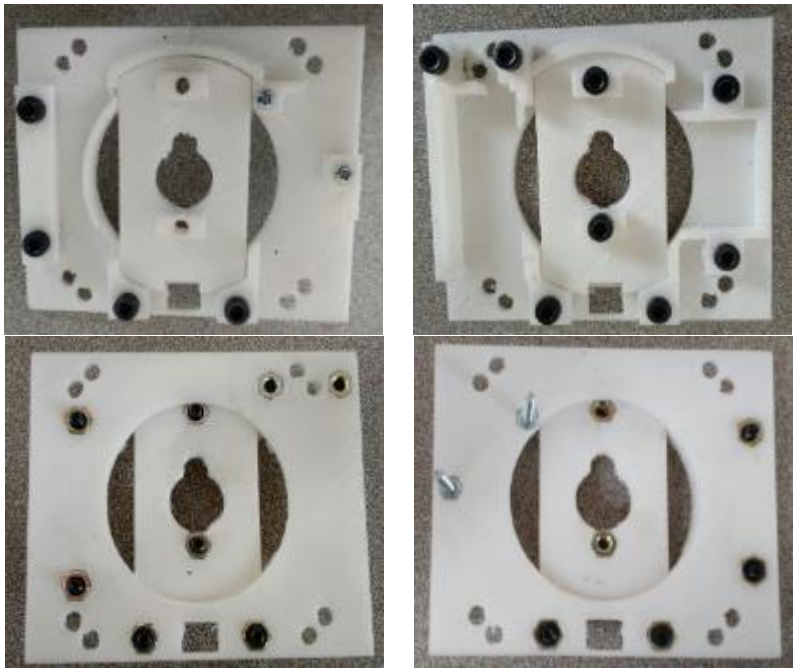


Figura 5-6 Caras semi-módulo MECABOT 3.0 con tuercas y tornillos.

3. Ubicación de los elementos electrónicos en el espacio destinado para cada uno de ellos (ver Figura 5-7).

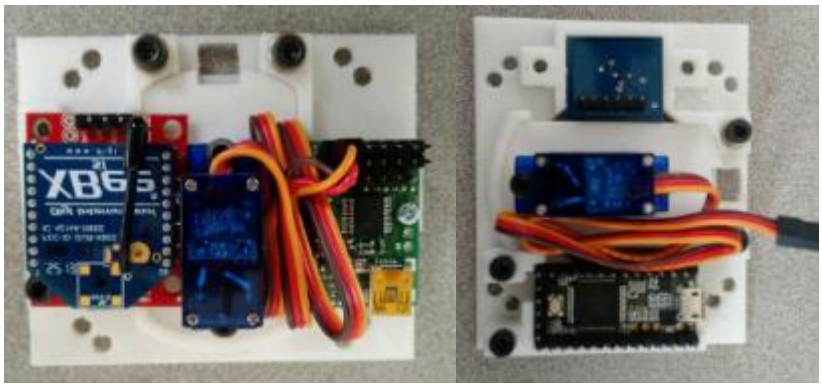


Figura 5-7 Componentes electrónicos integrados en semi-módulo MECABOT 3.0

4. Corte y soldadura de los cables de los servomotores al tamaño justo para ser conectados con el Micro Maestro 6-channels (ver Figura 5-8)

Figura 5-8 Corte de los cables en los servomotores.

5. Unión de las caras con cloruro de metileno para formar el semi-módulo como se evidencia en la Figura 5-9 y Figura 5-10.

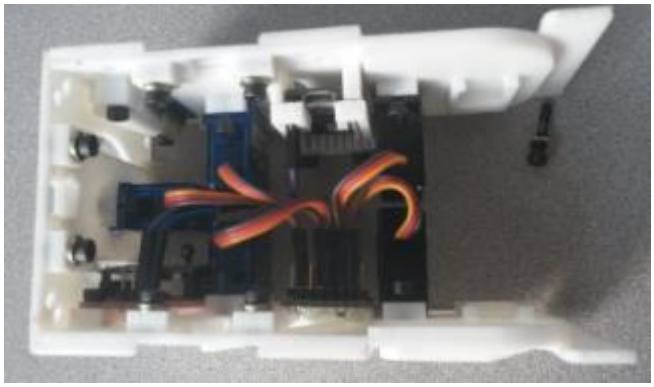


Figura 5-9 Vista superior ensamble semi-módulo MECABOT 3.0.

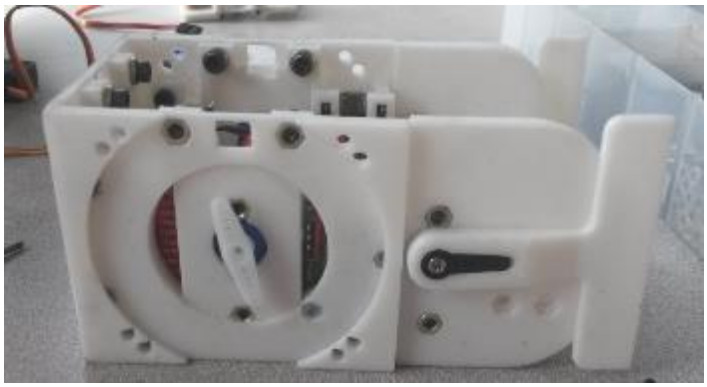


Figura 5-10 Vista lateral derecha ensamble semi-módulo MECABOT 3.0.

6. Soldadura de cables jumper hembra para realizar las conexiones de alimentación y comunicación entre los elementos electrónicos y conexión para su correcto funcionamiento en la Figura 5-11 se muestran los semi-módulos completamente conectados y ensamblados.

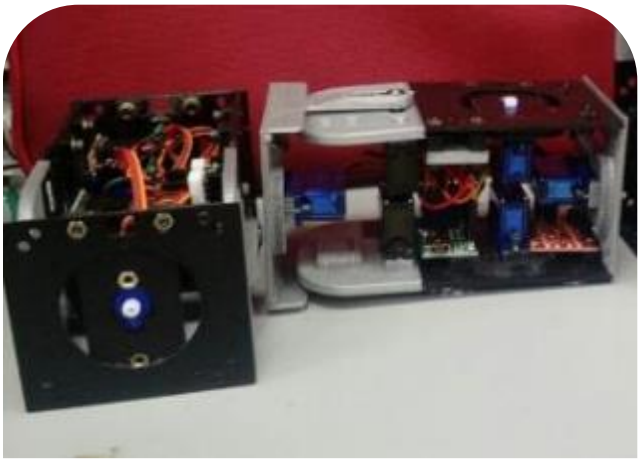


Figura 5-11 Semi-módulos MECABOT 3.0.

7. Construcción de un total de 10 semi-módulos MECABOT 3.0.

Figura 5-12 Construcción 10 semi-módulos MECABOT 3.0

5.2 Modificaciones al Diseño Electrónico de MECABOT 3.0

Los semi-módulos tienen toda la electrónica integrada lo que justifica su tamaño. Las dimensiones generales en unidades de milímetros se pueden detallar en la Figura 5-13.

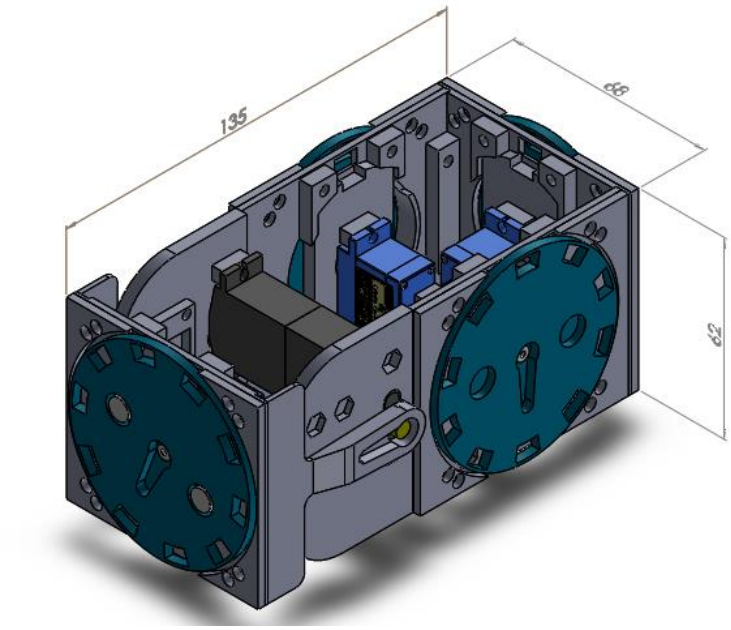


Figura 5-13 Dimensione en miíimetros semi-módulo MECABOT 3.0.

Para el presente trabajo de grado se utilizó una fuente de voltaje DC para alimentar los semi-módulos con salida a 5V 10A. A continuación, se describen los componentes electrónicos como actuadores, módulo de comunicación, driver, tarjeta de desarrollo con los que cuenta un semi-módulo MECABOT 3.0.

5.2.1 Motores

Para este proyecto, se cambiaron los motores implementados en MECABOT 3.0, cada semi-módulo cuenta con 6 servomotores como se evidenció en la sección 5.1. Se utilizan 3 referencias distintas las cuales son detalladas a continuación según su ubicación.

5.2.1.1 Micro-servomotor MG90S:

Los Micro-servomotores referencia MG90S (ver Figura 5-14) son usados para la rotación del Pivote. Tienen piñonería metálica, son de tamaño pequeño, con salida de potencia para poco peso y pueden rotar 180° aproximadamente.

Especificaciones MG90S (Tower Pro, 2016):

- Peso: 13.4 g
- Torque: 1.8 *kgf·cm* (4.8V)
- Velocidad de operación: 0.1 s/60 grados.
- Voltaje: 4.8 V -6.0 V

Figura 5-14 Micro-servomotor MG90S del Pivote (Tower Pro, 2016)

5.2.1.2 Micro-servomotor SG90:

Los Micro-servomotores SG90 (Figura 5-15) son usados en la cara Pivote y cara Centro. Estos tienen piñonería plástica, son pequeños, ligeros, tienen alta potencia de salida y pueden girar 180°.

Especificaciones SG90 (Tower Pro, 2016):

- Peso: 9 g
- Torque: 1.8 *kgf·cm*
- Velocidad de operación: 0.1 s/60 grados.
- Voltaje: 4.8 V -5.0 V.

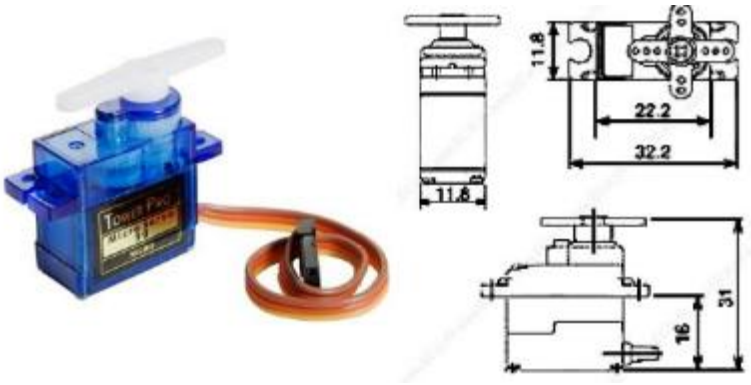


Figura 5-15 Micro-servomotor SG90 ubicado en las caras Pivote y Centro.
(Tower Pro, 2016)

5.2.1.3 Micro-servomotores FS90:

Los Micro-servomotores FS90 son usados en cada una de las caras laterales del semi-módulo. Estos tienen piñonería plástica, son pequeños, ligeros y de rotación continua.

Especificaciones FS90 (MANTECH ELECTRONICS, 2016):

- Peso: 9 g
- Torque: 1.3 *kgf·cm*
- Velocidad de operación: 0.12 s/60 grados.
- Voltaje: 4.8 V

5.2.2 Driver

Cada semi-módulo cuenta con un servo-controlador *Micro Maestro 6- Channel* de Pololu (Figura 5-16). Es un controlador muy versátil, compacto (0.85'' x 1.20''), compatible con tres métodos de control: USB para conexión con el computador, serial TTL para uso con sistemas embebidos y secuencias de comandos interno para aplicaciones. Los canales pueden ser configurados como: salidas de servos, salidas digitales o entradas analógicas. Los pulsos enviados a los servos son de alta resolución (tienen variación de menos de $200\,ns$), lo que los hace adecuados para aplicaciones en robótica. Cuenta con un programa de configuración y control gratuito, por lo que es fácil de probar a través de su conexión por USB. El Micro Maestro tiene 1kB de memoria interna de escritura que permite el almacenamiento de posiciones de los servos

que se pueden reproducir automáticamente sin ninguna computadora o un microcontrolador externo conectado (Pololu, 2016)

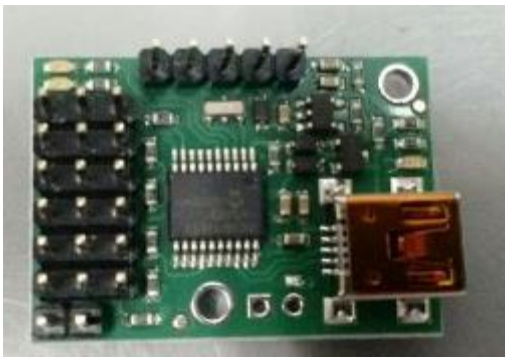


Figura 5-16 Servo-controlador Micro Maestro 6- Channel Pololu.

5.2.3 Tarjeta de Procesamiento

La tarjeta de desarrollo utilizada es: *Teensy 3.2* (ver Figura 5-17) basada en un sistema microcontrolador. Esta versión cuenta con un procesador ARM de 32 bits, dentro de sus características principales, están:

- Pulsador único de programación.
- Es compatible con software Arduino y sus bibliotecas.
- Fácil de utilizar ya que cuenta con una aplicación para cargar la programación.
- Herramienta de desarrollo de Software gratuito.
- Funciona con Mas OS, Linux y Windows.
- Tamaño pequeño (PJRC, 2016).

Figura 5-17 Teensy 3.2
(PJRC, 2016)

En la Tabla 5-3 se pueden observar las especificaciones técnicas de la tarjeta *Teensy 3.2*

Característica	Unidades
Procesador	MK20DX256VLH7
Núcleo	Cortex-M4
Velocidad nominal	72 MHz
Frecuencia de Reloj	96 MHz
EEPROM	2 kB
RAM	64kB
Voltaje de entrada	5V
Pines Digitales I/O	34
Entradas análogas	21
Memoria Flash	256 KB
Dimensiones	1.4’’ X 0.7’’

Tabla 5-3 Especificaciones técnicas Teensy 3.2 (PJRC, 2016)

5.3 Caracterización Servomotores

Antes de poner en funcionamiento los Micro-servomotores que se usaran para el movimiento corporal, es decir, los ubicados en el Pivote, se caracterizan por medio del programa Pololu Maestro Control Center. Lo primero es configurar el Baud Rate en la pestaña Serial Setting a 115200. Después, en la pestaña Channel Settings se configura el valor mínimo y máximo de giro de cada servomotor y por último en la pestaña Status, se mueve el Micro-servomor y se caracteriza en diferentes ángulos obteniendo el valor Target en cada uno de estos. Lo anterior se hace con ayuda de un transportador ubicado en el pivote para poder tomar los datos en cada valor.

En Microsoft Excel se recolectan los datos en una tabla, se grafican y se agrega una línea de tendencia, cambiando la función de la ecuación, a la que más se asemeje a la línea de datos graficados (ver Figura 5-18). Cada servomotor tiene su propia ecuación característica que puede, o no, ser lineal; ya que, a pesar de tener el mismo modelo de servomotor, son de diferentes

fabricantes. Las ecuaciones de este par de motores que componen cada semi-módulo se carga en el programa la tarjeta *Teensy 3.2* correspondiente.

Grados	Target
22	871,75
45	1113,5
90	1545,5
135	1973
180	2496

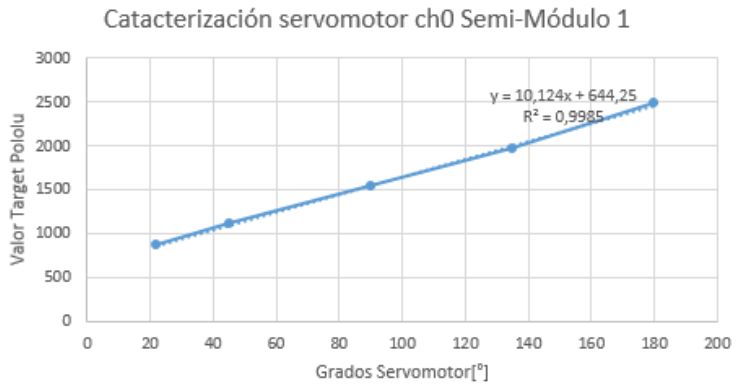


Figura 5-18 Caracterización servomotor ch0 Semi-módulo1.

5.4 Comunicación Mediante ZigBee

ZigBee es un protocolo de comunicaciones inalámbrico creado por ZigBee Alliance para dispositivos electrónicos de bajo consumo como redes de sensores. Se comunica a través de un único canal entre 16 posibles y puede unir hasta 65535 equipos. Presenta como ventajas su bajo costo, bajo consumo de potencia, uso de bandas de radio libres, instalación barata y simple y redes flexibles y extensibles. Permite desarrollar conexiones punto a punto, multipunto, peer to peer (todos los nodos interconectados) o redes complejas de sensores. Una red ZigBee está conformada por tres elementos:

- Coordinador: Es el responsable del canal de comunicaciones y el identificador de la red (PAN ID), permite que se le unan dispositivos Router y End Point, participa en el enrutado de paquetes y origina/recibe la información.
- Router: Se encarga de crear y mantener la mejor ruta para transmitir información, también retransmite paquetes de otros Router o de End Points.
- End Device: Interactúan con el coordinador o un Router para obtener información y transmitirla a otro End Device (Oyarce, 2010).

5.4.1 Circuito de Conexión

La Figura 5-19 muestra las conexiones mínimas para un módulo XBee. Requiere una alimentación de 2.8 a 3.4 V, conexión a tierra y transmisión de datos por los pines TXD y RXD (Oyarce, 2010).

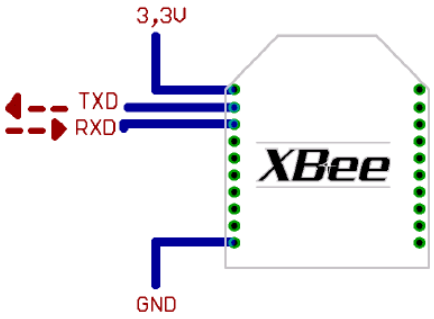


Figura 5-19 Circuito de conexión mínimo para XBee

5.4.2 Configuración Recepción/Transmisión Punto a Punto del Módulo XBee

Los XBee pueden conectarse en diferentes modos de comunicación: recepción/trasmisión, modo bajo consumo, modo comando, modo transparente, operación API e IDLE. Para el presente proyecto de grado se emplea únicamente el modo recepción/transmisión, que es cuando el dispositivo recibe algún paquete mediante la antena o cuando envía una señal para ser transmitida. Esta conexión permite la comunicación mediante la configuración de la dirección mediante un número de 16 bit dentro de la red, va entre 0x0000 y 0xFFFE Figura 5-20. Es la conexión por defecto más sencilla para los dispositivos.

Figura 5-20 Direccionamiento de 16 bit

Para realizar la configuración requerida para la comunicación entre todos los MECABOT 3.0 mediante los XBee se emplea software X-CTU. La interfaz se observa en la Figura 5-21, su descarga es gratuita mediante la página web de X-CTU.

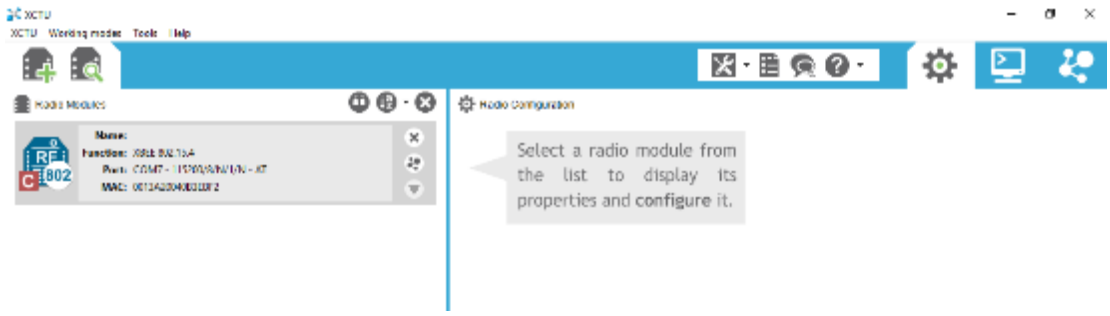


Figura 5-21 Interfaz de usuario software X-CTU

La configuración empleada para el XBee coordinador se observa en la Figura 5-22:

CH Channel	C	SL Serial Number Low	40B3EBF2
ID PAN ID	8888	MM MAC Mode	802.15.4 + MaxStream header w/AC
DH Destination Address High	0	RR XBee Retries	0
DL Destination Address Low	FFFF	RN Random Delay Slots	0
MY 16-bit Source Address	0	NT Node Discover Time	19 x 100 ms
SH Serial Number High	13A200	NO Node Discover Options	0
		CE Coordinator Enable	Coordinator [1]

Figura 5-22 Configuración XBee coordinador

La configuración empleada para los XBee End Device se observa en la Figura 5-23

CH Channel	C	SL Serial Number Low	40897818
ID PAN ID	8888	MM MAC Mode	802.15.4 + MaxStream header w/AC
DH Destination Address High	0	RR XBee Retries	0
DL Destination Address Low	Range: 0x0 - 0xFFFF	RN Random Delay Slots	0
MY 16-bit Source Address	0	NT Node Discover Time	19 x 100 ms
SH Serial Number High	13A200	NO Node Discover Options	0
		CE Coordinator Enable	End Device [0]

Figura 5-23 Configuración XBee End Device

5.5 Trama de Datos para la Comunicación

Dentro de este trabajo se desarrolló a una trama de datos de 6 bits, los dos primeros (bit 0 y bit 1) corresponden a la identificación del módulo ID que se asigna en la tarjeta Teensy (Sección 5.2.3), el siguiente bit (bit 2) identifica el canal del servomotor que debe moverse, los últimos tres

bits (bit 3, bit 4 y bit 5) corresponden al ángulo en grados que se desea que alcance el servomotor como se muestra en la Tabla 5-4.

bit 0	bit 1	bit 2	bit 3	bit 4	bit 5
1	3	4	0	9	3
ID		Motor	Ángulo en Grados		

Tabla 5-4 Trama de datos de comunicación

5.6 Programación Tarjeta de Procesamiento Teensy 3.2

La tarjeta Teensy (Sección 5.2.3) es la encargada de comunicar la información que reciben los XBee End Device al driver (Sección 5.2.2) que controla los servomotores. El esquema de conexión Figura 5-24 ilustra las conexiones físicas que se realizaron para cada MECABOT 3.0.

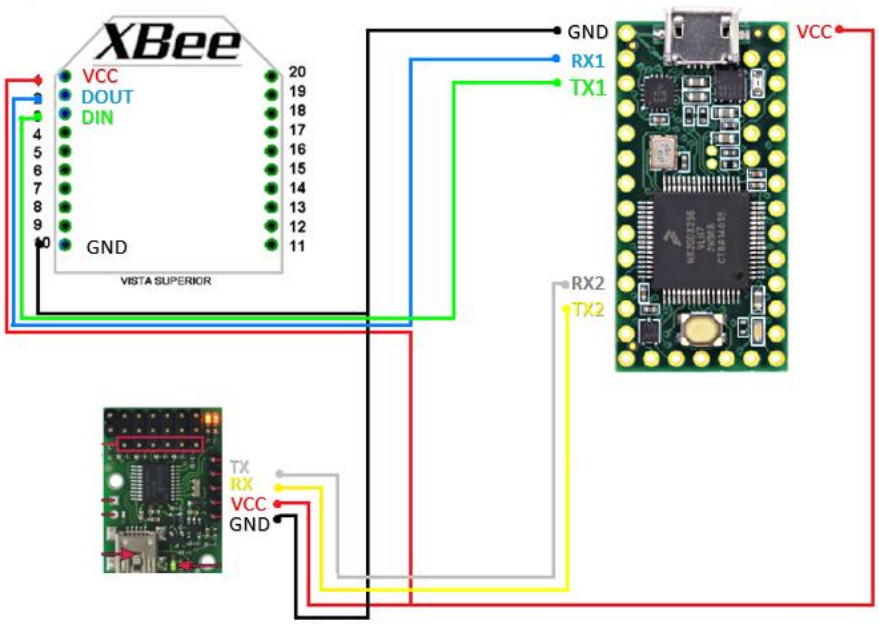


Figura 5-24 Conexiones electrónicas XBee – Pololu driver -Teensy 3.2

La programación de Teensy se realiza mediante Arduino IDE hacienso uso de la librería TeensyDuino. El diagrama de flujo implementado se muestra en la Figura 5-25

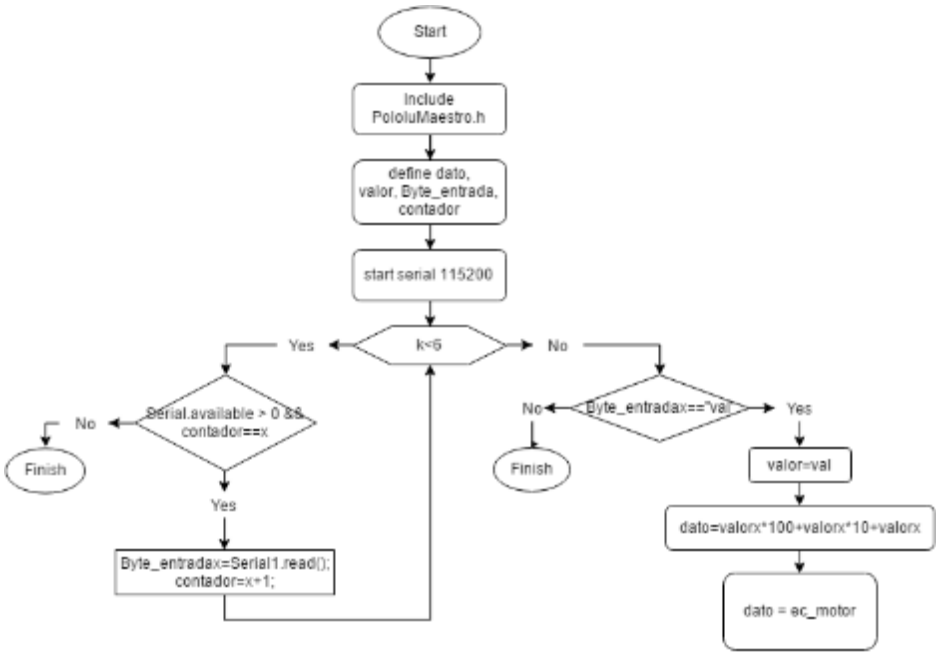


Figura 5-25 Diagrama de flujo código Teensy

5.7 Conclusiones del capítulo

En este capítulo se presentó el diseño mecánico y electrónico del módulo MECABOT 3.0 especificando cada uno de sus componentes, se presentó el proceso de construcción a partir de las piezas en ABS y el ensamblaje de sus unidades. Luego se presentó la caracterización de los servomotores con el fin de obtener la ecuación que rige su movimiento y emplearla en la programación de la tarjeta Teensy de cada módulo. Se detalló la comunicación mediante ZigBee para unir la el computador maestro y el módulo coordinador con los End Device de cada MECABOT 3.0.

6 Pruebas y Resultados

6.1 Introducción

En este capítulo se presenta el desarrollo de la interfaz gráfica de usuario construida en MATLAB® que permite la comunicación entre el computador maestro y los semi-módulos MECABOT 3.0; luego, se presentan las condiciones y los resultados de la implementación realizada en las instalaciones de la Universidad Militar Nueva Granada. Las pruebas se realizan con los 10 semi-módulos construidos, se varían los parámetros k, A_h, M, f, A con el fin de obtener datos de velocidad. Para esto se empleó una superficie con divisiones cada 5 cm ubicando la cadena de acuerdo a una referencia y cronometrando el tiempo que transcurre durante el desplazamiento desde la referencia hasta un punto final.

Se alimenta el sistema con una fuente de voltaje en paralelo para suministrar más corriente (10A) al sistema robótico módular. Se realizan pruebas con los semi-módulos completos, es decir, con todos los componentes electrónicos internos; pero, a medida que se incrementa el número módulos se reduce significativamente la velocidad, al punto de ser nula. Es por esto que se reduce el peso de cada semi-modulo, retirando los motores ubicados en las caras laterales, estos no se utilizan para la generación de movimiento en configuración tipo Oruga y Serpiente. Los valores obtenidos mediante las pruebas se registran en tablas y se grafican para poder ser comparados. Diseño Interfaz de Usuario en MATLAB®.

Para realizar la comunicación final entre el computador desde el que se realizan los cálculos matemáticos que definen la locomoción del sistema robótico modular MECABOT 3.0 se emplea el software MATLAB®, en este se generan dos archivos: un GUI (.fig) o interfaz gráfica de usuario y un archivo de código (.m).

Para la obtención de los indicadores de rendimiento del sistema robótico modular MECABOT 3.0 se toma como variable de rendimiento la velocidad de la cadena en las diferentes configuraciones y conexiones.

6.1.1 Interfaz de Usuario GUI (.fig)

La interfaz de usuario es la encargada de permitir la selección de los parámetros que permiten al controlador ejecutar la locomoción deseada. En la Figura 6-1 se presentan los componentes de la interfaz.

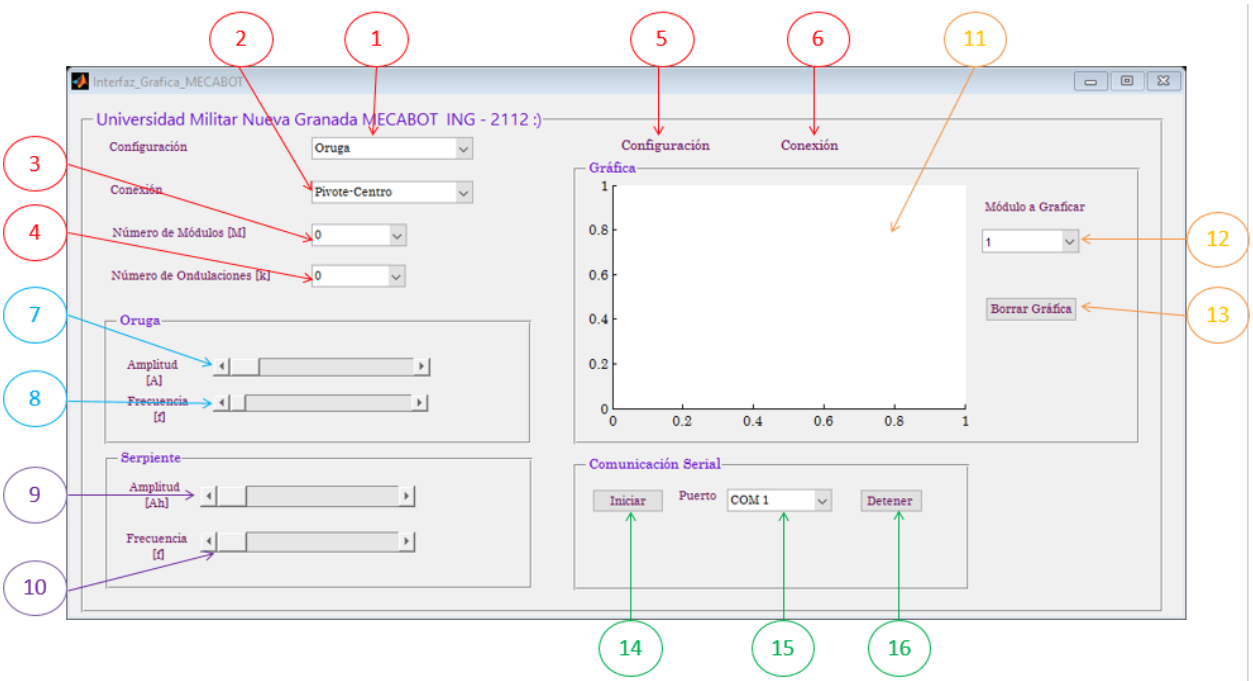


Figura 6-1 Interfaz de usuario en MATLAB®

La Tabla 6-1 indica el número y correspondiente a las funciones que se encuentran en la interfaz de usuario.

Número	Nombre
*1	Selección configuración
*2	Selección conexión
*3	Selección número de módulos [M]
*4	Selección número de ondulaciones [k]
*5	Texto configuración
*6	Texto conexión
*7	Amplitud configuración oruga [A]
*8	Frecuencia configuración oruga [f]
*9	Amplitud horizontal configuración serpiente [Ah]

*10	Frecuencia horizontal configuración serpiente [f]
*11	Área de gráfica
*12	Selección módulo a graficar
*13	Borrar gráfica
*14	Botón de inicio de comunicación
*15	Selección de puerto COM
*16	Botón de detención de comunicación

Tabla 6-1 Funciones interfaz de usuario MATLAB®

Con el fin de aclarar las funciones de cada uno de los objetos presentes en la interfaz, se detallan a continuación:

- Selección configuración: Selecciona la configuración a trabajar, puede ser oruga o serpiente, esta se muestra en el texto configuración.
- Selección conexión: Selecciona la conexión a trabajar, puede ser pivote – pivote, pivote – centro, centro - centro, esta se muestra en el texto conexión.
- Selección número de módulos [M]: Selecciona el número de módulos que componen el sistema robótico modular, dependiendo de la configuración, conexión y número de ondulaciones.
- Selección número de ondulaciones [k]: Selecciona el número de ondulaciones que componen el sistema robótico modular, dependiendo de la configuración y conexión.
- Texto configuración: Muestra la configuración seleccionada en selección configuración.
- Texto conexión: Muestra la conexión seleccionada en selección conexión.
- Amplitud configuración oruga [A]: Selecciona la amplitud de la onda para la configuración tipo oruga.
- Frecuencia configuración oruga [f]: Selecciona la frecuencia de la onda para la configuración tipo oruga.
- Amplitud horizontal configuración serpiente [Ah]: Selecciona la amplitud de la onda para la configuración tipo serpiente.
- Frecuencia horizontal configuración serpiente [f]: Selecciona la frecuencia de la onda para la configuración tipo serpiente.

- Área de gráfica: Permite visualizar una gráfica de los ángulos enviados a los servomotores según el módulo seleccionado en selección módulo a graficar.
- Selección módulo a graficar: Selecciona el módulo del que se desea obtener una gráfica de los ángulos enviados.
- Borrar gráfica: Borra la gráfica presente en el área de gráfica.
- Botón de inicio de comunicación: Inicia la comunicación del computador con el XBee coordinador e inicia el código del controlador.
- Selección de puerto COM: Selecciona el puerto al que está conectado el XBee controlador en el computador:
- Botón de detención de comunicación: Detiene la comunicación entre el computador y el XBee coordinador.

6.1.2 Archivo de Código MATLAB® (.m)

El archivo de código en MATLAB® está conformado por dos secciones principales, la primera es el código generado automáticamente por el software que encarga de la inicialización y funciones del GUI; la segunda es el archivo controlador, encargado de transferir la información de la interfaz gráfica a los módulos mediante la comunicación con el XBee coordinador, y el procesamiento del modelo matemático para la generación de la locomoción del sistema robótico modular, éste se basa en los controladores empleados en Webots para la simulación (Figura 4-16, Figura 4-17). El diagrama de flujo del código en MATLAB® del archivo controlador se presenta en la Figura 6-2.

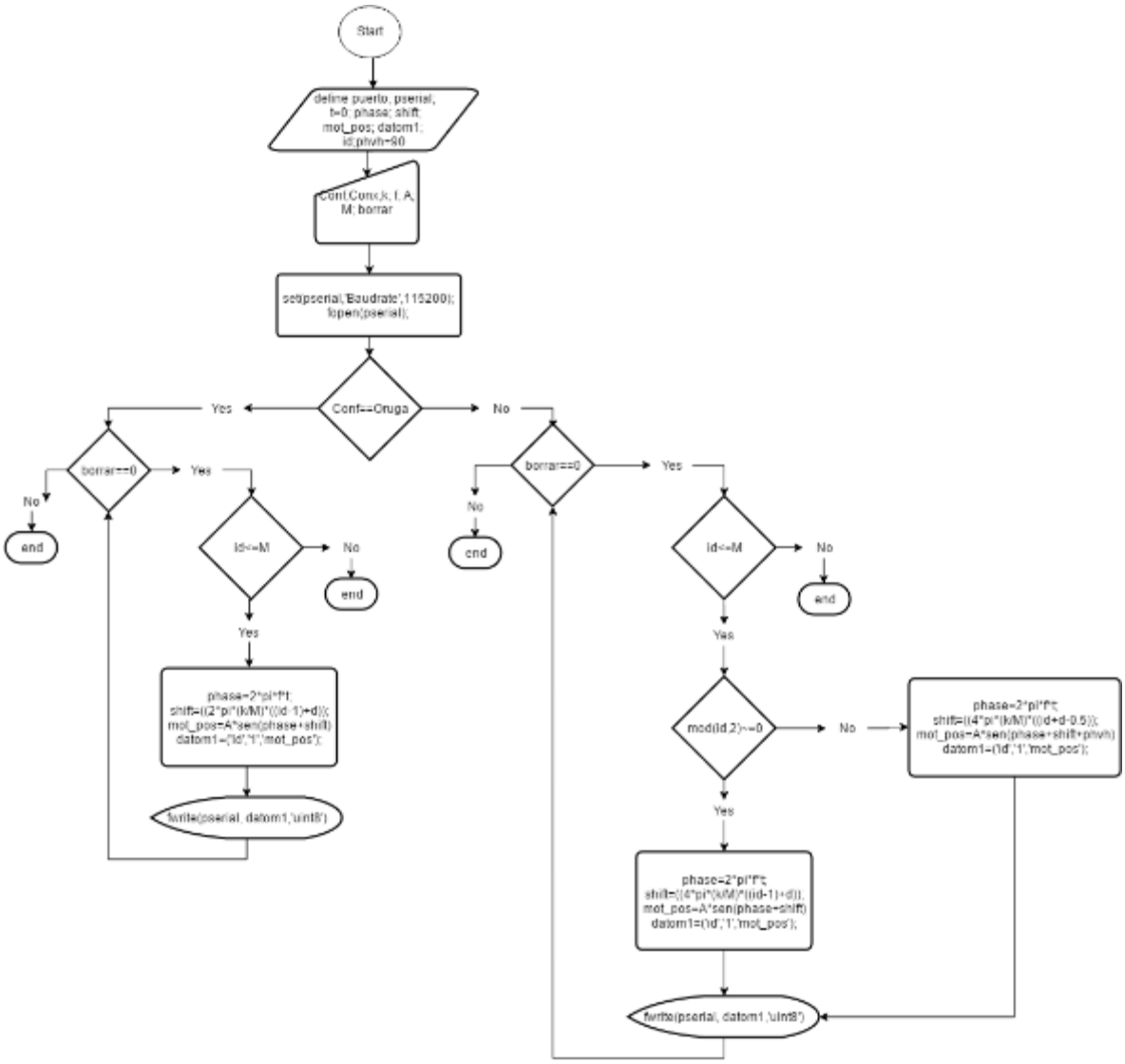


Figura 6-2 Diagrama de flujo código controlador MATLAB®

6.2 Implementación en Configuración Tipo Oruga

Para la implementación se utiliza la interfaz gráfica construida en MATLAB® (Sección 6.1.1). Para realizar las pruebas se varían los parámetros(f, A, M y k)y conexiones para obtener datos de velocidad, al igual que en el caso de las pruebas de simulación; para esto se hace uso de una superficie de *foamy* con divisiones cada 5 cm, este material se utiliza con el fin de que el robot tenga una mayor fricción con la superficie y permita su avance; tambien se hace uso de un cronómetro para la medición de tiempo.

6.2.1 Conexión Pivote-Centro

En la Figura 6-3 se muestra la conformación de un módulo en conexión pivote – centro.

Figura 6-3 Módulo configuración oruga conexión pivote – centro

Al ser la conexión más corta se realiza un mayor número de pruebas con diferente número de módulos. Con $k = 2,\ 4 \leq M \leq 6$, con $k = 3,\ 6 \leq M \leq 8$ y con $k = 4, M\ = 8$. Los datos de frecuencia entre $0.2 \leq f \leq 0.5$ y amplitud $0.2 \leq A \leq 0.5$. A continuación se resgistran los datos de las diferentes pruebas realizadas.

6.2.1.1 Implementación con k=2

Se hacen pruebas con los semi-módulos completos, es decir, con todos los servomotores ubicados al interior de este. Se obtienen los valores de velocidad registrados en Tabla Apéndice B 8-1.

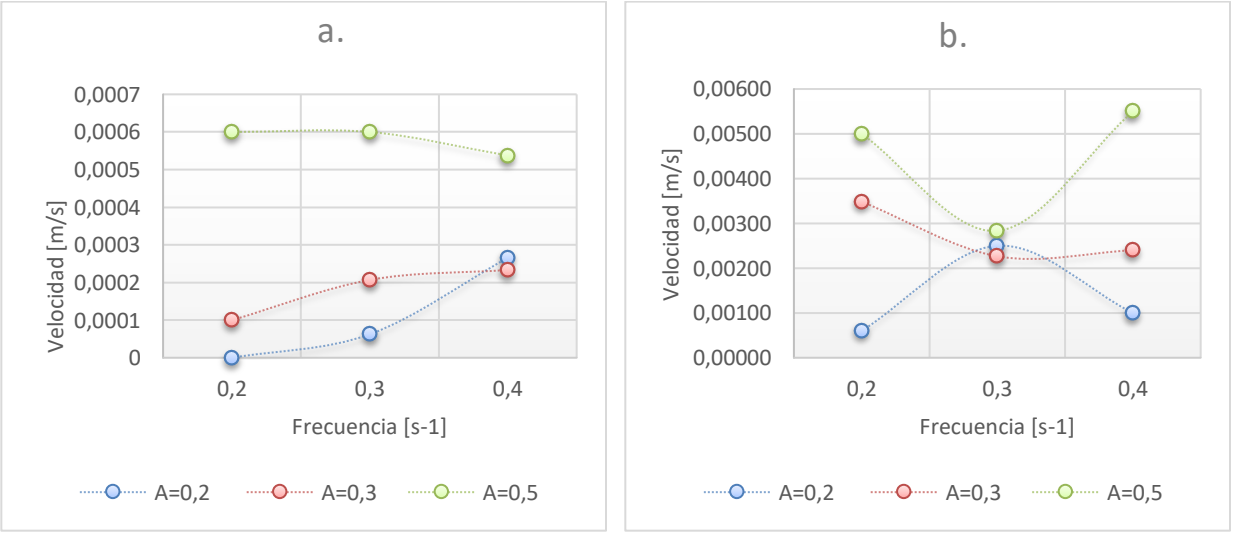


Figura 6-4 Gráficos velocidades configuración tipo Oruga Pivote - Centro k=2.
a. M=6 b. M=4. Semi-módulos completos.

En la Figura 6-4a se observa que al incrementar frecuencia y amplitud aumenta la velocidad, teniendo la máxima velocidad $V = 0.00054\frac{m}{s}$ en $f = 0.4$ y $A = 0.5$, la mínima velocidad es $0{,}00006\frac{m}{s}$ en la Figura 6-4b se tiene un valor pico en $f = 0.3$ para $A = 0.2$ y $A = 0.3$ a partir del cual se presenta un decremento en velocidad, en $f = 0.3$ y $A = 0.5$ existe un valle, para la siguiente frecuencia $f = 0.4$ la velocidad incrementa hasta la máxima velocidad $V = 0.0055\frac{m}{s}$, la mínima velocidad es $0.0006\frac{m}{s}$.

Se reduce el peso de los semi-módulos retirando los servomotores de las ruedas laterales y se hacen las pruebas de nuevo. Los valores de velocidad se registran en la Tabla Apéndice B 8-2 , las pruebas se repiten sólo con *M=4* y *M=5* ya que al tener *M=6* se evidencia que los semi-módulos de los extremos se ponen en contacto con la superficie al mismo tiempo, lo que no permite el avance de la cadena.

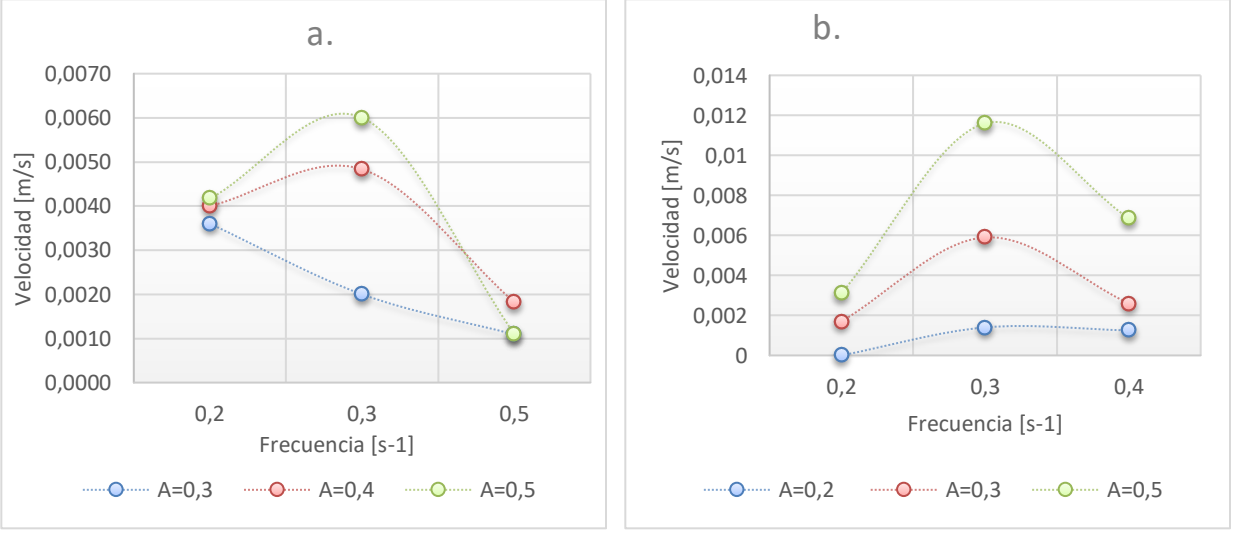


Figura 6-5 Gráficos velocidades configuración tipo Oruga Pivote - Centro k=2.
a. M=5 b. M=4. Semi-módulos Reducidos.

Se evidencia que respecto a la Figura 6-4a en la que se analiza para el módulo completo, las velocidades incrementan considerablemente, aunque la relación velocidad – frecuencia es inversamente proporcional. Se tiene la máxima velocidad $V = 0.006\frac{m}{s}$ en $f = 0.3$ y $A = 0.5$ y la

mínima $V = 0.0011$ en $A = 0.3$ y $f = 0.2$. En la Figura 6-4b se tiene la máxima velocidad $V = 0.0116\frac{m}{s}$ en $f = 0.3$ y $A = 0.5$ y la mínima $V = 0.0013$ en $f = 0.4$ y $A = 0.2$.

6.2.1.2 Implementación con k=3

Se obtienen los valores de velocidad registrados para la cadena con semi-módulos completos en la Tabla Apéndice B 8-3.

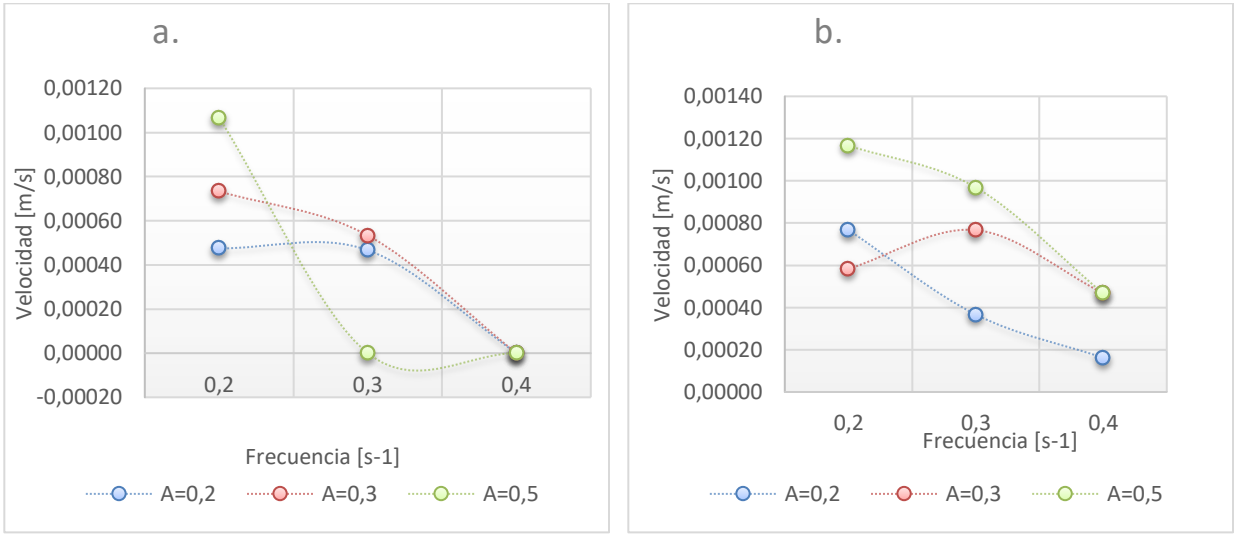


Figura 6-6 Gráficos velocidades configuración tipo Oruga Pivote - Centro k=3.
a. M=8 b. M=6. Semi-módulos Completos.

En la Figura 6-6 se observa que a mayor número de módulos y frecuencia la velocidad disminuye, en la Figura 6-6a el desplazamiento es nulo para la amplitud máxima $A = 0.5$ $f = 0.3, f = 0.4$ y en $f = 0.4$ con $A = 0.2$ y $A = 0.3$, la máxima velocidad $v = 0.00107\frac{m}{s}$ se da en $A = 0{,}2$ $f = 0.2$. En comparación, en la Figura 6-6 al tener un número de módulos menor todas las pruebas fueron medibles, la máxima velocidad $v = 0.00117\frac{m}{s}$ se da en $A = 0{,}2$ $f = 0.2$ y la mínima $v = 0.00016\frac{m}{s}$.

Se obtienen los valores de velocidad para los semi-módulos reducidos registrados en la Tabla Apéndice B 8-4.

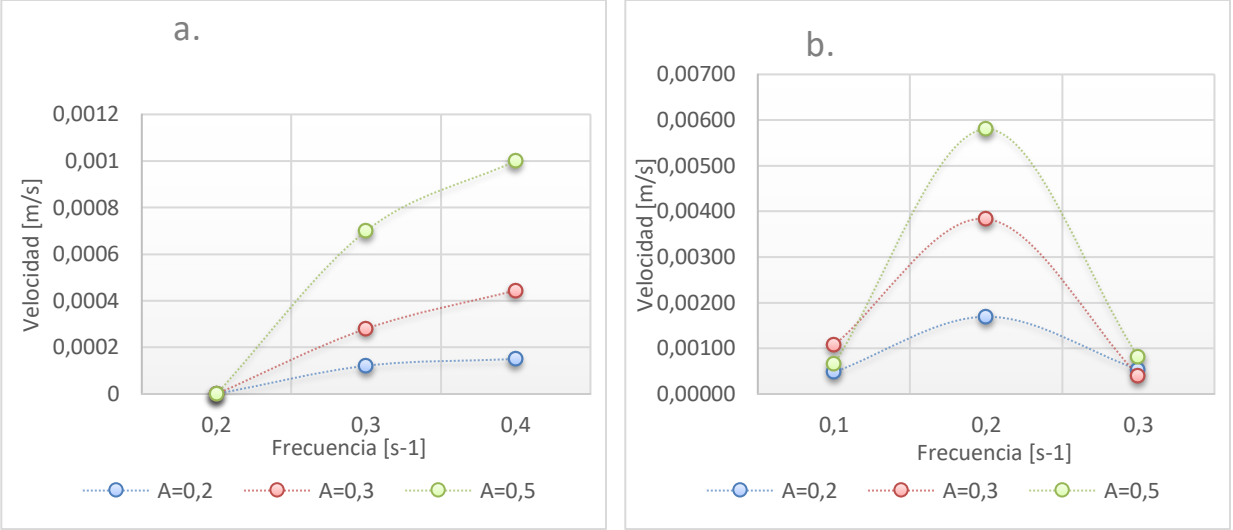


Figura 6-7 Gráficos velocidades configuración tipo Oruga Pivote - Centro k=3.
a. M=8 b. M=6. Semi-módulos Reducidos

En la Figura 6-7 en la que los semi-módulos están reducidos, se observa que con respecto a la Figura 6-6 (semi-módulos completos) se pueden obtener la mayoría de los datos, la Figura 6-7a velocidad incrementa con la frecuencia y la amplitud. La máxima velocidad $v = 0.001\frac{m}{s}$ se da en $A = 0.5\ f = 0.4$, la Figura 6-7b velocidad no aumenta de manera constante respecto a la frecuencia y la amplitud ya que existe un pico en $f = 0.2$ a partir de este valor la velocidad cae. La máxima velocidad $v = 0.0058\frac{m}{s}$ se da en $A = 0.5\ f = 0.2$, La minina velocidad $v = 0.0004\frac{m}{s}$ se da en $A = 0.3\ f = 0.3$.

6.2.2 Conexión Pivote – Pivote

En la Figura 6-8 se muestra la conformación de un módulo en conexión pivote – pivote.

Figura 6-8 Módulo MECABOT 3.0 configuración oruga conexión pivote – pivote

6.2.2.1 Implementación con k=2

Para esta conexión sólo se presentan pruebas con $M = 4, A = 0.5$ debido a que en valores de amplitud son menores, el sistema robótico modular no se desplaza. En la Tabla Apéndice B 8-5 se muestran los resultados.

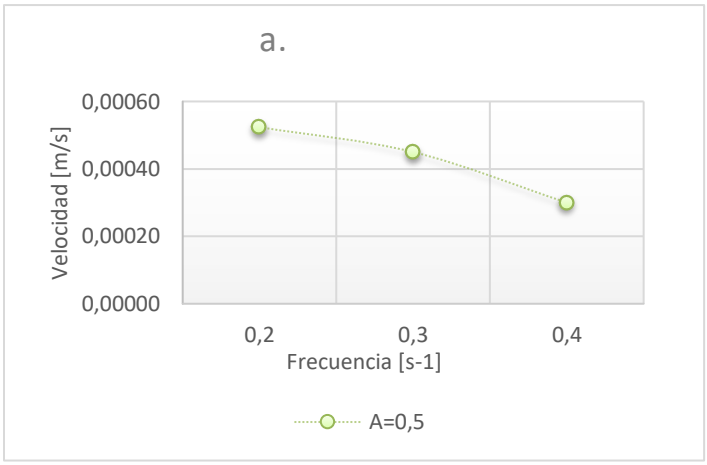


Figura 6-9 Gráficos velocidades configuración tipo Oruga Pivote - Pivote k=2. M=4 Semi-módulos Reducidos

En la Figura 6-9 se aprecia que, a mayor frecuencia, menor es la velocidad. La máxima velocidad $v = 0.00052\,\frac{m}{s}$ se da en $A = 0.5\ f = 0.2$, la mínima velocidad $v = 0.0003\,\frac{m}{s}$ se da en $A = 0.5\ f = 0.4$.

6.2.3 Conexión Centro – Centro

Figura 6-10 Módulo MECABOT 3.0 configuración oruga conexión centro – centro

Al ser esta conexión la que más semi-módulos requiere, la única prueba realizable es con $M = 4$, es decir 10 semi-módulo, la cadena es demasiado larga y pesada para los motores, que no logran elevarla, por tanto, no es factible la toma de datos.

6.3 Implementación en Configuración Tipo Serpiente

Para las mediciones de esta configuración se cuenta con la superficie dividida cada 5 cm en forma de cuadrícula (igual que para el caso de la configuración tipo oruga sección 6.2), la cadena se ubica en la referencia lateral para medir su desplazamiento y ser cronometrado. El valor de A_v se mantiene constante en $A_v = 0.1$. Los parámetros a variar son: k, M, f, A_h. Estas pruebas se realizan con los semi-módulos reducidos.

6.3.1 Conexión Pivote-Centro

6.3.1.1 Implementación con k=2

Los resultados obtenidos de la implementación de la conexión pivote - centro con $k = 2$ se muestran en la Tabla Apéndice B 8-6.

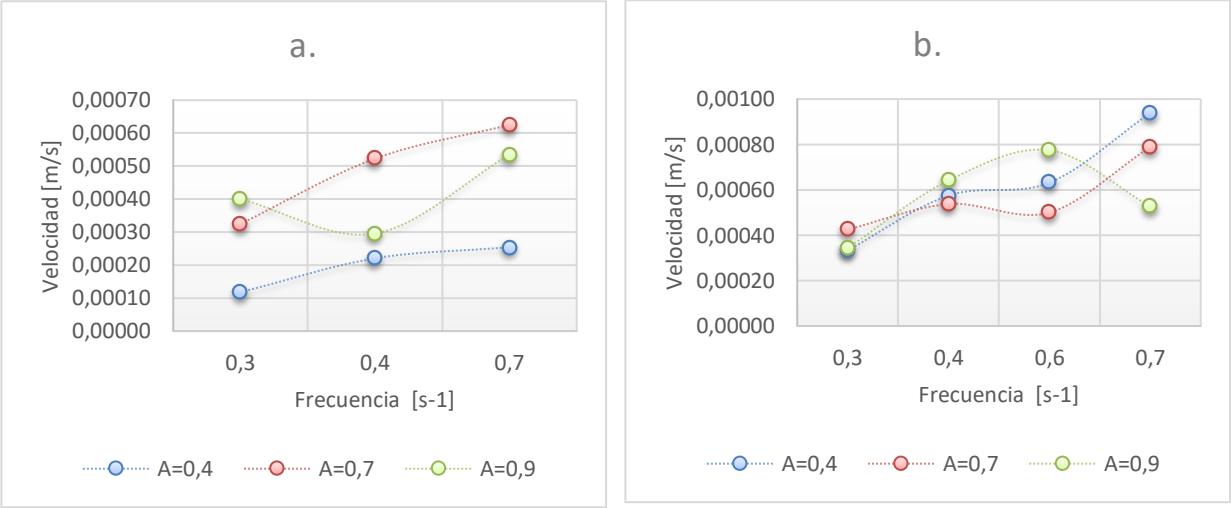


Figura 6-11 Gráficos velocidades configuración tipo serpiente Pivote – Centro k=2.
a. M=4 b. M=6.

En la Figura 6-11a y Figura 6-11b se observa que para $A = 0.4\, y\, A = 0.7$ la velocidad incrementa con la frecuencia, en $A = 0.9$ la amplitud es muy grande para la capacidad de los motores, por lo que, al aumenta la frecuencia disminuye la velocidad.

Para la Figura 6-11a la velocidad máxima se registra en $V = 0.00063\,\frac{m}{s}$ en $A_h = 0.7\ y\ f = 0.7$, la menor en $V = 0.00012\,\frac{m}{s}\ A = 0.7\ f = 0.3$.

Para la Figura 6-11b la velocidad máxima se registra en $V = 0.00094\,\frac{m}{s}$ en$A_h = 0.7\ y\ f = 0.4$, la menor en $V = 0.00012\,\frac{m}{s}\ A = 0.7\ f = 0.3$.

6.3.1.2 Implementación con k=3:

En la Tabla Apéndice B 8-7 se muestran los datos recolectados.

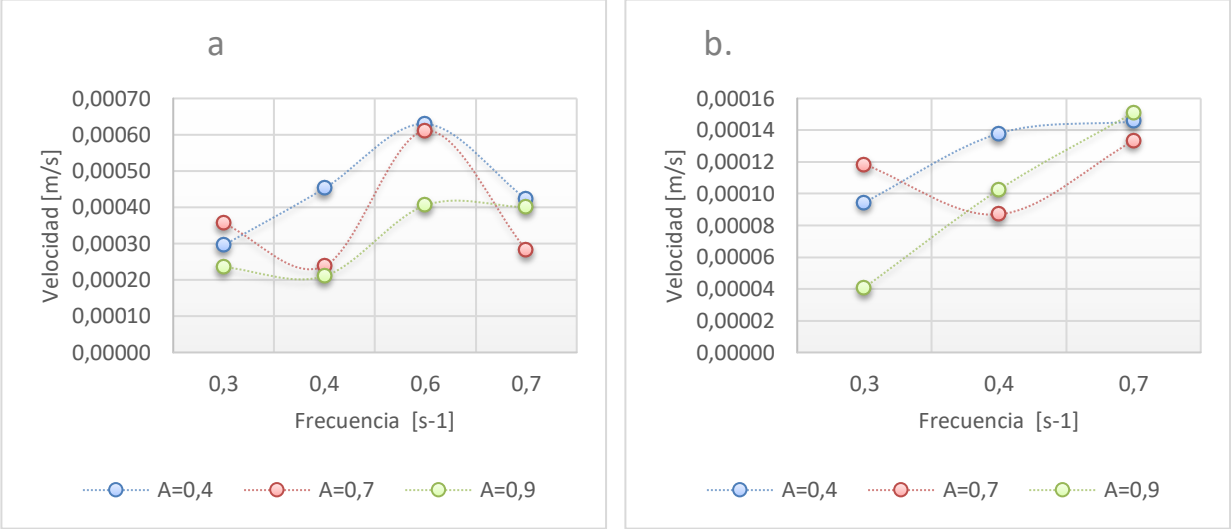


Figura 6-12 Gráficos velocidades configuración tipo serpiente Pivote – Centro k=3.
a. M=6 b. M=8

En la Figura 6-12a, en la amplitud $A_h = 0.4$ se presenta un incremento en la velocidad hasta $f = 0.6$, en donde se encuentra un pico, y a partir de este desciende. Para $A_h = 0.7$ y $A_h = 0.9$ no se obtiene un comportamiento definido. La velocidad máxima se registra en $V = 0.00063\,\frac{m}{s}$ en $A_h = 0.7\ y\ f = 0.7$, la menor en $V = 0.00021\,\frac{m}{s}\ A_h = 0.9\ f = 0.4$.

En la Figura 6-12b, en la amplitud $A_h = 0.4$ y $A_h = 0.9$ se presenta un incremento en la velocidad al aumentar la frecuencia. Para $A_h = 0.7$ no se obtiene un comportamiento definido. La velocidad máxima se registra en $V = 0.00015\,\frac{m}{s}$ en $A_h = 0.7, f = 0.9$ y $A_h = 0.7, f = 0.4$, la menor en $V = 0.00004\,\frac{m}{s}\ A_h = 0.9\ f = 0.3$.

6.3.2 Conexión Pivote – Pivote y Conexión Centro – Centro

Como se explica en la sección 2.3.2.5 el número mínimo de módulos que se requiere para realizar la configuración tipo serpiente, es $M \geq 6$, por tanto, con la cantidad de semi-módulos con los que se cuenta para la realización de este proyecto (sección 4.8.2 y 4.8.3), no es posible construirla.

6.4 Conclusiones del capítulo

En este capítulo se presentó la interfaz de usuario que permite la comunicación entre el computador y el sistema robótico modular para realizar las pruebas de implementación. Estas pruebas se documentaron mediante tablas y gráficas, cada una con su respectivo análisis. Debido al número limitado de semi-módulos no fue posible la realización de las pruebas en configuración tipo serpiente para las conexiones pivote – pivote y centro - centro. Se obtuvieron las siguientes conclusiones:

En simulación, la velocidad aumenta con el incremento de módulos en la cadena, esto difiere de las pruebas en implementación puesto que, a mayor número de módulos el peso del sistema robótico modular es mayor, generando una exigencia a los motores que sobrepasa sus capacidades.

Los rangos de funcionamiento en la implementación, tanto para la tipo configuración oruga como para serpiente, son distintos a los de simulación debido a que: en Webots se asume que el torque de los servomotores es de 10 $kgf * cm$, mientras que los servomotores MG90S (5.2.1) cuentan con un torque de 1.8 $kgf * cm$, los semi-módulos empleados en simulación son únicamente la estructura externa de MECABOT 3.0, no cuentan con los componentes electrónicos o mecánicos debido al elevado requerimiento computacional para ejecutar la simulación; las condiciones de fricción en Webots son ideales, es decir existe fricción con la superficie más no entre los semimódulos; la conexión magnética es rígida, no existe retraso en la comunicación de la información entre el controlador y el sistema robótico modular.

Se obtuvieron los datos de velocidad como indicador del rendimiento del sistema robótico modular MECABOT 3.0 para comparar el funcionamiento en las diferentes configuraciones y conexiones.

7 Conclusiones y Recomendaciones a Trabajos Futuros

Se compararon, analizaron y concluyeron los aspectos positivos y negativos de cada conexión para las configuraciones tipo oruga y tipo serpiente a partir de la velocidad como indicador de rendimiento del sistema robótico modular MECABOT 3.0.

Se constató la funcionalidad del simulador Webots al facilitar las herramientas para determinar las propiedades físicas del sistema robótico modular y su entorno permitiendo obtener los valores de velocidad requeridos.

Se comprobó la aplicabilidad de los generadores sinusoidales de frecuencia fija propuestos por (Gonzalez Gómez, 2008) al sistema robótico modular en cadena MECABOT 3.0, ya que, mediante la simulación y la implementación del modelo en tiempo discreto, se observa la representación de la curva serpentinoide a lo largo de la estructura, generando la locomoción en una y dos dimensiones.

Se verificó que el modelo de los generadores sinusoidales de frecuencia fija se puede implementar en un archivo controlador en lenguaje de código C y permite la generación de la locomoción en configuración tipo oruga y serpiente.

Al realizar la construcción, caracterización y comunicación de los semi-módulos MECABOT 3.0 se facilita el desarrollo de futuras configuraciones como rueda y hexápodo, para así lograr un avance en la investigación en el campo de la robótica modular en la Universidad Militar Nueva Granada.

Se evidenciaron dificultades en la locomoción que se podrían atenuar mediante variaciones en el diseño, se sugieren las siguientes ideas para tener en cuenta en un trabajo futuro:

- El estudio de fuerzas fricción entre el material ABS del que están construidas las piezas de los semi-módulos y diferentes superficies; para poder evaluar que materiales son los más indicados para optimizar la locomoción del robot.

- El cambio de estos servomotores por unos de mayor torque, manteniendo la característica de piñonería metálica.
- La implementación de un sistema que asegure la unión entre los semi-módulos.
- El uso de imanes con mayor fuerza magnética.
- Las dimensiones de los lados de la cara pivote y centro sean iguales para la conexión tipo serpiente.
- La habilitación de un espacio mayor para el XBee Pro Serie 1.

Referencias

Akiya Kamimura, H. K. (2005). Automatic Locomotion Design and Experiments for a Modular Robotic System. *IEEE/ASME Transactions on Mechatronics, 10*(3), 314 - 325.

Althoff, M., Giusti, A., & Icer, E. (2016). A task-driven algorithm for configuration synthesis of modular robots. *International Conference on Robotics and Automation (ICRA)* (págs. 5203-5209). Stockholm: IEEE.

Arai, T., Inagaki, S., & Yuasa, H. (2003). CPG model for autonomous decentralized multi-legged robot system—generation and transition of oscillation patterns and dynamics of oscillators. *Robotics and Autonomous Systems*, 171 - 179.

Aranda, E., Salgado, T., & Velasco, M. (2002). Control no Lineal Discontinuo de un Robot Móvil. *Computación y Sistemas*, 042- 049.

Aranda, M. M., Carbajal, M., González, G. S., Guzmán, V. M., Lozada, J. C., Ortigoza, G. S., . . . Vilchis, M. A. (2012). Wheeled Mobile Robots: A Review. *IEEE Latin America Transactions*, 2209 - 2217.

Behar, A., Castano, A., & Will, P. M. (2002). The Conro Modules for Reconfigurable Robots. *IEEE/ASME Transactions on Mechatronics, 7*(4), 403 - 409.

Bilgen, Y., Brown, B., Choset, H., Enner, F., Ford, S., Layton, C., . . . Willig, A. (2014). Design and Architecture of a Series Elastic Snake Robot. *IEEE/RSJ International Conference on Intelligent Robots and Systems* (págs. 4630 - 4636). Chicago: IEEE.

Bojinov, H., Casal, A., & Hogg, T. (2000). Multiagent Control of Self-reconfigurable Robot. *Fourth International Conference on MultiAgent Systems.* Palo Alto: IEEE.

Brandt, D., Christensen, D. J., & Hautop Lund, H. (2007). ATRON Robots: Versatility from Self-Reconfigurable Modules. *Internatonal conference on Mechatronics and automation* (págs. 26-32). IEEE.

Bucher, D., & Marder, E. (2001). Central pattern generators and the control of rhythmic movements. *Current Biology*, 986-996.

Calles, C., Hernández, M., Ortiz, M., & Rodríguez, J. (2015). *Robótica: Análisis, Modelado, Control e Implementación.* México: OmniaScience.

Calonge, T., & de la Fuente, M. (1999). *Aplicaciones de las Redes de Neuronas en Supervisión, Diagnosis y Control de Procesos.* Venezuela: Equinoccio.

Castellanos, G., Delgado, E., & Giraldo, L. F. (2006). *Cinemática Inversa de un Brazo Robot Utilizando Algoritmos Genéticos.* Bogotá: Universidad Nacional de Colombia.

Cheng, S., Gonzalez-Gomez, J., Jianwei, Z., Xie, Z., & Zhang, H. (2008). Development of a Low-cost Flexible Modular Robot GZ-I. *International Conference on Advanced Intelligent Mechatronics* (págs. 223-228). Xi'an: IEEE/ASME.

Cortes, F., Linares, D., Melo, K., & Patino, D. (2011). A distributed model predictive control (D-MPC) for modular robots in chain configuration. *Robotics Symposium, 2011 IEEE IX Latin American and IEEE Colombian Conference on Automatic Control and Industry Applications (LARC)* (págs. 1-6). Bogotá: IEEE.

Crespi, A., & Ijspeert, A. J. (2007). Online trajectory generation in an amphibious snake robot using a lamprey-like central pattern generator model. *IEEE International Conference on Robotics and Automation* (págs. 262-268). Roma: IEEE.

Cui, X., Tang, S. Z., Wang, X., Zhao, J., & Zhu, Y. (2013). Design and implementation of UBot: A modular Self-Reconfigurable Robot. *IEEE International Conference on Mechatronics and Automation* (págs. 1217-1222). Takamatsu: IEEE.

Cyberbotics Ltd. (2014). *Webots Reference Manual release 7.4.3.* Cyberbotics Ltd.

Cyberbotics Ltd. (2014). *Webots User Guide release 7.4.3.* Cyberbotics Ltd.

Dario, P., Menciassi, A., Sfakiotakis, M., la Spin, G., & Tsakiris, D. P. (2005). Polychaete-like Undulatory Robotic Locomotion. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 3018-3023.

Dobra, A. (2014). General classification of robots. Size criteria. *2014 23rd International Conference on Robotics in Alpe-Adria-Danube Region (RAAD)* (págs. 1-6). Smolenice: IEEE.

Dowling, K. J. (1997). *Limbless Locomotion: Learning to Crawl with a Snake Robot.* Pittsburg: NASA.

Duff, D. G., Roufas, D., & Yim, M. (2000). Polybot: a Modular Reconfigurable Robot. *International Conference on Robotics & Automation* (págs. 514-520). San Francisco: IEEE.

Fukuoka, Y., Hada, Y., Kimura, H., & Takase, K. (2003). Adaptive Dynamic Walking of a Quadruped Robot 'Tekken' on Irregular Terrain Using a Neural System Model. *lnternational Conference on Robotics &Automation* (págs. 2037 - 2042). Taipei: IEEE.

García, R. M. (2010). *Diseño y Simulación de un Algoritmo para el Control de un Robot Modular tipo Cadena.* Bogotá: Universidad Nacional de Colombia.

Gonzalez Gómez, J. (2008). *Robótica Modular y Locomoción: Aplicación a Robots Ápodos.* Madrid: Universidad Autónoma de Madrid.

Gorrostieta, E., & Vargas Soto, E. (2008). Algoritmo Difuso de Locomoción Libre para un Robot Caminante de Seis Patas. *Universidad Nacional Autónma de México*.

Granados, M., Melo, K., & Paez, L. (2013). Conceptual design of a modular snake origami robot. *International Symposium on Safety, Security, and Rescue Robotics (SSRR)* (págs. 1-2). Linköping: IEEE.

Hernandez, M., Calderon, F., Melo, K., Paez, L., Parra, C., & Velasco, A. (2011). Preliminary studies on modular snake robots applied on de-mining tasks. *IX Latin American and IEEE Colombian Conference on Automatic Control and Industry Applications (LARC)* (págs. 1-6). Bogotá: IEEE.

Hirose, S. (1993). *Biologically inspired robots (Snake-like locomotor and manipulator).* Oxford Science Press.

Honda. (2016). *ASIMO*. Obtenido de ASIMO HONDA: http://asimo.honda.com

Hurtado Erasso, C. A., & Rubiano Montaña, O. G. (2013). *Diseño y simulación de un robot modular reconfigurable.* Bogotá: Universidad Militar Nueva Granada.

Ijspeert, A. J., & Crespi, A. (2008). Online Optimization of Swimming and Crawling in an Amphibious Snake Robot. *IEEE Transactions on Robotics*, 75-87.

Kamimura, A., Kokaji, S., Kurokawa, H., Tomita, K., & Yoshida, E. (2005). Distributed Self-reconfiguration Control of Modular Robot M-TRAN. *International Conference on Mechatronics & Automation* (págs. 255-259). Niagara Falls: IEEE.

KeJun, N., & Wörgötter, F. (2009). A Novel Concept for Building a Hyper-Redundant Chain Robot. *IEEE TRANSACTIONS ON ROBOTICS*, *25*(6), 1237 -1249.

Kirkner, R. M. (2007). *Aprendizaje por Refuerzo en Robots Modulares Autoreconfigurables.* Buenos Aires: Facultad de Ciencias Exactas y Naturales Universidad de Buenos Aires.

Kova, R., Rubenstein, M., & Shen, W.-M. (2009). SINGO: A Single-End-Operative and Genderless Connector for Self-Reconfiguration, Self -Assembly and Self-Healing. *IEEE International Conference on Robotics and Automation* (págs. 4253 - 4258). Kobe: IEEE.

Liedke, J., Matthias, R., Winkler, L., & Wörn, H. (2013). The Collective Self-Reconfigurable Modular Organism (CoSMO). *International Conference on Advanced Intelligent Mechatronics (AIM)* (págs. 1-6). Wollongong: IEEE/ASME.

MANTECH ELECTRONICS. (15 de 09 de 2016). Obtenido de http://www.mantech.co.za/Datasheets/Products/FITEC_FS90.pdf

Mataric, M. J. (2007). What Is a Robot?: Defining Robotics. En M. J. Mataric, *What Is a Robot?: Defining Robotics* (págs. 1 - 6). MIT Press.

Melo, K., Calderon, F., Hernandez, M., Paez, L., Parra, C., & Velasco, A. (2011). Central pattern generators and hormone inspired messages: A hybrid control strategy to implement motor primitives on chain type modular reconfigurable robots. *International Conference on Robotics and Automation* (págs. 1014 - 1019). Shangai: IEEE.

Méndez-Polanco, J. A., & Muñoz-Meléndez, A. (2008). Collaborative Robots for Indoor Environment Exploration. *On Control, Automation, Robotics and Vision* (págs. 359-364). Hanoi: IEEE.

Mobile Robots. (10 de Junio de 2016). *Adept Mobile Robots*. Obtenido de http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx

Oyarce, A. (2010). *Guía del usuario.* Santiago de Chile: MCI Electronics.

Paez, L., Melo, K., & Parra, C. (2011). Center of mass displacements using rolling gaits for modular robots on the outside of pipes. *Robotics Symposium, 2011 IEEE IX Latin American and IEEE Colombian Conference on Automatic Control and Industry Applications (LARC).* Bogotá: IEEE.

Pardo Puentes, E. X. (2015). *Rediseño e implementación de un módulo robótico para sistema colaborativo.* Bogotá: Universidad Militar Nueva Granada.

Parra, C., Melo, K., & Paez, L. (2012). Indoor and outdoor parametrized gait execution with modular snake robots. *International Conference on Robotics and Automation (ICRA)* (págs. 3525 - 3526). Saint paul minnesota: IEEE.

PIONEER. (2015). *Mobile robots*. Obtenido de http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx

PJRC. (15 de 09 de 2016). Obtenido de https://www.pjrc.com/teensy/teensy31.html#specs

Pololu. (15 de 09 de 2016). Obtenido de https://www.pololu.com/product/1350

Real Academia Española. (2016). *Diccionario de la Lengua Española*. Obtenido de http://dle.rae.es/?w=robotica

Rus, D., Salemi, B., Shen, W.-m., & Yim, M. (2007). Modular Self-Reconfigurable Robot Systems [Grand Challenges of Robotics]. *IEEE Robotics & Automation Magazine*, 43-52.

Salemi, B., Moll, M., & Shen, W.-M. (2006). SUPERBOT: A deployable, multi-functional. and Modular self-reconfigurable robotic system. *International Conference on Intelligent Robots and Systems* (págs. 3636 - 3641). Beijing: IEEE.

Sparkfun. (15 de 09 de 2016). Obtenido de https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf

Stoy, K. (2006). The Deformatron Robot: a Biologically Inspired. *IEEE International Conference on Robotics and Automation* (págs. 2527 - 2531). Orlando: IEEE.

Tower Pro. (15 de 09 de 2016). Obtenido de http://www.towerpro.com.tw/product/mg90s-3/

Tower Pro. (15 de 09 de 2016). Obtenido de http://www.towerpro.com.tw/product/sg90-7/

Yim, M. (1994). *Locomotion with a unit-modular reconfigurable robot.* California: Stanford University.

8 Anexos

APÉNDICE 1 Tablas recolección pruebas de Simulación

k = 2			
Número de Módulos	**Amplitud**	**Frecuencia [s^{-1}]**	**Velocidad [m/s]**
10	0,1	0,5	0,0075
		1	0,01
		2	0,013
	0,3	0,5	0,024
		1	0,041
		2	0,046
	0,5	0,5	0,032
		1	0,0505
		2	0,079
8	0,1	0,5	0,0055
		1	0,0085
		2	0,012
	0,3	0,5	0,016
		1	0,0285
		2	0,039
	0,5	0,5	0,025
		1	0,0505
		2	0,0375
6	0,1	0,5	0,0045
		1	0,007
		2	0,01
	0,3	0,5	0,0125
		1	0,0205
		2	0,0285
	0,5	0,5	0,02
		1	0,0345
		2	0,0315
4	0,1	0,5	0,0005
		1	0,0005
		2	0,0035
	0,3	0,5	0,004

		1	0,0045
		2	0,0055
	0,5	0,5	0,0015
		1	0,0035
		2	0,012

Tabla Apéndice A 8-1 Velocidades conexión Pivote – Centro k = 2

k = 3

Numero de Módulos	**Amplitud**	**Frecuencia [s^{-1}]**	**Velocidad [m/s]**
12	0,1	0,5	0,0035
	0,1	1	0,0065
	0,1	2	0,0105
	0,3	0,5	0,015
	0,3	1	0,025
	0,3	2	0,0375
	0,5	0,5	0,019
	0,5	1	0,0405
	0,5	2	0,048
10	0,1	0,5	0,004
	0,1	1	0,007
	0,1	2	0,013
	0,3	0,5	0,0125
	0,3	1	0,0215
	0,3	2	0,031
	0,5	0,5	0,024
	0,5	1	0,0415
	0,5	2	0,0235
8	0,1	0,5	0,003
	0,1	1	0,005
	0,1	2	0,0065
	0,3	0,5	0,0065
	0,3	1	0,014
	0,3	2	0,0165
	0,5	0,5	0,013
	0,5	1	0,0245
	0,5	2	0,0375
6	0,1	0,5	0,0005
	0,1	1	0,0005
	0,1	2	0,0035

	0,3	0,5	0,004
	0,3	1	0,0045
	0,3	2	0,004
	0,5	0,5	0,0025
	0,5	1	0,0075
	0,5	2	0,0135

Tabla Apéndice A 8-2 Velocidades conexión Pivote – Centro con k = 3

k=2

Número de Módulos	**Amplitud**	**Frecuencia [s^{-1}]**	**Velocidad [m/s]**
5	0,1	0,5	0,0045
		1	0,009
		2	0,013
	0,3	0,5	0,02
		1	0,0315
		2	0,036
	0,5	0,5	0,048
		1	0,063
		2	0,0651
7	0,1	0,5	0,004765
		1	0,0102
		2	0,01695
	0,3	0,5	0,02965
		1	0,04435
		2	0,0672
	0,5	0,5	0,0744
		1	0,11835
		2	0,06645

Tabla Apéndice A 8-3 Velocidades conexión Pivote – Pivote k = 2

$k = 3$

Número de Módulos	**Amplitud**	**Frecuencia [s^{-1}]**	**Velocidad [m/s]**
7	0,1	0.5	0,00315
		1	0,006225
		2	0,011
	0,3	0.5	0,022
		1	0,0365

		2	0,046
	0,5	0.5	0,0545
		1	0,0756
		2	0,10015
9	0,1	0.5	0,00625
		1	0,0131
		2	0,0164
	0,3	0.5	0,0244
		1	0,0441
		2	0,05425
	0,5	0.5	0,05085
		1	0,0868
		2	N/A

Tabla Apéndice A 8-4 Velocidades conexión Pivote – Pivote k = 3

k=2			
Número de Módulos	**Amplitud**	**Frecuencia [s^{-1}]**	**Velocidad [m/s]**
5	0,1	0,5	0,00507
		1	0,00819
		2	0,0117
	0,3	0,5	0,0255
		1	0,039
		2	0,059
	0,5	0,5	0,067
		1	0,09
		2	0,15
7	0,1	0,5	0,00775
		1	0,01055
		2	0,0189
	0,3	0,5	0,0219
		1	0,04389
		2	0,0651
	0,5	0,5	0,0744
		1	0,14085
		2	0,0566

Tabla Apéndice A 8-5 Velocidades conexión Centro – Centro k = 2

k = 3

Número de Módulos	Amplitud	Frecuencia [s^{-1}]	Velocidad [m/s]
7	0,1	0,5	0,0085
	0,1	1	0,0145
	0,1	2	0,0275
	0,3	0,5	0,0335
	0,3	1	0,0372
	0,3	2	0,042
	0,5	0,5	0,0505
	0,5	1	0,0765
	0,5	2	0,0665
9	0,1	0,5	0,0105
	0,1	1	0,0075
	0,1	2	0,011
	0,3	0,5	0,0175
	0,3	1	0,033
	0,3	2	0,046
	0,5	0,5	0,0305
	0,5	1	0,049
	0,5	2	0,055

Tabla Apéndice A 8-6 Velocidades conexión Centro – Centro k = 3

k = 2

Número de Módulos	Ah	Frecuencia [s^{-1}]	Velocidad [m/s]
12	0,3	0,5	0,03079
		1	0,06168
		2	0,09107
	0,5	0,5	0,05193
		1	0,12868
		2	0,15432
	0,7	0,5	0,10776
		1	0,15625
		2	0,17090

Tabla Apéndice A 8-7 Velocidades configuración tipo Serpiente conexión Pivote – Centro k = 2

k=2

Número de Módulos	Ah	Frecuencia [s^{-1}]	Velocidad [m/s]
12	0,1	0,5	0,04
		1	0,0685

		1,5	0,0985
		2	0,11215
	0,3	0,5	0,10885
		1	0,2061
		1,5	0,362
		2	0,3635
	0,5	0,5	0,225
		1	0,347
		1,5	0,434
		2	0,5205

Tabla Apéndice A 8-8 Velocidades configuración tipo Serpiente conexión Pivote – Pivote k = 2

k=2			
Número de Módulos	**Amplitud**	**Frecuencia [s^{-1}]**	**Velocidad [m/s]**
12	0,3	0,5	0,008
		1	0,006
		2	0,015
	0,5	0,5	0,007
		1	0,023
		2	0,024
	0,7	0,5	0,026
		1	0,036
		2	0,013

Tabla Apéndice A 8-9 Velocidades configuración tipo Serpiente conexión Centro – Centro k = 2

APÉNDICE 2 Tablas recolección pruebas de Implementación

k=2			
Número de Módulos	**Amplitud**	**Frecuencia [s^{-1}]**	**Velocidad [m/s]**
6	0,2	0,2	N/A
		0,3	0,00006
		0,4	0,00027
	0,3	0,2	0,00010
		0,3	0,00021
		0,4	0,00023
	0,5	0,2	0,00060
		0,3	0,00060
		0,4	0,00054
4	0,2	0,2	0,00060
		0,3	0,00250
		0,4	0,00100
	0,3	0,2	0,00348
		0,3	0,00227
		0,4	0,00240
	0,5	0,2	0,00500
		0,3	0,00283
		0,4	0,00550

Tabla Apéndice B 8-1 Velocidades configuración tipo Oruga conexión Pivote – Centro k = 2, semi-módulos completos

k=2			
Número de Módulos	**Amplitud**	**Frecuencia [s^{-1}]**	**Velocidad [m/s]**
5	0,3	0,2	0,0036
		0,3	0,0020
		0,5	0,0011
	0,4	0,2	0,0040
		0,3	0,0048
		0,5	0,0018
	0,5	0,2	0,0042
		0,3	0,0060
		0,5	0,0011
4	0,2	0,2	N/A
		0,3	0,0014
		0,4	0,0013

	0,3	0,2	0,0017
		0,3	0,0059
		0,4	0,0026
	0,5	0,2	0,0031
		0,3	0,0116
		0,4	0,0069

Tabla Apéndice B 8-2 Velocidades configuración tipo Oruga conexión Pivote – Centro k = 2, semi-módulos Reducidos.

k=3			
Número de Módulos	**Amplitud**	**Frecuencia [s^{-1}]**	**Velocidad [m/s]**
8	0,2	0,2	0,00048
		0,3	0,00047
		0,4	N/A
	0,3	0,2	0,00073
		0,3	0,00053
		0,4	N/A
	0,5	0,2	0,00107
		0,3	N/A
		0,4	N/A
6	0,2	0,2	0,00077
		0,3	0,00037
		0,4	0,00016
	0,3	0,2	0,00058
		0,3	0,00077
		0,4	0,00047
	0,5	0,2	0,00117
		0,3	0,00097
		0,4	0,00047

Tabla Apéndice B 8-3 Velocidades configuración tipo Oruga conexión Pivote – Centro k = 3, semi-módulos Completos.

k=3			
Número de Módulos	**Amplitud**	**Frecuencia [s^{-1}]**	**Velocidad [m/s]**
8	0,2	0,2	N/A
		0,3	0,00012
		0,4	0,00015
	0,3	0,2	N/A

		0,3	0,00028
		0,4	0,00044
	0,5	0,2	N/A
		0,3	0,00070
		0,4	0,00100
6	0,2	0,1	0,00048
		0,2	0,00169
		0,3	0,00053
	0,3	0,1	0,00107
		0,2	0,00383
		0,3	0,00040
	0,5	0,1	0,00065
		0,2	0,00580
		0,3	0,00081

Tabla Apéndice B 8-4 Velocidades configuración tipo Oruga conexión Pivote – Centro k = 3, semi-módulos Reducidos.

Módulos	**Amplitud**	**Frecuencia [s^{-1}]**	**Velocidad [m/s]**
4	0,5	0,2	0,00052
		0,3	0,00045
		0,4	0,00030

Tabla Apéndice B 8-5 Velocidades configuración tipo Oruga conexión Pivote – Pivote k = 2, semi-módulos completos

k=2

Módulos	**Frecuencia [s^{-1}]**	**Ah**	**Velocidad [m/s]**
4	0,4	0,3	0,00012
		0,4	0,00022
		0,7	0,00025
	0,7	0,3	0,00033
		0,4	0,00052
		0,7	0,00063
	0,9	0,3	0,00040
		0,4	0,00030
		0,7	0,00053
6	0,4	0,3	0,00033
		0,4	0,00057
		0,6	0,00063

		0,7	0,00094
	0,7	0,3	0,00043
		0,4	0,00054
		0,6	0,00050
		0,7	0,00079
	0,9	0,3	0,00034
		0,4	0,00064
		0,6	0,00077
		0,7	0,00053

Tabla Apéndice B 8-6 Velocidades configuración tipo Serpiente conexión Pivote – Centro k = 2

k=3

Módulos	**Frecuencia [s^{-1}]**	**Ah**	**Velocidad [m/s]**
6	0,4	0,3	0,00030
		0,4	0,00045
		0,6	0,00063
		0,7	0,00042
	0,7	0,3	0,00036
		0,4	0,00024
		0,6	0,00061
		0,7	0,00028
	0,9	0,3	0,00024
		0,4	0,00021
		0,6	0,00041
		0,7	0,00040
8	0,4	0,3	0,00009
		0,4	0,00014
		0,7	0,00015
	0,7	0,3	0,00012
		0,4	0,00009
		0,7	0,00013
	0,9	0,3	0,00004
		0,4	0,00010
		0,7	0,00015

Tabla Apéndice B 8-7 Velocidades configuración tipo Serpiente conexión Pivote – Centro k = 3.

APÉNDICE 3 Código C en Webots Archivo Controlador Configuración Tipo Oruga

```
#include <webots/robot.h>
#include <webots/motor.h>
#include <webots/connector.h>
#include <stdlib.h>
#include <math.h>
#define TIME_STEP 32
static WbDeviceTag motor_rueda_posterior;
static WbDeviceTag motor_rueda_pivote;
static WbDeviceTag motor_pivote;
static double t=0.0;
static int id =-1;
static const double A=0.5;
static const double F=2;
static int k=2;
static int state = 0;
static int M=7;
static double d= 1/7;
static float deadline;
static void worm_modules_izq() {
   double shift=((2*M_PI*k/M)*((id-1)+d));
   double phase =2.0*M_PI*F*t;
   wb_motor_set_position(motor_pivote, -A*cos(phase-shift)); }
static void worm_modules_der() {
   double shift=((2*M_PI*k/M)*((id-1)+d));
   double phase =2.0*M_PI*F*t;
   wb_motor_set_position(motor_pivote, A*sin(phase+shift));  }
static void reposo() {
  wb_motor_set_position(motor_pivote,0); }
static void do_nothing() {
```

```
}
struct {
  void (*func) ();
  float duration; }
states[] = {
  {worm_modules_izq,50},
  {do_nothing},
  {worm_modules_der,50},
  {do_nothing},
  {reposo,1},
  { NULL, 1 } };
int main() {
  wb_robot_init();
  const char *name = wb_robot_get_name();
  id=atoi(name+8);
  motor_rueda_posterior=wb_robot_get_device("motor_rueda_posterior");
  motor_rueda_pivote=wb_robot_get_device("motor_rueda_pivote");
  motor_pivote=wb_robot_get_device("motor_pivote");
  wb_motor_set_velocity(motor_rueda_posterior,0);
  wb_motor_set_velocity(motor_rueda_pivote,0);
  wb_motor_set_velocity(motor_pivote,3);
  deadline = states[0].duration;
  while (wb_robot_step(TIME_STEP) != -1) {
       if (t >= deadline) {
      state++;
      deadline += states[state].duration;}
      (*states[state].func) ();
      t += TIME_STEP / 1000.0;  };
  wb_robot_cleanup();
    return 0; }
```

APÉNDICE 4 Código C Webots Archivo Controlador Configuración Tipo Serpiente

```
#include <webots/robot.h>
#include <webots/motor.h>
#include <webots/connector.h>
#include <stdlib.h>
#include <math.h>
#define TIME_STEP 32
static WbDeviceTag motor_rueda_posterior; //Dar nombre a los nodos que componen el módulo.
static WbDeviceTag motor_rueda_pivote;
static WbDeviceTag motor_pivote;
static double t=0.0; //inicializar varible de tiempo.
static int id =-1; // Variable de identificción del módulo.
static const double Av=0.3;//Amplitud vertical tender a cero .
static const double Ah=0.5;//Amplitud horizontal.
static const double F=1;// Frecuencia.
static const double phvh=0;// Fase entre vertical y horizontal.
static int k=2;//Número de ondulaciones
static int state = 0;
static int M=12;// Número de módulos según configuración.
static double d= 1/12; // Distancia.
static float deadline;
static void worm_modules_der() {
  if (id%2!=0){// Determinar si el número de modulos es par=vertical o impar=horizontal
   double shift=((4*M_PI*k/M)*((id-1)+d));//Ecuación vertical
   double phase =2.0*M_PI*F*t;
   wb_motor_set_position(motor_pivote, Av*sin(phase+shift));  //Ecuación en discreto
  }
else {//Movimiento horizontal.
   double shift=((4*M_PI*k/M)*(id+d-0.5));
```

```
    double phase =2.0*M_PI*F*t;
    double phvh = 1.5708;//90°
    wb_motor_set_position(motor_pivote, -Ah*sin(phase+shift+phvh));
    }
}
static void worm_modules_izq() {
  if (id%2!=0){
    double shift=((4*M_PI*k/M)*((id-1)+d));
    double phase =2.0*M_PI*F*t;
    wb_motor_set_position(motor_pivote, Av*sin(phase+shift));  //
   }
else {
    double shift=((4*M_PI*k/M)*(id+d-0.5));
    double phase =2.0*M_PI*F*t;
    double phvh = 1.5708;
    wb_motor_set_position(motor_pivote, Ah*sin(phase+shift+phvh));    }
 }
static void reposo() {
  wb_motor_set_position(motor_pivote,0); }
static void do_nothing(){
}
struct {
  void (*func) ();
  float duration;
}
states[] =
{
  {worm_modules_izq,50},
  {do_nothing},
  {worm_modules_der,50},
  {do_nothing},
```

```
  {reposo,1},
  { NULL, 1 }
};
int main()
{
  wb_robot_init(); // Función inicio robot.
  const char *name = wb_robot_get_name();// Obtener nombre del robot
  id=atoi(name+8);// Identificación del módulo.
  motor_rueda_posterior=wb_robot_get_device("motor_rueda_posterior");//  Asignación
nombre
  motor_rueda_pivote=wb_robot_get_device("motor_rueda_pivote");
  motor_pivote=wb_robot_get_device("motor_pivote");
  wb_motor_set_velocity(motor_rueda_posterior,0);//Definir velocidad
  wb_motor_set_velocity(motor_rueda_pivote,0);
  wb_motor_set_velocity(motor_pivote,3);
  deadline = states[0].duration;
  while (wb_robot_step(TIME_STEP) != -1) {
    if (t >= deadline) {
     state++;
     deadline += states[state].duration;}
   (*states[state].func) ();
   /* computed elapsed time */
   t += TIME_STEP / 1000.0;
  };
  wb_robot_cleanup();
  return 0;
}
```

APÉNDICE 5 Código MATLAB Archivo Controlador

```
global puerto;
global pserial;
global t; t=0;
global borrar;
borrar=0;
strk=get(handles.popk,'String'); valk=get(handles.popk,'Value');
k=str2double(strk(valk));
strm=get(handles.popm,'String'); valm=get(handles.popm,'Value');
val=get(handles.popconfig,'Value');
delete(instrfind({'Port'},{puerto}));
pserial=serial(puerto);
set(pserial,'Baudrate',115200);
fopen(pserial);
if iscell(strm)
   M=strm{valm};
else
   M=strm(valm,:);
end
M=str2num(M)
d=1/M;
Asv=0.2;
phvh=0;
graf=get(handles.popupmenu7,'Value')
if val==1
  Fo=str2double(get(handles.text14,'String'))
  Ao=str2double(get(handles.text11,'String'))
  while(t<=50 && borrar==0)
      temp = timer('TimerFcn', 'stat=false');
      start(temp)
      stat=true;
```

```
while(stat==true)
   for id =1:1:M
      t=t+0.005;
      phase=2*pi*Fo*t*100;
      shift=((2*pi*(k/M))*((id-1)+d));
      mot_pos=rad2deg(-(Ao*(cos(phase-shift))));
      mot_pos=round(mot_pos)+90;
       if id==graf
          g=t*10;
         plot(g,mot_pos,'*')
       end
     dato=int2str(mot_pos);
      idm=int2str(id);
      if id<10
         idm=strcat({'0'},idm);
      end
      if mot_pos<10
         datom1=strcat(idm,{'1'},{'00'},dato);
      elseif mot_pos>=10 && mot_pos<100
         datom1=strcat(idm,{'1'},{'0'},dato);
      elseif mot_pos>=100
         datom1=strcat(idm,{'1'},dato);
      end
      hold on
      pause(0.0001)
      datom1=char(datom1);
      fwrite(pserial,datom1,'uint8');
   end
   pause(0.100)
end
id=0;
```

```
    end
elseif val==2
    Fs=str2double(get(handles.text16,'String'));
    Ash=str2double(get(handles.text15,'String'));
    while(t<=50 && borrar==0)
        temp = timer('TimerFcn', 'stat=false');
        start(temp)
        stat=true;
        while(stat==true)
            for id =1:1:M
                t=t+0.005;
                idm=int2str(id);
                if id<10
                    idm=strcat({'0'},idm);
                end
                if mod(id,2)~=0
                    shift=((4*pi*k/M)*((id-1)+d)); phase =2.0*pi*Fs*t*100;
                    mot_posv=rad2deg(Asv*sin(phase+shift));
                    mot_posv=round(mot_posv)+90;
                    dats=int2str(mot_posv);
                    if mot_posv<10
                        datsm1=strcat(idm,{'1'},{'00'},dats)
                    elseif mot_posv>=10 && mot_posv<100
                        datsm1=strcat(idm,{'1'},{'0'},dats)
                    elseif mot_posv>=100
                        datsm1=strcat(idm,{'1'},dats)
                    end
         if id==graf
                        plot(t,mot_posv,'*')
                         hold on
                         set(handles.text18,'String','Módulo Vertical');
```

```
            end
        elseif mod(id,2)==0
            shift=((4*pi*k/M)*(id+d-0.5));
            phase =2.0*pi*Fs*t;
            phvh = 1.5708;
            mot_posh=rad2deg(Ash*sin(phase+shift+phvh));
            mot_posh=round(mot_posh)+90;
            dats=int2str(mot_posh);
            if mot_posh<10
               datsm1=strcat(idm,{'1'},{'00'},dats)
            elseif mot_posh>=10 && mot_posh<100
               datsm1=strcat(idm,{'1'},{'0'},dats)
            elseif mot_posh>=100
               datsm1=strcat(idm,{'1'},dats)
            end
             if id==graf
                plot(t,mot_posh,'+')
                hold on
                set(handles.text18,'String','Módulo Horizontal');
             end
        end
        hold on
        pause(0.0001)
        datsm1=char(datsm1);
        fwrite(pserial,datsm1,'uint8');
      end
          pause(0.100)
   end
   id=0;
 end
end
```

APÉNDICE 6 Código TeensyDuino Archivo Controlador

```
#include <PololuMaestro.h>
MiniMaestro maestro(Serial2);
int dato=90;
float datof=0;
float datof2=0;
int valor0=0;int valor1=0;
int valor2=0;int valor3=0;
int valor4=0;int valor5=0;
int valordos=0;
int Byte_entrada0=0;
int Byte_entrada1=0;
int Byte_entrada2=0;
int Byte_entrada3=0;
int Byte_entrada4=0;
int Byte_entrada5=0;
int contador=0;
int id=8;
void setup (){
Serial1.begin(115200);
Serial2.begin(115200);
}
void loop(){
  for(int k=0;k<6;k++){
  if(Serial1.available()>0  && contador==0){Byte_entrada0=Serial1.read();contador=1;}
  if(Serial1.available()>0  && contador==1){Byte_entrada1=Serial1.read();contador=2;}
  if(Serial1.available()>0  && contador==2){Byte_entrada2=Serial1.read();contador=3;}
  if(Serial1.available()>0  && contador==3){Byte_entrada3=Serial1.read();contador=4;}
  if(Serial1.available()>0  && contador==4){Byte_entrada4=Serial1.read();contador=5;}
  if(Serial1.available()>0  && contador==5){Byte_entrada5=Serial1.read();contador=0;}
```

```
 }
 if(Byte_entrada0=='0'){valor0=0;}if(Byte_entrada0=='1'){valor0=1;}
if(Byte_entrada0=='2'){valor0=2;}if(Byte_entrada0=='3'){valor0=3;}
if(Byte_entrada0=='4'){valor0=4;}if(Byte_entrada0=='5'){valor0=5;}
if(Byte_entrada0=='6'){valor0=6;}if(Byte_entrada0=='7'){valor0=7;}
if(Byte_entrada0=='8'){valor0=8;}if(Byte_entrada0=='9'){valor0=9;}
 if(Byte_entrada1=='0'){valor1=0;}if(Byte_entrada1=='1'){valor1=1;}
if(Byte_entrada1=='2'){valor1=2;}if(Byte_entrada1=='3'){valor1=3;}
if(Byte_entrada1=='4'){valor1=4;}if(Byte_entrada1=='5'){valor1=5;}
if(Byte_entrada1=='6'){valor1=6;}if(Byte_entrada1=='7'){valor1=7;}
if(Byte_entrada1=='8'){valor1=8;}if(Byte_entrada1=='9'){valor1=9;}
 if(Byte_entrada2=='0'){valor2=0;}if(Byte_entrada2=='1'){valor2=1;}
if(Byte_entrada2=='2'){valor2=2;}if(Byte_entrada2=='3'){valor2=3;}
if(Byte_entrada2=='4'){valor2=4;}if(Byte_entrada2=='5'){valor2=5;}
if(Byte_entrada2=='6'){valor2=6;}if(Byte_entrada2=='7'){valor2=7;}
if(Byte_entrada2=='8'){valor2=8;}if(Byte_entrada2=='9'){valor2=9;}
if(Byte_entrada3=='0'){valor3=0;}if(Byte_entrada3=='1'){valor3=1;}
if(Byte_entrada3=='2'){valor3=2;}if(Byte_entrada3=='3'){valor3=3;}
if(Byte_entrada3=='4'){valor3=4;}if(Byte_entrada3=='5'){valor3=5;}
if(Byte_entrada3=='6'){valor3=6;}if(Byte_entrada3=='7'){valor3=7;}
if(Byte_entrada3=='8'){valor3=8;}if(Byte_entrada3=='9'){valor3=9;}
if(Byte_entrada4=='0'){valor4=0;}if(Byte_entrada4=='1'){valor4=1;}
if(Byte_entrada4=='2'){valor4=2;}if(Byte_entrada4=='3'){valor4=3;}
if(Byte_entrada4=='4'){valor4=4;}if(Byte_entrada4=='5'){valor4=5;}
if(Byte_entrada4=='6'){valor4=6;}if(Byte_entrada4=='7'){valor4=7;}
if(Byte_entrada4=='8'){valor4=8;}if(Byte_entrada4=='9'){valor4=9;}
if(Byte_entrada5=='0'){valor5=0;}if(Byte_entrada5=='1'){valor5=1;}
if(Byte_entrada5=='2'){valor5=2;}if(Byte_entrada5=='3'){valor5=3;}
if(Byte_entrada5=='4'){valor5=4;}if(Byte_entrada5=='5'){valor5=5;}
if(Byte_entrada5=='6'){valor5=6;}if(Byte_entrada5=='7'){valor5=7;}
if(Byte_entrada5=='8'){valor5=8;}if(Byte_entrada5=='9'){valor5=9;}
```

```
  dato=(valor3*100)+(valor4*10)+(valor5);
  valordos=(valor0*10)+valor1;
    if(valordos==id){
 if(valor2==1){
 datof2=0.0081*pow(dato,2)-12.569*dato+2572.2;
 datof=0.0077*pow(dato,2)+9.4265*dato+551.73;
 datof=datof*4;
 datof2=datof2*4;
 maestro.setTarget(0,datof);
 delay(1);
 maestro.setTarget(1,datof2);}
 if(valor2==2){
  datof=10.074*dato+569;
  datof=datof*4;
  maestro.setTarget(2,datof);}
 if(valor2==3){
  datof=10.074*dato+569;
  datof=datof*4;
  maestro.setTarget(3,datof);}
 if(valor2==4){
  datof=10.074*dato+569;
  datof=datof*4;
  maestro.setTarget(4,datof);}
 if(valor2==5){
  datof=10.074*dato+569;
  datof=datof*4;
  maestro.setTarget(5,datof);}
  }
else{}}
```